



AMS Manual

Amsterdam Modeling Suite 2025.1

www.scm.com

Apr 04, 2025

CONTENTS

1	General	1
1.1	Overview	1
1.2	What's new in the AMS driver?	1
1.2.1	New in AMS2025.1	1
1.2.2	New in AMS2024.1	1
1.2.3	Removed in AMS2024.1	2
1.2.4	New in AMS2023.1	2
1.2.5	New in AMS2022.1	2
1.2.6	New in AMS2021.1	3
1.2.7	New in AMS2020.1	3
1.2.8	New in AMS2019.3	4
1.2.9	New in AMS2019.1	4
2	Input, execution and output	7
2.1	Input	7
2.1.1	Example	7
2.1.2	Tasks	8
2.1.3	Properties	9
2.1.4	General remarks on input structure and parsing	9
2.1.5	Keys	10
2.1.6	Blocks	11
2.1.7	Including an external file	12
2.1.8	Units	13
2.2	Execution	13
2.2.1	Shell script	13
2.2.2	Python interface	14
2.2.3	Interactive input file	14
2.2.4	Running AMS on compute clusters	15
2.2.5	AMS environment variables	17
2.3	Output	18
2.3.1	Results directory	18
2.3.2	Logfile ams.log	18
2.3.3	Binary output files	19
2.3.4	Standard output	20
2.3.5	Optimized geometry in xyz format	20
2.4	Driver level parallelism	20
2.5	Pipe interface	24
2.5.1	AMSPipe protocol specification	24
	Low-level message encoding	25
	Return messages and error handling	25

	Methods	26
2.5.2	AMS as a pipe master	28
2.5.3	AMS as a pipe worker	29
3	Geometry, System definition	31
3.1	Geometry, Lattice	31
3.1.1	Modifying the geometry	34
3.2	Symmetry	35
3.3	Regions	37
3.4	Charge, atomic masses, input bond orders	38
3.5	Homogeneous electric field and multipole charges	40
3.6	Load a System from file	41
3.7	Atom attributes	42
3.8	Force field related extensions	43
3.8.1	Load charges for a forcefield into regions	43
3.8.2	Load forcefield atom types	45
4	Structure and Reactivity, Molecular Dynamics	47
4.1	Single point calculations	47
4.2	Bond energy calculations	47
4.2.1	Ground state energy	48
4.2.2	Formation energy	48
4.2.3	Atomization energies	48
4.2.4	Chemisorption energies	48
4.2.5	Atomic corrections	49
4.2.6	Open shell systems	49
4.2.7	Impurities	49
4.3	Geometry optimization	49
4.3.1	Convergence criteria	50
4.3.2	Automatic restarts	54
4.3.3	Lattice optimization	55
4.3.4	Keep all results files	55
4.3.5	Constrained optimization	55
	Fixed atoms	56
	Internal degrees of freedom	57
	Block constraints	58
	Lattice constraints	59
	Restraints	59
4.3.6	Optimization under pressure / external stress	59
4.3.7	Optimization methods	59
	Quasi-Newton	60
	FIRE	63
	Limited-memory BFGS	66
	Conjugate gradients	68
	Dimer method for TS seach	68
4.3.8	Troubleshooting	70
	Failure to converge	70
	Converged, but did not find a minimum on the PES	71
	Restarting a geometry optimization	71
4.4	Transition state search	72
4.5	Linear Transit, PES scan	76
4.5.1	Distance, angle, and dihedral angle coordinates	77
4.5.2	Joint scan coordinates	77
4.5.3	Multidimensional PES scan	77

4.5.4	Lattice scan coordinates for periodic systems	78
	Isotropic scaling of the unit cell volume or area	78
	Scaling of the lattice vector lengths	78
	Strain matrix in Voigt notation	79
	Scan arbitrary lattices	79
4.5.5	Calculate properties for all PES points	79
4.5.6	Troubleshooting	80
4.5.7	Output	82
4.6	Nudged Elastic Band (NEB)	83
4.6.1	Input	84
4.6.2	Frozen atom constraints	92
4.6.3	Optimizations and convergence criteria	93
4.6.4	Output	93
4.6.5	Troubleshooting	93
4.7	Intrinsic Reaction Coordinate (IRC)	93
4.7.1	Method details	94
4.7.2	Input	95
4.7.3	Output	100
4.8	Excited state optimizations	101
4.9	Molecular dynamics	101
4.9.1	General	109
4.9.2	Constrained molecular dynamics	111
4.9.3	(Re-)Starting a simulation	113
4.9.4	Thermostats and barostats	116
	Temperature and pressure regimes	121
4.9.5	Trajectory sampling and output	121
4.9.6	Lattice deformations (volume regimes)	130
4.9.7	Molecule Gun: adding molecules during simulation	134
4.9.8	Removing molecules during simulation	140
4.9.9	Accelerated dynamics	144
	The PLUMED library support in AMS	144
	Metadynamics for Conformer-Rotamer Ensemble Sampling (CREST-MTD)	145
	Collective Variable-driven HyperDynamics (CVHD)	147
	Temperature Replica Exchange	154
	Bond Boost Method	156
	Reaction boost (Targeted MD)	159
	Force bias Monte Carlo (fbMC)	174
	Nanoreactor	177
4.9.10	Non-equilibrium MD (NEMD)	179
	T-NEMD for thermoconductivity: heat exchange	179
	NEMD for tribology: Applying velocities / forces	184
	NEMD viscosity via cosine shear perturbation	186
4.9.11	Applying reflective walls	188
4.9.12	Exit Conditions	189
4.10	Trajectory replay	191
4.11	Grand Canonical Monte Carlo (GCMC)	193
4.11.1	General info	193
4.11.2	Method Details	194
4.11.3	Input	195
4.11.4	Output	202
4.12	Automated PES Exploration	203
4.12.1	Overview	204
4.12.2	Job selection and main options	205
4.12.3	Results: the “Energy Landscape”	208

Results on the text output	208
4.12.4 Continue a PES exploration from a previous calculation	209
4.12.5 Troubleshooting	211
4.12.6 Structure comparison	211
4.12.7 Process Search job	213
Overview	213
Input options	213
4.12.8 Saddle Search job	214
Overview	214
Input options	215
4.12.9 Basin Hopping job	219
Overview	219
Input options	219
4.12.10 Fragmented states	221
4.12.11 Landscape Refinement	224
4.12.12 Creating and/or Extending an Energy Landscape from the Input File	225
4.12.13 Optimizer	228
4.12.14 Binding Sites	229
Input options	232
4.12.15 References	235
5 Gradients, Hessian, Stress tensor, Elasticity	237
5.1 Nuclear gradients	237
5.2 Hessian	238
5.3 PES point character	239
5.4 Thermodynamics, gas phase Gibbs free energy	241
5.5 Stress tensor	242
5.6 Elastic tensor	242
5.7 Numerical differentiation options	245
6 Vibrational Spectroscopy	247
6.1 General	247
6.1.1 Where are the results?	247
6.2 IR frequencies and normal modes	248
6.2.1 All vibrational Modes	248
Rescanning Imaginary modes	249
6.2.2 Symmetric Displacements	250
6.2.3 Mobile Block Hessian (MBH)	251
6.2.4 Mode Scanning	254
Theory	254
Input	255
6.2.5 Mode Refinement	255
Theory	256
Input	256
6.2.6 Mode Tracking	257
Theory	258
Input	259
Input: Tracking methods	262
Input: Selecting modes	264
Input: Convergence	265
6.2.7 Selecting modes	266
6.2.8 Thermodynamics (ideal gas)	269
Gibbs free energy change for a gas phase reaction	272
6.2.9 Moments of inertia	273

6.2.10	Partial Vibrational Spectra (PVDOS)	273
6.3	Phonons	274
6.4	(Resonance) Raman	278
6.4.1	Raman	278
6.4.2	Resonance Raman: excited-state finite lifetime	279
6.4.3	Resonance Raman: VG-FC	281
	Theory	281
	Input	282
6.5	VROA: (Resonance) vibrational Raman optical activity	283
6.5.1	Engine ADF	285
6.6	VCD: Vibrational Circular Dichroism	285
6.6.1	Atomic polar tensor (APT) model	285
6.6.2	Analytical VCD in ADF	286
6.7	Vibrational Polarizabilities	286
7	Vibrationally resolved electronic spectra	287
7.1	AH-FC: Adiabatic Hessian Franck-Condon	287
7.1.1	FCF module: Franck-Condon Factors	287
	Theory	288
	Algorithm	289
	Input	289
	Result: TAPE61	293
7.1.2	FCF example absorption and fluorescence	294
7.1.3	FCF Example phosphorescence	295
7.1.4	Density of States computed with AH-FC method	296
7.2	VG-FC: Vertical Gradient Franck-Condon	297
7.2.1	Theory	298
	Theory: Vibronic-Structure Tracking	299
	Theory: Vibronic-Structure Refinement	300
	Theory: Adiabatic excitation energy	300
7.2.2	Input: Vibronic-Structure all modes	301
7.2.3	Input: Vibronic-Structure Tracking	302
	Input: Restarting VST	302
7.2.4	Input: Vibronic-Structure Refinement	303
7.2.5	Input: Excited State	304
7.2.6	Input: Producing the spectrum	306
8	Dipole moment, Polarizability, Bond orders, Orbital Energies	309
8.1	Charges, Dipole Moment, Polarizability	309
8.2	Bond orders & Molecule detection	310
8.3	Bond guessing algorithm	312
8.4	Orbital energies, HOMO-LUMO gap	313
8.5	Uncertainties	315
9	Engines	317
9.1	Available engines	317
9.2	Summary of engine capabilities	318
9.3	External programs as engines	321
9.4	Toy engines	324
9.5	Load an Engine configuration from file	325
9.6	Engine add-ons	325
9.6.1	Dispersion corrections	326
9.6.2	Pressure	329
9.6.3	Non-isotropic external stress	330

9.6.4	Atom energies	332
9.6.5	Restraints	332
9.6.6	Wall potential	335
9.6.7	Short-range repulsive potential	337
10	Utilities	339
10.1	AMSbatch	339
10.1.1	Overview	339
10.1.2	Loading systems from file	340
10.1.3	Restart	342
10.2	AtomTyping	342
10.2.1	Command Line Tool	342
10.2.2	Python scripting	343
10.3	Conformers	344
10.3.1	New in Conformers-2025	344
	Overview	344
	Main Input Options	345
	Examples	351
	References	353
	Other input options	353
10.4	Trajectory Analysis	415
10.4.1	New in Trajectory Analysis-2025	415
10.4.2	New in Trajectory Analysis-2023	415
10.4.3	Quickstart Guide	415
10.4.4	Trajectory selection	416
10.4.5	Atom Selection	417
10.4.6	Convergence Check	418
10.4.7	Radial Distribution Function (RDF)	418
	Description	418
	Atom Seleccion	419
	Options	421
10.4.8	Histogram	422
10.4.9	Autocorrelation Functions	427
	Description	427
	Properties	428
	Options	430
	Keywords	432
10.4.10	Mean Square Displacement	436
	Description	436
	Properties	437
	Options	439
	Keyword	442
10.4.11	AverageBinPlot	446
10.4.12	Viscosity	449
10.4.13	Diffusion Coefficient	450
	From Velocity Autocorrelation	450
	From Mean Square Displacement	451
10.4.14	Ionic Conductivity	452
10.5	VCD Analysis: VCDtools	454
10.5.1	General Theory	454
10.5.2	General Coupled Oscillator Analysis	455
10.5.3	Available options	456
11	Examples	457

11.1	Geometry optimization	457
11.1.1	Example: Simple geometry optimization	457
11.1.2	Example: Two-stage geometry optimization with initial Hessian	457
11.1.3	Example: Periodic lattice optimization under pressure	459
11.1.4	Example: Phase Transition Due To External Nonuniform Stress	460
11.1.5	Example: Boron nitride optimization under external stress	462
11.1.6	Example: Graphene optimization under external stress	464
11.1.7	Example: Constrained optimizations	465
11.2	Transition state search	480
11.2.1	Example: TS search starting from initial Hessian	480
11.2.2	Example: PES scan and TS search for H2 on graphene	482
11.3	Nudged Elastic Band (NEB)	485
11.3.1	Nudged Elastic Band (NEB) Examples	485
	HCN isomerization reaction with NEB	486
	H2 dissociation on graphene	486
	Running multiple NEB calculations using PLAMS	486
11.4	Intrinsic reaction coordinate (IRC)	486
11.4.1	Example: IRC for HCN	486
11.4.2	Example: TS and IRC for Claisen reaction	488
11.5	PES scan	491
11.5.1	Example: Linear transit	491
11.5.2	Example: 2D PES scan	495
11.5.3	Example: PES scan for lattice degrees of freedom	498
11.6	PES Exploration	503
11.6.1	Example: Basin Hopping for Ar 13 cluster	503
11.6.2	Example: PES Exploration, Process Search for alanine with PLAMS	505
11.6.3	Example: PES Exploration, Binding Sites for O on Pt 111	505
11.6.4	Example: Importing conformers into PES explorations	508
11.7	Molecular dynamics	511
11.7.1	Example: Simple MD for H2	511
11.7.2	Example: MD for a box of water	511
11.7.3	Example: Lattice deformations in MD	512
11.7.4	Example: PLUMED spherical constraint on Ion-Water cluster	517
11.7.5	Example: MM/MM with constrained ion-water cluster in periodic water box	519
11.7.6	Example: Measuring the friction coefficient using NEMD	531
11.7.7	Example: Measuring the viscosity using NEMD	535
11.8	Trajectory replay	539
11.8.1	Example: Trajectory replays for PES scans, NEB and MD calculations	539
11.9	Vibrational analysis	544
11.9.1	Example: Mode Refinement	544
11.9.2	Example: Mode Tracking	547
11.9.3	Example: Vibronic-Structure Tracking	550
11.10	PES point properties	553
11.10.1	Example: Phonons for graphene	553
11.10.2	Example: Phonons with isotopes	554
11.10.3	Example: User-defined Brillouin zone for phonon dispersion	556
11.10.4	Example: Elastic tensor	557
11.11	Pipe interface	560
11.11.1	Example: ASE calculator as a pipe worker	560
11.11.2	Example: AMS as a pipe worker	562
12	Appendices	565
12.1	Extended XYZ file format	565
12.2	Developer options	565

12.3	Symmetry	568
12.3.1	Schönflies symbols and symmetry labels	568
12.3.2	Molecular orientation requirements	570
	Molecular orientation conventions	570
13	Required citations	571
13.1	General references	571
13.2	Feature references	571
13.2.1	Frequencies, IR Intensities, Raman, VCD	571
13.2.2	PES Exploration	573
14	External programs and Libraries	575
15	Keywords	577
15.1	Links to manual entries	577
15.2	Summary of all keywords	578
15.2.1	ams	578
15.2.2	analysis	762
15.2.3	chemtrayzer2	785
15.2.4	conformers	790
15.2.5	fcf	873
15.2.6	oled-deposition	873
15.2.7	oled-properties	885
16	KF output files	909
16.1	Accessing KF files	909
16.2	Sections and variables on ams.rkf	910
17	FAQ	1019
17.1	What is the difference between AMS and the AMS driver?	1019
17.2	What is the AMS driver?	1019
17.3	What will AMS do when one point fails to converge?	1019
	Index	1021

1.1 Overview

The AMS driver handles all changes in the simulated system's geometry, e.g. during a *geometry optimization* (page 49) or *molecular dynamics* (page 101) calculation, using the so-called *engines* (page 315) like ADF, BAND or DFTB for the calculation of energies and forces.

The AMS driver steers the engines across the potential energy surface.

1.2 What's new in the AMS driver?

1.2.1 New in AMS2025.1

- *Dispersion add-on* (page 326) use updated libraries for DFT-D3 (s-dftd3 1.2.1) and DFT-D4 (dftd4 3.7.0)
- Writing MDStep engine results files during MD can now be customized or completely disabled using *EngineResultsFreq* (page 121).
- When remapping atomic positions across periodic boundaries, the MD driver now uses intuitive and predictable coordinate ranges. This can be customized using the *Remap Range* (page 121) setting and pre-AMS2025 behaviour can be restored using `Range=AroundCenter`.
- MD *Deformations* (page 130) can now automatically scale a box to a given `TargetDensity`.

1.2.2 New in AMS2024.1

- Use *image dependent pair potential (IDPP)* (page 84) to generate an initial path for NEB calculations
- *Reaction boost (targeted MD) restraints* (page 159) for molecular dynamics
- Dipole moment added as an option to the *BinLog* (page 121) output section for calculation of IR spectra from ab initio MD
- The *dimer method* (page 68) for transition state search
- A new *Reactions Discovery* tool that can be used with any *AMS engine* (page 315).
- Use engine *uncertainties* (page 315).
- Stop MD trajectories early with *exit conditions* (page 189) based on too short atom-atom distances or too high engine uncertainties.
- Store as part of the MD trajectory: temperature of individual thermostats, total BondBoost or Reactor restraint energy.

1.2.3 Removed in AMS2024.1

- Old FCF program (\$AMSBIN/oldfcf) is no longer supported
 - the new *FCF program* (page 287) (\$AMSBIN/fcf) has replaced the old FCF program in AMS2022.1
 - see documentation in the AMS2021.1 manual on the old FCF program (https://www.scm.com/doc.2021/AMS/Utilities/FCF_module.html)

1.2.4 New in AMS2023.1

- A new *Conformers tool* (page 344) tool allows conformer generation with any *AMS engine* (page 315).
- *Molecular orbitals energies and occupations* (page 313) can now be requested from the `Properties` block in the driver input and are printed in an engine independent manner.
- The *AMSbatch* (page 339) utility program allows executing the same AMS job for multiple systems.
- A Martinez style *Nanoreactor* (page 177) was added to the MD driver.
- The *PES scan* (page 76) task can now use all *geometry optimizers* (page 59). Previously it was limited to using the Quasi-Newton optimizer. This will make pure *lattice PES scans* (page 78) (e.g. energy-volume scans) with fast *engines* (page 315) and large cells much faster.
- The `GeometryOptimization%Convergence%Quality` keyword can now be used to *easily tighten or relax all convergence thresholds* (page 52) for geometry optimizations at once.
- The calculation of normal modes now also calculates and prints the reduced masses and force constants of the individual modes.
- The *PES point characterization* (page 239) now uses the force constants instead of the vibrational frequencies to identify transition states, making it more reliable in the presence of numerical noise.
- Environment variables in the input will now be *expanded* (page 10).
- The default value of `VibrationalAnalysis%Displacement` has been changed to 0.005 Bohr, related to precision issues with derivatives of polarizability.
- The calculation of the *vibronic density of states* (page 296) for transfer phenomena using the Adiabatic-Hessian Franck-Condon formalism has been added.
- Calculation of *vibrational polarizabilities* (page 286) under the harmonic approximation.
- For the *PES exploration* (page 202) task, it is now possible to *read an energy landscape and/or add states from the input file* (page 225). The algorithm for detecting and labeling *binding sites* (page 229) was enhanced. Importantly, the *Landscape Refinement* (page 224) job can now be run in parallel.

1.2.5 New in AMS2022.1

- The MD driver now supports Rattle/Shake methods for *constrained molecular dynamics simulations* (page 111). The *FIRE optimizer* (page 63) now supports *distance constraints* (page 55).
- The *PES scan* (page 76) task now allows the *scanning of lattice degrees of freedom* (page 78).
- The *molecule gun* (page 134) now supports the *generation of molecule mixtures* (page 139) through the `Mole-Fraction` keyword.
- The *Replay task* (page 191) for recomputing properties for trajectory frames with a different engine.
- The *PES exploration* (page 202) can now include *fragmented states* (page 221) in the energy landscape when studying adsorption processes.

- A new option allows *Geometry optimizations* (page 49) with *PES point characterization* (page 239) and no *symmetry* (page 35) to converge more reliably to minima instead of transition states or higher order saddle points: in case a geometry optimization converges to a TS, it can will now *automatically restart* (page 54) after displacing all atoms along the imaginary mode.
- The calculation of Franck-Condon factors with the *FCF program* (page 287) has been improved regarding precision, speed, and memory requirements.

1.2.6 New in AMS2021.1

- *PES exploration* (page 202) tasks for the automated discovery of stationary points on the potential energy surface.
 1. *Process Search* (page 213): a composite method for finding escape mechanisms from a state. This will find both local minima and their connecting saddle points.
 2. *Basin Hopping* (page 219): a Monte Carlo method for finding local minima.
 3. *Saddle Search* (page 214): a single-ended method for finding nearby saddle points.
- *Force bias Monte Carlo* (page 174) is now available in AMS, enabling pure MC or mixed MD/MC simulations with any engine.
- The *molecule gun* (page 134) can now generate random shooting directions in a given cone.
- A new *wall potential* (page 335) engine add-on can be used to simulate spherical nanoreactors.
- The *D4 dispersion add-on* (page 326) can now also be used for calculations on periodic systems.
- Graceful interactive termination of running jobs through the *interactive input file* (page 14).
- *Geometry optimizations* (page 49) write the optimized geometry to the results directory as an *extended XYZ file* (page 565).

1.2.7 New in AMS2020.1

- ADF has been fully integrated as an AMS engine.
- New engines: *MLPotential* implementing several different type of machine learning (ML) potentials. A *Hybrid engine* for combining other engines in QM/MM calculations. Enhanced *ForceField* engine (formerly UFF).
- Improvements to the *Quasi-Newton geometry optimizer* (page 60): Periodic systems can now be optimized in delocalized coordinates. Better performance for systems made up of disconnected fragments.
- Geometry optimization can now be performed with *frozen and equal strain constraints* (page 58) for the lattice degrees of freedom. This option is currently only available with the *FIRE optimizer* (page 63).
- Support for *external electric fields* (page 39): Homogeneous as well as multipole charges.
- Enhanced support for *vibrational spectroscopy* (page 246): (resonance) Raman, (resonance) vibrational Raman optical activity, vibrational circular dichroism.
- The MD driver now supports time-dependent *lattice deformations* (page 130) as well as a method for accelerating bonding reactions (*Bond Boost method* (page 156)).
- Reactive MD calculations with the AMS driver can now be analyzed with ChemTraYzer.
- It is now possible to selectively disable writing some parts of a *MD trajectory* (page 121) to save space.
- More *trajectory analysis* (page 415) options: *autocorrelation functions* (page 426) and *diffusion coefficient* (page 449)
- The *molecular composition analysis* (page 310) can now be done with respect to an adsorption support region.

- The *transition state search* (page 72) task now allows the users to specify an approximate reaction coordinate (TSRC).
- For the calculation of normal modes, the *Mobile Block Hessian* (page 251) method now allows to treat parts of the system as rigid blocks.
- New coordinates for *constrained geometry optimizations* (page 55) and *PES scans* (page 76): Sum of distances and difference of distances.
- Added the ability to include additional potential terms (i.e. springs) through the *Restrains engine add-on* (page 332).
- A running AMS driver process can be used from Python through the new *AMSWorker class* in the *PLAMS* library. The communication between PLAMS and the AMS driver happens via the new *Pipe interface* (page 24) protocol.
- Introduced *Regions* (page 37) to simplify the input syntax for options that apply to a subset of atoms only.
- Symmetrization of systems from the input and the option to input a Z-matrix.
- Low frequencies contribution to *thermodynamics properties* (page 269) can be corrected using a free rotor interpolation method.

1.2.8 New in AMS2019.3

- The *Nudged elastic band method (NEB)* (page 83) for finding minimum energy paths of transitions has been added.
- The new *PES point character property* (page 239) can be used to quickly calculate a few of the lowest vibrational modes of a system and to verify the success of a geometry optimization or transitions state search.
- *Driver level parallelism* (page 20) is now enabled and managed automatically, improving the performance and scalability of many applications.
- A *Mode Scanning* (page 254) calculation can now be *started automatically* (page 249) for all modes within a specific frequency range.
- Methods for the quick calculation of the *vibrationally resolved electronic spectra* (page 286): *Vibronic-Structure Refinement* (page 300) and *Vibronic-Structure Tracking* (page 299).
- New geometry optimizer available: *Limited-memory BFGS* (page 66)
- Input keywords that expect lists of numbers can now be specified as *ranges using a Python slice-like notation* (page 11). Input keywords that expect a single real number now also accept fractions (of integers).
- New option to include a *non-isotropic external stress* (page 330) for 1D,2D and 3D periodic systems. This can be used to study structural deformation and mechanical properties of materials under non-isotropic stress.
- New *add-on system* (page 325) for manipulating and augmenting the results returned from the *engines* (page 315):
 1. Grimme's D4 and D3 dispersion corrections can be used with any engine through the *D4Dispersion and D3Dispersion add-ons* (page 326).

1.2.9 New in AMS2019.1

- *Intrinsic Reaction Coordinate (IRC) Scan* (page 93) is now available in the AMS driver for molecular and periodic systems.
- Support for the *Grand Canonical Monte Carlo (GCMC)* (page 193) method has been added in the AMS driver.
- Molecular composition analysis for molecular dynamics simulations (see [tutorial](#))
- *Collective Variable-driven Hyperdynamics (CVHD)* (page 147)
- *Molecule gun* (page 134) and *molecule sink* (page 140) for molecular dynamics

- *PLUMED library support* (page 144) for MD analysis and a wide variety of free energy methods
- The initial symmetry of a system is enforced during geometry optimizations with the Quasi-Newton optimizer.
- *Thermodynamic properties* (page 269) (assuming an ideal gas) are automatically computed after normal modes calculations.
- *Partial vibrational density of states (PVDOS)* (page 273) for normal modes.
- The system's symmetry is used to accelerate numerical nuclear derivatives and to provide symmetry labels for normal modes.
- The AMS driver starts up much faster, significantly speeding up scripting applications that launch AMS many times.
- New tools for mode selective vibrational analysis:
 1. *Mode Scanning* (page 254) (aka ADF's ScanFreq)
 2. *Mode Refinement* (page 255) (aka "Frequency range selection")
 3. *Mode Tracking* (page 257)

INPUT, EXECUTION AND OUTPUT

2.1 Input

The input for AMS has a block and keyword structure. See the *General remarks on input structure and parsing* (page 9) section for more details.

The input keys *System* (page 31), *Task* (page 8), and *Engine* (page 315), are obligatory. Most other input keys in AMS are optional, like *Properties* (page 9). In the rest of this manual one can find all input keys for AMS.

The Engine specific input can be found in the respective Engine Manuals, for example, the manual for *ADF*, *BAND*, *DFTB*, *ForceField*, *MOPAC*, and *ReaxFF*.

2.1.1 Example

The AMS input has a special *Engine* block, that selects which engine is used for the simulation and also contains all the details of its configuration.

The following AMS input will optimize the geometry of a methane molecule and calculates its normal modes of vibration at the optimized geometry:

```
$AMSBIN/ams << EOF
Task GeometryOptimization

GeometryOptimization
  Convergence
    Gradients 1.0e-4
  End
End

Properties
  NormalModes true
End

System
  Atoms
    C      0.00000000    0.00000000    0.00000000
    H      0.63294000   -0.63294000   -0.63294000
    H     -0.63294000    0.63294000   -0.63294000
    H      0.63294000    0.63294000    0.63294000
    H     -0.63294000   -0.63294000    0.63294000
  End
End
```

(continues on next page)

(continued from previous page)

```
Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-3-1
EndEngine
EOF
```

Note how DFTB is selected as the engine in the `Engine DFTB` line that opens the Engine block. All DFTB specific configuration is contained within this engine block, which is terminated by `EndEngine`. The fact that we want to run a geometry optimization with normal modes for methane and things like convergence criteria for the optimization are independent from which engine is actually used to perform this calculation. Therefore they are all found *outside* of the Engine block.

In this sense, the AMS input is split up into

- the driver level input (everything outside of the engine block), and
- the engine input, which is just a single Engine block.

This separation makes it easy to perform the same calculation at a different level of theory, by simply switching out the Engine block in the input. We could, for example, repeat the same calculation at the DFT-GGA level using the BAND engine:

```
Engine BAND
  XC
  GGA PBE
  End
EndEngine
```

Engines like ADF or BAND that have many options and can calculate many properties, consequently also have a large number of possible keywords in their input. The engine input options can be found in separate *engine specific manuals* (page 315).

This AMS driver manual only documents the driver level keywords **outside** of the Engine block.

In general all engines can be used with all tasks in AMS. There are only a few restrictions. For example, only engines which can handle periodic systems can be used for the calculation of phonons.

2.1.2 Tasks

The key Task is described in more detail in the section *Structure, Reactivity, and Molecular Dynamics* (page 47). The Task *VibrationalAnalysis* is described in more detail in the section *Vibrational Spectroscopy* (page 246) (Mode Scanning, Mode Refinement, Mode Tracking, VG-FC Resonance Raman) and in the section *Vibrationally resolved electronic spectra* (page 286) (VG-FC: Vertical Gradient Franck-Condon).

Below the possible arguments for Task are given.

```
Task [GCMC | GeometryOptimization | IRC | MolecularDynamics | NEB | PESExploration | ↵
↵PESScan |
      Replay | SinglePoint | TransitionStateSearch | VibrationalAnalysis]
```

2.1.3 Properties

The block key `Properties` is described in more detail in the section *Gradients, Hessian, Stress tensor, Elasticity* (page 235), in the section *Vibrational Spectroscopy* (page 246), and in the section *Dipole moment, Polarizability, Bond orders* (page 307).

Below the possible `Properties` are summarized.

```
Properties
  BondOrders Yes/No
  Charges Yes/No
  DipoleGradients Yes/No
  DipoleMoment Yes/No
  ElasticTensor Yes/No
  GSES Yes/No
  Gradients Yes/No
  Hessian Yes/No
  Molecules Yes/No
  NormalModes Yes/No
  OrbitalsInfo Yes/No
  Other Yes/No
  PESPointCharacter Yes/No
  Phonons Yes/No
  Polarizability Yes/No
  Raman Yes/No
  SelectedRegionForHessian string
  StressTensor Yes/No
  VCD Yes/No
  VROA Yes/No
End
```

2.1.4 General remarks on input structure and parsing

- Most keys are optional. Defaults values will be used for keys that are not specified in the input
- Keys/blocks can either be *unique* (i.e. they can appear in the input only once) or *non-unique*. (i.e. they can appear multiple times in the input)
- The order in which keys or blocks are specified in the input does not matter. Possible exceptions to this rule are a) the content of non-standard blocks b) some non-unique keys/blocks)
- Comments in the input file start with one of the following characters: `#`, `!`, `::`:

```
# this is a comment
! this is also a comment
:: yet another comment
```

- Empty lines are ignored
- The input parsing is **case insensitive** (except for string values):

```
# this:
UseSymmetry false
# is equivalent to this:
USESMMETRY FALSE
```

- Indentation does not matter and multiple spaces are treaded as a single space (except for string values):

```
# this:
    UseSymmetry      false
# is equivalent to this:
UseSymmetry false
```

- Environment variables in the input will be replaced by their value. We only support simple substitution of variables (optionally surrounded by {}). All other [parameter expansion features of the POSIX shell](https://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_06_02) (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_06_02) (e.g. default values) are **not** supported. Expansion of a variable can be prevented by prefixing the leading dollar (\$) with a backslash (\). Note that variables which are not set will expand to an empty string, just like in a shell:

```
File $SOME_DIRECTORY/file.txt
File ${SOME_DIRECTORY}_BACKUP/file.txt
Key  \ $WILL_NOT_EXPAND
```

2.1.5 Keys

Key-value pairs have the following structure:

```
KeyName Value
```

Possible types of keys:

bool key

The value is a single Boolean (logical) value. The value can be `True` (equivalently `Yes`) or `False` (equivalently `No`). Not specifying any value is equivalent to specifying `True`. Example:

```
KeyName Yes
```

integer key

The value is a single integer number. Example:

```
KeyName 3
```

float key

The value is a single float number. For scientific notation, the E-notation is used (e.g. -2.5×10^{-3} can be expressed as `-2.5E-3`). The decimal separator should be a dot (`.`), and **not** a comma (`,`). Example:

```
KeyName -2.5E-3
```

Note that fractions (of integers) can also be used:

```
KeyName 1/3      (equivalent to: 0.3333333333...)
```

string key

The value is a string, which can include white spaces. Only ASCII characters are allowed. Example:

```
KeyName Lorem ipsum dolor sit amet
```

multiple_choice key

The value should be a single word among the list options for that key (the options are listed in the documentation of the key). Example:

```
KeyName SomeOption
```

integer_list key

The value is list of integer numbers. Example:

```
KeyName 1 6 0 9 -10
```

Note that one can also specify ranges of integers by specifying the interval and (optionally) the step size separated by colons:

```
KeyName 1:5      (equivalent to: 1 2 3 4 5)
KeyName 2:10:2   (equivalent to: 2 4 6 8 10)
KeyName 20:10:-2 (equivalent to: 20 18 16 14 12 10)
```

Note also that ranges can be freely combined with individual numbers:

```
KeyName 1:5 10 20 (equivalent to: 1 2 3 4 5 10 20)
```

float_list key

The value is list of float numbers. The convention for float numbers is the same as for Float keys. Example:

```
KeyName 0.1 1.0E-2 1.3
```

Float lists can also be specified as a range with equidistant points, by specifying the interval's boundaries (inclusive) as well as the number of desired subintervals separated by colons:

```
KeyName 1.0:1.5:5 (equivalent to: 1.0 1.1 1.2 1.3 1.4 1.5)
```

Range specifications can be freely combined with each other and single numbers:

```
KeyName 0.0 1.0:1.5:5 2.0:3.0:10
```

2.1.6 Blocks

Blocks give a hierarchical structure to the input, grouping together related keys (and possibly sub-blocks). In the input, blocks generally span multiple lines, and have the following structure:

```
BlockName
  KeyName1 value1
  KeyName2 value2
  ...
End
```

Headers

For some blocks it is possible (or necessary) to specify a *header* next to the block name:

```
BlockName someHeader
  KeyName1 value1
  KeyName2 value2
  ...
End
```

Compact notation

It is possible to specify multiple key-value pairs of a block on a single line using the following notation:

```
# This:
BlockName KeyName1=value1 KeyName2=value2

# is equivalent to this:
BlockName
  KeyName1 value1
  KeyName2 value2
End
```

Notes on compact notation:

- The compact notation cannot be used for blocks with headers.
- Spaces (blanks) between the key, the equal sign and the value are ignored. However, if a value itself needs to contain spaces (e.g. because it is a list, or a number followed by a unit), the entire value must be put in either single or double quotes:

```
# This is OK:
BlockName Key1=value Key2 = "5.6 [eV]" Key3='5 7 3 2'
# ... and equivalent to:
BlockName
  Key1 value
  Key2 5.6 [eV]
  Key3 5 7 3 2
End

# This is NOT OK:
BlockName Key1=value Key2 = 5.6 [eV] Key3=5 7 3 2
```

Non-standard Blocks

A special type of block is the *non-standard block*. These blocks are used for parts of the input that do not follow the usual key-value paradigm.

A notable example of a non-standard block is the `Atoms` block (in which the atomic coordinates and atom types are defined).

2.1.7 Including an external file

You can include an external ASCII file in the input with the `@include` directive:

```
@include FileName.in
@include "file name with spaces.in"
```

The file name should include the path, either absolute or relative to the run-directory. The content of the file is included in the input at the point where the `@include` directive occurs. The `@include` directive may occur any number of times in the input.

The `@include` feature makes it easy to pack your preferred settings in one file and use them in every run with minimum input-typing effort.

2.1.8 Units

Some keys have a default unit associated (not all keys have units). For such keys, the default unit is mentioned in the key documentation. One can specify a different unit within square brackets at the end of the line:

```
KeyName value [unit]
```

For example, assuming the key `EnergyThreshold` has as default unit `hartree`, then the following definitions are equivalent:

```
# Use defaults unit:
EnergyThreshold 1.0

# use eV as unit:
EnergyThreshold 27.211 [eV]

# use kcal/mol as unit:
EnergyThreshold 627.5 [kcal/mol]

# hartree is the atomic unit of energy:
EnergyThreshold 1.0 [hartree]
```

Available units:

- **Energy:** hartree, joule, eV, kJ/mol, kcal/mol, cm1, MHz
- **Forces:** hartree/bohr, eV/bohr, hartree/angstrom, eV/angstrom, kcal/mol/angstrom, kJ/mol/angstrom
- **Length:** bohr, angstrom, meter
- **Angles:** radian, degree
- **Mass:** el, proton, atomic, kg
- **Pressure:** atm, pascal, GPa, a.u., bar, kbar
- **Electric field:** V/angstrom, V/meter, a.u.

2.2 Execution

2.2.1 Shell script

The AMS driver reads its input from standard input, i.e. what is called `STDIN` on Unix-like systems. Technically it is possible to run AMS and type the input file in interactively. This is however highly impractical and most people run AMS from a small shell script that contains the AMS text input and sends it directly to the AMS executable. For example, the content of the file `example.run` could be like:

```
#!/bin/sh

$AMSBIN/ams << EOF

... AMS text input goes here:
```

(continues on next page)

(continued from previous page)

```

Block
  Keyword value
  OtherKeyword value
End
EOF

```

The shell script ‘example.run’ needs to be executable, if it isn’t you will need to make it executable, e.g. `chmod u+x example.run`. The ‘example.run’ file needs to be executed as a shell script, not as input to AMS.

```
./example.run > example.out
```

2.2.2 Python interface

There is a complete Python interface to AMS, which allows users to set up and run arbitrary AMS jobs, and to conveniently analyze the calculation results directly from Python. In this way AMS jobs can be automatized and complex multi-stage workflows implemented.

The scripting framework is called **PLAMS** as in “Python Library for Automating Molecular Simulation”, which conveniently can also be read as “Python Layer for AMS”. It is documented in a separate manual:

- [PLAMS introduction](#)
- [Running AMS through PLAMS](#)

2.2.3 Interactive input file

As of the 2021.1 version the AMS driver has the concept of an interactive input file. The interactive input file is a second input file, that users may create in the [results directory](#) (page 18) of an already running job to interact with it. The name of the interactive input file is `interactive.in`. Note that the file does not exist by default, but has to be created by the user wanting to interact with the running AMS job.

In AMS2021.1 the interactive input file may be used to request the graceful and orderly termination of a job.

```
echo "Stop" > ams.results/interactive.in
```

Stop

Type

Bool

Default value

No

Description

Makes the AMS driver stop execution in an ordered manner at the next opportunity. This will not interrupt the execution in the middle of the numerical calculation of a PES point property (e.g. a Hessian).

It is generally preferable to request the termination through the interactive input file over killing a job at the operating system level. This makes sure that the job stops at a logical point and still has time to write intermediate results to disk, making restarting from such jobs easier and more reliable. From the graphical user interface, the stopping can be requested in AMSjobs by selecting the running job and clicking **Job → Request Early Stop** in the menu bar.

As of AMS2021.1, the tasks that can be stopped through the interactive input file are:

- *Molecular dynamics* (page 101) simulations will stop at the next sampling or checkpoint as if the maximum number of steps had been reached.
- *Geometry optimizations* (page 49) will stop after the current iteration and be considered converged. This can be used if a geometry optimization does not make progress anymore, e.g. because the convergence thresholds are set tighter than the noise level of the used engine.
- *Nudged elastic band* (page 83) calculations will stop after the current iteration (i.e. the calculations for all images) is complete.
- *PES scans* (page 76) will interrupt the currently running geometry optimization and skip the optimization for all further PES points.
- *PES explorations* (page 202) will terminate after the currently running expedition. No new explorers will be dispatched, but currently active explorers will be allowed to finish.

Note that the `Stop` keyword in the interactive input file will not interrupt the job in the middle of the numerical calculation of a *PES point property* (page 307), such as a Hessian or stress tensor. It will also never stop a job that is currently running an engine, and will hence not stop an engine that is calculating these properties analytically. It is possible to interrupt the numerical calculation of *PES point properties* (page 307) using the `StopProperties` keyword. We advise users to be careful with this keyword and always combine it with the `Stop` keyword, as it is generally not possible to skip the evaluation of properties, but continue with a task as the AMS driver level. (After all, the driver may need the properties that we interrupt.)

```
echo "Stop" > ams.results/interactive.in
echo "StopProperties" >> ams.results/interactive.in
```

StopProperties

Type

Bool

Default value

No

Description

Makes the AMS driver stop execution during the evaluation of PES point properties. This may for example be used to interrupt the numerical calculation of a Hessian. Note that this just works at the level of the loops used for the `**numerical**` calculation of the properties, so engines which calculate a property analytically may not be interruptible using this keyword. It is recommended to use this keyword in combination with the generic `Stop` keyword.

As of AMS2021.1 it is not possible to restart the calculation of properties that were interrupted like this. Hence the use of the `StopProperties` keyword is limited.

2.2.4 Running AMS on compute clusters

AMS is parallelized with MPI and can therefore be run in parallel on distributed memory machines, aka compute clusters. See the [installation manual](#) for general documentation on how to set up and run all the programs from the Amsterdam Modeling Suite on compute clusters. In this section we give some more advice that is specific to the AMS driver and its engines.

Normally users use the login node to prepare their jobs and input files somewhere in their home directory, and also want the results of their jobs to end up there. Quite often, compute clusters are set up such that the user's home directory is also mounted on the compute nodes, usually via NFS (Network File System). Before the introduction of the AMS driver it was not recommended to `cd` to the home directory in the submission script and have the compute nodes execute the job directly there. This was simply due to the fact that a lot of file I/O was done on temporary files in the present working directory, which in this case would be on a slow network-mounted file system.

On the other hand, with AMS, switching to the home directory is the preferred way of running on a cluster where the home directory is mounted on the compute nodes. Running in the home directory mounted over NFS does not come with a performance penalty for AMS, but has many advantages. This is because AMS and its engines are already built under the assumption that access to this directory is slow. Basically there are three directories that are used by the AMS driver and its engines:

1. The **starting directory**, i.e. the present working directory at the time the AMS driver is started. This folder is generally read-only for AMS, except for creating the results directory there at the beginning of a calculation. Note that all relative paths in the AMS input, e.g. for loading results from previous calculations, are relative to the starting directory. The starting directory is assumed to be on a slow filesystem, but since data is normally only read once from there in the beginning of a calculation, this is in practice not a problem.
2. The **results directory**, where the results of a calculation as well as important intermediate steps (e.g. restart files) are collected. It also contains the log file which can be used to monitor a running calculations. The results directory is assumed to be on a slow filesystem, so AMS and its engines will be very careful not to do much disk I/O there. Generally something is only written to the results directory when AMS is sure that it should remain on disk when the calculation finishes. The results directory can also contain some intermediate restart files, so the contents of the result directory should be all that is needed in case the calculation crashes or is killed before it finishes normally.
3. The **scratch directory**, the location of which is set with the `$SCM_TMPDIR` environment variable, see also the [installation manual](#). This directory should be put on a fast disk, e.g. an SSD in the compute node, as it will be used to store temporary results on disk. Users do not really need to care or know about the temporary files in the scratch directory. Normally, any files and directories created in the scratch directory are cleaned up at the end of the calculation. In case of errors, AMS tries to copy anything useful (e.g. the text output of all the different ranks) to the results directory in order to make finding the problem easier. However, for some kinds of crashes (or if the `SIGKILL` signal is sent to AMS), the cleanup of the scratch directory might not be performed, in which case users might want to manually check or remove the `amstmp_*` folders in the scratch directory.

With this setup there is no performance penalty for running directly on a network mounted home directory: Results will just be put there immediately, instead of being copied there at the end of a calculation.

Normally all batch systems provide an environment variable that is set to the directory from which the job was submitted, which is then where one should `cd` in the run script:

```
#!/bin/sh

if [ -z "$PBS_O_WORKDIR" ]; then
    # PBS batch system
    cd "$PBS_O_WORKDIR"
elif [ -z "$SLURM_SUBMIT_DIR" ]; then
    # Slurm batch system
    cd "$SLURM_SUBMIT_DIR"
elif [ -z "..." ]; then
    # add other batch systems as necessary ...
    cd "..."
fi

export AMS_JOBNAME=myJob

$AMSBIN/ams << EOF

# Normal AMS text input, but with all paths
# relative to where the job was submitted from, e.g.:
LoadSystem previousJob.results/ams.rkf

EOF
```

With this runscript the AMS driver would make a `myJob.results` folder in the directory where the job was submitted

from, and there is no need to copy results around manually in the run script. Furthermore this runscript always produces exactly the same files in the same locations, no matter if it is run interactively or submitted to a compute node through the batch system. Furthermore all paths in the input file can be specified relative to the location from where the runscript is submitted (normally the folder in which the runscript is located). This removes the need to copy or specify absolute paths to previous results, e.g. when restarting calculations. Finally, files useful for monitoring the running calculation are also conveniently there and not hidden somewhere on the compute node.

2.2.5 AMS environment variables

The behavior of AMS related to the output can be modified through a number of environment variables.

AMS_JOBNAME

Sets the name of a job. This name is used to determine the name of the results folder AMS creates, which is `$AMS_JOBNAME.results` or `ams.results` if this environment variable is not set.

AMS_RESULTSDIR

If this environment variable is set, instead of creating a new results folder, AMS will use the set directory as the results folder. Note that the directory set here will *not* be created by AMS and therefore has to exist before starting AMS. Note that this environment variable can be used to prevent AMS from creating result folders, by setting `AMS_RESULTSDIR=.` This reproduces the pre-AMS behavior of putting all result files into the directory from which a job is started.

AMS_SWITCH_LOGFILE_AND_STDOUT

If this environment variable is set, AMS will redirect what is normally printed on standard output to a file (`ams.out`) in the results directory. Instead the contents of the log file (`ams.log`) will be printed to standard output while a job is running, allowing users to easily monitor the jobs progress. Note that the log file will still be created normally as if this environment variable was not set. This environment variable is just a convenience feature for users that would always redirect their output into a file and then use `tail -f` on the log file to monitor the running calculation.

This is an example run-script using the `AMS_SWITCH_LOGFILE_AND_STDOUT` and `AMS_JOBNAME` environment variables:

```
#!/bin/sh

# By setting AMS_SWITCH_LOGFILE_AND_STDOUT, the (more compact)
# logfile will be printed to standard output while the full
# text output of the calculation is redirected to the file
# ams.out in the ams results folder

export AMS_SWITCH_LOGFILE_AND_STDOUT=true

# By default ams creates a folder 'ams.results' and puts the
# results of the calculation there. If we set AMS_JOBNAME, the
# results folder will instead be called $AMS_JOBNAME.results
# (in this case, 'H2_optimization.results')

AMS_JOBNAME=H2_optimization $AMSBIN/ams <<eor
System
  Atoms
    H 0 0 0
    H 0 0 1
  End
End
Task GeometryOptimization
Engine DFTB
```

(continues on next page)

(continued from previous page)

```
Model GFN1-xTB
EndEngine
eor
```

2.3 Output

AMS produces two ASCII files: standard output and the `ams.log` file, and produces several binary output data files. The `ams.log` file is a very concise summary of the calculation's progress during the run. The binary output data files contain job characteristics and computational results produced by AMS and the Engine. Part of what is written to the binary output files is also written in a human readable form to standard output.

2.3.1 Results directory

Note that AMS does not put any of its binary output files and the `ams.log` file into the present working directory, as virtually all of the standalone programs in the suite did. Instead AMS creates a `*.results` directory, which collects all result file associated with a job. Here `*` is replaced by the jobname, which is set with the `AMS_JOBNAME` environment variable:

```
AMS_JOBNAME=methane $AMSBIN/ams << EOF

... see Input example before ...

EOF
```

This would put all results related to our geometry optimization of methane into the newly created folder `methane.results`. (The default name of the results folder is `ams.results` if `AMS_JOBNAME` is not set, see the [environment variables](#) (page 17) section of this manual for documentation of all environment variables used by AMS.) In this way users can easily run multiple jobs in the same directory without danger of clashing output files, which was a common problem before the introduction of AMS. This new setup is also more consistent with the graphical user interface, which already collected all files associated with a specific job into a dedicated results directory. Note that AMS will by default **not overwrite** results directories if a job is rerun or another job is run with the same jobname.

2.3.2 Logfile `ams.log`

Inside of the results directory users will always find the logfile `ams.log`, which is written during a running calculation and can be used to monitor its progress.

The logfile `ams.log` is generated during the calculation and flushed after (almost) each message that is sent to it by the program. Consequently, the user can inspect it and see what is going on without being delayed by potentially large system I/O buffers. Each message contains date and time of the message plus additional info.

Be alert on error messages. Take them seriously: inspect the standard output carefully and try to understand what has gone wrong. Be also alert to warnings. They are not necessarily fatal but you should understand what they are about before being satisfied with the results of the calculation. Do not ignore them just because the program has not aborted: in some cases the program may not be able to determine whether or not you really want to do what appears to be wrong or suspicious. If you believe that the program displays erratic behavior, then the standard output file may contain more detailed information. Therefore, in such case save the complete standard output file, together with the logfile `ams.log`, in case we need these files for further analysis.

2.3.3 Binary output files

The results directory contains binary result files in the KF format, which can be opened and inspected with the [KFbrowser](#) GUI component. KF stands for Keyed File: KF files are keyword oriented, which makes them easy to process by simple procedures. KF files are Direct Access binary files.

- The main `ams.rkf` written by the AMS driver. It contains high level information about the trajectory that the AMS driver took over the potential energy surface. For a geometry optimization it would for example contains the history of how the systems geometry changed during the optimization as well as the final optimized geometry. For a molecular dynamics simulation it would contain the full trajectory. The format in which this information is written is independent from which engine was used for a calculation.

Note: For a full description of the data on the `ams.rkf` file, see the [KF output files](#) (page 909) page.

- The engine specific main binary output file written by the engine (and partly by the AMS driver). This file is kept for only one special point, e.g. the final geometry in a geometry optimization. The ADF engine writes `adf.rkf` (instead of `TAPE21` in older versions). The BAND engine writes `band.rkf` (instead of `RUNKF` in older versions). The DFTB engine writes `dftb.rkf`. If a property, like vibrational modes, is tied to this special point on the potential energy surface, it is stored in this file. Also all engine specific properties are written to the main binary output file, like orbitals in case of a quantum mechanical engine.

Table 2.1: Engine specific main binary output file

Engine	main file
ADF	<code>adf.rkf</code>
BAND	<code>band.rkf</code>
DFTB	<code>dftb.rkf</code>
ForceField	<code>forcefield.rkf</code>
MOPAC	<code>mopac.rkf</code>
ReaxFF	<code>reaxff.rkf</code>
External	<code>external.rkf</code>
MLPotential	<code>mlpotential.rkf</code>
ASE	<code>ase.rkf</code>
QuantumEspresso	<code>quantumespresso.rkf</code>

- Additionally there might be an engine specific binary output file for every point on the potential energy surface that was visited during the calculation. Like the engine specific main binary output file they contain information tied to a specific point on the potential energy surface. These engine output files all have the extension `.rkf`, but their filename is usually somehow descriptive of the point on the PES that they correspond to. Note that one does not always get an engine output file for every PES point that was visited during the calculation. For most applications this would just be too much data.
- Other engine specific binary (and ASCII) output files written by the engine.

Having multiple different binary output files could be confusing for people that are used to the single result file that was written by the standalone programs in ADF<=2017. After all, it brings up the question in which file the desired property is stored. The general rule is: If the property is tied to a particular point on the potential energy surface, it is stored in the engine output file belonging to that particular point. This includes the Hessian, stress tensor, elastic tensor, normal modes of vibration, phonons, Raman intensities and other vibrational properties. If the information depends on the entire trajectory over the PES, it is found in the main `ams.rkf` written by the AMS driver.

2.3.4 Standard output

The standard output file contains in a human readable form part of the job characteristics and computational results produced by AMS and the Engine.

2.3.5 Optimized geometry in xyz format

In case of *Geometry Optimizations* (page 49), the optimized geometry is saved as an *extended XYZ file* (page 565) in the results directory with the name `output.xyz`.

Note that the optimized geometry is also stored in the binary file `ams.rkf` and is printed to the standard output.

2.4 Driver level parallelism

See also:

See also the [GUI tutorial](#) on the parallel scalability of the calculation of elastic tensors.

AMS is a parallel program using MPI for efficient execution on distributed memory machines, aka compute clusters. For most jobs, the AMS driver part of a calculation is computationally not particularly costly and most of the execution time is spent inside of the *compute engines* (page 315). Therefore the main parallelization of AMS is inside of the engines, making sure that a good performance is obtained for *tasks* (page 47) such as *molecular dynamics* (page 101) or *geometry optimizations* (page 49), which consist of a series of interdependent engine invocations: We need to have completed step n before we can continue with step $n + 1$.

However, not all workloads are of this sequentially dependent type. Some jobs have a lot of independent work, that can be done in parallel. This kind of trivial parallelizability can be exploited at the AMS driver level: Instead of having all cores collaborate on a single PES point and then doing all needed PES points sequentially, we can just distribute the available PES points over the all the available cores. Normally this leads to a better parallel scaling than the default parallelization inside of the engines: Parallelizing the engines is relatively complicated and often requires a lot of communication between cores. Parallelizing on the driver level on the other hand is very easy, and often the only communication required is at the very end of the calculation, when results are collected.

Note that it is perfectly possible to combine both the in-engine parallelization and the driver level parallelism: At the driver level we could split our e.g. in total 32 cores into 4 groups of 8 cores, and then have each group of 8 use the in-engine parallelization to collaborate on a specific calculation. This is especially useful if the total number of cores is larger than then number of independent calculations we have to do. It might also be that we have a very large number of calculations to do, but not enough memory to let every core work alone on its own calculation, as would be ideal from a parallel scaling point of view.

Because of the two levels of parallelism – both at the driver and the engine level – we call this setup **double parallelization**.

Starting with the AMS2019.3 release, driver level parallelism is used and configured automatically. That means that the AMS driver will automatically parallelize at the driver level when it is possible and considered advantageous. As such it should normally not be necessary for users to explicitly configure the driver level parallelism.

Driver level parallelism can be used for the calculation of the *PES point properties* (page 235) which are derivatives, if these need to be done numerically:

- Numerical calculation of *forces / nuclear gradients* (page 237). With a double sided derivative this requires $6 \times n_{\text{atoms}}$ independent calculations on geometries with one atom displaced along a cartesian coordinate.
- Numerical calculation of the *stress tensor* (page 241) for periodic systems. This requires up to 12 calculations for a double sided derivative along the 6 strain directions, but might require less in case some of the strains are symmetry equivalent.

- Numerical calculation of the *Hessian* (page 238) and normal modes of vibration. This is currently only supported for engines that calculate nuclear gradients analytically and done by numerically differentiating this first (analytic) derivative. As such it requires $6 \times n_{\text{atoms}}$ independent calculations on geometries with one atom displaced along a cartesian coordinate.
- Numerical calculation of the *elastic tensor* (page 242). This requires 84 independent geometry optimizations on systems with differently strained lattices, with each optimization having a variable number of steps.
- Numerical calculation of *phonons* (page 274). This requires at most $6 \times n_{\text{atoms}}$ displacements, but might require less in case some of the displacements are symmetry equivalent. Note that the displacements are done in a super cell system, which for many engines will increase the memory requirements, but also improve the in-engine parallel scalability.
- The forward and backward displacements along normal modes for the *Mode Scanning* (page 254), *Mode Refinement* (page 255), and *Mode Tracking* (page 257).

There are also tasks using driver level parallelism, e.g. *Nudged Elastic Band* (page 83), for which the calculations of all the images is trivially parallel.

Details of the driver level parallelism, i.e. how much to parallelize at the driver level, are generally configured for the above mentioned cases on an individual basis, because one might want a different grouping strategy for each case. For each case there is a separate `Parallel` block somewhere in the input (e.g. `ElasticTensor%Parallel` for the calculation of the elastic tensor), which has the following keywords:

```
Parallel
  nGroups integer
  nCoresPerGroup integer
  nNodesPerGroup integer
End
```

Note that only one of them should be specified in the input, depending of course on what is the desired strategy for parallelization.

nGroups n

Splits all cores evenly into *n* groups. We recommend choosing *n* such that it divides the total number of cores without a remainder.

nCoresPerGroup n

Each group consists of *n* cores. As such `nCoresPerGroup 1` results in the maximum possible parallelism at the driver level. We recommend choosing *n* such that it divides the total number of cores without a remainder.

nNodesPerGroup n

Makes groups from all cores within *n* nodes, e.g. `nNodesPerGroup 1` would make every cluster node into a separate group. Note that this option should *only be used on homogeneous compute clusters*, where all used nodes have the same number of cores. Otherwise cores from different nodes will be grouped together in very surprising and unintended ways, probably resulting in suboptimal performance.

The optimal grouping strategy and number of groups depends on the total number of cores used in the calculation, the amount of independent tasks to be done in parallel, as well as the parallel scalability of the engine itself. In practice it can be a bit tricky, which is why the grouping strategy is determined automatically since AMS2019.3.

However, sometimes it can be useful to configure the groups manually. Suppose, as an example, that we want to calculate the elastic properties of a bulk material on a 32 core machine. The calculation of the *elastic tensor* (page 242) should be done on a relaxed geometry, including relaxed lattice degrees of freedom. We therefore first perform a geometry optimization, before calculating the elastic tensor. In AMS this can easily be done with the following input:

```
Task GeometryOptimization

GeometryOptimization
```

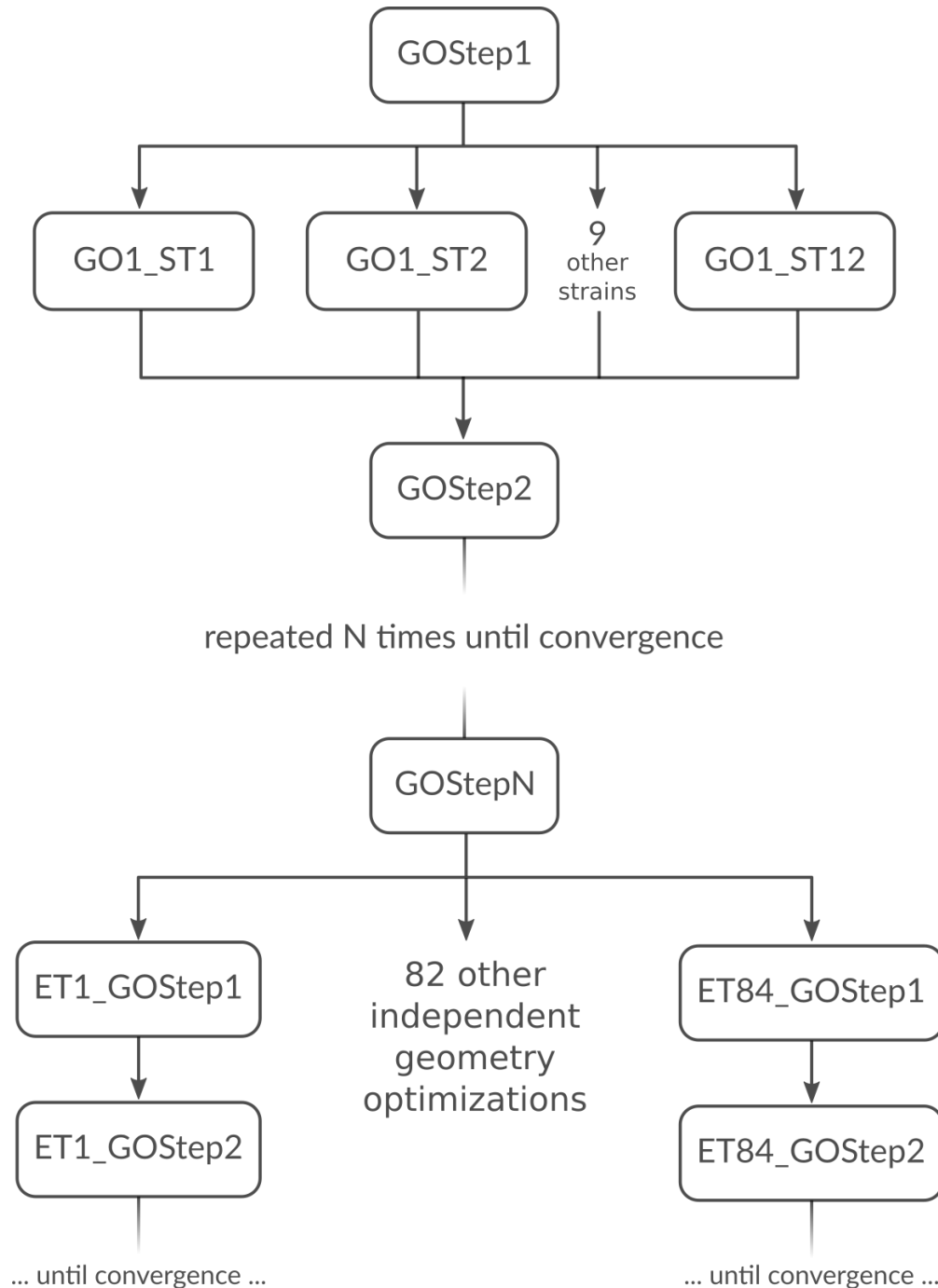
(continues on next page)

(continued from previous page)

```
    OptimizeLattice True
End

Properties
    ElasticTensor True
End
```

But what is the most optimal parallel setup for this calculation? First we recognize that performing a lattice optimization requires the calculation of the *stress tensor* (page 241) at every step of the optimization. Assuming that our bulk system does not have any symmetries AMS can exploit, the numerical calculation of the stress tensor (assuming the engine can not calculate it analytically) would require 12 independent strained calculations for every step in the geometry optimization. Once the geometry optimization is converged, we have to perform 84 independent geometry optimizations to determine the elements of the elastic tensor. In summary, the graph of dependencies between all these tasks looks like this:



How do we best parallelize this? For the main steps, e.g. **GOSStep1** there is no question: We have nothing to do in parallel and all 32 cores work on it together to finish it as quickly as possible. For the numerical calculation of the stress tensor we have 12 tasks that can be done in parallel by the 32 cores in our machine. Now 12 obviously does not divide 32 without a remainder, so there is no way to split into equally sized groups and do all 12 strains in parallel. The greatest common divisor of 12 and 32 is 4, so it's probably best to split into 4 groups of 8 cores each. This is done with `nGroups 4`. Each group would then do 3 of the 12 strained calculations sequentially, using the in-engine parallelization to speed up the individual calculations. Once the stress tensor is computed in this way all groups merge and all 32 cores work together on **GOSStep2**. This splitting and merging now continues until the geometry optimization is converged. For the elastic tensor we now have 84 tasks to perform in parallel, where each task is a completely separate geometry optimization (without optimizing the lattice) of a strained system. 84 tasks is more than double the number of cores we have. In this

case it is probably not too bad to just run as parallel as possible at the driver level and make 32 “groups” of just one core to throw the 84 tasks at. This is easily done by setting `nCoresPerGroup 1` in the `ElasticTensor` block. Putting everything together we should add the following to our input file in order to optimally utilize our machine for this example calculation:

```
NumericalDifferentiation
  Parallel
    nGroups 4
  End
End

ElasticTensor
  Parallel
    nCoresPerGroup 1
  End
End
```

2.5 Pipe interface

AMS can interact with other programs using a *custom communication protocol* (page 24). This enables two independent processes to communicate over a pair of data pipes (FIFOs), exchanging data in a highly efficient manner. One of the processes is the “pipe master”, driving the calculation and sending the atoms, coordinates etc. to the other process to perform calculations. The other process is the “pipe worker”, receiving requests from the master, performing the requested calculations and returning the results such as energies and gradients.

For technical details on the AMSPipe protocol see:

2.5.1 AMSPipe protocol specification

The AMSPipe protocol is a remote procedure call protocol used for communication between two processes on the same computer. One of these processes is the “pipe master”, submitting geometries and requests for calculations and receiving the results. The other process is the “pipe worker”, which listens for calls from the master, executes the requested calculations and returns their results.

Communication takes place over a pair of pipes, a “call pipe” (master to worker) and a “reply pipe” (worker to master). Each pipe carries a sequence of messages, which are processed in order. Each message is an associative object (like a JSON object or a Python dict) containing a single item. The key of this item denotes the name of the message, while the value of this item is another object containing the payload.

Each message sent over the call pipe constitutes a method call. The name of the message is equal to the name of the method being called, while the payload object contains any arguments. For methods without arguments, the payload is an empty object.

Once the worker receives a call message, it will execute the specified method and potentially write a sequence of zero or more messages to the reply pipe, containing the results of the calculation. After the execution of the method is completed, the worker will write a “return” messages to the reply pipe, denoting the success or failure of the method call.

Any method whose name starts with “Set” doesn’t send a “return” message. Instead, errors encountered during the execution of a Set method are buffered by the worker to be returned the next time a “return” message is generated. While an error is buffered, any further method calls are ignored by the worker and are not executed. Once a non-Set call is seen by the worker, such a call will also be discarded without being executed, but a “return” message will be immediately generated with the buffered error data. The buffered error will then be cleared and normal processing of calls resumes.

As a special case, the “Exit” method may be called at any time and will be immediately executed by the worker, terminating the protocol session. The “Exit” method never returns a “return” message, discarding any potentially buffered error.

All values are always in atomic units: Hartree for energies, Bohr for distances, Hartree/Bohr for gradients etc.

Low-level message encoding

All messages are encoded using Universal Binary JSON. AMSPipe implementations MAY encode messages using the UBJSON optimized container format (with explicit length and/or strong typing). Implementations MUST support decoding UBJSON containers whether or not they're written using optimized format.

UBJSON arrays in messages MUST NOT contain other arrays or objects. Multi-dimensional arrays MUST be flattened on encoding (and unflattened on decoding as necessary). Flattening SHOULD preserve the original order of elements in memory. All arrays MUST be accompanied by an additional integer array holding the original dimensions before flattening. The name of this auxiliary array is equal to the name of the main array with a `_dim_` suffix. The dimensions are written in Fortran (column-major) order, so that the first value in the `_dim_` array corresponds to the index that changes the fastest when iterating over consecutive elements of the flattened array.

Arrays also MUST NOT contain elements of incompatible types. The elements of an array MUST be either all integers, all real numbers, all Boolean, or all strings. UBJSON type "char" is equivalent to a UBJSON "string" of length 1 (elements of type "char" and "string" MAY thus be mixed in an array). An empty array is equivalent to an array that is not present at all.

When sending messages over stream transport mechanisms (such as UNIX pipes), each message MUST be prefixed with a 32-bit native endian integer length.

Return messages and error handling

The "return" message consists of the following fields:

- "status" (integer): Zero for success or one of the error codes listed below.
- "method" (string, optional): Name of the method in which the error occurred.
- "argument" (string, optional): Name of the argument in error.
- "message" (string, optional): Human-readable error message.

The "status" field can have one of the following values:

0: success

No error, method executed successfully.

1: decode_error

Message could not be decoded correctly (invalid UBJSON encoding or a violation of one of the constraints above).

2: logic_error

Programmer error, such as methods called in an incorrect sequence or with an invalid worker state.

3: runtime_error

An error occurred during the execution of a method (outside of the AMSPipe protocol).

4: unknown_version

(Only returned from "Hello".) Requested protocol version is not supported by the worker.

5: unknown_method

A method of the requested name is not supported by the worker.

6: unknown_argument

The argument in "argument" is not known to the worker and couldn't be processed. If multiple arguments to given single call are unknown, "argument" will be set to the one at the lowest nesting depth (if an argument contains nested objects). If multiple fields at the same nesting depth are unknown, the first one in ASCII sort order will be returned.

7: `invalid_argument`

An argument doesn't have the correct type or dimensions, or it has an invalid value.

Methods

`Hello(version)`

- “version” (integer):

Attempt to activate a given version of the AMSPipe protocol. Only version 1 is defined at the moment.

Other methods (with the exception of “Hello” and “Exit”) MUST NOT be called until a Hello has completed successfully. Pipe masters SHOULD attempt a Hello with the highest supported protocol version and iterate downwards if an `unknown_version` error is returned. Once a Hello has succeeded, it MUST NOT be called again during the lifetime of a pipe session.

`Exit()`

Terminate the worker and disconnect both pipes. This method never returns.

The worker MAY also discard any remembered calculations.

`SetCoords(coords)`

- “coords” (`real(3,:)`):

Replace the Cartesian coordinates in the current chemical system. The number of atoms must match.

`SetLattice(vectors)`

- “vectors” (`real(:,:)`):

Replace the lattice matrix of the current chemical system. If “vectors” is absent or of dimensions (0,0), make the system non-periodic.

`SetSystem(atomSymbols, coords, totalCharge)`

- “atomSymbols” (`string(:)`):
- “coords” (`real(3,:)`):
- “totalCharge” (`real`):

Define a new chemical system.

Solve(request, keepResults, prevTitle)

- **“request” (object):**
 - “title” (string): Unique string key identifying this calculation.
 - “quiet” (bool): If true, the worker SHOULD keep any standard output from the calculation to a minimum.
 - “gradients” (bool): Calculate gradients on atoms.
 - “stressTensor” (bool): Calculate the stress tensor.
 - “elasticTensor” (bool):
 - “hessian” (bool):
 - “dipoleMoment” (bool):
 - “dipoleGradients” (bool):
- “keepResults” (bool, default false): Remember worker state for future restart.
- “prevTitle” (string, optional): Title of a previously stored calculation to restart from.

Run a single point calculation on the current chemical system and return a “results” object if successful. If the calculation fails with a runtime error, a “results” object MAY still be returned (possibly with just some of the requested properties).

All Boolean fields in “request” default to false if not present. All non-Boolean fields in “request” except for “title” are optional and their default values are worker-dependent. The master SHOULD NOT explicitly set any Boolean fields in “request” to False. The worker MAY raise an unknown argument error if an unknown Boolean is set to False. The worker SHOULD raise an unknown argument error as usual if an unknown Boolean is set to True or if an unknown non-Boolean is set. Workers SHOULD raise these errors before performing any time-consuming calculations so that the master can efficiently retry the call.

If “keepResults” is not specified or set to false, the worker will discard all data from the calculation after returning a “results” object. If “keepResults” is set to true, the worker will remember any internal state related to the calculation. This internal state can later be reused for restart by passing the “title” of the stored calculation as “prevTitle”. The pipe master SHOULD call DeleteResults to discard the stored state as soon as it is no longer needed.

A “results” object consists of the following fields. Workers MAY include additional fields not listed here. A master MUST NOT signal an error due to any fields it does not expect or understand.

- **“results” (object):**
 - “messages” (string(:)): Runtime error or warning messages generated by the calculation.
 - “energy” (real):
 - “gradients” (real(3,:)):
 - “stressTensor” (real(:,,:)):
 - “elasticTensor” (real(:,,:)):
 - “hessian” (real(:,,:)):
 - “dipoleMoment” (real(:,,:)):
 - “dipoleGradients” (real(:,,:)):
 - “charges” (real(:)):

DeleteResults(title)

- “title” (string): Title of a previously remembered calculation.

Discard any worker state corresponding to a previously remembered calculation.

Forward and backward compatibility considerations

The AMSPipe protocol is designed to support combining new masters with old workers and vice versa. Major, incompatible changes in the protocol will be handled by increasing the protocol version number. Negotiating a suitable protocol version is then handled by call(s) to Hello at the beginning of a pipe session.

The following requirements are in place to ensure that the protocol stays extensible within a single protocol version:

- Workers **MUST** raise an `unknown_method` error on any call to a method they don’t implement. If the method name starts with `Set`, such an error will be buffered to be returned later. If the method name doesn’t start with `Set`, a “return” message will be generated immediately.
- Workers **MUST** raise `unknown_argument` errors any time they encounter an argument to a known method that they don’t know how to handle. The master **SHOULD** then retry the call without such an argument or choose an alternative sequence of calls if possible.
- The master **SHOULD** ignore unexpected messages of unsupported type on the reply pipe.

Namely, the following changes are permitted without increasing the protocol version number:

- Adding new methods to the protocol.
- Adding new optional arguments to existing methods.
- Adding new fields to returned objects.
- Adding new reply message types.

2.5.2 AMS as a pipe master

The AMS driver can play the role of a pipe master, allowing users to combine the features of the AMS driver with potentials implemented in external programs. Unlike a *traditional external engine* (page 320), the overhead introduced by the pipe interface is entirely negligible, because the external program is only started once at the beginning of the run and all communication is handled by an efficient binary protocol instead of text files. This mode is enabled by using `Engine Pipe` in the input for the master.

```
Engine Pipe
  WorkerCommand /path/to/pipe/worker
EndEngine
```

WorkerCommand

Type
String

Description

The command to execute to run the external worker. The command is executed in a subdirectory of the results directory.

All calculations requested by the driver will then be forwarded over the pipe to the worker for processing.

Note: AMS currently must be run in serial (NSCM=1) when serving as a pipe master.

A Python module implementing the worker side of the AMSPipe protocol is available in `scm.amspipe`. To facilitate interfacing with various existing computational engines, this module provides the `ASEPipeWorker` class. This class can wrap any ASE calculator object and make it serve as a pipe worker.

```
calculator = ase.calculators.lj.LennardJones()
# calculator.parameters = ...

engine = scm.amspipe.ASPipeWorker(calculator=calculator)
engine.run()
```

See also:

Example: ASE calculator as a pipe worker (page 560)

2.5.3 AMS as a pipe worker

AMS can also serve as a pipe worker, allowing external drivers to take advantage of its *engines* (page 315). This mode is enabled by using `Task Pipe`. No *Geometry, System definition* (page 31) is required on the input because the system will be supplied by the pipe master.

Hint: For most users it will easiest to use this functionality through the new `AMSWorker` class in the `PLAMS` library. This class hides all the details of the underlying AMSPipe protocol and provides users an easy way to get very fast access to energies, gradients and other properties from any of the engines in the AMS driver. See the *respective page* in the PLAMS manual for details.

See also:

Example: AMS as a pipe worker (page 562)

Additionally AMS can also be driven through the `FlexMD` library using `AMSPipeForceJob`:

```
forcejob = AMSPipeForceJob(mdmol)
forcejob.settings.engine = 'ReaxFF'
forcejob.settings.engineSettings = { 'ForceField': 'Glycine.ff' }
```


GEOMETRY, SYSTEM DEFINITION

The definition of the system to simulate, i.e. the positions and types of the nuclei, the total charge, and potentially lattice vectors, is enclosed in the `System` block:

```
System header
  Atoms header # Non-standard block. See details.
  ...
End
  Lattice header # Non-standard block. See details.
  ...
End
  FractionalCoords Yes/No
  AllowCloseAtoms Yes/No
  GeometryFile string
  Symmetrize Yes/No
  LatticeStrain float_list
  SuperCell integer_list
  Charge float
  BondOrders # Non-standard block. See details.
  ...
End
End
```

3.1 Geometry, Lattice

The geometry of the system is specified with the `Atoms` and `Lattice` blocks.

System

Type

Block

Recurring

True

Description

Specification of the chemical system. For some applications more than one system may be present in the input. In this case, all systems except one must have a non-empty string ID specified after the `System` keyword. The system without an ID is considered the main one.

Atoms

Type

Non-standard block

Description

The atom types and coordinates. Unit can be specified in the header. Default unit is Angstrom.

Lattice**Type**

Non-standard block

Description

Up to three lattice vectors. Unit can be specified in the header. Default unit is Angstrom.

FractionalCoords**Type**

Bool

Default value

No

Description

Whether the atomic coordinates in the Atoms block are given in fractional coordinates of the lattice vectors. Requires the presence of the Lattice block.

The `Atoms` block contains one line per atoms, similar to the lines found in an `.xyz` file: First the name of the element, then three real numbers representing the coordinates of that atom in Angstrom. The following `Atoms` block shows how one would define a water molecule:

```
System
  Atoms
    O  0.0  0.0  0.59372
    H  0.0  0.76544 -0.00836
    H  0.0 -0.76544 -0.00836
  End
End
```

Note that it is possible to specify a different unit of length in the header of the block (that is in the line after the keyword opening the block) by putting the name of the unit in [and] brackets. So the same water molecule could also be specified as follows:

```
System
  Atoms [Bohr]
    O  0.0  0.0  1.12197
    H  0.0  1.44647 -0.01580
    H  0.0 -1.44647 -0.01580
  End
End
```

It is also possible to specify the input geometry as a Z-Matrix, by putting the string `Z-Matrix` in the header of the block:

```
System
  Atoms Z-Matrix
    C
    H  1 1.089000
    H  1 1.089000 2 109.4710
    H  1 1.089000 2 109.4710 3 120.0000
    H  1 1.089000 2 109.4710 3 -120.0000
  End
End
```

Periodic systems require the specification of 1 (for chains), 2 (for slabs) or 3 (for bulk) lattice vectors in addition to the nuclear coordinates. Every lattice vector is specified on a separate line of three numbers, representing the vectors x,y and

z-component. Note that for chain systems, the single lattice vector **must** point along the x-axis, while for slab systems the two lattice vectors **must** be in the xy-plane. Consider the following input for graphene:

```
System
  Atoms
    C  0.0  0.0  0.0
    C  1.23  0.71014  0.0
  End
  Lattice
    2.46  0.0  0.0
    1.23  2.13042  0.0
  End
End
```

As with the `Atoms` block, the length unit in which the lattice vectors are given can be changed by specifying the desired unit in the header of the block (enclosed in `[]`). It is also possible to define a system given the fractional coordinates of the atoms using the `FractionalCoords` keyword. The numbers in the `Atoms` block are then interpreted as fractional coordinates according to the lattice vectors in the `Lattice` block. Note that for chain and slab systems, the coordinates perpendicular to the periodic direction (z and y for chains, z for slabs) are of course still in Angstrom (or alternatively the unit set in the header of the `Atoms` block). Using the `FractionalCoords` keyword we could specify the geometry of table salt (NaCl) as follows:

```
System
  Lattice
    0.0  2.75  2.75
    2.75  0.0  2.75
    2.75  2.75  0.0
  End
  FractionalCoords True
  Atoms
    Na  0.0  0.0  0.0
    Cl  0.5  0.5  0.5
  End
End
```

Instead of specifying the geometry of the system directly in the input file it can also be read from an external file.

System

GeometryFile

Type

String

Description

Read the geometry from a file (instead of from `Atoms` and `Lattice` blocks). Supported formats:
.xyz

Note that the `GeometryFile` key replaces both the `Atoms` and the `Lattice` blocks in the input. So if you specify the `GeometryFile` keyword in the input, the `Atoms` and `Lattice` blocks must not appear there. At the moment only the *extended XYZ file format* (page 565) is supported.

3.1.1 Modifying the geometry

Finally there are a number of keywords that *modify* the system geometry:

System

Symmetrize

Type

Bool

Default value

No

Description

Whether to symmetrize the input structure. This might also rototranslate the structure into a standard orientation. This will symmetrize the atomic coordinates to machine precision. Useful if the system is almost symmetric or to rototranslate a symmetric molecule into a standard orientation.

LatticeStrain

Type

Float List

Description

Deform the input system by the specified strain. The strain elements are in Voigt notation, so one should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

SuperCell

Type

Integer List

Description

Create a supercell of the input system (only possible for periodic systems). The integer numbers represent the diagonal elements of the supercell transformation; you should specify as many numbers as lattice vectors (i.e. 1 number for 1D, 2 numbers for 2D and 3 numbers for 3D periodic systems).

SuperCellTrafo

Type

Integer List

Description

Create a supercell of the input system (only possible for periodic systems) $\vec{a}'_i = \sum_j T_{ij} \vec{a}_j$. The integer numbers represent the supercell transformation T_{ij} : 1 number for 1D PBC, 4 numbers for 2D PBC corresponding to a 2x2 matrix (order: (1,1),(1,2),(2,1),(2,2)) and 9 numbers for 3D PBC corresponding to a 3x3 matrix (order: (1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)).

PerturbCoordinates

Type

Float

Default value

0.0

Unit

Angstrom

Description

Perturb the atomic coordinates by adding random numbers between [-PerturbCoordinates,PerturbCoordinates] to each Cartesian component. This can be useful if you want to break the symmetry of your system (e.g. for a geometry optimization).

PerturbLattice**Type**

Float

Default value

0.0

Description

Perturb the lattice vectors by applying random strain with matrix elements between [-PerturbLattice,PerturbLattice]. This can be useful if you want to deviate from an ideal symmetric geometry, for example if you look for a phase change due to high pressure.

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

For periodic systems the atoms will be moved to the central cell.

These modifications are applied immediately after the system block is read. To the rest of AMS (and the input) it looks exactly as if the modified system was specified explicitly in the `System` block input. That means that the `SuperCell` keyword is not easily usable with input options that require the specification of atom indices, e.g. the [constraints](#) (page 55) block. Note that the randomization of the coordinates is applied after a potential supercell creation.

3.2 Symmetry

Symmetry can be used in AMS optimizations of molecules and of periodic structures.

In case of molecules at the start of an AMS calculation one can symmetrize an almost symmetric structure. In the [Appendix Symmetry](#) (page 570) one can find molecular orientation requirements that AMS needs such that AMS can use the (sub)symmetry of a molecule. If the system is symmetrized (and no symmetry is given in the `System` block key) the molecular structure is rotranslated into this standard orientation. In the [Appendix Symmetry](#) (page 568) one can also find Schönflies symbols for molecular point groups and symmetry labels that are used in AMS for molecules to label normal modes.

```
System header
  Symmetrize Yes/No
  Symmetry [...]
End
```

```
Symmetry
  SymmetrizeTolerance float
  Tolerance float
End
```

```
UseSymmetry Yes/No
```

System

Symmetrize**Type**

Bool

Default value

No

Description

Whether to symmetrize the input structure. This might also rototranslate the structure into a standard orientation. This will symmetrize the atomic coordinates to machine precision. Useful if the system is almost symmetric or to rototranslate a symmetric molecule into a standard orientation.

Symmetry**Type**

Multiple Choice

Default value

AUTO

Options

[AUTO, NOSYM, C(LIN), D(LIN), C(I), C(S), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(2V), C(3V), C(4V), C(5V), C(6V), C(7V), C(8V), C(2H), C(3H), C(4H), C(5H), C(6H), C(7H), C(8H), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(2D), D(3D), D(4D), D(5D), D(6D), D(7D), D(8D), D(2H), D(3H), D(4H), D(5H), D(6H), D(7H), D(8H), I, I(H), O, O(H), T, T(D), T(H), S(4), S(6), S(8)]

Description

Use (sub)symmetry with this Schoenflies symbol. Can only be used for molecules. Orientation should be correct for the (sub)symmetry. If used icw Symmetrize, the symmetrization will not reorient the molecule.

Symmetry**SymmetrizeTolerance****Type**

Float

Default value

0.05

Description

Tolerance used to detect symmetry in case symmetrize is requested.

Tolerance**Type**

Float

Default value

1e-07

Description

Tolerance used to detect symmetry in the system.

UseSymmetry**Type**

Bool

Default value

Yes

Description

Whether to use the system's symmetry in AMS. Symmetry is recognized within a tolerance as given in the Symmetry key.

3.3 Regions

Some options of the AMS driver and its engines require specifying a subset of a system's atoms. For example one might want to freeze part of the system in a molecular dynamics calculation. We refer to these subsets of atoms as "regions". Atoms are assigned to regions by specifying the region names behind the atomic coordinates in the `Atoms` block of the input:

```
System
  Atoms
    C      1.22110608      -0.00000000      1.65928963      region=ring_1
    C      0.37734253      -1.16134090      1.65928963      region=ring_1
    C      0.37734253      1.16134090      1.65928963      region=ring_1
    C     -0.98789557      -0.71774815      1.65928963      region=ring_1
    C     -0.98789557      0.71774815      1.65928963      region=ring_1
    H      2.30849293      -0.00000000      1.64874914      region=ring_1
    H      0.71336355      -2.19550724      1.64874914      region=ring_1
    H      0.71336355      2.19550724      1.64874914      region=ring_1
    H     -1.86761001      -1.35689810      1.64874914      region=ring_1
    H     -1.86761001      1.35689810      1.64874914      region=ring_1
    Fe      0.00000000      0.00000000      0.00000000
    H      0.71336355      2.19550724     -1.64874914      region=ring_2
    H      0.71336355      -2.19550724     -1.64874914      region=ring_2
    C      1.22110608      -0.00000000     -1.65928963      region=ring_2
    C      0.37734253      1.16134090     -1.65928963      region=ring_2
    C      0.37734253     -1.16134090     -1.65928963      region=ring_2
    C     -0.98789557      0.71774815     -1.65928963      region=ring_2
    C     -0.98789557     -0.71774815     -1.65928963      region=ring_2
    H      2.30849293     -0.00000000     -1.64874914      region=ring_2
    H     -1.86761001      1.35689810     -1.64874914      region=ring_2
    H     -1.86761001     -1.35689810     -1.64874914      region=ring_2
  End
End
```

In the above example of a ferrocene molecule, we have created two separate regions for the two cyclopentadienyl rings. The region names (`ring_1` and `ring_2` in the example above) can be freely chosen by the user. Note that an atom can be in more than one region, in which case the region names should be separated by commas:

```
System
  Atoms
    ...
    Mg  0.0  0.0  0.0      region=metal_centers,mol_1
    ...
  End
End
```

Technically the region is handled as an *atomic attribute* (page 42).

3.4 Charge, atomic masses, input bond orders

AMS allows to set user-defined masses for particular atoms. This can be used to simulate isotopes of different atoms. Masses are specified by adding the desired mass (in Dalton) at the end of the atom's line. The following input shows the system specification for a heavy water molecule:

```
System
  Atoms
    O   0.0   0.0           0.59372
    H   0.0   0.76544   -0.00836   mass=2.014
    H   0.0  -0.76544   -0.00836   mass=2.014
  End
End
```

(Observe that the mass specified this way is an example of the *atomic attributes* (page 42) system.)

Finally the `System` block also contains the specification of the system's total charge as well as optionally defined bond orders.

System

Charge

Type

Float

Default value

0.0

GUI name

Total charge

Description

The system's total charge in atomic units.

BondOrders

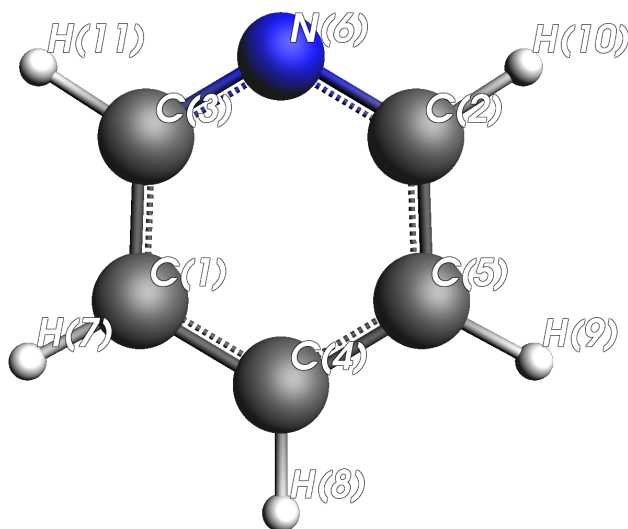
Type

Non-standard block

Description

Defined bond orders. Each line should contain two atom indices, followed by the bond order (1, 1.5, 2, 3 for single, aromatic, double and triple bonds) and (optionally) the cell shifts for periodic systems. May be used by MM engines and for defining constraints. If the system is periodic and none of the bonds have the cell shift defined then AMS will attempt to determine them following the minimum image convention.

The following example shows the `System` block with defined bond orders for pyridine:



```

System
Atoms
  C  0.0  -1.19129014  -0.47777633
  C  0.0   1.1420834   0.90885782
  C  0.0  -1.1420834   0.90885782
  C  0.0   0.0         -1.18887601
  C  0.0   1.19129014  -0.47777633
  N  0.0   0.0         1.5894377
  H  0.0  -2.14522862  -0.99205897
  H  0.0   0.0        -2.27338592
  H  0.0   2.14522862  -0.99205897
  H  0.0   2.05492975   1.4973896
  H  0.0  -2.05492975   1.4973896
End
BondOrders
  1 3  1.5
  1 4  1.5
  1 7  1.0
  2 5  1.5
  2 6  1.5
  2 10 1.0
  3 6  1.5
  3 11 1.0
  4 5  1.5
  4 8  1.0
  5 9  1.0
End
End

```

Note that the specified bond orders are currently only used by the ForceField engine and for generating distance constraints via the `All bonds` option in the constraints block.

3.5 Homogeneous electric field and multipole charges

A homogeneous electric field and multipole charges can be requested at the AMS level. Currently this option is supported by the engines ADF, BAND, DFTB, MOPAC, and ReaxFF.

Homogeneous electric field:

```
System
  ElectrostaticEmbedding
    ElectricField ex ey ez
  End
End
```

ElectrostaticEmbedding

ElectricField

Type

Float List

Unit

V/Angstrom

Description

External homogeneous electric field with three Cartesian components: ex, ey, ez, the default unit being V/Å.

In atomic units: Hartree/(e bohr) = 51.422 V/Angstrom; the relation to SI units is: 1 Hartree/(e bohr) = 5.14 ... e11 V/m.

Supported by the engines adf, band, dftb and mopac.

For periodic systems the field may only have nonzero components orthogonal to the direction(s) of periodicity (i.e. for 1D periodic system the x-component of the electric field should be zero, while for 2D periodic systems both the x and y components should be zero. This options cannot be used for 3D periodic systems.

Point and multipole charges:

```
System
  ElectrostaticEmbedding
    MultipolePotential
      Coordinates
        x y z q py pz px
        x y z q py pz px
        ...
      End
    End
  End
End
```

ElectrostaticEmbedding

MultipolePotential

Type

Block

Description

External point charges (and dipoles).

ChargeModel**Type**

Multiple Choice

Default value

Point

Options

[Point, Gaussian]

Description

A multipole may be represented by a point (with a singular potential at its location) or by a spherical Gaussian distribution.

ChargeWidth**Type**

Float

Default value

-1.0

Description

The width parameter in a.u. in case a Gaussian charge model is chosen. A negative value means that the width will be chosen automatically.

Coordinates**Type**

Non-standard block

Description

Positions and values of the multipoles, one per line. Each line has the following format:

x y z q, or

x y z q μ_x μ_y μ_z .

Here x, y, z are the coordinates in Å, q is the charge (in atomic units of charge) and μ_x , μ_y , μ_z are the (optional) dipole moment components (in atomic units, i.e. e*Bohr).

Periodic systems are not supported.

Note: When running geometry optimizations (or other similar tasks) in combination with point charges, you should be aware that the system might end up on top of the point charge(s), resulting in non-physical situations. You should consider using *constraints* (page 55), *restraints* (page 332) or explicitly setting *rigid motions* (page 740) options.

3.6 Load a System from file

Instead of specifying the system to simulate in the `System` block of the input, it is also possible to read the system used in a previous calculation from the binary `.rkf` result files of AMS. This is done with the `LoadSystem` block in the input:

```
LoadSystem header
  File string
  Section string
End
```

LoadSystem**Type**

Block

Recurring

True

Description

Block that controls reading the chemical system from a KF file instead of the [System] block.

File**Type**

String

Description

The path of the KF file from which to load the system. It may also be the results directory containing it.

Section**Type**

String

Default value

Molecule

Description

The section on the KF file from which to load the system.

Note that the `LoadSystem` block is mutually exclusive with the `System` block: The system *either* needs to be specified in the input, *or* loaded from a previous results file.

Any `.rkf` file written by AMS should be suitable to load a system from. For *engine output files* (page 19) the loaded geometry is just the one for which the engine was invoked when it wrote this file. For the *main result file* (page 19) `ams.rkf` written by the AMS driver, which geometry is loaded depends on the *task* (page 47) that AMS was performing when this file was written. Generally the `ams.rkf` file contains two systems:

- The input system corresponding just to the `System` block that was read in by AMS. This system is written to the `InputMolecule` section on the `ams.rkf`, and can be loaded from there using the `LoadSystem%Section` keyword. This can be useful in order to repeat a previous AMS calculation for the same system, but with different settings, e.g. a different engine.
- The system which was the result of a previous AMS calculation, e.g. a geometry optimization or transition state search. This system is written to the `Molecule` section on the `ams.rkf`. What exactly is considered the resulting geometry of a calculation depends in the *task* (page 47) of the previous calculation. (For tasks that do not change the geometry (like a single point calculation) or where no configuration is particularly special (e.g. a PES scan), the result system is normally just the same as the input system.)

3.7 Atom attributes

There is in general the possibility to add text after the coordinates of an atom, we call these atomic attributes. The text should be of the form `key=value`, separated by spaces:

```
System
Atoms
O   0.0   0.0       0.59372   key1=1 key2=y
H   0.0   0.76544  -0.00836   file=ams.rkf
```

(continues on next page)

(continued from previous page)

```

      H   0.0  -0.76544  -0.00836
    End
  End

```

You can enter any arbitrary key/value list, but that will have no effect in general. Although there is one important effect that atoms with different atomic attributes (even unrecognized ones) are considered different elements during the symmetry detection.

The order in which keys are specified has no effect.

Examples of atomic attributes are the region, and mass.

3.8 Force field related extensions

For forcefield there are two dedicated *atomic attributes* (page 42), the ForceField.Type and ForceField.Charge. Obviously these attributes are ignored by all other engines, although it can reduce the symmetry of the system:

```

System
  Atoms
    O   0.0   0.0           0.59372  ForceField.Type=O ForceField.Charge=-0.1
    H   0.0   0.76544  -0.00836  ForceField.Type=H ForceField.Charge=+0.05
    H   0.0  -0.76544  -0.00836  ForceField.Type=H ForceField.Charge=+0.05
  End
End

```

3.8.1 Load charges for a forcefield into regions

When you want to use a forcefield, the charges obtained by a previous calculation can be loaded into a specific region (with the same molecule). Currently the order of the atoms has to be the same for this to work.

Once loaded, it is as if the user had typed in the ForceField.Charge attributes.

System

LoadForceFieldCharges

Type

Block

Recurring

True

Description

This is a mechanism to set the ForceField.Charge attribute in the input. This information is currently only used by the ForceField engine.

CheckGeometryRMSD

Type

Bool

Default value

No

Description

Whether the geometry RMSD test should be performed, see MaxGeometryRMSD. Otherwise only basic tests are performed, such as number and atom types. Not doing the RMSD test allows you to load molecular charges in a periodic system.

File**Type**

String

Description

Name of the (kf) file

MaxGeometryRMSD**Type**

Float

Default value

0.1

Unit

Angstrom

Description

The geometry of the charge producing calculation is compared to the one of the region, and need to be the same within this tolerance.

Region**Type**

String

Default value

*

Description

Region for which the charges should be loaded

Section**Type**

String

Default value

AMSResults

Description

Section name of the kf file

Variable**Type**

String

Default value

Charges

Description

Variable name of the kf file

3.8.2 Load forcefield atom types

After you have done a forcefield calculation the atom types are stored. You can load those for a next calculation.

Once loaded, it is as if the user had typed in the ForceField.Type atom attributes.

System

LoadForceFieldAtomTypes

Type

Block

Description

This is a mechanism to set the ForceField.Type attribute in the input. This information is currently only used by the ForceField engine.

File

Type

String

Description

Name of the (kf) file. It needs to be the result of a forcefield calculation.

STRUCTURE AND REACTIVITY, MOLECULAR DYNAMICS

4.1 Single point calculations

A single point calculation is the simplest task available in the AMS driver. It simply runs the *engine* (page 315) once for the given geometry. In other words, the AMS driver does not explore the potential energy surface (PES), but simply samples a “single point” of it.

A single point calculation is performed by selecting it with the `Task` keyword:

```
Task SinglePoint
```

Note that a single point calculation in AMS includes the calculation of *PES point properties* (page 235). Many of these, such as the nuclear gradients and the Hessian, are derivatives at this PES point with respect to nuclear displacements. These derivatives might be done numerically by the AMS driver, in which case it would technically run the engine multiple times and sample PES points around the initial point. However, in AMS this is still considered a single point calculation. Take for example the calculation of the normal modes of vibration of a molecule. This used to be a separate task in the 2017 release of the DFTB program, but in AMS is just a single point calculation with a request for normal modes:

```
Task SinglePoint

Properties
  NormalModes True
End
```

See the manual section on *PES point properties* (page 235) for an overview of which properties can be calculated with the `SinglePoint` task in AMS.

4.2 Bond energy calculations

The way to calculate bonding energies is always the same, regardless of the engine. It is about combining ground state energies for several systems.

4.2.1 Ground state energy

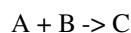
Let system A have a ground state energy $E(A)$

The ground state energy of a system is obtained by a full relaxation with respect to the geometry coordinates and (if relevant) to the electronic degrees of freedom. In case of periodic materials the lattice vectors need relaxation as well.

Electronic degrees of freedom are specific for the method underpinning the engine. A simple force field may have a charge equilibration scheme, whereas more advanced engines such DFT- and DFTB-based ones have orbitals. The electronic relaxation can become a non-trivial problem in case of *open shell* (page 49) systems.

4.2.2 Formation energy

Say we have a reaction of two molecules (A and B) forming a new one (C).



The interaction energy follows from three *ground state* (page 48) calculations

$$E(\text{bond}) = E(C) - E(A) - E(B)$$

Let us look at a slightly more complicated reaction. The (metastable) material Octanitrocubane with the formula $C_8(NO_2)_8$ can be formed from 8 CO_2 molecules and four N_2 molecules. The formation energy is

$$E(\text{bond}) = E(\text{Octanitrocubane}) - 8 E(CO_2) - 4 E(N_2)$$

the result being positive as this reaction is highly endothermic.

4.2.3 Atomization energies

Let us now look at atomization energies. For instance, the cohesive energy the NaCl crystal is

$$E(\text{cohesive energy}) = E(\text{NaCl-crystal}) - E(\text{Na-atom}) - E(\text{Cl-atom})$$

(Atomization energies can also be calculated for molecules.)

For engines with electronic degrees of freedom, the tricky part here is how to calculate the atomic energies, in particular $E(\text{Cl-atom})$, because they are open shell systems. See the notes on a *atomic corrections* (page 49).

Of course you need to weigh the atomic energies by how often they occur in the system (molecule or crystal).

4.2.4 Chemisorption energies

What is the adsorption energy of a molecule on a surface?

The main thing here is whether the “real” system has translational symmetry. For instance in an experiment this may be the case and correspond to a certain coverage. This can be perfectly modeled in a periodic calculation

$$E(\text{chemisorption}) = E(\text{mol@surface}) - E(\text{mol}) - E(\text{bare surface})$$

You need to choose the right super cell to get the correct coverage (how many adsorbed atoms per unit cell).

If the experiment is about the adsorption of a single molecule you need to try to converge the result with progressively larger super cells.

The other main issue is that almost always in the experimental setup the surface is macroscopically thick. Therefore the slab thickness is an issue to be tested. Fortunately, often less than 10 layers are needed.

4.2.5 Atomic corrections

Sometimes we need to calculate the ground state energy of a single atom. As there are no geometric variables, the only degrees of freedom are electronic. In general atoms are *open shell* (page 49) systems. The idea is to let the engine run without any restrictions imposed on the spin-polarization, or the symmetry of the orbitals.

One possible trick is to run two atoms quite far away (10-20 angstrom), thus reducing the exact spherical symmetry.

Calculating atomic corrections can be very tricky, as there may be many nearly degenerate orbitals near the fermi level. Also SCF convergence can be very difficult. There is not a single solution for all atoms.

4.2.6 Open shell systems

The idea is to let the engine run without any restrictions imposed on the spin-polarization, or the symmetry of the orbitals.

Calculating open shell systems can be very tricky, as there may be many close lying states (of which the lowest is searched for). Also SCF convergence can be very difficult.

From a theoretical point of view these states should ultimately be described by multi-determinantal wave functions, lacking from methods such as DFT.

There may be no automatic way to find the absolute minimum. If possible try to avoid the need to do such calculations.

4.2.7 Impurities

Let's say that we want to introduce a single impurity in a crystal. Currently we do not support such a method. But using periodic boundary conditions (PBCs) we can approximate this.

Say we want to introduce a single Li atom in an Al crystal. This can be done by considering progressively larger super cells.

First we make a 2x2x2 super cell, with 8 Al atoms, then we insert one Li, thus modeling an "impurity", with a 1/8 Li/Al ratio, rather than a 1/infinity. With bigger super cells this approximation becomes better.

The insertion energy is then

$$E(8\text{Al}+\text{Li}) - E(8\text{Al}) - E(\text{Li-atom})$$

See also the comments for Atomization energies.

4.3 Geometry optimization

A geometry optimization is the process of changing the system's geometry (the nuclear coordinates and potentially the lattice vectors) to minimize the total energy of the systems. This is typically a local optimization, i.e. the optimization converges to the next local minimum on the potential energy surface (PES), given the initial system geometry specified in the `System` block. In other words: The geometry optimizer moves "downhill" on the PES into the local minimum.

See also:

[Examples](#) (page 457) and [silicon lattice optimization and phonons tutorial](#)

Geometry optimizations are performed by selecting them as the `Task`. The details of the optimization can be configured in the corresponding block:

Task GeometryOptimization

```

GeometryOptimization
  Convergence
    Energy float
    Gradients float
    Quality [VeryBasic | Basic | Normal | Good | VeryGood | Custom]
    Step float
    StressEnergyPerAtom float
  End
  MaxIterations integer
  CalcPropertiesOnlyIfConverged Yes/No
  OptimizeLattice Yes/No
  KeepIntermediateResults Yes/No
  PretendConverged Yes/No
  MaxRestarts integer
  RestartDisplacement float
End

```

4.3.1 Convergence criteria

GeometryOptimization

Type

Block

Description

Configures details of the geometry optimization and transition state searches.

Convergence

Type

Block

Description

Convergence is monitored for up to 4 quantities: the energy change, the Cartesian gradients, the Cartesian step size, and for lattice optimizations the stress energy per atom. Convergence criteria can be specified separately for each of these items.

Energy

Type

Float

Default value

1e-05

Unit

Hartree

Value Range

value > 0

GUI name

Energy convergence

Description

The criterion for changes in the energy. The energy is considered converged when the change in energy is smaller than this threshold times the number of atoms.

Gradients

Type

Float

Default value

0.001

Unit

Hartree/Angstrom

Value Range

value > 0

GUI name

Gradient convergence

Description

Threshold for nuclear gradients.

Quality**Type**

Multiple Choice

Default value

Custom

Options

[VeryBasic, Basic, Normal, Good, VeryGood, Custom]

GUI name

Convergence

Description

A quick way to change convergence thresholds: 'Good' will reduce all thresholds by an order of magnitude from their default value. 'VeryGood' will tighten them by two orders of magnitude. 'Basic' and 'VeryBasic' will increase the thresholds by one or two orders of magnitude respectively.

Step**Type**

Float

Default value

0.01

Unit

Angstrom

Value Range

value > 0

GUI name

Step convergence

Description

The maximum Cartesian step allowed for a converged geometry.

StressEnergyPerAtom**Type**

Float

Default value

0.0005

Unit

Hartree

Value Range

value > 0

Description

Threshold used when optimizing the lattice vectors. The stress is considered 'converged' when the maximum value of $\text{stress_tensor} * \text{cell_volume} / \text{number_of_atoms}$ is smaller than this threshold (for 2D and 1D systems, the cell_volume is replaced by the cell_area and cell_length respectively).

A geometry optimization is considered converged when all the following criteria are met:

1. The difference between the bond energy at the current geometry and at the previous geometry step is smaller than $\text{Convergence\%Energy}$ times the number of atoms in the system.
2. The maximum Cartesian nuclear gradient is smaller than $\text{Convergence\%Gradient}$.
3. The root mean square (RMS) of the Cartesian nuclear gradients is smaller than $2/3 \text{ Convergence\%Gradient}$.
4. The maximum Cartesian step is smaller than Convergence\%Step .
5. The root mean square (RMS) of the Cartesian steps is smaller than $2/3 \text{ Convergence\%Step}$.

Note: If the maximum and RMS gradients are 10 times smaller than the convergence criterion, then criteria 4 and 5 are ignored.

The $\text{Convergence\%Quality}$ sets the criteria as follows:

Quality	Energy (Ha)	Gradients (Ha/Å)	Step (Å)	StressEnergyPerAtom (Ha)
VeryBasic	10^{-3}	10^{-1}	1	5×10^{-2}
Basic	10^{-4}	10^{-2}	0.1	5×10^{-3}
Normal	10^{-5}	10^{-3}	0.01	5×10^{-4}
Good	10^{-6}	10^{-4}	0.001	5×10^{-5}
VeryGood	10^{-7}	10^{-5}	0.0001	5×10^{-6}

Some remarks on the choice of the convergence thresholds:

- Molecules may differ very much in the stiffness around the energy minimum. Using the standard convergence thresholds without second thought is therefore not recommended. Strict criteria may require a large number of steps, while a loose threshold may yield geometries that are far from the minimum (with respect to atom-atom distances, bond-angles etc...) even when the total energy of the molecule might be very close to the value at the minimum. It is good practice to consider first what the objectives of the calculation are. The default settings in AMS are intended to be reasonable for most applications, but inevitably situations may arise where they are inadequate.
- The convergence threshold for the coordinates (Convergence\%Step) is not a reliable measure for the precision of the final coordinates. Usually it yields a reasonable estimate (order of magnitude), but to get accurate results one should tighten the criterion on the gradients, rather than on the steps (coordinates). (The reason for this is that with the Quasi-Newton based optimizers the estimated uncertainty in the coordinates is related to the used Hessian, which is updated during the optimization. Quite often it stays rather far from an accurate representation of the true Hessian. This does usually not prevent the program from converging nicely, but it does imply a possibly incorrect calculation of the uncertainty in the coordinates.)

- Note that tight convergence criteria for the geometry optimization require accurate and noise-free gradients from the engine. For some engines this might mean that their numerical accuracy has to be increased for geometry optimization with tight convergence criteria, see e.g. the `NumericalQuality` keyword in the BAND manual.

The maximum number of geometry iterations allowed to locate the desired structure is specified with the `MaxIterations` keyword:

GeometryOptimization

MaxIterations

Type

Integer

Value Range

value ≥ 0

GUI name

Maximum number of iterations

Description

The maximum number of geometry iterations allowed to converge to the desired structure.

CalcPropertiesOnlyIfConverged

Type

Bool

Default value

Yes

Description

Compute the properties requested in the 'Properties' block, e.g. Frequencies or Phonons, only if the optimization (or transition state search) converged. If False, the properties will be computed even if the optimization did not converge.

PretendConverged

Type

Bool

Default value

No

Description

Normally a non-converged geometry optimization is considered an error. If this keyword is set to True, the optimizer will only produce a warning and still claim that the optimization is converged. (This is mostly useful for scripting applications, where one might want to consider non-converged optimizations still successful jobs.)

If the geometry optimization does not converge within this many steps it is considered failed and the iteration aborted, i.e. *PES point properties* (page 235) block will not be calculated at the last geometry. The default maximum number of steps is chosen automatically based on the used optimizer and the number of degrees of freedom to be optimized. The default is a fairly large number already, so if the geometry has not converged (at least to a reasonable extent) within that many iterations you should step back and consider the underlying cause rather than simply increase the allowed number of iterations and try again.

4.3.2 Automatic restarts

While a geometry optimization aims to find a (local) PES minimum, it may occur that it ends up finding a saddle point instead. The `PESPointCharacter` property keyword can be used to quickly calculate the lowest few Hessian eigenvalues to determine what kind of stationary PES point the optimization found. More information on this feature can be found on its [Documentation Page](#) (page 239). Since AMS2022.1, geometry optimizations with enabled PES point characterization can automatically restart when a transition state (or higher order saddle point) is found: the geometry is distorted along the lowest frequency mode and the optimizer run again. The applied distortion is often symmetry breaking in this case, so this automatic restarting is only enabled if the system does not have any symmetry operators or the use of symmetry is explicitly disabled using the `UseSymmetry` keyword. Furthermore the automatic restarting must be explicitly enabled by setting the `MaxRestarts` option to a value >0. Of course the PES point characterization needs to be enabled too:

```
GeometryOptimization
  MaxRestarts 5
End

UseSymmetry False

Properties
  PESPointCharacter True
End
```

Details of the automatic restarting can configured with the following keywords:

GeometryOptimization

MaxRestarts

Type

Integer

Default value

0

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small displacement along the imaginary vibrational mode. In case the restarted optimization again does not find a minimum, this can happen multiple times in succession. This keyword sets the maximum number of restarts. The default value is 0, so the automatic restarting is disabled by default.

RestartDisplacement

Type

Float

Default value

0.05

Unit

Angstrom

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small

displacement along the imaginary vibrational mode. This keywords sets the size of the displacement for the furthest moving atom.

4.3.3 Lattice optimization

For periodic systems the lattice degrees of freedom can be optimized in addition to the nuclear positions.

GeometryOptimization

OptimizeLattice

Type

Bool

Default value

No

Description

Whether to also optimize the lattice for periodic structures. This is currently supported with the Quasi-Newton, FIRE, and L-BFGS optimizers.

See also *Constrained optimization* (page 55) for constrained lattice optimizations.

4.3.4 Keep all results files

The `GeometryOptimization` block also contains some technical options:

GeometryOptimization

KeepIntermediateResults

Type

Bool

Default value

No

Description

Whether the full engine result files of all intermediate steps are stored on disk. By default only the last step is kept, and only if the geometry optimization converged. This can easily lead to huge amounts of data being stored on disk, but it can sometimes be convenient to closely monitor a tricky optimization, e.g. excited state optimizations going through conical intersections, etc.
...

4.3.5 Constrained optimization

The AMS driver also allows to perform constrained optimizations, where a number of specified degrees of freedom are fixed to particular values.

See also:

Example demonstrating all supported constraints (page 465)

The desired constraints are specified in the `Constraints` block at the root level of the AMS input file:

```

Constraints
  Atom integer
  AtomList integer_list
  FixedRegion string
  Coordinate integer [x|y|z] float?
  Distance (integer){2} float
  All ... [Bonds|Triangles] ...
  Angle (integer){3} float
  Dihedral (integer){4} float
  SumDist (integer){4} float
  DifDist (integer){4} float
  BlockAtoms integer_list
  Block string
  FreezeStrain [xx] [xy] [xz] [yy] [yz] [zz]
  EqualStrain [xx] [xy] [xz] [yy] [yz] [zz]
End

```

Constraints

Type

Block

Description

The Constraints block allows geometry optimizations and potential energy surface scans with constraints. The constraints do not have to be satisfied at the start of the calculation.

The different types constraints are described with examples below.

Note that in principle an arbitrary number of constraints can be specified and thus combined. However, it is the user's responsibility to ensure that the specified constraints are actually *compatible with each other*, meaning that it is theoretically possible to satisfy all of them at the same time. The AMS driver does **not** detect this kind of problems, but the optimization will show very unexpected results. Furthermore, for calculations involving block constraints the following restrictions apply:

- There should be no other constrained coordinates used together with block constraints although this may work in many situation.
- The user should absolutely avoid specifying other constraints that include atoms of a frozen block.

Note that a constraint optimization may break symmetry, even if the constraint(s) themselves would not break symmetry.

Fixed atoms

The following constraints can be used to fix atoms in space.

Atom atomIdx

Fix the atom with index `atomIdx` at the initial position, as given in the `System%Atoms` block.

AtomList [atomIdx1 .. atomIdxN]

Fix all atoms in the list at the initial position, as given in the `System%Atoms` block.

FixedRegion regionName

Fix all atoms in a *region* (page 37) to their initial positions.

Coordinate atomIdx [x|y|z] coordValue?

Constrain the atom with index `atomIdx` (following the order in the `System%Atoms` block) to have a cartesian coordinate (x, y or z) of `coordValue` (given in Angstrom). If the `coordValue` is missing, the coordinate will be fixed to its initial value.

Internal degrees of freedom

The following options allow constraints of internal degrees of freedom, such as distances, angles and dihedral angles.

Distance atomIdx1 atomIdx2 distValue

Constrain the distance between the atoms with index atomIdx1 and atomIdx2 (following the order in the System%Atoms block) to distValue, given in Angstrom.

Angle atomIdx1 atomIdx2 atomIdx3 angleValue

Constrain the angle (1)–(2)–(3) between the atoms with indices atomIdx1–3 (as given by their order in the System%Atoms block) to angleValue, given in degrees.

Dihedral atomIdx1 atomIdx2 atomIdx3 atomIdx4 dihedValue

Constrain the dihedral angle (1)–(2)–(3)–(4) between the atoms with indices atomIdx1–4 (as given by their order in the System%Atoms block) to dihedValue, given in degrees.

SumDist atomIdx1 atomIdx2 atomIdx3 atomIdx4 sumDistValue

Constrain the sum of the distances $R(1,2)+R(3,4)$ between the atoms with indices atomIdx1–4 (as given by their order in the System%Atoms block) to sumDistValue, given in Angstrom.

DifDist atomIdx1 atomIdx2 atomIdx3 atomIdx4 difDistValue

Constrain the difference between the distances $R(1,2)-R(3,4)$ of the atoms with indices atomIdx1–4 (as given by their order in the System%Atoms block) to difDistValue, given in Angstrom.

Note that the above constraints do **not** need to be satisfied at the beginning of the optimization.

The Distance keyword described above can be used to set up constraints for specific pairs of atoms in the system. For some applications, this input can be cumbersome and one would rather specify a constraint for **all** bonds of a particular type. Since the AMS2022.1 release, there is a simplified input for this using the All keyword. The All keyword allows generating distance constraints for matching *bonds defined in the System block* (page 38). The bonds for which distance constraints will be generated can be selected based on the elements of the bonded atoms, as well as the bond order defined in the System block. The following example would constrain all carbon-carbon single bonds to a distance of 1.4 Angstrom, and all bonds with hydrogen atoms to their initial distance:

```
Constraints
  All single bonds C C to 1.4
  All bonds H *
End
```

It is also possible to constrain neighboring bonds and the angle between them using the combination All triangles. This is mostly useful for water molecules, which can then easily be made entirely rigid:

```
All triangles H O H
```

Warning: It is very easy to generate a large number of interdependent distance constraints using the All triangles keyword. Interdependent distance constraints can impair optimizer performance and should be avoided. Consider for example the input All triangles H C H for methane: there are 6 triangles matching this description in CH₄, generating 18 interdependent distance constraints. A rigid CH₄ would be better set up using a block constraint.

See below for a full description of the Constraints%All keyword.

Constraints

All

Type
String

Recurring

True

Description

Fix multiple distances using one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 as well as the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then any bond between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. If the distance is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at the start of the simulation can be constrained, which means that the bonds may need to be specified in the System block.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

Block constraints

Block constraints can be used to treat parts of the System as a rigid unit in the optimization.

BlockAtoms [atomIdx1 ... atomIdxN]

Creates a block constraint (freezes all internal degrees of freedom) for a set of atoms identified by the list of integers [atomIdx1 ... atomIdxN]. These atom indices refer to the order of the atoms in the System%Atoms block.

Block regionName

Creates a block constraint (freezes all internal degrees of freedom) for all atoms in a *region* (page 37) defined in the System%Atoms block. Example:

```
System
  Atoms
    C  0.0  0.0  0.0    region=myblock
    C  0.0  0.0  1.0    region=myblock
    C  0.0  1.0  0.0
  End
End
Constraints
  Block myblock
End
```

Lattice constraints

For lattice optimizations, the following constraints can be used on the lattice degrees of freedom:

FreezeStrain [xx] [xy] [xz] [yy] [yz] [zz]

Exclusively for lattice optimizations: Freezes any lattice deformation corresponding to a particular component of the strain tensor. Accepts a set of strain components [xx, xy, xz, yy, yz, zz] to be frozen.

EqualStrain [xx] [xy] [xz] [yy] [yz] [zz]

Exclusively for lattice optimizations: Accepts a set of strain components [xx, xy, xz, yy, yz, zz] which are to be kept equal. The applied strain will be determined by the average of the corresponding stress tensors components.

Restraints

Not all optimizers support constraints. An alternative is to use so-called restraints. These are not exact constraints, but rather a number of springs that pull the system towards the preferred constraints, see [Restraints](#) (page 332).

4.3.6 Optimization under pressure / external stress

Pressure (page 329) or *non-isotropic external stress* (page 330) can be included in your simulation via the corresponding *engine addons* (page 325).

4.3.7 Optimization methods

The AMS driver implements a few different geometry optimization algorithms. It also allows to choose the coordinate space in which the optimization is performed:

```
GeometryOptimization
  Method [Auto | Quasi-Newton | FIRE | L-BFGS | ConjugateGradients | Dimer]
  CoordinateType [Auto | Delocalized | Cartesian]
End
```

GeometryOptimization

Method

Type

Multiple Choice

Default value

Auto

Options

[Auto, Quasi-Newton, FIRE, L-BFGS, ConjugateGradients, Dimer]

GUI name

Optimization method

Description

Select the optimization algorithm employed for the geometry relaxation. Currently supported are:

the Hessian-based Quasi-Newton-type BFGS algorithm,

the fast inertial relaxation method (FIRE),

the limited-memory BFGS method,

and the conjugate gradients method. The default is to choose an appropriate method automatically based on the engine's speed, the system size and the supported optimization options.

CoordinateType

Type

Multiple Choice

Default value

Auto

Options

[Auto, Delocalized, Cartesian]

GUI name

Optimization space

Description

Select the type of coordinates in which to perform the optimization. 'Auto' automatically selects the most appropriate CoordinateType for a given Method.

If 'Auto' is selected, Delocalized coordinates will be used for the Quasi-Newton method, while Cartesian coordinates will be used for all other methods.

We **strongly** advise leaving both the Method as well as the Coordinate type on the Auto setting. There are many restrictions as to which optimizer and coordinate type can be used together with which kind of optimization. The following (roughly) sketches the compatibility of the different optimizers and options:

Optimizer	Constraints	Lattice opt.	Coordinate types
Quasi-Newton	All, except strain	Yes	All
FIRE	Fixed coordinates, distances, strain	Yes	Cartesian
L-BFGS	No	Yes	Cartesian
Conjugate gradients	No	No	Cartesian

Furthermore for optimal performance the optimizer should be chosen with the speed of the engine: a faster engine in combination should use an optimizer with little overhead (FIRE), while slower engines should use optimizers that strictly minimize the number of steps (Quasi-Newton). This is all handled automatically by default, and we recommend changing Method and Coordinate only in case there are problems with the automatic choice.

The following subsections list the strengths and weaknesses of the individual optimizers in some more detail, motivating why which optimizer is chosen automatically under which circumstances.

Quasi-Newton

This optimizer implements a quasi Newton approach^{1,2,3}, using the Hessian for computing changes in the geometry so as to reach a local minimum. The Hessian itself is typically *approximated* (page 62) in the beginning and updated in the process of optimization. It uses delocalized coordinates by default both for molecules and periodic systems. The molecular part is based mainly on the work by Marcel Swart⁴. Cartesian coordinates are used in the presence of an external electric field and/or frozen atom constraints.

¹ L. Versluis and T. Ziegler, *The determination of Molecular Structure by Density Functional Theory*, *Journal of Chemical Physics* 88, 322 (1988) (<https://doi.org/10.1063/1.454603>)

² L. Versluis, *The determination of molecular structures by the HFS method*, PhD thesis, University of Calgary, 1989

³ L. Fan and T. Ziegler, *Optimization of molecular structures by self consistent and non-local density functional theory*, *Journal of Chemical Physics* 95, 7401 (1991) (<https://doi.org/10.1063/1.461366>)

⁴ M. Swart and F.M. Bickelhaupt, *Optimization of strong and weak coordinates*, *International Journal of Quantum Chemistry* 106, 2536 (2006) (<https://doi.org/10.1002/qua.21049>)

The Quasi-Newton (QN) optimizer supports all types of constraints and can be used for both molecular and periodic systems, including lattice optimizations. For cases where the optimization can be performed in delocalized coordinates, the number of steps taken to reach the local minimum is usually smaller than when optimizing in Cartesian ones. For fast compute *engines* (page 315), the overhead of the QN optimizer can become a bottleneck of the calculation, thus a more light-weight optimizer such as *FIRE* (page 63) may give an better overall performance. In principle, a QN optimization in delocalized coordinates may run out of memory for a very large system (say over 1000 atoms) because of the SVD step. However, since it is going to be used for a moderate-to-slow engine we still recommend sticking to it for the benefit of fewer steps. Because of these properties the QN optimizer is the default in AMS for all kinds of optimizations with moderate and slow engines, such DFTB and ADF. It is also used as the optimizer back-end for the *PES scan task* (page 76), the *transition state search* (page 72) as well as the calculation of the *elastic tensor* (page 242).

Details of the Quasi-Newton optimizer are configured in a dedicated block:

```
GeometryOptimization
  Quasi-Newton
    MaxGDIISVectors integer
    Step
      TrustRadius float
      VaryTrustRadius Yes/No
    End
    UpdateTSVectorEveryStep Yes/No
  End
End
```

GeometryOptimization

Quasi-Newton

Type

Block

Description

Configures details of the Quasi-Newton geometry optimizer.

MaxGDIISVectors

Type

Integer

Default value

0

Description

Sets the maximum number of GDIIS vectors. Setting this to a number >0 enables the GDIIS method.

Step

Type

Block

Description

TrustRadius

Type

Float

Description

Initial value of the trust radius.

VaryTrustRadius

Type

Bool

Description

Whether to allow the trust radius to change during optimization. By default True during energy minimization and False during transition state search.

UpdateTSVectorEveryStep**Type**

Bool

Default value

Yes

GUI name

Update TSRC vector every step

Description

Whether to update the TS reaction coordinate at each step with the current eigenvector.

The Quasi-Newton optimizer uses the Hessian to compute the step of the geometry optimization. The Hessian is typically approximated in the beginning and then updated during the optimization. A very good initial Hessian can therefore increase the performance of the optimizer and lead to faster and more stable convergence. The choice of the initial Hessian can be configured in a dedicated block:

```
GeometryOptimization
  InitialHessian
    File string
    Type [Auto | UnitMatrix | Swart | FromFile | Calculate | ↪
↪CalculateWithFastEngine]
  End
End
```

GeometryOptimization**InitialHessian****Type**

Block

Description

Options for initial model Hessian when optimizing systems with the Quasi-Newton method.

File**Type**

String

GUI name

Initial Hessian from

Description

KF file containing the initial Hessian (or the results dir. containing it). This can be used to load a Hessian calculated in a previously with the [Properties%Hessian] keyword.

Type**Type**

Multiple Choice

Default value

Auto

Options

[Auto, UnitMatrix, Swart, FromFile, Calculate, CalculateWithFastEngine]

GUI name

Initial Hessian

Description

Select the type of initial Hessian. Auto: let the program pick an initial model Hessian. Unit-Matrix: simplest initial model Hessian, just a unit matrix in the optimization coordinates. Swart: model Hessian from M. Swart. FromFile: load the Hessian from the results of a previous calculation (see InitialHessian%File). Calculate: compute the initial Hessian (this may be computationally expensive and it is mostly recommended for TransitionStateSearch calculations). CalculateWithFastEngine: compute the initial Hessian with a faster engine.

While there are some options for the construction of approximate model Hessians, the best initial Hessians are often those calculated explicitly at a lower level of theory, e.g. the real DFTB Hessian can be used the initial Hessian for an optimization with the more accurate BAND engine. Using the `CalculateWithFasterEngine` keyword can be used to automatically chose a fast engine at a lower level of theory. What the lower level of theory is depends on the main engine used in the calculation: DFTB with the GFN1-xTB model is used as the lower level of theory for relatively slow engines, e.g. DFT based engines. For semi-empirical engines like DFTB or MOPAC, the lower level of theory is currently UFF. If more control over the lower level engine is needed, the initial Hessian can be calculated with a user defined engine and then loaded from file, see [this example](#) (page 457).

FIRE

The Fast Inertial Relaxation Engine⁵ based optimizer has basically no overhead per step, so that the speed of the optimization purely depends on the performance of the used compute [engine](#) (page 315). As such it is a good option for large systems or fast compute engines, where the overhead of the Quasi-Newton optimizer would be significant. Note that it also supports [fixed atom constraints](#) (page 56) and [coordinate constraints](#) (page 56) (as long as the value of the constrained coordinate is already satisfied in the input geometry), distance constraints, as well as lattice optimizations (with strain constraints).

FIRE is selected as the default optimizer for fast compute engines if it is compatible with all other settings of the optimization (i.e. no unsupported constraints or coordinate types).

Note: FIRE is a very robust optimizer. In case of convergence problems with the other methods, it is a good idea to see if the optimization converges with FIRE. If it does not, it is very likely that the problem is not the optimizer but the shape of the potential energy surface ...

The details of the FIRE optimizer are configured in a dedicated block. It is quite easy to make the optimization numerically unstable when tweaking these settings, so we strongly recommend leaving everything at the default values.

```
GeometryOptimization
  FIRE
    AllowOverallRotation Yes/No
    AllowOverallTranslation Yes/No
    MapAtomsToUnitCell Yes/No
    NMin integer
    alphaStart float
    dtMax float
    dtStart float
    fAlpha float
```

(continues on next page)

⁵ E. Bitzek, P. Koskinen, F. Gähler, M. Moseler and P. Gumbusch, *Structural Relaxation Made Simple*, *Physical Review Letters* 97, 170201 (2006) (<https://doi.org/10.1103/PhysRevLett.97.170201>)

(continued from previous page)

```
fDec float
fInc float
strainMass float
End
End
```

GeometryOptimization**FIRE****Type**

Block

Description

This block configures the details of the FIRE optimizer. The keywords name correspond the the symbols used in the article describing the method, see PRL 97, 170201 (2006).

AllowOverallRotation**Type**

Bool

Default value

Yes

Description

Whether or not the system is allowed to freely rotate during the optimization. This is relevant when optimizing structures in the presence of external fields.

AllowOverallTranslation**Type**

Bool

Default value

No

Description

Whether or not the system is allowed to translate during the optimization. This is relevant when optimizing structures in the presence of external fields.

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

Map the atoms to the central cell at each geometry step.

NMin**Type**

Integer

Default value

5

Description

Number of steps after stopping before increasing the time step again.

alphaStart**Type**

Float

Default value

0.1

Description

Steering coefficient.

dtMax**Type**

Float

Default value

1.0

Unit

Femtoseconds

Description

Maximum time step used for the integration. For ReaxFF and APPLE&P, this value is reduced by 50%.

dtStart**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

Initial time step for the integration.

fAlpha**Type**

Float

Default value

0.99

Description

Reduction factor for the steering coefficient.

fDec**Type**

Float

Default value

0.5

Description

Reduction factor for reducing the time step in case of uphill movement.

fInc

Type

Float

Default value

1.1

Description

Growth factor for the integration time step.

strainMass**Type**

Float

Default value

0.5

Description

Fictitious relative mass of the lattice degrees of freedom. This controls the stiffness of the lattice degrees of freedom relative to the atomic degrees of freedom, with smaller values resulting in a more aggressive optimization of the lattice.

Limited-memory BFGS

AMS also offers an L-BFGS based geometry optimizer. It usually converges faster than *FIRE* (page 63), but does not support constrained optimizations. For periodic systems it can be quite good for lattice optimizations. The new implementation has not been thoroughly tested yet, therefore never selected automatically. For large systems and fast engines you may want to disable symmetry: simply the detection of (non-existing) symmetry may be a huge overhead.

```
GeometryOptimization
  HessianFree
    Step
      MaxCartesianStep float
      MinRadius float
      TrialStep float
      TrustRadius float
    End
  End
End
```

GeometryOptimization**HessianFree****Type**

Block

Description

Configures details of the Hessian-free (conjugate gradients or L-BFGS) geometry optimizer.

Step**Type**

Block

Description**MaxCartesianStep****Type**

Float

Default value

0.1

Unit

Angstrom

Description

Limit on a single Cartesian component of the step.

MinRadius**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Minimum value for the trust radius.

TrialStep**Type**

Float

Default value

0.0005

Unit

Angstrom

Description

Length of the finite-difference step when determining curvature. Should be smaller than the step convergence criterion.

TrustRadius**Type**

Float

Default value

0.2

Unit

Angstrom

Description

Initial value of the trust radius.

Conjugate gradients

AMS also offers a conjugate gradients based geometry optimizer, as it was also implemented in the pre-2018 releases of the DFTB program. However, it is usually slightly slower than *FIRE* (page 63), and supports neither lattice nor constrained optimizations. It is therefore never selected automatically, and we do not recommend using it. Like L-BFGS, the conjugate gradients optimizer is also configured in the `HessianFree` block, see L-BFGS section above for details.

Dimer method for TS search

The dimer method⁶ works differently from the quasi-Newton. It is designed to always follow the lowest mode from the initial geometry without requiring the Hessian. The mode (a.k.a. the dimer vector) is defined by two close-lying points (midpoint and endpoint) on the potential energy surface. The vector's direction is found by minimizing the PES curvature (finite-difference second derivative of the energy along the vector) on the hypersphere around the midpoint. This is achieved by iteratively rotating the vector until the estimated rotation angle drops below the `AngleThreshold` value. These are called rotation iterations. After that, a translation iteration is performed that maximized the energy along the dimer vector and minimizes it in all other directions. The optimization is considered converged when the largest energy gradient component drops below the `Convergence%Gradient` value.

Both the rotation and the translation are performed in Cartesian coordinates using the L-BFGS method as described in⁷. The L-BFGS trust radii are set by the `RotationTrustRadius` and `TranslationTrustRadius` keywords, respectively. The TS search can be restricted to a subset of atomic coordinates defined by the `Region` keyword, in which case only atoms of the selected region are moved during rotations.

Each translation iteration requires two engine evaluations: one for the midpoint and one for the endpoint. Each rotation iteration additionally requires one or two engine evaluations depending on the `ExtrapolateForces` setting. It should be noted that with the default settings the rotations are usually necessary only at the beginning of the optimization. At the later steps the dimer vector (the direction of the search) does not usually change much.

By default, the initial value of the search vector is chosen randomly. This can be changed using either a `ReactionCoordinate` or a `System final` input block described in the *Transition state search* (page 72) section.

```
GeometryOptimization
  Dimer
    AngleThreshold float
    DimerDelta float
    ExtrapolateForces Yes/No
    LBFSGMaxVectors integer
    MaxRotationIterations integer
    Region string
    RotationTrustRadius float
    TranslationTrustRadius float
  End
End
```

GeometryOptimization

Dimer

6

G. Henkelman and H. Jonsson, *A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives*, *Journal of Chemical Physics* 111, 7010 (1999) (<https://doi.org/10.1063/1.480097>)

7

J. Kästner and P. Sherwood, *Superlinearly converging dimer method for transition state search*, *Journal of Chemical Physics* 128, 014106 (2008) (<https://doi.org/10.1063/1.2815812>)

Type

Block

Description

Options for the Dimer method for transition state search.

AngleThreshold**Type**

Float

Default value

1.0

Unit

Degree

Description

The rotation is considered converged when the the rotation angle falls below the specified threshold.

DimerDelta**Type**

Float

Default value

0.01

Unit

Angstrom

Description

Euclidian distance between the midpoint and the endpoint.

ExtrapolateForces**Type**

Bool

Default value

Yes

Description

Set to false to call engine to calculate forces at the extrapolated rotation angle instead of extrapolating them.

LBFGSMaxVectors**Type**

Integer

Default value

10

Description

Max number of vectors for the L-BFGS algorithm to save.

MaxRotationIterations**Type**

Integer

Default value

10

Description

Maximum number of rotation iterations for a single translation step.

Region**Type**

String

Default value

*

Description

Include only atoms of the specified region(s) in the rotations, which allows searching for a transition state involving selected atoms only.

RotationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during rotation iterations.

TranslationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during translation iterations.

4.3.8 Troubleshooting

Failure to converge

First of all one should look how the energy changed during the latest ten or so iterations. If the energy is decreasing more or less consistently, possibly with occasional jumps, then there is probably nothing wrong with the optimization. This behavior is typical in the cases when the starting geometry was far away from the minimum and the optimization has a long way to go. Just increase the allowed number of iterations, restart from the latest geometry and see if the optimization converges.

If the energy oscillates around some value and the energy gradient hardly changes then you may need to look at the calculation setup.

The success of geometry optimization depends on the accuracy of the calculated forces. The default accuracy settings are sufficient in most cases. There are, however, cases when one has to increase the accuracy in order to get geometry optimization converged. First of all, this may be necessary if you tighten the optimization convergence criteria. In some cases it may be necessary to increase the accuracy also for the default criteria. Please refer to the [engine manuals](#)

(page 315) for instructions on how to increase the accuracy of an engine's energies and gradients. Often this is done with the `NumericalQuality` keyword in the engine input.

A geometry optimization can also fail to converge because the underlying potential energy surface is problematic, e.g. it might be discontinuous or not have a minimum at which the gradients vanish. This often indicates real problems in the calculation setup, e.g. an electronic structure that changes fundamentally between subsequent steps in the optimization. In these cases it is advisable to run a single point calculation at the problematic geometries and carefully check if the results are physically actually sensible.

Finally it can also be a technical problem with the specific *optimization method* (page 59) used. In these cases switching to another method could help with convergence problems. We recommend first trying the *FIRE* (page 63) optimizer, as it is internally relatively simple and stable.

Converged, but did not find a minimum on the PES

If *Normal Modes* (page 246) or *PES point characterization* (page 239) were requested, the *PES character* (page 239) of the final optimized geometry is checked.

While you would generally expect the final optimized geometry to be a local minimum (i.e. no imaginary frequencies), it can happen for the optimization procedure to converge to a saddle point instead (i.e. a stationary point on the PES with one or more imaginary frequency). In that case you will get the following error message:

```
ERROR: Geometry optimization failed! (Converged, but did not find a minimum on the
→PES.)
```

Depending on your application, (small) imaginary frequencies might not be a problem. But if you want to get rid of all imaginary frequencies, these are a few recommendations:

- Try using a tighter *gradient convergence threshold* (page 50) (`GeometryOptimization%Convergence%Gradient`). E.g. `1.0E-4` (instead of the default `1.0E-3`).
- Try using the *MaxRestart* (page 54) option. Reasonable values for `MaxRestart` range from 2 to 20 (if there are many imaginary frequencies, a larger value might be needed). When using the `MaxRestart` option, you might try increasing `RestartDisplacement`.
- Open the results with the **AMSSpectra** GUI module and visually inspect the negative/imaginary vibrational modes. You can then manually perturb the input geometry along those modes and re-run the optimization.
- If the system is symmetric and *symmetry* (page 35) is enabled, the optimizer will try not to break the symmetry of the system. It might be that in order to reach the local minimum, symmetry must be broken. Try disabling symmetry (`UseSymmetry False`) and slightly perturb the atomic coordinates in order to break the symmetry (or use the *PerturbCoordinates* (page 34) option.)

Restarting a geometry optimization

During a running optimization the system's geometry is written out to the AMS driver's output file `ams.rkf` after every step (in the `Molecule` section). This means that crashed or otherwise canceled geometry optimizations can be restarted by simply loading the last frame from there using the `LoadSystem` keyword, see *its documentation* (page 41) in the system definition section of this manual:

```
LoadSystem File=my_crashed_GO.results/ams.rkf
```

This can of course also be used to continue an optimization but e.g. with tighter convergence criteria or a different optimizer, as it essentially starts a new geometry optimization from the previous geometry, and does not propagate any information internal to the optimizer (e.g. the approximate Hessian for the Quasi-Newton optimizer or the FIRE velocities) to the new job. As such it might take a few more steps to convergence than if the original job had continued, but allows for additional flexibility.

Restart an Engine from file

```
EngineRestart string
```

EngineRestart

Type

String

Description

The path to the file from which to restart the engine.

Should be a proper engine result file (like adf.rkf, band.rkf etc), or the name of the results directory containing it.

EngineRestart should typically be combined with [LoadSystem](#) (page 41) and depending on whether one wants to change the engine configuration may be combined [LoadEngine](#) (page 325). For example, in case of ADF a restart of a geometry optimization (use TAPE13 instead of adf.rkf in case of a crash) may look like:

```
Task GeometryOptimization
LoadSystem
  File crashed_or_not_crashed.results/ams.rkf
End
EngineRestart not_crashed.results/adf.rkf
LoadEngine not_crashed.results/adf.rkf
```

4.4 Transition state search

A transition state (TS) search is very much like a [geometry optimization](#) (page 49): the purpose is to find a stationary point on the potential energy surface, primarily by monitoring the energy gradients, which should vanish. The difference between a transition state and a minimum is that at the transition state the Hessian has a negative eigenvalue: We are at a saddle point, not a minimum, with the “negative” mode connecting the two basins on the potential energy surface.

See also:

- [Examples](#) (page 457)
- Tutorial [Transition state search and characterization of a Ziegler Natta Catalyst](#)
- Tutorial [Tips and Tricks for Transition State Searches for Click Reactions](#)

A transition state search in AMS is performed by selecting the corresponding task:

```
Task TransitionStateSearch
```

Due to the similarities between energy minimization and transition state search, the `TransitionStateSearch` task in AMS is actually implemented as a special kind of geometry optimization using either the [quasi-Newton](#) (page 60) or the [dimer](#) (page 68) optimizer. As such most settings and keywords described on the [geometry optimization manual page](#) (page 49) also apply to transition state searches.

In a geometry optimization with a quasi-Newton based optimizer the Hessian is used to make a reasonably sized step in the “downhill” direction on the potential energy surface, as the goal is simply to minimize the energy. A transition state search is a bit different: In the first step a normal mode is picked along which the energy is to be *maximized*, while it is *minimized* along all other directions. Normally the mode with the lowest eigenvalue is picked, since we know that there should be exactly one negative eigenvalue at the TS geometry, but one can also choose an approximate reaction coordinate, either by setting the `ReactionCoordinate` or by adding another System block named `final`, similar to how it’s done for NEB. If the initial geometry is sufficiently close to the transition state, i.e. we are close to the saddle, the lowest

mode is normally the correct one to follow in order to get to the ridge of the saddle. Alternatively a different mode can also be selected manually.

```

TransitionStateSearch
  ModeToFollow integer
  ReactionCoordinate
    Angle string
    Block string
    BlockAtoms integer_list
    Coordinate string
    Dihedral string
    Distance string
  End
End

```

TransitionStateSearch

Type

Block

Description

Configures some details of the transition state search.

ModeToFollow

Type

Integer

Default value

1

Description

In case of Transition State Search, here you can specify the index of the normal mode to follow (1 is the mode with the lowest frequency).

ReactionCoordinate

Type

Block

Description

Specify components of the transition state reaction coordinate (TSRC) as a linear combination of internal coordinates (distances or angles).

Angle

Type

String

Recurring

True

Description

The TSRC contains the valence angle between the given atoms. Three atom indices followed by the weight.

Block

Type

String

Recurring

True

Description

Name of the region. Only atoms of the region will be included in the TSRC. It is useful when computing the reaction coordinate from the initial Hessian, in which case only part of the Hessian will be analyzed.

BlockAtoms**Type**

Integer List

Value Range

value > 0

Recurring

True

Description

List of atom indices. Only the listed atoms will be included in the TSRC. It is useful when computing the reaction coordinate from the initial Hessian, in which case only part of the Hessian will be analyzed.

Coordinate**Type**

String

Recurring

True

Description

The TSRC contains Cartesian displacement of an atom: atom index followed by [x|y|z] and the weight.

Dihedral**Type**

String

Recurring

True

Description

The TSRC contains the dihedral angle between the given atoms. Four atom indices followed by the weight.

Distance**Type**

String

Recurring

True

Description

The TSRC contains the distance between the given atoms. Two atom indices followed by the weight.

This selection happens only in the first step. Subsequent steps will attempt to maximize along the mode that resembles most (by overlap) the previous maximization direction.

Practice shows that a transition state are much harder to find than a minimum. For a large part this is due to the much stronger anharmonicities that usually occur near the TS, which threaten to invalidate the quasi-Newton methods to find

the stationary point. For this reason it is good advice to be more cautious in the optimization strategy when approaching a transition state:

- We recommend starting the transition state search with an initial geometry that is already close to the transition state. One can use a *potential energy surface scan* (page 76) along something resembling the reaction coordinate to get a rough idea where the transition state is. This geometry can then be used as an initial geometry for the transition state search.
- It is strongly recommended to provide a good initial Hessian for the transition state search. Otherwise the first step of the search may not be taken in the correct direction and subsequent steps may keep steering in the wrong direction. In AMS this is easily possible by loading a Hessian from a previous calculation, see the *initial Hessian section* (page 62) of this manual. A good way to obtain a reasonable Hessian is to compute it explicitly with one of the fast engines (i.e. at a lower level of theory) and read that Hessian as the initial Hessian for the transition state search at a higher level of theory.
- When no accurate initial Hessian is available, it may be a good idea to specify an approximate normal mode vector using the `ReactionCoordinate` input block. A reaction coordinate (RC) is a linear combination of distances and/or valence and dihedral angles:

```
TransitionStateSearch
  ReactionCoordinate
    Distance  i j      fac
    Distance  i j      fac
    Angle     i j k    fac
    Dihedral  i j k l  fac
  End
End
```

Here, *i*, *j*, *k*, and *l* are atom indices, and the *fac* is the factor with which the internal coordinate enters the linear combination.

One should be careful when specifying more than one bond or angle as the reaction coordinate. For example, suppose atom 2 is located between atoms 1 and 3. Then the following `ReactionCoordinate` block:

```
TransitionStateSearch
  ReactionCoordinate
    Distance  1 2  1.0
    Distance  2 3 -1.0
  End
End
```

means that the RC consists of two distances: $R(1-2)$ and $R(2-3)$. The positive direction of the RC is defined as an increase of the $R(1-2)$ and a decrease of the $R(2-3)$. In other words, this RC corresponds to atom 2 moving along the $R(1-3)$ axis.

- As an alternative to `ReactionCoordinate` one can add a second `System` to the input and call it `final`. The difference between the initial and the `final` geometry will be used as the reaction coordinate. For non-periodic systems, the difference is computed in the internal coordinates and only coordinates that differ significantly (by more than about 0.1 Bohr or radian) are included. For this reason, the `final` geometry should not be too close to the initial. For periodic systems, the difference in the Cartesian coordinates is used. In this case, only coordinates that differ by more than 0.5 Bohr are included.
- When the method converges it is usually a good idea to verify that the found geometry is indeed a transition state. This can be done by performing a frequency analysis and checking whether the Hessian has exactly one negative eigenvalue (represented by a mode with a negative frequency). Doing so is expensive however. Since we are really only interested in the lowest two normal modes calculating the full Hessian is however not necessary and one can use the faster PES point characterization instead. This uses a Davidson-type algorithm to obtain the lowest few normal modes without constructing the full Hessian. The user is referred to the *PES point characterization* (page 239) documentation page for further details.

4.5 Linear Transit, PES scan

The PES scan task in AMS allows users to scan the potential energy surface of a system along one or multiple degrees of freedom, while relaxing all other degrees of freedom. If only one coordinate is scanned, this kind of calculation is usually just called a linear transit. However, since AMS allows scanning of multiple coordinates, and linear transit is just a special case of such a calculation, the task is always called a PES scan in AMS.

A linear transit may be used for instance to sketch an approximate path over the transition states between reactants and products. From this a reasonable guess for the Transition State can be obtained which may serve as starting point for a true transition state search for instance.

See also:

- [Examples](#) (page 457)
- [Tutorial Transition state search and characterization of a Ziegler Natta Catalyst](#)

The PES scan task is enabled by selecting it with the Task keyword:

```
Task PESScan
```

The PESScan block configures all details of the scan:

```
PESScan
  CalcPropertiesAtPESPoints [True | False]
  FillUnconvergedGaps [True | False]
  ScanCoordinate
    nPoints integer
    Coordinate integer [x|y|z] (float){2}
    Distance (integer){2} (float){2}
    Angle (integer){3} (float){2}
    Dihedral (integer){4} (float){2}
    SumDist (integer){4} (float){2}
    DifDist (integer){4} (float){2}
    FromLattice
      ...
    End
    ToLattice
      ...
    End
    FromStrainVoigt (float){6}
    ToStrainVoigt (float){6}
    CellVolumeRange (float){2}
    CellVolumeScalingRange (float){2}
    LatticeARange (float){2}
    LatticeBRange (float){2}
    LatticeCRange (float){2}
  End
End
```

The PESScan block needs to contain at least one ScanCoordinate block specifying which coordinate to scan, and how many points (keyword nPoints) to sample along this coordinate. By default, 10 points are sampled along each scanned coordinate (including the start and end point of the scan).

4.5.1 Distance, angle, and dihedral angle coordinates

The coordinate descriptors are very similar to the *constraint descriptors* (page 55) in the `Constraints` block used by the geometry optimization task, but are followed by two values delimiting the start and end of the coordinates, instead of just a single value:

Coordinate atomIdx [x|y|z] startValue endValue

Moves the atom with index `atomIdx` (following the order in the `System` block) along the a cartesian coordinate (`x`, `y` or `z`), starting at `startValue` and ending at `endValue` (given in Angstrom).

Distance atomIdx1 atomIdx2 startDist endDist

Scans the distance between the atoms with index `atomIdx1` and `atomIdx2`, starting from `startDist` and ending at `endDist`, both given in Angstrom.

Angle atomIdx1 atomIdx2 atomIdx3 startAngle endAngle

Scans the angle (1)–(2)–(3) between the atoms with indices `atomIdx1`–3, as given by their order in the `System%Atoms` block. The scanned angle starts at `startAngle` and ends at `endAngle`, given in degrees.

Dihedral atomIdx1 atomIdx2 atomIdx3 atomIdx4 startAngle endAngle

Scans the dihedral angle (1)–(2)–(3)–(4) between the atoms with indices `atomIdx1`–4, as given by their order in the `System%Atoms` block. The scanned dihedral starts at `startAngle` and ends at `endAngle`, given in degrees.

SumDist atomIdx1 atomIdx2 atomIdx3 atomIdx4 start end

Scans the sum of the distances $R(1,2)+R(3,4)$ between the atoms with indices `atomIdx1`–4, as given by their order in the `System%Atoms` block. The values to be scanned start at `start` and end at `end`, given in Angstrom.

DifDist atomIdx1 atomIdx2 atomIdx3 atomIdx4 start end

Scans the difference between the distances $R(1,2)-R(3,4)$ of the atoms with indices `atomIdx1`–4, as given by their order in the `System%Atoms` block. The values to be scanned start at `start` and end at `end`, given in Angstrom.

4.5.2 Joint scan coordinates

Note that multiple of these coordinate descriptors can be combined within a single `ScanCoordinate` block. This combines the individual coordinates into one compound coordinate, i.e. all coordinates will transit together through their respective ranges. In this way the symmetric stretch in water could be scanned by specifying the following single `ScanCoordinate` block (assuming that the oxygen atom is the first in the `System%Atoms` block):

```
ScanCoordinate
  Distance  1 2  0.8 1.1
  Distance  1 3  0.8 1.1
End
```

4.5.3 Multidimensional PES scan

A multidimensional PES scan can be performed by specifying multiple `ScanCoordinate` blocks in the input. To scan the space spanned by the bending and symmetric stretch modes in water, one would use the following scan coordinates:

```
ScanCoordinate
  Distance  1 2  0.8 1.1
  Distance  1 3  0.8 1.1
End
ScanCoordinate
```

(continues on next page)

(continued from previous page)

```
Angle  2 1 3  90 130
End
```

In principle an arbitrary number of `ScanCoordinate` blocks can be combined to specify the scanned configuration space. However, the total number of sample points is the product of the number of points along all coordinates, and hence grows quickly with the number of dimensions. Furthermore, only 1D (linear transit) and 2D PES scans can be visualized in the GUI. We therefore suggest sticking with ≤ 2 dimensional PES scans. (Note that it is possible to constrain additional degrees of freedom through the `Constraints` block. This could be used to sample a few points along a third dimension “manually”, while still being able to see the surfaces in the GUI.)

4.5.4 Lattice scan coordinates for periodic systems

Several ways to scan lattice degrees of freedom were added to the AMS2022.1 release. **Note that for each unit cell, a geometry optimization is performed.** To keep the fractional coordinates fixed during the PES Scan, set `GeometryOptimization%MaxIterations` to 0.

There can be only one scan coordinate for lattice degrees of freedom in a single PES scan job. Also note that scan coordinates for lattice degrees of freedom may not contain other coordinate descriptors within the same scan coordinate. It is for example *not* possible to have a *joint scan coordinate* (page 77) for a concerted lattice and bond length stretch.

It is, however, perfectly fine to combine a lattice scan coordinate with another scan coordinate for a *two-dimensional PES scan* (page 77).

See also:

Example input files (page 498) for PES scan jobs with lattice degrees of freedom.

Important: If you use k-space sampling (e.g., with BAND or DFTB), then the k-space grid is determined for the **input structure**, which is not necessarily any of the sampled points.

Isotropic scaling of the unit cell volume or area

CellVolumeScalingRange startValue endValue

Isotropic scaling of the unit cell. Example: `CellVolumeScalingRange 0.9 1.1` will scale the volume between 90% and 110% of the original unit cell. For 2D-periodic crystals, the area will be scaled instead.

CellVolumeRange startValue endValue

Isotropic scaling of the unit cell. Example: `CellVolumeRange 300 500` will scale the volume between 300 Å³ and 500 Å³ for a 3d-periodic system. For 2D-periodic systems, the area will be scaled between 300 Å² and 500 Å².

Scaling of the lattice vector lengths

These options keep the angles between lattice vectors fixed.

LatticeARange startValue endValue

Scans the length of the **first** lattice vector. Can be combined with the `LatticeBRange` and `LatticeCRange` keywords, but no other coordinates within the same `ScanCoordinate`. Unit: angstrom.

LatticeBRange startValue endValue

Scans the length of the **second** lattice vector. Can be combined with the `LatticeARange` and `LatticeCRange` keywords, but no other coordinates within the same `ScanCoordinate`. Unit: angstrom.

LatticeCRange startValue endValue

Scans the length of the **third** lattice vector. Can be combined with the LatticeARange and LatticeBRange keywords, but no other coordinates within the same ScanCoordinate. Unit: angstrom.

Strain matrix in Voigt notation**3D crystal: FromStrainVoigt xx yy zz yz xz xy, ToStrainVoigt xx yy zz yz xz xy**

The FromStrainVoigt and ToStrainVoigt keywords need to be applied together. Example: FromStrainVoigt -0.1 -0.1 -0.1 -0.1 -0.1 -0.1, ToStrainVoigt 0.1 0.1 0.1 0.1 0.1 0.1

2D crystal: FromStrainVoigt xx yy xy, ToStrainVoigt xx yy xy

The FromStrainVoigt and ToStrainVoigt keywords need to be applied together. Example: FromStrainVoigt -0.1 -0.1 -0.1, ToStrainVoigt 0.1 0.1 0.1

1D crystal: FromStrainVoigt xx, ToStrainVoigt xx

The FromStrainVoigt and ToStrainVoigt keywords need to be applied together. Example: FromStrainVoigt -0.1, ToStrainVoigt 0.1

Scan arbitrary lattices

Scan arbitrary lattices specifying the initial and final lattice vectors, using the same format as in the *System%Lattice* (page 32) block. The PES scan will then interpolate the lattice vectors linearly between these two values:

```
ScanCoordinate
  FromLattice
    ! lattice vectors as in System%Lattice
  End
  ToLattice
    ! ...
  End
End
```

4.5.5 Calculate properties for all PES points

By default the engine result files for the individual PES points are not saved on disk, as this can easily lead to huge amounts of data to be stored. This behavior can be changed with the PESScan%CalcPropertiesAtPESPoints keyword:

PESScan**CalcPropertiesAtPESPoints****Type**

Bool

Default value

No

Description

Whether to perform an additional calculation with properties on all the sampled points of the PES. If this option is enabled AMS will produce a separate engine output file for every sampled PES point.

Note that this performs a full single point calculation on every sampled PES point, including the calculation of any *PES point properties* (page 235) selected in Properties block.

4.5.6 Troubleshooting

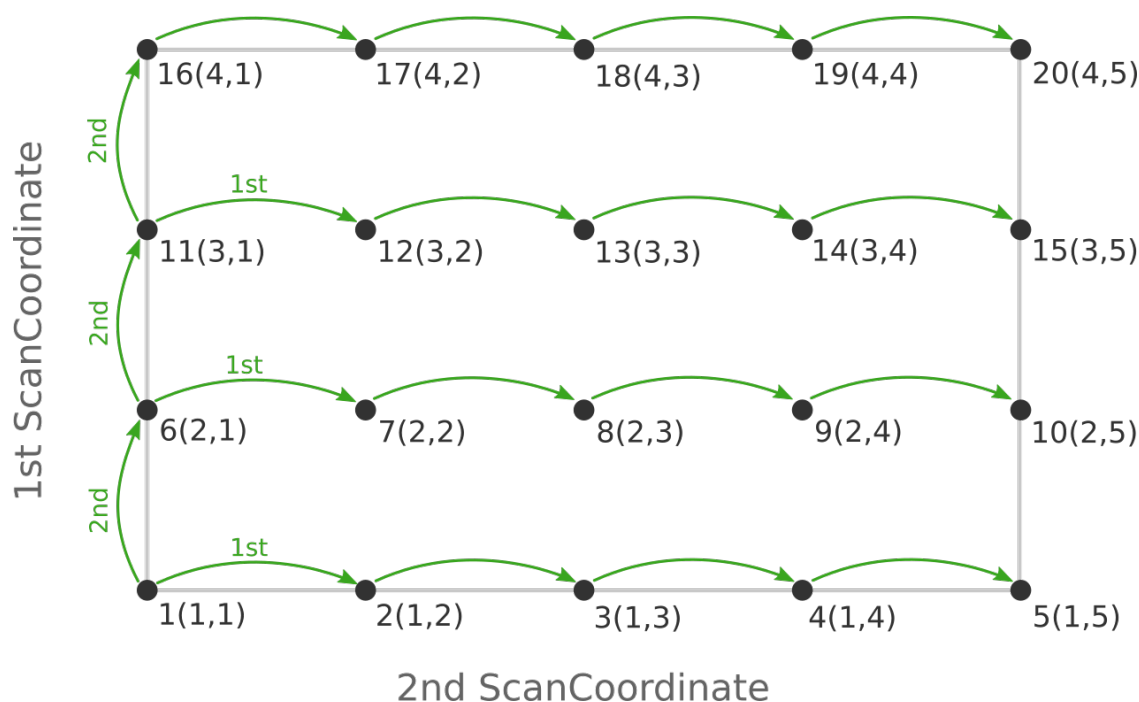
Technically all PES scan calculations are conducted as a series of geometry optimizations with constraints for the scanned coordinates, where the value of the constraint varies slowly through the scanned range. In this way every sampled point on the potential energy surface corresponds to a particular set of constraints. As with any geometry optimization, it can happen that an optimization towards a particular point on the potential energy surface does not converge. This is the most common problem encountered during PES scan calculations.

Since PES scans are implemented as a series of geometry optimizations, they are influenced by the settings used for the geometry optimizer, e.g. its convergence thresholds and the maximum number of steps before an optimization is considered to have failed. The optimizer is configured in the `GeometryOptimization` block, see the page on [geometry optimization](#) (page 49) in the AMS manual.

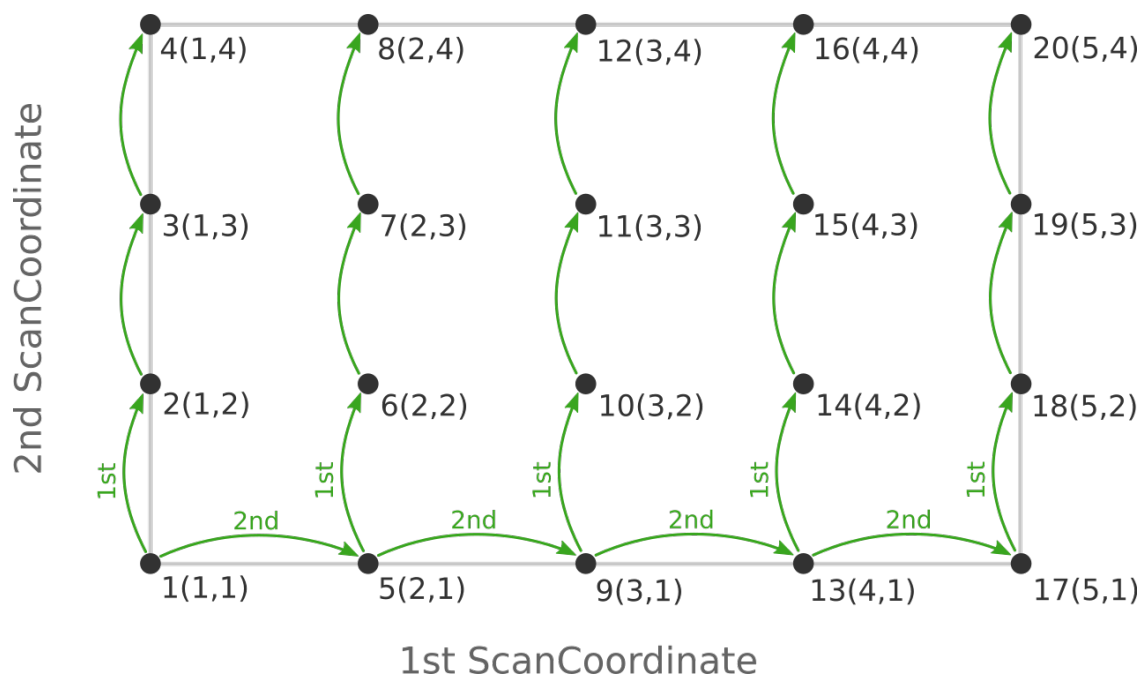
While tweaking the geometry optimizer's settings can sometimes help with convergence problems, these problems can also be easily caused by errors in the user input.

A very common problem is that the geometry in the input, i.e. the `System` block, is incompatible with the starting values of the scanned coordinates. This would for example be the case if one wants to scan a dihedral angle from 0 to 90 degrees, but the actual angle on the input geometry is close to 90 degrees. In this case it would be better to flip the scanned range from 90 to 0 degrees, so that the input geometry already close to the first sampled point on the PES. Otherwise the optimization for the first point has to cross a very long distance on the PES, making convergence much harder. AMS automatically detects this and prints a warning. We generally advise preparing the input geometry for a PES scan by first running a geometry optimization with constraints set to lower bound of the scanned coordinate intervals.

For multidimensional PES scans the order in which the PES points are visited depends on the order in which the scanned coordinates are specified, i.e. the order of the `ScanCoordinate` blocks in the input. Generally, the order in which the PES points are visited is such that the coordinate which was specified in the first `ScanCoordinate` block varies **slowest**. This is illustrated in the following figure:



Here the scan starts at point 1 (1, 1) at the bottom left corner of the PES and first moves along the entire range of the 2nd scan coordinate, before taking a step along the 1st coordinate to point 6 (2, 1). The same PES points could be visited in a different order (and under different names) if the order of the two `ScanCoordinate` blocks is reversed in the AMS input:



Depending on the shape of the scanned potential energy surface a particular order of visiting the PES points might be easier or harder for the optimizer, and convergence problems can sometimes be fixed by simply changing the order of the scanned coordinates. In the example above, it might be that scanning along the “vertical” direction is “harder” than scanning along the “horizontal” direction. In this case one should use the scan order from the first picture, which has only three “vertical” steps (whereas the other scan order has 15).

Note that AMS has a little safe-guard built in to help with PES scan convergence issues: If the optimization towards a particular PES point did not succeed in the initial attempt, AMS will later try again, but starting from a different (converged) point close to unconverged one. This “PES gap filling” happens at the very end of the calculation, after the initial scan has been completed. This gap filling step is enabled by default, but can be controlled with the `PESScan%FillUnconvergedGaps` keyword:

PESScan

FillUnconvergedGaps

Type

Bool

Default value

Yes

Description

After the initial pass over the PES, restart the unconverged points from converged neighboring points.

4.5.7 Output

Results are printed to the text output and stored in the binary result file `ams.rkf`. Most information is found in the `PESScan` section, which references the corresponding geometries from the `History` section via the `PESScan%HistoryIndices` entry.

PESScan

Section content: Data related to the Potential Energy Surface (PES) Scan procedure.

PESScan%GOConverged

Type

bool_array

Description

Whether the (constrained) optimization at the various PES scan points converged.

Shape

[nPoints]

PESScan%HistoryIndices

Type

int_array

Description

The indices of the frames in the History section corresponding to the PES point values.

Shape

[nPoints]

PESScan%HistoryPESPoints

Type

int_array

Description

?

PESScan%nPoints

Type

int

Description

The total number of scanned PES points. This is the product of all `nPoints(#)` values.

PESScan%nPoints (#)

Type

int

Description

Number of points along the corresponding scan coordinate.

PESScan%nScanCoord

Type

int

Description

Number of (independent) coordinates along which the PES scan is performed.

PESScan%PES

Type

float_array

Description

The total energy at each particular PES point.

Unit

hartree

Shape

[nPoints]

PESScan%PESCoords**Type**

float_array

Description

The values of all coordinates for each particular PES point.

Shape

[:, nPoints]

PESScan%RangeEnd (#)**Type**

float_array

Description

The final value(s) for the corresponding scan coordinate.

PESScan%RangeStart (#)**Type**

float_array

Description

The starting value(s) for the corresponding scan coordinate.

PESScan%ScanCoord (#)**Type**

string_fixed_length

Description

A human readable description of the scan coordinate.

4.6 Nudged Elastic Band (NEB)

The Nudged Elastic Band (NEB) method¹ can be used to find a reaction path and the transition state between a reactant and a product state.

At the beginning of a NEB calculation, the geometry of the initial and final systems are optimized to minimize their energy (unless the *OptimizeEnds* (page 86) option is set to False).

Then, a rough approximation of the reaction path is build: a set of *images* is created by performing a linear interpolation between the **initial** and **final** systems. If there is more than two systems provided, a spline interpolation is performed between the initial, all the intermediate (in the lexicographical order) and the final system (see below).

¹ G. Henkelman, B.P. Uberuaga and H. Jonsson, *A climbing image nudged elastic band method for finding saddle points and minimum energy paths*, Journal of Chemical Physics 113, 9901 (2000) (<https://doi.org/10.1063/1.1329672>)

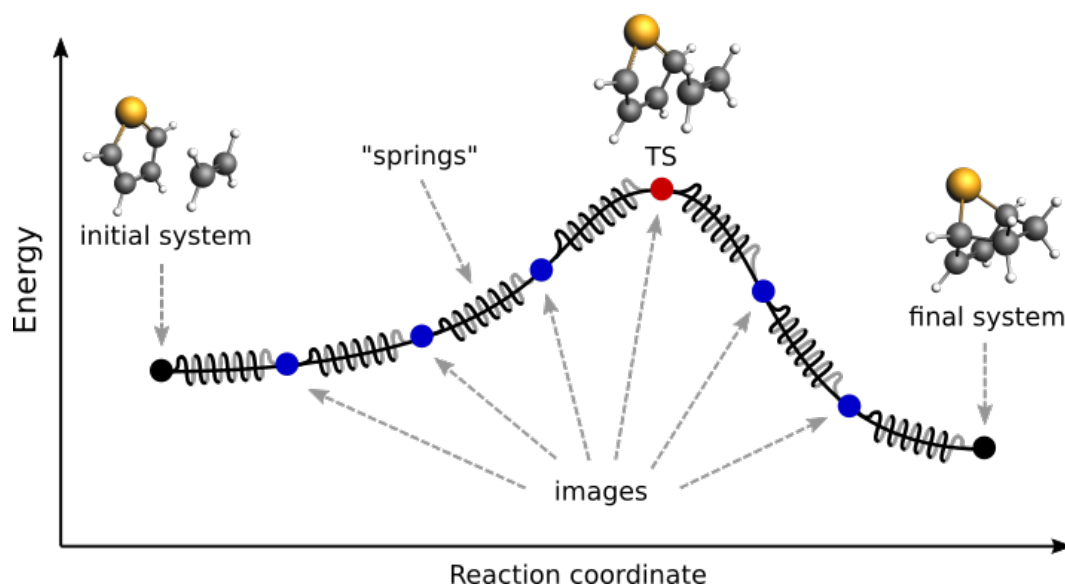


Fig. 4.1: Pictorial representation of a reaction path computed with NEB

Finally, a reaction path is found by performing a simultaneous optimization of all the images. In the NEB method the images are not independent from each other. The force on each image depends on its neighboring images: at each step the forces parallel to the reaction path are eliminated and a so-called spring force is added that tries to keep each image in the middle between its neighbors. This does not let images slide to the initial or final reaction state and ensures that they are evenly distributed along the reaction path.

During the NEB path optimization, a climbing image algorithm is used to drive the highest-energy image in the path to the transition state (unless the *Climbing* (page 86) option is set to False).

Be aware that NEB is a computationally expensive method, typically involving hundreds if not thousands of energy and gradients evaluations.

See also:

- *Nudged Elastic Band (NEB) Examples* (page 485)
- *GUI Nudged Elastic Band tutorial*

4.6.1 Input

A NEB calculation in AMS is triggered by setting the `Task` to NEB:

```
Task NEB
```

The NEB method requires two or more input systems. The first, unnamed system is used as the **initial system** and the system called `final` is used as a **final system**. These two systems are mandatory, unless the `LoadPath` feature is used, see below. This is an example of system-definitions for a HCN isomerization reaction:

```
Task NEB

# This is the initial system:
System
  Atoms
```

(continues on next page)

(continued from previous page)

```

      C   0.0000   0.0000   0.0000
      N   1.1800   0.0000   0.0000
      H   2.1960   0.0000   0.0000
End
End

# This is the final system (note the header 'final' in the next line):
System final
  Atoms
      C   0.0000   0.0000   0.0000
      N   1.1630   0.0000   0.0000
      H  -1.0780   0.0000   0.0000
  End
End

```

Optionally, **more systems can be used to provide a better approximation for the reaction path**. An intermediate system can have any name, but the name determines its position with respect to others. The intermediate systems are ordered lexicographically using their (case-sensitive) names and then placed evenly in that order between the initial and the final states. After that, the spline interpolation between the geometries (including lattice parameters) will be used to generate the initial images. For example, if you call the intermediate systems `a`, `b`, and `c` then the initial NEB path will be constructed by interpolating along the `initial-a-b-c-final` chain. When providing more than two input systems it may be a good idea to optimize the end points in advance and set *OptimizeEnds* (page 86) to `False` to prevent creating an unbalanced reaction path. The reason is that the interpolation is performed after the end points are optimized.

For molecular systems, the interpolation can be done either in the redundant internal or in the Cartesian coordinates. For periodic systems, the interpolation can only be done in the Cartesian coordinates.

When using only two systems (the initial and the final) for linear interpolation, AMS will may optionally apply a slightly modified version of the **image dependent pair potential (IDPP)** method described in², see `PreOptimizeWithIDPP` below. The differences with the original method are: (a) the initial structure of an image is obtained using interpolation in the redundant internal or Cartesian coordinates as described above, and (b) the weighting factor is using the target distance instead of the current one, which avoids putting too much weight on very short distances.

Note that not only the atomic coordinates, but also the lattice parameters and the charge (if non-zero) must be set for all input systems.

Important: The order in which atoms are specified in the `System%Atoms` blocks **must be the same for all systems**. The order of the atoms must be consistent because the images-interpolation algorithm maps the *n*-th atom of the initial system to the *n*-th atom of the final system.

If you have run a PESScan calculation for the system and would like to do NEB, or would like to repeat NEB for it with a different number of images then you can use the `LoadPath` feature. You need to provide the file name from which to load the path using the `File` key. The program will load the last geometries for the path points found in the file and use them for interpolation. Note that this will replace both the initial and the final point. Neither the initial nor the final geometry needs to be present in the input in this case because, if present, they will be replaced.

You can choose to use only part of the path, for example if the the PESScan revealed more than one barrier or if its end points do not correspond to minima on the path. In such a case, a list of point to be included in the interpolation should be specified using the `Points` key. Remember that you can specify a list of integers as `i:j`, for example to include points 13 through 25 use `Points 13:25`.

2

S. Smidstrup, A. Pedersen, K. Stokbro and H. Jonsson, *Improved initial guess for minimum energy path calculations*, *Journal of Chemical Physics* 140, 214106 (2014) (<http://doi.org/10.1063/1.4878664>)

Alternatively, you can choose to use arbitrary geometries from the History section of the ams.rkf file. In this case you can specify their indices in the `Geometries` key. When using the `Geometries` key, the ams.rkf does not have to come from a PESScan or an NEB calculation. It just needs to have the History section with some geometries. It may come, for example, from molecular dynamics.

NEB**LoadPath****Type**

Block

Description

Provide details about the trajectory to get the initial NEB path from. PESScan and NEB trajectories are supported. Only the last geometry for each point on the trajectory is considered.

File**Type**

String

GUI name

Initial path file

Description

Provide an ams.rkf file to load the initial path from. All geometries of this calculation, including initial and final, will be taken from the History section of the file.

Note that for a PESScan it should be a 1D path.

Geometries**Type**

Integer List

GUI name

Raw geometry indices

Description

Raw indices of the geometries from the History section. By default the last geometry of each path point is used.

Points**Type**

Integer List

GUI name

Path points

Description

By default the whole path is used, which may sometimes be not desirable. For example when a PESScan revealed multiple barriers. In this case one can specify indices of the path points to be used. The last geometry of the specified path point will be loaded.

All NEB-specific options are specified in the NEB input block:

```
NEB
Climbing Yes/No
ClimbingThreshold float
Images integer
InterpolateInternal Yes/No
InterpolateShortest Yes/No
```

(continues on next page)

(continued from previous page)

```

Iterations integer
Jacobian float
LoadPath
    File string
    Geometries integer_list
    Points integer_list
End
MapAtomsToCell Yes/No
OldTangent Yes/No
OptimizeEnds Yes/No
OptimizeLattice Yes/No
Parallel
    nCoresPerGroup integer
    nGroups integer
    nNodesPerGroup integer
End
PreOptimizeWithIDPP Yes/No
ReOptimizeEnds Yes/No
Restart string
Skewness float
Spring float
End

```

All keys of the NEB block have reasonable defaults or are optional. Thus, in principle, the NEB block can be omitted altogether. These are the main options:

NEB**Images****Type**

Integer

Default value

8

GUI name

Number of images

Description

Number of NEB images (not counting the chain ends). Using more images will result in a smoother reaction path and can help with convergence problems, but it will also increase the computation time.

Iterations**Type**

Integer

GUI name

Maximum number of iterations

Description

Maximum number of NEB iterations. The default value depends on the number of degrees of freedom (number of images, atoms, periodic dimensions).

Spring**Type**

Float

Default value

1.0

UnitHartree/Bohr²**GUI name**

Spring value

Description

Spring force constant in atomic units.

Skewness**Type**

Float

Default value

1.0

GUI name

Skewness

Description

Degree of how much images are shifted towards or away from the TS, which may help tackle problems with a long reaction path (for example involving a loose adsorption complex) without needing too many images. A value greater than 1 will make sure that images are concentrated near the transition state. The optimal value depends on the path length, the number of images (larger [Skewness] may be needed for a longer path and fewer images). Technically [Skewness] is equal to the ratio between the optimized distances to the lower and the higher neighbor image on the path.

Climbing**Type**

Bool

Default value

Yes

GUI name

Climb highest image to TS

Description

Use the climbing image algorithm to drive the highest image to the transition state.

ClimbingThreshold**Type**

Float

Default value

0.0

Unit

Hartree/Bohr

GUI name

CI force threshold

Description

Climbing image force threshold. If ClimbingThreshold > 0 and the max perpendicular force component is above the threshold then no climbing is performed at this step. This entry can

be used to get a better approximation for the reaction path before starting the search for the transition state. A typical value is 0.01 Hartree/Bohr.

InterpolateInternal

Type

Bool

Default value

Yes

GUI name

Interpolate in Internal coordinates

Description

The initial NEB image geometries are calculated by interpolating between the initial and the final state. By default, for non-periodic systems the interpolation is performed in internal coordinates but the user can choose to do it in the Cartesian ones. For periodic systems the interpolation is always done in Cartesian coordinates. If PreOptimizeWithIDPP is set then the path may be further refined using the image-dependent pair potential (IDPP).

InterpolateShortest

Type

Bool

Default value

Yes

GUI name

Interpolate across cell boundary

Description

Allow interpolation across periodic cell boundaries. Set to false if an atom is intended to move more than half across the simulation box during reaction.

PreOptimizeWithIDPP

Type

Bool

Default value

No

GUI name

Use IDPP

Description

(Experimental)

When there is only initial and final system available, the image-dependent pair potential (IDPP, doi: 10.1063/1.4878664) can be used to determine the initial NEB path by interpolating all interatomic distances between the two points and optimizing intermediate images towards them. The optimization starts from the geometries obtained using the selected interpolation options.

OptimizeEnds

Type

Bool

Default value

Yes

GUI name

Optimize reactants/products

Description

Start the NEB with optimization of the reactant and product geometries.

Restart**Type**

String

GUI name

Restart from

Description

Provide an ams.rkf file from a previous NEB calculation to restart from. It can be an unfinished NEB calculation or one performed with different engine parameters.

ReOptimizeEnds**Type**

Bool

Default value

No

GUI name

Re-optimize reactants/products

Description

Re-optimize reactant and product geometries upon restart.

The following keys are related to solid-state NEB (SS-NEB):

NEB**OptimizeLattice****Type**

Bool

Default value

No

GUI name

Optimize lattice

Description

Turn on the solid-state NEB (SS-NEB).

Jacobian**Type**

Float

GUI name

Jacobian value

DescriptionScaling factor used to convert the lattice strain to a NEB coordinate value. Default value: $\sqrt{N} \cdot (V/N)^{1/d}$, where V - lattice volume (area for 2D, length for 1D), N - number of atoms, and d - number of periodic dimensions.**MapAtomsToCell**

Type

Bool

Default value

Yes

GUI name

Map atoms to cell

Description

Translate atoms to the $[-0.5, 0.5]$ cell before every step. This option cannot be disabled for SS-NEB.

At each iteration, the images may be computed in parallel. The parallel execution is normally configured completely automatically, but users can override the automatic parallelization using the keys in the `Parallel` block.

NEB**Parallel****Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

The following keys modify other aspects of the NEB and should, in principle, be left to their defaults:

NEB**OldTangent****Type**

Bool

Default value

No

GUI name

Use old tangent

Description

Turn on the old central difference tangent.

4.6.2 Frozen atom constraints

It is possible to perform NEB with part of the system frozen, using any of the following keys of the Constraints block or a combination thereof:

Constraints**Atom****Type**

Integer

Recurring

True

Description

Fix the position of an atom. Just one integer referring to the index of the atom in the [System%Atoms] block.

AtomList**Type**

Integer List

Recurring

True

Description

Fix positions of the specified atoms. A list of integers referring to indices of atoms in the [System%Atoms] block.

FixedRegion**Type**

String

Recurring

True

Description

Fix positions of all atoms in a region.

Note: the frozen atom constraints will be enforced both during the geometry optimizations of the initial and final systems and during the NEB optimization.

4.6.3 Optimizations and convergence criteria

The NEB path is optimized using a limited-memory BFGS (l-BFGS) method where the system being optimized is a union of all NEB images with their respective molecular and spring forces.

The NEB convergence thresholds are defined in the *GeometryOptimization%Convergence* block (page 50). NEB is considered converged when the following criteria are satisfied:

- the change in the highest image energy must be less than [GeometryOptimization%Convergence%Energy]
- the max atomic force component for the highest image must be less than [GeometryOptimization%Convergence%Gradients]
- the max atomic force component for all other images must be less than ten times the [GeometryOptimization%Convergence%Gradients] value.

If the optimization of the initial NEB end point fails to converge, you can try using the *FIRE optimization method* (page 59).

4.6.4 Output

Results are printed to the text output and stored in the binary result file 'ams.rkf'. In the 'ams.rkf' file, NEB calculation results are stored in the History section just like in a normal geometry optimization. The NEB section of the RKF file contains additional, NEB-specific, information.

The NEB reaction path can be visualized using the **AMSmovie GUI module**.

4.6.5 Troubleshooting

- In case the geometry optimization of the initial and final systems fails: try using the *FIRE optimization method* (page 59)
- In case the optimization of the NEB path does not converge:
 - make sure that the order in which the atoms are defined is consistent between the initial and final systems (see the *important* note in the *NEB input section* (page 84));
 - try increasing the number of *NEB images* (page 86);
 - try tweaking the *TrustRadius* (page 66) or *TrialStep* (page 66) options (see *Limited-memory BFGS* (page 66)) ;
 - try specifying one or more **intermediate** systems.

4.7 Intrinsic Reaction Coordinate (IRC)

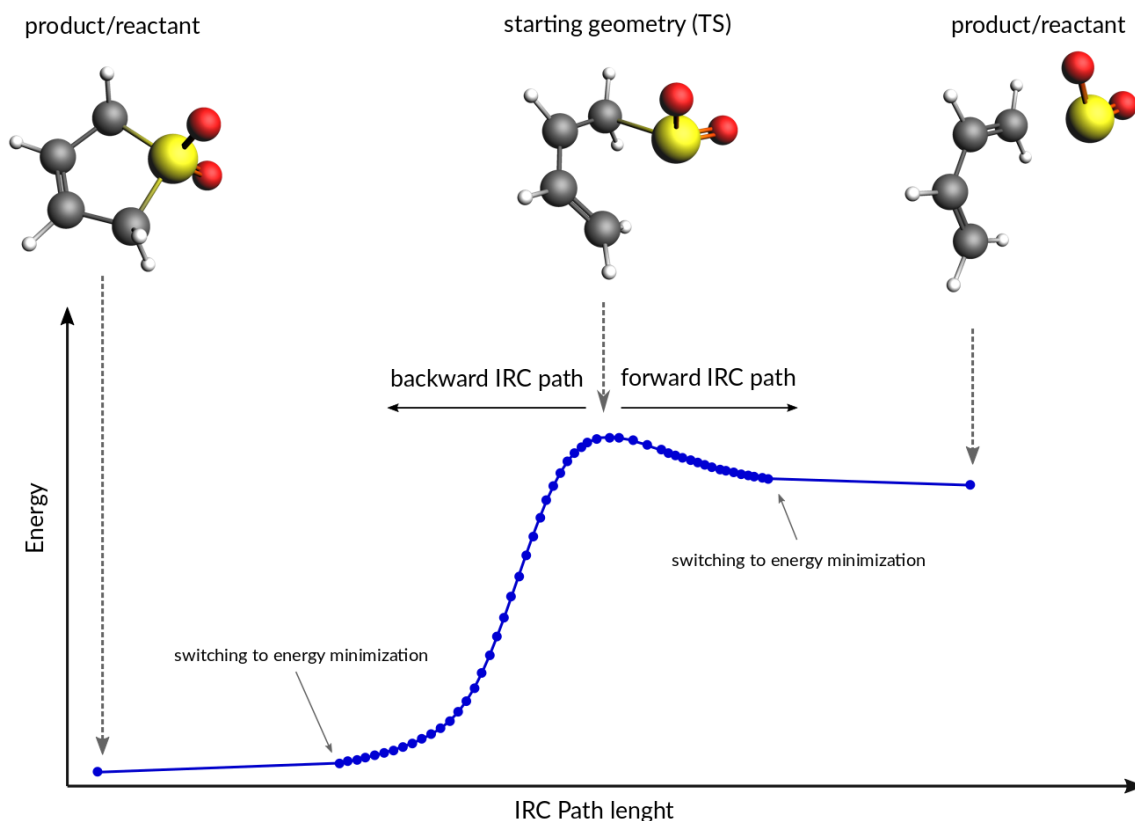
The path of a chemical reaction can be traced from the transition state (TS) to the products and/or reactants using the Intrinsic Reaction Coordinate (IRC) method^{1,2}. **The method assumes that the starting geometry is a fair approximation of the TS.** A minimum energy profile (MEP) is defined as the steepest-descent path on the potential energy surface from the transition state down towards a local minimum. An IRC path is defined similarly but in the mass-weighted coordinates³, which means that instead of the steepest descent direction it follows that of the maximum instantaneous

¹ L. Deng, T. Ziegler and L. Fan, *A combined density functional and intrinsic reaction coordinate study on the ground state energy surface of H₂ CO*, *Journal of Chemical Physics* 99, 3823 (1993) (<https://doi.org/10.1063/1.466129>)

² L. Deng and T. Ziegler, *The determination of Intrinsic Reaction Coordinates by density functional theory*, *International Journal of Quantum Chemistry* 52, 731 (1994) (<https://doi.org/10.1002/qua.560520406>)

³ C. Gonzalez and H.B. Schlegel, *Reaction Path Following In Mass-Weighted Internal Coordinates* *J. Phys. Chem.* 94, 5523-5527 (1990) (<https://doi.org/10.1021/j100377a021>)

acceleration. This makes IRC somewhat related to the Molecular Dynamics method. The energy profile is obtained as well as the length and curvature properties of the path, providing the basic quantities for an analysis of the reaction path.



See also:

[Examples](#) (page 457) and the [PES exploration for hydrohalogenation tutorial](#)

4.7.1 Method details

Calculation of an IRC path consists of two nested loops, the so-called outer and inner loops. The outer loop runs over IRC points and the inner loop is over geometry optimization steps for the given IRC point. The first IRC point starts from the transition state geometry, which is a saddle point, in one of the two possible downhill directions. Each IRC point after that starts from the optimized geometry of the previous point. At the start of every step, the pivot point is determined, which is a point at the Step/2 distance in the direction opposite to the gradient. When working in the mass-weighted coordinates, this direction corresponds to the acceleration of the corresponding atom. The final point of the given IRC step corresponds to the energy minimum point at the same distance (Step/2) from the pivot point further downhill. More precisely, the coordinates of the target point are optimized during the inner loop to minimize projection of the gradient on the hypersphere of radius Step/2 around the pivot point. The angle between the (pivot-start) and (pivot-final) vectors determines the curvature of the reaction path. If this angle becomes smaller than 90 degrees then the IRC scan is considered to have reached vicinity of an endpoint and the program switches to energy minimization (options for this energy minimization can be specified in the [Geometry Optimization](#) (page 50) block.). If the angle is between 90 and 120 degrees then the current IRC step is canceled and a new one is started from the same starting point with half the initial Step parameter. In all other cases the optimized geometry becomes a starting point for the next IRC step. By default, when the forward path is completed the backward one is started from the same TS geometry. When both forward and backward paths are complete, a summary of the whole reaction path is printed to the output.

4.7.2 Input

The IRC scan in AMS is triggered by setting the Task to IRC:

```
Task IRC
```

All IRC-related options are specified in the IRC input block:

```
IRC
  Convergence
    Gradients float
    Step float
  End
  CoordinateType [Cartesian | Delocalized]
  Direction [Both | Forward | Backward]
  InitialHessian
    File string
    Type [Calculate | FromFile]
  End
  KeepConvergedResults Yes/No
  MaxIRCSteps integer
  MaxIterations integer
  MaxPoints integer
  MinEnergyProfile Yes/No
  MinPathLength float
  Restart
    File string
    RedoBackward integer
    RedoForward integer
  End
  Step float
End
```

All keys of the IRC block have reasonable defaults or are optional. Thus, in principle, the IRC block can be omitted altogether. These are some of the main options:

IRC

Direction

Type

Multiple Choice

Default value

Both

Options

[Both, Forward, Backward]

Description

Select direction of the IRC path. The difference between the Forward and the Backward directions is determined by the sign of the largest component of the vibrational normal mode corresponding to the reaction coordinate at the transition state geometry. The Forward path correspond to the positive sign of the component. If Both is selected then first the Forward path is computed followed by the Backward one.

Step

Type

Float

Default value

0.2

GUI name

Step size

Description

IRC step size in mass-weighted coordinates, $\text{sqrt}(\text{amu}) \cdot \text{bohr}$. One may have to increase this value when heavy atoms are involved in the reaction, or decrease it if the reactant or products are very close to the transition state.

InitialHessian**Type**

Block

Description

Options for initial Hessian at the transition state. The first eigenvalue of the initial Hessian defines direction of the first forward or backward step. This block is ignored when restarting from a previous IRC calculation because the initial Hessian found in the restart file is used.

File**Type**

String

GUI name

File

Description

If 'Type' is set to 'FromFile' then in this key you should specify the RKF file containing the initial Hessian (or the ams results dir. containing it). This can be used to load a Hessian calculated previously with the 'Properties%Hessian' keyword. If you want to also use this file for the initial geometry then also specify it in a 'LoadSystem' block.

Type**Type**

Multiple Choice

Default value

Calculate

Options

[Calculate, FromFile]

GUI name

Initial Hessian

Description

Calculate the exact Hessian for the input geometry or load it from the results of a previous calculation.

The following keys set limits on the number of steps for the inner and outer IRC loops and, related to that, the geometry optimization criteria. Note that tighter criteria may require a greater MaxIterations limit. Please also note that the outer loop limits are valid for each half of the path (forward and backward) separately. That is, if all settings are left at their defaults then up to 200 IRC points may be calculated, each of them may require up to 300 energy evaluations.

IRC**MaxIRCSteps**

Type

Integer

GUI name

Maximum IRC steps

Description

Soft limit on the number of IRC points to compute in each direction. After the specified number of IRC steps the program will switch to energy minimization and complete the path. This option should be used when you are interested only in the reaction path area near the transition state. Note that even if the soft limit has been hit and the calculation has completed, the IRC can still be restarted with a 'RedoBackward' or 'RedoForward' option.

MaxPoints**Type**

Integer

Default value

100

GUI name

Maximum points

Description

Hard limit on the number of IRC points to compute in each direction. After the specified number of IRC steps the program will stop with the current direction and switch to the next one. If both 'MaxPoints' and 'MaxIRCSteps' are set to the same value then 'MaxPoints' takes precedence, therefore this option should be used to set a limit on the number of IRC steps if you intend to use the results later for a restart.

MaxIterations**Type**

Integer

Default value

300

GUI name

Maximum iterations

Description

The maximum number of geometry iterations allowed to converge the inner IRC loop. If optimization does not converge within the specified number of steps, the calculation is aborted.

Convergence**Type**

Block

Description

Convergence at each given point is monitored for two items: the Cartesian gradient and the calculated step size. Convergence criteria can be specified separately for each of these items. The same criteria are used both in the inner IRC loop and when performing energy minimization at the path ends.

Gradients**Type**

Float

Default value

0.001

Unit

Hartree/Angstrom

GUI name

Gradient convergence

Description

Convergence criterion for the max component of the residual energy gradient.

Step**Type**

Float

Default value

0.001

Unit

Angstrom

GUI name

Step convergence

Description

Convergence criterion for the max component of the step in the optimization coordinates.

MinPathLength**Type**

Float

Default value

0.1

Unit

Angstrom

Description

Minimum length of the path required before switching to energy minimization. Use this to overcome a small kink or a shoulder on the path.

The following keys modify other aspects of the IRC scan:

IRC**CoordinateType****Type**

Multiple Choice

Default value

Cartesian

Options

[Cartesian, Delocalized]

GUI name

Coordinates used for optimization

Description

Select the type of coordinates in which to perform the optimization. Note that the Delocalized option should be considered experimental.

MinEnergyProfile**Type**

Bool

Default value

No

GUI name

Minimum energy profile

Description

Calculate minimum energy profile (i.e. no mass-weighting) instead of the IRC.

KeepConvergedResults**Type**

Bool

Default value

Yes

Description

Keep the binary RKF result file for every converged IRC point. These files may contain more information than the main ams.rkf result file.

It is possible to restart an IRC calculation that crashed, has been killed or exceeded the MaxPoints limit, or to re-compute the path starting from a certain point, using the Restart key:

IRC**Restart****Type**

Block

Description

Restart options. Upon restart, the information about the IRC input parameters and the initial system (atomic coordinates, lattice, charge, etc.) is read from the restart file. The IRC input parameters can be modified from input. Except for 'MaxPoints' and 'Direction' all parameters not specified in the input will use their values from the restart file. The 'MaxPoints' and 'Direction' will be reset to their respective default values if not specified in the input. By default, the IRC calculation will continue from the point where it left off. However, the 'RedoForward' and/or 'RedoBackward' option can be used to enforce recalculation of a part of the reaction path, for example, using a different 'Step' value.

File**Type**

String

GUI name

Restart

Description

Name of an RKF restart file generated by a previous IRC calculation. Do not use this key to provide an RKF file generated by a TransitionStateSearch or a SinglePoint calculation, use the 'LoadSystem' block instead.

RedoBackward**Type**

Integer

Default value

0

Description

IRC step number to start recalculating the backward path from. By default, if the backward path has not been completed then start after the last completed step. If the backward path has been completed and the 'RedoBackward' is omitted then no point on the backward path will be recomputed.

RedoForward**Type**

Integer

Default value

0

Description

IRC step number to start recalculating the forward path from. By default, if the forward path has not been completed then start after the last completed step. If the forward path has been completed and the 'RedoForward' is omitted then no point on the forward path will be recomputed.

4.7.3 Output

A summary of reaction path is printed to the output file at the end of the IRC calculation.

The IRC reaction path can be visualized using the AMSmovie GUI module.

Results of an IRC calculation are also stored in the History section of the 'ams.rkf' file, just like in a normal geometry optimization. In addition to the standard KF variables such as "Coords" and "Energy", the following IRC-specific variables are also created:

- *IRCDirection* - IRC direction to which this point belongs: 1 - forward, 2 - backward.
- *IRCIteration* - the IRC (a.k.a. the outer loop) iteration number.
- *OptIteration* - the geometry optimization (a.k.a. the inner loop) iteration number (0 means the results correspond to the converged geometry at this IRC step).
- *IRCGradMax* - value of the max component of the IRC gradient that determines convergence of the inner loop.
- *IRCGradRms* - the RMS value of the IRC gradient that determines convergence of the inner loop. Both the *ircGradRms* and the *ircGradMax* are given in the mass-weighted atomic units for IRC steps and in the atomic units for the final minimization loop.
- *ArcLength* - length, in Angstrom, of the arc that connects the initial and the final point of this IRC step. The corresponding pivot point is located near the the middle point of the arc.
- *Angle* - value of the angle (in degrees) between lines connecting the pivot point with the initial and final points. A value of 180 degrees means the path is passing straight through the pivot point, while a smaller value means the path makes a bend at this point.
- *PathLength* - sum of the *ArcLength* values from the transition state up to this point, in Angstrom.
- *Converged* - a Fortran logical value containing the convergence status of the given geometry.

The IRC section of the RKF file contains all the data needed for a successful restart procedure.

4.8 Excited state optimizations

It depends on the engine if it is possible to do an electronically excited state optimization, like an electronically excited state *geometry optimization* (page 49), *transition state search* (page 72), *linear transit*, *PES scan* (page 76), or *IRC* (page 93). Required is that the engine should be able to calculate the nuclear gradient for a particular electronically excited state. ADF and DFTB are engines that can do such calculation. If the excited state gradient is available one can also calculate an *IR spectrum* (page 247) of the excited state, and one could calculate a *vibrationally resolved electronic spectrum* (page 286).

One should look in the documentation of the engine how to set up the calculation of the excited state gradient.

4.9 Molecular dynamics

Molecular dynamics (MD) can be used to simulate the evolution of a system in time.

See also:

- [Examples](#) (page 457)
- [AMS GUI Tutorial](#)
- [MD trajectory analysis tool](#) (page 415)

To perform a MD simulation, first select the corresponding Task:

Task MolecularDynamics

All aspects of the simulation can then be configured using the MolecularDynamics block.

```
MolecularDynamics
  AddMolecules
    AtomTemperature float
    ContactDistance float
    Coords float_list
    CoordsBox float_list
    CoordsSigma float_list
    DeviationAngle float
    Energy float
    EnergySigma float
    FractionalCoords float_list
    FractionalCoordsBox float_list
    FractionalCoordsSigma float_list
    Frequency integer
    MinDistance float
    MoleFraction float
    NumAttempts integer
    Rotate Yes/No
    StartStep integer
    StopStep integer
    System string
    Temperature float
    TemperatureSigma float
    Velocity float
    VelocityDirection float_list
    VelocitySigma float
  End
  ApplyForce
    Force float_list
```

(continues on next page)

(continued from previous page)

```

    PerAtom Yes/No
    Region string
End
ApplyVelocity
    Components [X | Y | Z | XY | YZ | XZ | XYZ]
    Region string
    Velocity float_list
End
Barostat
    BulkModulus float
    ConstantVolume Yes/No
    Duration integer_list
    Equal [None | XYZ | XY | YZ | XZ]
    Pressure float_list
    Scale [XYZ | Shape | X | Y | Z | XY | YZ | XZ]
    Tau float
    Type [None | Berendsen | MTK]
End
BinLog
    BiasEnergy Yes/No
    BoostFactor Yes/No
    ConservedEnergy Yes/No
    Density Yes/No
    DipoleMoment Yes/No
    Hypertime Yes/No
    KineticEnergy Yes/No
    MaxBiasEnergy Yes/No
    MaxBoostFactor Yes/No
    PotentialEnergy Yes/No
    Pressure Yes/No
    PressureTensor Yes/No
    Step Yes/No
    Temperature Yes/No
    Time Yes/No
    TotalEnergy Yes/No
    Volume Yes/No
End
BondBoost
    Chain
        AtomNames string
        MaxDistances float_list
        MinDistances float_list
    End
    DistanceRestraint string
    Moving Yes/No
    NSteps integer
    NumInstances integer
    Units [Default | MD]
End
CRESTMTD
    AddEnergy Yes/No
    GaussianScaling
        ScaleGaussians Yes/No
        ScalingSlope float
    End
    Height float
    NGaussiansMax integer

```

(continues on next page)

(continued from previous page)

```

    NSteps integer
    Region string
    RestartFile string
    Width float
End
CVHD
    Bias
        DampingTemp float
        Delta float
        Height float
    End
    ColVarBB
        at1
            region string
            symbol string
        End
        at2
            region string
            symbol string
        End
        cutoff float
        p integer
        rmax float
        rmin float
    End
    ColVarBM
        at1
            region string
            symbol string
        End
        at2
            region string
            symbol string
        End
        cutoff float
        p integer
        rmax float
        rmin float
    End
    Frequency integer
    MaxEvents integer
    StartStep integer
    StopStep integer
    WaitSteps integer
End
CalcPressure Yes/No
Checkpoint
    Frequency integer
    WriteProperties Yes/No
End
CopyRestartTrajectory Yes/No
CosineShear
    Acceleration float
    Enabled Yes/No
    FlowDirection float_list
    ProfileAxis [X | Y | Z]
End

```

(continues on next page)

(continued from previous page)

```

Deformation
  LatticeVelocity # Non-standard block. See details.
  ...
End
  LengthRate float_list
  LengthVelocity float_list
  Period float
  ScaleAtoms Yes/No
  StartStep integer
  StopStep integer
  StrainRate # Non-standard block. See details.
  ...
End
  TargetDensity float
  TargetLattice # Non-standard block. See details.
  ...
End
  TargetLength float_list
  Type [Linear | Exponential | Sine | Cosine]
End
Gravity
  Acceleration float
End
HeatExchange
  HeatingRate float
  Method [Simple | HEX | eHEX]
  Sink
    AtomList integer_list
    Box
      Amax float
      Amin float
      Bmax float
      Bmin float
      Cmax float
      Cmin float
    End
    Region string
  End
  Source
    AtomList integer_list
    Box
      Amax float
      Amin float
      Bmax float
      Bmin float
      Cmax float
      Cmin float
    End
    Region string
  End
  StartStep integer
  StopStep integer
End
InitialVelocities
  File string
  RandomVelocitiesMethod [Exact | Boltzmann | Gromacs]
  Temperature float

```

(continues on next page)

(continued from previous page)

```

    Type [Zero | Random | FromFile | Input]
    Values # Non-standard block. See details.
    ...
End
End
MovingRestrains
  Change [Linear | Sine | Cosine]
  Distance
    Atom1 integer
    Atom2 integer
    EndValue float
    StartValue float
  End
  Erf
    ForceConstant float
    MaxForce float
  End
  GaussianWell
    Sigma float
    WellDepth float
  End
  Harmonic
    ForceConstant float
  End
  Hyperbolic
    ForceConstant float
    MaxForce float
  End
  Name string
  Period float
  RestraintType [None | Harmonic | Hyperbolic | Erf | GaussianWell]
  StartStep integer
  StopStep integer
End
NSteps integer
Plumed
  Input # Non-standard block. See details.
  ...
End
Parallel
  nCoresPerGroup integer
  nGroups integer
  nNodesPerGroup integer
End
End
Preserve
  AngularMomentum Yes/No
  CenterOfMass Yes/No
  Momentum Yes/No
End
Print
  System Yes/No
  Velocities Yes/No
End
ReactionBoost
  BondBreakingRestrains
    Erf

```

(continues on next page)

(continued from previous page)

```

        ForceConstant float
        MaxForce float
    End
    GaussianWell
        Sigma float
        WellDepth float
    End
    Harmonic
        ForceConstant float
    End
    Hyperbolic
        ForceConstant float
        MaxForce float
    End
    Taper
        Enabled Yes/No
        MaxDistance float
        MinDistance float
    End
    Type [None | Harmonic | Hyperbolic | Erf | GaussianWell]
End
BondMakingRestrains
    Erf
        ForceConstant float
        MaxForce float
    End
    GaussianWell
        Sigma float
        WellDepth float
    End
    Harmonic
        ForceConstant float
    End
    Hyperbolic
        ForceConstant float
        MaxForce float
    End
    Type [None | Harmonic | Hyperbolic | Erf | GaussianWell]
End
BondedRestrains
    Harmonic
        ForceConstant float
    End
    Type [None | Harmonic]
End
Change [TargetCoordinate | Force | LogForce]
InitialFraction float
InterEquilibrationSteps integer
MinBondChange float
MinBondStrength float
NSteps integer
NonBondedRestrains
    Exponential
        Epsilon float
        Sigma float
    End
    Type [None | Exponential]

```

(continues on next page)

(continued from previous page)

```

End
PreEquilibrationSteps integer
RMSDRestraint
  Erf
    ForceConstant float
    MaxForce float
  End
  GaussianWell
    Sigma float
    WellDepth float
  End
  Harmonic
    ForceConstant float
  End
  Hyperbolic
    ForceConstant float
    MaxForce float
  End
  Type [None | Harmonic | Hyperbolic | Erf | GaussianWell]
End
Region string
TargetSystem string
Type [None | Pair | RMSD]
End
Reactor
  ForceConstant float
  MassScaled Yes/No
  NSteps integer
  Radius float
End
ReflectiveWall
  Axis float_list
  Region string
  Threshold float
End
Remap
  Range [Auto | ZeroToOne | PlusMinusHalf | AroundCenter]
  Type [None | Atoms]
End
RemoveMolecules
  Formula string
  Frequency integer
  SafeBox
    Amax float
    Amin float
    Bmax float
    Bmin float
    Cmax float
    Cmin float
    FractionalCoordsBox float_list
  End
SinkBox
  Amax float
  Amin float
  Bmax float
  Bmin float
  Cmax float
  Cmin float

```

(continues on next page)

(continued from previous page)

```
Cmin float
FractionalCoordsBox float_list
End
StartStep integer
StopStep integer
End
ReplicaExchange
  AllowWrongResults Yes/No
  EWMALength integer
  SwapFrequency integer
  TemperatureFactors float_list
  Temperatures float_list
  nReplicas integer
End
Restart string
Shake
  All string
  ConvergeR2 float
  ConvergeRV float
  Iterations integer
  ShakeInitialCoordinates Yes/No
End
Thermostat
  BerendsenApply [Local | Global]
  ChainLength integer
  Duration integer_list
  Region string
  Tau float
  Temperature float_list
  Type [None | Berendsen | NHC]
End
TimeStep float
Trajectory
  EngineResultsFreq integer
  ExitConditionFreq integer
  PrintFreq integer
  SamplingFreq integer
  TProfileGridPoints integer
  WriteBonds Yes/No
  WriteCharges Yes/No
  WriteCoordinates Yes/No
  WriteEngineGradients Yes/No
  WriteMolecules Yes/No
  WriteVelocities Yes/No
End
fbMC
  Frequency integer
  MassRoot float
  MolecularMoves
    Enabled Yes/No
    RotationStepAngle float
    TranslationStepLength float
  End
  NSteps integer
  PrintFreq integer
  StartStep integer
  StepLength float
```

(continues on next page)

(continued from previous page)

```

    StopStep integer
    Temperature float
End
End

```

4.9.1 General

The time evolution of the system is simulated by numerically integrating the equations of motion. A velocity Verlet integrator is used with a time step set by the `TimeStep` key. The MD driver will perform `NSteps` timesteps in total.

Because the overall computational cost depends on `NSteps` but not on `TimeStep`, it is desirable to set the timestep as large as possible to maximize the sampled timescale with a given computational budget. However, numerical integration errors grow rapidly as the timestep increases. These errors will cause a loss of energy conservation, crashes, and other artifacts. It is thus important to set the `TimeStep` value carefully, as its optimal value strongly depends on the studied system and simulated conditions.

As a rule of thumb, reasonable timesteps for systems not undergoing chemical reactions are 10-20 times lower than the period of the fastest vibration mode. Systems containing hydrogen atoms at room temperature can thus be accurately simulated using a 1 fs timestep. Longer timesteps can be safely used for systems containing only heavy atoms (vibration periods scale with the square root of the atomic mass). Conversely, the timestep needs to be made shorter for high-temperature simulations. The same also applies to simulations of chemical reactions, which are usually accompanied by significant transient local heating. The default timestep of 0.25 fs should work for most of these cases.

MolecularDynamics

NSteps

Type

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

TimeStep

Type

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

During a long simulation, numerical integration errors will cause some system-wide quantities to drift from their exact values. For example, the system may develop a nonzero net linear velocity, causing an overall translation or flow. Non-periodic (molecular) and 1D-periodic systems may also develop nonzero angular momentum (overall rotation) and a Brownian motion of their center of mass through space. These problems are corrected by periodically removing any accumulated drift. This feature can be controlled using the `Preserve` key.

MolecularDynamics**Preserve****Type**

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and 3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass**Type**

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum**Type**

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

4.9.2 Constrained molecular dynamics

There are two types of constraint available for MD: frozen atoms and Shake/Rattle. Atoms can be kept frozen using the `Atom`, `AtomList` and `FixedRegion` keys of the `Constraints` top-level input block.

Constraints

Atom

Type

Integer

Recurring

True

Description

Fix the position of an atom. Just one integer referring to the index of the atom in the `[System%Atoms]` block.

AtomList

Type

Integer List

Recurring

True

Description

Fix positions of the specified atoms. A list of integers referring to indices of atoms in the `[System%Atoms]` block.

FixedRegion

Type

String

Recurring

True

Description

Fix positions of all atoms in a region.

The Shake/Rattle constraints are implemented following [H.C. Andersen, J Comp. Phys., 52 \(1983\) 24](https://doi.org/10.1016/0021-9991(83)90014-1) ([https://doi.org/10.1016/0021-9991\(83\)90014-1](https://doi.org/10.1016/0021-9991(83)90014-1)). The procedure consists of two steps: update of the coordinates to satisfy the constraints (the Shake part) and orthogonalization of atomic velocities to them (the Rattle part). The Shake part is implemented as an additional correcting force that brings each bond length exactly to its constrained value upon the subsequent velocity Verlet integration step. Application of the constraints is an iterative procedure and the convergence greatly depends on how intertwined the constraints are. For diatomic molecules, usually no more than two iterations are necessary, while tens of iterations may be required for rings (a triangle constraint is treated as a 3-membered ring).

Note: During simulations with a changing simulation box (NpT, NpH), the absolute Cartesian coordinates of the frozen atoms cannot be kept fixed. In this case, their fractional cell coordinates are maintained at the original values. Also the Shake constraints cannot be satisfied exactly during such simulations but the difference is usually small enough.

The Shake/Rattle constraints are specified using the Shake input block:

MolecularDynamics

Shake

Type

Block

Description

Parameters of the Shake/Rattle algorithm.

A11**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, "All triangles H C H" in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2**Type**

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV**Type**

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations**Type**

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates**Type**

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Note: The list of constrained bonds is built by matching the input against the *bonds as specified* (page 38) in the *System block* (page 31). This means that bonds not specified in the System block will *not* be constrained.

4.9.3 (Re-)Starting a simulation

The state of a system at the beginning of a simulation is defined by the positions and momenta of all atoms. The positions can be set in the input or loaded from a file as described under *Geometry, System definition* (page 31). Initial velocities are then supplied using the `InitialVelocities` block.

Probably the most common way to start up a simulation is to draw the initial velocities from a Maxwell-Boltzmann distribution by setting `Type=Random` and `Temperature` to a suitable value. Alternatively, velocities can be loaded from an `ams.rkf` file produced by an earlier simulation using `Type=FromFile` and `File`. This is the recommended way to start a production simulation from the results of a short preparation/equilibration run.

Velocities of all atoms in units of Å/fs can also be explicitly defined in the `Values` block after setting `Type=Input`. This is mainly useful to repeat or extend simulations done by other programs. For example, velocities can be extracted from the `vels` or `moldyn.vel` files used by the *standalone ReaxFF* program. A simple AWK script is supplied in `scripting/standalone/reaxff-ams/vels2ams.awk` to help with the conversion.

MolecularDynamics**InitialVelocities****Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod**Type**

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature**Type**

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type**Type**

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values**Type**

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

The MD module also supports exact restarts of interrupted simulations by pointing the `Restart` key to an `ams.rkf` file. This will restore the entire state of the MD module from the last available checkpoint (if the previous simulation was interrupted) or from the final state (if the previous simulation ended after `NSteps`). An earlier trajectory can thus be seamlessly extended by increasing `NSteps` and using `Restart`.

Warning: The `Restart` feature is only intended for exact restarts, so the rest of the `MolecularDynamics` settings should be the same as in the original run. Only `NSteps` and engine settings (contents of the `Engine` block) can always be changed safely across restarts.

Although some MD settings (such as the trajectory sampling options) can in practice be changed without problems, changing others (such as thermostat or barostat settings) will cause the restart to fail or produce physically incorrect results. It is thus strongly recommended to only use `Restart` for exact continuation and `InitialVelocities Type=FromFile` together with `LoadSystem` (or a `System` block containing the most recent geometry to start from) otherwise.

MolecularDynamics**Restart****Type**

String

GUI name

Restart from

Description

The path to the `ams.rkf` file from which to restart the simulation.

4.9.4 Thermostats and barostats

By default, the MD simulation samples the microcanonical (NVE) ensemble. Although this is useful to check energy conservation and other basic physical properties, it does not directly map to common experimental conditions. The canonical (NVT) ensemble can be sampled instead by applying a `Thermostat`, which serves as a simulated heat bath around the system, keeping its average temperature at a set value.

AMS offers two thermostats with drastically different properties, mode of operation, and applicability, selected using the `Type` key:

Berendsen

The Berendsen friction thermostat drives the system to a particular target temperature by rescaling the velocities of all atoms in each step. This ensures rapid (exponential) convergence of the temperature with a time constant `Tau`. However, this thermostat produces an incorrect velocity distribution and should thus be avoided in all situations where correct energy fluctuations are important. Additionally, using a too short time constant `Tau` tends to cause incorrect equipartition of energy between different degrees of freedom in the system, leading to the “flying ice cube” phenomenon. The time constant `Tau` should thus be set as large as possible to limit these artifacts while still providing sufficient temperature control. Common values of `Tau` for condensed-phase systems lie between 100 fs (strong damping, rapid convergence) and 10 ps (weak coupling with minimal artifacts).

This thermostat is mainly useful for systems far from equilibrium, for example during the initial preparation and equilibration phase of a simulation. The `NHC` thermostat should be preferred where possible.

NHC

This enables a chain of coupled Nosé-Hoover thermostats. This method introduces artificial degrees of freedom representing the heat bath and ensures correct sampling of the canonical ensemble. The combined total energy of the system and the heat bath is conserved and shown in the GUI as `Conserved Energy`. Checking this quantity for drift and artifacts thus offers a valuable test of the correctness of the simulation. This thermostat exhibits oscillatory relaxation with a period of `Tau`. It is thus not well suited for systems starting far from equilibrium, because the oscillations may take long to settle. The time constant `Tau` should be at least comparable to the period of some natural oscillation of the system to ensure efficient energy transfer. It is commonly on the order of hundreds of femtoseconds, although higher values may be used if weak coupling is desired.

Multiple independent thermostats can be used to separately control different non-overlapping regions of the system at the same time. This is done by first defining appropriate *Regions* (page 37) in the `System` block and then specifying the `Thermostat` block multiple times with the `Region` key of each thermostat set to an appropriate region expression.

MolecularDynamics

Thermostat

Type

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply

Type

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)
- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

Just like using a `Thermostat` to control the temperature of the system, a `Barostat` can be applied to keep the pressure constant by adjusting the volume. This enables sampling the isenthalpic-isobaric (NpH) ensemble by using only a barostat or the isothermal-isobaric (NpT) ensemble by combining a barostat and a thermostat. Unlike thermostats, a barostat always applies to the entire system and there can thus be at most one barostat defined.

AMS offers two barostats with similar properties to the related thermostats:

Berendsen

The Berendsen friction-like isobaric ensemble method rescales the system in each step to drive the pressure towards a target value. Similarly to the `Berendsen` thermostat, the relaxation is exponential with a time constant `Tau`. Similar considerations for the choice of `Tau` apply as in the case of the thermostat, but the value of `Tau` for the barostat is usually at least several times higher than the corresponding `Tau` used for the thermostat. This barostat does not have any conserved quantity.

MTK

This enables the Martyna-Tobias-Klein extended Lagrangian barostat, which generates a true isobaric ensemble by integrating the cell parameters as additional degrees of freedom. This barostat is derived from the Andersen-Hoover isotropic barostat and the Parrinello-Rahman-Hoover anisotropic barostat. Like the `NHC` thermostat, it exhibits oscillatory relaxation unsuitable for systems far from equilibrium. This barostat must always be combined with a `NHC` thermostat. One instance of such thermostat coupled to the atoms as usual, while a second instance is created internally and coupled to the cell degrees of freedom.

MolecularDynamics**Barostat**

Type

Block

Description

This block allows to specify the use of a barostat during the simulation.

BulkModulus**Type**

Float

Default value

2200000000.0

Unit

Pascal

Description

An estimate of the bulk modulus (inverse compressibility) of the system for the Berendsen barostat.

This is only used to make Tau correspond to the true observed relaxation time constant. Values are commonly on the order of 10-100 GPa (1e10 to 1e11) for solids and 1 GPa (1e9) for liquids (2.2e9 for water). Use 1e9 to match the behavior of standalone ReaxFF.

ConstantVolume**Type**

Bool

Default value

No

Description

Keep the volume constant while allowing the box shape to change.

This is currently supported only by the MTK barostat.

Duration**Type**

Integer List

Description

Specifies how many steps should a transition from a particular pressure to the next one in sequence take.

Equal**Type**

Multiple Choice

Default value

None

Options

[None, XYZ, XY, YZ, XZ]

Description

Enforce equal scaling of the selected set of dimensions. They will be barostatted as one dimension according to the average pressure over the components.

Pressure

Type

Float List

Unit

Pascal

Description

Specifies the target pressure.

You can specify multiple pressures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one p to the next p (using a linear ramp).

Scale**Type**

Multiple Choice

Default value

XYZ

Options

[XYZ, Shape, X, Y, Z, XY, YZ, XZ]

Description

Dimensions that should be scaled by the barostat to maintain pressure. Selecting Shape means that all three dimensions and also all the cell angles are allowed to change.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

Specifies the time constant of the barostat.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, MTK]

GUI name

Barostat

Description

Selects the type of the barostat.

Temperature and pressure regimes

Arbitrary temperature and pressure regimes can be generated by setting `Temperature` or `Pressure` to a list of values, corresponding to the successive set points. This needs to be accompanied by a `Duration` key specifying the length of each regime segment in steps:

```
Thermostat
  Temperature 0      300      300      500      500      300
  Duration    100     200      100      200      100
End
```

Note that there is always N-1 `Duration` values for N `Temperature` values. The target temperature of the thermostat in this example will evolve as follows:

1. Increase linearly from 0 to 300 K over 100 steps.
2. Stay constant at 300 K for 200 steps.
3. Increase linearly from 300 to 500 K over 100 steps.
4. Stay constant at 500 K for 200 steps.
5. Decrease linearly from 500 to 300 K over 100 steps.
6. Stay constant at 300 K for the rest of the simulation.

4.9.5 Trajectory sampling and output

A basic principle of the numerical integration of motion in MD is that the changes in the state of the system between successive time steps are small. This means that storing the results of every step is not useful, because all the data is strongly correlated. Instead, a snapshot of the system is taken every N steps, where N is set low enough to still capture the fastest motion of interest but high enough to avoid wasting space due to correlations. The resulting sequence of snapshots is then commonly called the trajectory.

AMS writes the trajectory to the `History` and `MDHistory` sections of `ams.rkf`, according to the settings in the `Trajectory` block. A snapshot of the system and various MD variables is stored every `SamplingFreq` timesteps. By default, this frequency is also used to print basic thermodynamic parameters of the simulation to the output and log file. Set `PrintFreq` to override this.

Frequently sampling a long trajectory can generate large volumes of data. If the space usage becomes a concern, one can selectively disable storing some parts of the trajectory to save space using the `Write*` keys. Note however that this will make it impossible to use some analysis methods on the resulting trajectory:

- `WriteBonds` is necessary for reaction network analysis (ChemTraYzer). Disabling `WriteBonds` also makes AMSmovie show only guessed bonds instead of those calculated by the engine.
- `WriteMolecules` is required by the Molecule Fractions panel in AMSmovie.
- `WriteVelocities` is required to calculate the velocity autocorrelation functions needed for diffusivity and IR spectra.

The trajectory itself contains only the data needed for subsequent analysis of the dynamics of the system. However, much more data is usually generated on every integration step. This includes, for example, the internal data used by an engine when evaluating the energies and forces. This information is normally discarded after most steps, because it is often very large. However, every `EngineResultsFreq` steps this engine-specific data is written to a new engine result file called `MDStep*.rkf`. These files can be inspected afterwards to extract various engine-specific details about the system, such as the orbitals for QM engines.

For some engines such as ForceField or ReaxFF, the data on engine results files is rarely needed for subsequent analysis. In that case, setting `EngineResultsFreq` to zero can be used to disable writing engine results files and save disk space.

To enable restarting interrupted MD simulations or extending them to a larger number of steps, a `Checkpoint` containing the complete internal state of the MD driver is stored every `Frequency` steps. An interrupted simulation can then be restarted from this checkpoint using the `Restart` keyword.

MolecularDynamics

Trajectory

Type

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the `.rkf` file.

EngineResultsFreq

Type

Integer

GUI name

Engine results frequency

Description

Write `MDStep*.rkf` files with engine-specific results once every `N` steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of `Checkpoint%Frequency`.

ExitConditionFreq

Type

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every `N` steps. By default this is done every `SamplingFreq` steps.

PrintFreq

Type

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every `N` steps. By default this is done every `SamplingFreq` steps.

SamplingFreq

Type

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds**Type**

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges**Type**

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps. Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

CalcPressure**Type**

Bool

Default value

No

GUI name

Calculate pressure

Description

Calculate the pressure in periodic systems.

This may be computationally expensive for some engines that require numerical differentiation.

Some other engines can calculate the pressure for negligible additional cost and will always do so, even if this option is disabled.

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

System**Type**

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities**Type**

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

In rare cases, one may require some properties from every MD step, for example, the pressure tensor for the Green-Kubo viscosity calculation. In this case one can use the `BinLog` (binary log) feature. When enabled, the selected MD variables will be stored in the `BinLog` section of `ams.rkf` in the format similar to that of the `MDHistory` section with a few differences: the blocking factor is larger (1000 instead of 100) and the pressure tensor is stored per component, for example, P_{xx} is stored in `PressureTensor_xx`, and so on. Note that none of the atomic properties (coordinates, velocities) are included in the binlog.

MolecularDynamics**BinLog****Type**

Block

Description

This block controls writing the `BinLog` section in `ams.rkf`, which contains the selected MD state scalars and tensors from every MD step. No per-atom data is written. If you need the per-atom data then you can set the sampling frequency to 1 and get the required data from the `MDHistory` section. The tensors are written per component, that is, the pressure tensor is

written as six variables: PressureTensor_xx, PressureTensor_yy, etc.. To reduce the file size, all data is written in blocks.

BiasEnergy**Type**

Bool

Default value

No

Description

Write the CVHD bias energy at every time step.

BoostFactor**Type**

Bool

Default value

No

Description

Write the CVHD boost factor at every time step.

ConservedEnergy**Type**

Bool

Default value

No

Description

Write the conserved energy value at every time step.

Density**Type**

Bool

Default value

No

Description

Write the density at every time step.

DipoleMoment**Type**

Bool

Default value

No

GUI name

Dipole moment binlog

Description

Write the dipole moment at every time step. Each component of the tensor is written in its own variable.

Hypertime

Type

Bool

Default value

No

Description

Write the CVHD hypertime at every time step.

KineticEnergy**Type**

Bool

Default value

No

Description

Write the kinetic energy value at every time step.

MaxBiasEnergy**Type**

Bool

Default value

No

Description

Write the max CVHD bias energy at every time step.

MaxBoostFactor**Type**

Bool

Default value

No

Description

Write the max CVHD boost factor at every time step.

PotentialEnergy**Type**

Bool

Default value

No

Description

Write the potential energy value at every time step.

Pressure**Type**

Bool

Default value

No

Description

Write the pressure at every time step.

PressureTensor

Type

Bool

Default value

No

GUI name

Pressure tensor binlog

Description

Write the pressure tensor in Voigt notation at every time step. Each component of the tensor is written in its own variable.

Step**Type**

Bool

Default value

No

Description

Write the step index during the simulation at every time step.

Temperature**Type**

Bool

Default value

No

Description

Write the temperature at every time step.

Time**Type**

Bool

Default value

No

GUI name

Time (fs) binlog

Description

Write the simulation time (fs) at every time step.

TotalEnergy**Type**

Bool

Default value

No

Description

Write the total energy value at every time step.

Volume**Type**

Bool

Default value

No

Description

Write the simulation cell volume, area or length, depending on the system periodicity, at every time step.

Every time a snapshot of the system is stored in the trajectory file, atomic coordinates are remapped (wrapped) across periodic boundaries into the unit cell. This can be disabled by setting `Remap Type=None`. By default, the MD driver tries to detect which of the two common PBC conventions the system uses, based on the atomic positions at the start of the simulation. This can be customized using the `Remap Range` setting.

MolecularDynamics**Remap****Type**

Block

Description

Control periodic remapping (backtranslation) of atoms into the PBC box.

Range**Type**

Multiple Choice

Default value

Auto

Options

[Auto, ZeroToOne, PlusMinusHalf, AroundCenter]

Description

Select the range of fractional coordinates to which atoms are remapped.

Auto: Use PlusMinusHalf if the geometrical center of the starting positions of all atoms lies between -0.25 and 0.25 in fractional coordinates in all periodic directions, ZeroToOne otherwise.

PlusMinusHalf: Remap atoms to lie between -0.5 and 0.5 in fractional coordinates.

ZeroToOne: Remap atoms to lie between 0 and 1 in fractional coordinates.

AroundCenter: Remap atoms to lie within 0.5 in fractional coordinates around the geometrical center of the starting positions of all atoms.

Type**Type**

Multiple Choice

Default value

Atoms

Options

[None, Atoms]

Description

Select the method used to remap atoms into the unit cell.

None: Disable remapping completely.

Atoms: Remap any atoms that leave the unit cell.

4.9.6 Lattice deformations (volume regimes)

The `Deformation` block can be used to gradually deform the periodic lattice of the system during a MD simulation. This block can be repeated to define multiple deformations, which will be applied on every MD step in the order in which they are listed in the input.

See also:

- [Example demonstrating various lattice deformations](#) (page 512)
- The [ReactionsDiscovery](#) workflow uses lattice deformations to predict chemical reactions

Warning: While a `Deformation` and a `Barostat` can be used at the same time, remember to set the `Scale` parameter of the barostat so that no dimension is simultaneously being deformed and barostatted.

Each `Deformation` block will be active on MD steps between `StartStep` and `StopStep`.

The time dependence of the lattice parameters is defined by the `Type` key:

Linear

This deformation type adds the same constant amount to the selected lattice parameters on every simulation step. When used with `TargetLattice`, `LatticeVelocity`, or `StrainRate`, the lattice matrix H evolves as $H(t) = H_0 + \Delta H \cdot t$, where H_0 is the lattice at `StartStep`.

Exponential

When used with `StrainRate`, this type strains the lattice by the given strain matrix ϵ on every step, so that the lattice matrix H evolves as $H(t) = H_0(1 + \epsilon)^t$. When used with `LengthRate`, the length of each lattice vector evolves as $l(t) = l_0(1 + r)^t$.

Sine

This is a periodic deformation going from the starting value of the selected lattice parameters to a set target, and then with the same amplitude to the opposite direction from the starting lattice.

Cosine

This periodic deformation oscillates between the starting lattice and a defined target.

The period of the oscillation for the `Sine` and `Cosine` types must be set using the `Period` key.

The extent of the deformation is defined by setting one of the seven mutually exclusive input keys. These belong to two main groups, depending on whether they operate on the lattice matrix as a whole, or just on the lengths of the individual lattice vectors:

TargetLattice, Velocity, and Rate

These input keys expect a “lattice-like” matrix of numbers, consisting of up to three rows containing up to three values each. Each row contains the components of a single lattice vector and corresponds to a row of the [Lattice block](#) (page 32). For systems with 1D or 2D periodicity the matrix may be padded to 3x3 with zeros.

TargetLength, LengthVelocity, and LengthRate

These input keys expect a list of up to three values, defining the desired length of each lattice vector or the (absolute or relative) rate of its change.

TargetDensity

This input key serves as a shorthand for setting `TargetLattice` to the current lattice at `StartStep` isotropically scaled by a factor needed to reach a particular target density at `StopStep`.

MolecularDynamics

Deformation

Type

Block

Recurring

True

Description

Deform the periodic lattice of the system during the simulation.

LatticeVelocity**Type**

Non-standard block

Description

Velocity of individual lattice vector components in Angstrom/fs. The format is identical to the System%Lattice block. For Type Sine and Cosine, this defines the maximum velocity (at the inflection point).

LengthRate**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Description

Relative rate of change of each lattice vector per step.

LengthVelocity**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Angstrom/fs

Description

Change the length of each lattice vector with this velocity. With Type=Exponential, Length-Velocity is divided by the current lattice vector lengths on StartStep to determine a Length-Rate, which is then applied on all subsequent steps. For Type Sine and Cosine, this defines the maximum velocity (at the inflection point).

Period**Type**

Float

Unit

Femtoseconds

Description

Period of oscillation for Type Sine and Cosine.

ScaleAtoms**Type**

Bool

Default value

Yes

Description

Scale the atomic positions together with the lattice vectors. Disable this to deform only the lattice, keeping the coordinates of atoms unchanged.

StartStep**Type**

Integer

Default value

1

Description

First step at which the deformation will be applied.

StopStep**Type**

Integer

Default value

0

Description

Last step at which the deformation will be applied. If unset or zero, nSteps will be used instead.

StrainRate**Type**

Non-standard block

Description

Strain rate matrix to be applied on every step. The format is identical to the System%Lattice block.

TargetDensity**Type**

Float

Unit

g/cm3

Description

Target density of the system to be achieved by StopStep by isotropically rescaling the lattice.

This can only be used for 3D-periodic systems. Note that the selected Type determines how the lengths of the lattice vectors will evolve (linearly or as a sine/cosine), the evolution of the volume or density is thus necessarily non-linear.

TargetLattice**Type**

Non-standard block

Description

Target lattice vectors to be achieved by StopStep. The format is identical to the System%Lattice block.

TargetLength

Type

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Angstrom

Description

Target lengths of each lattice vector to be achieved by StopStep. The number of values should equal the periodicity of the system. If a value is zero, the corresponding lattice vector will not be modified.

Type**Type**

Multiple Choice

Default value

Linear

Options

[Linear, Exponential, Sine, Cosine]

Description

Function defining the time dependence of the deformed lattice parameters.

Linear increments the lattice parameters by the same absolute amount every timestep. Exponential multiplies the lattice parameters by the same factor every timestep. Only StrainRate, LengthRate, and LengthVelocity are supported for Type=Exponential. Sine deforms the system from the starting lattice to TargetLattice/TargetLength and then by the same amount to the opposite direction, while Cosine deforms the system from the starting lattice to the target and back.

Example: Transition from the initial lattice to a 10 Å cube over 1000 steps:

```
MolecularDynamics
  Deformation
    StopStep 1000
    TargetLattice
      10.0  0.0  0.0
      0.0 10.0  0.0
      0.0  0.0 10.0
    End
  End
End
```

Example: Oscillate the length of the *c* lattice vector between the initial value and 20 Å, leaving the *a* and *b* vectors unchanged:

```
MolecularDynamics
  Deformation
    Type Cosine
    Period 100
    TargetLength 0 0 20
  End
End
```

Example: Stretch the box in the “z” direction by a true exponential strain of 10 ppm per timestep while barostatting the remaining dimensions:

```

MolecularDynamics
  Barostat Type=MTK Pressure=1e5 Tau=1000 Scale=XY
  Deformation
    Type Exponential
    StrainRate
      0.0  0.0  0.0
      0.0  0.0  0.0
      0.0  0.0  1e-5
    End
  End
End

```

4.9.7 Molecule Gun: adding molecules during simulation

The molecule gun allows you to “shoot” (add with velocity) a molecule into the simulation box.

See also:

The [GUI tutorial](#) on the molecule gun.

Molecules can be continuously added to the simulation or only once. The initial position can be pre-set or be random within the simulation box or a part thereof. It can be defined either in the Cartesian or fractional coordinates. The initial velocity can be specified either directly (in Angstrom per femtosecond) or as translational temperature or kinetic energy. Possible applications of the molecule gun include e.g. the simulation of enforced collisions or deposition processes on surfaces.

MolecularDynamics

AddMolecules

Type

Block

Recurring

True

GUI name

Add molecules

Description

This block controls adding molecules to the system (a.k.a. the Molecule Gun). Multiple occurrences of this block are possible.

By default, molecules are added at random positions in the simulation box with velocity matching the current system temperature. The initial position can be modified using one of the following keywords: Coords, CoordsBox, FractionalCoords, FractionalCoordsBox. The Coords and FractionalCoords keys can optionally be accompanied by CoordsSigma or FractionalCoordsSigma, respectively.

AtomTemperature

Type

Float

Default value

0.0

Unit

Kelvin

Description

Add random velocity corresponding to the specified temperature to individual atoms of the molecule. This only affects rotational and internal degrees of freedom, not the net translational velocity of the inserted molecule as set by the other options.

ContactDistance**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Translate the bullet along the velocity vector until it comes within ContactDistance of any other atom.

Coords**Type**

Float List

Unit

Angstrom

Description

Place molecules at or around the specified Cartesian coordinates.

This setting takes precedence over other ways to specify initial coordinates of the molecule: [CoordsBox], [FractionalCoords], and [FractionalCoordsBox].

CoordsBox**Type**

Float List

Unit

Angstrom

Description

Place molecules at random locations inside the specified box in Cartesian coordinates.

Coordinates of the box corners are specified as: Xmin, Xmax, Ymin, Ymax, Zmin, Zmax.

This setting is ignored if Coords is used.

In AMSinput, if this field is not empty it will be used instead of the default Coords.

CoordsSigma**Type**

Float List

Unit

Angstrom

Description

Sigma values (one per Cartesian axis) for a Gauss distribution of the initial coordinates. Can only be used together with Coords.

DeviationAngle

Type

Float

Default value

0.0

Unit

Degree

Description

Randomly tilt the shooting direction up to this angle away from the VelocityDirection vector.

Energy**Type**

Float

Unit

Hartree

Description

Initial kinetic energy of the molecule in the shooting direction.

EnergySigma**Type**

Float

Default value

0.0

Unit

Hartree

Description

Sigma value for the Gauss distribution of the initial kinetic energy around the specified value.
Should only be used together with Energy.

FractionalCoords**Type**

Float List

Description

Place molecules at or around the specified fractional coordinates in the main system's lattice. For non-periodic dimensions a Cartesian value in Angstrom is expected. This setting is ignored if [Coords] or [CoordsBox] is used.

FractionalCoordsBox**Type**

Float List

Description

Place molecules at random locations inside the box specified as fractional coordinates in the main system's lattice.

Coordinates of the box corners are specified as: Xmin, Xmax, Ymin, Ymax, Zmin, Zmax.

For non-periodic dimensions the Cartesian value in Angstrom is expected.

This setting is ignored if [Coords], [CoordsBox], or [FractionalCoords] is used.

FractionalCoordsSigma

Type

Float List

Description

Sigma values (one per axis) for a Gauss distribution of the initial coordinates. For non-periodic dimensions the Cartesian value in Angstrom is expected. Can only be used together with FractionalCoords.

Frequency**Type**

Integer

Default value

0

Description

A molecule is added every [Frequency] steps after the StartStep.

There is never a molecule added at step 0.

MinDistance**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Keep the minimal distance to other atoms of the system when adding the molecule.

MoleFraction**Type**

Float

Description

Defines a mixture to be deposited using one AddMolecules block per component.

AMS will randomly alternate between any guns that have MoleFraction set. These need to all have the same settings for StartStep, StopStep and Frequency. Any additional AddMolecules blocks without MoleFraction will remain completely independent.

NumAttempts**Type**

Integer

Default value

10

Description

Try adding the molecule up to the specified number of times or until the MinDistance constraint is satisfied. If all attempts fail a message will be printed and the simulation will continue normally.

Rotate**Type**

Bool

Default value

No

Description

Rotate the molecule randomly before adding it to the system.

StartStep**Type**

Integer

Default value

0

Description

Step number when the first molecule should be added. After that, molecules are added every Frequency steps.

For example, if StartStep=99 and Frequency=100 then a molecule will be added at steps 99, 199, 299, etc...

No molecule will be added at step 0, so if StartStep=0 the first molecule is added at the step number equal to [Frequency].

StopStep**Type**

Integer

Description

Do not add this molecule after the specified step.

System**Type**

String

Description

String ID of the [System] that will be added with this 'gun'.

The lattice specified with this System is ignored and the main system's lattice is used instead.

AMSinput adds the system at the coordinates of the System (thus setting Coords to the center of the System).

Temperature**Type**

Float

Unit

Kelvin

Description

Initial energy of the molecule in the shooting direction will correspond to the given temperature.

TemperatureSigma**Type**

Float

Default value

0.0

Unit

Kelvin

Description

Sigma value for the Gauss distribution of the initial temperature the specified value. Should only be used together with Temperature.

Velocity**Type**

Float

Unit

Angstrom/fs

Description

Initial velocity of the molecule in the shooting direction.

VelocityDirection**Type**

Float List

Description

Velocity direction vector for aimed shooting. It will be random if not specified.

In AMSinput add one or two atoms (which may be dummies).

One atom: use vector from center of the system to add to that atom.

Two atoms: use vector from the first to the second atom.

VelocitySigma**Type**

Float

Default value

0.0

Unit

Angstrom/fs

Description

Sigma value for the Gauss distribution of the initial velocity around the specified value. Should only be used together with Velocity.

Since the 2022 release of the AMS driver, it is possible to generate molecule mixtures by using multiple guns that each have the `MoleFraction` keyword set: AMS will then randomly alternate between any of these guns. The cooperating guns need to all have the same settings for `StartStep`, `StopStep` and `Frequency`. (Any additional `AddMolecules` blocks without `MoleFraction` will remain completely independent.) The following example shows the generation of a 80:20 mixture of `molA` and `molB`:

```
MolecularDynamics
  AddMolecules
    System molA
    MoleFraction 0.8
    Frequency 2000
    StartStep 1
  End
  AddMolecules
    System molB
    MoleFraction 0.2
```

(continues on next page)

(continued from previous page)

```

        Frequency 2000
        StartStep 1
    End
End

System anthracene
    ...
End

System benzene
    ...
End

```

Note that the exact number of inserted molecules of each type is not deterministic: running the same job twice may give you slightly different ratios. However, when running the job many times, the relative ratios averaged over the different runs should approach the fractions as requested in the input.

4.9.8 Removing molecules during simulation

Molecules *detected* (page 310) by the AMS driver can also be removed from the system. This feature can, for example, be used to remove reaction products.

MolecularDynamics

RemoveMolecules

Type

Block

Recurring

True

GUI name

Remove molecules

Description

This block controls removal of molecules from the system. Multiple occurrences of this block are possible.

Formula

Type

String

Description

Molecular formula of the molecules that should be removed from the system.

The order of elements in the formula is very important and the correct order is: C, H, all other elements in the strictly alphabetic order.

Element names are case-sensitive, spaces in the formula are not allowed. Digit '1' must be omitted.

Valid formula examples: C2H6O, H2O, O2S. Invalid formula examples: C2H5OH, H2O1, OH, SO2. Invalid formulas are silently ignored.

Use * to remove any molecule, which must be combined with SinkBox or SafeBox.

Frequency

Type

Integer

Default value

0

Description

The specified molecules are removed every so many steps after the StartStep. There is never a molecule removed at step 0.

SafeBox**Type**

Block

Description

Part of the simulation box where molecules may not be removed. Only one of the SinkBox or SafeBox blocks may be present. If this block is present the molecule will not be removed if any of its atoms is within the box. For a periodic dimension it is given as a fraction of the simulation box (the full 0 to 1 range by default). For a non-periodic dimension it represents absolute Cartesian coordinates in Angstrom.

Amax**Type**

Float

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Description

Coordinate of the upper bound along the third axis.

Cmin

Type

Float

Description

Coordinate of the lower bound along the third axis.

FractionalCoordsBox**Type**

Float List

GUI name

Safe box

Description

Do not remove molecules that are (partly) inside the safe box.

Borders of the safe box specified as: Amin, Amax, Bmin, Bmax, Cmin, Cmax.

For periodic dimensions fractional coordinates between 0 and 1 and for non-periodic dimensions Cartesian values in Angstrom are expected.

SinkBox**Type**

Block

Description

Part of the simulation box where matching molecules will be removed. By default, molecules matching the formula will be removed regardless of their location. If this block is present then such a molecule will only be removed if any of its atoms is within the box. For a periodic dimension it is given as a fraction of the simulation box (the full 0 to 1 range by default). For a non-periodic dimension it represents absolute Cartesian coordinates in Angstrom.

Amax**Type**

Float

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Description

Coordinate of the upper bound along the third axis.

Cmin**Type**

Float

Description

Coordinate of the lower bound along the third axis.

FractionalCoordsBox**Type**

Float List

GUI name

Sink box

Description

Remove molecules that are (partly) inside the sink box.

Borders of the sink box specified as: Amin, Amax, Bmin, Bmax, Cmin, Cmax.

For periodic dimensions fractional coordinates between 0 and 1 and for non-periodic dimensions Cartesian values in Angstrom are expected.

StartStep**Type**

Integer

Default value

0

Description

Step number when molecules are removed for the first time. After that, molecules are removed every [Frequency] steps.

For example, if StartStep=99 and Frequency=100 then molecules will be removed at steps 99, 199, 299, etc...

No molecule will be removed at step 0, so if StartStep=0 the first molecules are removed at the step number equal to [Frequency].

StopStep**Type**

Integer

Description

Do not remove the specified molecules after this step.

Warning: When there is a `Molecules%AdsorptionSupportRegion` defined, the molecule formulas depend on whether the molecule is adsorbed or not.

4.9.9 Accelerated dynamics

The PLUMED library support in AMS

PLUMED (<http://www.plumed.org/>) is a plugin that works with various MD programs and is also available in AMS. It can be used for on-the-fly analysis of the dynamics, or to perform a wide variety of free energy methods. The interface with the plugin is really simple: you just need to specify the PLUMED input in the `MolecularDynamics%Plumed%Input` block and it will be passed to the library “as is”. At each MD step, the current state of the system will be passed to the plugin to be updated according to the PLUMED input.

MolecularDynamics

Plumed

Type

Block

Description

Input for PLUMED (version 2.5.0). The parallel option is still experimental.

Input

Type

Non-standard block

Description

Input for PLUMED. Contents of this block is passed to PLUMED as is.

Parallel

Type

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup

Type

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups

Type

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup

Type

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

Metadynamics for Conformer-Rotamer Ensemble Sampling (CREST-MTD)

This is a very specific implementation of metadynamics that is only meant for exploration of conformer space, as used in the Conformer-Rotamer Ensemble Sampling (CREST) approach. It is an RMSD-based metadynamics that places a new 1-dimensional Gaussian on the potential energy surface every `NSteps` steps. The Gaussian is a function of the RMSD from the current geometry, which will always be zero at the moment of placement. All Gaussians have the same maximum height (`Height`) and the same width (`Width`). There is an upper limit of `NGaussiansMax` to the number of Gaussians present in the system, and when this is exceeded the oldest Gaussian is removed. By default the Gaussians are gradually introduced, using a scaling function that increases exponentially from 0 to 1 with the simulation time. The keyword `ScalingSlope` in the input block `GaussianScaling` determines the slope of the scaling function with respect to time. The default value of 0.03 steps^{-1} yields a scaling factor of nearly 1 after 100 steps. The keyword `ScaleGaussians` in the inputblock `GaussianScaling` determines whether the Gaussians are scaled at all.

By default, all atoms are included in the RMSD calculation. However, the selection can be restricted to specific regions using the `Region` keyword. This can be a single region or a combination thereof using region expressions (for example, `reg1+reg2` for a union).

MolecularDynamics**CRESTMTD****Type**

Block

GUI name

CREST_MTD

Description

Input for CREST metadynamics simulation.

AddEnergy**Type**

Bool

Default value

No

Description

Add the bias energy to the potential energy (to match the gradients)

GaussianScaling**Type**

Block

Description

Options for gradual introduction of the Gaussians

ScaleGaussians

Type

Bool

Default value

Yes

Description

Introduce the Gaussians gradually, using a scaling function

ScalingSlope**Type**

Float

Default value

0.03

Description

Slope of the scaling function for the Gaussians with respect to time

Height**Type**

Float

Unit

Hartree

Description

The height of the Gaussians added

NGaussiansMax**Type**

Integer

Description

Maximum number of Gaussians stored

NSteps**Type**

Integer

Description

Interval of Gaussian placement

Region**Type**

String

Default value

*

Description

Restrict the range of atoms for RMSD calculation to the specified region.

RestartFile**Type**

String

Description

Filename for file from which to read data on Gaussians placed previously.

Width**Type**

Float

Unit

Bohr

Description

The width of the Gaussians added in terms of the RMSD

By default, the gradients of the Gaussians with respect to the atom coordinates are added to the state gradients, but the value of the Gaussians is not added to the energy, which is common in metadynamics. For testing purposes it can be useful to add the Gaussian value to the energy, and this can be done with the keyword `AddEnergy`. Irrespective of this choice, the energy value of the Gaussian at each geometry is printed in the `ams.rkf` file in the section `CrestMTDHistory`. It is also possible to use Gaussians from an earlier CREST-MTD simulation, using the keyword `RestartFile`.

Collective Variable-driven HyperDynamics (CVHD)

The Collective Variable-driven HyperDynamics is a molecular dynamics acceleration method that allows observation of rare events by filling energy minima with a bias potential. In this sense it is similar to metadynamics. The difference of the hyperdynamics is that it ensures that the bias disappears in the transition state region. This difference allows hyperdynamics to calculate the rate of slow processes, for example the ignition phase of combustion.

See also:

The [GUI tutorial](#) on CVHD.

The CVHD implementation in AMS follows the algorithm described in [K.M. Bal, E.C. Neyts, JCTC, 11 \(2015\) \(https://doi.org/10.1021/acs.jctc.5b00597\)](#)

The `StartStep`, `Frequency`, `StopStep`, and `WaitSteps` keys define when and how often the bias potential is added, and when it is removed. The `Bias` block defines parameters of the bias potential peaks and the `ColVarBB` block describes parameters of the bond-breaking collective variable.

MolecularDynamics**CVHD****Type**

Block

Recurring

True

GUI name

CVHD

Description

Input for the Collective Variable-driven HyperDynamics (CVHD).

Bias**Type**

Block

Description

The bias is built from a series of Gaussian peaks deposited on the collective variable axis every `[Frequency]` steps during MD. Each peak is characterized by its (possibly damped) height and the RMS width (standard deviation).

DampingTemp

Type

Float

Default value

0.0

Unit

Kelvin

GUI name

Bias damping T

Description

During well-tempered hyperdynamics the height of the added bias is scaled down with an $\exp(-E/kT)$ factor [PhysRevLett 100, 020603 (2008)], where E is the current value of the bias at the given CV value and T is the damping temperature DampingTemp. If DampingTemp is zero then no damping is applied.

Delta**Type**

Float

Description

Standard deviation parameter of the Gaussian bias peak.

Height**Type**

Float

Unit

Hartree

Description

Height of the Gaussian bias peak.

ColVarBB**Type**

Block

Recurring

True

GUI name

Collective Variable

Description

Description of a bond-breaking collective variable (CV) as described in [Bal & Neyts, JCTC, 11 (2015)]. A collective variable may consist of multiple ColVar blocks.

at1**Type**

Block

Description

Specifies the first bonded atom in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection of bonded atoms to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the first atom of the bond. The name must be as it appears in the System block. That is, if the atom name contains an extension (e.g C.1) then the full name including the extension must be used here.

at2**Type**

Block

Description

Specifies the second bonded atom in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection of bonded atoms to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the second atom of the bond. The value is allowed to be the same as [at1], in which case bonds between atoms of the same type will be included.

cutoff**Type**

Float

Default value

0.3

GUI name

Bond order cutoff

Description

Bond order cutoff. Bonds with BO below this value are ignored when creating the initial bond list for the CV. The bond list does not change during lifetime of the variable even if some bond orders drop below the cutoff.

p

Type

Integer

Default value

6

GUI name

Exponent p

Description

Exponent value p used to calculate the p-norm for this CV.

rmax**Type**

Float

Unit

Angstrom

GUI name

R max

Description

Max bond distance parameter Rmax used for calculating the CV. It should be close to the transition-state distance for the corresponding bond.

rmin**Type**

Float

Unit

Angstrom

GUI name

R min

Description

Min bond distance parameter Rmin used for calculating the CV. It should be close to equilibrium distance for the corresponding bond.

ColVarBM**Type**

Block

Recurring

True

GUI name

Collective Variable

Description

Description of a bond-making collective variable (CV). A collective variable may consist of multiple ColVar blocks.

at1**Type**

Block

Description

Specifies selection criteria for the first atom of a pair in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the first atom of the pair. The name must be as it appears in the System block. That is, if the atom name contains an extension (e.g C.1) then the full name including the extension must be used here.

at2**Type**

Block

Description

Specifies selection criteria for the second atom of a pair in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the second atom of the pair. The value is allowed to be the same as [at1], in which case pairs of atoms of the same type will be included.

cutoff**Type**

Float

Default value

0.3

GUI name

Bond order cutoff

Description

Bond order cutoff. Bonds with BO above this value are ignored when creating the initial

atom-pair list for the CV. The list does not change during lifetime of the variable even if some bond orders rise above the cutoff.

p**Type**

Integer

Default value

6

GUI name

Exponent p

Description

Exponent value p used to calculate the p-norm for this CV.

rmax**Type**

Float

Unit

Angstrom

GUI name

R max

Description

Max bond distance parameter Rmax used for calculating the CV. It should be much larger than the corresponding Rmin.

rmin**Type**

Float

Unit

Angstrom

GUI name

R min

Description

Min bond distance parameter Rmin used for calculating the CV. It should be close to the transition-state distance for the corresponding bond.

Frequency**Type**

Integer

Description

Frequency of adding a new bias peak, in steps.

New bias is deposited every [Frequency] steps after [StartStep] if the following conditions are satisfied:

the current CV value is less than 0.9 (to avoid creating barriers at the transition state),

the step number is greater than or equal to [StartStep], and

the step number is less than or equal to [StopStep].

MaxEvents

Type

Integer

Default value

0

Description

Max number of events to allow during dynamics. When this number is reached no new bias will be added for this input block.

StartStep**Type**

Integer

Description

If this key is specified, the first bias will be deposited at this step. Otherwise, the first bias peak is added at the step number equal to the Frequency parameter. The bias is never deposited at step 0.

StopStep**Type**

Integer

Description

No bias will be deposited after the specified step. The already deposited bias will continue to be applied until the reaction event occurs. After that no new CVHD will be started. By default, the CVHD runs for the whole duration of the MD calculation.

WaitSteps**Type**

Integer

Description

If the CV value becomes equal to 1 and remains at this value for this many steps then the reaction event is considered having taken place. After this, the collective variable will be reset and the bias will be removed.

During a CVHD calculation, the following variables are saved to the MDHistory section of the RKF file, in addition to other MD properties:

- *BiasEnergy* - value the bias energy at the current MD step, in Hartree.
- *MaxBiasEnergy* - max BiasEnergy since the last sampling step.
- *BoostFactor* - the boost factor at the given MD step. The boost factor is calculated at each MD step as $boost = e^{E_{bias}/kT}$, where T is the MD ensemble temperature.
- *MaxBoostFactor* - max BoostFactor value since the last sampling step.
- *HyperTime* - boosted MD time, in femtoseconds, which is a sum of the hyper-time steps calculated from the current boost factor and the MD time step as $\Delta t_{boost} = boost * \Delta t$. In hyperdynamics, the hyper-time value is directly related to the rate of the process boosted by the corresponding collective variable.

Temperature Replica Exchange

Sampling of rare events can be accelerated using the Replica Exchange Molecular Dynamics (REMD) method, also known as Parallel Tempering. This method runs multiple replicas (copies) of the simulated system in parallel, each in a different ensemble. In the case of Temperature REMD, these ensembles are all NVT or NpT, each at a different temperature. Periodically, Monte Carlo swaps are attempted between neighboring ensembles. If the current configuration of replica A has a sufficient Boltzmann probability in the ensemble of replica B (and vice versa), the two configurations will be swapped. This causes high-energy configuration to migrate into the high-temperature replicas while low-energy configurations eventually end up in the coldest ensemble. This facilitates the crossing of energy barriers in the high-temperature ensembles while keeping the coldest replica at a given temperature of interest. Because each replica always samples an unbiased ensemble, any property can be calculated using standard MD analysis methods without special preparation.

The method is controlled using the ReplicaExchange block:

MolecularDynamics

ReplicaExchange

Type

Block

Description

This block is used for (temperature) Replica Exchange MD (Parallel Tempering) simulations.

AllowWrongResults

Type

Bool

Default value

No

Description

Allow combining Replica Exchange with other features when the combination is known to produce physically incorrect results.

EWMALength

Type

Integer

Default value

10

Description

Length of the exponentially weighted moving average used to smooth swap probabilities for monitoring.

This value is equal to the inverse of the EWMA mixing factor.

SwapFrequency

Type

Integer

Default value

100

Description

Attempt an exchange every N steps.

TemperatureFactors

Type

Float List

Description

This is the ratio of the temperatures of two successive replicas.

The first value sets the temperature of the second replica with respect to the first replica, the second value sets the temperature of the third replica with respect to the second one, and so on. If there are fewer values than `nReplicas`, the last value of `TemperatureFactor` is used for all the remaining replicas.

Temperatures**Type**

Float List

Description

List of temperatures for all replicas except for the first one.

This is mutually exclusive with `TemperatureFactors`. Exactly `nReplicas-1` temperature values need to be specified, in increasing order. The temperature of the first replica is given by `[Thermostat%Temperature]`.

nReplicas**Type**

Integer

Default value

1

GUI name

Number of replicas

Description

Number of replicas to run in parallel.

The number of replicas set by `nReplicas` must never exceed the total number of processors used for the simulation. If possible, the total number of processors should be an integer multiple of `nReplicas` to ensure good load balancing.

The temperature of the base (coldest) replica is determined by the `Thermostat` input block, just like in an ordinary MD simulation. There are two ways to set the temperatures of the remaining replicas, either using `Temperatures` or `TemperatureFactors`. The latter is typically more convenient, as it makes it easy to set up the optimal geometric progression of temperatures. In the simplest case, it is enough to supply just a single value in `TemperatureFactors`, setting the common ratio of temperatures of any two adjacent replicas.

`SwapFrequency` should be set as low as practical for maximum efficiency. The value of this parameter isn't critical because it doesn't affect the validity of the results. However, setting it too high will decrease overall acceleration by missing some opportunities to exchange. Conversely, using a value that is too low will increase the communication overhead and lead to useless back-and-forth swaps between adjacent replicas. Ideally, `SwapFrequency` should be comparable to the correlation time of the system to ensure that individual exchange attempts are uncorrelated.

The trajectory of each replica is written to a separate RKF file: *ams.rkf* for the base replica and *replicaX.rkf* for the other replicas. One can easily switch between these files in the GUI using *File* → *Related Files*. In addition to data present in any MD trajectory, these files also contain an extra section *ReplicaExchangeHistory* with the following data items written every `SwapFrequency` steps:

- *AvgSwapProbability* – Average swap acceptance for each pair of replicas, smoothed using an exponentially weighted moving average with a mixing factor equal to the inverse of `EWMALength`.
- *{Min,Max,Mean,StdDev}PotentialEnergy* – Statistics of the potential energy for each ensemble over the last `SwapFrequency` steps.

- *SystemInEnsemble i* – Identifies the system (continuous trajectory) currently running in ensemble (replica) *i*.
- *EnsembleOfSystem i* – Inverse mapping of *SystemInEnsemble*, giving the current replica number in which the system number *i* runs.
- *TemperatureOfSystem i* – Equivalent to *EnsembleOfSystem* using temperatures instead of integer numbers to identify ensembles.

These data items can be plotted using the MD Replica Exchange menu in AMSmovie. For example, plotting *TemperatureOfSystem* or *EnsembleOfSystem* is useful to visualize the migration of each system through the space of ensembles, where each curve represents one continuous trajectory. Plotting potential energy statistics or average acceptances facilitates tuning the number of replicas and their temperatures to achieve efficient acceleration. The replica exchange method can only work when the potential energy distributions of adjacent ensembles have a sufficient overlap. This can be easily seen by comparing *MaxPotentialEnergy* of ensemble *i* with *MinPotentialEnergy* of ensemble *i+1*. The optimal degree of overlap is such that leads to approximately 20 % of swap attempts getting accepted. The acceptance of swaps can be monitored by plotting *AvgSwapProbability* and the corresponding *TemperatureFactors* can then be adjusted to keep it near the optimal value.

Bond Boost Method

The bond boost method implemented here is described in³. In this method, the distances between atoms that are relevant to the reaction of interest are calculated to determine the orientation of the reactant molecules. If a suitable initial configuration is recognized, an additional restraint energy (possibly consisting of more than one term) is added to the system that is intended to stretch or compress bonds at a pre-defined rate such that this additional energy can help achieve the energy to cross the reaction barrier. A single term of the restraint energy depends on the restraints type and its parameters, see *restraint definitions* (page 332) for details. If more than one suitable configuration is found then the one with the smallest sum of distances is used to create the restraints.

See also:

The [GUI tutorial](#) on the Bond Boost Method.

MolecularDynamics

BondBoost

Type

Block

Recurring

True

Description

Forced reaction (bond boost) definitions. Multiple BondBoost blocks may be specified, which will be treated independently.

Chain

Type

Block

Description

Specifications of a chain of atoms. When a chain is detected the distance restraints will be activated. No other chain of this type will be detected while any restraints for this chain is active.

AtomNames

³ A. Vashisth, C. Ashraf, W. Zhang, C.E. Bakis, and A.C.T. van Duin, *Accelerated ReaxFF simulations for describing the reactive cross-linking of polymers*, J. Phys. Chem. A, 122, 6633-6642 (2018) (<https://doi.org/10.1021/acs.jpca.8b03826>)

Type

String

Description

Atom names specifying the chain. An atom name can optionally be followed by '@' and a region name, in this case only atoms of this type from the given region will be matched. A leading '@' followed by a number indicates that this position in the chain must be occupied by the atom found earlier at the specified position in the chain. For example "O H N C @1" indicates that the last atom in the chain of the five atoms must be the first oxygen, thus defining a 4-membered ring. This is the only way to define a ring because implicit rings will not be detected. For example, "O H N C O" does not include rings.

MaxDistances**Type**

Float List

Unit

Angstrom

Description

Maximum distances for each pair of atoms in the chain. The number of distances must be one less than the number of AtomNames.

MinDistances**Type**

Float List

Unit

Angstrom

Description

Minimum distances for each pair of atoms in the chain. The number of distances must be one less than the number of AtomNames.

DistanceRestraint**Type**

String

Recurring

True

Description

Specify two atom indices followed by the target distance in Angstrom, the first parameter and, optionally, the profile type and the second parameter.

For periodic systems, restraints follow the minimum image convention.

Each atom index indicates a position of the corresponding atom in the AtomNames key. Currently recognized restraint profile types: Harmonic (default), Hyperbolic, Erf, GaussianWell. The profile name can optionally be prefixed by "OneSided-", thus making this particular restraint one-sided. A one-sided restraint will only push or pull the distance towards its target value depending on the sign of the difference between the initial and the target distance. For example, if the initial distance is larger than the target one then the restraint will only push the value down. This is especially useful with moving restraints because then the restraint will effectively disappear as soon as the system crosses a barrier along the reaction path. By default, the restraints are two-sided, which means they will try to keep the distance between the two specified atoms near the (possibly moving) target value. For moving restraints, this

means that after crossing a reaction barrier the system will slide slowly towards products held back by the restraints.

The first parameter is the force constant for the Harmonic, Hyperbolic, and Erf profiles, or well depth for GaussianWell. The second parameter is the asymptotic force value $F(\text{Inf})$ for Hyperbolic and Erf profiles, or the factor before x^2 in the exponent for GaussianWell.

Note: the GaussianWell restraint should be used with the Moving flag.

Moving**Type**

Bool

Default value

No

GUI name

Move restraint

Description

Move the restraints created with this BondBoost. The restraint value will start at the current coordinate's value and will move towards the optimum during the restraint's lifetime. The increment is calculated from the initial deviation and the [NSteps] parameter.

This feature should be used with the GaussianWell restraint types.

NSteps**Type**

Integer

GUI name

Boost lifetime

Description

Number of steps the restraints will remain active until removed. Atoms participating in one reaction are not available for the given number of steps.

NumInstances**Type**

Integer

Default value

1

GUI name

Number of instances

Description

Number of reactions of this type taking place simultaneously.

Units**Type**

Multiple Choice

Default value

Default

Options

[Default, MD]

GUI name

Restr. parameter units

Description

Change energy, force and force constant units in the DistanceRestraint key from the default (atomic units) to those often used in the MD community (based on kcal/mol and Angstrom). Units for the optimum distances are not affected.

For example:

```
MolecularDynamics
  BondBoost
    NSteps 10000
    Chain
      AtomNames      N.1   C.t   O     H     @1
      MinDistances    3.0   1.2   1.5   0.8
      MaxDistances    3.8   3.0   2.5   1.5
    End
#   Restraints      i   j   R_0   FC
  DistanceRestraint 1  2   1.5   0.05 Hyperbolic 0.7
  DistanceRestraint 2  3   3.5   0.03 Hyperbolic 0.5
  DistanceRestraint 3  4   1.0   0.03 Hyperbolic 0.5
End
End
```

The AtomNames, MinDistances, and MaxDistances keys constitute the atom chain definitions for the initial configuration. Thus, the example above defines a chain of atoms N.1-C.t-O-H-N.1 with R(N.1-C) in the (3.0,3.8) range, R(C.t-O) in the (1.2,3.0) range, etc.. In this example, the last atoms in the chain is required to be the same as the first one, thus defining a ring. The specified restraints will push pairs of atoms C-N and O-H close together, which will hopefully let them form a bond, and pull atoms C.t and O away from each other, thus breaking the C-O bond. The restraint type is set to Hyperbolic to avoid very large forces that would otherwise result from a harmonic potential at a large deviation.

Note: When detecting coordinates, the program uses the full atom name and not just the element name. An atom name consists of the element name optionally followed by a period and a suffix, just like N.1 and C.t in the example above. Using extended names for some atoms one may allow only a subset of bonds to be boosted.

Reaction boost (Targeted MD)

The principle of targeted (or steered) MD is simple: during molecular dynamics, the system is pushed towards a certain known configuration in small steps. The steering can be in the form of holonomic constraints, then it is the standard Targeted MD (TMD). When restraints instead of constraints are used, then some authors call it steered MD⁶.

The ReactionBoost feature uses restraints to push the system towards the target configuration. The restraints can be one of the following two types: atom-pair or (mass-weighted) RMSD. In both cases, the user can select a region to which the steering will be limited. AMS will then apply the restraints only to the atoms of the region(s). The RMSD type of restraints is suitable for very large systems or regions while the pair restraints can be used for small to moderately-sized ones. The atom-pair restraints can be further sub-divided into the following types, depending on the mutual bonding situation of the two atoms in the initial and the final state: bonded, non-bonded, bond-breaking and bond-making. The type is determined automatically at the time the steering is started based on the interatomic distances and the covalent radii of the atoms. A non-bonded restraint is a static repulsive potential, it is intended to keep atoms from forming a bond. The other three are moving: the restraint's target distance changes between its initial and final values as the MD progresses. Setting the profile to None disables the corresponding pair restraint type.

⁶ He Huang, E. Ozkirimli, C.B. Post, *A Comparison of Three Perturbation Molecular Dynamics Methods for Modeling Conformational Transitions*, J Chem Theory Comput. 5 (2009) 1301–1314 (<https://doi.org/10.1021/ct9000153>)

The following keys define parameters of the method:

MolecularDynamics**ReactionBoost****Type**

Block

GUI name

Reaction Boost

Description

Define a series of transitions between different states of the system.

Each transition is defined by a TargetSystem and by a set of restraints that force the transition.

BondBreakingRestrains**Type**

Block

Description

Define parameters for moving restraints that are added for pairs of atoms that become disconnected during the transition.

It is intended to make sure the corresponding bonds get broken, although this may not always be required because forming other bonds will likely get these bonds broken.

Erf**Type**

Block

Description

Define parameters for the Int(erf) potential $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi}))$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce.

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

GaussianWell**Type**

Block

Description

Define parameters in the Gaussian well potential $V = -\text{WellDepth} \cdot \exp(-\text{Sigma} \cdot (r-r_0)^2)$.

Sigma**Type**

Float

Default value

1.0

Unit

1/Bohr²

Description

Sigma parameter in the potential expression.

WellDepth**Type**

Float

Default value

1.0

Unit

Hartree

Description

WellDepth parameter in the potential expression.

Harmonic**Type**

Block

Description

Define parameters for the harmonic potential $V = 0.5 \cdot \text{FC} \cdot (r-r_0)^2$.

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The FC parameter of the harmonic potential.

Hyperbolic**Type**

Block

Description

Define parameters for the hyperbolic potential $V = \alpha * (\sqrt{1 + \beta * x^2} - 1)$. The α and β parameters are computed from the user-defined ForceConstant and MaxForce: $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

Taper**Type**

Block

Description**Enabled****Type**

Bool

Default value

No

GUI name

Tapering

Description

Enable tapering of the restraint potential and force between the given range of bond distances. A 7-th order tapering function on the actual (not target!) distance will be used. The MaxDistance must be greater than MinDistance.

MaxDistance**Type**

Float

Default value

0.0

Unit

Angstrom

GUI name

End tapering at

Description

Bond length at which the restraint potential and force decays to zero.

MinDistance**Type**

Float

Default value

0.0

Unit

Angstrom

GUI name

Start tapering at

Description

Bond length at which the restraint potential and force will start decaying to zero.

Type**Type**

Multiple Choice

Default value

Erf

Options

[None, Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Bond breaking restraints

Description

Select type of the moving restraint profile.

Harmonic: $V=0.5*FC*(r-r_0)^2$ Hyperbolic: $V=\alpha*(\sqrt{1 + \beta*x^2} - 1)$ Erf: $V = \alpha*(\beta*x*\text{erf}(\beta*x) + (\exp(-(\beta*x)^2) - 1)/\sqrt{\pi}))$ GaussianWell: $V=\text{WellDepth}*(1-\exp(-\text{Sigma}*(r-r_0)^2))$ Here $\beta=\text{ForceConstant}/\text{MaxForce}$, $\alpha=\text{MaxForce}/\beta$.

The force for Hyperbolic and Erf is bounded by a user-defined value, the latter converging to it faster than the former.

The GaussianWell has a finite depth so it is suitable for cases when crossing a high reaction barrier is not desirable.

Moving restraints are added for pairs of atoms that become disconnected during the transition.

It is intended to make sure the corresponding bonds get broken, although this may not always be required because forming other bonds will likely get these bonds broken.

BondMakingRestraints

Type

Block

Description

Define parameters for moving restraints that are added for pairs of atoms that become connected during the transition.

It is intended to make sure the bonds are created as required.

Erf**Type**

Block

Description

Define parameters for the Int(erf) potential $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi}))$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce.

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

GaussianWell**Type**

Block

Description

Define parameters in the Gaussian well potential $V = -\text{WellDepth} * \exp(-\text{Sigma} * (r - r_0)^2)$.

Sigma**Type**

Float

Default value

1.0

Unit1/Bohr²

Description

Sigma parameter in the potential expression.

WellDepth**Type**

Float

Default value

1.0

Unit

Hartree

Description

WellDepth parameter in the potential expression.

Harmonic**Type**

Block

Description

Define parameters for the harmonic potential $V=0.5*FC*(r-r_0)^2$.

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The FC parameter of the harmonic potential.

Hyperbolic**Type**

Block

Description

Define parameters for the hyperbolic potential $V=\alpha*(\sqrt{1 + \beta*x^2} - 1)$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce: $\beta=ForceConstant/MaxForce$, $\alpha=MaxForce/\beta$

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce

Type

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

Type**Type**

Multiple Choice

Default value

Erf

Options

[None, Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Bond making restraints

Description

Select type of the moving restraint profile.

Harmonic: $V=0.5*FC*(r-r0)^2$ Hyperbolic: $V=\alpha*(\sqrt{1 + \beta*x^2} - 1)$ Erf: $V = \alpha*(\beta*x*\text{erf}(\beta*x) + (\exp(-(\beta*x)^2) - 1)/\sqrt{\pi}))$ GaussianWell: $V=-\text{WellDepth}*\exp(-\text{Sigma}*(r-r0)^2)$ Here $\beta=\text{ForceConstant}/\text{MaxForce}$, $\alpha=\text{MaxForce}/\beta$.

The force for Hyperbolic and Erf is bounded by a user-defined value.

The GaussianWell has a finite depth so it is suitable for cases when crossing a high reaction barrier is not desirable.

Moving restraints are added for pairs of atoms that become connected during the transition.

It is intended to make sure the bonds are created as required.

BondedRestrains**Type**

Block

Description

Define parameters for bonded restraints. A bonded restraint is added for each pair of atoms that are bonded both in the current and in the final state.

It is intended to make sure they remain bonded during simulation.

Harmonic**Type**

Block

DescriptionDefine parameters for the harmonic potential $V=0.5*FC*(r-r0)^2$.

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The FC parameter of the harmonic potential.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Harmonic]

GUI name

Bonded restraints

Description

Select type of the bonded restraints:

Harmonic: $V=0.5*FC*(r-r_0)^2$

A bonded restraint is added for each pair of atoms that are bonded both in the current and in the final state.

It is intended to make sure they remain bonded during simulation.

Change**Type**

Multiple Choice

Default value

TargetCoordinate

Options

[TargetCoordinate, Force, LogForce]

GUI name

Move type

Description

Select what to change during dynamics.

By default, once the restraints are switched on, AMS will change the restraint's target coordinate towards its final value.

If the [Force] or [LogForce] option is selected then the target coordinate is set to its final value immediately and instead the restraint force is gradually scaled from 0 to 1. The scaling is either linear (Force) or logarithmic (LogForce).

InitialFraction

Type

Float

Default value

0.0

Description

Initial fraction of the boost variable.

At the first boosting step, the restraint's target value (or force or $\log(\text{force})$) is equal to $\text{InitialFraction} + 1/\text{NSteps}$.

InterEquilibrationSteps**Type**

Integer

Default value

0

Description

Number of equilibration steps after reaching a target before setting up restraints for the next one.

MinBondChange**Type**

Float

Default value

1.0

Unit

Bohr

Description

Minimal change in the distance for an individual restraint to be considered bond-breaking/making vs bonded.

MinBondStrength**Type**

Float

Default value

0.5

Description

Minimum strength (usually ranges from 0 to 1) for a bond to be considered.

NSteps**Type**

Integer

Default value

500

GUI name

Steps per target

Description

Number of steps per target the restraints should be active for.

NonBondedRestrains

Type

Block

Description

Define parameters for non-bonded restraints. A non-bonded restraint is added for each pair of atoms that are bonded neither in the current nor in the final state.

It is intended to keep them from forming a bond unintentionally. They are represented by a repulsive potential

Exponential**Type**

Block

Description

Define parameters for the repulsive potential $V = \text{Epsilon} \cdot \exp(-\text{Sigma} \cdot r)$.

Epsilon**Type**

Float

Default value

1.0

Unit

Hartree

Description

Epsilon parameter in the repulsive potential expression.

Sigma**Type**

Float

Default value

1.0

Unit

1/Bohr

Description

Sigma parameter in the repulsive potential expression.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Exponential]

GUI name

Non-bonded restraints

Description

Select type of the non-bonded restraints:

Exponential: $V = \text{Epsilon} \cdot \exp(-\text{Sigma} \cdot r)$

A non-bonded restraint is added for each pair of atoms that are bonded neither in the current nor in the final state.

It is intended to keep them from forming a bond unintentionally. They are represented by a repulsive potential.

PreEquilibrationSteps**Type**

Integer

Default value

0

Description

Number of steps before enabling the first set of restraints.

RMSDRestraint**Type**

Block

GUI name

RMSD restraint

Description

Define a static restraint that pulls each atom to its position in the target system, but in contrast to the individual restraints, the force for this one depends on the total mass-weighted root-mean-squared distance (RMSD) between the two structures.

Erf**Type**

Block

Description

Define parameters for the Int(erf) potential $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi}))$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce.

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

GaussianWell**Type**

Block

Description

Define parameters in the Gaussian well potential $V = -\text{WellDepth} \cdot \exp(-\text{Sigma} \cdot (r-r_0)^2)$.

Sigma**Type**

Float

Default value

1.0

Unit

1/Bohr²

Description

Sigma parameter in the potential expression.

WellDepth**Type**

Float

Default value

1.0

Unit

Hartree

Description

WellDepth parameter in the potential expression.

Harmonic**Type**

Block

Description

Define parameters for the harmonic potential $V = 0.5 \cdot \text{FC} \cdot (r-r_0)^2$.

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The FC parameter of the harmonic potential.

Hyperbolic**Type**

Block

Description

Define parameters for the hyperbolic potential $V = \alpha * (\sqrt{1 + \beta * x^2} - 1)$. The α and β parameters are computed from the user-defined ForceConstant and MaxForce: $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Type

Description

Select type of the RMSD restraint profile:

Harmonic: $V = 0.5 * FC * (r - r_0)^2$

Hyperbolic: $V = \alpha * (\sqrt{1 + \beta * x^2} - 1)$

Erf: $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi})$,

GaussianWell: $V = -\text{WellDepth} * \exp(-\text{Sigma} * (r - r_0)^2)$

Here $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$.

The Harmonic profile can be problematic at large deviations as it may result in large forces. The force for Hyperbolic and Erf is bounded by a user-defined value. The GaussianWell has a finite depth so it is suitable for cases when crossing a high reaction barrier is not desirable.

Region

Type

String

Default value

*

GUI name

Region

Description

Region to which the restraints should be limited.

TargetSystem**Type**

String

Recurring

True

GUI name

Target system

Description

The target system's name for this transition. Multiple targets can be specified to request multiple transitions in one simulation.

Note that only the lattice and the atomic coordinates of the target system are used and other properties (bonds, charge, etc.) are ignored. The target system's lattice is used only to determine connections and it cannot be restrained.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Pair, RMSD]

GUI name

Restraint set type

Description

Reaction Boost uses a series of transitions between different states of the system.

Each transition is defined by a TargetSystem and by a set of restraints that force the transition.

Select the type of the restraint set:

-None: no Reaction Boost

- Pair: use pair restraints

- RMSD: use RMSD restraints.

Pair restraints are defined per atom pair while the RMSD defines one collective restraint for all atoms and is thus suitable for very large systems.

The pair restraints are further divided into four sub-types: bonding, non-bonding, bond-breaking and bond-making. The sub-type of restraints for each pair is determined automatically depending on whether the two atoms are bonded in the initial/final state.

Parameters of the pair restraints are defined by [NonBondedRestrains], [BondedRestrains], [BondBreakingRestrains] and [BondMakingRestrains] blocks, while those of the RMSD restraint by the [RMSDRestraint] block.

Force bias Monte Carlo (fbMC)

Configurational sampling and structural relaxation can be accelerated by the Force bias Monte Carlo method. Unlike molecular dynamics, this approach does not use the equations of motion or atomic velocities. Instead, all atoms are randomly displaced in each fbMC steps. AMS uses an uniform-acceptance force bias Monte Carlo flavour of fbMC called tfMC, as published by Mees et al.⁴ Unlike other well-known Monte Carlo methods such as the Metropolis algorithm, fbMC does not generate completely random moves and then test whether these should be accepted or rejected. Instead, the atomic displacements are generated so that atoms preferably move in the direction of the forces acting on them. This special biasing enables fbMC to accept every move, sampling the correct canonical ensemble with high efficiency.

In AMS the fbMC method is implemented as an add-on to molecular dynamics, enabling mixed MC-MD simulations. After every `Frequency` MD steps, the MD procedure is stopped and fbMC takes over, executing `NSteps` Monte Carlo steps. Once the fbMC procedure completes, the MD simulation continues from the new geometry generated by fbMC while reusing atomic velocities from before the start of fbMC.

The fbMC module requires three main input settings. Apart from the `NSteps` and `Frequency` defining the mix of molecular dynamics and Monte Carlo, `Temperature` needs to be set as desired to sample a particular canonical (NVT) ensemble. Although the fbMC procedure always runs in constant temperature mode, the rest of the MD simulation can in principle use any ensemble (with or without a thermostat or barostat).

Note: To run a pure fbMC simulation with no molecular dynamics component, leave `Frequency` set to 1 (the default), set the MD `TimeStep` and `InitialVelocities` to zero, and do not use other MD features such as a thermostat or a barostat.

Note: AMS currently cannot write sampled geometries to the trajectory during the fbMC procedure. To see the evolution of the system during a long fbMC run, split it into multiple shorter fbMC passes to let the MD driver sample the current state between them. To do this, set the `NSteps` for fbMC to the desired trajectory sampling interval, keep `Frequency` set to 1 and set `Trajectory SamplingFreq` to 1.

The amount of acceleration provided by fbMC depends on the `StepLength` parameter, which defines the maximum displacement of the lightest atom in the system. The default value of 0.1 Å is relatively conservative. Increasing `StepLength` leads to faster evolution of the system, but setting it too high can lead to inaccurate sampling or cause crashes by generating geometries that are too distorted.

An estimate of the timescale associated with fbMC (tfMC) steps will be printed to the output.

MolecularDynamics

fbMC

Type

Block

Recurring

True

GUI name

fbMC

⁴ M.J. Mees, G. Pourtois, E. C. Neyts, B. J. Thijsse, and André Stesmans, *Uniform-acceptance force-bias Monte Carlo method with time scale to study solid-state diffusion*, Phys. Rev. B 85, 134301 (2012) (<https://doi.org/10.1103/PhysRevB.85.134301>)

Description

This block sets up force bias Monte Carlo interleaved with the molecular dynamics simulation.

Frequency**Type**

Integer

Default value

1

Description

Run the fbMC procedure every Frequency MD steps.

MassRoot**Type**

Float

Default value

2.0

Description

Inverse of the exponent used to mass-weight fbMC steps.

MolecularMoves**Type**

Block

Description

Move molecules as rigid bodies in addition to normal atomic moves.

Enabled**Type**

Bool

Default value

No

GUI name

Enable molecular moves

Description

Enable moving molecules as rigid bodies based on net forces and torques. Ordinary per-atom displacements will then be based on residual atomic forces.

RotationStepAngle**Type**

Float

Default value

0.1

Unit

Radian

Description

Maximum allowed angle of rotation of each molecule in one fbMC step.

TranslationStepLength

Type

Float

Default value

0.1

Unit

Angstrom

Description

Maximum allowed displacement of each molecule in each Cartesian coordinate in one fbMC step.

NSteps**Type**

Integer

GUI name

Number of steps

Description

Number of fbMC steps to perform on every invocation of the procedure.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N fbMC steps. This defaults to the PrintFreq set in the Trajectory block. Setting this to zero disables printing fbMC steps.

StartStep**Type**

Integer

Default value

1

Description

First step at which the fbMC procedure may run.

StepLength**Type**

Float

Default value

0.1

Unit

Angstrom

Description

Maximum allowed displacement of the lightest atom in the system in each Cartesian coordinate in one fbMC step.

StopStep

Type

Integer

Default value

0

Description

Last step at which the fbMC procedure may run. If unset or zero, there is no limit.

Temperature**Type**

Float

Unit

Kelvin

Description

Temperature used for fbMC.

Nanoreactor**See also:**

The [ReactionsDiscovery](#) workflow uses the nanoreactor to predict chemical reactions.

Nanoreactor is implemented following the article by L.-P. Wang et al.⁵. The nanoreactor is a spherical region of space centered at the origin and surrounded by an elastic wall (represented by a harmonic potential), its size changing over time. The idea of the nanoreactor is to have two phases: reaction and relaxation. In the following, each nanoreactor phase is referred to as a *reactor*. During the relaxation phase, the reactor is large and if the temperature is high enough the reactants are distributed more or less evenly throughout its volume. In the reaction phase, the reactor shrinks and the atoms that now fall outside the reactor's sphere accelerate towards its center. During this phase, high-energy collisions may lead to reactions. After the reaction phase is over, the relaxation phase begins again and the reactor is expanded. This cycle is repeated indefinitely.

Each phase is specified by its own `Reactor` input block. There should be at least two of them but there can be more. The `NSteps` key specifies the duration of the phase in MD steps. The radius of the reactor and the force constant of its wall are set by the `Radius` and `ForceConstant` keys, respectively. To minimize additional strain on internal degrees of freedom of the affected molecules, the force applied by the wall on an atom is proportional to the atom's mass by default. This means that atoms situated at the same distance from the center of the reactor will receive the same acceleration irrespective of their mass. This should be taken into account when choosing a value for the `ForceConstant`. Alternatively, this feature can be disabled using the `MassScaled` key.

MolecularDynamics**Reactor****Type**

Block

Recurring

True

Description

Define one phase of the nanoreactor. A reactor is a region of space surrounded by an elastic wall. Atoms inside the region are not affected. Atoms outside it will be pushed back with force depending on the `[ForceConstant]` and the `[MassScaled]` flag.

⁵ L.-P. Wang, A. Titov, R. McGibbon, F. Liu, V.S. Pande and T.J. Martínez, *Discovering chemistry with an ab initio nanoreactor*, *Nature Chemistry* 6, 1044 (2014) (<https://doi.org/10.1038/NCHEM.2099>)

ForceConstant**Type**

Float

GUI name

Reactor force constant

Description

Force constant of the reactor wall in Hartree/Bohr² (or Hartree/Bohr²/Dalton if [MassScaled] is true).

MassScaled**Type**

Bool

Default value

Yes

GUI name

Scale force by mass

Description

If this flag is disabled the force on an atom outside of the reactor depends only on the atomic coordinates and the force constant. Otherwise, the force is also multiplied by the mass of the atom. This means that atoms at the same distance from the wall will receive the same accelerate due to the wall potential.

NSteps**Type**

Integer

GUI name

Reactor lifetime

Description

Number of steps for which the reactor will remain active until disabled. The next reactor will be activated immediately after this. After the last reactor is disabled the cycle will repeat.

Radius**Type**

Float

Unit

Angstrom

GUI name

Reactor radius

Description

Radius of the reactor sphere.

4.9.10 Non-equilibrium MD (NEMD)

T-NEMD for thermoconductivity: heat exchange

There are different methods to study thermal conductivity using non-equilibrium molecular dynamics (NEMD). A common feature of these methods is that they require the system to be divided into three or more zones, each with its own thermostat and other properties. One method maintains the temperature of the heat source and the heat sink zones at the given temperature using two different thermostats and measures the amount of heat transferred. These method does not require any special features besides a standard thermostat and a possibility to calculate the amount of heat accumulated by the thermostat per unit of time. The accumulated thermostat energies are available in the MDHistory section of ams.rkf file, in variables called 'XXXXstat#Energy', where XXXX is a 4-letter abbreviation of the thermo-/barostat ('BerT' for a Berendsen thermostat, 'NHCT' for an NHC thermostat, 'NHTB' for an MTK barostat, etc.) and '#' is a 1-digit index of the thermo-/barostat.

In the other method, the heat flow is constant and the induced temperature gradient is measured. This method is implemented in AMS in three variants: a simple heat exchange, HEX¹ and eHEX². In the simple heat exchange method the atom velocities are scaled up (or down) by a factor corresponding to the amount of heat deposited to the heat source (or withdrawn from the heat sink) without paying attention to the conservation of the total momentum of the heat source (or sink). In the HEX method the velocities are scaled in such a way that the total momentum is conserved. This, however, introduces a small but measurable drift in the total energy. The eHEX algorithm improves upon the HEX by adding a third-order time-integration correction to remove the drift.

This method is controlled by the HeatExchange sub-block of the MolecularDynamics block:

MolecularDynamics

HeatExchange

Type

Block

Recurring

True

GUI name

Heat exchange

Description

Input for the heat-exchange non-equilibrium MD (T-NEMD).

HeatingRate

Type

Float

Unit

Hartree/fs

Description

Rate at which the energy is added to the Source and removed from the Sink. A heating rate of 1 Hartree/fs equals to about 0.00436 Watt of power being transferred through the system.

Method

Type

Multiple Choice

¹ T. Ikeshoji and B. Hafskjold, *Non-equilibrium molecular dynamics calculation of heat conduction in liquid and through liquid-gas interface* *Molecular Physics* 81, 251-261 (1994) (<https://doi.org/10.1080/00268979400100171>)

² P. Wirnsberger, D. Frenkel, and C. Dellago, *An enhanced version of the heat exchange algorithm with excellent energy conservation properties* *Journal of Chemical Physics* 143, 124104 (2015) (<http://dx.doi.org/10.1063/1.4931597>)

Default value

Simple

Options

[Simple, HEX, eHEX]

Description

Heat exchange method used.

Simple: kinetic energy of the atoms of the source and sink regions is modified irrespective of that of the center of mass (CoM) of the region (recommended for solids).

HEX: kinetic energy of the atoms of these regions is modified keeping that of the corresponding CoM constant.

eHEX: an enhanced version of HEX that conserves the total energy better (recommended for gases and liquids).

Sink**Type**

Block

Description

Defines the heat sink region (where the heat will be removed).

AtomList**Type**

Integer List

GUI name

Sink region

Description

The atoms that are part of the sink.

This key is ignored if the [Box] block or [Region] key is present.

Box**Type**

Block

Description

Part of the simulation box (in fractional cell coordinates) defining the heat sink. If this block is specified, then by default, the whole box in each of the three dimensions is used, which usually does not make much sense. Normally, you will want to set the bounds along one of the axes.

Amax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the third axis.

Cmin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the third axis.

Region**Type**

String

GUI name

Sink region

Description

The region that is the sink.

This key is ignored if the [Box] block is present.

Source

Type

Block

Description

Defines the heat source region (where the heat will be added).

AtomList**Type**

Integer List

GUI name

Source region

Description

The atoms that are part of the source.

This key is ignored if the [Box] block or [Region] key is present.

Box**Type**

Block

Description

Part of the simulation box (in fractional cell coordinates) defining the heat source. If this block is specified, then by default, the whole box in each of the three dimensions is used, which usually does not make much sense. Normally, you will want to set the bounds along one of the axes. This block is mutually exclusive with the FirstAtom/LastAtom setting.

Amax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the second axis.

Bmin

Type

Float

Default value

0.0

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the third axis.

Cmin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the third axis.

Region**Type**

String

GUI name

Source region

Description

The region that is the source.

This key is ignored if the [Box] block is present.

StartStep**Type**

Integer

Default value

0

Description

Index of the MD step at which the heat exchange will start.

StopStep**Type**

Integer

Description

Index of the MD step at which the heat exchange will stop.

One should be careful when choosing a value for the HeatingRate because a too large value may lead to pyrolysis of the heat source or to an abnormal termination when all the kinetic energy of the heat sink has been drained. The optimal

value depends on the size of the system, its heat conductivity and the desired temperature gradient value. The thermal conductivity k can be calculated by dividing the heat flow rate W by the temperature gradient $grad(T)$ and by the flow cross-section area S : $k = W / (S \cdot grad(T))$. See the [trajectory sampling](#) (page 121) section above on how to obtain the temperature profile from which the $grad(T)$ can be computed.

NEMD for tribology: Applying velocities / forces

Non-equilibrium Molecular Dynamics (NEMD) is a commonly used tool to study phenomena like friction & wear in tribochemical systems. In these MD simulations, an constant external force or velocity is applied to a specific subgroup of atoms. For example, when studying sliding solid surfaces, the `ApplyForce` feature can be used to exert a compressive normal force in the -Z direction on one of the solid slabs while the `ApplyVelocity` block simultaneously slides the same group of atoms along the X or Y axis.

In the case of `ApplyVelocity`, the specified group of atoms (defined as a `Region`) will move with the specified constant net `Velocity`. The imposed velocity is automatically excluded from thermostating and system-wide momentum removal. The `Components` input key can be used to select just a subset of Cartesian components of the net velocity of the group to be controlled, so that other components can be left to evolve freely. In the sliding example in the previous paragraph, one would use `Components=XY` to let the Z motion of the sliding slab to freely follow the applied normal force.

In the case of `ApplyForce`, the selected `Region` (group of atoms) will experience an additional specified constant `Force` on each timestep of the simulation. By default, the `Force` parameter defines the total (net) force acting on the whole `Region`. This can be changed by setting the `PerAtom` option to apply the `Force` to each atom in the `Region` separately.

See also:

[Example demonstrating a NEMD friction simulation](#) (page 531)

MolecularDynamics

ApplyVelocity

Type

Block

Recurring

True

Description

The `ApplyVelocity` keyword can be used to move an arbitrary group of atoms in the system with a constant net velocity

Components

Type

Multiple Choice

Default value

XY

Options

[X, Y, Z, XY, YZ, XZ, XYZ]

Description

Select which components of the `Velocity` vector are used to set the corresponding components of the net velocity of the specified set of atoms. Any other components of `Velocity` are ignored and the motion of the selected atoms in those directions is unaffected by `ApplyVelocity`.

Region

Type

String

Recurring

True

Description

Applies the defined velocity to all atoms in this region.

Velocity**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Angstrom/fs

Recurring

False

Description

The constant velocity that will be applied to the specified atoms.

MolecularDynamics**ApplyForce****Type**

Block

Recurring

True

Description

The ApplyForce keyword can be used to apply an arbitrary constant force to a certain (subgroups of) atoms in the system

Force**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Hartree/Bohr

Description

Defines the constant force vector

PerAtom**Type**

Bool

Default value

No

Description

If enabled, the Force vector is applied separately to every atom in the selected Region, so that the total net force on the Region equals the value of Force times the number of atoms

in Region. This was the behavior of ApplyForce in AMS2022. By default, with PerAtom disabled, the Force vector defines the total net force on the Region, so the force applied to each atom equals the value of Force divided by the number of atoms in Region.

Region

Type

String

Recurring

True

Description

Apply the constant force to all atoms in this region.

When the ApplyVelocity block is used to slide a region, the measured net force experienced by that region is written to the MDHistory section on the trajectory. This force is calculated directly from the gradients returned by an Engine and thus excludes the effect of any ApplyForce blocks. The ratio of the tangential (sliding) and normal (compression) component of the mean force is thus equal to the friction coefficient. Three variables will be written by any ApplyVelocity block:

- *EngXFrcRgnA*: instantaneous engine force on region A, Cartesian component X
- *MeanEngXFrcRgnA*: average of *EngXFrcRgnA* over all timesteps since the previous trajectory frame
- *StdevEngXFrcRgnA*: standard deviation of *EngXFrcRgnA* over all timesteps since the previous trajectory frame

Note: Only the first eight characters of the region name are used to construct the *EngXFrcRgn* variables. When using multiple ApplyVelocity instances with different regions, make sure their names are not ambiguous after shortening to eight characters.

NEMD viscosity via cosine shear perturbation

In order to calculate the viscosity using NEMD, apply a periodic perturbation to the system using the CosineShear input block. This will add an amount of acceleration to the atoms in the system according to a cosine function running along the specified Cartesian ProfileAxis. The acceleration applied to each atom will have the direction of FlowDirection and magnitude equal to $a_x(z) = A \cos(\frac{2\pi z}{L_z})$, where A is the amplitude defined by the Acceleration input key and L_z is the length of the periodic box in the Z axis (assuming ProfileAxis=Z and FlowDirection="1 0 0"). It is typically the most efficient to use a box roughly three times taller in the ProfileAxis direction than in the other two axes. AMS currently requires the ProfileAxis to be aligned with one of the lattice vectors and perpendicular to the remaining lattice vectors.

See also:

[Example demonstrating a NEMD viscosity simulation](#) (page 535)

Warning: CosineShear simulations are typically performed in the NVT ensemble. While using a Barostat is theoretically possible, such a combination has not been physically validated by SCM.

MolecularDynamics

CosineShear

Type

Block

Description

Apply an external acceleration to all atoms of a fluid using a periodic (cosine) function along a selected coordinate axis. This induces a periodic shear flow profile which can be used to determine the viscosity.

Acceleration**Type**

Float

Default value

5e-06

Unit

Angstrom/fs²

Description

Amplitude of the applied cosine shear acceleration profile. The default value should be a rough first guess for water and it needs to be adjusted by experimentation for other systems.

Enabled**Type**

Bool

Default value

No

GUI name

Enable cosine shear

Description

Apply a cosine shear acceleration profile for a NEMD calculation of viscosity.

FlowDirection**Type**

Float List

Default value

[1.0, 0.0, 0.0]

Description

The direction in which to apply the shear acceleration, in Cartesian coordinates. The magnitude of this vector is ignored (AMS will normalize it internally). FlowDirection has to be perpendicular to ProfileAxis.

ProfileAxis**Type**

Multiple Choice

Default value

Z

Options

[X, Y, Z]

Description

The Cartesian coordinate axis along which the cosine wave runs

This generates a velocity profile in the steady state that looks like: $v_x(z) = V \cos(\frac{2\pi z}{L_z})$. After the system reaches a steady state, the velocity amplitude V will be approximately constant, although typically relatively noisy. Averaging V over a long enough trajectory yields a reasonable estimate of the steady-state flow velocity, which can then be used to calculate

the viscosity as $\eta = \frac{A\rho L_z^2}{4\pi^2}$. The measured value of V averaged over every `Trajectory SamplingFreq` timesteps is written to the trajectory as *MeanCosineShearVelocity* in the units of Bohr/fs.

The calculated viscosity value will be different from the equilibrium viscosity of the studied liquid due to shear thinning and heating caused by the applied acceleration A . It is thus necessary to measure the viscosity at several different values of A and extrapolate the obtained values to the zero-shear limit $A = 0$. To do this, it is useful to calculate the “shear parameter” $S = \frac{A^2 m \rho L_z^2}{8\pi^2 N_f k_B}$, where m is the total mass of the system and N_f is the number of degrees of freedom. The measured viscosity η is then approximately a linear function of S , enabling a straightforward extrapolation to zero S . Typical values of the shear parameter S in SI units are on the order of $10^9 - 10^{10} \text{ kg m s}^{-2} \text{ K}$.

4.9.11 Applying reflective walls

Note: This feature is still experimental. It may contain bugs and some input keys are still subject to change in future releases of AMS. If you are interested in trying it out, monitor the [bug fix changelog](https://www.scm.com/support/downloads/changelog/) (<https://www.scm.com/support/downloads/changelog/>) for updates and feel free to get in touch with us at support@scm.com if you run into any issues.

The `ReflectiveWall` keyword can be used to define impenetrable planes in space, causing molecules to bounce off by an elastic collision. Therefore, the main usage of reflective walls is to make sure that molecules will stay within a certain area in space.

The reflective wall is defined by a normal vector (`Axis`) and a `Threshold`. The plane of the wall is perpendicular to `Axis` and passes through a point given by multiplying `Axis` by `Threshold`. Any atom moving in the direction of `Axis` will be reflected back, but the opposite does not apply. Particles moving in the direction opposite to `Axis` will not be affected (the wall works like a one-way mirror). The `Axis` vector does not need to be normalized to unity, AMS will normalize it internally and adjust `Threshold` accordingly.

MolecularDynamics

ReflectiveWall

Type

Block

Recurring

True

Description

Apply a reflective wall in space

Axis

Type

Float List

Unit

Angstrom

Description

Defines the normal vector perpendicular to the plane of the reflective wall. Any particle moving in this direction will be reflected back.

Region

Type

String

Recurring

True

Description

Apply the reflective wall to all atoms in this region.

Threshold**Type**

Float

Unit

Angstrom

Description

Defines the threshold value determining the position of the reflective wall. If the dot product of a position of a particle with Axis exceeds Threshold, the particle will be reflected. This means that the plane of the wall passes through a point given by Axis times Threshold.

4.9.12 Exit Conditions

It is possible to define exit conditions to end a molecular dynamics calculations before all timesteps have been performed. If an exit condition is met, the variable `History%ExitConditionMsg` is populated with a message describing why the molecular dynamics task was stopped. When an exit condition is met, its status is considered to be NORMAL TERMINATION.

```
ExitCondition
  AtomsTooClose
    MinimumDistance float
    PairCalculation [NeighborList | DistanceMatrix]
  End
  EngineEnergyUncertainty
    MaxUncertainty float
    Normalization float
  End
  EngineGradientsUncertainty
    MaxUncertainty float
  End
  Type [AtomsTooClose | EngineEnergyUncertainty | EngineGradientsUncertainty]
End
```

ExitCondition**Type**

Block

Recurring

True

Description

If any of the specified exitconditions are met, the AMS driver will exit cleanly.

AtomsTooClose**Type**

Block

Description

If any pair of atoms is closer than the specified minimum value, the program will exit cleanly.

MinimumDistance**Type**

Float

Default value

0.7

Unit

Angstrom

Description

Two atoms closer than this threshold value are considered too close.

PairCalculation**Type**

Multiple Choice

Default value

NeighborList

Options

[NeighborList, DistanceMatrix]

Description

Two atoms closer than this threshold value are considered too close.

EngineEnergyUncertainty**Type**

Block

Description

If the engine reports an uncertainty that is too high, the program will exit cleanly.

MaxUncertainty**Type**

Float

Default value

0.001

Unit

Hartree

Description

Threshold for Engine Energy Uncertainty divided by Normalization (by default the number of atoms)

Normalization**Type**

Float

Value Rangevalue \geq 0.0**Description**

Divide the reported Engine Energy Uncertainty by this normalization. Will divide by the number of atoms if unset.

EngineGradientsUncertainty

Type

Block

Description

If the engine reports an uncertainty in the magnitude of the nuclear gradient of any atom that is too high, the program will exit cleanly.

MaxUncertainty**Type**

Float

Default value

0.01580221

Unit

Hartree/Angstrom

Description

Threshold for Engine Gradients Uncertainty.

Type**Type**

Multiple Choice

Default value

AtomsTooClose

Options

[AtomsTooClose, EngineEnergyUncertainty, EngineGradientsUncertainty]

Description

The type of exitcondition specified

4.10 Trajectory replay

For some applications it is necessary to run calculations on all (or a subset of) frames from a trajectory obtained in another job. Often this is done with a different engine in order to learn something about the differences in the potential energy surface between the engines. This could easily be scripted with *PLAMS* (page 14), but since the 2022.1 release the AMS driver itself also implements a convenient “Replay” task to do this.

The input for the replay task is quite minimal. The following input file would replay every tenth of the first 1000 frames on the trajectory of `oldjob` with the DFTB engine to calculate nuclear gradients:

```
Task Replay

Replay
  File oldjob.results
  Frames 1:1000:10
End

Properties
  Gradients True
End

Engine DFTB
EndEngine
```

See also:

[Example input files](#) (page 539) for replaying PES scans, NEB and MD calculations.

Note that it is not necessary to specify a *System block* (page 31) in the input of a replay job, as the system will just be loaded from the job to be replayed. What properties are evaluated for each frame in the trajectory can be configured in the `Properties` block of the input file. For a list of available properties, see for example the manual pages on *Gradients*, *Hessian*, *Stress tensor*, *Elasticity* (page 235) and *Dipole moment*, *Polarizability*, *Bond orders* (page 307).

Depending on the kind of job that is being replayed, you will get a slightly different default behavior for the selection of frames to be replayed:

- For *PES scans* (page 76) only the frames corresponding to the converged PES points will be replayed. The resulting `ams.rkf` output file will contain the `PESScan` section, as if it had been produced by running a PES scan job.
- For *Nudged Elastic Band (NEB)* (page 83) jobs only the frames corresponding to the converged images are replayed. The resulting `ams.rkf` file will have an `NEB` section that looks like it is from a NEB job that converged immediately.
- All other job types (such as *Molecular Dynamics* (page 101)) will just replay whatever is on the `History` section on the `ams.rkf` file. By default all frames are replayed, but a subset can be chosen with the `Replay%Frames` keyword.

Note that the replay task does not use *driver level parallelism* (page 20). The engine is called on each frame in turn, always restarting from the previously calculated frame. (The restarting should not cause problems, but it can be disabled with the `EngineDebugging%IgnorePreviousResults` option, see [developer options](#) (page 565)).

Details of the Replay task can be configure in the `Replay` block in the input.

```
Replay
  File string
  Frames integer_list
  StoreAllResultFiles Yes/No
End
```

Replay**Type**

Block

Description

Configures the details of the Replay task.

File**Type**

String

GUI name

Restart from

Description

Provide an `ams.rkf` file (or a `.results` folder) from a previously run job to replay. The file needs to contain a `History` section.

Frames**Type**

Integer List

Description

List of frames from the `History` section to recompute.

If not specified the recomputed frames are determined automatically based on the task of the job that is being replayed: PES scans and NEB calculations will only have the converged points replayed, while all other tasks will have all frames recomputed.

Specifying the frames to recompute in the input is probably only useful when replaying trajectories from MolecularDynamics calculations.

StoreAllResultFiles

Type

Bool

Default value

No

Description

If this option is enabled AMS will produce a separate engine output file for every replayed frame.

While basic properties like energy, gradients, stress tensor, etc. are stored anyway on the History section in the AMS driver output file (if they were requested in the Properties block), engine specific properties (e.g. excitations energies from ADF) will only be available if the full result files are stored.

4.11 Grand Canonical Monte Carlo (GCMC)

Tip: Take a look at the [GCMC tutorial](#) and learn how to setup a GCMC calculation.

4.11.1 General info

About Monte Carlo / the Grand Canonical Ensemble

It is best to read a bit about Monte Carlo and ensembles before working with the GCMC code. Almost every book or review text on molecular simulations will do, for example: Frenkel D, Smit B. Understanding molecular simulation: from algorithms to applications. Academic Press; 2002. 672 p.

Wikipedia also has some pages of interest:

- http://en.wikipedia.org/wiki/Monte_Carlo_method
- http://en.wikipedia.org/wiki/Grand_canonical_ensemble

It is important to note that this method heavily relies on random numbers, and simulations are thus non-repeatable in detail, but should converge to the same answer.

About the AMS GCMC code

The GCMC code was originally developed for standalone Reaxff by Thomas Senftle, working as a Graduate Student at Penn State University under the supervision of Dr. Adri van Duin^{1,2}. The original version was a wrapper code that called an external executable to perform the reaxff minimization step and energy calculation, and relied on file modification and parsing to steer the reaxff code and get the results back.

¹ T.P. Senftle, R.J. Meyer, M.J. Janik, A.C.T. van Duin, *Development of a ReaxFF potential for Pd/O and application to palladium oxide formation*, J. Chem. Phys. 139, 044109 (2013) (<https://doi.org/10.1063/1.4815820>)

² T.P. Senftle, A.C.T. van Duin, M.J. Janik, *Determining in situ phases of a nanoparticle catalyst via grand canonical Monte Carlo simulations with the ReaxFF potential*, Catalysis Communications 52, 72–77 (<https://doi.org/10.1016/j.catcom.2013.12.001>)

The code was later rewritten by Hans van Schoot (SCM), in close collaboration with Thomas Senftle, to integrate it directly into the ADF-ReaxFF code. The current version is an AMS re-implementation so the method can be used with almost any engine supported by AMS (support for 3D periodic boundary conditions by the engine is currently a requirement).

4.11.2 Method Details

Overview

The GCMC method will perform a number of Grand Canonical Monte Carlo trial moves (set by the `Iterations` option of the GCMC input block), and accept or reject them based on the energy produced by the geometry optimization of the trial geometry for the given engine. The Monte Carlo algorithm will always accept a step if it results in a decrease of the energy, and accept steps that go up in energy with a probability. This section will give some details about how the method works.

MC Moves (Insert/Delete/Displace/ChangeVolume)

The GCMC method currently supports 4 types of MC Moves: Insert, Delete, Displace (sometimes also called Move), and ChangeVolume. The first three MC moves are always available and the ChangeVolume becomes available only if the `ChangeVolume` option is set to True. The first three move types change coordinates of atoms in the system, while the ChangeVolume move changes the lattice only.

On every MC iteration, the method first selects one of the molecules defined by the `Molecule` input blocks at random and then selects an applicable MC move type. If there are no molecules of this type in the system then no Delete or Displace is attempted. If the selected molecule has the `NoAddRemove` option set then the Insert and Delete moves will not be attempted. If no MC move is possible with the selected molecule type then another one is selected or a VolumeChange is attempted, if allowed. If no moves with any of the provided molecules is possible (i.e. if all molecules have `NoAddRemove` set to True, there is nothing to displace and the volume is fixed) then the program will stop.

The Insert and Displace MC move will rotate the molecule randomly and put it at a random position, and then check if the minimum interatomic distance between the molecule and the rest of the system is within the `MinDistance` and `MaxDistance` boundaries. If the condition is not satisfied, a new set of coordinates is generated and the check is performed again. This is repeated up to `NumAttempts` times before stopping with an error.

The volume change is controlled by the `VolumeChangeMax` keyword. This sets the volume change limit, and it should be a value between 0 and 1. The new volume will be calculated as: $V_{\text{new}} = \exp(\text{random}(-1:1) * \text{VolumeChangeMax}) * V_{\text{old}}$.

Calculating the chemical potential

The chemical potential of the molecule (or atom) reservoir is used when calculating the Boltzmann accept/reject criteria after a MC move is executed. This value can be derived from first principles using statistical mechanics, or equivalently, it can be determined from thermochemical tables available in literature sources.

For example, the proper chemical potential for a GCMC simulation in which single oxygen atoms are exchanged with a reservoir of O₂ gas, should be equal to 1/2 the chemical potential of O₂ at the temperature and pressure of the reservoir. Page 193, 1:

$$\mu_O^{MC}(T, P) = \frac{1}{2} \mu_{O_2}^{MC}(T, P) = \frac{1}{2} \left[\mu_{O_2}^{ref}(T, P_{ref}) + kT \ln \left(\frac{P}{P_{ref}} \right) - E_{O_2}^{diss} \right]$$

where the reference chemical potential $\mu_{O_2}^{ref}(T, P_{ref})$ is the experimentally determined chemical potential of O₂ at T and P_{ref} , $kT \ln \left(\frac{P}{P_{ref}} \right)$ is the pressure correction to the free energy, and $E_{O_2}^{diss}$ is the dissociation energy of the O₂ molecule.

Calculating energies

Because the GCMC simulation adds and deletes atoms or molecules during the runtime, it cannot directly compare the AMS energies for the MC acceptance criteria: inserting a molecule will usually lower the total energy of the system, causing the MC to always accept it, and always reject a deletion. To compensate this, the GCMC method calculates a “corrected” MC energy to compare the trial energy with, consisting of the previously accepted AMS energy and a correction depending on the move:

$E_{old}^{MC} = E_{old}^{AMS} + \mu^{MC}$ for an Insert move;

$E_{old}^{MC} = E_{old}^{AMS} - \mu^{MC}$ for a Delete move;

$E_{old}^{MC} = E_{old}^{AMS}$ for a Displace move;

$E_{old}^{MC} = E_{old}^{AMS} - P(V_{new} - V_{old}) + N_{inserted} \ln \left(\frac{V_{new}^{avail}}{V_{old}^{avail}} \right) kT$ for a ChangeVolume move.

Here, μ^{MC} is the chemical potential of the inserted/deleted molecule, P is the pressure, V is the volume, and $N_{inserted}$ is the total number of MC molecules. The “new” and “old” subscripts refer to the current and the last accepted values. The V^{avail} values are calculated from the MC-available volume as described below.

Calculating volumes

The available volume can be calculated in a few different ways, depending on the `VolumeOption` setting:

- *Free*: volume = total volume - occupied volume - specified vacuum volume (`NonAccessibleVolume`)
- *Total*: volume = total cell volume
- *Accessible*: volume = specified accessible volume (`AccessibleVolume`)
- *FreeAccessible*: volume = specified accessible volume (`AccessibleVolume`) - occupied volume

Here, the occupied volume is calculated as a sum of volumes of atoms that do not belong to the MC part of the system (i.e. that were not inserted during calculation and are not `Removables`). The volume of an atom is calculated using the average of the covalent and the Van der Waals radii of the atom defined in the `atominfo` module used throughout AMS.

The `AccessibleVolume` and `NonAccessibleVolume` keywords can be used to get a more accurate available volume.

Acceptance criteria

An MC move is always accepted if the AMS energy is lower than the corrected MC energy of the last accepted MC move, or if the energy increase is small enough. If the new energy is higher, the code generates a random number between 0 and 1, and accepts the move if the random number is greater than:

```
prob = preFactor * exp(-Beta*deltaE)
```

The prefactor is calculated (for insert and delete moves) using the deBroglie wavelength of the inserted molecules, the number of inserted molecules and the available MC volume of the system.

4.11.3 Input

The GCMC functionality in AMS is triggered using the following Task key:

```
Task GCMC
```

```
GCMC
  AccessibleVolume float
  Box
    Amax float
    Amin float
    Bmax float
    Bmin float
    Cmax float
    Cmin float
  End
  Ensemble [Mu-VT | Mu-PT]
  Iterations integer
```

(continues on next page)

(continued from previous page)

```

MapAtomsToOriginalCell Yes/No
MaxDistance float
MinDistance float
Molecule
    ChemicalPotential float
    NoAddRemove Yes/No
    SystemName string
End
NonAccessibleVolume float
NumAttempts integer
Pressure float
Removables # Non-standard block. See details.
...
End
Restart string
SwapAtoms
    Probability float
    Regions string
End
Temperature float
UseGCPreFactor Yes/No
VolumeChangeMax float
VolumeOption [Free | Total | Accessible | FreeAccessible]
End

```

The following keys are common for all GCMC calculations and should always be specified. The ChemicalPotential value should correspond to the μ^{MC} expression above, and not to the experimental chemical potential μ^{ref} , which means it should include the (engine-dependent) free molecule's energy.

GCMC**Molecule****Type**

Block

Recurring

True

GUI name

Molecules

Description

This block defines the molecule (or atom) that can be inserted/moved/deleted with the MC method. The coordinates should form a reasonable structure. The MC code uses these coordinates during the insertion step by giving them a random rotation, followed by a random translation to generate a random position of the molecule inside the box. Currently, there is no check to make sure all atoms of the molecule stay inside the simulation box. The program does check that the MaxDistance/MinDistance conditions are satisfied.

ChemicalPotential**Type**

Float

Unit

Hartree

Description

Chemical potential of the molecule (or atom) reservoir.

It is used when calculating the Boltzmann accept/reject criteria after a MC move is executed. This value can be derived from first principles using statistical mechanics, or equivalently, it can be determined from thermochemical tables available in literature sources.

For example, the proper chemical potential for a GCMC simulation in which single oxygen atoms are exchanged with a reservoir of O₂ gas, should equal 1/2 the chemical potential of O₂ at the temperature and pressure of the reservoir:

$$\text{cmpot} = \mu_{\text{O}}(T,P) = 1/2 * \mu_{\text{O}_2}(T,P) = 1/2 * [\mu_{\text{ref}}(T,P_{\text{ref}}) + kT * \log(P/P_{\text{ref}}) - E_{\text{diss}}]$$

where the reference chemical potential [$\mu_{\text{ref}}(T,P_{\text{ref}})$] is the experimentally determined chemical potential of O₂ at T and P_{ref}; $kT * \log(P/P_{\text{ref}})$ is the pressure correction to the free energy, and E_{diss} is the dissociation energy of the O₂ molecule.

NoAddRemove

Type

Bool

Default value

No

GUI name

Fix molecule

Description

Set to True to tell the GCMC code to keep the number of molecules/atoms of this type fixed.

It will thus disable Insert/Delete moves on this type, meaning it can only do a displacement move, or volume change move (for an NPT ensemble).

SystemName

Type

String

GUI name

Molecule

Description

String ID of a named [System] to be inserted. The lattice specified with this System, if any, is ignored and the main system's lattice is used instead.

Iterations

Type

Integer

GUI name

Number of GCMC iterations

Description

Number of GCMC moves.

Temperature

Type

Float

Default value

300.0

Unit

Kelvin

Description

Temperature of the simulation. Increase the temperature to improve the chance of accepting steps that result in a higher energy.

The following keys are related to Insert and Displace moves.

GCMC**NumAttempts****Type**

Integer

Default value

1000

GUI name

Max tries

Description

Try inserting/moving the selected molecule up to the specified number of times or until all constraints are satisfied. If all attempts fail a message will be printed and the simulation will stop. If the MaxDistance-MinDistance interval is small this number may have to be large.

MinDistance**Type**

Float

Default value

0.3

Unit

Angstrom

GUI name

Add molecules not closer than

Description

Keep the minimal distance to other atoms of the system when adding the molecule.

MaxDistance**Type**

Float

Default value

3.0

Unit

Angstrom

GUI name

Add molecules within

Description

The max distance to other atoms of the system when adding the molecule.

The following keys influence computation of the acceptance probability and of the MC energy correction.

GCMC

UseGCPreFactor**Type**

Bool

Default value

Yes

GUI name

Use GC prefactor

Description

Use the GC pre-exponential factor for probability.

VolumeOption**Type**

Multiple Choice

Default value

Free

Options

[Free, Total, Accessible, FreeAccessible]

GUI name

Volume method

Description

Specifies the method to calculate the volume used to calculate the GC pre-exponential factor and the energy correction in the Mu-PT ensemble:

Free: $V = \text{totalVolume} - \text{occupiedVolume} - \text{NonAccessibleVolume}$;

Total: $V = \text{totalVolume}$;

Accessible: $V = \text{AccessibleVolume}$;

FreeAccessible: $V = \text{AccessibleVolume} - \text{occupiedVolume}$.

The AccessibleVolume and NonAccessibleVolume are specified in the input, the occupiedVolume is calculated as a sum of atomic volumes.

AccessibleVolume**Type**

Float

Default value

0.0

Description

Volume available to GCMC, in cubic Angstroms. AccessibleVolume should be specified for “Accessible” and “FreeAccessible” [VolumeOption].

NonAccessibleVolume**Type**

Float

Default value

0.0

GUI name

Non-accessible volume

Description

Volume not available to GCMC, in cubic Angstroms. NonAccessibleVolume may be specified for the “Free” [VolumeOption] to reduce the accessible volume.

The following keys apply to the ensemble choice and options for the Mu-PT ensemble.

GCMC**Ensemble****Type**

Multiple Choice

Default value

Mu-VT

Options

[Mu-VT, Mu-PT]

Description

Select the MC ensemble: Mu-VT for fixed volume or Mu-PT for variable volume. When the Mu-PT ensemble is selected the [Pressure] and [VolumeChangeMax] should also be specified.

VolumeChangeMax**Type**

Float

Default value

0.05

Description

Fractional value by which logarithm of the volume is allowed to change at each step. The new volume is then calculated as $V_{\text{new}} = \exp(\text{random}(-1:1) * \text{VolumeChangeMax}) * V_{\text{old}}$

Pressure**Type**

Float

Default value

0.0

Unit

Pascal

Description

Pressure used to calculate the energy correction in the Mu-PT ensemble. Set it to zero for incompressible solid systems unless at very high pressures.

The GCMC code can insert multiple atom/molecule types in a single simulation, so it needs to keep track of what atom belongs to which insert. This information is automatically stored and updated when insertion/deletion/moving of atoms or molecules during the simulation, but is by default unknown for the atoms of the starting geometry. The GCMC code will therefore by default not modify the atoms in the original input in the MC trial moves. The `Restart` key and the `Removables` block are two ways to provide information about Deletable/Movable atoms/molecules in the input structure. If the `Restart` key is present the `Removables` block will be ignored.

GCMC**Restart****Type**

String

Description

Name of an RKF restart file. Upon restart, the information about the GCMC input parameters, the initial system (atomic coordinates, lattice, charge, etc.) and the MC molecules (both already inserted and to be inserted) are read from the restart file. The global GCMC input parameters and the MC Molecules can be modified from input. Any parameter not specified in the input will use its value from the restart file (i.e. not the default value). Molecules found in the restart file do not have to be present as named Systems in the input, however if there is a System present that matches the name of a molecule from restart then the System's geometry will replace that found in the restart file. It is also possible to specify new Molecules in the input, which will be added to the pool of the MC molecules from restart.

Removables**Type**

Non-standard block

Description

The Removables can be used to specify a list of molecules that can be removed or moved during this GCMC calculation. Molecules are specified one per line in the format following format:

MoleculeName atom1 atom2 ...

The MoleculeName must match a name specified in one of the [Molecule] blocks. The atom indices refer to the whole input System and the number of atoms must match that in the specified Molecule. A suitable Removables block is written to the standard output after each accepted MC move. If you do so then you should also replace the initial atomic coordinates with the ones found in the same file. If a [Restart] key is present then the Removables block is ignored.

An example of the Removables block:

```
Removables
  Oatom 41
  O2 44 45
  Oatom 42
  Oatom 43
End
```

This example specifies that 5 atoms belong to 4 different GCMC molecules of two different types, Oatom and O2. Thus in addition to the main input System there should be at least two additional Systems defined, one called "Oatom" (containing one atom) and the other "O2" (containing two atoms). The first one was inserted three times (atoms 41, 42, and 43) and the second one was inserted once.

Finally there are more technical keywords:

GCMC**MapAtomsToOriginalCell****Type**

Bool

Default value

Yes

Description

Keeps the atom (mostly) in the original cell by mapping them back before the geometry optimizations.

Note that the GeometryOptimization block is also read by the GCMC task, and the settings used for the individual optimizations. The documentation for these keywords can be found in the [Geometry Optimization](#) (page 49) section of this manual.

4.11.4 Output

In addition to the standard KF variables in the “History” section on `ams.rkf` such as “Coords” and “Energy”, the following GCMC-specific variables are also created for each accepted MC step:

- *MCMove* - integer index of the MC move.
- *MCMoveType* - string containing the type of the MC move.
- *MCMolecule* - string containing the name of the inserted/displaced/removed molecule.
- *Converged* - a Fortran logical value containing the convergence status of the given geometry.

Results of a GCMC calculation are stored in the GCMC section of the RKF file, in a number of variables. The following variables contain a summary of the MC statistics up to and including the latest step:

- *NIterMCtried* - the latest iteration number.
- *NIterMCaccept* - the number of accepted MC moves.
- *NIterMCreject* - the number of rejected MC moves.
- *NMCacceptAdd* - the number of accepted MC molecule insertions.
- *NMCacceptRemove* - the number of accepted MC molecule removals.
- *NMCacceptMove* - the number of accepted MC molecule moves.
- *NMCacceptVolume* - the number of accepted volume changes.
- *NMCrejectAdd* - the number of rejected MC molecule insertions.
- *NMCrejectRemove* - the number of rejected MC molecule removals.
- *NMCrejectMove* - the number of rejected MC molecule moves.
- *NMCrejectVolume* - the number of rejected volume changes.

The following variables (actually arrays of the size `Iterations`) in the GCMC section contain the detailed information about all MC moves in the current simulation. Only the first *NIterMCtried* elements of each array contain valid data.

- *HistoryAccepted* - MC move status value (1 - accepted, 0 - rejected, -1 - not done yet).
- *HistoryAMSEnergy* - the AMS energy (the E^{AMS} above).
- *HistoryMCenergy* - the corrected MC energy ($E^{MC} = E^{AMS} - \sum \mu_i^{MC}$, where $\sum \mu_i^{MC}$ is the total chemical potential of all inserted molecules).
- *HistoryVolume* - the simulation box volume.
- *HistoryMoveType* - the MC move type index (0 - insert, 1 - delete, 2 - displace, 3 - change volume). The name of the move type with index *i* can be found in the *MoveType(i)* variable.
- *HistoryMoleculeType* - the inserted/deleted/displaced molecule type index. The name of the molecule type with index *i* can be found in the *MoleculeName(i)* variable.
- *HistoryMoleculeIndex* - the inserted/deleted/displaced molecule index within its type.

4.12 Automated PES Exploration

In chemistry and materials science, two types of Potential Energy Surface (PES) critical points are of particular interest: local minima and (first-order) saddle points.

The `PESExploration` task consists of a set of algorithms (which we will refer to as *jobs*) that will automatically explore the PES of a given system, looking for local minima and saddle points.

The available PES exploration jobs are:

Process Search (page 213)

A composite method for finding escape mechanisms from a state. This will find both local minima and saddle points.

Basin Hopping (page 219)

A Monte Carlo method for finding local minima.

Saddle Search (page 214)

A single-ended method for finding nearby saddle points.

Landscape Refinement (page 224)

Given a pre-calculated Energy Landscape, re-optimize local minima and saddle points using a different computational engine/settings.

Binding Sites (page 229)

Given a pre-calculated Energy Landscape, compute the binding sites.

The AMS driver links to the client program of the [EON software package](http://theory.cm.utexas.edu/eon/index.html) (<http://theory.cm.utexas.edu/eon/index.html>)¹ and uses its implementation of the dimer method. EON is developed by the University of Texas at Austin and the University of Iceland. See also the *Required citations* (page 571) section.

See also:

- PES Exploration GUI tutorials:
 - [Automated reaction pathway discovery for hydrohalogenation](#)
 - [Cluster Growth: Cobalt Clusters](#)
 - [Water dissociation on an oxide surface](#)
- Examples:
 - *Example: Basin Hopping for Ar 13 cluster* (page 503)
 - *Example: PES Exploration, Process Search for alanine with PLAMS* (page 505)
 - *Example: PES Exploration, Binding Sites for O on Pt 111* (page 505)

Tip: There is a separate tool specifically for the generation of conformers, see this [manual page](#) (page 344). While the PES exploration can in principle also be used to find conformers, using the dedicated *Conformers tool* (page 344) is usually easier and faster. Note that the set of minima found by the Conformers tool can be used as input for a PES exploration, see [this example](#) (page 508).

¹ Samuel T Chill, Matthew Welborn, Rye Terrell, Liang Zhang, Jean-Claude Berthet, Andreas Pedersen, Hannes Jónsson and Graeme Henkelman *EON: software for long time simulations of atomic scale systems*, *Modelling Simul. Mater. Sci. Eng.* 22 055002 (2014) (<https://doi.org/10.1088/0965-0393/22/5/055002>)

4.12.1 Overview

While many details of a PES exploration calculation depend on the specific *job* (page 205) selected, some aspects are common to all PES explorations jobs. Here we give a brief overview of a typical PES exploration, using a Process Search job as example.

A PES exploration calculation generally consists of multiple *expeditions*, each with several *explorers*.

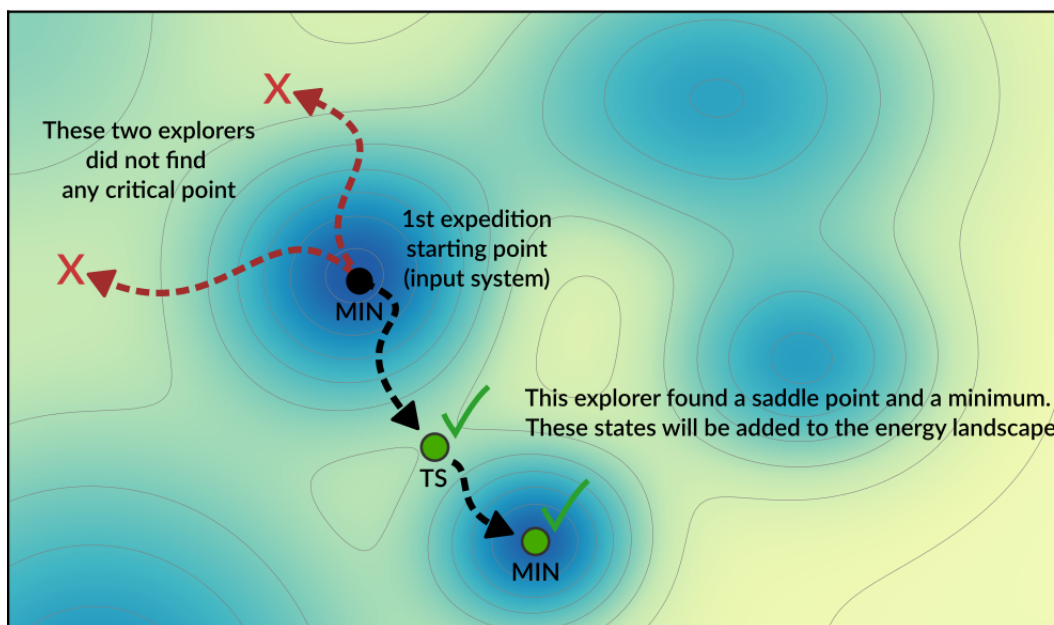
Explorers are given an initial structure, and their goal is to find a nearby critical point (a local minimum, a saddle point or both). The explorer moves around the PES by modifying the atomic positions of the system and by using the specified *Engine* (page 315) to compute energy and gradients.

An *expedition* is a collection of *explorers* all starting from the same point on the PES.

Before setting off for the first expedition, the input structure is optimized and added to the *Energy Landscape*, which is the database of all interesting points found during the exploration (see the section *Results: the “Energy Landscape”* (page 208) for more details).

Starting from this initial structure, a number of PES explorers (3 in the diagram below) will set off in random directions exploring the potential energy surface and looking for nearby critical points.

1st expedition with 3 explorers



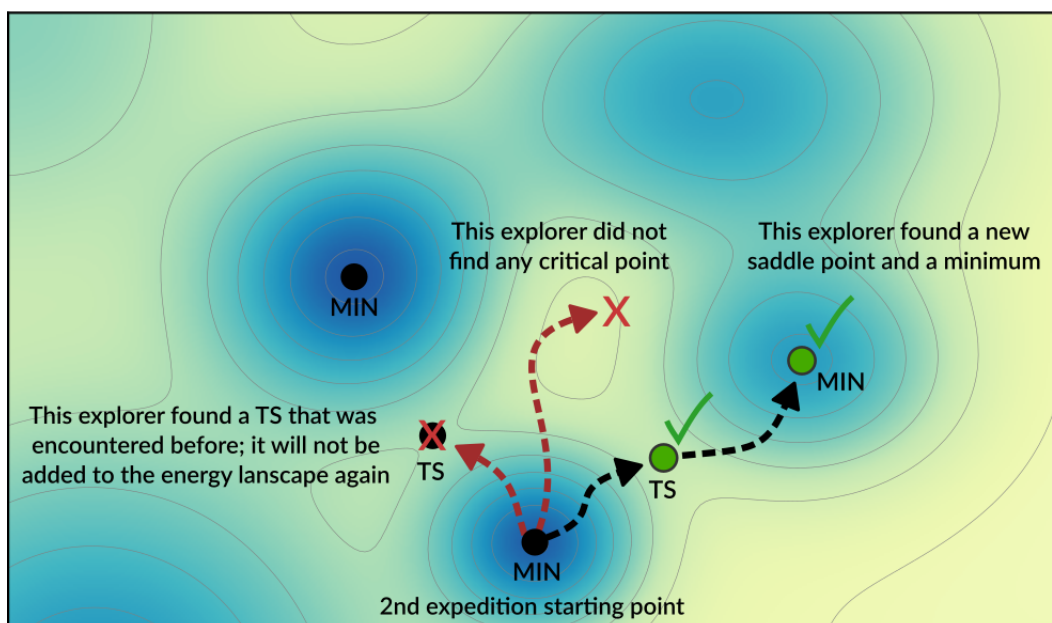
Explorers have “stop conditions” (e.g. a maximum number of steps or a maximum energy above the starting point) so in general not all explorers will successfully find a critical point. In this case, only one of the explorers found relevant critical points: a saddle point and a local minimum. These newly found critical points are added to the Energy Landscape.

After the first expedition is over, the program will start with the second expedition. The starting point for the next expedition will be a randomly chosen local minimum from the list of minima present in the Energy Landscape (the possible starting points for an exploration are called *seed states*. See also the *DynamicSeedStates* (page 205) option).

In this example, the starting point of the second expedition is the new local minimum found during the first expedition. A new set of explorers will set off in random directions from this point.

In the diagram above, one of the explorer found a saddle point that was already found in a previous exploration. This structure will not be added to Energy Landscape since it was already *seen* before (see the *Structure comparison* (page 211) section for more details). The newly found critical points are added to the Energy Landscape.

2nd expedition with 3 explorers



Usually, many expeditions and/or many explorers are needed to map the PES, but you should keep in mind that the computation time of the calculation will roughly be proportional to the product $\text{NumExpeditions} \times \text{NumExplorers}$.

By having many explorers you will have a higher chance of comprehensively mapping the PES near the starting point of each expedition. By having many expeditions, you will have a higher chance of traveling further away from the initial structure.

It should be emphasized that the PES exploration task is stochastic in nature, as it uses random numbers to perform initial-displacements. This means that if you run the same calculation twice you might find different critical points.

4.12.2 Job selection and main options

To use one of the PES Exploration procedures you should set the [Task](#) (page 9) to `PESExploration` and specify one of the jobs in the `PESExploration%Job` key:

```
Task PESExploration

PESExploration
  Job [ProcessSearch | BasinHopping | SaddleSearch | LandscapeRefinement | ↵
↵BindingSites]
End
```

The input options for the various jobs are described in the corresponding sections: [Process Search](#) (page 213), [Basin Hopping](#) (page 219), [Saddle Search](#) (page 214), [Landscape Refinement](#) (page 224), [Binding Sites](#) (page 229).

It is then important pick an appropriate number of expeditions and explorers. Having many expeditions and explorers will result in a more comprehensive PES exploration, but since the computation time will roughly be proportional to the product $\text{NumExpeditions} \times \text{NumExplorers}$ you'll need to find an appropriate balance.

```
PESExploration
  NumExpeditions integer
  NumExplorers integer
```

(continues on next page)

(continued from previous page)

```
DynamicSeedStates Yes/No
End
```

PESExploration**Type**

Block

Description

Configures details of the automated PES exploration methods.

NumExpeditions**Type**

Integer

Default value

1

Description

Sets the number of subsequent expeditions our job will consist of. Larger values result in a more comprehensive exploration of the potential energy surface, but will take more computational time.

NumExplorers**Type**

Integer

Default value

1

Description

Sets the number of independent PES explorers dispatched as part of each expedition. Larger values will result in a more comprehensive exploration of the potential energy surface, but will take more computational time. By default an appropriate number of explorers are executed in parallel.

DynamicSeedStates**Type**

Bool

Default value

Yes

Description

Whether subsequent expeditions may start from states discovered by previous expeditions. This should lead to a more comprehensive exploration of the potential energy surface. Disabling this will focus the PES exploration around the initial seed states.

The following miscellaneous input option generally apply to most PES exploration calculations:

```
PESExploration
  WriteHistory [None | Converged | All]
  Temperature float
  FiniteDifference float
End
```

PESExploration

Type

Block

Description

Configures details of the automated PES exploration methods.

WriteHistory**Type**

Multiple Choice

Default value

Converged

Options

[None, Converged, All]

Description

When to write the molecular geometry (and possibly other properties) to the history on the ams.rkf file. The default is to only write the converged geometries to the history. Can be changed to write no frames at all to the history, or write all frames (should only be used when testing because of the performance impact). Note that for parallel calculations, only the first group of processes writes to ams.rkf.

Temperature**Type**

Float

Default value

300.0

Unit

Kelvin

Description

The temperature that the job will run at. This may be used in different ways depending on the job, e.g. acceptance probabilities for Monte-Carlo based jobs, thermostating for dynamics based jobs, kinetic prefactors for jobs that find transition states. Some jobs may not use this temperature at all.

FiniteDifference**Type**

Float

Default value

0.0026458861

Unit

Angstrom

Description

The finite difference distance to use for Dimer, Hessian, Lanczos, and optimization methods.

4.12.3 Results: the “Energy Landscape”

The results of a PES Exploration are the structures and energies of the critical points found. There are multiple ways for you to inspect the results:

- The energy landscape can be visualized using the AMSMovie GUI module (In AMSMovie: File → Open and select the `ams.rkf` file of your calculation). See the GUI tutorials (e.g. [Automated reaction pathway discovery for hydrohalogenation](#)) for more details.
- The results are printed to the text output under the header `Final Energy Landscape`, see [below](#) (page 208).
- The results are stored on the `ams.rkf` binary results file in the section `EnergyLandscape`
- In **PLAMS**, you can use the `get_energy_landscape` method of the `AMSResults` object to conveniently extract the results.

Results on the text output

These are printed at the end of the text output under the header `Final Energy Landscape`. Here is an output example for a `ProcessSearch` job for the simple `HCN` molecule (computed with the `DFTB` engine):

```
-----
Final Energy Landscape
-----
```

Id	Energy (a.u.)	RE (eV)	RE (kcal/mol)	Counts	Crit. point
1	-5.782789	0.00000	0.0000	4	Min
2	-5.748792	0.92510	21.3334	4	Min
3	-5.689025	2.55143	58.8375	4	TS

```
→ 1 <--> 2
```

```
Number of configurations      3
Number of local minima      2
Number of transition states  1
Energy range (a.u.)         0.093763
Energy range (eV)           2.55143
Energy range (kcal/mol)     58.8375
```

```
Configurations
-----
```

```
3
Id 1 Energy(hartree) -5.78278852 isTS=False
H   -5.05604362312999  1.04394551246415  0.36836337958238
C   -4.06812264817497  0.95005534859126  -0.01533183100739
N   -3.00601476684712  0.84907755770632  -0.42810754191928
3
Id 2 Energy(hartree) -5.74879169 isTS=False
H   -4.01886309319822  1.09046314142766  1.01610923624838
C   -4.06910389782095  0.79751242545583  -1.12016730790896
N   -4.04221404713293  0.95510285187825  0.02898207831626
3
Id 3 Energy(hartree) -5.68902502 isTS=True
H   -4.51488871821561  1.04261603660424  0.52096081259126
C   -4.32990923520194  0.88203060069718  -0.58636688911847
N   -3.28538308473452  0.91843178146029  -0.00966991681712
```

From this you can see that 3 critical points were found: 2 minima (Id 1 and Id 2) and 1 transition state (Id 3) connecting the first two states (indicated by 1 <--> 2).

Under `Counts` you can see how many times each state was encountered during the exploration (often the same state is found multiple times) See the [Structure comparison](#) (page 211) section for more details on how states are compared.

RE is the “relative energy” with respect to the lowest-energy state found.

Under the header `Configurations` you will find the XYZ geometries (in Angstrom) corresponding to the various critical points.

4.12.4 Continue a PES exploration from a previous calculation

You can load an Energy Landscape obtained from a PES exploration calculation (or by the [Conformers utility](#) (page 344)) and use it as starting point for a new PES exploration. In this way you can extend your Energy Landscape, and potentially use different PES exploration algorithms.

To load a previously computed energy landscape, use the `PESExploration%LoadEnergyLandscape%Path` option (note: you should still provide an input system in the [System](#) (page 31) block even if you are loading a previous Energy Landscape. The input system will be optimized and added to the energy landscape as a minimum).

It is often convenient to load only some of the states from a previous calculation; this can be done via the `Remove` or `KeepOnly` input options. Specify the seed states with the `SeedStates` option.

These are all the input options related to the loading an Energy Landscape:

```
PESExploration
  LoadEnergyLandscape
    GenerateSymmetryImages Yes/No
    KeepOnly integer_list
    Path string
    Remove integer_list
    RemoveWithNoBindingSites Yes/No
    SeedStates integer_list
  End
End
```

PESExploration

LoadEnergyLandscape

Type

Block

Description

Options related to the loading of an Energy Landscape from a previous calculation.

GenerateSymmetryImages

Type

Bool

Default value

No

Description

By activating this option, after loading the energy landscape, it will create the complete set of symmetry-related copies by using the symmetry operators of the reference structure. Be aware that rkf result files of the generated symmetry images are copies from the parent structures but only atomic coordinates are updated.

KeepOnly**Type**

Integer List

GUI name

List of states to keep

Description

Upon loading the Energy Landscape, only keep the states specified here. The states should be specified via a list of integers referring to the indices of the states you want to keep.

Path**Type**

String

GUI name

Load energy landscape from

Description

AMS results folder to load an energy landscape from. In the text input file, you may alternatively specify a `.con` file in the native EON format.

Remove**Type**

Integer List

GUI name

List of states to remove

Description

Upon loading the Energy Landscape, remove (i.e. do not load) the states specified here. The states should be specified via a list of integers referring to the indices of the states you want to remove (i.e. the states you don't want to load).

RemoveWithNoBindingSites**Type**

Bool

Default value

No

Description

Upon loading the Energy Landscape, it removes states with no associated binding sites. Associated transition states are also removed. This is an advantageous option to remove physisorbed states automatically. Notice that it requires that the previous calculation was executed, enabling the option `[BindingSites%Calculate]`.

SeedStates**Type**

Integer List

GUI name

List of seed states

Description

By default when you start a new PES Exploration from a loaded Energy Landscape, expeditions can start from any of the loaded minima. By using this input option, you can instruct the program to only use some of the states as 'expedition starting point'. The states that serve

as ‘expedition starting points’ should be specified via a list of integers referring to the indices of the states.

4.12.5 Troubleshooting

The PES exploration in AMS is quite sensitive to noise on the potential energy surface. A too noisy PES may result in bad convergence of the individual explorers, or states being rejected because their *PES point character* (page 239) is not the expected one.

In case of problems we therefore advise to set engine options that result in a smoother PES. Often this can be accomplished by setting the `NumericalQuality` keyword in the engine input.

Especially the `ReaxFF` engine has a rather noisy PES by default, which may cause problems in a PES exploration job. Please refer to the `ReaxFF` manual for advice on how to smoothen the `ReaxFF` potential energy surface:

- [ReaxFF manual: Smoothened potential energy surface](#)
- [ReaxFF manual: Geometry optimization issues](#)

4.12.6 Structure comparison

During a PES exploration some critical points will most likely be encountered more than once.

Whenever a critical point is found, the program will ask itself: is this a new structure, or have I already encountered this before? If the structure was not seen before (i.e. it’s a novel configuration), it will be added to the list of structures found during the exploration. If the structure was already found before, the program will simply increase a counter keeping track of how many times a specific structure was found during the PES exploration.

When deciding if two structures are the same or not, the program will compare both the geometry and the energy of the two configurations. Several parameters and thresholds can be tweaked in the `PESExploration%StructureComparison` block to tell AMS under which conditions two systems should be considered equal.

```
PESExploration
  StructureComparison
    CheckRotation Yes/No
    CheckSymmetry Yes/No
    DistanceDifference float
    EnergyDifference float
    IndistinguishableAtoms Yes/No
    NeighborCutoff float
    RemoveTranslation Yes/No
  End
End
```

PESExploration

StructureComparison

Type

Block

Description

Settings for structure comparison.

CheckRotation

Type

Bool

Description

Rotates the system optimally before comparing structures. The default is to do this only for molecular systems when there are no fixed atom constraints.

CheckSymmetry**Type**

Bool

Default value

No

Description

Considers that two systems are equal if they are equivalent by symmetry.

DistanceDifference**Type**

Float

Default value

0.1

Unit

Angstrom

Description

If the distance between two mapped atoms is larger than this threshold, the two configurations are considered different structures.

EnergyDifference**Type**

Float

Default value

0.01

Unit

eV

Description

If the energy difference between two configurations is larger than this threshold, the two configurations are considered to be different structures.

IndistinguishableAtoms**Type**

Bool

Default value

Yes

Description

If yes, the order of the atoms does not affect the structural comparison. Atoms of the same element are then indistinguishable.

NeighborCutoff**Type**

Float

Default value

3.3

Unit

Angstrom

Description

Atoms within this distance of each other are considered neighbors.

RemoveTranslation**Type**

Bool

Description

Translates the system optimally before comparing structures. The default is to do this only when there are no fixed atom constraints.

4.12.7 Process Search job

Overview

Process Search is a composite method for finding escape mechanisms from a state. Each PES explorer will perform the following steps:

- Starting from a minimum, a [SaddleSearch](#) (page 214) calculation is performed to find a nearby saddle point
- If a saddle point is found, two images are placed on either side of the saddle along the min mode.
- The two images are then minimized; one of the images is expected to fall back into the original minimum, while the other image is expected to end up in the product state on the other side of the saddle. If this is case, the product, TS and reactant states are added to the energy landscape.

The Process Search procedure will therefore not only find minima and saddle points, but it will also keep track of which minima each saddle point is connecting.

See also:

- GUI tutorials [Automated reaction pathway discovery for hydrohalogenation and Water dissociation on an oxide surface](#).

Input options

To perform a Process Search calculation you should specify:

```
Task PESExploration
PESExploration
  Job ProcessSearch

  ProcessSearch
    # Options for the Process Search procedure. See description below.
  End

  SaddleSearch
    # Options for the Saddle Search part of the Process Search procedure.
  End
End
```

The options for the saddle search part of a Process Search job are described in the [SaddleSearch](#) (page 214) section.

Options specific to the Process Search procedure can be specified in the `PESExploration%ProcessSearch` block:

```

PESExploration
  ProcessSearch
    MinimizationOffset float
  End
End

```

PESExploration**ProcessSearch****Type**

Block

Description

Input options specific to the process search procedure.

MinimizationOffset**Type**

Float

Default value

0.2

Description

After a saddle is found, images are placed on either side of the saddle along the mode and minimized to ensure that the saddle is connected to the original minimum and to locate the product state. MinimizationOffset is the distance those images are displaced from the saddle.

4.12.8 Saddle Search job

Overview

Saddle Search is a single-ended method for finding nearby saddle points.

In a Saddle Search calculation, each PES explorer starts by performing a random displacement from the initial geometry (or seed state). Then, starting from this point, the explorer searches for a nearby saddle point using a minimum-mode following method (the available min-mode following methods are dimer^{3,4,5} and Lanczos⁶).

If you already know which atoms are likely to be involved in the reactions, you can use the `DisplaceAtomsInRegion` option to nudge the Saddle Search procedure towards the right part of the PES (*regions* (page 37) can be specified in the *system input block* (page 31), or via the “Model → Regions” panel in AMSinput).

³ G. Henkelman and H. Jónsson, *A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives*, J. Chem. Phys. 111, 7010-7022 (1999) (<http://dx.doi.org/10.1063/1.480097>)

⁴ A. Heyden, A.T. Bell, and F.J. Keil *Efficient methods for finding transition states in chemical reactions: Comparison of improved dimer method and partitioned rational function optimization method*, J. Chem. Phys. 123, 224101 (2005) (<http://dx.doi.org/10.1063/1.2104507>)

⁵ J. Kästner and P. Sherwood *Superlinearly converging dimer method for transition state search*, J. Chem. Phys. 128, 014106 (2008) (<http://dx.doi.org/10.1063/1.2815812>)

⁶ R. Malek and N. Mousseau, *Dynamics of Lennard-Jones clusters: A characterization of the activation-relaxation technique*, Phys. Rev. E 62, 7723 (2000) (<http://dx.doi.org/10.1103/PhysRevE.62.7723>)

Input options

To perform a Saddle Search calculation you should specify:

```
Task PESExploration

PESExploration
  Job SaddleSearch

  SaddleSearch
    # Options for the Saddle Search procedure. See description below.
  End
End
```

Options for the Saddle Search procedure can be specified in the PESExploration%SaddleSearch block.

Note: these options also affect the Saddle Search part of *Process Search Jobs* (page 213).

```
PESExploration
  SaddleSearch
    ConvergedForce float
    DisplaceAlongNormalModesActiveModes string
    DisplaceAlongNormalModesWeight float
    DisplaceAtomsInRegion string
    DisplaceAtomsInRegionWeight float
    DisplaceListedAtoms string
    DisplaceListedTypes string
    DisplaceMagnitude float
    DisplaceRadius float
    MaxEnergy float
    MaxIterations integer
    MinEnergyBarrier float
    MinModeMethod [dimer | lanczos]
    RelaxFromSaddlePoint Yes/No
    ZeroModeAbortCurvature float
  End
End
```

PESExploration

SaddleSearch

Type

Block

Description

Configuration for the Saddle Search procedure (used in SaddleSearch and ProcessSearch Jobs).

ConvergedForce

Type

Float

Default value

-1.0

Unit

eV/Angstrom

Description

Convergence threshold for nuclear gradients. Note: Special value of -1.0 means using the

same convergence criterion as the PES explorer's geometry optimizer.

DisplaceAlongNormalModesActiveModes**Type**

String

Default value**GUI name**

Displace active modes

Description

Sets the active modes to be used when the option [SaddleSearch%DisplaceAlongNormalModesWeight] is enabled. e.g. 1,2,3,5. By default, all normal modes are considered active.

DisplaceAlongNormalModesWeight**Type**

Float

Default value

0.0

GUI name

Displace modes weight

Description

The probability of generating a displacement resulting in a random linear combination of the normal modes specified in [SaddleSearch%DisplaceAlongNormalModesActiveModes]. This parameter is a numeric value that should fall within the interval [0.0, 1.0].

DisplaceAtomsInRegion**Type**

String

Default value**Description**

A string corresponding to the name of a region. When performing the initial random displacement, only displace atoms in the specified region.

DisplaceAtomsInRegionWeight**Type**

Float

Default value

0.0

Description

The probability of generating a displacement involving only atoms from the region specified in [SaddleSearch%DisplaceAtomsInRegion]. This parameter is a numeric value that should fall within the interval [0.0, 1.0].

DisplaceListedAtoms**Type**

String

Default value**Description**

Sets the active atoms to be used when the option [Saddle-

Search%DisplaceAlongNormalModesWeight] is enabled. e.g. 1,2,3,5. By default, all normal modes are considered active.

DisplaceListedTypes

Type

String

Default value

Description

???

DisplaceMagnitude

Type

Float

Default value

0.1

Unit

Angstrom

Description

The standard deviation of the Gaussian displacement in each degree of freedom for the selected atoms.

DisplaceRadius

Type

Float

Default value

4.0

Unit

Angstrom

Description

Atoms within this distance of the epicenter will be displaced.

MaxEnergy

Type

Float

Default value

20.0

Unit

eV

Description

The energy (relative to the starting point of the saddle search) at which a saddle search explorer considers the search bad and terminates it.

MaxIterations

Type

Integer

Default value

400

Description

Maximum number of iterations for each saddle search run.

MinEnergyBarrier**Type**

Float

Default value

0.001

Unit

eV

Description

Minimum energy barrier to accept a new transition state.

MinModeMethod**Type**

Multiple Choice

Default value

dimer

Options

[dimer, lanczos]

Description

The minimum-mode following method to use.

RelaxFromSaddlePoint**Type**

Bool

Default value

No

Description

Relaxes the saddle point geometries following the imaginary mode to get both reactants and products.

ZeroModeAbortCurvature**Type**

Float

Default value

0.01

Unit

eV/Angstrom²

Description

The threshold in the frequency below which the minimum mode is considered zero. The calculation is aborted if the negative mode becomes zero.

4.12.9 Basin Hopping job

Overview

Basin hopping² is an iterative Monte Carlo method for PES exploration and global minimization.

An “iteration” in basin hopping consists of the following steps:

1. The atomic coordinates of the system are randomly perturbed (the details of this random perturbation can be configured via the keywords `Displacement`, `DisplacementDistribution`, `SingleAtomDisplace`, `SwapProbability`. See [below](#) (page 219) for more details).
2. A local optimization of the randomly perturbed structure is performed (see also the [Optimizer](#) (page 228) section)
3. The new optimized structure is accepted or rejected based of its energy (if the energy of the newly optimized structure lower than the lowest energy so far, the new structure is readily accepted. Otherwise it is accepted with probability $e^{(-\Delta E/kT)}$ where ΔE is the difference between the energy at the previous iteration and the current energy, T is temperature specified in the `PESExploration%Temperature` keyword, see [Job selection and main options](#) (page 205)).

By iteratively performing these steps, the basin hopping procedure explores the highly-dimensional configuration space of the system, looking for the global minimum and finding local minima along the way. The number of iteration can be specified via the `Steps` keywords (see [below](#) (page 219) for more details).

For a more comprehensive PES exploration, AMS can perform several expeditions (potentially with several explorers for each expedition) in a single calculation. See the `NumExpeditions` and `NumExplorers` keywords in the [Job selection and main options](#) (page 205) section for more details.

See also:

- [Example: Basin Hopping for Ar 13 cluster](#) (page 503)
- GUI tutorials [Cluster Growth: Cobalt Clusters](#) and [Water dissociation on an oxide surface](#).

Input options

To perform a Basin Hopping calculation you should specify:

```
Task PESExploration
PESExploration
  Job BasinHopping

  Temperature float

  BasinHopping
    # Options for the Basin Hopping procedure. See description below.
  End
End
```

The `PESExploration%Temperature` keyword is described in the [Job selection and main options](#) (page 205) section.

The following input keywords are specific to Basin Hopping procedure:

² David J. Wales and Jonathan P. K. Doye *Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms*, J. Phys. Chem. A 1997, 101, 28, 5111–5116 (<https://doi.org/10.1021/jp970984n>)

```
PESExploration
  BasinHopping
    DisplaceAtomsInRegion string
    Displacement float
    MainSystemAsSeed Yes/No
    PESPointCharacterization Yes/No
    PushApartDistance float
    RotateNonListedAtoms Yes/No
    Steps integer
  End
End
```

PESExploration**BasinHopping****Type**

Block

Description

Configures the details of the Basin Hopping subtask.

DisplaceAtomsInRegion**Type**

String

Default value**Description**

If you specify a region name here, only the atoms belonging to this region will be displaced during the basin hopping procedure. For more details on regions, see the documentation on the System definition.

Displacement**Type**

Float

Default value

0.5

Unit

Angstrom

Description

Displacement in each degree of freedom.

MainSystemAsSeed**Type**

Bool

Default value

No

Description

If true, only the main system will be used as a seed state. The main system is not added to the database.

PESPointCharacterization**Type**

Bool

Default value

Yes

Description

If true, a PES point characterization based on a vibrational analysis is carried out to confirm each detected state is an actual local minimum (no imaginary frequencies). Conversely, if this option is false, the PES point characterization is avoided, which will assume that all located states are local minima (zero gradients). Enabling this option is very useful for large systems. It circumvents the need for computing and diagonalizing the Hessian matrix, a typically expensive computational process.

PushApartDistance**Type**

Float

Default value

0.4

Unit

Angstrom

Description

Push atoms apart until no atoms are closer than this distance. This criterion is enforced for the initial structure and all those generated by random displacements.

RotateNonListedAtoms**Type**

Bool

Default value

No

Description

If true, each iteration randomly rotates all atoms as a rigid body, excluding those listed in [BasinHopping%DisplaceListedAtoms], [BasinHopping%DisplaceListedTypes], or [BasinHopping%DisplaceAtomsInRegion].

Steps**Type**

Integer

Default value

20

Description

Number of displace & optimize Monte-Carlo steps to take.

4.12.10 Fragmented states

When studying adsorption processes, it is required to review those situations where a transition state (TS) does not mediate the adsorption process. They are formally known as **non-activated exothermic adsorption** processes. These processes are not discovered naturally in the PES exploration. Since the AMS2022 release, there is a systematic way of including those processes. To this aim, we defined the **fragmented states**, where the adsorbate and adsorbent are infinitely far away from each other.

The fragmented states can be included in the *Energy Landscape* (page 208) through the use of the `CalculateFragments` keyword:

```

PESExploration
  CalculateFragments Yes/No
End

```

PESExploration**CalculateFragments****Type**

Bool

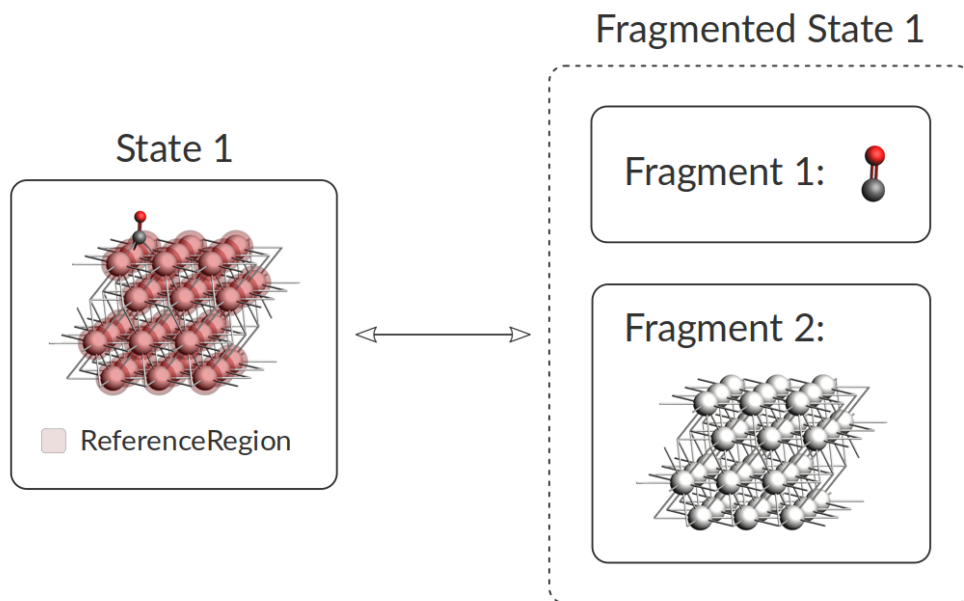
Default value

No

Description

Must be used together with an adsorbent set as the `StatesAlignment%ReferenceRegion`. Runs a final calculation of the adsorbate and adsorbent (marked by the `ReferenceRegion`) individually. The fragmented state is included in the energy landscape.

The keyword has to be used together with a *reference region* (page 234) set via the `StatesAlignment%ReferenceRegion` keyword. The calculation of the fragmented states will happen at the very end of a PES exploration. For every minimum in the energy landscape, two geometry optimizations will be performed: one with only the atoms belonging to the reference region (the adsorbent), the other with only the atoms outside of the reference region (the adsorbate). This relaxes the structure of the adsorbent in the *absence* of the adsorbate, and vice versa.



The total energy of the fragmented state is then simply taken as the sum of the energies of both fragments, and the adsorption energy is the difference between the adsorbed and fragmented state.

$$E(\text{fragmented state 1}) = E(\text{fragment 1}) + E(\text{fragment 2})$$

$$E_{ad} = E(\text{state 1}) - E(\text{fragmented state 1})$$

You will find fragmented states and the fragments in the *text output* (page 208) following the `Final Energy Landscape` section. The example shown above would look like this:

Final Energy Landscape

Id	Energy (a.u.)	RE (eV)	RE (kcal/mol)	Counts	Crit. point
1	-7.651642	0.00000	0.0000	1	Min
2	-7.651572	0.00191	0.0441	1	Min
3	-7.623820	0.75709	17.4589	1	Min
4	-7.622548	0.79170	18.2571	7	TS
↔2<-->3					
5	-7.622430	0.79491	18.3310	5	TS
↔3<-->1					
Number of configurations		5			
Number of local minima		3			
Number of transition states		2			
Energy range (a.u.)		0.029212			
Energy range (eV)		0.79491			
Energy range (kcal/mol)		18.3310			

Fragments

Number of fragments 2

Id	Formula	Energy (a.u.)	Region
1	CO	-0.424454	active
2	Pt36	-7.154286	surface

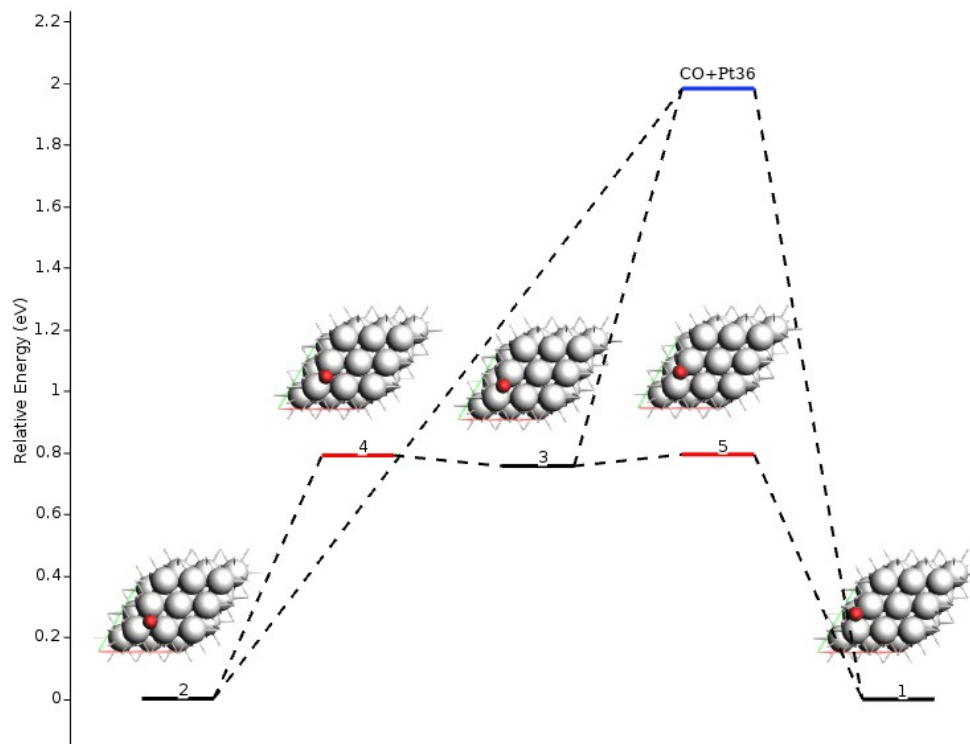
Fragmented States

Number of fragmented states 1

Id	Fragments	Energy (a.u.)	RE (eV)	RE (kcal/mol)	
1	1, 2	-7.578740	1.98377	45.7467	-->1
					-->2
					-->3

Notice the following: 1) one fragment may be included in multiple fragmented states, and 2) multiple states may be connected to the same fragmented state. The latter happens when the individual optimizations of adsorbate and adsorbent yield the same results for multiple states. By definition, these states are considered connected if the difference in energy is lower than `PESExploration%SaddleSearch%MaxEnergy`, which by default is 20.0 eV. Interestingly, you may find, for example, that an adsorbed fragment is in multiple fragmented states together with different surface defects. In other words: the adsorbed fragment is always the same when you remove it from the surface, but the surface may look different, depending on the geometry from which the adsorbent is removed.

The following figure shows a graphical representation of the energy landscape of the example above (made with **amsmovie**). The blue line represents the fragmented state.



4.12.11 Landscape Refinement

The `LandscapeRefinement` Job can be used to re-optimize the critical points (local minima and saddle points) found in a previous PES exploration calculation using a different engine or different engine settings.

Since PES exploration calculations can be computationally demanding, a possible strategy is to first perform a PES exploration using either a fast engine or computationally cheap settings for the engine of choice, and then to *refine* the energy landscape obtained using a more accurate (and computationally more expensive) method.

See also:

Check the [tutorial](#) on automated reaction pathway discovery for hydrohalogenation. There the landscape refinement is used to go from the DFTB level of theory to DFT.

The `LandscapeRefinement` job needs a previously computed Energy Landscape. See the section [Continue a PES exploration from a previous calculation](#) (page 209) for more details.

To perform a Landscape Refinement calculation you should specify:

```
Task LandscapeRefinement
PESExploration
  Job LandscapeRefinement

  LoadEnergyLandscape
    Path path/to/previous/calculation/ams.results
  End
End
```

Warning: If you perform a LandscapeRefinement of an Energy Landscapes obtained with *ProcessSearch* (page 213) job, the connections between TS and minima are **NOT** recomputed by default unless the parameter `LandscapeRefinement%RelaxFromSaddlePoint` is activated. By turning on `LandscapeRefinement%RelaxFromSaddlePoint`, both reactants and products can be reconstructed for each transition state using a geometry optimization that follows the imaginary vibrational mode.

Lets say, for example, that after a *ProcessSearch* calculation using DFTB we found a TS connecting minima with Id 3 and 4. After a *LandscapeRefinement* using a different engine (for example ADF) it is no longer assured that the refined TS will still connect the same two minima.

When you visualize a refined energy landscape using *AMSmovie*, be aware that some of the connections might be incorrect.

4.12.12 Creating and/or Extending an Energy Landscape from the Input File

It is pretty common to have a set of structures or geometries from the literature or previous calculations that you want to use to continue an energy exploration or an energy landscape refinement. In this case, you can use *System* blocks in your input file to create and/or extend an energy landscape.

The header of the *System* (page 31) block is extended using the following syntax:

```
System label ts=Yes/No reactant=string/int product=string/int
  Atoms header
  ...
  End
  ...
End
```

The label `label` is a string. It is required and unequivocally identifies each *System* block defined by the user. The parameters `ts`, `reactant`, and `product` are optional. The latter two, however, can be specified only if `ts` is set to `Yes`. They are otherwise ignored. Notice that it is possible to have transition states with no reactants, or products, or none of them.

The example below illustrates how to use this feature:

```
PESEExploration
  Job LandscapeRefinement
End

Task PESEExploration

System state1
  Atoms
    H      -1.08012803      1.17268451      0.09177647
    O      1.82672244      -0.54837752      0.33538717
    N      0.81411565      0.05127399      0.25048411
    C     -0.17673006      0.63776902      0.16748225
  End
End

System state2
  Atoms
    H      -0.71003072      1.06490526      0.04975824
    O      1.41363000      0.55626133      0.07710644
    N      0.55490964     -0.71415530      0.51752525
```

(continues on next page)

(continued from previous page)

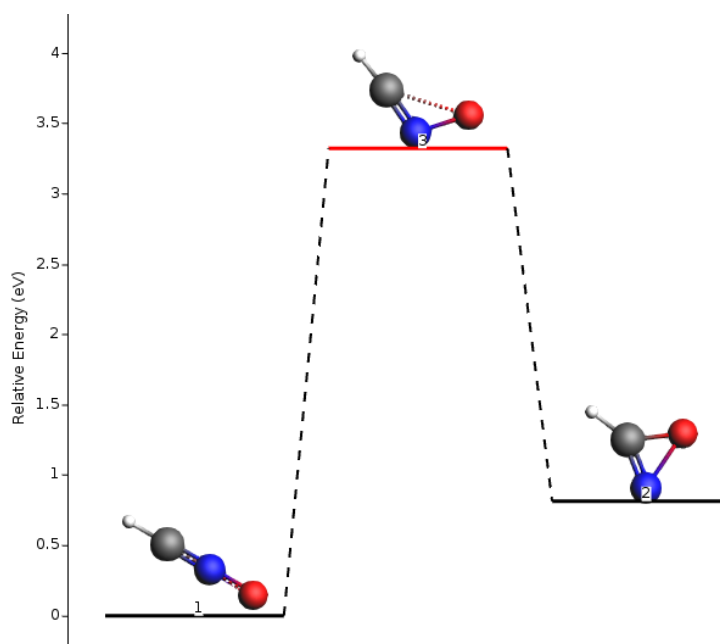
```

      C      0.12547108      0.40633871      0.20074007
    End
  End
System state3 ts=Yes reactant=state1 product=state2
  Atoms
    H      -0.78063395      1.32905463     -0.01879702
    O      1.78011528     -0.06505936      0.19126583
    N      0.53076199     -0.47128241      0.41047761
    C     -0.14626334      0.52063714      0.26218359
  End
End
Engine DFTB
EndEngine

```

This example executes an Energy Landscape Refinement calculation at the DFTB level of theory for the energy landscape loaded from the `System` blocks. It loads the coordinates of two local minima (`state1`, and `state2`) and one transition state (`state3`). The transition state is connected with both minima, being `state1`, the reactant, and `state2`, the product. Below you can see the output text lines and a graphical representation of the obtained energy landscape:

Id	Energy (a.u.)	RE (eV)	RE (kcal/mol)	Counts	Crit. point
1	-10.424825	0.00000	0.0000	1	Min
2	-10.394956	0.81277	18.7428	1	Min
3	-10.302822	3.31986	76.5577	1	TS 1<-->2



To add states to an existing Energy Landscape, use the keyword `PESExploration%LoadEnergyLandscape%Path` to specify the KF file to load. Then, define new links to its states using their ids. The following example shows how to use this feature:

```

PESExploration
  Job LandscapeRefinement

```

(continues on next page)

(continued from previous page)

```

LoadEnergyLandscape
  Path previous.results
End
End

Task PESEExploration

System state3
  Atoms
    H      0.51485007      1.62340128      -0.20665749
    O      1.05880703      0.84482125      -0.00604063
    N      0.23989165      -0.13660458      0.36835485
    C      -0.42956874      -1.01826795      0.68947328
  End
End

System state4 ts=Yes reactant=2 product=state3
  Atoms
    H      -0.10604142      1.11162691      0.00025950
    O      1.08875440      0.73993713      0.03512861
    N      0.74116588      -0.56372211      0.45954036
    C      -0.33989886      0.02550806      0.35020153
  End
End

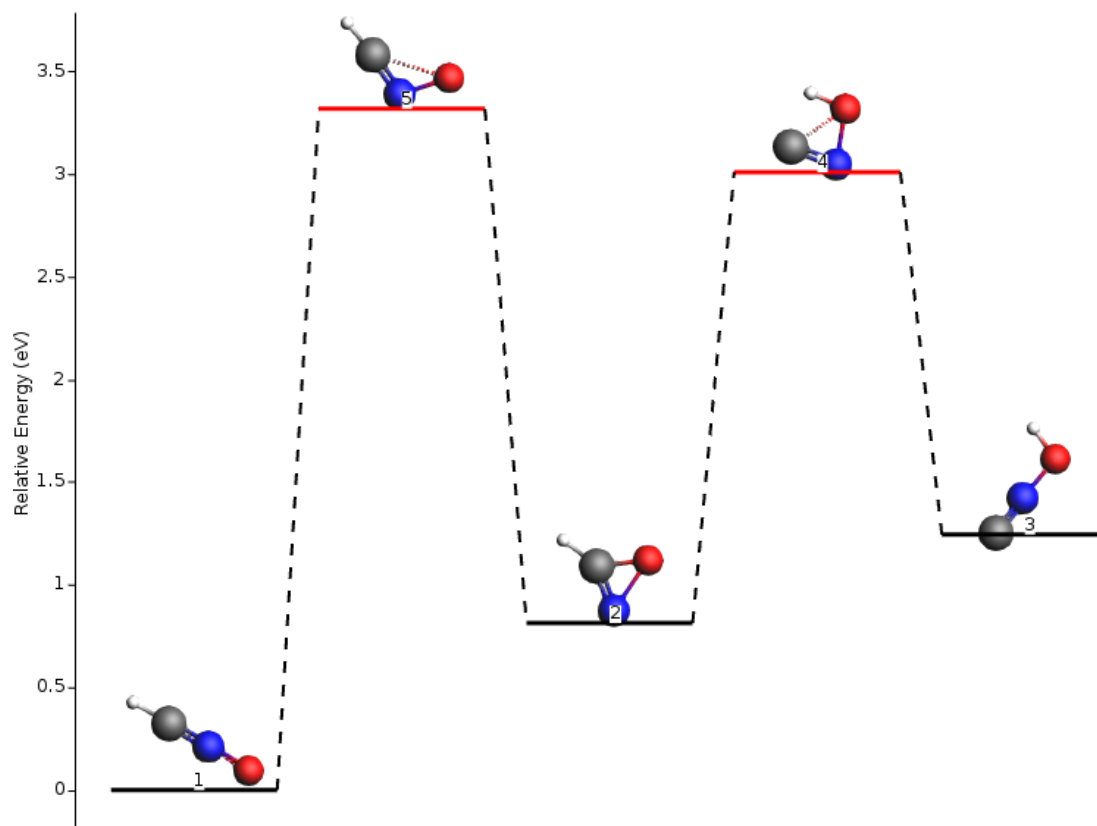
Engine DFTB
EndEngine

```

This example performs an Energy Landscape Refinement calculation at the DFTB level of theory on the energy landscape loaded from the previous calculation's directory results (`previous.results`). It also loads the coordinates of an additional local minimum (`state3`) and one transition state (`state4`) from the input file using the `System` blocks. The latter transition state is linked to the product `state3` (defined in the same input file) and the reactant state with `id=2` from the loaded energy landscape (see image above). The output text lines and the graphical representation of the obtained energy landscape are shown below. Notice that the states have been rearranged according to energy.

Id	Energy (a.u.)	RE (eV)	RE (kcal/mol)	Counts	Crit. point
1	-10.424825	0.00000	0.0000	1	Min
2	-10.394956	0.81277	18.7428	1	Min
3	-10.379113	1.24388	28.6845	1	Min
4	-10.314160	3.01134	69.4432	1	TS 2<-->3
5	-10.302822	3.31986	76.5577	1	TS 1<-->2

Note: If you want to combine this feature of creating and/or extending an energy landscape with any other job (`PESEExploration%Job`) other than `LandscapeRefinement`. Don't forget to include a `System` block with no header (or label) to define the main chemical system.



4.12.13 Optimizer

Geometry optimizations are performed for most PES Explorations job types. In the `PESExploration%Optimizer` block you may configure some of the parameters for these geometry optimizations:

```
PESExploration
  Optimizer
    ConvergedForce float
    MaxIterations integer
    Method [CG | QM | LBFGS | FIRE | SD]
  End
End
```

PESExploration

Optimizer

Type

Block

Description

Configures the details of the geometry optimizers used by the PES explorers.

ConvergedForce

Type

Float

Default value

0.005

Unit

eV/Angstrom

Description

Convergence threshold for nuclear gradients.

MaxIterations**Type**

Integer

Default value

400

Description

Maximum number of iterations allowed for optimizations.

Method**Type**

Multiple Choice

Default value

CG

Options

[CG, QM, LBFGS, FIRE, SD]

Description

Select the method for geometry optimizations.

4.12.14 Binding Sites

Binding sites can be determined from an Energy Landscape.

When computing binding sites you will first need to define a **reference region**, which typically will be a surface or a cluster (*regions* (page 37) can be specified in the *system input block* (page 31), or via the “Model → Regions” panel in AMSinput). AMS will then go through all the local minima of Energy Landscape and mark as ‘binding sites’ the positions where an atom of an adsorbed molecule is neighboring atoms in the reference region.

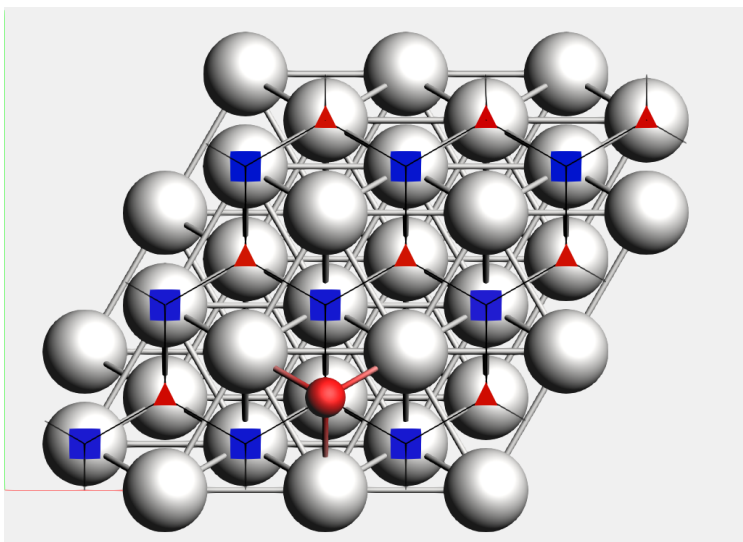
For instance, in *Example: PES Exploration, Binding Sites for O on Pt 111* (page 505), the platinum surface is the reference region, while the oxygen atom is the adsorbate. In the picture below, you can see the oxygen binding sites on the platinum 111 surface (the two different types of binding sites are marked by a blue square and a red triangle respectively).

Lines connecting the the binding sites will be drawn if 1) there is a transition state connecting two local minima associated to these binding-sites (notice that may there are multiple local minima associated with the same binding site), and 2) there is at least an atom (from the adsorbate region) that changes its position from the first binding site to the second one mediated by the same transition state described above. In this process, AMS will align as much as possible all local minima and transition states to the input’s file structure but ignoring the atoms in the region adsorbate trying to establish a common reference frame (see also the `StatesAlignment` input block below).

See also:

- *Example: PES Exploration, Binding Sites for O on Pt 111* (page 505)
- *Cluster Growth: Cobalt Clusters*

Binding sites are labeled according to the number of neighbors they have on each of their coordination shells. It has the format `N<int><int>....`. For example, the label `N334` denotes three neighbors in the first shell, three in the second, and four in the third. The parameter `PESExploration%BindingSites%MaxCoordinationShellsForLabels` or the largest shell contained in the



unit cell, determines the maximum number of coordination shells to consider. For labeling purposes, the center of the binding site is taken to be 1.5 Å from the center of the atoms in the first shell in the direction of the adsorbed atom position. `PESExploration%BindingSites%DistanceDifference` is an important parameter to remember. It is critical because it specifies the tolerance for determining whether or not two atoms are in the same coordination shell based on their distances from the binding site.

The figure below depicts the four types of binding sites observed on a Pt(111) surface. There, you can see the AMS label (e.g., N33, N34, ...) as well as the traditional one (e.g., fcc, hcp, ...). The distances between the binding site and the highlighted atoms on that coordination shell (or neighborhood) are listed below each geometry. If two numbers are listed, it is because two different distances are involved, but they are considered in the same coordination shell because their difference is less than `PESExploration%BindingSites%DistanceDifference`.

The following is an example of the section in the output file describing the obtained binding sites. In particular, you can find the above details regarding distances involved in calculating their labels.

Binding Sites

```
Number of binding sites:      4
Number of connections:      3
```

Binding sites:

id	Label	Aver.E(a.u.)	Stdev.E(a.u.)	x	y	z	Parents
1	N34	-3.773094	0.000000	3.59065	2.09219	3.88471	1 (1)
2	N33	-3.772967	0.000000	4.93783	2.86841	3.88471	2 (1)
3	N223	-3.746180	0.000000	4.25120	2.47351	3.88462	3 (1)
4	N16	-3.683784	0.000000	3.60490	3.59820	3.88382	6 (1)

Labels details:

Label	Center example (Å)			Radius (Å)
N33	4.94	2.87	3.88	2.16 - 2.66
				3.45 - 3.95
N34	3.59	2.09	3.88	2.16 - 2.66
				3.45 - 3.95
N223	4.25	2.47	3.88	2.02 - 2.52

(continues on next page)

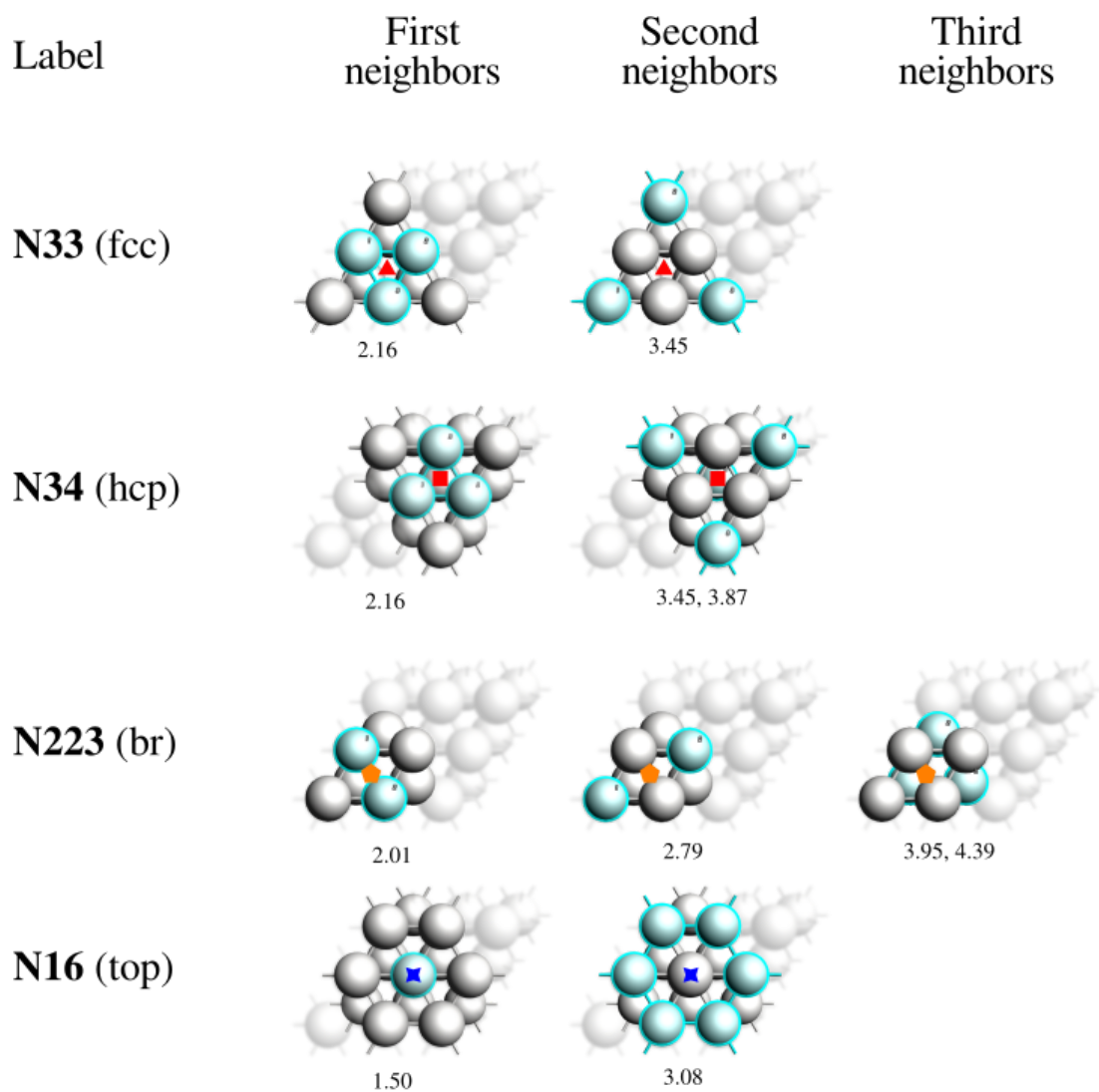


Fig. 4.2: Binding sites obtained for a CO molecule adsorbed on a Pt(111) surface using PESExploration%BindingSites%DistanceDifference = 0.5 Å.

(continued from previous page)

				2.76 – 3.26
				3.86 – 4.36
N16	3.60	3.60	3.88	1.50 – 2.00
				3.04 – 3.54

Input options

There are two distinct ways of triggering the computation of binding sites. You can either:

compute the binding sites at the end of a *Process Search* (page 213), *Basin Hopping* (page 219) or *Landscape Refinement* (page 224) calculation by setting the `PESExploration%BindingSites%Calculate` option to `yes`:

```
PESExploration
  Job [ProcessSearch | BasinHopping | LandscapeRefinement]

  BindingSites
    Calculate Yes
  End

  StatesAlignment
    ReferenceRegion reference_region_name
  End
End
```

or

compute the binding sites by setting the `PESExploration%Job` to `BindingSites` and load a previously computed Energy Landscape (see *Example: PES Exploration, Binding Sites for O on Pt 111* (page 505)):

```
PESExploration
  Job BindingSites

  LoadEnergyLandscape
    Path path/to/previous/calculation/ams.results
  End

  StatesAlignment
    ReferenceRegion reference_region_name
  End
End
```

The following input options are related to the calculation of binding sites:

```
PESExploration
  BindingSites
    Calculate Yes/No
    DistanceDifference float
    MaxCoordinationShellsForLabels integer
    NeighborCutoff float
    ReferenceRegion string
  End
End
```

PESExploration

BindingSites

Type

Block

Description

Options related to the calculation of binding sites.

Calculate**Type**

Bool

Default value

No

Description

Calculate binding sites at the end of a job. Not needed for Binding Sites job.

DistanceDifference**Type**

Float

Default value

-1.0

Unit

Angstrom

Description

If the distance between two mapped binding-sites is larger than this threshold, the binding-sites are considered different. If not specified, its value will set equal to [PESExploration%StructureComparison%DistanceDifference]

MaxCoordinationShellsForLabels**Type**

Integer

Default value

3

Description

The binding site labels are given based on the coordination numbers of shells in the reference region, using the following format: N<int><int>..., e.g., the label 'N334' means 3 atoms in the first coordination shell, 3 in the second one, and 4 in the third one. This parameter controls the maximum number of shells to include.

NeighborCutoff**Type**

Float

Default value

-1.0

Unit

Angstrom

Description

Atoms within this distance of each other are considered neighbors for the calculation of the binding sites. If not specified, its value will set equal to [PESExploration%StructureComparison%NeighborCutoff]

ReferenceRegion

Type

String

Default value**Description**

Defines the region that is considered as the reference for binding sites detection. Binding sites are projected on this region using the geometry from the reference system. If not specified, its value will set equal to [PESExploration%StatesAlignment%ReferenceRegion]

The following input options are related to the definition of the reference region and alignment thereof:

```
PESExploration
  StatesAlignment
    DistanceDifference float
    ReferenceRegion string
  End
End
```

PESExploration**StatesAlignment****Type**

Block

Description

Configures details of how the energy landscape configurations are aligned respect to the main chemical system [System].

DistanceDifference**Type**

Float

Default value

-1.0

Unit

Angstrom

Description

If the distance between two mapped atoms is larger than this threshold, the configuration is considered not aligned. If not specified, its value will set equal to [PESExploration%StructureComparison%DistanceDifference]

ReferenceRegion**Type**

String

Default value**Description**

Defines the region that is considered as the reference for alignments. Atoms outside this region are ignored in the alignments.

4.12.15 References

GRADIENTS, HESSIAN, STRESS TENSOR, ELASTICITY

No matter what application the AMS driver is used for, in one way or another it always explores the potential energy surface (PES) of the system. One can furthermore ask AMS to calculate additional properties of the PES in the points that are visited. These are mostly (but not exclusively) derivatives of the energy, e.g. we can ask AMS to calculate the gradients or the Hessian in the visited points. In general all these PES point properties are requested through the `Properties` block in the AMS input:

```
Properties
  Gradients Yes/No
  StressTensor Yes/No
  Hessian Yes/No
  SelectedRegionForHessian string
  PESPointCharacter Yes/No
  ElasticTensor Yes/No
End
```

This properties described on this page in the AMS manual are all related to derivatives of the energy.

Note that because these properties are tied to a particular point on the potential energy surface, they are found on the *engine output files* (page 19). Note also that the properties are not always calculated in every PES point that the AMS driver visits during a calculation. By default they are only calculated in *special* PES points, where the definition of special depends on the *task* (page 47) AMS is performing: For a *geometry optimization* (page 49) properties would for example only be calculated at the final, converged geometry. This behavior can often be modified by keywords special to the particular running task.

5.1 Nuclear gradients

The first derivative with respect to the nuclear coordinates can be requested as follows:

```
Properties
  Gradients Yes/No
End
```

Properties

Gradients

Type

Bool

Default value

No

GUI name

Nuclear gradients

Description

Calculate the nuclear gradients.

Note that these are gradients, not forces, the difference being the sign. The gradients are printed in the output and written to the *engine result file* (page 19) belonging to the particular point on the PES in the `AMSResults%Gradients` variable as a $3 \times n_{\text{atoms}}$ array in atomic units (Hartree/Bohr).

5.2 Hessian

The calculation of the second derivative of the total energy with respect to the nuclear coordinates is enabled by:

```
Properties
  Hessian Yes/No
  SelectedRegionForHessian string
End
```

Properties**Hessian****Type**

Bool

Default value

No

Description

Whether or not to calculate the Hessian.

SelectedRegionForHessian**Type**

String

GUI name

Hessian only for

Description

Compute the Hessian matrix elements only for the atoms in a particular region. If not specified, the Hessian will be computed for all atoms.

The Hessian is not printed to the text output, but is saved in the engine result file as variable `AMSResults%Hessian`. Note that this is just the plain partial second derivatives (no mass-weighting) of the total energy. The $3 \times N_{\text{atoms}}$ columns/rows of the matrix are grouped by atom: the first three rows/columns correspond to the first atom, etc..

Note that the AMS driver also supports the *Mobile Block Hessian* (page 251) (MBH) method, which allows treating parts of the system as rigid blocks.

Often one is not interested in the Hessian matrix itself, but just in using it for the calculation of IR frequencies or to characterize a PES point (as e.g. a local minimum or a saddle point). For these application, see the following pages in the manual:

- *IR frequencies* (page 247)
- *PES point character* (page 239)

5.3 PES point character

A PES point can according to the slope and curvature of the PES at that point be classified in the following categories:

- A local minimum on the PES with vanishing nuclear gradients and no negative frequencies.
- A transition state with vanishing nuclear gradients and exactly one negative frequency, i.e. a first order saddle point on the PES.
- A higher order saddle point, i.e. a stationary point on the PES with vanishing nuclear gradients but more than one imaginary frequency.
- A non-stationary point on the PES. Here the gradients are non-zero.

This classification can easily be done if both the gradients and the normal modes have already been calculated. However, calculating the full Hessian needed for the entire set of normal modes is very expensive and undesirable if one only wants to know the character of a PES point. The AMS driver can quickly, and without calculating the full Hessian, characterize a PES point into one of the above categories. This can be used to confirm the success of e.g. a *transition state search* (page 72) or *geometry optimization* (page 49). A PES point can be characterized by requesting `PESPointCharacter` as a property:

```
Properties
  PESPointCharacter Yes/No
End
```

Properties

PESPointCharacter

Type

Bool

Default value

No

GUI name

Characterize PES point

Description

Determine whether the sampled PES point is a minimum or saddle point. Note that for large systems this does not entail the calculation of the full Hessian and can therefore be used to quickly confirm the success of a geometry optimization or transition state search.

This will calculate the few lowest normal modes using an iterative diagonalization of the Hessian¹ based on a Davidson algorithm implemented in the PRIMME library². The procedure has been optimized for finding a small number of low-lying eigenvalues in as few matrix-vector multiplications (and thus single point calculations) as possible. This is facilitated by performing the iterative method using a pre-conditioning matrix based on an approximation of the Hessian. The approximate Hessian is obtained from the full Hessian at a lower level of theory. This calculation also provides the initial guesses for the desired normal modes. What the lower level of theory is depends on the main engine used in the calculation: DFTB with the GFN1-xTB model is used as the lower level of theory for relatively slow engines, e.g. DFT based engines. For semi-empirical engines like DFTB or MOPAC, the lower level of theory is currently UFF. It is currently not possible to change the engine used to obtain the preconditioning Hessian and the approximate modes.

- Note that the iterative calculation of the normal modes is skipped when ...

¹ P. Deglmann and F. Furche, *Efficient characterization of stationary points on potential energy surfaces*, J. Chem. Phys. 117, 9535 (2002) (<https://doi.org/10.1063/1.1523393>)

² A. Stathopoulos and J. R. McCombs, *PRIMME: Preconditioned Iterative MultiMethod Eigensolver: Methods and software description*, ACM Transactions on Mathematical Software, Vol. 37, No. 2, (2010), 21:1–21:30. (<https://doi.org/10.1145/1731022.1731031>)

1. ... the nuclear gradients are so large that the PES point is considered non-stationary. The calculation of the modes is then just not necessary for classifying it.
 2. ... the full normal modes or Hessian have also been requested. The iterative calculation is then not necessary, as all modes are already known.
 3. ... the molecule is very small. (For small systems the iterative calculation of the few lowest normal modes is not faster than the full calculation of all modes, so all modes are calculated instead.)
- The classification as a stationary or non-stationary point uses the gradient convergence criterion from the geometry optimizer as the tolerance, see [geometry optimization](#) (page 49). This makes sure that the criterion for what is considered converged/stationary is always in sync between the optimizer and the PES point characterization.
 - For periodic systems the PES point characterization does not take the lattice degrees of freedom into account. A PES point where the nuclear gradients are small enough would for example be classified as a stationary point, even if the system is under stress.

Details of the iterative procedure can be configured in the `PESPointCharacter` block:

```
PESPointCharacter
  Displacement float
  NegativeEigenvalueTolerance float
  NumberOfModes integer
  Tolerance float
End
```

PESPointCharacter

Type

Block

Description

Options for the characterization of PES points.

Displacement

Type

Float

Default value

0.04

Description

Controls the size of the displacements used for numerical differentiation: The displaced geometries are calculated by taking the original coordinates and adding the mass-weighted mode times the reduced mass of the mode times the value of this keyword.

NegativeEigenvalueTolerance

Type

Float

Default value

-0.0001

Unit

Hartree/Bohr²

Description

The threshold in Hessian eigenvalue below which a mode is considered imaginary, i.e. indicating a transition state. This is a small negative number, as very small negative eigenvalues may be due to numerical noise on an essentially flat PES and do not indicate true transition states.

NumberOfModes**Type**

Integer

Default value

2

Description

The number of (lowest) eigenvalues that should be checked.

Tolerance**Type**

Float

Default value

0.016

Description

Convergence tolerance for residual in iterative Davidson diagonalization.

- Note that the residual tolerance that can be achieved is limited by the numerical differentiation that is performed. The default values should apply in most cases, but if convergence becomes a problem one may choose to increase the tolerance or to increase the step size (slightly). Note that the default residual tolerance is lower than for the other mode selective methods. This is because PRIMME uses a different convergence criteria than mode tracking/refinement. The higher value used as a default will therefore not result in decreased levels of accuracy. The method will bail if the number of iterations exceeds the number of normal modes as at this point still achieving convergence becomes unlikely, in part due to the next point.
- In order to avoid producing the known and irrelevant rigid modes, the method searches for normal modes orthogonal to six (or five) rigid modes. Imperfections due to the numerical differentiation may mean that the translational and rotational rigid modes are not exact eigenmodes of the Hessian that is constructed. As a result, some part of the lowest vibrational normal mode may lie in the span of the theoretical rigid modes and therefore be inaccessible to the Davidson method. This places a lower bound on the residual tolerance that can be achieved, which is directly related to the numerical differentiation accuracy. The take-away: do not set the tolerance too low, the default usually suffices.
- Behind the scenes, the method actually computes a few more modes than requested. In the case of multiplicities, eigenvalues may still converge out of order. These additional eigenvalues essentially guarantee that the obtained modes are indeed the lowest ones.

5.4 Thermodynamics, gas phase Gibbs free energy

At the end of a completed IR Frequencies (normal modes) calculation, a survey is given of thermodynamic properties: entropy, internal energy, constant volume heat capacity, enthalpy and Gibbs free energy, see:

- *IR frequencies* (page 247)
 - *Thermodynamics* (page 269)
 - *Gibbs free energy change for a gas phase reaction* (page 272)

5.5 Stress tensor

For periodic systems (chains, slabs, bulk) one can also request the clamped-ion stress tensor (note: the clamped-ion stress is only part of the *true* stress tensor):

```
Properties
  StressTensor Yes/No
End
```

Properties

StressTensor

Type

Bool

Default value

No

GUI name

Stress tensor

Description

Calculate the stress tensor.

The clamped-ion stress tensor σ_α (Voigt notation) is computed via numerical differentiation of the energy E WRT a strain deformations ϵ_α keeping the atomic fractional coordinates constant:

$$\sigma_\alpha = \frac{1}{V_0} \left. \frac{\partial E}{\partial \epsilon_\alpha} \right|_{\text{constant atomic fractional coordinates}}$$

where V_0 is the volume of the unit cell (for 2D periodic system V_0 is the area of the unit cell, and for 1D periodic system V_0 is the length of the unit cell).

The clamped-ion stress tensor (in Cartesian notation) is written to the engine result file in `AMSResults%StressTensor`.

5.6 Elastic tensor

The elastic tensor $c_{\alpha,\beta}$ (Voigt notation) is computed via second order numerical differentiation of the energy E WRT strain deformations ϵ_α and ϵ_β :

$$c_{\alpha,\beta} = \frac{1}{V_0} \frac{\partial^2 E}{\partial \epsilon_\alpha \partial \epsilon_\beta}$$

where V_0 is the volume of the unit cell (for 2D periodic system V_0 is the area of the unit cell, and for 1D periodic system V_0 is the length of the unit cell).

For each strain deformation $\epsilon_\alpha \epsilon_\beta$, the atomic positions will be optimized. The elastic tensor can be computed for any periodicity, i.e. 1D, 2D and 3D. Computing the elastic tensor will also calculate the bulk modulus, Young's modulus, and shear modulus, as well as the Poisson ratio.

See also:

Example: Elastic tensor (page 557)

To compute the elastic tensor, request it in the `Properties` input block of AMS:

```

Properties
  ElasticTensor Yes/No
End

```

Note: The elastic tensor should be computed at the fully optimized geometry. One should therefore perform a geometry optimization of all degrees of freedom, **including the lattice vectors**. It is recommended to use a tight gradient convergence threshold for the geometry optimization (e.g. Quality “Good”). Note that all this can be done in one job by combining the *geometry optimization task* (page 49) with the elastic tensor calculation.

The elastic tensor (in Voigt notation) is printed to the output file and stored in the *engine result file* (page 19) in the AMSResults section (for 3D system, the elastic tensor in Voigt notation is a 6x6 matrix; for 2D systems is a 3x3 matrix; for 1D systems is just one number).

Options for the numerical differentiation procedure can be specified in the ElasticTensor input block:

```

ElasticTensor
  ConvergenceQuality [Normal | Good | VeryGood]
  StrainStepSize float
End

```

ElasticTensor

Type
Block

Description
Options for numerical evaluation of the elastic tensor.

ConvergenceQuality

Type
Multiple Choice

Default value
Good

Options
[Normal, Good, VeryGood]

GUI name
Convergence

Description
The tightness of the convergence of the geometry optimizations for each strain deformation. This should not be set higher than the overall convergence quality of the preceding geometry optimization configured by the GeometryOptimization%Convergence%Quality keyword.

StrainStepSize

Type
Float

Default value
0.001

Description
Step size (relative) of strain deformations used for computing the elastic tensor numerically.

Pressure (page 329) or *non-isotropic external stress* (page 330) can be included in your simulation via the corresponding *engine addons* (page 325).

The elastic tensor calculation supports AMS' *double parallelization* (page 20), which can perform the calculations for the individual displacements in parallel. This is configured automatically, but can be further tweaked using the keys in the `NumericalDifferentiation%Parallel` block:

```
ElasticTensor
  Parallel
    nCoresPerGroup integer
    nGroups integer
    nNodesPerGroup integer
  End
End
```

ElasticTensor

Parallel

Type

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup

Type

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups

Type

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup

Type

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

5.7 Numerical differentiation options

The following options apply whenever AMS computes gradients, Hessians or stress tensors via numerical differentiation.

```
NumericalDifferentiation
  NuclearStepSize float
  StrainStepSize float
End
```

NumericalDifferentiation

Type

Block

Description

Define options for numerical differentiations, that is the numerical calculation of gradients, Hessian and the stress tensor for periodic systems.

NuclearStepSize

Type

Float

Default value

0.005

Unit

Bohr

Description

Step size for numerical nuclear gradient calculation.

StrainStepSize

Type

Float

Default value

0.001

Description

Step size (relative) for numerical stress tensor calculation.

AMS may use symmetry (key `NumericalDifferentiation%UseSymmetry`) in case of numerical differentiation calculations. If symmetry is used only symmetry unique atoms are displaced. Symmetry is only recognized if the starting geometry has symmetry. Symmetry is only used for molecules if the molecule has a specific orientation in space, like that the z-axis is the main rotation axis. If the GUI is used one can click the Symmetrize button (the star), such that the GUI will (try to) symmetrize and reorient the molecule. There are some cases that even after such symmetrization, the orientation of the molecule is not what is needed for the symmetry to be used in case of numerical differentiation calculations. If that is the case or if key `NumericalDifferentiation%UseSymmetry` is set to 'False', then no symmetry will be used.

The numerical differentiation calculation supports AMS' *double parallelization* (page 20), which can perform the calculations for the individual displacements in parallel. This is configured automatically, but can be further tweaked using the keys in the `NumericalDifferentiation%Parallel` block:

```
NumericalDifferentiation
  Parallel
    nCoresPerGroup integer
    nGroups integer
```

(continues on next page)

(continued from previous page)

```
nNodesPerGroup integer
End
End
```

NumericalDifferentiation**Parallel****Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

VIBRATIONAL SPECTROSCOPY

6.1 General

The starting point is the Hessian of the system, being the second derivative of the energy with respect to the atomic coordinates.

The eigenvalues of the Hessian are the frequencies and the eigen vectors are the normal modes.

As the calculation of the full Hessian is very expensive there are several ways to avoid it, so that you only get a part of the full spectrum, or only modes for a region of the system, see *IR frequencies and normal modes* (page 247) section.

A full, partial, or approximate Hessian in itself can be useful for a (Hessian-based) geometry optimization or a transition state search.

Vibrational spectra are obtained by differentiating a property along the normal modes at a (local) minimum of the PES. So for spectra you need to optimize the geometry first, otherwise you get negative frequencies.

The Normal modes and or vibrational spectra can be requested via the `Properties` block

```
Properties
  NormalModes Yes/No
  Raman Yes/No
  VROA Yes/No
  VCD Yes/No
  Phonons Yes/No
End
```

When requesting the normal modes, the IR intensities are calculated, as they are very cheap.

See also:

[Tutorials on Vibrational Spectroscopy](#)

6.1.1 Where are the results?

Because the results of a vibrational spectroscopy calculation are tied to a particular point on the potential energy surface, they are found on the *engine output files* (page 19). Note also that the properties are not always calculated in every PES point that the AMS driver visits during a calculation. By default they are only calculated in *special* PES points, where the definition of special depends on the *task* (page 47) AMS is performing: For a *geometry optimization* (page 49) properties would for example only be calculated at the final, converged geometry. This behavior can often be modified by keywords special to the particular running task.

6.2 IR frequencies and normal modes

6.2.1 All vibrational Modes

The calculation of the normal modes of vibration can be requested with:

```
Properties
  NormalModes Yes/No
End
```

Typically used icw with *Task SinglePoint* (page 47), *Task GeometryOptimization* (page 49), or *Task TransitionStateSearch* (page 72). In case of geometry optimization or transition state search the normal modes will only be calculated at the final, converged geometry.

Properties

NormalModes

Type

Bool

Default value

No

GUI name

Frequencies

Description

Calculate the frequencies and normal modes of vibration, as well as the corresponding IR intensities, if the engine supports these calculations natively or can calculate dipole moments.

The molecular normal modes are normally calculated within the harmonic oscillation model. If the molecule is in its equilibrium conformation, it sits in the lowest point (at least locally) on the PES. The cross-section of the PES profile close to this point can then be assumed to be approximately parabolic, such that the second derivative of the energy w.r.t a nuclear coordinate can be interpreted as a force constant for the harmonic oscillation of an atom along this coordinate. Since molecular vibrations in polyatomics involve the simultaneous displacement of multiple atoms, this harmonic oscillator model can be generalized to multiple nuclear coordinates. The normal modes and their frequencies then become eigenvectors and eigenvalues of a force constant matrix, the Hessian:

$$H_{ij} = \frac{\partial^2 E}{\partial R_i \partial R_j}$$

The (non-mass-weighted) Hessian is saved in the engine result file as variable `AMSResults%Hessian`. It is not printed to the text output. The column/row indices are ordered as: x-component of atom 1, y-component of atom 1, z-component of atom 1, x-component of atom 2, etc.

Most *engines* (page 315) cannot calculate the Hessian analytically. Only the ADF engine can calculate the Hessian analytically, however, only for a limited number of XC functionals, see the [ADF manual](#). In case the Hessian is not calculated analytically the Hessian is constructed column-wise through numerical differentiation of the energy gradients w.r.t. each nuclear coordinate. AMS will set up 2 single-point calculations (1 for the positive displacement, 1 for the negative displacement), and the requested engine will return the energy gradients at these displacements. These gradients are calculated analytically for most engines.

Note: Numerical calculation of the full Hessian requires 6N single points calculation, which can take a considerable amount of time for large systems. A mode selective method can be a fast alternative, see [mode scanning](#) (page 254), [mode refinement](#) (page 255), and [mode tracking](#) (page 257).

When requesting the normal modes calculation, integrated IR intensities are simultaneously calculated during the finite differentiation steps when constructing the Hessian (as long as dipole moments are supported by the engine). These IR intensities are calculated from the numerical dipole gradients:

$$I_{IR} = \frac{N\pi}{3c^2} \sum_{\alpha} \left(\sum_j \frac{\partial \mu_{\alpha}}{\partial R_j^m} Q_j^m \right)^2$$

Where α denotes the x-,y- and z-components of the dipole moment μ , and Q^m is the mass-weighted vibrational normal mode.

The resulting IR spectrum can be visualized by opening the engine result file with AMSspectra. The normal modes of vibration and the IR intensities are saved to the [engine result file](#) (page 19) in the Vibrations section.

Note: The calculation of the normal modes of vibration needs to be done the system's equilibrium geometry. So one should either run the normal modes calculation using an already optimized geometry, or combine both steps into one job by using the [geometry optimization task](#) (page 49) together with the `Properties%NormalModes` keyword.

Symmetry labels of the normal modes may be calculated if AMS uses symmetry in the calculation (key `UseSymmetry`). If symmetry is used, the normal modes are projected against symmetric displacements for each irrep. If that is not successful the symmetry label is 'MIX'. Symmetry is only recognized if the geometry is (almost) perfectly symmetric and has a specific orientation in space. You can use the `Symmetrize` key in the [System block](#) (page 31) to symmetrize and reorient the molecule. If the AMSinput GUI module is used, one can click the Symmetrize button (the star) and the GUI will try to symmetrize and reorient the molecule. Unless one is an expert user, best is not to specify an explicit symmetry in any part of the input.

Rescanning Imaginary modes

The `ReScanModes` keyword can be used to calculate more accurately frequencies of specific modes after a normal modes calculation. It is identical to the `ScanFreq` option that was available for older versions of ADF and BAND. Primarily used to identify spurious imaginary modes, and is on by default for this purpose. See also the [Mode Scanning](#) (page 254) task, which is an extension of this method, but which is not on by default. One should not specify an explicit symmetry in any part of the input, otherwise the rescanning of the normal modes might fail.

```
NormalModes
  ReScanModes Yes/No
  ReScanFreqRange float_list
End
```

NormalModes

ReScanModes

Type

Bool

Default value

Yes

GUI name

Re-scan modes

Description

Whether or not to scan imaginary modes after normal modes calculation has concluded.

ReScanFreqRange

Type

Float List

Default value

[-10000000.0, 10.0]

Unit

cm-1

Recurring

True

GUI name

Re-scan range

Description

Specifies a frequency range within which all modes will be scanned. 2 numbers: an upper and a lower bound.

6.2.2 Symmetric Displacements

```
NormalModes
  Displacements Symmetric
End
```

Specify `Displacements Symmetric` to calculate the energy Hessian using finite differences in symmetry-adapted displacements, and the corresponding normal modes.

```
NormalModes
  SymmetricDisplacements
    Type [All | Infrared | Raman | InfraredAndRaman]
  End
End
```

If `Type Infrared` or `Type Raman` is specified then only irreps that result in non-zero intensities for the corresponding spectroscopy will be included in the calculation. Using this feature may save a lot of time for large symmetric molecules by skipping calculation of normal modes that would not contribute to the spectrum anyway. If `Type InfraredAndRaman` is specified then vibrations that have a non-zero IR or Raman intensity will be calculated. If `Type All` is specified then all vibrations will be calculated. For multi-dimensional irreps (such as E and T) only the first component will be computed. For any component beyond the first, the frequencies and intensities will be copied from the first one.

NormalModes**SymmetricDisplacements****Type**

Block

Description

Configures details of the calculation of the frequencies and normal modes of vibration in symmetric displacements.

Type**Type**

Multiple Choice

Default value

All

Options

[All, Infrared, Raman, InfraredAndRaman]

GUI name

Symm Frequencies

Description

For symmetric molecules it is possible to choose only the modes that have non-zero IR or Raman intensity (or either of them) by symmetry.

In order to calculate the Raman intensities the Raman property must be requested.

Warning: Specifying `Type Raman` alone does not trigger calculation of the Raman intensities. In order to calculate the Raman spectrum one should also specify `Raman True`.

Note: `Displacements Symmetric` will also produce a $3N$ -by- $3N$ Hessian matrix but if the `Type` key's argument is not `All` then this matrix will likely have many zero eigenvalues.

6.2.3 Mobile Block Hessian (MBH)

```
NormalModes
  Displacements Block
End
```

Specify `Displacements Block` for the Block Normal Modes option (also known as **Mobile Block Hessian**, or **MBH**¹²). MBH is useful when calculating vibrational frequencies of a small part of a very large system (molecule or cluster). Calculation of the full spectrum of such a system may be inefficient and is unnecessary if one is interested in one particular part. Besides, it may be difficult to extract normal modes related to the interesting sub-system out of the whole spectrum. Using Block Normal Modes it is possible to treat parts of the system as rigid blocks. Each block will usually have only six frequencies related to its rigid motions compared to $3 \cdot N$ for when each atom of the block is treated separately.

MBH is suitable to calculate frequencies in partially optimized structures. Assume a geometry optimization is performed with the `Block` key in the Constraints input block [see constrained geometry optimizations]. During the geometry optimization, the shape of the block is not changed. The internal geometry of the block is kept fixed, but the block as a whole can still translate or rotate.

At the end of such a partial geometry optimization, the position and orientation of the block is optimized, thus the total force on the block is zero. However, there might be still some residual forces within a block, since those degrees of freedom were not optimized. A traditional frequency calculation performed on this partially optimized structure might result in non-physical imaginary frequencies without a clear interpretation. Therefore one should use an adapted formulation of normal mode analysis: the Mobile Block Hessian method. MBH does not consider the internal degrees of freedom of the block (on which residual forces) apply, but instead uses the position/orientation of the block as coordinates. In the resulting normal mode eigenvectors, all atoms within the same block move collectively.

Of course, MBH can also be applied on a fully optimized structure.

Accuracy

¹ A. Ghysels, D. Van Neck, V. Van Speybroeck, T. Verstraelen and M. Waroquier, *Vibrational Modes in partially optimized molecular systems*, *Journal of Chemical Physics* 126, 224102 (2007) (<https://doi.org/10.1063/1.2737444>)

² A. Ghysels, D. Van Neck and M. Waroquier, *Cartesian formulation of the Mobile Block Hessian Approach to vibrational analysis in partially optimized systems*, *Journal of Chemical Physics* 127, 164108 (2007) (<https://doi.org/10.1063/1.2789429>)

```

NormalModes
  BlockDisplacements
    AngularDisplacement float
    BlockAtoms integer_list
    BlockRegion string
    Parallel
      nCoresPerGroup integer
      nGroups integer
      nNodesPerGroup integer
    End
    RadialDisplacement float
  End
End

```

NormalModes**BlockDisplacements****Type**

Block

Description

Configures details of a Block Normal Modes (a.k.a. Mobile Block Hessian, or MBH) calculation.

AngularDisplacement**Type**

Float

Default value

0.5

Unit

Degree

Description

Relative step size for rotational degrees of freedom during Block Normal Modes finite difference calculations. It will be scaled with the characteristic block size.

BlockAtoms**Type**

Integer List

Recurring

True

Description

List of atoms belonging to a block. You can have multiple BlockAtoms.

BlockRegion**Type**

String

Recurring

True

Description

The region to to be considered a block. You can have multiple BlockRegions, also in combination with BlockAtoms.

Parallel**Type**

Block

Description

Configuration for how the individual displacements are calculated in parallel.

nCoresPerGroup**Type**

Integer

Description

Number of cores in each working group.

nGroups**Type**

Integer

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Cores per task

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

RadialDisplacement**Type**

Float

Default value

0.005

Unit

Angstrom

Description

Step size for translational degrees of freedom during Block Normal Modes finite difference calculations.

The second derivatives of the energy with respect to Cartesian displacements of the free atoms and those with respect to block motions (3 translation plus 3 rotations) are calculated by numerical differentiation of the gradient. The accuracy of the second derivatives is determined by the accuracy of the gradient evaluation and the step size in the numerical differentiation. The `RadialDisplacement` and `AngularDisplacement` parameters can be specified to set the step size for Cartesian displacements (translations) and block rotations respectively. The step size for angles is automatically scaled with the block size.

Note: Blocks should consist of at least 3 atoms (i.e. block of 1 or 2 atoms are not supported).

6.2.4 Mode Scanning

Mode Scanning can be used to obtain more accurate approximations for properties obtained by numerical differentiation along the vibrational normal modes (frequencies, intensities, Raman, etc.), without changing the modes themselves. Mode Scanning is an extension of the frequency scanning options (`ScanFreq`) that were part of ADF and BAND in earlier versions of the Amsterdam Modeling Suite. These latter options are still available as the `ReScanModes` keyword in the `NormalModes` block, if these are requested during a calculation.

- Primarily used to identify spurious imaginary modes.
- Improve numerical accuracy of normal mode properties.
- Rescanning modes using a different level of theory.

Theory

Vibrational normal modes are usually obtained as eigenvectors of the Hessian matrix. A common problem with this scheme however, is that due to numerical errors in constructing this Hessian, low-frequency vibrations may be reported to have imaginary frequencies instead. The Mode Scanning task allows for re-calculation of the frequency of these modes. The Mode Scanning task does not change the normal modes itself, only its properties. This Mode Scanning task allows you to confirm whether reported imaginary frequencies are attributed to transition states or whether they are simply due to numerical errors.

Given a user-supplied mode Q , the frequency is calculated from the force constant:

$$k = \frac{\partial^2 E}{\partial^2 Q}$$

$$\nu = \frac{1}{2\pi c} \sqrt{\frac{k}{\mu_r}}$$

This is again done by numerical differentiation of the energy gradients, requiring AMS to set up 2 single point calculations per selected normal mode. Integrated IR intensities are also calculated simultaneously (if dipole moments are supported by the *engine* (page 315)):

$$I_{IR} = \frac{N\pi}{3c^2} \sum_{\alpha} \left(\frac{\partial \mu_{\alpha}}{\partial Q^m} \right)^2$$

Where the derivative is with respect to the mass-weighted normal mode.

It is also possible to use this method to selectively re-calculate the normal mode properties for different engine settings. This has two distinct uses:

- If the modes were originally generated using a finite difference method, a different stepsize can be used. For strong vibrations (high frequencies), large stepsizes may cause inaccuracies due to increasing anharmonic contributions. For weak vibrations (low frequencies) on the other hand, stepsizes can often be too small. The displacements associated with these vibrations are small, which can give incorrect sampling of the PES profile. This should be compensated for by choosing a larger stepsize. The stepsize can be set using the `Displacement` key.
- Users can also recalculate modes using higher levels of theory. Modes generated from a full frequency analysis using e.g. DFTB can be recalculated using e.g. LDA DFT to obtain more realistic integrated IR intensities. The method used for the single point calculations can be set in the *Engine block* (page 315).

Input

A numerical frequency calculation is performed by requesting the `VibrationalAnalysis` task with `Type ModeScanning`:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeScanning
  Displacement 0.001
  NormalModes
    ModeFile adf.rkf
    # select all modes with imaginary frequencies
  ModeSelect
    ImFreq true
  End
End
End
```

The Mode Scanning tasks uses only the `NormalModes` block for its input handling. Here, `ModeFile` specifies the AMS output file containing the normal modes for which you want to calculate the frequencies. The `ModeSelect` block is used to specify which of the modes in this file should be recalculated, since we are often only interested in a select few of them. A more detailed overview of this block is given in the section *Selecting Modes* on the [main page](#) (page 266). Finally, `Displacement` can be used to specify the stepsize (in Bohr) for the finite differences. The stepsize is provided for displacements along the Cartesian normal modes.

The Mode Scanning module is the main driving force for the *Mode Tracking* (page 257) and *Vibrational Mode Refinement* (page 255) tasks, which provide more advanced options for refining not only the properties of the modes, but also the modes themselves. Consult the relevant pages for more information. Alternatively, a simplified version of Mode Scanning is available which follows the old implementation in ADF and BAND (as the `ScanFreq` option). This version can be enabled when doing a full frequency analysis by enabling the `Properties%NormalModes` keyword. See the *Full Analysis* (page 248) page for further details.

6.2.5 Mode Refinement

With this option you can improve the normal modes, by importing previously calculated modes and then applying a more accurate engine, or more accurate settings, typically for only part of the spectrum. The vibrational Mode Refinement method not only refines frequencies from a previous calculation, but also tries to correct the vibrational modes themselves.

- Refinement of spectral regions requires a sufficient number of modes in the basis to be accurate.
- 1-step refinement. No iterative improvement possible. (Unless followed by a separate Mode Tracking calculation.)
- Quality of the results depends on accuracy of the selected guess modes.

If we start from e.g. a semi-empirical method such as in MOPAC, we can get approximations for the vibrational modes. Mode Refinement then re-calculates part of the Hessian for a subset of these modes using a more accurate method such as GGA DFT, and updates the normal modes themselves to fit this more accurate method. It is intended to circumvent the expensive calculation of the Hessian if you are only interested in a (small) part of the full spectrum. This is based on the method in reference³.

Because the Mode Refinement method uses linear combinations of the guess modes, its accuracy depends on the set of modes that is supplied.

- If we want to e.g. obtain a mode which includes a C=O stretch, then the initial set must contain a mode which has this C=O stretch, otherwise this cannot be included in the refined modes.

³ T.Q. Teodoro, M.A.J. Koenis, S.E. Galembeck, V.P. Nicu, W.J. Buma, L. Visscher, *A frequency range selection method for vibrational spectra*, J. Phys. Chem. Lett., 9 (23), 6878 (2018) (<https://doi.org/10.1021/acs.jpclett.8b02963>)

- If we are refining a region containing many similar modes, e.g. vibrations of aromatic ring backbones, and we only use part of this spectral region for the initial set, the set of refined modes will “drift” towards the centre of the spectral region as a result of mode-mixing. This is again an artefact of missing character in the modes.
- This mode-mixing may result in reduced accuracy for some of the modes, as this procedure minimizes the total error for all of the modes. Instead of having a couple of modes with large errors, mode-mixing tends to spread out the error across multiple normal modes. Adding 1 “bad” mode to the basis can then negatively affect your results.
- The advantage of Mode Refinement over Mode Tracking is the ability to refine entire spectral regions at once. If we have a good basis, Mode Refinement can be less computationally expensive than Mode Tracking. If you want to refine larger sections of the spectrum, Mode Refinement is therefore recommended. If you only want to calculate a select few modes, use Mode Tracking to avoid basis dependence and to assure accuracy of the obtained modes.
- For characteristic peaks, Mode Tracking shows very good convergence, and will thus be cheaper to use than Mode Refinement. For (semi-)degenerate modes however, Mode Refinement works better due to the poor tracking performance for these modes.

See also:

The [GUI tutorial](#) on Mode Refinement.

Theory

We are going to start from a set of normal modes b , obtained from e.g. a semi-empirical or force-field method. First, this task runs the *numerical frequency* (page 254) calculation for all selected normal modes, but this time using an ab initio method such as DFT. During the finite difference steps, we also calculate the projection of the Hessian onto the normal modes:

$$\sigma_i = H^m \cdot b_i^m = \frac{\partial^2 E}{\partial R_i^m \partial b^m}$$

This term is calculated through finite differences on the analytical gradients of the electronic energy along the mass-weighted normal modes b^m . The index i denotes the $3N$ nuclear coordinates. These projections are then used to construct a Rayleigh matrix:

$$\tilde{H}^m = B^{mT} \cdot H^m \cdot B^m = B^{mT} \cdot \Sigma$$

Here, B^m and Σ are matrices containing the b^m and σ vectors. The eigenvectors of \tilde{H}^m give us the coefficient series for linear combinations of the normal modes b^m such that we obtain a new set of modes q :

$$q^m = \sum_k c_k \cdot b_k^m$$

These modes q are the closest approximation to the DFT-modes that we could obtain from a linear combination of the approximate modes b . In other words: the approximate modes b are used as a basis for finding the modes from a more sophisticated theory.

Input

This method inherently features a trade-off:

- The computational benefit comes from only performing the finite difference calculations for the selected modes. By only selecting a small set of modes that we are interested in, we minimize computational expense.
- The more modes we select, the larger the basis for constructing the refined modes. Running for a larger number of modes yields better results. (In the extreme case, running for all $3N$ modes equates to constructing the full Hessian.)

In practice, Mode Refinement requires you to select a reasonable portion of the spectrum to get accurate results. Specifically, you should select all modes in a region of the spectrum which look similar. Ring structures for instance often feature broad frequency regions with many ring distortions. Even if you are only interested in a couple of these, you should still select all modes in this region, to assure sufficient basis size. Vibrational modes involving ring substituents can however be omitted, which is where we save computation time.

If you are interested only in IR-active vibrations, you could further minimize the basis by only selecting the approximate modes which are IR-active (since adding the non-active modes to the linear expansion does not affect the IR-intensity of the refined modes). Do note that if the semi-empirical method used for calculating the approximate modes yields poor approximations for the dipole gradients, it may be safer to include also modes with very low IR intensity. This is because their low IR-activity may have only been due to the low accuracy of the approximate method.

See also:

A [tutorial](#) showing this basis representability.

A Mode Refinement calculation is set up by requesting the `VibrationalAnalysis` task with the `Type ModeRefinement`:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeRefinement
  Displacement 0.001
  NormalModes
    ModeFile adf.rkf
    ModeSelect
    ...
  End
  ScanModes true
End
End
```

The details of the calculation are specified in the `NormalModes` block. Here, `ModeFile` specifies the AMS output file containing the normal modes for which you want to calculate the frequencies. The `ModeSelect` block is used to specify which of the modes in this file will be selected for refinement. A more detailed overview of this block is given in the section *Selecting modes* on the [main page](#) (page 266). Finally, `Displacement` can be used to specify the stepsize (in Bohr) for the finite differences. The stepsize is provided for displacements along the Cartesian normal modes.

The `ScanModes` key in the `NormalModes` block can be used to automatically run a [numerical frequencies](#) (page 254) calculation on the new modes q . Mode Refinement uses a linear combination of modes and properties, all obtained through finite differences. These results may still contain some minor errors due to the accumulation of numerical errors from the linear expansion, or stepsize issues in the numerical frequency calculations. While commonly not necessary, it is possible to run an additional numerical refinement calculation on the new modes to minimize these errors. Only in exceptional cases will these errors be significant. Running this additional refinement step is therefore only necessary if you need complete certainty that the results are accurate.

6.2.6 Mode Tracking

The Mode Tracking task is an interface for mode- and intensity-tracking methods, adapted from the MoViPac suite⁴⁻⁵. These methods can be used to obtain select normal modes, without having to calculate the entire vibrational spectrum. It does this through an iterative procedure.

⁴ T. Weymuth, M.P. Haag, K. Kiewisch, S. Luber, S. Schenk, C.R. Jacob, C. Herrmann, J. Neugebauer, M. Reiher, *MoViPac: Vibrational Spectroscopy with a Robust Meta-Program for Massively Parallel Standard Inverse Calculations*, *Journal of Computational Chemistry* 33, 2186 (2012) (<https://doi.org/10.1002/jcc.23036>)

⁵ S. Luber, J. Neugebauer, M. Reiher, *Intensity tracking for theoretical infrared spectroscopy of large molecules*, *Journal of Chemical Physics* 130, 064105 (2009) (<https://doi.org/10.1063/1.3069834>)

- Calculations are conducted for each mode separately. Converges fastest for characteristic (non-highly degenerate) modes.
- Iterative approximation to the true modes. Guaranteed to give the correct normal modes if the procedure converges.
- Will not necessarily reproduce the entire spectrum as multiple guess modes can converge to the same normal mode.

Mode Tracking uses information about the known parts of the Hessian to expand its basis iteratively:

- Missing C-O stretch character can thus be recovered in this procedure, and there is no basis dependency.
- For large regions with similar modes however, it is possible that multiple guess modes converge to the same normal mode. Running mode tracking for all modes in this region might not reproduce all unique normal modes.
- The advantage of Mode Refinement over Mode Tracking is the ability to refine entire spectral regions at once. If we have a good basis, Mode Refinement can be less computationally expensive than Mode Tracking. If you want to refine larger sections of the spectrum, Mode Refinement is therefore recommended. If you only want to calculate a select few modes, use Mode Tracking to avoid basis dependence and to assure accuracy of the obtained modes.
- For characteristic peaks, Mode Tracking shows very good convergence, and will thus be cheaper to use than Mode Refinement. For (semi-)degenerate modes however, Mode Refinement works better due to the poor tracking performance for these modes.

Mode Tracking starts with a *numerical frequency* (page 254) calculation, which refines the initial guess b^m for the selected mode. The error of this mode with respect to the true Hessian eigenvector is calculated. This error is used in a (Jacobi-)Davidson algorithm to generate an additional mode. In subsequent iterations, we use these modes as approximations to the true normal modes. In this way, the error of the mode is minimized iteratively, yielding a closer approximation to true normal modes. This is how Mode Tracking differs from the Mode Refinement methods, in that it guarantees that the obtained modes are correct (assuming the procedure has converged).

See also:

The [GUI tutorial](#) on Mode Tracking.

Theory

During the numerical frequency calculation, we obtain also the projection of the Hessian onto this mode:

$$\sigma_i = H^m \cdot b_i^m = \frac{\partial^2 E}{\partial R_i^m \partial b^m}$$

This term is calculated through finite differences on the analytical gradients of the electronic energy along the mass-weighted normal modes q^m . The index i denotes the $3N$ nuclear coordinates. From this projection a Rayleigh matrix is generated:

$$\tilde{H}^m = B^{mT} \cdot \Sigma$$

Here, B^m and Σ are matrices containing the b^m and σ vectors for all foregoing iterations. During each iteration k , if we have not converged, we generate an updated guess vector b_k^m , and so the number of vectors in the matrices above is equal to the number of iterations k . The eigenvectors of \tilde{H}^m give us the coefficient series for linear combinations of the guess modes b^m such that we obtain approximations for the true normal modes:

$$Q^m = \sum_k c_k \cdot b_k^m$$

Each iteration, we expand the vector basis B^m , which allows this series expansion to come closer to the true normal modes each time. We can also calculate the error of this mode with respect to how close it is to being an eigenvalue of the real Hessian:

$$r = \sum_k c_k \cdot [\sigma_k - \lambda \cdot b_k]$$

Here, λ is the corresponding eigenvalue of \tilde{H}^m . r is the residual vector, giving the error for each vector element. It should be zero if the mode is an exact eigenvector of the true Hessian.

Since \tilde{H}^m may give multiple eigenvectors, several approximate modes will be generated during those iterations. Out of these, 1 mode is identified as the mode of interest according to the specified *tracking method* (page 262). If the residual of this mode has been minimized sufficiently, the procedure has converged. If not, we generate a new guess vector b_k^m . There are 2 algorithms for generating this new guess, set by `UpdateMethod` in the `ModeTracking` block:

Davidson method

The Davidson method uses a pre-conditioner D to generate a new guess mode from the residual vector of the mode selected by the tracking method:

$$b_k^m = D^{-1} \cdot r$$

This preconditioner is constructed from an approximation of the Hessian:

$$D = H_A - \lambda \cdot I$$

The Davidson method works reasonably well, but can have trouble converging if the approximate modes or the Hessian are too accurate. This results as the new vectors that are generated do not necessarily extend the span of the basis.⁶

vdVorst-Slejpen-Jacobi-Davidson

This variant of the Jacobi-Davidson scheme from Sleijpen & vdVorst^{Page 259, 6} automatically makes the new guess vector orthogonal to the normal mode selected by the tracking method:

$$b_k^m = \left(\frac{Q^m D^{-1} r}{Q^m D^{-1} Q^m} \right) D^{-1} Q^m - D^{-1} r$$

The new vector is therefore guaranteed to extend the span of the basis as much as possible, and thus also eliminates the aforementioned issue with the Davidson method. In general, it is therefore recommended to use this Jacobi-Davidson method since it is found to converge faster, and be more reliable, as a result of yielding better guess modes.

Input

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeTracking
  ...
  ModeTracking
    HessianGuess [Unit | File | UFF | Inline]
    HessianInline # Non-standard block. See details.
    ...
  End
  HessianPath string
  UpdateMethod [JD | D | I]
  MaxIterations integer
  ...
  GramSchmidt [True | False]
```

(continues on next page)

⁶ G.L.G. Sleijpen, H.A. van der Vorst, *A Jacobi-Davidson Iteration Method for Linear Eigenvalue Problems*, SIAM Journal on Matrix Analysis and Applications 17, 401 (1996) (<https://doi.org/10.1137/S0895479894270427>)

(continued from previous page)

```

    GramSchmidtIterations integer
    GramSchmidtTolerance float
End
End

```

There are 4 methods to obtain the approximate Hessian H_A , used by both update methods. They are set by `HessianGuess`:

HessianGuess [Unit | File | UFF | Inline]

UFF

is the default, which generates the approximate Hessian using UFF. While this Hessian may not yield the correct modes by itself, it produces good results as a preconditioner since it correctly represents the molecular structure.

File

will read the Hessian from an AMS output file, which can be specified in `HessianPath`. Using a Hessian from a more advanced method will generally yield better results for the Jacobi-Davidson method. The Davidson method will however experience difficulties with convergence as the Hessian becomes too accurate.⁶

Inline

will read a Hessian specified in the input file, in the `HessianInline` block. This allows you to use Hessians generated in external programs:

```

Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeTracking
  ModeTracking
    HessianGuess Inline
    # Approximate Hessian for H2O: 3 x nAtoms = 9 so 9x9 Hessian
    HessianInline
      0.62088786  0.00000000  0.00000000 -0.31044393  0.00000000 -0.
↪21902068 -0.31044393  0.00000000  0.21902068
      0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.
↪00000000  0.00000000  0.00000000  0.00000000
      0.00000000  0.00000000  0.32143213 -0.15284008  0.00000000 -0.
↪16071607  0.15284008  0.00000000 -0.16071607
      -0.31044393  0.00000000 -0.15284008  0.33598889  0.00000000  0.
↪18593038 -0.02554496  0.00000000 -0.03309030
      0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.
↪00000000  0.00000000  0.00000000  0.00000000
      -0.21902068  0.00000000 -0.16071607  0.18593038  0.00000000  0.
↪15761846  0.03309030  0.00000000  0.00309761
      -0.31044393  0.00000000  0.15284008 -0.02554496  0.00000000  0.
↪03309030  0.33598889  0.00000000 -0.18593038
      0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.
↪00000000  0.00000000  0.00000000  0.00000000
      0.21902068  0.00000000 -0.16071607 -0.03309030  0.00000000  0.
↪00309761 -0.18593038  0.00000000  0.15761846
    End
  End
End

```

Unit

uses the unit matrix. This is evidently not a good approximation for the Hessian, and is not intended to be used for proper Mode Tracking runs. However: using a poor approximation for the Hessian can result in basis vectors being generated that we would not obtain otherwise. Running Mode Tracking with this option can allow you to “probe” the vector space to obtain guesses for normal modes, which can be used as starting points

for proper Mode Tracking calculations. It is however generally recommended to instead do e.g. a DFTB or UFF run if your goal is to obtain guess modes.

UpdateMethod [JD | D | I]

JD

vdVorst-Slejpen variant of Jacobi-Davidson (Mode tracking default).

D

Davidson

I

No preconditioner (VST default). This is not recommended for typical mode tracking applications, but is useful for a variation of mode tracking, *Vibronic-Structure Tracking* (page 299).

In later iterations, the basis B^m will become larger. In order to improve the guess modes even further, an iterative Gram-Schmidt procedure is used to orthogonalize the new guess mode to the existing basis. An iterative procedure is necessary to account for numerical noise.

GramSchmidt [True | False]

Expert key. Sets whether to perform this Gram-Schmidt orthogonalization step. It is `True` by default.

GramSchmidtTolerance float

Expert key. Sets the absolute tolerance for orthogonality of the basis. It is evaluated with respect to the norm of the overlap vector between the new guess mode and the basis of the previous iteration $\|b_k^{mT} B^m\|$.

GramSchmidtIterations

Expert key. Sets the maximum number of allowed iterations during the Gram-Schmidt procedure.

The default settings for the Gram-Schmidt procedure should work for almost all systems.

MaxIterations integer

Finally, the Mode Tracking input block contains the `MaxIterations` key. It sets the maximum allowed number of iterations that the Mode Tracking calculation may go through. If this number is reached, the calculation will stop even if convergence was not achieved. If no value is supplied, a default of $3N/2$ will be used. This is approximately the maximum number of iterations where the procedure remains computationally competitive with the construction of the full Hessian.

Additional input parameters

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeTracking
  Displacement float
  ...
  NormalModes
    ScanModes [True | False]
    ...
  End
End
```

Displacement float

is the displacement stepsize (in Bohr) that is used for calculating frequencies, IR intensities and the Hessian projections through finite differences. The stepsize is provided for displacements along the Cartesian normal modes.

ScanModes [True | False]

key (False by default) in the `NormalModes` vibrational analysis sub-block can be used to automatically run a *numerical frequencies* (page 254) calculation on the new modes Q after the Mode Tracking calculation has finished. Ritz vectors are obtained here as linear combinations of the guess modes, which in turn follow from finite difference

calculations. This makes it possible for numerical errors to accumulate in the normal modes. Only in exceptional cases will these errors be significant, and running this additional refinement step is therefore only necessary if you need complete certainty that the results are accurate.

Input: Tracking methods

The `TrackingMethod` parameter allows you to select what property of the normal modes you want to track. At the end of each iteration, we obtain a set of approximate normal modes. The tracking method identifies which of these modes fits best for some criterion, and either returns this mode as the calculation result, or, if convergence was not achieved, uses it to generating a new basis mode for the next iteration. In general these methods are distinguished in 3 categories:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeTracking
  ModeTracking
    TrackingMethod [OverlapInitial, DifferenceInitial, FreqInitial, IRInitial,
                   OverlapPrevious, DifferencePrevious, FreqPrevious, IRPrevious,
                   HighestFreq, HighestIR, LowestFreq, LowestResidual]
    ...
  End
End
```

Mode Tracking

The original tracking methods focus on obtaining as accurate as possible a normal mode for the system. This class of tracking methods focuses either on accuracy of the mode, or obtaining modes with particular vibrational character:

TrackingMethod [**OverlapInitial**, **DifferenceInitial**, **FreqInitial**, **OverlapPrevious**, **DifferencePrevious**, **FreqPrevious**, **HighestFreq**, **LowestFreq**, **LowestResidual**]

OverlapInitial

is the default tracking method. Here, we choose the mode which resembles most closely the guess mode that was initially supplied b_1^m . This is done by choosing the mode which has the greatest overlap with the initial guess vector. This method allows us to direct the optimization towards modes that e.g. involve particular atoms or include particular bending/stretching vibrations.

OverlapPrevious

instead chooses the mode which resembles closest the approximate mode of the previous iteration Q_k^m . This procedure allows a bit more flexibility in the optimization. Since we essentially “forget” about earlier iterations, this procedure allows the optimization to correct errors in the initial guess. (It is possible for instance that the initial guess included 2 different bond stretches which do not mutually occur in the true modes. This method will then converge quicker to a mode involving only 1 of these stretches, whereas `OverlapInitial` will take a much larger number of iterations to achieve this, if it does so at all.) Do note that this means that the final mode that you obtain does not necessarily represent the mode you initially supplied.

DifferenceInitial

works the same as `OverlapInitial`, except that it chooses the mode which has the smallest norm for the difference vector between the initial mode and the approximate normal modes of this iteration. The use of the difference vector prioritizes deviations in the dominant parts of the vibrational character. E.g. if a mode consists primarily of a CO stretch, plus some minor vibrations in a carbon backbone, it may be desired to prioritize getting the correct force constant for the dominant CO stretch. This is achieved using these difference vector methods. In general, overlap methods still work well in these situations, and the use of difference methods should only be necessary in extreme cases.

DifferencePrevious

is also the same as `DifferenceInitial` except for the use of the difference vector norm as the selection criterion.

FreqInitial

chooses the mode with the frequency closest to that of the initial guess. This allows us to direct the tracking towards modes in a particular frequency region of the spectrum. Note that convergence for these frequency-based methods is slightly slower since the character of the mode itself is not included in the selection criteria, allowing for larger differences in the modes between iterations.

FreqPrevious

is similar to `FreqInitial` except that we choose the mode with the frequency closest to that of the previous iteration. This allows the optimization more freedom to move away from the frequency region of the initial guess, and thus allows to correct somewhat for poor initial guesses.

HighestFreq

chooses the mode with the highest frequency. This method can be used if it is desired to track particular characteristic high-frequency vibrations.

LowestResidual

chooses the mode which has the smallest norm for the residual vector (see the ‘Convergence’ section below.) This method only focuses on obtain the most accurate mode, regardless of vibrational character or where it lies in the spectrum. This method should generally only be used as a pre-conditioner if you have very little information on what the normal modes should look like. (Since it is basically a non-directed optimization.) This method will then try and find the normal mode closest to your guess. The approximate normal mode obtained this way will most likely not have converged yet, but should give you an indication of what the normal modes may look like. You can use these modes to refine your initial guess, and then do a new Mode Tracking run using any of the other tracking parameters to obtain the desired mode. Although this strategy is possible, it is generally recommended to use an approximate method to get an initial guess for the normal modes instead (as shown in the [examples](#) (page 457)).

Intensity Tracking

This class of methods focuses on tracking modes based on their intensity in e.g. the infrared spectrum, rather than focusing on getting a mode with a particular type of vibration.

TrackingMethod [IRInitial, IRPrevious, HighestIR]

`IRInitial` chooses the mode with the IR intensity closest to that of the initial guess. This constrains the optimization to modes which are IR active, a property that may be lost when using mode tracking update methods.

`IRPrevious` similarly chooses the mode with the IR intensity closest to that of the previous iteration. This allows the method some more flexibility in varying the intensity of the vibration, and thus works better if the initial guess is not that good.

`HighestIR` chooses the mode with the highest IR intensity. This option can be used to find the modes associated with sharp peaks in the IR spectrum.

With Intensity Tracking, we essentially add an additional requirement to the modes: they must have a particular IR intensity. This constrained search has different convergence characteristics than conventional mode tracking, which you should take into account when setting up the mode tracking calculations.

- The majority of modes will have near-zero IR intensity. If we use a near-zero IR intensity mode as our initial guess, and request `IRInitial` or `IRPrevious`, then we could be tracking any of one of these. Conversely, convergence behavior will be poor since the generated basis modes are essentially random. If you are trying to obtain a high IR-intensity mode, use an IR-susceptible mode.

Note: In our conventional work-flow, we recommend starting mode tracking or refinement calculations from a set of

approximate normal modes obtained from a semi-empirical or force-field method. Note however, that these method often do not produce accurate IR intensities. When selecting the initial guess mode, do **not** use the `IRRange` or related options in the `ModeSelect` block. This will cause you to miss vibrations which were incorrectly labeled with low IR intensity, or vice versa. Instead, rely on chemical intuition to identify the modes which contain commonly IR active vibrational components (such as C-O or N-H stretches). You can use `AMSSpectra` in the GUI to visualize the vibrational modes, to help you in this process.

- To allow the intensity tracking procedure to converge faster, it is recommended to use the `IRPrevious` tag instead of the `IRInitial` tag. As discussed earlier, the former allows more flexibility in the optimization procedure, which counters the rigidity imposed by the intensity constraint. Intensity tracking methods often need this additional flexibility in generating guess modes to converge to the desired modes.
- **Poor Initial Guesses:** During each iteration, we still use the mode tracking methods to generate new basis modes. These basis modes try to expand the span of the basis with respect to the vibrational character of the modes. Note that this expansion does not guarantee that we will expand the basis specifically in the sub-span of IR-susceptible vibrations. If the initial guess for intensity tracking is correct, we already start our search in the sub-span vicinity of the normal modes. Basis expansion is then more efficient and there is a high chance that new guess modes sample the IR characteristic vibrations. For intensity tracking it is therefore discouraged to use poor initial guess modes.
- `HighestIR` is considered a “pure” intensity tracking method, in that it is used specifically to target characteristics of the IR spectrum irrespective of the underlying vibrational character. Consequently, the normal mode character can vary a lot between iterations. In order to assure that the procedure converges to the desired modes, it is recommended to use sufficiently strict tolerances (see the *Convergence* section). If the tolerances are too lax, the program may consider the modes to be “good enough” based on residual minimization, even though there may be another mode with a higher IR intensity. For this reason it is generally recommended to use `ToleranceForNorm` values 1 order of magnitude lower than the default, or around 0.00005.

Input: Selecting modes

It is possible to track multiple modes in a single Mode Tracking calculation. The Mode Tracking task will then run the Mode Tracking algorithm for each mode in order.

The initial guess for the mode which will be tracked can be supplied in several ways. This is governed by `ModeInputFormat`:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeTracking
  NormalModes
    ModeInputFormat [File | Inline | Hessian]
    ModeFile string
    ModeInline # Non-standard block. See details.
    ...
  End
  ModeSelect
    ...
  End
  MassWeighInlineMode [True | False]
End
End
```

ModeInputFormat [File | Inline | Hessian]

Inline

will make the module read the mode from the input file. If this option is selected, you can supply the mode in the `ModeInline` block. It is possible to supply multiple modes by adding additional `ModeInline` blocks.

The modes are given with one line for the x,y,z-displacement per atom, and in the same order, as the `Atoms` block in `System`:

```
ModeTracking
  TrackedMode Inline
  ModeInline
    0.00000000  0.00000000 -0.03815965
    -0.18888544 0.00000000  0.30281066
    0.18888544  0.00000000  0.30281066
  End
  ModeInline
    0.00000000  0.00000000 -0.02243153
    0.32132452 0.00000000  0.17800237
    -0.32132452 0.00000000  0.17800237
  End
  ...
End
```

File

will make the module read modes from an AMS or engine output file, specified by `ModePath`. Modes generated using DFTB can be read from the `dftb.rkf` file and optimized using Mode Tracking for example. When this option is selected, all the vibrational modes present in the file are read first. The `ModeSelect` block then specifies for which of these modes you want to perform the Mode Tracking calculation.

Hessian

will generate modes as the eigenvectors of the approximate Hessian selected for the preconditioner in `HessianGuess`. This also allows modes to be generated for Hessians obtained from external programs. `ModeSelect` specifies which of the generated vibrational modes are selected for Mode Tracking.

- Settings for the `ModeSelect` block are discussed on the [main page](#) (page 266).

MassWeighInlineMode [True | False]

decides whether the initial guess modes need to be mass-weighted (default True). As discussed above, Mode Tracking uses mass-weighted normal modes. In most cases, the normal modes are given in regular Cartesian coordinates however. By setting `MassWeighInlineMode true`, these Cartesian modes are converted into mass-weighted modes by the program. If you supply a mass-weighted mode through the `ModeInline` block however, you do not need the program to do the mass-weighting, and you should set `MassWeighInlineMode false`.

Input: Convergence

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ModeTracking
  ModeTracking
    ToleranceForNorm float
    ToleranceForResidual float
    ToleranceForBasis float
  End
End
```

In order to guide the Mode Tracking procedure, several convergence criteria are used:

ToleranceForNorm float

is the absolute tolerance for convergence of the norm of the residual vector. The residual vector is a vector containing the error for each element of the normal mode, and we use the norm as a measure for the total error. If the total error is smaller than this threshold, we consider the mode to be a true normal mode and we stop iterating. Since

the value of this norm depends on the length of the residual vector hence the number of atoms in the system, this tolerance is scaled internally to the number of atoms. 0.0005 is used as a default value for which most systems will converge to reasonably accurate modes in not too many iterations. If you want a more accurate approximation, you can decrease this value by e.g. 1 order of magnitude. (Consider running using the default settings, and reading the norm at convergence from the logfile. The new norm can be chosen to be lower than this value to ‘force’ the method into another iteration.)

ToleranceForResidual float

is the absolute tolerance for the maximum component of the residual vector. Particularly in larger systems, where the vibration may be dominated by a small number of atoms, the error associated with the vibration of the majority of atoms may be small (the scaled residual norm will be small). The error for the atoms involved in the vibration may be comparatively large then, which is why we also check convergence for the maximum component of the error. Note that both the norm and this max. error are checked simultaneously. By varying strictness of the criteria for the norm and the max. error separately, you can prioritize either the total vibration or more localized character.

ToleranceForBasis float

checks that the basis mode generated in the previous iteration, through the (Jacobi-)Davidson method, contributes to the approximate normal mode. Since the approximate mode is taken as a linear combination of the basis modes, its linear expansion coefficient must be larger than this tolerance.

The iterative procedure is stopped in one of two cases. Either both the residual criteria are achieved, in which case the mode is deemed to be converged and the program exits normally. Alternatively, the basis criterion is met in which case a warning is broadcast indicating that the desired level of accuracy of the mode may not have been reached yet, but the basis has stopped expanding. The default values for these parameters should be applicable for most cases, but can be adjusted if needed. If stricter criteria are required, it is recommended to adjust both `ToleranceForNorm` and `ToleranceForResidual`.

6.2.7 Selecting modes

Mode Scanning, Mode Refinement and Mode Tracking as well as VG-FC Vibronic-Structure, VG-FC Vibronic-Structure Refinement and VG-FC resonance Raman all require a set of normal modes to operate on. For Mode Scanning these are the modes that you want to calculate the properties of, for Mode Refinement these modes form the basis modes, and for Mode Tracking these are the initial guess modes. For the VG-FC based methods these modes are the modes responsible for the vibronic coupling to the electronic excitation (in VG-FC Vibronic-Structure Refinement they are refined first).

Note: VG-FC Vibronic-Structure Tracking does not require any normal modes and as such does not support the `ModeSelect` (nor does it support the `NormalModes` block for that matter).

These methods provide options to load a large set of modes, after which the program will filter out the modes of interest. This is done according to the keys set in the `ModeSelect` block.

Note: The `ModeSelect` block is part of the `NormalModes` block of the Vibrational Analysis input. All Vibrational Analysis methods share this block, with the exception of VG-FC Vibronic-Structure Tracking. The methods for obtaining the set of modes that we will filter can differ per method. Particularly Mode Tracking features a lot of additional options, and the vibronic variants feature more specialized options.

Below is an overview of all the available options for the `ModeSelect` block as they appear in the basic vibrational analysis tools. The vibronic variants are discussed in more detail on their respective documentation pages.

The options below are not mutually exclusive.

```
ModeSelect
  DisplacementBound float
```

(continues on next page)

(continued from previous page)

```

FreqAndIRRange float_list
FreqRange float_list
Full Yes/No
HighFreq integer
HighIR integer
IRRange float_list
ImFreq Yes/No
LargestDisplacement integer
LowFreq integer
LowFreqNoIm integer
LowIR integer
ModeNumber integer_list
End

```

ModeSelect**DisplacementBound****Type**

Float

Description

Vibronic Structure (Refinement), Resonance Raman: Select all modes with a dimensionless oscillator displacement greater than the specified value.

FreqAndIRRange**Type**

Float List

Recurring

True

Description

Specifies a combined frequency and IR intensity range within which all modes will be selected. First 2 numbers are the frequency range in cm⁻¹, last 2 numbers are the IR intensity range in km/mol.

FreqRange**Type**

Float List

Unitcm⁻¹**Recurring**

True

Description

Specifies a frequency range within which all modes will be selected. 2 numbers: an upper and a lower bound. Calculating all modes higher than some frequency can be achieved by making the upper bound very large.

Full**Type**

Bool

Default value

No

GUI name

All modes

Description

Select all modes. This only make sense for Mode Scanning calculations.

HighFreq**Type**

Integer

GUI name

High frequencies

Description

Select the N modes with the highest frequencies.

HighIR**Type**

Integer

GUI name

High IR

Description

Select the N modes with the largest IR intensities.

IRRange**Type**

Float List

Unit

km/mol

Recurring

True

Description

Specifies an IR intensity range within which all modes will be selected. 2 numbers: an upper and a lower bound.

ImFreq**Type**

Bool

Default value

No

GUI name

All imaginary frequencies

Description

Select all modes with imaginary frequencies.

LargestDisplacement**Type**

Integer

Description

Vibronic Structure (Refinement), Resonance Raman: Select the N modes with the largest VG-FC displacement.

LowFreq**Type**

Integer

GUI name

Low frequencies

Description

Select the N modes with the lowest frequencies. Includes imaginary modes which are recorded with negative frequencies.

LowFreqNoIm**Type**

Integer

GUI name

Low positive frequencies

Description

Select the N modes with the lowest non-negative frequencies. Imaginary modes have negative frequencies and are thus omitted here.

LowIR**Type**

Integer

GUI name

Low IR

Description

Select the N modes with the smallest IR intensities.

ModeNumber**Type**

Integer List

GUI name

Mode numbers

Description

Indices of the modes to select.

6.2.8 Thermodynamics (ideal gas)

The following thermodynamic properties are calculated by default whenever normal modes are computed: entropy, internal energy, constant volume heat capacity, enthalpy and Gibbs free energy. Translational, rotational and vibrational contributions are calculated for entropy, internal energy and constant volume heat capacity.

The results are written to the output file (section: “Statistical Thermal Analysis”) and to the engine binary results file (section: “Thermodynamics”).

The thermodynamic properties are computed assuming an ideal gas, and electronic contributions are ignored. The latter is a serious omission if the electronic configuration is (almost) degenerate, but the effect is small whenever the energy difference with the next state is large compared to the vibrational frequencies. The thermal analysis is based on the temperature dependent partition function. The energy of a (non-linear) molecule is (if the energy is measured from the

zero-point energy)

$$\frac{E}{NkT} = \frac{3}{2} + \frac{3}{2} + \sum_j^{3N-6} \left(\frac{h\nu_j}{2kT} + \frac{h\nu_j}{kT(e^{h\nu_j/(kT)} - 1)} \right) - \frac{D}{kT}$$

The summation is over all harmonic ν_j , h is Planck's constant and D is the dissociation energy

$$D = D_0 + \sum_j \frac{h\nu_j}{2}$$

Contributions from low (less than 20 1/cm) frequencies to entropy, heat capacity and internal energy are excluded from the total values, but they are listed separately (so the user can add them if they wish).

As an alternative to outright excluding low-frequency contributions, a correction scheme is available that is based on interpolating between harmonic oscillator and free rotor values⁷⁸ (Li/Head-Gordon and Grimme). It can greatly reduce the impact of the inaccuracies of the harmonic oscillator model on thermodynamic properties at these low frequencies. The scheme corrects vibrational contributions to entropies, internal energies and constant volume heat capacities. This correction is applied automatically and its results are printed separately (in the text output, the corrected terms are marked with the symbol (c)). When applied, the correction considers all real frequencies, including those less than 20 1/cm.

The interpolation for a corrected thermodynamic property f at pressure p and temperature T in terms of harmonic oscillator terms f_{HO} , free rotor terms f_{FR} , and interpolator terms x for each harmonic oscillator frequency ν_j is:

$$f(p, T) = \sum_j x(\nu_j) \cdot f_{HO}(p, T, \nu_j) + (1 - x(\nu_j)) \cdot f_{FR}(p, T, \nu_j)$$

$$x(\nu_j) = \frac{1}{1 + \left(\frac{\nu_0}{\nu_j}\right)^\alpha}$$

Where α is an arbitrary exponent and ν_0 is the harmonic oscillator frequency around which x interpolates, with $x = 0.5$ when $\nu_j = \nu_0$ and $x \approx 1.0$ when $\nu_j \gg \nu_0$.

While the free rotor terms used for internal energies and heat capacities are the standard ones, the terms used for entropies have to use the rotors' moments of inertia μ_{FR} and symmetry σ , which formally cannot be calculated from harmonic frequencies alone. The correction scheme instead estimates each moment of inertia as being of a $\sigma = 1$ free rotor whose first excited state has an energy equal to the given $h\nu_j$. After this, each obtained moment of inertia μ_{FR} is modified by an averaging moment of inertia μ_{av} to avoid grossly overestimating entropies at very small frequencies (less than around 1 1/cm):

$$\mu = \frac{\mu_{FR} \cdot \mu_{av}}{\mu_{FR} + \mu_{av}}$$

Input options

```
Thermo
  Temperatures float_list
  Pressure float
  LowFrequencyCorrector
    Alpha float
    Frequency float
    MomentOfInertia float
  End
End
```

⁷ Yi-Pei Li, Joseph Gomes, Shaama Mallikarjun Sharada, Alexis T. Bell, Martin Head-Gordon, *Improved Force-Field Parameters for QM/MM Simulations of the Energies of Adsorption for Molecules in Zeolites and a Free Rotor Correction to the Rigid Rotor Harmonic Oscillator Model for Adsorption Enthalpies*, J. Phys. Chem. C 2015, 119, 4, 1840-1850 (<https://doi.org/10.1021/jp509921r>)

⁸ Stefan Grimme, *Supramolecular Binding Thermodynamics by Dispersion-Corrected Density Functional Theory*, Chem. Eur. J., 18: 9955-9964 (<https://doi.org/10.1002/chem.201200497>)

Thermo**Type**

Block

Description

Options for thermodynamic properties (assuming an ideal gas). The properties are computed for all specified temperatures.

Temperatures**Type**

Float List

Default value

[298.15]

Unit

Kelvin

Value Rangevalue ≥ 0 **Description**

List of temperatures at which the thermodynamic properties will be calculated.

Pressure**Type**

Float

Default value

1.0

Unit

atm

Description

The pressure at which the thermodynamic properties are computed.

LowFrequencyCorrector**Type**

Block

Description

Options for the dampener-powered free rotor interpolator that corrects thermodynamic quantities for low frequencies. See DOI:10.1021/jp509921r and DOI:10.1002/chem.201200497.

Alpha**Type**

Float

Default value

4.0

Description

The exponent term used in the dampener.

Frequency**Type**

Float

Default value

100.0

Unit

cm-1

Description

The frequency around which the dampener interpolates between harmonic oscillator and free rotor quantities.

MomentOfInertia**Type**

Float

Default value

1e-44

Unit

kg m^2

GUI name

Averaging Moment of Inertia

Description

The moment of inertia used to restrict entropy results for very small frequencies (generally around less than 1 cm-1).

Gibbs free energy change for a gas phase reaction

Here an example is given how to calculate the free energy change for a reaction. In the AMS output of a normal modes calculation you can find the electronic bonding energy and nuclear kinetic energies, at room temperature. Example part of the AMS output of a nonlinear molecule:

```

Zero-point energy (Hartree):      0.0333

...
...

Temp                                Transl      Rotat      -
↪Vibrat      Total
-----
↪--      -----

298.15  Entropy (cal/mol-K):      34.441      11.474      0.
↪137      46.052
      Nuclear Internal Energy (kcal/mol):      0.889      0.889      20.
↪941      22.718
      Constant Volume Heat Capacity (cal/mol-K):      2.981      2.981      0.
↪565      6.526

Summary of energy terms

                                hartree      eV      kcal/
↪mol      kJ/mol
-----
↪
Energy from Engine:      -0.743995039793930      -20.2451      -466.
↪86      -1953.36
Nuclear Internal Energy:      0.036203917534227      0.9852      22.

```

(continues on next page)

(continued from previous page)

→72	95.05			
Internal Energy U:		-0.707791122259703	-19.2599	-444.
→14	-1858.31			
pV/n = RT:		0.000944186013486	0.0257	0.
→59	2.48			
Enthalpy H:		-0.706846936246217	-19.2343	-443.
→55	-1855.83			
-T*S:		-0.021880868282982	-0.5954	-13.
→73	-57.45			
Gibbs free energy:		-0.728727804529199	-19.8297	-457.
→28	-1913.27			

The Energy from Engine = -466.86 kcal/mol. It depends on the engine how this energy is calculated. In the ADF and BAND engines the energy is normally calculated with respect to (artificial) spherical averaged neutral atoms.

The Nuclear Internal Energy = zero point energy + 3 kT + small correction term = 22.72 kcal/mol. 3 kT = 3/2 kT for rotation, and 3/2 kT for translation (i.e. 1/2 kT for each degree of freedom). The small correction term is a term due to the vibration partition function, depending on the temperature not only the ground state vibrational levels are occupied, see also the previous discussion.

The Internal Energy U = Energy from Engine + Nuclear Internal Energy = -466.86 + 22.72 = -444.14 kcal/mol. Gas phase pV/n = RT = 8.314472 * 298.15 / 4184 = 0.59 kcal/mol. The enthalpy H = U + pV = -444.14 + 0.59 = -443.55 kcal/mol. The Gibbs free energy G = H - TS = -443.55 - 298.15*46.052/1000 = -457.28 kcal/mol.

For a calculation of the free energy change for reaction (ΔG), you will have to do this for the reactant and product molecules, and add and subtract these energies, for each molecule proportional to the number of molecules that take place in the reaction. Application of ADF for obtaining enthalpy, entropy and Gibbs free energy can for instance be obtained in Refs.^{9,10}.

6.2.9 Moments of inertia

In case normal modes are computed in AMS, AMS also reports the moments of inertia of the molecule in units of amu bohr² (amu = atomic mass unit) and its corresponding principal axes.

6.2.10 Partial Vibrational Spectra (PVDOS)

The Partial Vibrational Spectra (also known as PVDOS) is computed by default whenever normal modes are requested. The PVDOS $P_{I,n}$ for atom I and normal mode n is defined as:

$$P_{I,n} = \frac{m_I |\vec{\eta}_{I,n}|^2}{\sum_J m_J |\vec{\eta}_{J,n}|^2}$$

where m_I is the nuclear weight of atom I , and $\vec{\eta}_{I,n}$ is the displacement vector for atom I in normal normal mode n .

Tip: The Partial Vibrational Spectra (PVDOS) can be visualized using the **AMSspectra** GUI module (**Vibrations → Partial Vibrational Spectra (PVDOS)**). When plotting a partial vibrational spectrum, the IR intensity of normal modes is scaled by the corresponding PVDOS of the selected atoms.

The PVDOS matrix is not printed to the text output, but only saved to the engine binary output (.rkf) in the variable `Vibrations%PVDOS`.

⁹ M. Swart, E. Rösler, and F. M. Bickelhaupt, *Proton affinities of maingroup-element hydrides and noble gases: Trends across the periodic table, structural effects, and DFT validation*, *Journal of Computational Chemistry* 27, 1486 (2006) (<https://doi.org/10.1002/jcc.20431>)

¹⁰ M. Swart, and F. M. Bickelhaupt, *Proton Affinities of Anionic Bases: Trends Across the Periodic Table, Structural Effects, and DFT Validation*, *Journal of Chemical Theory and Computation* 2, 281 (2006) (<https://doi.org/10.1021/ct0502460>).

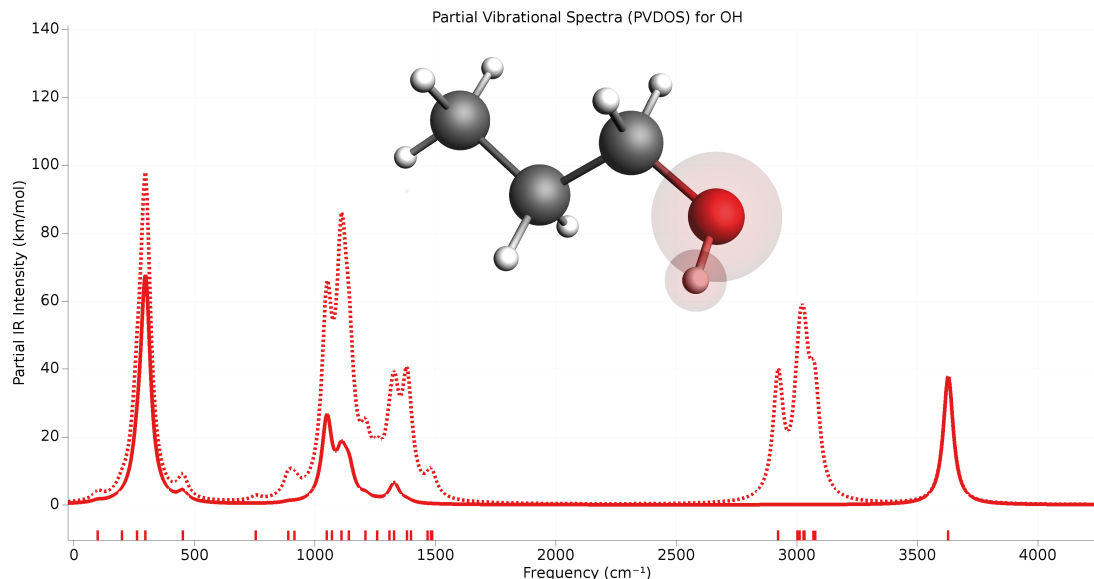


Fig. 6.1: Example of partial vibrational spectrum (PVDOS). The dotted line is the full IR spectrum of 1-propanol. The solid line is the PVDOS-scaled IR spectrum of the OH group (IR spectrum computed using GFN1-xTB).

6.3 Phonons

Collective oscillations of atoms around their equilibrium positions, giving rise to lattice vibrations, are called phonons. AMS can calculate phonon dispersion curves within standard harmonic theory, implemented with a finite difference method. Within the harmonic approximation we can calculate the partition function and therefore thermodynamic properties, such as the specific heat and the free energy.

See also:

Example: Phonons for graphene (page 553), *Example: Phonons with isotopes* (page 554), *Example: User-defined Brillouin zone for phonon dispersion* (page 556) and *silicon lattice optimization and phonons tutorial*

The calculation of phonons is enabled in the `Properties` block.

```
Properties
  Phonons Yes/No
End
```

Note: Phonon calculations should be performed on optimized geometries, **including the lattice vectors**. This can be done by either using an already optimized system as input, or by combining the phonon calculation with the *geometry optimization task* (page 49) (you should set the `GeometryOptimization%OptimizeLattice` input option to `True`).

The details of the phonon calculations are configured in the `NumericalPhonons` block:

```
NumericalPhonons
  SuperCell # Non-standard block. See details.
  ...
End
  StepSize float
  DoubleSided Yes/No
```

(continues on next page)

(continued from previous page)

```

Interpolation integer
NDosEnergies integer
AutomaticBZPath Yes/No
BZPath
  Path # Non-standard block. See details.
  ...
End
End
Parallel
  nCoresPerGroup integer
  nGroups integer
  nNodesPerGroup integer
End
End

```

NumericalPhonons**SuperCell****Type**

Non-standard block

Description

Used for the phonon run. The super lattice is expressed in the lattice vectors. Most people will find a diagonal matrix easiest to understand.

The most important setting here is the super cell transformation. In principle this should be as large as possible, as the phonon bandstructure converges with the size of the super cell. In practice one may want to start with a 2x2x2 cell and increase the size of the super cell until the phonon band structure converges:

```

NumericalPhonons
  SuperCell
    2 0 0
    0 2 0
    0 0 2
  End
End

```

By default the phonon dispersion curves are computed for the standard path though the Brillouin zone (see <https://doi.org/10.1016/j.commatsci.2010.05.010>). One can request the a different path using the following keywords (for an example of how to specify a user-defined path see *Example: User-defined Brillouin zone for phonon dispersion* (page 556)):

```

NumericalPhonons
  AutomaticBZPath Yes/No
  BZPath
    Path # Non-standard block. See details.
    ...
  End
End
End

```

NumericalPhonons**AutomaticBZPath****Type**

Bool

Default value

Yes

GUI name

Automatic BZ path

Description

If True, compute the phonon dispersion curve for the standard path through the Brillouin zone.
If False, you must specify your custom path in the [BZPath] block.

BZPath**Type**

Block

Description

If [NumericalPhonons%AutomaticBZPath] is false, the phonon dispersion curve will be computed for the user-defined path in the [BZPath] block. You should define the vertices of your path in fractional coordinates (with respect to the reciprocal lattice vectors) in the [Path] sub-block. If you want to make a jump in your path (i.e. have a discontinuous path), you need to specify a new [Path] sub-block.

Path**Type**

Non-standard block

Recurring

True

Description

A section of a k space path. This block should contain multiple lines, and in each line you should specify one vertex of the path in fractional coordinates. Optionally, you can add text labels for your vertices at the end of each line.

Other keywords in the NumericalPhonons block modify the details of the numerical differentiation procedure and the accuracy of the results:

NumericalPhonons**StepSize****Type**

Float

Default value

0.04

Unit

Angstrom

Description

Step size to be taken to obtain the force constants (second derivative) from the analytical gradients numerically.

DoubleSided**Type**

Bool

Default value

Yes

Description

By default a two-sided (or quadratic) numerical differentiation of the nuclear gradients is used. Using a single-sided (or linear) numerical differentiation is computationally faster but much less accurate. Note: In older versions of the program only the single-sided option was available.

Interpolation**Type**

Integer

Default value

100

Description

Use interpolation to generate smooth phonon plots.

NDosEnergies**Type**

Integer

Default value

1000

Description

Nr. of energies used to calculate the phonon DOS used to integrate thermodynamic properties. For fast compute engines this may become time limiting and smaller values can be tried.

The numerical phonon calculation supports AMS' *double parallelization* (page 20), which can perform the calculations for the individual displacements in parallel. This is configured automatically, but can be further tweaked using the keys in the `NumericalPhonons%Parallel` block:

```
NumericalPhonons
  Parallel
    nCoresPerGroup integer
    nGroups integer
    nNodesPerGroup integer
  End
End
```

NumericalPhonons**Parallel****Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

6.4 (Resonance) Raman

6.4.1 Raman

In this method the Raman scattering spectrum is calculated from the geometrical derivatives of the frequency-dependent polarizability. Engine ADF is required. Raman scattering intensities and depolarization ratios for all or a selected number of molecular vibrations at a certain laser frequency can be calculated. The Raman scattering calculation is very similar to an IR intensity calculation. In fact, all IR output is automatically generated as well. At all distorted geometries the dipole polarizability tensor is calculated. This is time-consuming and is only feasible for small molecules.

```
Properties
  Raman Yes/No
End
```

```
Raman
  IncidentFrequency float
  FreqRange float_list
End
```

If a `FreqRange` is included the Raman intensities are calculated for a range of vibrational frequencies only. Using this option is a fast alternative for calculating all Raman intensities.

Note that in order to change the nuclear step size that is used in calculating the numerical derivative of the polarizability in case of `FreqRange` one should use `VibrationalAnalysis%Displacement` and `NumericalDifferentiation%NuclearStepSize` otherwise. A larger value for the nuclear step size may be needed since it is harder to calculate the polarizability with enough precision than, for example, the dipole moment.

Properties**Raman****Type**

Bool

Default value

No

Description

Requests calculation of Raman intensities for vibrational normal modes.

Raman**IncidentFrequency****Type**

Float

Default value

0.0

Unit

eV

Description

Frequency of incident light.

FreqRange**Type**

Float List

Unit

cm-1

Recurring

True

GUI name

Frequency range

Description

Specifies a frequency range within which all modes will be scanned. 2 numbers: an upper and a lower bound.

6.4.2 Resonance Raman: excited-state finite lifetime

Resonance Raman spectroscopy uses incident light with a wavelength close to that of an electronic transition. In this method (Ref.¹¹) the resonance Raman-scattering (RRS) spectra is calculated from the geometrical derivatives of the frequency-dependent polarizability. Engine ADF is required. The polarizability derivatives are calculated from resonance polarizabilities by including a finite lifetime (phenomenological parameter) of the electronic excited states.

```
Raman
  FreqRange float_list
  IncidentFrequency float
  LifeTime float
End
```

Raman**FreqRange****Type**

Float List

¹¹ L. Jensen, L. Zhao, J. Autschbach and G.C. Schatz, *Theory and method for calculating resonance Raman scattering from resonance polarizability derivatives*, *Journal of Chemical Physics* 123, 174110 (2005) (<https://doi.org/10.1063/1.2046670>)

Unit

cm-1

Recurring

True

GUI name

Frequency range

Description

Specifies a frequency range within which all modes will be scanned. 2 numbers: an upper and a lower bound.

IncidentFrequency**Type**

Float

Default value

0.0

Unit

eV

Description

Frequency of incident light.

LifeTime**Type**

Float

Default value

0.0

Unit

hartree

Description

Specify the resonance peak width (damping) in Hartree units. Typically the lifetime of the excited states is approximated with a common phenomenological damping parameter. Values are best obtained by fitting absorption data for the molecule, however, the values do not vary a lot between similar molecules, so it is not hard to estimate values. A typical value is 0.004 Hartree.

It is similar to the simple excited-state gradient approximation method (see next section) if only one electronic excited state is important, however, it is not restricted to only one electronic excited state. In the limit that there is only one possible state in resonance the two methods should give more or less the same results. However, for many states and high-energy states and to get resonance Raman profiles (i.e., Raman intensities as a function of the energy of the incident light beam) this approach might be more suitable. The resonance Raman profiles in this approach are averaged profiles since vibronic coupling effects are not accounted for.

6.4.3 Resonance Raman: VG-FC

According to a the time-dependent picture of resonance-Raman (RR) scattering the relative intensities of RR scattering cross sections are, under certain assumptions, proportional to the square of the excited-state energy gradients projected onto the ground-state normal modes of the molecule (see Ref.¹²). For an alternative implementation of RR scattering using a finite lifetime of the excited states, and a discussion of some of the differences, see the previous section. Engine ADF or DFTB is required.

The vertical gradient Franck-Condon (VG-FC) method, also called the Independent Mode Displaced Harmonic Oscillator (IMDHO) model, we use to calculate vibrationally resolved absorption spectra can also be applied to the calculation of resonance Raman spectra. In resonance Raman spectroscopy a molecule is excited from its ground state to some electronically excited state. After a short period of time, the molecule then relaxes back to its electronic ground state. However, when doing so, it might end up in a different vibrational state than it started off in. The result is an energy difference between the incident and emitted photon. One can then plot the intensity for different energy differences to produce what is known as a Raman spectra. Resonance Raman spectroscopy uses incident light with a wavelength close to that of an electronic transition.

AMS supports the calculation of such spectra by modeling the vibronic coupling of electronic transitions using the VG-FC model. This model is discussed also on the *Vibronic-Structure* (page 297) documentation page. Here we will discuss the modifications necessary to use the VG-FC model for resonance Raman spectroscopy. It is worth noting that this VG-FC resonance Raman application does not support the mode selective options. As a result the VG-FC Resonance Raman application will always first perform a full frequency analysis to obtain its normal modes.

Theory

While the basic theory behind the VG-FC model is explained in detail on the *Vibronic-Structure* (page 297) documentation page, we will briefly summarize the most important points here, as well as the modifications necessary for its application to resonance Raman spectroscopy. It applies the harmonic approximation to both the ground state and excited state PES and then goes on to assume that neither frequency changes nor normal mode rotations occur. Thus the excited state PES is a shifted version of the ground state PES. We do not include temperature effects (so all initial states will be ground states) and work at the Franck-Condon point. Under these assumptions, the Raman polarizability of a particular excited state n , for a transition between initial and final vibrational states I and F can be written as:

$$(\alpha_{n,ij})_{F \leftarrow I} = \mu_{n,i} \mu_{n,j} \int_0^\infty \langle F | I_n(t) \rangle e^{i[\omega - (E_{n,0} - E_{m,0})]t} \cdot e^{-\Gamma t} dt$$

Here, i, j label the components of the polarizability tensor and $\langle F | I_n(t) \rangle$ denotes the overlap of the initial state I , propagated along the excited state PES with the final state F . Under the *assumptions of the VG-FC model* (page 286), this overlap is equal to:

$$\langle F | I_n(t) \rangle = \prod_{j=1}^{N_{modes}} \left\{ \frac{(-1)^{m_j} \Delta_{n,j}^{m_j}}{2^{m_j/2} m_j!} (1 - e^{-i\omega_j t})^{m_j} \right\} \exp \left[-\frac{\Delta_{n,j}^2}{2} (1 - e^{-i\omega_j t}) \right]$$

Where the m_j denote the excitation number of mode j in final state F . For a more detailed discussion, we refer to¹³. The only parameters that appear in our expression are the dimensionless oscillator displacements $\Delta_{n,j}$ that represent displacement of the excited state PES along normal mode j . Under the simplifying assumptions of the VG-FC, these can be obtained from the ground state normal modes and a single excited state gradient. The Raman intensity is then proportional to the square of the polarizabilities:

$$\sigma(\omega)_{F \leftarrow I} \propto \sum_{i,j} \left| \sum_n (\alpha_{n,ij})_{F \leftarrow I} \right|^2$$

¹² J. Neugebauer, E.J. Baerends, E. Efremov, F. Ariese and C. Gooijer, *Combined Theoretical and Experimental Deep-UV Resonance Raman Studies of Substituted Pyrenes*, *Journal of Physical Chemistry A* 109, 2100 (2005) (<https://doi.org/10.1021/jp045360d>)

¹³ T. Petrenko and F. Neese, *Analysis and prediction of absorption band shapes, fluorescence band shapes, resonance Raman intensities, and excitation profiles using the time-dependent theory of electronic spectroscopy* *The Journal of Chemical Physics* 127, 164319 (2007) (<https://doi.org/10.1063/1.2770706>)

A spectrum is then generated by including various different final states F , which are defined by different combinations of normal mode excitation numbers, and assigning a relative intensity to each transition equal to the above expression. AMS only supports spectra which display relative intensities so the results are plotted in arbitrary units and are normalized such that the largest peak reaches an intensity of 1.

Input

The calculation setup for resonance Raman spectra largely proceeds as it does for [Absorption Spectra](#) (page 297). We need a set of ground state normal modes as well as an excited state gradient. The former are calculated at the start using the selected AMS engine, or, in case the user has a pre-calculated set of normal modes, these can be read from a .rkf file using the `ModeFile` key in the `NormalModes` sub-block. In this latter case, the engine is not used. The `ModeSelect` block can be used to select specific modes from the full set of normal modes for which the spectrum should be calculated. For details see the [Mode Select](#) (page 266) documentation on the main page. If one simply wants the spectrum for the full set of normal modes, the `Full` key in the `ModeSelect` block can be set to `True`. The excited state information is passed to the application via the `ExcitationSettings` block.

Another point to note is that since our states are labeled by discrete indices we will be calculating stick spectra (which can be homogeneously broadened in `amsspectra`). By contrast, the absorption spectra produced by `VibronicStructure` are raw x,y data. Due to this difference in nature of the Raman spectrum compared to the absorption spectrum, this method uses the `ResonanceRaman` block for input options related to its spectrum (as opposed to the `Absorption-Spectrum` block).

The `ExcitationSettings` block is discussed on the [Vibronic-Structure](#) (page 304) page. One important difference with the latter is that Resonance Raman calculations are supported for more than one excitation at once. This is more important for the case of Raman spectra as the intensity associated with a set of transitions is not equal to the sum of their individual intensities (we sum over electronic states n before we square the polarizabilities). Here we will address settings specific to the Raman spectrum, all of which can be found in the `ResonanceRaman` block. A short example of how a typical input file might look is included at the end of this section.

```
Task VibrationalAnalysis
VibrationalAnalysis
  Task ResonanceRaman
  ResonanceRaman
    IncidentFrequency float
    LifeTime float
    RamanOrder integer
    RamanRange float_list
    MaximumStates integer
  End
  ...
End
```

IncidentFrequency float

Frequency of incident light.

LifeTime float

sets the value of Γ (in Hartree) that controls the exponential damping in our integral. This phenomenological parameter can be interpreted as the (inverse) life time of the Raman excited state and can be used to help the results agree with experiment. The default value of $4.5\text{e-}4$ is on the low end of reasonable values but should provide a good starting point for most cases.

RamanOrder integer

determines the set of final states and overtones to be included in the spectrum. It is an integer and the application considers only final states such that the sum of excitation numbers of all normal modes is less than or equal to this number. Setting this to 1 means we only include the fundamental band.

RamanRange float_list

this keyword specifies the frequency range (in cm^{-1}) the Raman shift is restricted to lie in. This prevents us from including excessively many states and overtones for high frequency modes. The default is $[0, 2000] \text{ cm}^{-1}$ but this can be changed to whatever is desired.

MaximumStates integer

Expert key. Due to the combinatorial explosion of included final states that occurs for combinations of large values of the raman order, large molecules and wide spectrum ranges, there is a maximum number of final states that can be included in the spectrum. This is to prevent the program from using excessive amounts of memory/computation times. The user can set this number using the `MaximumStates` key but this should be done with caution.

Finally we give an example of a typical `VibrationalAnalysis` block for a resonance Raman calculation. This also gives an idea of how the settings that were not explicitly mentioned above work:

```
VibrationalAnalysis
  Type ResonanceRaman
  NormalModes
    ModeSelect
      Full True
    End
  End
  ExcitationSettings
    ExcitationInfo File
    ExcitationFile ./your_excitation/dftb.rkf
    Singlet
      A 1 2 4
    End
  End
  ResonanceRaman
    RamanOrder 3
    RamanRange 0.0 3000.0
  End
End
```

6.5 VROA: (Resonance) vibrational Raman optical activity

The normal and resonance VROA spectra are calculated from geometric derivatives of the different generalized polarizabilities obtained using linear response theory which may include a damping term to account for the finite lifetime. Engine ADF is required. These polarizabilities are the electric dipole - electric dipole polarizability, the electric dipole - magnetic dipole polarizability, and the the electric dipole - electric quadrupole polarizability. For resonance VROA one should include a finite lifetime.

```
Properties
  VROA Yes/No
End
```

```
Raman
  FreqRange float_list
  IncidentFrequency float
  LifeTime float
End
```

Properties

VROA

Type

Bool

Default value

No

Description

Requests calculation of VROA for vibrational normal modes.

Raman**FreqRange****Type**

Float List

Unit

cm-1

Recurring

True

GUI name

Frequency range

Description

Specifies a frequency range within which all modes will be scanned. 2 numbers: an upper and a lower bound.

IncidentFrequency**Type**

Float

Default value

0.0

Unit

eV

Description

Frequency of incident light.

LifeTime**Type**

Float

Default value

0.0

Unit

hartree

Description

Specify the resonance peak width (damping) in Hartree units. Typically the lifetime of the excited states is approximated with a common phenomenological damping parameter. Values are best obtained by fitting absorption data for the molecule, however, the values do not vary a lot between similar molecules, so it is not hard to estimate values. A typical value is 0.004 Hartree.

6.5.1 Engine ADF

In the ADF engine a method is implemented to calculate both on- and off-resonance vibrational Raman optical activities (VROAs) of molecules using time-dependent density functional theory, see Ref.¹⁴. This is an extension of a method to calculate the normal VROA by including a finite lifetime of the electronic excited states in all calculated properties. The method is based on a short-time approximation to Raman scattering and is, in the off-resonance case, identical to the standard theory of Placzek. The normal and resonance VROA spectra are calculated from geometric derivatives of the different generalized polarizabilities obtained using linear response theory which includes a damping term to account for the finite lifetime. Gauge-origin independent results for normal VROA have been ensured using either the modified-velocity gauge or gauge-included atomic orbitals. In ADF2016 the velocity gauge tensors required for the calculation of VROA are now correctly calculated with the life time damping parameter. With these complex tensors fixed, resonance VROA intensities are now origin invariant in the velocity gauge, see also Ref.¹⁵.

6.6 VCD: Vibrational Circular Dichroism

Vibrational circular dichroism (VCD) is the differential absorption of left and right circularly polarized infrared light by vibrating molecules. Most engines can be used to calculate VCD with the approximate Atomic polar tensor (APT) model. Engine ADF is required for the more accurate analytical VCD.

```
Properties
  VCD Yes/No
End
```

Properties

VCD

Type

Bool

Default value

No

Description

Requests calculation of VCD for vibrational normal modes.

VCDtools (page 454) is a program that can be used to do an analysis of the VCD spectrum. *VCDtools* (page 454) can be used with the AMS-GUI module AMSspectra.

6.6.1 Atomic polar tensor (APT) model

In the so-called atomic polar tensor (APT) model the atomic axial tensors (AATs) can be calculated from electric dipole gradients. Note that the APT model may not be very reliable for predicting VCD bands and its implementation should not be blindly applied beyond a quick assessment. Results using the engine DFTB can be found in¹⁶.

For the engines BAND and DFTB only the APT model can be used. In case of the engine ADF the default is to calculate the VCD analytically, see next section. One can calculate VCD using the APT model with the ADF engine if one includes:

¹⁴ L. Jensen, J. Autschbach, M. Krykunov, and G.C. Schatz, *Resonance vibrational Raman optical activity: A time-dependent density functional theory approach*, *Journal of Chemical Physics* 127, 134101 (2007) (<https://doi.org/10.1063/1.2768533>)

¹⁵ D.V. Chulhai and L. Jensen, *Simulating Surface-Enhanced Raman Optical Activity Using Atomistic Electrodynamics-Quantum Mechanical Models*, *Journal of Physical Chemistry A* 118, 9069 (2014) (<https://doi.org/10.1021/jp502107f>)

¹⁶ T.Q. Teodoro, M.A.J. Koenis, R. Rüger, S.E. Galembeck, W.J. Buma, V.P. Nicu, L. Visscher, *Use of Density Functional Based Tight Binding Methods in Vibrational Circular Dichroism*, *Journal of Physical Chemistry A* 122, 9435 (2018) (<https://doi.org/10.1021/acs.jpca.8b08218>)

```
NormalModes
  Hessian Numerical
End
```

6.6.2 Analytical VCD in ADF

In the ADF engine the VCD intensities are calculated using Stephens' equations for VCD. For the calculation of the atomic axial tensors (AATs), analytical derivatives techniques and London atomic orbitals (the so called GIAO) are employed. As a result the calculated rotational strengths are origin independent, and therefore the common origin gauge is used¹⁷.

New in AMS2020 is that one can calculate analytical VCD also for open-shell systems in a spin-unrestricted calculation.

Calculation of the AATs requires an analytical frequencies calculation. This limits the choice of functionals that can be used for VCD calculations.

The accuracy of the vibrational rotational strengths are determined by the accuracy of the harmonic force field, atomic polar tensors (APT) and AATs. The most critical parameter being the harmonic force field. Thus, for a fair comparison with experimental data, accurate geometries and functionals that yield accurate force fields (e.g. BP86, OLYP, etc) should be used. Our tests showed that the BP86 functional in combination with TZP basis sets is always a safe choice. For a comparison of VCD spectra calculated with various functionals (e.g BP86, OLYP, BLYP, B3PW91 and B3LYP) see [Page 286, 17](#). Regarding the geometries, we recommend the following strict settings, 10^{-4} for the geometry convergence of the gradients, and BeckeGrid quality good. The default settings should be used for the calculation of the frequencies.

By default, only the vibrational rotational strengths are printed in the AMS output file. For a deeper insight regarding the origin of the VCD intensity of a given normal mode one can use the auxiliary program *VCDtools* (page 454). *VCDtools* (page 454) can be used with the ADF-GUI module ADFspectra.

6.7 Vibrational Polarizabilities

If IR intensities are available it is possible to calculate the pure vibrational component for static electric polarizabilities¹⁸ of the system. The tensor can be expressed as a sum-over-state that runs over all normal vibrational modes j :

$$\alpha_{\alpha,\beta}^{vib} = \frac{2}{\hbar} \sum_j \frac{1}{\omega_j} \langle 0 | \mu_\alpha | j \rangle \langle j | \mu_\beta | 0 \rangle$$

Where Q_j are the normal mode coordinates, ω_j are the normal modes angular frequencies, and the bracket notation is used for the system's vibrational states. The full polarizability tensor is printed in atomic units along with the isotropic average of the tensor.

¹⁷ V.P. Nicu, J. Neugebauer, S.K. Wolff, and E.J. Baerends, *A vibrational circular dichroism implementation within a Slater-type-orbital based density functional framework and its application to hexa- and hepta-helicenes*, *Theoretical Chemical Accounts* 119, 245 (2008) (<https://doi.org/10.1007/s00214-006-0234-x>)

¹⁸

M. J. Cohen, A. Willetts, R. D. Amos, and N. C. Handy, *Vibrational contributions to static polarizabilities and hyperpolarizabilities*, *Journal of Chemical Physics* 100, 4467 (1994) (<https://doi.org/10.1063/1.474503>)

VIBRATIONALLY RESOLVED ELECTRONIC SPECTRA

7.1 AH-FC: Adiabatic Hessian Franck-Condon

Electronic spectra, such as absorption, emission, phosphorescence, and ionization, may contain a vibrational structure.

In the Adiabatic Hessian Franck-Condon (AH-FC) method one needs to do a frequency calculation both at the ground state as well as the excited state of interest. The model makes the following assumptions:

- 1. It employs the adiabatic (Born-Oppenheimer) approximation and treats the nuclei as moving in an effective potential defined by the electronic configuration.
- 2. It works at the Franck-Condon (FC) point and assumes that the transition occurs at the ground state equilibrium structure for absorption and at the excited state equilibrium structure for emission.
- 3. It applies the harmonic approximation to both the ground and excited state potential energy surfaces.

Franck-Condon factors can be calculated for the transition between the two electronic states, which can be done with the *FCF Module* (page 287), described below. These Franck-Condon factor can then be used to predict the relative intensities of absorption or emission lines in the electronic spectra. Note that the Herzberg-Teller effect is not taken into account.

In the vertical gradient Franck-Condon (VG-FC) method some extra assumptions are made compared to the AH-FC method. This approach is particularly effective for large molecules as it shows linear scaling with the number of normal modes.

- See the documentation of the *VG-FC method* (page 297).

7.1.1 FCF module: Franck-Condon Factors

fcf is an auxiliary program which can be used to calculate Franck-Condon factors from two vibrational mode calculations¹. The program assumes that the starting vibrational wavefunction is in the ground state, as is the case for most spectroscopic experiments performed at room temperature.

fcf requires an ASCII input file where the user specifies the .rkf files from two vibrational mode calculations, carried out for two different electronic, spin or charge states of the same molecule. These calculations can be either numerical or analytical.

fcf produces a (binary) KF file TAPE61, which can be inspected using the KF utilities. Furthermore, *fcf* writes the frequencies, vibrational displacements and electron-phonon couplings for both states too the standard output, including any error messages.

In AMS2022 the algorithm for the calculation of the Franck-Condon factors has been changed in order to improve the precision, increase the speed, and lower the memory requirements.

¹ J.S. Seldenthuis, H.S.J. van der Zant, M.A. Ratner and J.M. Thijssen, *Vibrational Excitations in Weakly Coupled Single-Molecule Junctions: A Computational Analysis*, ACS Nano 2, 1445 (2008) (<https://doi.org/10.1021/nn800170h>)

Theory

Franck-Condon factors are the squares of the overlap integrals of vibrational wave functions. Given a transition between two electronic, spin or charge states, the Franck-Condon factors represent the probabilities for accompanying vibrational transitions. As such, they can be used to predict the relative intensities of absorption or emission lines in spectroscopy or excitation lines in transport measurements.

When a molecule makes a transition to another state, the equilibrium position of the nuclei changes, and this will give rise to vibrations. To determine which vibrational modes will be active, we first have to express the displacement of the nuclei in the normal modes:

$$\mathbf{k} = \mathbf{L}'^T \mathbf{m}^{1/2} (\mathbf{B}_0 \mathbf{x}_0 - \mathbf{x}'_0)$$

Here, \mathbf{k} is the displacement vector, \mathbf{L} is the normal mode matrix, \mathbf{m} is a matrix with the mass of the nuclei on the diagonal, \mathbf{B} is the zero-order axis-switching matrix and \mathbf{x}_0 is the equilibrium position of the nuclei. For free molecules, depending on symmetry constraints, the geometries of both states may have been translated and/or rotated with respect to each other. To remove the six translational and rotational degrees of freedom, we can center the equilibrium positions around the center of mass and rotate one of the states to provide maximum overlap. The latter is included with the zero-order axis-switching matrix \mathbf{B} , implemented according to².

When we have obtained the displacement vector, it is trivial to calculate the dimensionless electron-phonon couplings. They are given by:

$$\lambda = (\mathbf{\tilde{\omega}}/2)^{1/2} \mathbf{k}$$

Here, $\mathbf{\tilde{\omega}} = 2\pi\omega/h$ is a vector containing the reduced frequencies.³ The Huang-Rhys factor \mathbf{g} is related to λ as:

$$\mathbf{g} = \lambda^2$$

The reorganization energy per mode is

$$\mathbf{E} = (h/2\pi) * \omega \lambda^2$$

When the displacement vector \mathbf{k} , the reduced frequencies $\mathbf{\tilde{\omega}}$ and $\mathbf{\tilde{\omega}'}$, and the Duschinsky rotation matrix $\mathbf{J} = \mathbf{L}'^T \mathbf{B}_0 \mathbf{L}$ have been obtained, the Franck-Condon factors can be calculated using the two-dimensional array method of Ruhoff and Ratner^{Page 288, 3}.

There is one Franck-Condon factor for every permutation of the vibrational quanta over both states. Since they represent transition probabilities, all Franck-Condon factors of one state which respect to one vibrational state of the other state must sum to one. Since the total number of possible vibrational quanta, and hence the total number of permutations, is infinite, in practice we will calculate the Franck-Condon factors until those sums are sufficiently close to one. Since the number of permutations rapidly increases with increasing number of vibrational quanta, it is generally possible to already stop after the sum is greater than about two thirds. The remaining one third will be distributed over so many Franck-Condon factors that their individual contributions are negligible.

In the limiting case of one vibrational mode, with the same frequency in both states, the expression for the Franck-Condon factors of transitions from the ground vibrational state to an excited vibrational state are given by the familiar expression:

$$|l_{0,n}|^2 = e^{-\lambda^2} \lambda^{2n} / n!$$

² G.M. Sando and K.G. Spears, *Ab Initio Computation of the Duschinsky Mixing of Vibrations and Nonlinear Effects*, *Journal of Physical Chemistry A* 105, 5326 (2001) (<https://doi.org/10.1021/jp004230b>)

³ P.T. Ruhoff and M.A. Ratner, *Algorithms for computing Franck-Condon overlap integrals*, *International Journal of Quantum Chemistry* 77, 383 (2000) ([https://doi.org/10.1002/\(SICI\)1097-461X\(2000\)77:1%3C383::AID-QUA38%3E3.0.CO;2-0](https://doi.org/10.1002/(SICI)1097-461X(2000)77:1%3C383::AID-QUA38%3E3.0.CO;2-0))

Algorithm

The algorithm for the calculation of the FC integrals is based on the work by Santoro et al.⁴ which is here summarized. There are in principle infinite FC factors that should be computed in order to achieve 100% convergence in the spectrum. This number is reduced by dividing the vibrational into classes.

A class is defined based on the number of simultaneously excited vibrational modes, thus a class 3 state has exactly 3 modes which are simultaneously excited, while all the remaining modes have zero quanta. The *fcf* program first evaluates the class 1 e 2 FC up to a fairly large vibrational quantum number and uses these results to infer how much each mode should be excited for the calculation of the integrals involving states of higher classes. We refer to the original paper for the details of the algorithm.

The convergence of the results after each class can be estimated by summing all FC factors, which should converge to 1 in the limit of a complete set. The program is terminated when the maximum allowed class is reached, when there is little gain from one class to the next, or when a sufficient convergence criterion is met. It should be noted that if one is interested in the overall shape of the spectrum then in most cases it isn't necessary to reach a convergence that is close to 100%.

Input

The input for *fcf* is keyword-oriented and is read from the standard input. *fcf* recognizes several keywords, but only the states have to be specified to perform the calculation. All input therefore contains at least the following lines:

```
$AMSBIN/fcf << eor
  STATE1 state1
  STATE2 state2
eor
```

The spectrum is always calculated from the ground vibrational state belonging to the potential energy surface specified by *state1* to the vibronic states belonging to *state2*. Documentation for all keys of *fcf*:

```
$AMSBIN/fcf << eor
  State1 state1
  State2 state2
  Lambda float
  Modes first last
  Rotate [True | False]
  Translate [True | False]
  NumericalQuality ["Basic" | "Normal" | "Good" | "Excellent"]
  Prescreening
    Class1 integer
    Class2 integer
    MaxClass integer
    MaxFCI integer
  End
  Spectrum FreqMin FreqMax NFreq
    LineShape ["Stick" | "Gaussian" | "Lorentzian" ]
    SpcMin float
    SpcMax float
    SpcLen integer
    HWHM float
  End
  Printing
    Verbose
```

(continues on next page)

⁴ F. Santoro, R. Improta, A. Lami, J. Bloino, V. Barone, *Effective method to compute Franck-Condon integrals for optical spectra of large molecules in solution*, J. Chem. Phys. 126, 084509 (2007) (<https://doi.org/10.1063/1.2437197>)

(continued from previous page)

```

FCFThresh float
End
eor

```

Convergence**Type**

Float

Description

Minimum fraction of the FC factors sum to be calculated

Lambda**Type**

Float

Default value

0.0

GUI name

Minimum coupling

Description

Optional. The minimum value of the electron-phonon coupling for a mode to be taken into account in the calculation. Together with the MODES option, this provides a way to significantly reduce the total number of Franck-Condon factors. As with the MODES option, always check if the results do not change too much.

Modes**Type**

Integer List

Default value

[0, 0]

Description

Optional. The first and last mode to be taken into account in the calculation. By default, all modes are taken into account. This option can be used to effectively specify an energy range for the Franck-Condon factors. When using this option, always check if the results (electron-phonon couplings, ground state to ground overlap integral, average sum of Franck-Condon factors, etc.) do not change too much.

NumericalQuality**Type**

Multiple Choice

Default value

Normal

Options

[0-0, Basic, Normal, Good, Excellent]

Description

Set the quality of several technical aspects of an FC calculation, including details for the pre-screening, the amount of computed integrals, the maximum integral class evaluated, as well as the convergence criteria.

Printing

Type

Block

Description

Optional. Printing Options.

FCFThresh**Type**

Float

Default value

0.01

GUI name

FC factor threshold

Description

The minimum value for the printing of a FC factor.

Verbose**Type**

Bool

Default value

No

GUI name

Verbose printing

Description

Increase output verbosity.

Rotate**Type**

Bool

Default value

Yes

Description

Recommended to be used. Rotate the geometries to maximize the overlap of the nuclear coordinates.

Spectrum**Type**

Block

Description

Optional. Controls the details of the spectrum calculation

HWHM**Type**

Float

Default value

100.0

Description

Half-Width at Half-Maximum for the spectral lineshape function

LineShape**Type**

Multiple Choice

Default value

Stick

Options

[Stick, Gaussian, Lorentzian]

Description

Type of lineshape function

SpcLen**Type**

Integer

Default value

2001

Description

Number of points in the spectrum

SpcMax**Type**

Float

Default value

9500.0

Description

Maximum absolute energy difference between the states relative to the 0-0 transition

SpcMin**Type**

Float

Default value

-500.0

Description

Minimum absolute energy difference between the states relative to the 0-0 transition

Type**Type**

Multiple Choice

Default value

Absorption

Options

[Absorption, Emission]

Description

Type of spectrum

State1**Type**

String

Description

Filename of the result files of a numerical or analytical frequency calculation for the first (initial) state.

State2**Type**

String

Description

Filename of the result files of a numerical or analytical frequency calculation for the second (final) state.

Translate**Type**

Bool

Default value

Yes

Description

Recommended to be used. Move the center of mass of both geometries to the origin.

Only a few keys from the .rkf file are used for the calculation of the Franck-Condon factors. Disk space usage can be significantly reduced by extracting just these keys from the .rkf file before further analysis. The following shell script will extract the keys from the KF file specified by the first argument and store them in a new KF file specified by the second argument using the *cpkf* utility:

```
#!/bin/sh
cpkf $1 $2 General Molecule Vibrations
```

Result: TAPE61

After a successful calculation, *fcf* produces a TAPE61 KF file. All results are stored in the Fcf section:

contents of TAPE61	comments
lambda	the minimum value of the electron-phonon coupling
translate, rotate	whether the TRANSLATE and ROTATE options were specified in the input
natoms	number of atoms in the molecule
mass	atomic mass vector (m)
xyz1, xyz2	equilibrium geometries of both states (x ₀ and x ₀ ')
b0	zero-order axis-switching matrix matrix (B ₀)
nmodes	number of vibrational modes with a non-zero frequency
gamma1, gamma2	reduced frequencies of both states (Γ and Γ')
lmat1, lmat2	mass-weighted normal modes of both states (L and L')
jmat	Duschinsky rotation matrix (J)
kvec1, kvec2	displacement vectors for both states (k and k' , kvec1 is used for the calculation of the Franck-Condon factors)
lambda1, lambda2	electron-phonon couplings for both states (λ and λ')
i0	ground state to ground state overlap integral ($I_{0,0}$)

In addition to producing a binary TAPE61 file, *fcf* also writes the frequencies, displacement vectors and electron-phonon couplings for both states to the standard output.

7.1.2 FCF example absorption and fluorescence

In this example it is assumed that the molecule has a singlet ground state S_0 , and the interesting excited state is the lowest singlet excited state S_1 . First one needs to do the ground state geometry optimization, followed by a frequency calculation. For the ground state frequency calculation we will use `AMS_JOBNAME=S0`. Next one needs to do an excited state geometry optimization. Here it is assumed that the lowest singlet excited state S_1 is of interest:

```
AMS_JOBNAME=S1_GEO $AMSBIN/ams <<eor
...
Task GeometryOptimization
Engine ADF
...
Excitation
  Onlysing
  Lowest 1
End
ExcitedGO
  State A 1
  Singlet
End
EndEngine
eor
```

To get the frequencies for this excited state, numerical frequencies need to be calculated, at the optimized geometry of the first excited state:

```
AMS_JOBNAME=S1 $AMSBIN/ams <<eor
...
LoadSystem
  File S1_GEO.results/adf.rkf
End
Task SinglePoint
Properties
  NormalModes True
End
Engine ADF
...
Excitation
  Onlysing
  Lowest 1
End
ExcitedGO
  State A 1
  Singlet
End
EndEngine
eor
```

Next for the absorption spectrum, we look at excitations from the lowest vibrational state of the electronic ground state to the vibrational levels of the first singlet excited state S_1 ($S_1 \leftarrow S_0$), using the *FCF program* (page 287), which calculates the Franck-Condon factors between the vibrational modes of the two electronic states, with input

```
$AMSBIN/fcf << eor
STATE1 S0.results/adf.rkf
STATE2 S1.results/adf.rkf
TRANSLATE
ROTATE
eor
```


See *the description of FCF program* (page 287) for more details.

For the fluorescence spectrum, we look at excitations from the lowest vibrational state of the first singlet excited state S_1 to the vibrational levels of the singlet ground state S_0 ($S_1 \rightarrow S_0$). Input for the *FCF program* (page 287) is in this case:

```
$AMSBIN/fcf << eor
STATE1 S1.results/adf.rkf
STATE2 S0.results/adf.rkf
TRANSLATE
ROTATE
eor
```

Note that the FCF program calculates the spectrum relative to the 0-0 transition. Thus one should add to spectrum calculated with FCF the difference in energy of the lowest vibrational state of the ground state S_0 and the lowest vibrational state of the electronically singlet excited state S_1 .

7.1.3 FCF Example phosphorescence

In this example it is assumed that the molecule has a singlet ground state S_0 , and the interesting excited state is the lowest triplet excited state T_1 . Emission from a triplet state to a singlet state is spin forbidden, however, due to spin-orbit coupling such transitions may occur. In the following we assume that the geometry of the triplet excited state is not influenced much by spin-orbit coupling.

First one needs to do a the ground state geometry optimization, followed by a frequency calculation, using `AMS_JOBNAME=S0`. Next one needs to do an excited state geometry optimization of the lowest triplet excited state, followed by a frequency calculation.

```
AMS_JOBNAME=T1 $AMSBIN/ams <<eor
...
Task GeometryOptimization
Properties
  NormalModes True
End
Engine ADF
  Unrestricted
  SpinPolarization 2.0
EndEngine
eor
```

For the phosphorescence spectrum, we look at excitations from the lowest vibrational state of the first triplet excited state T_1 to the vibrational levels of the singlet ground state S_0 ($T_1 \rightarrow S_0$). Input for the *FCF program* (page 287) is in this case:

```
$AMSBIN/fcf << eor
STATE1 T1.results/adf.rkf
STATE2 S0.results/adf.rkf
TRANSLATE
ROTATE
eor
```

Note that the FCF program calculates the spectrum relative to the 0-0 transition. Thus one should add to spectrum calculated with FCF the difference in energy of the lowest vibrational state of the ground state S_0 and the lowest vibrational state of the electronically triplet excited state T_1 .

Zero field splitting (ZFS) and the radiative rate constants (i.e. radiative phosphorescence lifetimes) could be calculated with spin-orbit coupled ZORA time-dependent density functional theory (ZORA-TDDFT). With the ADF engine spin-

orbit coupling can be treated self-consistently (i.e. non perturbatively) during both the SCF and TDDFT parts of the computation.

An alternative to the use of the unrestricted formalism to calculate the lowest triplet excited state is to use the TDDFT formalism:

```
AMS_JOBNAME=T1_GEO $AMSBIN/ams <<eor
...
Task GeometryOptimization
Engine ADF
  Excitation
    Onlytrip
    Lowest 1
  End
ExcitedGO
  State A 1
  Triplet
End
EndEngine
eor
```

To get the frequencies for this excited state, numerical frequencies need to be calculated, at the optimized geometry of the first excited state.

```
AMS_JOBNAME=T1 $AMSBIN/ams <<eor
...
LoadSystem
  File T1_GEO.results/adf.rkf
End
Task SinglePoint
Properties
  NormalModes True
End
Engine ADF
...
Excitation
  Onlytrip
  Lowest 1
End
ExcitedGO
  State A 1
  Triplet
End
EndEngine
eor
```

7.1.4 Density of States computed with AH-FC method

The FCF module can also be used to evaluate the density of states ρ that enters the expression of the rate constant for a number of transfer phenomena based on Fermi's Golden Rule:

$$\mathbf{k} = \frac{2\pi}{\hbar} J^2 \rho$$

To calculate the transfer rate from a donor molecule to an acceptor molecule it is sufficient to employ the FCF code as described above to evaluate the emission spectrum of the donor and the absorption spectrum of the acceptor. In fact, it

can be shown⁵ that the expression for the density of states can be written as:

$$\rho = \frac{1}{\hbar} \int_{-\infty}^{+\infty} F^D(\omega) F^A(\omega) d\omega$$

Where F denotes the Franck-Condon spectrum calculated by the FCF code, specifically the absorption for the acceptor and the emission for the donor.

The convolution may be easily calculated by using the PLAMS library.

7.2 VG-FC: Vertical Gradient Franck-Condon

Electronic spectra, such as absorption, emission, phosphorescence, and ionization, may contain a vibrational structure.

In the vertical gradient Franck-Condon (VG-FC) method some extra assumptions are made compared to the Adiabatic Hessian Franck-Condon (AH-FC) method. VG-FC is also called the Independent Mode Displaced Harmonic Oscillator (IMDHO) model. The AH-FC model (used in the *FCF Module* (page 287)) makes the following assumptions:

- 1. It employs the adiabatic (Born-Oppenheimer) approximation and treats the nuclei as moving in an effective potential defined by the electronic configuration.
- 2. It works at the Franck-Condon point and assumes that the transition occurs at the ground state equilibrium structure for absorption and at the excited state equilibrium structure for emission.
- 3. It applies the harmonic approximation to both the ground and excited state potential energy surfaces.

The VG-FC model makes following additional assumptions:

- 4. It assumes the excited state PES has the same shape as that of the ground state, but it is displaced from it, i.e. both states have the same normal modes and frequencies but different equilibrium geometries.
- 5. The excited state equilibrium structure is found from a Newton-Raphson step from the ground state geometry using the excited state gradient at this point.

Under these simplifying approximations we can reduce the ingredients necessary for a spectrum calculation to an excited state gradient and a set of ground state normal modes. Furthermore it uses a time-domain description of the absorption cross-section (this introduces no further approximations) so we do not need any explicit Franck-Condon factors as is done in the *FCF Module* (page 287). This approach is particularly effective for large molecules as it shows linear scaling with the number of normal modes. Further details on how to use this method, and the VG-FC model in general can be found below.

The approximations we make remove the need for an excited state geometry optimization and frequency analysis. This means the most expensive step in a typical calculation is actually the ground state frequency analysis. The ability of Mode Tracking to generate a set of approximate normal modes, without performing a full frequency analysis can be applied to the calculation of vibrationally resolved optical spectra. This is exactly what is done in *Vibronic-Structure Tracking* (page 299). On the other hand, the ability of Mode Refinement to refine entire spectral regions at once means this method can be transferred to calculating vibrationally resolved optical spectra from a select number of (vibronically active) approximate normal modes. This idea is applied in *Vibronic-Structure Refinement* (page 300). Under suitable assumptions, the computational cost of calculating an approximation of the vibronic-structure can be reduced significantly through the use of these two methods, allowing for application to large molecules inaccessible to more detailed methods. While they are clearly best suited for these large molecules, they can also serve as an efficient method to obtain quick, first approximations for more moderately sized molecule.

With the VG-FC model one can calculate the 0-0 energy difference between the ground state and excited state. In this VG-FC method the vibrationally resolved emission spectrum will be a mirror image of the vibrationally resolved absorption spectrum, in which the 0-0 transitions are the same.

⁵ X. de Vries *High energy acceptor states strongly enhance exciton transfer between metal organic phosphorescent dyes*, *Nature Communications* 11, 1292 (2020) (<https://doi.org/10.1038/s41467-020-15034-0>)

Note: The VG-FC model uses certain approximations which may not always be valid. Something to look out for that may affect the reliability of VG-FC results are large changes in conformation upon excitation. In this case the normal modes of the excited state may be completely different from those of the ground state, at which point the model is not expected to produce reliable results. Should non-adiabatic effects become important, the BO approximation could potentially break down as well. In these cases the *FCF Module* (page 287) may provide a more suitable alternative to the VG-FC model.

See also:

Tutorials: [Resonance Raman](#), [Vibrationally resolved electronic spectra with ADF](#), [Vibrationally resolved electronic spectra with DFTB](#)

7.2.1 Theory

Here we will introduce the theory behind the model used to compute vibronic couplings. We use the so-called Vertical Gradient Franck-Condon (VG-FC) method, also called Independent Mode Displaced Harmonic Oscillator model (IMDHO). The VG-FC model as used here works as follows: The PES in the ground state is assumed parabolic (the harmonic approximation). Upon excitation, we assume that the only change to the PES that occurs is a shift in origin (i.e. no stretching or rotation of the parabolic well). This means, both the normal modes and their frequencies remain the same in both the ground and excited state. The result is that a spectrum can be calculated using only a single excited state gradient at the optimized ground state geometry. Hence we avoid the expensive excited state geometry optimization and frequency analysis used by methods such as the *FCF Module* (page 287).

Moving to a slightly more detailed discussion, we can start with an expression for the absorption cross-section in terms of a sum over contributions from different Franck-Condon factors, each homogeneously broadened with an assumed Lorentzian of linewidth Γ . For excitation from the vibrational ground state of electronic state m , to electronic state n we get the following expression (Hartree atomic units are used throughout):

$$\sigma_{n \leftarrow m}(\omega) = \frac{4\pi\omega}{3c} |\mu_{mn}|^2 \sum_{l_{n,1}} \dots \sum_{l_{n,N_{modes}}} \frac{\Gamma \prod_{i=1}^{N_{modes}} |\langle l_{n,i} | 0_{m,i} \rangle|^2}{\left(E_{n,0} + \sum_i^{N_{modes}} \omega_i l_{n,i} - E_{m,0} - \omega \right)^2 + \Gamma^2}$$

Here the μ_{mn} is the transition dipole moment and the $\prod_{i=1}^{N_{modes}} |\langle l_{n,i} | 0_{m,i} \rangle|^2$ denote the Franck-Condon factors for excitation from the vibrational ground state to the vibrationally excited state of the excited state PES, with quantum numbers $l_{n,i}$. This sum-over-states formulation is what is also used to compute the spectrum in the *FCF Module* (page 287). While efficient pre-screening and thresholding techniques can make this method applicable to moderately sized molecules, it is still quite expensive for large molecules. We can derive an expression which is equivalent to the one above, but which enjoys significant computational benefits. This is done by writing the cross-section as an integral over time:

$$\sigma_{n \leftarrow m}(\omega) = \frac{4\pi\omega}{3c} |\mu_{mn}|^2 \text{Re} \int_0^\infty \langle i | i_n(t) \rangle e^{i[\omega + E_{m,0}]t} \cdot e^{-\Gamma t} dt$$

Here, $\langle i | i_n(t) \rangle$ denotes the overlap of the initial state, propagated along the excited state PES with itself at time $t = 0$. This expression is still completely general, however, using the *assumptions of the VG-FC model* (page 286), this overlap admits a simple closed form expression:

$$\langle i | i_n(t) \rangle = \prod_{j=1}^{N_{modes}} \exp \left[-\frac{\Delta_{n,j}^2}{2} (1 - e^{-i\omega_j t}) \right] e^{-iE_{n,0}t}$$

This expression can be derived by different methods and admits a few interpretations, the details of which we will not go into here. The interested reader is referred to reference¹. Instead we will focus on its numerical implementation. The

¹ J.R. Reimers, K.R. Wilson and E.J. Heller, *Complex time dependent wave packet technique for thermal equilibrium systems: Electronic spectra* The Journal of Chemical Physics 79, 4749 (1983) (<https://doi.org/10.1063/1.445618>)

great computational advantage of this expression is that it scales only weakly with the number of degrees of freedom. Compare this with the sum-over-states formalism above, where the number of FCF's needed to have a converged sum proliferates for large molecules and the advantage becomes clear.

The $\Delta_{n,j}$ appearing in the expression are the dimensionless normal mode displacements at the equilibrium geometry in state n , compared to state m along normal mode j which has angular frequency ω_j . They represent the change in origin of the PES along a particular normal mode. These are the parameters which control the shape of the spectrum. Under the assumptions we listed above they can be calculated from the excited state energy gradient at the ground state equilibrium structure as:

$$\Delta_j = \frac{q_j^m \cdot \nabla_m E}{\omega_j^{3/2}}$$

Where we have suppressed the n subscript. The gradient is with respect to mass weighted coordinates and the q_j^m denote the mass-weighted normal modes.

Essentially, we use the (assumed) parabolic shape of the PES in the harmonic approximation along each of the normal modes to determine the displacement from the gradient projection along that mode. An important consequence of the simplified VG-FC model is that the Franck-Condon factors can be expressed in closed form:

$$|\langle l_i | 0_i \rangle|^2 = \frac{\left(\frac{\Delta_i}{2}\right)^2}{l_i!} e^{-\frac{\Delta_i^2}{2}}$$

Modes with the largest displacement are then expected to also have the largest impact on the structure of the vibrational progression in the spectrum. This gives us an obvious candidate for a tracking method to be used with the Mode Tracking protocol. This approach is taken in the *Vibronic-Structure Tracking* (page 299) module.

Theory: Vibronic-Structure Tracking

Vibronic-structure tracking (VST) is a method for obtaining quick and approximate vibrationally resolved optical spectra for larger sized molecules. It is based on the mode-tracking algorithm and works by tracking those modes that are expected to have the largest impact on the vibronic-structure of the spectrum. Its AMS implementation follows that in reference [Page 298, 1](#).

The modes that are tracked by the mode tracking routine are used to generate a UV/vis spectrum. This is done using the Vertical Gradient Franck-Condon (VG-FC) method, also called the Independent Mode Displaced Harmonic Oscillator (IMDHO) model as described on the *Vibronic-Structure* (page 297) page. This model reduces the necessary TD-DFT calculations to a single excited state gradient performed at the ground state equilibrium geometry. At this point it is typically the ground state frequency analysis which becomes the bottleneck for large molecules. Through the use of Mode Tracking we can circumvent this step and calculate only those modes which will significantly impact the vibronic fine structure. This yields a very cheap and efficient method for obtaining vibrationally resolved optical spectra.

The use of Mode Tracking within the context of vibronic-structure calculations proceeds as follows: from an initial guess for a normal mode, we construct a new basis vector which we hope will improve the quality of our initial guess. We then iteratively expand the basis by refining those modes which have the largest impact on the vibronic fine structure. We keep iterating until we deem the spectrum to be converged. We'll address the technical details of this procedure in this section.

Let us first consider how we can gauge which modes will have a large impact on our spectrum. Over on the *Vibronic-Structure* (page 297) page, we showed that according to the VG-FC model, the Franck-Condon factors can be written as:

$$|\langle l_i | 0_i \rangle|^2 = \frac{\left(\frac{\Delta_i}{2}\right)^2}{l_i!} e^{-\frac{\Delta_i^2}{2}}$$

This expression implies modes with large displacements Δ_j will have the largest Franck-Condon factors. Thus, we expect that modes with large oscillator displacements will contribute most strongly to the vibronic fine structure of the spectrum.

This suggests a tracking method in addition to those described in the [Mode Tracking](#) (page 257) documentation. Namely, to track the mode with the largest Δ_j . This is the only setting for the `TrackingMethod` supported by Vibronic-Structure Tracking and is automatically set when choosing Vibronic-Structure Tracking as the Vibrational Analysis type. As a reminder, the oscillator displacements can be obtained from the assumed parabolic shape of the excited state PES and the known excited state gradient:

$$\Delta_j = \frac{q_j^m \cdot \nabla_m E}{\omega_j^{3/2}}$$

This expression for our normal mode displacements suggests that a reasonable guess for a hypothetical normal mode with a large effect on the spectrum is given by the normalized excited state gradient, as this maximizes the projection.

The choice of preconditioner is of course an important one. While better preconditioners such as the Jacobi-Davidson method can lead to fast and tightly directed convergence of the modes, this may not be ideal in the case of Vibronic-Structure Tracking, as we are not necessarily focused on obtaining converged modes but rather obtaining a converged spectrum. By default the method does not use any preconditioner for generating normal modes from the residual vectors to allow the procedure to more freely explore the entire space of modes. This means that the new basis vector produced on each iteration is the (normalized) residual vector of that iterations selected mode. Note that this is different from using the Davidson method with an identity matrix as guess for the Hessian. The user is still able to use any of the pre-conditioning options provided by the stand-alone mode tracking method, however we suggest the user uses the default `I` (for identity) as the `UpdateMethod`. This also conveniently means no approximate Hessian is required.

It is important to choose a suitable convergence criterion. When using Vibronic-Structure Tracking, the aim is to get an approximation of the progression in the spectrum. The convergence of the modes then takes a backseat to the convergence of the spectrum. Although the two usually go hand in hand, requiring that all modes be converged in the usual mode-tracking sense is likely to be far too restrictive. Thus the convergence criterion used for VST is the following: at every iteration, the previous iteration's spectrum is subtracted from the new one. The absolute difference is then integrated on the requested frequency range. Convergence is achieved once the so-obtained number drops below a pre-set convergence threshold. The result is that it may not be the case that all modes are exact normal modes yet, however further refinement will have a limited effect on the resulting spectrum, for example due to these modes having very small Franck-Condon factors.

Convergence according to this criterion is affected by the other options that the user chooses, see the section on how the adiabatic excitation energy is calculated.

Theory: Vibronic-Structure Refinement

Vibronic-Structure Refinement takes the idea of Mode Refinement, and applies it to the calculation of vibrationally resolved optical spectra. Under the assumptions of the Vibronic-Structure application, the most time-consuming step becomes the calculation of the ground state normal modes. However, we may not be interested in all normal modes that contribute to the Vibronic fine structure of a spectrum. By using Mode Refinement we can limit ourselves to the spectral region of interest, or to modes which we expect to have the largest impact on the fine structure. The details of the Mode Refinement protocol can be found on the relevant [documentation page](#) (page 255).

Theory: Adiabatic excitation energy

The last aspect we will comment on here is the calculation of the adiabatic excitation energy. The number we get from an excited state calculation is the vertical excitation energy (ΔE_{vert}), this number is generally unobservable in spectra. The more interesting quantity is the adiabatic excitation energy (ΔE_{ad}). The former is the difference in energy between the two PES'es at the ground state geometry, the latter is the difference in energy between the ground and excited state vibrational ground states. Since we assumed both PES'es to have the same frequencies, the zero-point energy of the two states are the same and hence, the adiabatic energy difference is the same as the 0-0 energy difference (ΔE_{0-0}). This is simply the difference between the bottom of the two PES'es. It can be reconstructed from the excited state gradients

projected onto our normal modes, as well as their frequencies. For a molecule with N modes we obtain:

$$\Delta E_{ad} = \Delta E_{vert} - \sum_j^{N_{modes}} \frac{1}{2} \omega_j \Delta_j^2$$

Where the sum represents the reorganization energy stored in the excited oscillator modes. Depending on the desired information, one can use two different representations of the spectrum. The first is to simply plot the spectrum against an energy range with the ground state energy as an offset. This would represent a true absorption spectrum that can be compared directly with experiment. However, the energy difference $\Delta E_{vert} - \Delta E_{ad}$ in the VG-FC model is reconstructed only from the ground state equilibrium structure and can thus be somewhat unreliable. If the user is primarily interested in the shape of the spectrum, using the 0-0 excitation energy as an offset may be a more suitable choice. The latter is the default setting in AMS, but the method supports both options.

Note that in case of a Vibronic-structure tracking (VST) calculation we only have access to a small set of approximate normal modes, we will have an approximation of ΔE_{ad} that changes from iteration to iteration as new normal modes are introduced and the old normal modes are refined. If a spectrum relative of the 0-0 energy is requested, the shift caused by the iterative convergence of ΔE_{ad} is not taken into account, although it will still have some effect on the spectrum as it also appears in the integral that defines the spectrum. If one is interested in a converged approximation of ΔE_{0-0} it may be more reliable to provide an absolute frequency range for the spectrum as the shift in the peak locations will then also affect convergence.

7.2.2 Input: Vibronic-Structure all modes

As mentioned, we need both a set of ground state normal modes as well as an excited state gradient. All the normal modes are calculated using the AMS engine that was selected, or, in case the user has a pre-calculated set of normal modes, these can be read from a .rkf file using the `ModeFile` key in the `NormalModes` sub-block. In this latter case, the engine is not used. One can then use the keys in the `ModeSelect` block to filter out specific modes, or, simply select all modes using the `Full` key in this block. For further details on how to use the `Mode Select` block for more specialized selection options, see the [Mode Select](#) (page 266) documentation. As for the excited state information, this is passed to the application via the `ExcitationSettings` block. Additionally there are (optional) settings related to the appearance of the spectrum under the `AbsorptionSpectrum` block. For an overview of the input options see the list of keys at the end of this page.

For completeness we provide an example of what the user input may look like:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type VibronicStructure
  NormalModes
    # Select all modes present in the .rkf file
    ModeSelect
      Full True
    End
  End
  ExcitationSettings
    ExcitationInfo File
    ExcitationFile ./your_excitation.t21
    Singlet
    B1.u 2
  End
End
AbsorptionSpectrum
  LineWidth 100.0
  AbsorptionRange -500.0 4000.0
End
End
```


7.2.3 Input: Vibronic-Structure Tracking

VST is notably the only Vibrational Analysis task that does not support the `NormalModes` block as no initial normal modes are necessary. The excited state gradient fills this role instead. Furthermore, the recommended default of using the identity matrix as a preconditioner means that no approximate Hessian is required either.

One thing to note regarding the Mode Tracking settings is that VST features a convergence criterion not present in standard Mode Tracking. The tolerance for the convergence of the spectrum is set using the `ToleranceForSpectrum` keyword in the `ModeTracking` block. The default is set to 0.01 and should be sufficient for most purposes, but *restarting* (page 302) with a lower value may improve the reliability of the convergence.

Note: The usual Mode Tracking tolerances are still present in addition to the spectral tolerance mentioned above. This is so we do not track modes that have already converged. Instead, once a mode is converged in the usual mode tracking sense, we switch to the next mode that satisfies our tracking criterion. The defaults should normally apply, but may be loosened a bit to allow for more free subspace exploration.

A typical Vibronic-Structure Tracking run may be setup as follows:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type VibronicStructureTracking
  # Select our excited state energy+gradient from a previous calculation
  ExcitationSettings
    ExcitationInfo File
    ExcitationFile ./your_excitation.t21
  End
  # Tuning how our spectrum will look
  AbsorptionSpectrum
    LineWidth 250.0
    AbsorptionRange -500.0 6000.0
  End
  ModeTracking
    UpdateMethod I
  End
End
```

Input: Restarting VST

The Vibronic-Structure Tracking task is fully restartable from a previously completed VST run. The main reason why it may be useful to perform a restart is related to the way the spectrum is generated. Most modules/applications in AMS that calculate a spectrum usually produce a stick spectrum. This stick spectrum is then convoluted against either a gaussian or lorentzian of specified width or area. This allows one to tweak the homogeneous broadening post-calculation to improve agreement with experiments. VST operates differently, in that the Lorentzian linewidth is specified at the start of the calculation as an essential input parameter. If upon convergence it becomes clear that a different linewidth would have been better suited for the spectrum at hand, one can restart the calculation with this new linewidth. The program then computes a new spectrum using these new settings and repeats the last iteration of the previous run. The latter is done because the linewidth directly influences the convergence (larger linewidths tend to converge more easily as the features are not as sharply resolved). If the spectrum does not change after the first iteration the program terminates as usual and the new spectrum is now available to the user. However, if this additional iteration does not pass the convergence test, the program keeps iterating until the spectrum converges for this new linewidth.

In addition to changing the linewidth post-calculation, the user is also free to change the spectral range that is to be computed as well as its resolution through the `FrequencyGridPoints` keywords. Alternatively, the tolerance used may be lowered if tighter convergence is required. Importantly however, none of the previous run's input settings are

retrieved upon restart, so any settings the user doesn't want to change have to be same in the new run script as in the old. The easiest way to restart is to simply re-use the previous run's runscript under a new name and then changing the relevant settings.

The restart functionality is controlled by the `RestartPath` keyword. One simply states the path to a previous calculation's .rkf file from which the relevant data is to be read. The .rkf file of any VST runs contains a section labeled `VSTRestart` which contains the data needed to restart from that run. Restarting is then a simple one-liner in the `VibrationalAnalysis` block:

```
VSTRestartFile ./previous_run.results/ams.rkf
```

7.2.4 Input: Vibronic-Structure Refinement

Vibronic-Structure Refinement (VSR) assumes that the modes relevant to the vibronic-structure are, at least in an approximate sense, known a priori. The VSR method takes these approximate modes and refines them via the Mode Refinement algorithm. The approximate normal modes should thereby span a subspace containing the exact normal modes and, the more similar they are to the exact mode, the better they allow to reproduce the spectrum of a full ground state frequency analysis. Naturally, the refinement becomes more accurate with increasing dimensionality of the spanned subspace, although this increases the computational costs. In any case, the subspace needs to contain those modes that are most relevant for the vibronic progression for the entire approach to work, which leads to the question, how do we select such modes?

This is done using familiar `ModeSelect` block from the mode selective analysis tasks. While all the usual options are available, for which further details can be found on the [Vibrational Analysis](#) (page 266) page, there are two selection methods that are geared specifically towards VSR. Within the VG-FC model, the parameters defining the effect a mode will have on the spectrum are the oscillator displacements Δ_j . We can determine these values from the projection of the approximate normal modes onto the gradient in combination with their respective frequencies. We are then interested in those modes with the largest displacement. To select these, two options are available:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type VibronicStructureRefinement
  NormalModes
    ModeSelect
      LargestDisplacement integer
      DisplacementBound float
      ...
    End
  ...
End
...
```

LargestDisplacement integer

sets an integer value N , to select the N modes with the largest displacements as calculated within the VG-FC model. This method limits the total number of modes we refine and thus places an upper bound on the computational cost required by the method, but it does not ensure that all relevant modes have been selected. In particular there may still be modes with large displacements that have been omitted and as such peaks or progressions may be missing from the spectrum.

DisplacementBound float

Alternatively this selects all approximate modes with a displacement greater than the supplied lower bound. An appropriate choice for this value ensures all modes which visibly affect the spectrum (at least at the lower level of theory used to produce the approximate modes) are included in the subspace basis used for Mode Refinement. The displacements Δ_j are dimensionless and the largest ones will typically be of order one. Based on this, a

value of 0.01 for this parameter will generally select all relevant modes. This of course comes at the trade-off that for general molecules it is not known how many such modes there will be and the computational cost may be larger than expected. But, if one is not discouraged by this, this will of course yield the most accurate results. Values up to 0.05 may still provide reasonable results while minimizing computation times. Using larger linewidths (AbsorptionSpectrum%LineWidth) for the homogeneous broadening can help to correct for the non-resolved modes.

The calculation setup for Vibronic-Structure Refinement is essentially the same as that for Vibronic-Structure, the only difference being that the modes that we provide will first be refined, before a spectrum is computed. A detailed description is given on the *Vibronic-Structure* (page 297) page. A typical VSR calculation can be setup as follows:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type VibronicStructureRefinement
  # Select our normal modes from a previous calculation
  NormalModes
    ModeFile ams.rkf
    ModeSelect
      # select all modes with VG-FC displacements over 0.01
      DisplacementBound 0.01
    End
  End
  # Select our excited state energy+gradient from a previous calculation
  ExcitationSettings
    ExcitationInfo File
    ExcitationFile ./your_excitation.t21
    # Select second singlet excitation with symmetry label A
    Singlet
      A 2
    End
  End
  # Tuning how our spectrum will look
  AbsorptionSpectrum
    LineWidth 250.0
    AbsorptionRange -500.0 6000.0
  End
End
```

7.2.5 Input: Excited State

Both the excited state gradient at the ground state equilibrium structure as well as the vertical excitation energy are necessary to obtain a spectrum. These can be provided by most TD-DFT programs, including ADF. Unfortunately, the AMS driver does not yet support computing these properties on the fly. As a result they are currently required as user input. One can use ADF or DFTB to compute them and then either read them from the produced output file, or copy and paste them to an inline input block.

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ...
  ExcitationSettings
    ExcitationInputFormat [File | Inline]
    ExcitationFile string
    EnergyInline float
    GradientInline # Non-standard block. See details.
    ...
  End
```

(continues on next page)

(continued from previous page)

```

Singlet # Non-standard block. See details.
...
End
Triplet # Non-standard block. See details.
...
End
End
End
End

```

ExcitationInputFormat [File | Inline]

this keyword is used to specify whether the gradient and energy are to be read from a kf file or if one chooses to use the inline block for them. Currently, no TD-DFT engine is implemented in the AMS driver. Furthermore, ADF uses the old tape21 format for its output which is quite different from the more streamlined AMS format. The `inline` option allows one to bypass any possible confusion here. The `File` option requires the specification of the specific excitation to read, which we will discuss below.

GradientInline

is the block where the gradient can be specified if the inline option is selected. The format can be either a N by 3 block (N rows, 3 columns) or a 3N long column vector if one happens to have a gradient in this format. The gradient values should be in the unit Hartree/Bohr.

EnergyInline float

is used to specify the excitation energy. This will just be a single floating point number (in Hartree).

The resulting input would look like:

```

Task VibrationalAnalysis
VibrationalAnalysis
  Type ...
  ...
  ExcitationSettings
    ExcitationInfo Inline
    # Excited state gradient for transhexatriene 14 atoms x 3 coordinates (to be_
    ↪provided in Hartree/Bohr)
    GradientInline
      -0.03786125      0.01786798      0.00003833
      0.05322148     -0.00798712     -0.00004152
      -0.06658803      0.01373495      0.00004727
      0.06656379     -0.01374825      0.00002398
      -0.05318451      0.00799875     -0.00002097
      0.03783718     -0.01786722      0.00001362
      0.00382226      0.00327391      0.00001046
      -0.00046176     -0.00499971     -0.00003583
      0.00014312      0.00534412     -0.00001335
      0.00011081     -0.00558254     -0.00002107
      -0.00011074      0.00558350     -0.00000933
      -0.00014517     -0.00534330      0.00000839
      -0.00381513     -0.00327314     -0.00000248
      0.00046796      0.00499807      0.00000250
    End
    # The vertical excitation energy is simply a float (to be provided in Hartree)
    EnergyInline 0.17062882
  End
End

```

ExcitationFile string

If `ExcitationInputFormat` is set to `File` one also has to set the `ExcitationFile` keyword to provide

a path to the file from which the info should be read.

Singlet or Triplet

In case the excited state information should be read directly from a file, one should specify which excitation AMS should read. This is done analogously to how this is done for excited state gradients in ADF. One chooses either the `Singlet` or `Triplet` blocks (for singlet-singlet or singlet-triplet excitations respectively). Once such a block is chosen, a line containing the symmetry label, followed by the number of the excitation of that symmetry. As for the symmetry labels, the notational convention used can be found in the [ADF manual appendix](#) on symmetry labels. If symmetry was disabled, either explicitly by the user or when using TD-DFTB which does not support symmetry, when calculating the excited state properties, the symmetry label A should be used. Vibronic-structure calculations are only supported for one excitation at a time.

The resulting input block looks something like this:

```
ExcitationSettings
  ExcitationInputFormat File
  ExcitationFile ./ExcitedState.results/myengine.rkf
  Singlet
    A 1
  End
End
```

7.2.6 Input: Producing the spectrum

To go from normal modes to an actual spectrum, we have to solve the integral that we introduced in the first section, the details of which will be discussed here.

For long times, the integrand is dominated by the exponential damping term. We can define a suitable cut-off by demanding that this damping term be smaller than some threshold. As mentioned, the integral is evaluated using a Chebyshev quadrature. The number of Chebyshev nodes used for the integral can then be determined from this integration limit and the time step size. This time step size should be large enough to accurately represent the highest frequencies that are present in the highly oscillatory integrand. The Nyquist frequency can be used as a guide here: the highest frequency should be represented by at least two samples each period. Note however, that in principle there is no upper bound to the frequencies present in the integrand due to the fact that modes can be excited to arbitrarily high energy levels. Fortunately, the Franck-Condon factors will quickly act to dampen these high frequencies, restricting their significance. The user may either supply a time step size directly or leave this up to the program to determine. If the latter is chosen the highest relevant frequency is estimated by assuming that oscillator excitations are relevant up to about twice the Huang-Rhys parameter (which follow directly from the displacements). This default should in most cases be sufficient to produce a converged integral.

The keywords associated with the different input parameters are collected under the `AbsorptionSpectrum` block. It contains the following keys:

```
Task VibrationalAnalysis
VibrationalAnalysis
  Type ...
  ...
  AbsorptionSpectrum
    AbsorptionRange float_list
    FrequencyGridPoints integer
    LineWidth float
    SpectrumOffset [absolute | relative]
  End
End
```

LineWidth float

sets the lorentzian line width and thus has a large impact on the appearance of the final spectrum. It may be used to

improve the agreement of the computed spectrum with experiment. Another purpose that the linewidth may serve is that some low frequency modes may not be represented very accurately within the VG-FC model. However, these modes do not result in particularly distinct vibronic progressions but rather cause an unspecific broadening of existing peaks. Applying a homogeneous peak broadening by increasing the value of Γ may be an effective (albeit a little ad hoc) way of resolving such issues, should they occur. The default value is 200cm^{-1} but it is recommended to do some experimentation with this value.

AbsorptionRange float_list

keyword specifies the frequency range (in cm^{-1}) that is to be computed by VST.

SpectrumOffset [absolute | relative]

specifies whether the range provided by the `AbsorptionRange` keyword is relative to the 0-0 excitation energy (`Relative`), or relative to the vertical excitation energy (`Absolute`). Due to the limited accuracy of the VG-FC model at predicting ΔE_{0-0} the default is `Relative` with a `AbsorptionRange` of `[-500,4000]`.

FrequencyGridPoints integer

sets the number of points we use on our frequency grid. It is set to 400 by default which generally produces a smooth looking spectrum.

The current implementation of VG-FC vibronic-structure supports only spectra for one excitation at a time. The spectrum is normalized such that the highest peak is equal to 1 in arbitrary units. For this reason the prefactor of the integral is irrelevant and transition dipole moments do not affect the appearance of the spectrum.

DIPOLE MOMENT, POLARIZABILITY, BOND ORDERS, ORBITAL ENERGIES

This page in the AMS manual describes the calculation of the dipole moment, the polarizability, and bond orders.

```
Properties
  DipoleMoment Yes/No
  Polarizability Yes/No
  BondOrders Yes/No
End
```

Note that because these properties are tied to a particular point on the potential energy surface, they are found on the *engine output files* (page 19). Note also that the properties are not always calculated in every PES point that the AMS driver visits during a calculation. By default they are only calculated in *special* PES points, where the definition of special depends on the *task* (page 47) AMS is performing: For a *geometry optimization* (page 49) properties would for example only be calculated at the final, converged geometry. This behavior can often be modified by keywords special to the particular running task.

8.1 Charges, Dipole Moment, Polarizability

```
Properties
  Charges Yes/No
  DipoleMoment Yes/No
  DipoleGradients Yes/No
  Polarizability Yes/No
End
```

Properties

Charges

Type

Bool

Default value

No

Description

Requests the engine to calculate the atomic charges.

DipoleMoment

Type

Bool

Default value

No

Description

Requests the engine to calculate the electric dipole moment of the molecule. This can only be requested for non-periodic systems.

DipoleGradients**Type**

Bool

Default value

No

Description

Requests the engine to calculate the nuclear gradients of the electric dipole moment of the molecule. This can only be requested for non-periodic systems.

Polarizability**Type**

Bool

Default value

No

Description

Requests the engine to calculate the polarizability tensor of the system.

8.2 Bond orders & Molecule detection

Many engines can determine bond orders between atoms. For engines based on force fields, these might just be the bond orders used internally by the force field, while for quantum mechanical engines the bond orders are usually determined by analyzing the results of the quantum mechanical calculation, e.g. the electronic density. We refer users to the manuals of the respective engine for details.

If bond orders are requested but the engine does not implement a method for determining bond orders, a bond guessing algorithm will be used instead (see also [BondOrders block](#) (page 311)). The bond guessing algorithm (which is also used by the graphical user interface to guess bonds) uses inter-atomic distances and atomic valences to guess the bond orders.

You can request bond orders to be calculated in the `Properties` input block:

```
Properties
  BondOrders Yes/No
End
```

Properties**BondOrders****Type**

Bool

Default value

No

Description

Requests the engine to calculate bond orders.

For MM engines these might just be the defined bond orders that go into the force-field, while for QM engines, this might trigger a bond order analysis based on the electronic structure. For engines that do not have a bond order analysis method, a bond guessing algorithm will be used. See also the input options in the `BondOrders` block.

More options can be specified in the `BondOrders` block:

```
BondOrders
  Method [Engine | Guess | EngineWithGuessFallback]
End
```

BondOrders

Method

Type

Multiple Choice

Default value

EngineWithGuessFallback

Options

[Engine, Guess, EngineWithGuessFallback]

Description

How to compute the bond orders when they are requested via the 'Properties%BondOrders' key.

'Engine': let the engine compute the bond orders. The specific method used to compute the bond orders depends on the engine selected, and it may be configurable in the engine's input. Note: the calculation may stop if the engine cannot compute bond orders.

'Guess': Use a bond guessing algorithm based on the system's geometry. This is the same algorithm that is used by the Graphical User Interface to guess bonds.

'EngineWithGuessFallback': let the engine compute the bond orders (same as in 'Engine' option) but if the engine did not produce any bond orders, use the bond guessing algorithm as a fallback option.

Based on the bond orders, the AMS driver can analyze the atomic connectivity graph in order to determine which sets of atoms together constitute molecules. This allows for example to monitor the changes in molecular composition during a reactive molecular dynamics calculation. For *molecular dynamics* (page 101) calculations this option is enabled by default. For other *tasks* (page 47) the molecular analysis can explicitly be requested in the `Properties` block.

```
Properties
  Molecules Yes/No
End
```

Properties

Molecules

Type

Bool

Default value

No

Description

Requests an analysis of the molecular components of a system, based on the bond orders calculated by the engine.

Details of the molecule detection are configured in a dedicated block:

```
Molecules
  AdsorptionSupportRegion string
  BondOrderCutoff float
End
```

Molecules

Type

Block

Description

Configures details of the molecular composition analysis enabled by the `Properties%Molecules` block.

AdsorptionSupportRegion

Type

String

GUI name

Adsorption support region

Description

Select region that will represent a support for adsorption analysis. Adsorbed molecules will receive an '(ads)' suffix after name of the element bonded to the support. Such elements will be listed separate from atoms of the same element not bonded to the support, for example, HOH(ads) for a water molecule bonded to a surface via one of its H atoms.

BondOrderCutoff

Type

Float

Default value

0.5

Description

Bond order cutoff for analysis of the molecular composition. Bonds with bond order smaller than this value are neglected when determining the molecular composition.

8.3 Bond guessing algorithm

The algorithm to determine bond orders purely from atomic positions, which is used if `BondOrders%Method = Guess`, is roughly as follows:

- Each chemical element has a maximal number of bonds (`connectors` property taken from the PLAMS `PeriodicTable` class) and a covalent radius.
- Calculate interatomic distances
- Determine bond orders between all non-metal atoms
- Add bonds to any isolated H atoms (these bonds do not count towards the maximal number of bonds for an element).
- Find central atoms of common anions like carbonate, nitrate, sulfate, phosphate, and arsenate
- Add bonds between metal atoms and all other atoms, except the ones from the previous step

- Delete metal-metal bonds and metal-hydrogen bonds if the metal is bonded to enough electronegative atoms and not enough metal atoms. This means that the metal is likely a cation.

The calculated bonds are useful for visualization, but not necessarily for setting up ForceField simulations.

The bond orders to metal atoms are always 1.

Note: The problem of finding molecular bonds for a given set of atoms in space does not have a general solution, especially considering the fact the chemical bond in itself is not a precisely defined concept. For every method, no matter how sophisticated, there will always be corner cases for which the method produces disputable results. Moreover, depending on the context (area of application) the desired solution for a particular geometry may vary. The algorithm used here gives good results for geometries that are not very far from the optimal geometry, especially consisting of lighter atoms. All kinds of organic molecules, including aromatic ones, usually work very well. Problematic results can emerge for transition metal complexes, transition states, incomplete molecules etc.

Warning: This method works reliably only for geometries representing complete molecules. If some atoms are missing (for example, a protein without hydrogens) the resulting set of bonds would usually contain more bonds or bonds with higher order than expected.

A version of this algorithm is also implemented in the PLAMS `Molecule.guess_bonds()` method.

8.4 Orbital energies, HOMO-LUMO gap

Ab-initio or semiempirical engines, such as ADF or DFTB, internally compute molecular orbital energies and occupations (see *Summary of engine capabilities* (page 318)). This information is printed to the AMSResults section of the text output and is stored in the AMSResults section of the binary engine results.

Example of molecular orbitals information printed to the text output:

```
-----
Molecular orbitals info
-----
```

```
n orbitals:      8
n spin:          1

HOMO (eV) :      -13.5919
LUMO (eV) :      -4.2036
HOMO-LUMO gap (eV) : 9.3883

  Index   Energy (eV)   Occupation
    8      12.3038      0.000
    7       9.4801      0.000
    6      -1.3196      0.000
    5      -4.2036      0.000 < LUMO
    4     -13.5919      2.000 < HOMO
    3     -14.8182      2.000
    2     -16.6543      2.000
    1     -20.6183      2.000
```

n spin: 1 in case of spin-restricted or spin-orbit calculations, 2 in case of spin-unrestricted calculations.

Occupations: in case of spin-unrestricted calculations the occupation numbers will be in the range 0-2. In case of spin-restricted calculation or spin-orbit calculations, the the occupation numbers will be in the range 0-1.

Note: In case of **fractional occupation** (i.e. non-integer occupations numbers), the definition of HOMO and LUMO is somewhat ill defined.

If fractional occupations are detected, a message will be printed to the text output.

The logical variable `AMSResults%fractionalOccupation` in the engine .rkf results file indicates whether fractional occupation was detected.

Here, we use the following convention:

- The HOMO is the highest-energy orbital for which `occupation[iOrbital] > occupied_threshold`. The LUMO is HOMO+1.
- if `occupation[LUMO] < unoccupied_threshold` we have integer occupations.
- if `occupation[LUMO] > unoccupied_threshold` we have we have **fractional occupations**

where `occupied_threshold = 0.9 * max_occupation`, `unoccupied_threshold = 0.1 * max_occupation` (max occupation = 2 for spin-restricted calculations and 1 for spin-unrestricted or spin-orbit calculations)

In case of spin-unrestricted calculation, the “smallest HOMO-LUMO gap” is the smallest HOMO-LUMO gap irrespective of spin (i.e. $\min(\text{LUMO}) - \max(\text{HOMO})$).

The molecular orbital information is generally automatically computed by engines supporting it.

You can explicitly request this in the input via the following keyword:

```
Properties
  OrbitalsInfo Yes/No
End
```

Properties

OrbitalsInfo

Type

Bool

Default value

No

Description

Basic molecular orbitals information: orbital energies, occupations, HOMO, LUMO and HOMO-LUMO gap.

If you explicitly request `OrbitalsInfo` to be computed, but the engine cannot compute it, AMS will stop with an error.

Molecular orbital information can only be calculated in case of non-periodic systems.

8.5 Uncertainties

Some engines may report uncertainties for some properties. Currently only the [Hybrid engine](#) supports uncertainties and only when running in committee mode.

Uncertainties are reported to text output:

```
Engine energy uncertainty (hartree)           0.01253685

Uncertainty in Gradients (hartree/bohr)
  Index  Atom      d/dx      d/dy      d/dz      Magnitude
    1     C      0.0005234578  0.0001727093  0.0002762926  0.0002311556
    2     C      0.0000008845  0.0007601116  0.0000014245  0.0007601017
    3     C      0.0005218907  0.0001737959  0.0002777874  0.0002325467
    4     O      0.0004150496  0.0002459280  0.0000738370  0.0004012048
    5     O      0.0004146230  0.0002469354  0.0000735124  0.0004003429
    6     H      0.0000668939  0.0001815939  0.0001866700  0.0001726131
    7     H      0.0001069923  0.0001412011  0.0001620722  0.0001367296
    8     H      0.0001069626  0.0001408593  0.0001624397  0.0001365849
    9     H      0.0000668212  0.0001810509  0.0001866413  0.0001723906
```

Additionally, they can be obtained from the engine rkf file in the `AMSResults%EnergyUncertainty`, `AMSResults%GradientsUncertainty` for the uncertainty in the energy in the energy and gradients respectively. For convenience, `AMSResults&GradientsMagnitudeUncertainty` is provided for convenience, which reports the uncertainty in the magnitude of the gradients based on the variance formula, or error propagation.

During Task `MolecularDynamics`, the uncertainties are additionally provided in the *History* section of *ams.rkf* in the variables `EngineEnergyU(#)`, `EngineGradientsU(#)` and `EngineGradientsNormU(#)` (Magnitude).

It is possible to define an [exit condition](#) (page 189) when uncertainties are too high.

See also:

In [PLAMS](#), you can use dedicated getters to extract this information from `AMSResults`.

ENGINES

The engines are the core of the Amsterdam Modeling Suite: While the AMS driver steers the calculation over the potential energy surface in e.g. a *geometry optimization* (page 49) or *molecular dynamics* (page 101) calculation, the engines calculate energies and gradients and in this way *define* the PES on which the driver works.

```
Engine header
```

Engine**Type**

Block

Description

The input for the computational engine. The header of the block determines the type of the engine.

The engine used for an AMS calculation is selected and configured with the special `Engine` block in the AMS input:

```
Engine DFTB
... input for the DFTB engine ...
EndEngine
```

Here the type of engine, e.g. `DFTB` as in the example above, is specified on the line that opens the block. Note that the `Engine` block ends with `EndEngine`, and is in this way different from all the other blocks in the AMS input, which close just with `End`. The content of the engine block is what we call the “engine input”. Generally the engine input consists of a series of blocks and keywords, and looks just like the AMS driver input. However, many engines have a lot of options and keywords, which are documented in a separate engine manual. In other words: This AMS driver manual documents all the keywords outside of the `Engine` block, while the individual engine manuals document the contents of the engine block.

9.1 Available engines

The following engines are available in the 2025.1 release of the Amsterdam Modeling Suite:

- **ADF**
A DFT engine particularly strong in understanding and predicting structure, reactivity, and spectra of molecules.
- **BAND**
An atomic-orbital based DFT engine aimed at periodic systems (crystals, slabs, chains) but supporting also molecular systems.
- **DFTB**
An engine implementing Density Functional based Tight-Binding, a fast approximation to DFT.

- **ReaxFF**
An engine for modeling chemical reactions with atomistic potentials based on the reactive force field approach.
- **MLPotential**
Machine learning potentials (machine learning force fields).
- **ForceField**
An engine implementing classical non-reactive force fields such as UFF (a non-reactive force field covering the entire periodic table).
- **GFNFF**
The GFNFF force field
- **Hybrid**
Hybrid engine, for embedding and QM/MM calculations that combine multiple engines.
- **MOPAC**
An engine wrapping the MOPAC code, a general-purpose semiempirical molecular orbital package for the study of solid state and molecular structures and reactions.
- **ASE**
A flexible scripting interface that allows advanced users to use external modeling programs as engines in AMS through the Atomic Simulation Environment (ASE).
- **External (page 320)**
A flexible scripting interface that allows advanced users to use external modeling programs as engines in AMS.
- **LennardJones (page 324)**
A simple toy engine implementing a Lennard-Jones potential.

9.2 Summary of engine capabilities

Some options/properties can only be computed in combination with some of the engines (i.e., not all engines support all features). These tables summarize the capabilities of the engines of the Amsterdam Modeling Suite:

Table 9.1: Engine support for AMS driver options

Feature	ADF	BAND	DFTB	ReaxFF	MLPot.	Force-Field	MOPAC	GFNFF	Engine ASE
All elements available	✓	✓	✓ ²			✓ ³	✓	✓	✓ ⁵
<i>Non-periodic systems</i> (page 31)	✓	✓	✓	✓	✓ ⁴	✓	✓	✓	✓ ^{Page 319, 5}
<i>1D periodic systems</i> (page 31)		✓	✓	✓		✓	✓		✓ ^{Page 319, 5}
<i>2D periodic systems</i> (page 31)		✓	✓	✓		✓	✓		✓ ^{Page 319, 5}
<i>3D periodic systems</i> (page 31)		✓	✓	✓	✓ ^{Page 319, 4}	✓	✓	✓	✓ ^{Page 319, 5}
Charged molecular systems	✓	✓	✓	✓		✓	✓	✓	✓ ^{Page 319, 5}
External electric field	✓	✓	✓				✓		
External point charges	✓	✓	✓						

Table 9.2: Engine support for AMS driver properties

Feature	ADF	BAND	DFTB	ReaxFF	MLPot.	Force-Field	MOPAC	GFNFF	Engine ASE
<i>Atomic charges</i> (page 309)	✓	✓	✓	✓		✓	✓	✓	
<i>Bond orders</i> (page 310) ¹	✓		✓	✓		✓	✓	✓	
<i>Orbital energies</i> (page 313)	✓	✓	✓				✓		
<i>HOMO-LUMO gap</i> (page 313)	✓	✓	✓				✓		
<i>Dipole gradients</i> (page 309)	✓	✓	✓				✓	✓	✓ ⁵
<i>Dipole moment</i> (page 309)	✓	✓	✓				✓	✓	✓ ⁵
<i>Elastic tensor</i> (page 242)		✓	✓	✓	✓	✓	✓	✓	
<i>Gradients / Forces</i> (page 237)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>Hessian</i> (page 238)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>Infrared spectra</i> (page 247)	✓	✓	✓				✓	✓	
<i>Molecule detection</i>	✓		✓	✓		✓	✓	✓	
<i>Normal modes</i> (page 247)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>PES point character</i> (page 239)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>Phonons</i> (page 274)		✓	✓	✓	✓	✓	✓	✓	✓
<i>Polarizability</i> (page 309)	✓								
<i>Raman</i> (page 278)	✓								
<i>Stress tensor</i> (page 241)		✓	✓	✓	✓	✓	✓	✓	✓
<i>Thermodynamic properties</i> (page 269)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>VCD</i> (page 285)	✓	✓	✓				✓		
<i>VROA</i> (page 283)	✓								

Note: The features/options in the following tables are **engine-specific**, and are described in the corresponding engine's manual. The input for these options should be specified in the *Engine block* (page 7) section of the input file.

² All elements are available for the GFN1-xTB model, but other DFTB parameter sets are not parametrized for all elements.

³ All elements are available with UFF (universal force field), but other force fields are not parametrized for all elements.

⁵ The ASE Engine only supports communicating these properties to and from the concomitant ASE Calculator. Actual support is determined by the ASE Calculator used.

⁴ MLPotential backends may support non-periodic and/or 3D-periodic calculations.

¹ If the engine does not support the calculation of bond orders, a bond guessing algorithm will be used instead. See *Bond Orders* (page 310).

Table 9.3: Engine-specific capabilities

Feature	ADF	BAND	DFTB	ReaxFF	MLPot.	Force-Field	MOPAC	GFNFF	Engine ASE
EFG	✓ (doc)	✓ (doc)							
Electronic transport	✓ (doc)	✓ (doc)	✓ (doc)						
Energy decomposition analysis	✓ (doc)	✓ (doc)							
ESR/EPR	✓ (doc)								
Hyperpolarizabilities	✓ (doc)								
NMR	✓ (doc)								
pKa							✓ (doc)		
Solvation models	✓ (doc)	✓ (doc)	✓ (doc)				✓ (doc)		
Spin-polarization	✓ (doc)	✓ (doc)	✓ (doc)				✓ (doc)		
UV-VIS optical spectra	✓ (doc)		✓ (doc)						

Table 9.4: Engine-specific capabilities for periodic systems

Feature	ADF	BAND	DFTB	ReaxFF	MLPot.	Force-Field	MOPAC	Engine ASE
Density of states (DOS)		✓ (doc)	✓ (doc)					
Dielectric function		✓ (doc)						
Effective mass		✓ (doc)	✓ (doc)					
Electronic band structures		✓ (doc)	✓ (doc)					
k-space sampling		✓ (doc)	✓ (doc)					
X-ray form factors		✓ (doc)						

9.3 External programs as engines

The AMS driver allows running external programs as an engine. In this way users can combine the functionality in the AMS driver (tasks and PES point properties) with the energies and gradients of any molecular modeling program they have access to. For example, the graphical user interface supports setting up [VASP as an external engine to the AMS driver](#).

See also:

As an alternative to Engine External, [Engine ASE](#) also lets you couple external programs to AMS through the Python library ASE (Atomic Simulation Environment).

In general, the interfacing between the AMS driver and the external program has to be done by the user in form of a small script, which allows users to hook up any external program without access to the source code of AMS. The graphical user interface of the Amsterdam Modeling Suite can then be used to analyze the results of these calculations.

An external engine is specified with `Engine External`. The keyword `Execute` is used to specify the command that is run to execute the external program:

```
Engine External
  Execute /path/to/my_interface_script.sh
EndEngine
```

The command can in principle be anything, as it will just be executed as is by the system shell. However, it should not use relative paths (e.g. to files in the directory where the input file is). We recommend writing the interfacing script in Python and using the Python interpreter that ships with AMS:

```
Engine External
  Execute "$AMSBIN"/amspython /path/to/my_python_interface_script.py
EndEngine
```

AMS then starts running and for every geometry prepares a folder in which the external engine is supposed to run. This is the folder in which the interface script specified with the `Execute` key is executed (so any relative paths are relative to that folder). AMS puts two files into this folder:

```
system.xyz
request.json
```

The `system.xyz` just contains the geometry AMS wants the external engine to calculate. It is an *extended format XYZ file* (page 565) with the VEC1, VEC2, VEC3 extension at the end for periodic systems, e.g. diamond would look like this:

```
2
C      -0.51292147   -0.51292147   -0.51292147
C       0.51292147    0.51292147    0.51292147
VEC1    0.00000000    2.05168587    2.05168587
VEC2    2.05168587   -0.00000000    2.05168587
VEC3    2.05168587    2.05168587    0.00000000
```

The `request.json` file is just a small JSON file that specifies what exactly AMS wants the external engine to calculate:

```
{
  "title": "GOSTep28",
  "quiet": false,
  "gradients": true,
  "stressTensor": false,
  "hessian": true,
```

(continues on next page)

(continued from previous page)

```
"dipoleMoment": false,  
"properties": true,  
"prevResults": "GOStep27"  
}
```

The job of the interfacing script is now to read these files, run the external program and convert its output into a format understood by AMS. Generally these are simple text files with the name of the property and the extension `.txt`. The bare minimum the interfacing script needs to produce is the file `energy.txt` containing a single number, i.e. the total energy in atomic units (Hartree). Other properties are optional, and it is easiest to go through the `request.json` entries one by one to see what AMS might request and what the interfacing script could produce in response.

title

Just a title for this particular engine run. It can be passed on to the external program if desired, or can just be ignored.

quiet

Whether AMS wants the external engine to write to standard output. This can be ignored in principle, but that might lead to really incomprehensible text output files of AMS if the external engine has to be called many times, e.g. for numerical derivatives.

gradients

Whether or not to calculate nuclear gradients. The interface script should put the gradients in a file called `gradients.txt` with `nAtoms` lines of 3 real numbers each, in atomic units, i.e. Hartree/Bohr. Note that AMS wants the gradients, not forces (beware the - sign!).

stressTensor

Whether to calculate the stressTensor for periodic systems. Should be written to `stresstensor.txt` in atomic units.

hessian

Whether to calculate the Hessian, that is just the second derivative of the energy with respect to the nuclear coordinates, *without* applying any mass weighing to it. If the Hessian has been calculated, it should be put in `hessian.txt` as a 3 `nAtoms` x 3 `nAtoms` matrix in atomic units.

dipoleMoment

If true, calculate the dipole moment and put it in `dipolemoment.txt` in atomic units, in one line with three numbers.

properties

This is set to true if AMS considers this “geometry” important and wants the engine to calculate further properties that the user might be interested in. In practice this is set to “true” for e.g. the final converged step in a geometry optimization, so that the user can then let the engine calculate e.g. the band structure, which one would not want to do at all the steps *during* the optimization. AMS can’t do anything with the properties that the engine might calculate, but the files will remain on disk for people to inspect them.

prevResults

This is the title of a previous similar calculation that the engine has already performed. These results can be accessed in `../$prevResults/`, so for the example above `GOStep28` can access the results from the previous step in the geometry optimization in `../GOStep27/`. This is just the directory in which the interfacing script was run when the `title` field was set to `GOStep27`, so files that were written back then are still accessible. They can in principle be used to restart for example the SCF of the engine from step to step. Of course all of that has to be done by the interfacing script. The AMS driver does not know anything about how to restart the external program and can only point the interfacing script to the right location.

That is really all there is to the external engine: AMS prepares a folder with `system.xyz` and `request.json` and runs the user’s interfacing script in there, which has to take care of preparing the input for the external engine, running it, and putting the results in the text files that AMS expects, e.g. `gradients.txt`.

Note for properties that are in one way or another derivatives of the energy, it is generally ok if the external engine does not calculate what was requested by the AMS driver in `request.json`. If AMS requests, for example, the gradients from the external engine, but then does not find the `gradients.txt` in the directory after the interfacing script has run, it will just assume that the engine was not capable of calculating the gradient analytically. AMS will then just do the gradient numerically by rerunning the external engine for displaced geometries, reading only the energy from `energy.txt`. In this sense it is only absolutely required for the external engine to produce the energy, the rest can be done numerically by AMS if required. It is of course best to let the engine do as much as possible, especially if it implements analytical derivatives. Note that currently AMS can not calculate the Hessian numerically for engines that do not provide gradients. This is just a technical limitation, as it is of course possible to do a second derivative numerically, but it is just not implemented in AMS yet. (And it would also be a very slow way to calculate a Hessian.)

In addition to the `Execute` keyword that specifies the interfacing script, it is also possible to use the `ExecuteAtEnd` keyword to specify another command to be run at the end of the calculation, for example after the last step of a geometry optimization or molecular dynamics simulation. This command is run in the results directory.

Moreover, the `Engine External` block can also contain some information about the capabilities of the external engines:

```
Engine External
  Execute {...}
  ExecuteAtEnd {...}
  Supports
    DipoleMoment      {true|false}
    PeriodicityNone   {true|false}
    PeriodicityChain  {true|false}
    PeriodicitySlab   {true|false}
    PeriodicityBulk   {true|false}
  End
EndEngine
```

The normal engines that come with AMS (e.g. DFTB and BAND) produce the engine output files with extension `.rkf` in the results directory, see [here](#) (page 19). These files are also produced when an external engine is used and the information on them (anything related to the shape of the PES at that point, e.g. normal modes, phonons, ...) can be visualized normally with the graphical interface. In addition to each engine output `.rkf` file, external engines will also produce a correspondingly named folder per engine file, which is just the working directory of the interfacing script for that particular invocation of the external program. These folders just contain the full output of the external program and anything that the interfacing script might have produced. In this way users still have access to all results from the external program, even if these results were not communicated back to the AMS driver.

This last point is probably best illustrated with a simple example. Consider the following job that uses an external engine to do a linear transit calculation of ethane, going from the staggered to the eclipsed configuration, calculating normal modes at all converged points along the path:

```
AMS_JOBNAME=ethane_torsion $AMSBIN/ams << EOF

Task PESScan

System
  Atoms
    C      0.00000000      0.00000000      0.76576000
    C      0.00000000      0.00000000     -0.76576000
    H     -0.88668938      0.51193036      1.16677000
    H      0.88668938      0.51193036      1.16677000
    H      0.00000000     -1.02386071      1.16677000
    H      0.00000000      1.02386071     -1.16677000
    H     -0.88668938     -0.51193036     -1.16677000
    H      0.88668938     -0.51193036     -1.16677000
```

(continues on next page)

(continued from previous page)

```

    End
End

PESScan
  CalcPropertiesAtPESPoints True
  ScanCoordinate
    nPoints 5
    Dihedral 3 1 2 6 60.0 0.0
  End
End

Properties
  NormalModes True
End

Engine External
  ...
EndEngine

EOF

```

If we run this job and look into the results folder, we will find the standard `ams.log` and `ams.rkf` as well as the usual engine result files `PESPoint(1).rkf` to `PESPoint(5).rkf`. Just as if we had used one of the native AMS engines, like DFTB. Each of these files can be opened in AMSspectra to visualize the normal modes for this particular point. For an external engine we additionally have one folder per engine file, so for this example we would have `PESPoint(1)/` to `PESPoint(5)/`. These are the folders in which the interfacing script ran for these particular points, so they contain all the native output files of the external program.

9.4 Toy engines

The AMS driver comes with a simple built-in toy engine that implements a Lennard-Jones potential. This can sometimes be useful for testing, as many properties of the Lennard-Jones gas/liquid/solid can be calculated analytically and compared to the results from AMS. Note that the potential is exactly the same for all elements, i.e. the N-N bond has exactly the same strength as the He-He bond.

The Lennard-Jones engine only has three keywords, which define the shape of the potential:

```

Engine LennardJones
  RMin float
  Eps float
  Cutoff float
EndEngine

```

Cutoff

Type

Float

Default value

15.0

Unit

Angstrom

Description

The distance at which the interaction is truncated.

Eps**Type**

Float

Default value

1.0

Unit

Hartree

Description

The depth of the potential well.

RMin**Type**

Float

Default value

1.0

Unit

Angstrom

Description

The distance of the potential minimum.

9.5 Load an Engine configuration from file

Loading an engine configuration from file from a previous calculation is possible if the engine information is written to an *engine output file* (page 19).

```
LoadEngine string
```

LoadEngine**Type**

String

Description

The path to the file from which to load the engine configuration. Replaces the Engine block.

Note that the `LoadEngine` block is mutually exclusive with the `Engine` block: The Engine *either* needs to be specified in the input, *or* loaded from a previous results file.

9.6 Engine add-ons

Engine add-ons can be used to augment the results returned from an engine.

9.6.1 Dispersion corrections

For engines that do not natively support Grimme's D3 and D4 dispersion corrections, the `D3Dispersion` and `D4Dispersion` engine add-ons can be used to add such corrections.

Version 1.2.1 of `s-dftd3` for DFT-D3 is used (<https://github.com/dftd3/simple-dftd3>). For elements for which no parameters are present ($Z > 103$), the parameters for the element $Z' = Z - 32$ will be used. Version 3.7.0 of `dftd4` for DFT-D4 is used (<https://github.com/dftd4/dftd4>). For elements for which no parameters are present ($103 < Z < 112, Z > 118$), the parameters for the element $Z' = Z - 32$ will be used.

```
EngineAddons
  D3Dispersion
    Damping [BJ | Zero]
    Enabled Yes/No
    Functional string
    a1 float
    a2 float
    s6 float
    s8 float
    sr6 float
  End
  D4Dispersion
    Enabled Yes/No
    Functional [...]
    Verbosity [Silent | Normal | Verbose | VeryVerbose]
    a1 float
    a2 float
    s6 float
    s8 float
    s9 float
  End
End
```

EngineAddons

D3Dispersion

Type

Block

Description

This block configures the add-on that adds the Grimme D3 dispersion correction to the engine's energy, gradients, and stress tensor.

Damping

Type

Multiple Choice

Default value

BJ

Options

[BJ, Zero]

Description

Type of damping: BJ (Becke-Johnson) or Zero. BJ is recommended for most applications.

Enabled

Type

Bool

Default value

No

GUI name

D3 dispersion

Description

Enables the D3 dispersion correction addon.

Functional**Type**

String

Default value

PBE

Description

Use the D3 parameterization by Grimme for a given xc-functional. Accepts the same values as the `-func` command line option of the official `dftd3` program. Note: the naming convention is different from elsewhere in the AMS suite. For example, BLYP should be called b-lyp.

a1**Type**

Float

Description

The a1 parameter. Only used if Damping is set to BJ. If set, it overwrites the a1 value for the chosen functional.

a2**Type**

Float

Description

The a2 parameter. Only used if Damping is set to BJ. If set, it overwrites the a2 value for the chosen functional.

s6**Type**

Float

Description

The s6 parameter, global scaling parameter. If set, it overwrites the s6 value for the chosen functional.

s8**Type**

Float

Description

The s8 parameter. If set, it overwrites the s8 value for the chosen functional.

sr6**Type**

Float

Description

The sr6 parameter. Only used if Damping is set to Zero. If set, it overwrites the sr6 value for the chosen functional.

D4Dispersion**Type**

Block

Description

This block configures the addon that adds the Grimme D4(EEQ) dispersion correction to the engine's energy, gradients, stress tensor and Hessian.

Enabled**Type**

Bool

Default value

No

GUI name

D4 dispersion

Description

Enables the D4 dispersion correction addon.

Functional**Type**

Multiple Choice

Default value

PBE

Options

[HF, BLYP, BPBE, BP86, BPW, LB94, MPWLYP, MPWPW91, OLYP, OPBE, PBE, RPBE, REVPBE, PW86PBE, RPW86PBE, PW91, PW91P86, XLYP, B97, TPSS, REVTPSS, SCAN, B1LYP, B3LYP, BHLYP, B1P86, B3P86, B1PW91, B3PW91, O3LYP, REVPBE0, REVPBE38, PBE0, PWP1, PW1PW, MPW1PW91, MPW1LYP, PW6B95, TPSSH, TPSS0, X3LYP, M06L, M06, OMEGAB97, OMEGAB97X, CAM-B3LYP, LC-BLYP, LH07TSVWN, LH07SSVWN, LH12CTSSIRPW92, LH12CTSSIFPW92, LH14TCALPBE, B2PLYP, B2GPPLYP, MPW2PLYP, PWPB95, DSDBLYP, DSDPBE, DSDPBEB95, DSDPBEP86, DSDSVWN, DODBLYP, DODPBE, DODPBEB95, DODPBEP86, DODSVWN, PBE02, PBE0DH, DFTB(3ob), DFTB(mio), DFTB(pbc), DFTB(matsci), DFTB(ob2), B1B95, MPWB1K, REVTPSSH, GLYP, REVPBE0DH, REVTPSS0, REVDSDPBEP86, REVDSDPBEPBE, REVDSDBLYP, REVDODPBEP86, B97M, OMEGAB97M, R2SCAN]

Description

Use the D4 parameterization by Grimme for a given xc-functional.

Verbosity**Type**

Multiple Choice

Default value

Silent

Options

[Silent, Normal, Verbose, VeryVerbose]

Description

Controls the verbosity of the dftd4 code. Equivalent to the `–silent` and `–verbose` command line switches of the official dftd4 program.

a1**Type**

Float

Description

The a1 parameter, see D4 article. The physically reasonable range for a1 is [0.0,1.0]. If set, it overwrites the a1 value for the chosen functional.

a2**Type**

Float

Description

The a2 parameter, see D4 article. The physically reasonable range for a2 is [0.0,7.0]. If set, it overwrites the a2 value for the chosen functional.

s6**Type**

Float

Description

The s6 parameter, see D4 article. The physically reasonable range for s6 is [0.0,1.0]. If set, it overwrites the s6 value for the chosen functional.

s8**Type**

Float

Description

The s8 parameter, see D4 article. The physically reasonable range for s8 is [0.0,3.0]. If set, it overwrites the s8 value for the chosen functional.

s9**Type**

Float

Description

The s9 parameter, see D4 article. If set, it overwrites the s9 value for the chosen functional.

9.6.2 Pressure

Pressure can be included using the following keyword (this can only be used for **3D periodic systems**):

```
EngineAddons
  Pressure float
End
```

EngineAddons**Pressure****Type**

Float

Default value

0.0

Unit

GPa

Description

Add a hydrostatic pressure term to the engine's energy and stress tensor. Can only be used for 3D periodic boundary conditions.

The engine's energy will include the following extra term: $P \cdot V$, where V is the volume of the unit cell. The engine's stress tensor will include the following extra term: $P \cdot \mathbb{I}$, where \mathbb{I} is the identity matrix. This can only be used for 3D periodic boundary conditions (i.e. bulk). The energy and stress tensor printed in output and written to the `.rkf` binary file will include these extra terms (i.e. the printed stress tensor is the sum of the internal stress and the external stress due to pressure).

When studying the effect that pressure has on the structure and properties of your system, one should generally start by *optimizing the structure* (page 49) **including the lattice vectors** under pressure. Properties such as *phonons* (page 274) or *elastic tensor* (page 242) can then be computed at the relaxed geometry. If you are investigating phase transitions under pressure (or if you simply expect some symmetry breaking) you should disable symmetry and/or perturb the initial geometry of your system.

See also:

Example: Periodic lattice optimization under pressure (page 459)

Warning: If you want to include pressure in *molecular dynamics* (page 101) calculations, you should not use this engine addon, but use the MD-specific pressure option.

9.6.3 Non-isotropic external stress

An **non-isotropic external stress tensor** can be included by using the following keywords:

```
EngineAddons
  ExternalStress
    StressTensorVoigt float_list
    UpdateReferenceCell Yes/No
  End
End
```

EngineAddons**ExternalStress****Type**

Block

Description

This block configures the addon that adds external stress term to the engine's energy and stress tensor.

StressTensorVoigt**Type**

Float List

UnitHartree/Bohr³

GUI name

External stress tensor

Description

The elements of the external stress tensor in Voigt notation. One should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

UpdateReferenceCell**Type**

Bool

Default value

No

Description

Whether or not the reference cell should be updated every time the system changes (see documentation).

The energy and stress tensor printed in output and written to the `.rkf` binary file already include the corresponding extra terms, i.e. the printed stress tensor is the sum of the internal stress and the input-specified external stress.

When studying the effect that an external stress has on the structure and properties of your system, one should generally start by *optimizing the structure* (page 49) **including the lattice vectors** under the external stress (depending on the magnitude of the applied external stress, you might have to adjust the *stress energy per atom convergence threshold* (page 50)). If your system is symmetric, you should **disable symmetry** when optimizing structures under external stress. Be aware that the geometry optimization might go completely astray (e.g. the material will break apart) if you apply a) too large shear stress or b) too large tension stress (too large negative stresses for the diagonal values).

Following this paper⁶ from Parrinello and Rahman, the extra energy term due to a non-isotropic external stress⁷ is defined with respect to a *reference unit cell*, which in our case is the unit cell at the beginning of the simulation. If, during the simulation, large deformations of the unit cell occur, the above mentioned energy expression is only approximately consistent with the stress tensor. This affects the calculation differently depending on whether the stress tensor is computed by the engine or by AMS via numerical differentiation: a) If the stress tensor is computed directly by the engine, for large unit-cell deformations the energy might increase during the optimization; b) If the stress tensor is computed by AMS via numerical differentiation, the actual final value of the stress tensor might not match perfectly the external stress specified in the input (for large cell deformations, this error can be in the order of 10%).

After the optimization under external stress is converged, it is therefore good practice to validate the results. To do this, you should compute the *stress tensor* (page 241) at the optimized geometry by performing a *single point calculation* (page 47) **without** applying the external stress. The values in the computed stress tensor should have opposite sign compare to the external stress applied during the optimization. If the values differ too much, then you can run a second geometry optimization under external stress starting from the optimized geometry.

See also:

Example: Boron nitride optimization under external stress (page 462)

An alternative option is to set the key `ExternalStress%UpdateReferenceCell` to `True`; this will update the reference unit cell at every optimization step, effectively changing the definition of the energy expression at every geometry optimization step. The energy might not consistently go down during the optimization, but the resulting internal stress

⁶ M. Parrinello, and A. Rahman, *Polymorphic transitions in single crystals: A new molecular dynamics method*, *Journal of Applied Physics* 52, 7182 (1981) (<https://doi.org/10.1063/1.328693>)

⁷ The energy term due to a non-isotropic external stress is:

$$E_{\text{stress}} = p_h(V - V_0) + \frac{1}{2}V_0\text{Tr}[(\sigma - p_h\mathbb{I})(h_0^{T-1}h^Thh_0^{-1} - \mathbb{I})]$$

where: σ is the external stress tensor, p_h is the hydrostatic pressure associated with σ (i.e. the average of the diagonal elements of σ), V is the volume of the unit cell (for 2D periodic systems this is the area of the cell, and for 1D periodic systems is the length of the cell), V_0 is the volume of the *reference unit cell*, h are the lattice vectors in matrix form, h_0 are the lattice vectors of the reference unit cell in matrix form and \mathbb{I} is the identity matrix.

will match much better the applied external stress. This option should only be used during geometry optimizations (i.e. it should not be used when computing properties such as the elastic tensor).

See also:

Example: Graphene optimization under external stress (page 464)

9.6.4 Atom energies

This add-on adds an element-dependent energy for each atom in a system, effectively applying an energy shift. It does not affect any other property.

```
EngineAddons
  AtomEnergies # Non-standard block. See details.
  ...
End
End
```

EngineAddons

Type

Block

Description

This block configures all the engine add-ons.

AtomEnergies

Type

Non-standard block

Description

Add an element-dependent energy per atom. On each line, give the chemical element followed by the energy (in atomic units).

The below example will add -0.5 Hartree for every H atom and -5.0 Hartree for every C atom in the system to the total energy. The add-on will ignore any atoms of other elements:

```
EngineAddons
  AtomEnergies
    H -0.5
    C -5.0
  End
End
```

9.6.5 Restraints

A restraint is potential energy function (a spring) attached to a certain coordinate, for example, an interatomic distance, the minimum of the potential energy being at the specified optimal value. A restraint can have one or two parameters: a ForceConstant and, for some types, a F(Inf) value. The ForceConstant parameter corresponds to second derivative of the restraint potential energy $\frac{d^2 V(x)}{dx^2}$ for any x (harmonic restraints) or only at $x = 0$ (other restraints). Here, x is a deviation from the restraint's optimal value.

Geometry restraints can be added using the following input block:

```

Restrains
  Profile [Harmonic | Hyperbolic | Erf | GaussianWell]
  fInfinity float
  Distance atomIdx1 atomIdx2 OptValue ForceConstant {Profile} {fInfinity}
  Angle
  Dihedral
  SumDist
  DifDist
End

```

The shape of the restraint potential function is defined by its profile type (Harmonic, Hyperbolic, Erf) and the appropriate parameter(s). The `Harmonic` profile is most suitable for geometry optimizations but may result in very large forces that can be problematic in molecular dynamics. For MD simulations the `Hyperbolic` or `Erf` may be more suitable because the restraint force is bounded by a user-defined value. A `Harmonic` restraint is defined as $V(x) = C \frac{x^2}{2}$. A `Hyperbolic` restraint is defined as $V(x) = a(\sqrt{1 + (bx)^2} - 1)$. An `Erf` restraint is defined as $V(x) = a(bx \cdot \operatorname{erf}(bx) + \frac{e^{-(bx)^2} - 1}{\sqrt{\pi}})$, which corresponds to an indefinite integral of the error function $\frac{dV(x)}{dx} = ab \cdot \operatorname{erf}(bx)$. The a and b parameters are calculated from the `FInfinity` and `ForceConstant` values. The `Erf` profile is very similar to `Hyperbolic` however its derivative converges to `F(Inf)` much faster.

There is a special restraint type, `GaussianWell`, that should only be used with MD `BondBoost` because it has a negligible force at large deviations. It is defined as $V(x) = a(1 - e^{-bx^2})$. Here, instead of the force constant and the asymptotic force value, the two parameters define the depth and the width of the well, respectively.

The following keys define global parameters for all restraints.

Restrains

Profile

Type

Multiple Choice

Default value

Harmonic

Options

[Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Default restraint profile

Description

Select the default type of restraint profile.

The harmonic profile is most suitable for geometry optimizations but may result in very large forces that can be problematic in molecular dynamic.

For MD simulations the `Hyperbolic` or `Erf` may be more suitable because the restraint force is bounded by a user-defined value.

A per-restraint profile type can be specified after the `ForceConstant` value on the corresponding restraint line.

FInfinity

Type

Float

Default value

1.0

GUI name

Default F(inf)

Description

Specify the default asymptotic value for the restraint force for the Hyperbolic and Erf profiles, in Hartree/Bohr or Hartree/radian.

A per-restraint value can be specified after the profile type on the corresponding restraint line.

The following keys define individual restraints. The force constant, profile type and the F(Inf) value can be set for each restraint individually.

Restraints**Distance****Type**

String

Recurring

True

Description

Specify two atom indices followed by the distance in Angstrom and, optionally, by the Force-Constant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the distance between the two specified atoms at the given value. For periodic systems this restraint follows the minimum image convention.

Angle**Type**

String

Recurring

True

Description

Specify three atom indices i j k followed by an angle in degrees and, optionally, by the Force-Constant (default is 0.3 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the i-j-k angle at the given value. For periodic systems this restraint follows the minimum image convention.

Dihedral**Type**

String

Recurring

True

Description

Specify four atom indices i j k l followed by an angle in degrees and, optionally, by the Force-Constant (default is 0.1 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the i-j-k-l dihedral angle at the given value. For periodic systems this restraint follows the minimum image convention.

DifDist**Type**

String

Recurring

True

Description

Specify four atom indices i j k l followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the difference R(ij)-R(kl) at the given value. For periodic systems this restraint follows the minimum image convention.

SumDist**Type**

String

Recurring

True

Description

Specify four atom indices i j k l followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the sum R(ij)+R(kl) at the given value. For periodic systems this restraint follows the minimum image convention.

A default value for the force constant depends on the restraint type: 1.0 Hartree/Bohr for Distance, SumDist and DifDist, 0.3 Hartree/radian for Angle and 0.1 Hartree/radian for Dihedral.

Below is an example if the Restraints block.

```
Restraints
  Profile Hyperbolic      # Change the default profile type
  fInfinity 10.0          # Change the asymptotic value for the restraint force away_
  ↪from optimum
# Type   Atoms   OptValue  FC   Profile  F(Inf)
Distance 1 2     5.0       1.0   Erf      1.0
Angle    1 2 3   90.0
Dihedral 4 1 2 3 180.0    0.1
# The next two together are equivalent to imposing two distance restraints: R(23) to_
↪0.85 Angstrom and R(14) to 0.65 Angstrom
SumDist  1 4 2 3   1.5     # Keep R(14)+R(23) as close to 1.5 Angstrom as possible
DifDist  1 4 2 3   0.2     # Keep R(14)-R(23) as close to 0.2 Angstrom as possible
End
```

9.6.6 Wall potential

This add-on adds an inward repulsive potential around the origin of the form

$$E_{\text{wall}}(r) = k \log(1 + \exp(-b(R - r)))$$

where the prefactor k, the radius R, and the gradient b are parameters, and r is the distance to the origin. The potential is time-independent and spherical. Inside the radius R, values for potential and forces are near zero. Outside of R, the potential exerts a nearly constant inward pointing force of $k \cdot b$ on each mass point⁸.

```
EngineAddons
  WallPotential
    Enabled Yes/No
    Gradient float
    Prefactor float
    Radius float
```

(continues on next page)

⁸ S. Grimme, *Exploration of Chemical Compound, Conformer, and Reaction Space with Meta-Dynamics Simulations Based on Tight-Binding Quantum Chemical Calculations*, Journal of Chemical Theory and Computation 15, 2847-2862 (2019) (<https://doi.org/10.1021/acs.jctc.9b00143>)

(continued from previous page)

End

End

EngineAddons**WallPotential****Type**

Block

Description

This block configures the addon that adds a spherical wall potential to the engine's energy and gradients.

Enabled**Type**

Bool

Default value

No

Description

Enables the wall potential addon. When enabled, a spherical wall of radius [Radius] around the origin will be added. The force due to the potential will decay exponentially inside the wall, will be close to [Prefactor*Gradient] outside and exactly half of that at the wall.

Gradient**Type**

Float

Default value

10.0

Unit

1/Angstrom

Description

The radial gradient outside the sphere.

Prefactor**Type**

Float

Default value

0.01

Unit

Hartree

Description

The multiplier for the overall strength of the potential.

Radius**Type**

Float

Default value

30.0

Unit

Angstrom

Value Range

value > 0

Description

The radius of the sphere, wherein the potential is close to zero.

9.6.7 Short-range repulsive potential

This add-on adds a repulsive Weeks-Chandler-Andersen potential to all atom pairs, which is equivalent to the shifted and truncated Lennard-Jones potential. It is useful when running high-temperature dynamics with a force-field that does not have a strong enough repulsion term of its own.

$$E_{\text{WCA}}(r) = \begin{cases} 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 + \frac{1}{4} \right), & \text{if } r < \sqrt[6]{2}\sigma \\ 0, & \text{if } r \geq \sqrt[6]{2}\sigma \end{cases}$$

The addon parameters are listed below.

```
EngineAddons
  Repulsion
    Enabled Yes/No
    Epsilon float
    HydrogenSigmaScale float
    Sigma float
    SkinLength float
  End
End
```

EngineAddons**Repulsion****Type**

Block

Description

This block configures an addon that adds a repulsive Weeks-Chandler-Andersen potential to all atom pairs.

Enabled**Type**

Bool

Default value

No

GUI name

Repulsion

Description

Enables the repulsive Weeks-Chandler-Andersen potential addon.

When enabled, all atom pairs will experience repulsion $E = 4 \cdot \epsilon \cdot \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 + \frac{1}{4} \right)$ at the distances shorter than about $1.12 \cdot \sigma$.

Epsilon

Type

Float

Default value

0.01

Unit

Hartree

Description

The epsilon parameter in the potential equation. It is equal to the amount of energy added at $r=\text{sigma}$.

HydrogenSigmaScale**Type**

Float

Default value

0.75

Unit

Angstrom

Description

The sigma parameter for a pair of atoms where one of them is hydrogen is scaled with the given factor. For H-H interactions the sigma is scaled with this value squared.

Sigma**Type**

Float

Default value

0.55

Unit

Angstrom

Description

The sigma parameter in the potential equation. The potential is exactly zero at the distances larger than about $1.12 \cdot \text{sigma}$

SkinLength**Type**

Float

Default value

2.0

Unit

Angstrom

Description

Technical parameter specifying skin length for the neighbor list generation. A larger value increases the neighbor list cutoff (and cost) but reduces the frequency it needs to be re-created.

10.1 AMSbatch

It is often necessary to repeat an identical job for many different systems. The 2023 release of the Amsterdam Modeling Suite adds the AMSbatch utility program to do this.

Tip: AMSbatch only covers the very simple case of repeating the exact same jobs for multiple systems. More complicated workflows with processing of results can easily be implemented by using our *scripting tools* (page 14) (e.g. *PLAMS*).

10.1.1 Overview

The input of AMSbatch is basically identical to the input of an AMS job. The only difference is with the *System blocks* (page 31). For AMSbatch every System block needs to have a unique name (supplied in the header of the block), which will be used as the identifier of the system. Consider the following example input, which independently does a geometry optimization followed by a normal modes calculation for two systems (CO and water):

```
#!/bin/sh

AMS_JOBNAME=go_freq $AMSBIN/amsbatch << EOF

Task GeometryOptimization
Properties NormalModes=True

Engine DFTB
[...]
EndEngine

System Water
  Atoms
    O 0 0 0
    H 0 1 0
    H 1 0 0
  End
End

System CO
  Atoms
    C 0 0 0
    O 0 0 1
  End
End
```

(continues on next page)

(continued from previous page)

```
End
EOF
```

Just like the AMS driver, running AMSbatch will create a *results directory* (page 18) on disk. Inside of the results directory, you will find subdirectories for the individual jobs (identified by their System name), containing the usual AMS output files. Running the script above would produce the following output:

```
go_freq.results/
├── amsbatch.log
├── amsbatch.rkf
├── CO
│   ├── ams.log
│   ├── ams.rkf
│   ├── CO.dill
│   ├── CO.err
│   ├── CO.in
│   ├── CO.out
│   ├── CO.run
│   ├── dftb.rkf
│   └── output.xyz
├── Water
│   ├── ams.log
│   ├── ams.rkf
│   ├── dftb.rkf
│   ├── output.xyz
│   ├── Water.dill
│   ├── Water.err
│   ├── Water.in
│   ├── Water.out
│   └── Water.run
```

10.1.2 Loading systems from file

Instead of supplying all systems individually in the input, AMSbatch also supports to a few ways to load multiple systems from files at once.

The `SystemsFromSMILES` keyword allows to run jobs directly on collections of SMILES strings, using RDKit to generate the initial 3D structure of the system:

```
SystemsFromSMILES
Methane, C
Ethane, CC
Propane, CCC
Butane, CCCC
Pentane, CCCCC
Hexane, CCCCCC
End
```

SystemsFromSMILES

Type

Non-standard block

Description

Generates systems from SMILES strings using the `plams.from_smiles()` functions. Each line should

contain first the system identifier separated by a comma from its SMILES string, e.g.: 'propanol, CCCO'

The `SystemsFromConformers` keyword can be used to load a set of conformers as found by our [Conformers tool](#) (page 344). This is especially useful as the last step of a conformer workflow, e.g. to do a single point for the calculation of spectra on each conformer within an energy window:

```
SystemsFromConformers
  File string
  MaxEnergy float
End
```

SystemsFromConformers

Type

Block

Description

Constructs a series of systems (with identifiers 'conformer1', 'conformer2', etc.) from a RKF or XYZ file from a Conformers run.

File

Type

String

Description

A path to an RKF or concatenated XYZ file.

MaxEnergy

Type

Float

Unit

Hartree

Description

Energy cut-off relative to lowest energy conformer.

See also:

[Tutorial on Conformer generation and Boltzmann-weighted spectra](#)

Finally, the `SystemsFromTrajectory` keyword allows reading frames from a trajectory in RKF or XYZ format:

```
SystemsFromTrajectory
  File string
  Frames integer_list
End
```

SystemsFromTrajectory

Type

Block

Description

Constructs a series of systems (with identifiers 'frame1', 'frame2', etc.) from a trajectory file.

File

Type

String

Description

A path to an RKF or concatenated XYZ file.

Frames**Type**

Integer List

Description

List of frames to load from the file. If not specified, all frames will be loaded.

10.1.3 Restart

AMSbatch based on the [PLAMS](#) scripting framework. As such it can rely on the [PLAMS rerun prevention](#) to implement restarting of interrupted depositions. The easiest way to restart an AMSbatch job is to include the `--restart` (or short: `-r`) command line flag:

```
#!/bin/sh

AMS_JOBNAME=myjob $AMSBIN/amsbatch --restart << EOF
...
EOF
```

This first (interrupted) run will have created the `myjob.results` directory. Running the above script again will move that directory to `myjob.results.bak` and reuse all successful jobs from the first run. (People already familiar with PLAMS will recognize that this works just like the `-r` flag on the [PLAMS launch script](#).)

10.2 AtomTyping

AtomTyping is a simple tool for assigning atom types to molecular systems. Currently only the GAFF force field is implemented. It uses AMBER's Antechamber under the hood.

10.2.1 Command Line Tool

This is a simple example showing how to generate force field parameters for ethanol:

```
#!/bin/sh

$AMSBIN/atomtyping << EOF

Type GAFF

System
  Atoms
    C -4.537101 -0.602542 -0.008859
    C -3.601843 -1.804345  0.037274
    O -2.267293 -1.387375 -0.058902
    H -4.301877  0.096761  0.821305
    H -5.588912 -0.942676  0.095614
    H -4.425039 -0.071643 -0.977931
    H -3.829607 -2.474689 -0.818032
    H -3.766077 -2.377916  0.977441
    H -2.012143 -1.043722  0.837186
```

(continues on next page)

(continued from previous page)

```

    End
  End
EOF

```

The most important input options are:

- **Type:** The force field type for which atom types and charges should be generated. Currently only GAFF is implemented, and this is also default.
- **System:** in the *System block* (page 31) we specify the initial structure of the molecule for which you want to generate force field information.

The utility creates a new AMS input file for a ForceField engine job using the new forcefield parameters. The new input is written to the logfile, and it is also stored in the .rkf file. The user can change the file as needed, or import it into amsinput and change settings from there.

Tip: GAFF atomtyping by Antechamber involves an AM1 geometry optimization of all molecules, in order to determine the atomic charges. For very big systems this can be very time consuming. There is an option to only compute AM1 single point atomic charges, with the option *AntechamberTask SinglePoint*. Do keep in mind that this will make the resulting force field parameters less reliable.

10.2.2 Python scripting

It is also possible to run atomtyping via Python scripting.

```

from scm.plams import Settings, Molecule
from scm.plams import init, finish
from scm.atomtyping import AtomTypingJob, AtomTypingResults

init()

# Run the atomtyping job
mol = Molecule('mol.in')
settings = Settings()
job = AtomTypingJob(molecule=mol, settings=settings)
results = job.run()

# Create the forcefield job (change the settings if needed)
newjob = results.get_plamsjob()
newjob.settings.input.ams.Task = GeometryOptimization

finish()

```

10.3 Conformers

Conformers is a flexible tool for conformer generation. It implements the RDKit¹, CREST² and Simulated Annealing conformer generation methods and can be combined with any *AMS engine* (page 315).

Tip: You can use **Conformers** from the Graphical User Interface or from *python with PLAMS*.

See also:

Tutorial on Conformer generation and Boltzmann-weighted spectra

10.3.1 New in Conformers-2025

- Initial conformer sets in many different formats can be used as input to the conformers tool (*Input molecule(s)* (page 346)).
- Details about the equivalence filtering can be written to output (*Conformers Equivalence* (page 348)).

Overview

This is a simple example showing how to generate conformers of ethanol:

```
#!/bin/sh

$AMSBIN/conformers << EOF

Task Generate

System
  Atoms
    C -4.537101 -0.602542 -0.008859
    C -3.601843 -1.804345  0.037274
    O -2.267293 -1.387375 -0.058902
    H -4.301877  0.096761  0.821305
    H -5.588912 -0.942676  0.095614
    H -4.425039 -0.071643 -0.977931
    H -3.829607 -2.474689 -0.818032
    H -3.766077 -2.377916  0.977441
    H -2.012143 -1.043722  0.837186
  End
End

Engine ForceField
EndEngine
EOF
```

The most important input options are:

- **Task:** the task for the Conformers tool. Here we simply **Generate** a set of conformers starting from an initial structure. See the *Tasks* (page 345) section for more information.

¹ G. Landrum, *Rdkit documentation* Release 2013 (<https://buildmedia.readthedocs.org/media/pdf/rdkit/latest/rdkit.pdf>)

² P. Pracht, F. Bohle, S. Grimme, *Automated exploration of the low-energy chemical space with fast quantum chemical methods* Physical Chemistry Chemical Physics: PCCP, 19 Feb 2020, 22(14):7169-7192 (<https://doi.org/10.1039/c9cp06869d>)

- **System:** in the *System block* (page 31) we specify the initial structure of the molecule for which you want to generate conformers, in this case an ethanol molecule.
- **Engine:** during the conformers generation, atomistic calculations will be executed under the hood to compute energy and forces. With the *engine* (page 347) block you can specify which atomistic engine to use and the options for the engine.

Tip: By default, the computationally fast (but not very accurate) *ForceField engine* is used to compute energy and forces. If you need more accurate geometries and energies for your conformers, you can specify a more accurate (but slow) engine in the generation step. Alternatively, you can use the *Optimize*, *Filter* and *Score tasks* (page 345) to efficiently get accurate geometries and energies while keeping the computational cost to a minimum. See *Workflow* (page 351) for an example.

When running the script above, a folder called `conformers.results` will be created and the generated conformers will be saved in there (in `.xyz` and `.rkf` formats). See the general *Input, execution and output* (page 7) section for more information.

Main Input Options

Tasks

You can specify the task for the Conformers tool via the `Task` input option:

```
Task [Generate | Optimize | Filter | Score | Expand]
```

The main (and default) **Task** is **Generate**: given an initial structure, the Conformers tool will generate a set of conformers.

The other tasks, described below, can be used to refine or combine previously generated conformer sets. The *Workflow* (page 351) example shows how the various tasks can be combined in a simple workflow to efficiently obtain accurate conformers.

Task

Type

Multiple Choice

Default value

Generate

Options

[Generate, Optimize, Filter, Score, Expand]

Description

The task to be performed by the Conformers tool.

‘Generate’: given a molecule, generate a set of conformers. Note: this task will automatically optimize, filter and score the conformers.

‘Optimize’: given a previously generated set of conformers, optimize, filter and score the structures using the specified engine.

‘Filter’: given one or more previously generated set of conformers, merge them into a single conformer set and filter out duplicate conformers. Note: this will not optimize or re-score the conformers.

‘Score’ given one or more previously generated set of conformers, re-score them by computing the energy using the specified engine. Note: this will only do a single point calculation, and will not optimize the structures.

In case of 'Optimize', 'Filter' and 'Score' you can specify the input conformer set(s) using the 'InputConformersSet' keyword.

Input molecule(s)

For the Generate task you can use the System or LoadSystem blocks to specify the input molecule. See the *System block* (page 31) documentation for a detailed description.

For the Optimize, Filter and Score tasks, you should instead provide the results of a previously run Conformers calculation as input via the InputConformersSet keyword. See also the *Constraints, Restraints, Walls* (page 350) section.

```
InputConformersSet string
```

InputConformersSet

Type

String

Recurring

True

Description

The path to a file containing a set of conformers.

The file should be either an 'conformers.rkf' file (i.e. the results file of conformers), a concatenated .xyz file, a .sdf file, or even .dcd file. In the case of a .dcd file, an additional System block is required. If the specified file does not contain energies in the AMS conformers format (as produced by the conformers tool), all conformer energies will be set to zero.

You can specify multiple input conformers sets by including the InputConformersSet keyword multiple times.

When loading conformers using the InputConformersSet key, you can optionally specify the InputMaxEnergy option:

```
InputMaxEnergy float
```

InputMaxEnergy

Type

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers when loading conformers sets using the Input-ConformerSet keyword. Conformers with an larger relative energy will not be loaded.

A similar option is the InputMaxConfs key, which will use only the k lowest energy conformers for an integer input k :

```
InputMaxConfs integer
```

InputMaxConfs

Type

Integer

Description

The maximum number of conformers to carry forward when loading conformer sets. If this input is not specified, this limit will not be imposed.

Engine

With the **Engine block** you can specify which atomistic engine to use and the options for the engine. See the [AMS engine](#) (page 315) documentation for more info.

If no engine block is specified, the [ForceField engine](#) (with the UFF force field) will be used as default.

Generators

Three different methods of conformer generation are implemented: RDKit^{Page 344, 1}, CREST^{Page 344, 2} and ANNEALING. The **default** generation method is **RDKit**.

You can select which method to use in the `Generator` block:

```
Generator
  Method [RDKit | CREST | ANNEALING]
End
```

Generator**Type**

Block

Description

Options for the conformer generator.

Method**Type**

Multiple Choice

Default value

RDKit

Options

[RDKit, CREST, ANNEALING]

GUI name

Generator method

Description

Method used to generate the conformers.

The RDKit generator is based on random distance matrix method. This is the recommended (and default) method.

The CREST Generator uses a multi-step workflow with meta-dynamics simulations to explore the conformers space of a molecule. This can be a powerful conformer search method, but it is generally computationally expensive compared to the RDKit generator.

The ANNEALING generator performs a simulated annealing simulation to explore conformer space.

Input options for the various methods can be specified in their respective input blocks. See the section [Other input options](#) (page 353) for the complete set of input options.

Conformers Equivalence

An important aspect of conformers generation is the reduction of a set of geometries into a set of unique conformers. As the Conformer tool generates new potential conformers, it needs to be able to determine whether a structure is a new conformer (in which case it is added to the list of unique conformers) or whether is simply a slightly different geometry of an already observed conformer (in which case, the structure is discarded).

The conformer tool therefore needs a procedure that, given two geometries, is able to discriminate between the following situations:

- the two geometries correspond to the same conformer
- the two geometries correspond to distinct conformers

In the `Equivalence` block various options related to this equivalence procedure can be specified.

```
Equivalence
  Method [AMS | TFD | RMSD | CREST]
End
```

Equivalence

Type

Block

Description

Options for the procedure determining whether two structures are equivalent or distinct conformers.

Method

Type

Multiple Choice

Default value

CREST

Options

[AMS, TFD, RMSD, CREST]

GUI name

Equivalence method

Description

Method used to determine (and filter out) equivalent conformers.

The CREST equivalence method relies on rotational constants comparisons. For this reason, conformers with the same rotational constants (such as mirror images) will be considered equivalent conformers.

The AMS equivalence method uses a distance matrix and dihedrals to compare conformers. This equivalence method can be computationally expensive for large molecules.

The TFD equivalence method uses the Torsion Fingerprint Difference as implemented in RDKit.

The RMSD equivalence method uses the RDKit GetBestRMS implementation.

Whenever the equivalence procedure is employed, details about the detected duplicates can be written to file with the `LogDuplicates` keyword.

```
Equivalence
  LogDuplicates Yes/No
End
```

Equivalence**Type**

Block

Description

Options for the procedure determining whether two structures are equivalent or distinct conformers.

LogDuplicates**Type**

Bool

Default value

No

Description

If set to True, print information on equivalence to logfile and standard output.

Input options for the various methods can be specified in their respective input blocks. See the section *Other input options* (page 353) for the complete set of input options.

Filtering

The Conformers tool can automatically filter out conformers based on some criteria, e.g. it will discard high energy conformers.

The filtering options can be specified in the `Filter` block:

```
Filter
  MaxEnergy float
  RemoveNonMinima Yes/No
End
```

Filter**Type**

Block

Description

Options for the conformer filtering.

MaxEnergy**Type**

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers. If the relative energy of a conformer with respect to the lowest conformer is larger than this value, the conformer will be discarded.

RemoveNonMinima**Type**

Bool

Default value

No

Description

For the final set of conformers, explicitly check that the geometry corresponds to a local minimum, and remove it if it does not. Note: this will run a PES point characterization, which can be computationally expensive!

Constraints, Restraints, Walls

The Conformers tool can be used in combination with Constraints (*constraints* (page 55)), Restraints (*restraints* (page 332)), and EngineAddons (*engine addons* (page 325)).

The constraints fix the specified degrees of freedom to certain values, while the restraints restricts the specified degrees of freedom to be near the specified values using an added potential energy function. The engine add-ons cover a wide range of functionality, but the most important one for conformers is the `WallPotential`, which adds a spherical repulsive potential around the origin, trapping the atoms inside.

The `RDKit Generate` method consists of two steps:

1. Call to `RDKit ETKDG` method for initial conformer generation
2. Geometry optimization of all conformer candidates with an AMS engine

Step 1 only accepts fixed atom constraints. Distance, angle, and dihedral constraints will be ignored during the initial conformer generation, and reapplied during the subsequent geometry optimization.

The other `Generate` methods of the Conformers tool make use of both molecular dynamics simulations (`Crest` and `Annealing`) and geometry optimizations. Not all constraints can be used in molecular dynamics simulations, so where possible constraints are automatically converted to the corresponding restraints.

Conformer Generation with Multiple Molecules

The wall potential described above can be used in combination with the `Crest` and `Annealing` methods to generate conformers for systems with multiple molecules. A simple example below demonstrates this for an acetic acid molecule in the presence of water. The radius of the spherical constraint is manually set to 4.5 Angstrom.

```
#!/bin/sh

"$AMSBIN/conformers" << eor
  Engine ForceField
  EndEngine
  Generator
    Method CREST
  End
  System
    Atoms
      C -5.168356138856709 -0.9040308913456915 -0.14509537743978
      C -3.674177007131515 -0.8549162177628133 -0.1466485108795087
      O -3.036130963738181 0.3677379219191289 -0.1592569297095029
      O -3.033827233992448 -1.897394223042741 -0.1762700959655665
      H -5.578445514437495 0.05058176564435996 0.2458678935590599
      H -5.524620190394747 -1.732922198697688 0.5010944819578638
      H -5.536566707641177 -1.061832129921519 -1.180309622581397
      H -2.058681010424327 0.197909193054116 -0.1773409518219506
      O -4.51655946130152 -4.444022963974381 0.9796661086035751
      H -4.406160106170592 -3.748388119201934 1.675984760241468
      H -5.428884582974359 -4.296939008632186 0.6238436183248668
    End
  End
```

(continues on next page)

(continued from previous page)

```

      BondOrders
      1 2 1.0
      1 5 1.0
      1 6 1.0
      1 7 1.0
      2 3 1.0
      2 4 2.0
      3 8 1.0
      9 10 1.0
      9 11 1.0
    End
  End
  EngineAddons
    WallPotential
      Enabled Yes
      Radius 4.5
    End
  End
eor

```

Examples

Workflow

During the conformers generation (i.e. the “generate” *task* (page 345)), a large number of atomistic calculations will be executed under the hood. This can get computationally very expensive if an accurate *engine* (page 347) (such as ADF) is used. This is why the default engine is the computationally fast (but not very accurate) *ForceField* engine.

By combining the *Generate*, *Optimize* and *Score* tasks you can build a simple workflow to get accurate geometries and energies while keeping the computational cost to a minimum.

A typical workflow might look like this:

1. Generate a set of conformers using Task *Generate* with the fast but approximate *ForceField* engine
2. Re-optimize the geometries generated in the previous step using Task *Optimize* with a more accurate engine (e.g. *DFTB*)
3. Re-score the conformers using the using Task *Score* with an accurate engine (e.g. *ADF*)

```

#!/bin/sh

# Simple workflow:
# - generate an initial set of 1-bromo-3-chloropropane conformers with UFF_
→ (ForceField engine)
# - re-optimize the conformers with GFN1-xTB (DFTB engine)
# - re-score the GFN1-xTB-optimized conformers using DFT energies (ADF engine)

AMS_JOBNAME=generate_uff $AMSBIN/conformers << EOF

  Task Generate

  System
    Atoms
      Br      -4.16969843      2.48953245      -0.00000000
      C      -2.69926018      3.75062981      -0.00000000

```

(continues on next page)

(continued from previous page)

```

      C      -1.36372383      3.00977020      -0.00000000
      C      -0.19648214      3.99512105      0.00000000
      Cl     1.35297276      3.11657146      0.00000000
      H      -2.78129583      4.39009878      0.90457474
      H      -2.78129583      4.39009878     -0.90457474
      H      -1.30112035      2.36381643     -0.90271681
      H      -1.30112035      2.36381643      0.90271681
      H      -0.24025734      4.63864134      0.90438917
      H      -0.24025734      4.63864134     -0.90438917
    End
  End

  Engine ForceField
    Type UFF
  EndEngine
EOF

AMS_JOBNAME=optimize_gfn1xtb $AMSBIN/conformers << EOF

  Task Optimize

  InputConformersSet generate_uff.results/conformers.rkf

  Engine DFTB
    Model GFN1-xTB
  EndEngine
EOF

AMS_JOBNAME=score_dft $AMSBIN/conformers << EOF

  Task Score

  InputConformersSet optimize_gfn1xtb.results/conformers.rkf

  # Here we use the default ADF settings. For real calculations, you
  # should use a larger basis set (e.g. TZP) and a better XC functional

  Engine ADF
  EndEngine
EOF

echo "Final conformers:"
cat score_dft.results/conformers.xyz

```

References

Other input options

Here we document more advanced input options for the Conformer tool. See the section *Main Input Options* (page 345) for the most important input option.

Output

Type

Block

Description

Options regarding the output and result files.

KeepWorkDir

Type

Bool

Default value

No

Description

Do not remove the working directories after the conformer generation is finished.

rkf

Type

Bool

Default value

Yes

Description

Save the final conformers in .rkf format. The file 'conformers.rkf' will be located in the results directory. You can visualize this file using the AMSMovie GUI module.

sdf

Type

Bool

Default value

Yes

Description

Save the final conformers in .sdf format. The file 'conformers.sdf' will be located in the results directory.

xyz

Type

Bool

Default value

Yes

Description

Save the final conformers in .xyz format. The file 'conformers.xyz' will be located in the results directory.

RNGSeed

Type

Integer

Description

Initial seed for the (pseudo)random number generator. If this is unset, the generator will be seeded randomly from external sources of entropy and the generated conformers will be non-deterministic.

Generator**Type**

Block

Description

Options for the conformer generator.

ANNEALING**Type**

Block

Description

Options for the annealing generator. This generator creates conformers by performing a simulated annealing simulation.

MolecularDynamics**Type**

Block

Description

Configures molecular dynamics (with the velocity-Verlet algorithm) with and without thermostats. This block allows to specify the details of the molecular dynamics calculation. Default values will be ignored.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps. Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

InitialVelocities**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod**Type**

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature**Type**

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type**Type**

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values**Type**

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

NSteps**Type**

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

Preserve

Type

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and 3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass**Type**

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum**Type**

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

System

Type

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities**Type**

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

Restart**Type**

String

GUI name

Restart from

Description

The path to the ams.rkf file from which to restart the simulation.

Shake**Type**

Block

Description

Parameters of the Shake/Rattle algorithm.

A11**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, “All triangles H C H” in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2

Type

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV

Type

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations

Type

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates

Type

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Thermostat**Type**

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply**Type**

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)
- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

TimeStep**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

Trajectory**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

EngineResultsFreq**Type**

Integer

GUI name

Engine results frequency

Description

Write MDStep*.rkf files with engine-specific results once every N steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of Checkpoint%Frequency.

ExitConditionFreq**Type**

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every N steps. By default this is done every SamplingFreq steps.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N steps. By default this is done every SamplingFreq steps.

SamplingFreq**Type**

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds**Type**

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges**Type**

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

Temperatures**Type**

Float List

Default value

[298.0, 798.0]

Unit

Kelvin

Description

The minimum and maximum temperature of the annealing simulation. The simulation will start at the highest temperature, and cool down to the lowest.

UseShake**Type**

Bool

Default value

No

Description

Constrain all -H bonds with shake. If turned on, the MD timestep is automatically increased.

CREST**Type**

Block

Description

Options for the CREST generator. The CREST generator performs a set of metadynamics simulations (using the METADYNAMICS generator), then a set of MD simulations (using the MD expander), and finally it does a combinatorial expansion of the generated conformers using the GC expander. This sequence is repeated in an iterative fashion until the lowest energy conformer no longer changes.

ConvergenceQualityCrude

Type

Multiple Choice

Default value

None

Options

[Normal, Good, VeryGood, Excellent, None]

Description

The tightness of the convergence of the crude geometry pre-optimizations. If set to none it will be selected two levels below ConvergenceQuality.

ConvergenceQualityTight**Type**

Multiple Choice

Default value

None

Options

[Normal, Good, VeryGood, Excellent, None]

Description

The tightness of the convergence of the final geometry optimizations. If set to none it will be selected the same as ConvergenceQuality.

GCStep**Type**

Bool

Default value

Yes

Description

Whether or not to include the combinatorial expansion of the conformers using the GC Generator. For big systems this step can be very time consuming. By default it is set to True.

NCycles**Type**

Integer

Default value

10

Description

The maximum number of CREST cycles (by default the number is 10). If the lowest conformer energy converges before then, Crest exits.

UseShake**Type**

Bool

Default value

Yes

Description

Whether or not SHAKE should be turned on in the MD and Metadynamics simulations. If this is turned on, the MD timestep is automatically increased (from 2 to 5 fs).

METADYNAMICS**Type**

Block

Description

Produces conformers by running a set of CREST-RMSD metadynamics simulations with different biases, extracting snapshots, and optimizing those.

ConvergenceQualityCrude**Type**

Multiple Choice

Default value

none

Options

[normal, good, verygood, excellent, none]

Description

The tightness of the convergence of the crude geometry pre-optimizations. If set to none it will be selected two levels below ConvergenceQuality.

MolecularDynamics**Type**

Block

Description

Configures molecular dynamics (with the velocity-Verlet algorithm) with and without thermostats. This block allows to specify the details of the molecular dynamics calculation. Default values will be ignored.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps. Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

InitialVelocities**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod**Type**

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature**Type**

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type**Type**

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values**Type**

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

NSteps**Type**

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

Preserve

Type

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and 3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass**Type**

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum**Type**

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

System

Type

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities**Type**

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

Restart**Type**

String

GUI name

Restart from

Description

The path to the ams.rkf file from which to restart the simulation.

Shake**Type**

Block

Description

Parameters of the Shake/Rattle algorithm.

A11**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, “All triangles H C H” in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2

Type

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV

Type

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations

Type

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates

Type

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Thermostat**Type**

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply**Type**

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)
- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

TimeStep**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

Trajectory**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

EngineResultsFreq**Type**

Integer

GUI name

Engine results frequency

Description

Write MDStep*.rkf files with engine-specific results once every N steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of Checkpoint%Frequency.

ExitConditionFreq**Type**

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every N steps. By default this is done every SamplingFreq steps.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N steps. By default this is done every SamplingFreq steps.

SamplingFreq**Type**

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds**Type**

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges**Type**

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

NCycles**Type**

Integer

Default value

5

Description

The maximum number of cycles of metadynamics simulations. The generator stops when either this number is reached, or the conformer set is stable.

NWidthsHeights**Type**

Integer List

Default value

[4, 3]

Description

The number of different Gaussian widths and heights respectively used in the metadynamics simulations. By default 4 different widths are used and 3 different heights, resulting in 12 different metadynamics simulations.

SimulationSuccessFraction**Type**

Float

Default value

0.4

Description

The fraction of planned metadynamics steps that has to have succeeded in order for the metadynamics iteration to be considered a success.

UseShake**Type**

Bool

Default value

No

Description

Constrain all -H bonds with shake. If turned on, the MD timestep is automatically increased.

MaxEnergy**Type**

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers. If the relative energy of a conformer with respect to the lowest conformer is larger than this value, the conformer will be discarded.

Method**Type**

Multiple Choice

Default value

RDKit

Options

[RDKit, CREST, ANNEALING]

GUI name

Generator method

Description

Method used to generate the conformers.

The RDKit generator is based on random distance matrix method. This is the recommended (and default) method.

The CREST Generator uses a multi-step workflow with meta-dynamics simulations to explore the conformers space of a molecule. This can be a powerful conformer search method, but it is generally computationally expensive compared to the RDKit generator.

The ANNEALING generator performs a simulated annealing simulation to explore conformer space.

Preoptimization**Type**

Block

Description

If this block is enabled geometries will be preoptimized. After preoptimization the high energy conformers will be discarded, and then from the remaining set the unoptimized geometries will be optimized at higher level. This is to prevent the preoptimizer from collapsing different conformers into a single false minimum. As a result, preoptimization is only useful if MaxEnergy is chosen low.

Enable**Type**

Bool

Default value

No

Description

Perform preoptimization at a low level of accuracy.

Engine**Type**

Block

Description

The engine specifics to be used for preoptimization.

PreoptFactor**Type**

Integer

Default value

2

Description

This factor is multiplied with MaxEnergy, to determine which high energy conformers can be discarded after preoptimization.

RDKit**Type**

Block

Description

Options for the RDKit generator. This generator produces initial guesses for conformers using the RDKit ETKDG method, followed by AMS geometry optimizations.

InitialNConformers**Type**

Integer

GUI name

Initial no. of conformers

Description

Number of geometries initially created by RDKit, before AMS geometry optimization and filtering. If not set, the number will be automatically set, based on the number of rotational bonds.

MaxConfs**Type**

Integer

Default value

5000

Description

If InitialNConformers is not set, then the number of conformers will be automatically set, with a maximum of MaxConfs.

MinConfs**Type**

Integer

Default value

10

Description

If InitialNConformers is not set, then the number of conformers will be automatically set, with a minimum of MinConfs.

NconfsEstimationFactor**Type**

Integer

Default value

100

Description

If InitialNConformers is not set, then the number of conformers will be automatically set based on the number of rotational bonds. The resulting number is then multiplied by this factor (default: 100), to ensure that enough conformers will be created.

RDKitETKDG**Type**

Block

Description

Settings for the call to RDKits ETKDG conformer generator tool.

BestRMSDThreshold**Type**

Float

Default value

-1.0

Description

After ETKDG conformer generation by RDKit, RDKit can be used to remove duplicates via the BestRMS algorithm. This filter does exactly the same as the RMSD equivalence detector in the Equivalence block.

ConstrainedAtoms**Type**

Integer List

Description

The indices of the atoms to constrain during ETKDG conformer generation.

Forcefield**Type**

Multiple Choice

Default value

None

Options

[None, UFF, MMFF]

Description

The name of the RDKit forcefield to use for geometry optimization at the end of ETKDG conformer generation (by default no geometry optimization is performed). Using the RDKit internal optimization may make the subsequent geometry optimizations with AMS faster.

Parallel

Type

Bool

Default value

No

Description

Experimental: Parallelize the RDKit generation step by calling the RDKit conformer generation method in parallel from multiple processes.

RMSDThreshold**Type**

Float

Default value

-1.0

Description

Root Mean Square deviation threshold for removing similar/equivalent conformations during the RDKit ETKDG procedure. By default there is no pruning (value: -1).

UseExpTorsionAnglePrefs**Type**

String

Default value

default

Description

Impose experimental torsion angle preferences in RDKit ETKDG conformer generation. By default the RDKit version determines whether or not to switch this on.

RNGSeed**Type**

Integer

Description

Initial guesses for conformers can be randomly generated by RDKit, using the ETKDG algorithm. For reproducibility, a random number seed can be provided here.

TORSION**Type**

Block

Description

Options for the TorsionGenerator, which generates geometries by enumerative rotation around rotatable bonds. This is the slowest of all generator, and quickly becomes infeasible for large systems. Currently does not work for systems with interconnected rings.

Dtheta**Type**

Float

Default value

60.0

Description

The angle over which the bonds are rotated, in order to create a new conformer.

Expander**Type**

Block

Description

Options for the conformer expander. The Expander expands an existing conformer set, by adding new conformers to it. The new conformers are generated from the original conformers in the set. Unlike the generators, the outcome of an expander is therefore very dependent on the input conformations.

The GC generator uses a genetic algorithm to create a combinatorial expansion by combining local substructures.

The MD expander start MD simulations from the conformers in the set, and extracts snapshots to create new conformers. Both these expanders are part of the CREST generator.

GC**Type**

Block

Description

Options for the genetic algorithm for combinatorial expansion of conformer geometries. This generator only works if a non-zero set of conformers is already provided.

MaxGCenergy**Type**

Float

Default value

6.0

Description

The maximum energy (relative to the lowest in the set) of the conformers we are going to use for expansion. The default is 6.0 kcal/mol, but if MaxEnergy was set, then that value is used.

Parallel**Type**

Bool

Default value

Yes

Description

Determines if the combinatorial expansion of conformers should be performed in parallel or not (default: True).

RMSDThreshold**Type**

Float

Default value

0.25

Description

Newly generated geometries are only considered unique if their RMSD from all other newly generated geometries is larger than this threshold.

MD

Type

Block

Description

Produces conformers by running a set of MD simulations at different elevated temperatures, extracting snapshots, and optimizing those.

MolecularDynamics**Type**

Block

Description

Configures molecular dynamics (with the velocity-Verlet algorithm) with and without thermostats. This block allows to specify the details of the molecular dynamics calculation. Default values will be ignored.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps. Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

InitialVelocities**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod**Type**

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature**Type**

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type**Type**

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values**Type**

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

NSteps**Type**

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

Preserve**Type**

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and

3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass

Type

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum

Type

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

Print

Type

Block

Description

This block controls the printing of additional information to stdout.

System

Type

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities

Type

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

Restart

Type

String

GUI name

Restart from

Description

The path to the ams.rkf file from which to restart the simulation.

Shake**Type**

Block

Description

Parameters of the Shake/Rattle algorithm.

A11**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, "All triangles H C H" in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2**Type**

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV**Type**

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations**Type**

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates**Type**

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Thermostat**Type**

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply**Type**

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)
- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

TimeStep**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

Trajectory**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

EngineResultsFreq**Type**

Integer

GUI name

Engine results frequency

Description

Write MDStep*.rkf files with engine-specific results once every N steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of Checkpoint%Frequency.

ExitConditionFreq**Type**

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every N steps. By default this is done every SamplingFreq steps.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N steps. By default this is done every SamplingFreq steps.

SamplingFreq**Type**

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds

Type

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges**Type**

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

Ngeoms**Type**

Integer

Default value

4

Description

At each temperature, MD simulations are started from Ngeoms different starting geometries. The starting geometries are extracted from the provided conformer set. If the conformer set is empty, then no more than a single geometry per temperature can be provided, limiting the total number of MD simulations.

Temperatures**Type**

Float List

Default value

[400.0, 500.0]

Unit

Kelvin

Description

The list of different temperatures at which MD simulations are run.

UseShake**Type**

Bool

Default value

No

Description

Constrain all -H bonds with shake. If turned on, the MD timestep is automatically increased.

MaxEnergy**Type**

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers. If the relative energy of a conformer with respect to the lowest conformer is larger than this value, the conformer will be discarded.

Method**Type**

Multiple Choice

Default value

GC

Options

[MD, GC]

GUI name

Generator method

Description

Method used to generate the conformers.

Preoptimization**Type**

Block

Description

If this block is enabled geometries will be preoptimized. After preoptimization the high energy conformers will be discarded, and then from the remaining set the unoptimized geometries will be optimized at higher level. This is to prevent the preoptimizer from collapsing different conformers into a single false minimum. As a result, preoptimization is only useful if MaxEnergy is chosen low.

Enable**Type**

Bool

Default value

No

Description

Perform preoptimization at a low level of accuracy.

Engine**Type**

Block

Description

The engine specifics to be used for preoptimization.

PreoptFactor**Type**

Integer

Default value

2

Description

This factor is multiplied with MaxEnergy, to determine which high energy conformers can be discarded after preoptimization.

RDKitETKDG**Type**

Block

Description

Settings for the call to RDKits ETKDG conformer generator tool.

BestRMSDThreshold**Type**

Float

Default value

-1.0

Description

After ETKDG conformer generation by RDKit, RDKit can be used to remove duplicates via the BestRMS algorithm. This filter does exactly the same as the RMSD equivalence detector in the Equivalence block.

Forcefield**Type**

Multiple Choice

Default value

None

Options

[None, UFF, MMFF]

Description

The name of the RDKit forcefield to use for geometry optimization at the end of ETKDG conformer generation (by default no geometry optimization is performed). Using the RDKit internal optimization may make the subsequent geometry optimizations with AMS faster.

Parallel**Type**

Bool

Default value

No

Description

Experimental: Parallelize the RDKit generation step by calling the RDKit conformer generation method in parallel from multiple processes.

RMSDThreshold**Type**

Float

Default value

-1.0

Description

Root Mean Square deviation threshold for removing similar/equivalent conformations during the RDKit ETKDG procedure. By default there is no pruning (value: -1).

UseExpTorsionAnglePrefs**Type**

String

Default value

default

Description

Impose experimental torsion angle preferences in RDKit ETKDG conformer generation. By default the RDKit version determines whether or not to switch this on.

RNGSeed**Type**

Integer

Description

Initial guesses for conformers can be randomly generated by RDKit, using the ETKDG algorithm. For reproducibility, a random number seed can be provided here.

Equivalence**Type**

Block

Description

Options for the procedure determining whether two structures are equivalent or distinct conformers.

AMS**Type**

Block

Description

Options for the AMS method of checking equivalence. This method uses the atomic distance matrices and the torsion angles between heavy atoms to determine if conformer candidates are duplicates.

DihedralThreshold**Type**

Float

Default value

30.0

Description

Maximum difference a dihedral can have for a conformer to be considered a duplicate.

DistanceThreshold**Type**

Float

Default value

0.1

Description

Maximum difference a distance between two atoms can have for a conformer to be considered a duplicate.

EnergyThreshold**Type**

Float

Default value

0.2

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

IrrelevantAtoms**Type**

Integer List

Description

To detect equivalence, only a subset of atoms is used. The atoms that are excluded from equivalence comparison should be specified here. By default only non-hydrogen atoms will be used for the comparison. Numbering starts at 0.

AcceptAll**Type**

Bool

Default value

No

Description

If set to True, add any candidate to the set without checks of connectivity changes, stereo- or cis/trans isomerization, or duplication.

AcceptIsomers**Type**

Bool

Default value

No

Description

If set to True, perform all checks of a new conformer candidate, except for the stereo- or cis/trans isomerization check.

CREST**Type**

Block

Description

Options for the CREST method of checking equivalence. This method uses the rotational constants of conformer candidates to determine if they are duplicates.

EnergyThreshold**Type**

Float

Default value

0.05

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

RMSDThreshold**Type**

Float

Default value

0.125

Description

Threshold for the RMSD between two conformers that determines if they are duplicates or rotamers (according to the CREST rotamer definition.)

ScaledRotationalConstantSettings**Type**

Block

Description

By default, the equivalence of two geometries is determined mainly by comparing the rotational constants, and weighing the difference based on the average anisotropy of the two systems. This procedure has several settings that can be user defined.

RotationalConstantThreshold**Type**

Float

Default value

0.003

Description

Threshold for the difference in rotational constants that determines if two geometries are duplicates. The threshold is weighed by the anisotropy of the systems. Note: in the grimme code they use 0.01 as `bconst_threshold`, but this leads to a lot of misclassifications (i.e. different conformers are classified as equivalent rotamers) So, here we use a smaller default value.

CheckForDuplicates**Type**

Bool

Default value

Yes

Description

If set to True, check any new conformer candidate for duplication, and only accept unique conformers. If set to False, accept duplicates into the set.

LogDuplicates**Type**

Bool

Default value

No

Description

If set to True, print information on equivalence to logfile and standard output.

Method**Type**

Multiple Choice

Default value

CREST

Options

[AMS, TFD, RMSD, CREST]

GUI name

Equivalence method

Description

Method used to determine (and filter out) equivalent conformers.

The CREST equivalence method relies on rotational constants comparisons. For this reason, conformers with the same rotational constants (such as mirror images) will be considered equivalent conformers.

The AMS equivalence method uses a distance matrix and dihedrals to compare conformers. This equivalence method can be computationally expensive for large molecules.

The TFD equivalence method uses the Torsion Fingerprint Difference as implemented in RDKit.

The RMSD equivalence method uses the RDKit GetBestRMS implementation.

RMSD

Type

Block

Description

Options for the RMSD method of checking equivalence. This method uses the RDKit implementation of GetBestRMS, which enumerates over atomic permutations for pairs of geometries to detect duplicates based on the RMSD value.

EnergyThreshold

Type

Float

Default value

0.05

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

RMSDThreshold

Type

Float

Default value

0.125

Description

Threshold on the RMSD difference to determine if two geometries represent the same conformer. This value is in Angstrom.

Reorder

Type

Bool

Default value

Yes

Description

Reorder conformers based on energy, whenever a new conformer is added.

TFD

Type

Block

Description

Options for the TFD method of checking equivalence. This method uses the Torsion Finger Print method to determine if two conformer candidates are duplicates.

EnergyThreshold**Type**

Float

Default value

0.05

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

TFDThreshold**Type**

Float

Default value

0.05

Description

Threshold on the torsion fingerprint difference to determine if two geometries represent the same conformer. This value is unit-less.

GeometryOptimization**Type**

Block

Description

Some options / details regarding the optimization procedure.

ConvergenceQuality**Type**

Multiple Choice

Default value

VeryGood

Options

[Normal, Good, VeryGood, Excellent]

GUI name

Convergence

Description

The tightness of the convergence of the geometry optimizations. Lower quality levels may lead to badly converged geometries being classified as distinct conformers.

GeometryOptimization**Type**

Block

Description

Configures details of the geometry optimization and transition state searches.

CalcPropertiesOnlyIfConverged

Type

Bool

Default value

Yes

Description

Compute the properties requested in the 'Properties' block, e.g. Frequencies or Phonons, only if the optimization (or transition state search) converged. If False, the properties will be computed even if the optimization did not converge.

Convergence**Type**

Block

Description

Convergence is monitored for up to 4 quantities: the energy change, the Cartesian gradients, the Cartesian step size, and for lattice optimizations the stress energy per atom. Convergence criteria can be specified separately for each of these items.

Energy**Type**

Float

Default value

1e-05

Unit

Hartree

Value Range

value > 0

GUI name

Energy convergence

Description

The criterion for changes in the energy. The energy is considered converged when the change in energy is smaller than this threshold times the number of atoms.

Gradients**Type**

Float

Default value

0.001

Unit

Hartree/Angstrom

Value Range

value > 0

GUI name

Gradient convergence

Description

Threshold for nuclear gradients.

Quality

Type

Multiple Choice

Default value

Custom

Options

[VeryBasic, Basic, Normal, Good, VeryGood, Custom]

GUI name

Convergence

Description

A quick way to change convergence thresholds: 'Good' will reduce all thresholds by an order of magnitude from their default value. 'VeryGood' will tighten them by two orders of magnitude. 'Basic' and 'VeryBasic' will increase the thresholds by one or two orders of magnitude respectively.

Step**Type**

Float

Default value

0.01

Unit

Angstrom

Value Range

value > 0

GUI name

Step convergence

Description

The maximum Cartesian step allowed for a converged geometry.

StressEnergyPerAtom**Type**

Float

Default value

0.0005

Unit

Hartree

Value Range

value > 0

Description

Threshold used when optimizing the lattice vectors. The stress is considered 'converged' when the maximum value of $\text{stress_tensor} * \text{cell_volume} / \text{number_of_atoms}$ is smaller than this threshold (for 2D and 1D systems, the cell_volume is replaced by the cell_area and cell_length respectively).

CoordinateType**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Delocalized, Cartesian]

GUI name

Optimization space

Description

Select the type of coordinates in which to perform the optimization. 'Auto' automatically selects the most appropriate CoordinateType for a given Method.

If 'Auto' is selected, Delocalized coordinates will be used for the Quasi-Newton method, while Cartesian coordinates will be used for all other methods.

Dimer**Type**

Block

Description

Options for the Dimer method for transition state search.

AngleThreshold**Type**

Float

Default value

1.0

Unit

Degree

Description

The rotation is considered converged when the the rotation angle falls below the specified threshold.

DimerDelta**Type**

Float

Default value

0.01

Unit

Angstrom

Description

Euclidian distance between the midpoint and the endpoint.

ExtrapolateForces**Type**

Bool

Default value

Yes

Description

Set to false to call engine to calculate forces at the extrapolated rotation angle instead of extrapolating them.

LBFGSMaxVectors**Type**

Integer

Default value

10

Description

Max number of vectors for the L-BFGS algorithm to save.

MaxRotationIterations**Type**

Integer

Default value

10

Description

Maximum number of rotation iterations for a single translation step.

Region**Type**

String

Default value

*

Description

Include only atoms of the specified region(s) in the rotations, which allows searching for a transition state involving selected atoms only.

RotationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during rotation iterations.

TranslationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during translation iterations.

EngineAutomations

Type

Block

Description

The optimizer can change some settings of the engine, based for instance on the error.

The idea is to allow the engine to be a bit quicker at the start, and more accurate towards the end.

Automations are always engine specific.

Enabled**Type**

Bool

Default value

Yes

Description

Whether or not automations are enabled at all.

Gradient**Type**

Block

Recurring

True

Description

A gradient-based automation.

FinalValue**Type**

Float

Description

This value will be used whenever the gradient is less than GradientLow

HighGradient**Type**

Float

Default value

1.0

Unit

Hartree/Angstrom

Description

Defines a large gradient. When the actual gradient is between GradientHigh and GradientLow a linear interpolation scheme is used for kT (on a log scale).

InitialValue**Type**

Float

Description

This value will be used at the first geometry, and whenever the gradient is higher than GradientHigh

LowGradient**Type**

Float

Default value

1.0

Unit

Hartree/Angstrom

Description

Defines a small gradient, see GradientHigh

UseLogInterpolation**Type**

Bool

Default value

Yes

Description

Whether to use interpolation on a log (y) scale or not

Variable**Type**

String

Default value**Description**

variable to be tweaked for the engine.

Iteration**Type**

Block

Recurring

True

Description

Geometry step based automation.

FinalValue**Type**

Float

Description**FirstIteration****Type**

Integer

Default value

1

Description

When the actual gradient is between the first and last iteration, a linear interpolation is used.

InitialValue

Type

Float

Description

This value will be used when the iteration number is smaller or equal to FirstIteration

LastIteration**Type**

Integer

Default value

10

Description

Where the automation should reach the FinalValue

UseLogInterpolation**Type**

Bool

Default value

Yes

Description

Whether to use interpolation on a log (y) scale or not

Variable**Type**

String

Default value**Description**

variable to be tweaked for the engine.

FIRE**Type**

Block

Description

This block configures the details of the FIRE optimizer. The keywords name correspond the the symbols used in the article describing the method, see PRL 97, 170201 (2006).

AllowOverallRotation**Type**

Bool

Default value

Yes

Description

Whether or not the system is allowed to freely rotate during the optimization. This is relevant when optimizing structures in the presence of external fields.

AllowOverallTranslation**Type**

Bool

Default value

No

Description

Whether or not the system is allowed to translate during the optimization. This is relevant when optimizing structures in the presence of external fields.

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

Map the atoms to the central cell at each geometry step.

NMin**Type**

Integer

Default value

5

Description

Number of steps after stopping before increasing the time step again.

alphaStart**Type**

Float

Default value

0.1

Description

Steering coefficient.

dtMax**Type**

Float

Default value

1.0

Unit

Femtoseconds

Description

Maximum time step used for the integration. For ReaxFF and APPLE&P, this value is reduced by 50%.

dtStart**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

Initial time step for the integration.

fAlpha**Type**

Float

Default value

0.99

Description

Reduction factor for the steering coefficient.

fDec**Type**

Float

Default value

0.5

Description

Reduction factor for reducing the time step in case of uphill movement.

fInc**Type**

Float

Default value

1.1

Description

Growth factor for the integration time step.

strainMass**Type**

Float

Default value

0.5

Description

Fictitious relative mass of the lattice degrees of freedom. This controls the stiffness of the lattice degrees of freedom relative to the atomic degrees of freedom, with smaller values resulting in a more aggressive optimization of the lattice.

HessianFree**Type**

Block

Description

Configures details of the Hessian-free (conjugate gradients or L-BFGS) geometry optimizer.

Step**Type**

Block

Description**MaxCartesianStep****Type**

Float

Default value

0.1

Unit

Angstrom

Description

Limit on a single Cartesian component of the step.

MinRadius**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Minimum value for the trust radius.

TrialStep**Type**

Float

Default value

0.0005

Unit

Angstrom

Description

Length of the finite-difference step when determining curvature. Should be smaller than the step convergence criterion.

TrustRadius**Type**

Float

Default value

0.2

Unit

Angstrom

Description

Initial value of the trust radius.

InitialHessian**Type**

Block

Description

Options for initial model Hessian when optimizing systems with the Quasi-Newton method.

File**Type**

String

GUI name

Initial Hessian from

Description

KF file containing the initial Hessian (or the results dir. containing it). This can be used to load a Hessian calculated in a previously with the [Properties%Hessian] keyword.

Type**Type**

Multiple Choice

Default value

Auto

Options

[Auto, UnitMatrix, Swart, FromFile, Calculate, CalculateWithFastEngine]

GUI name

Initial Hessian

Description

Select the type of initial Hessian. Auto: let the program pick an initial model Hessian. UnitMatrix: simplest initial model Hessian, just a unit matrix in the optimization coordinates. Swart: model Hessian from M. Swart. FromFile: load the Hessian from the results of a previous calculation (see InitialHessian%File). Calculate: compute the initial Hessian (this may be computationally expensive and it is mostly recommended for TransitionStateSearch calculations). CalculateWithFastEngine: compute the initial Hessian with a faster engine.

KeepIntermediateResults**Type**

Bool

Default value

No

Description

Whether the full engine result files of all intermediate steps are stored on disk. By default only the last step is kept, and only if the geometry optimization converged. This can easily lead to huge amounts of data being stored on disk, but it can sometimes be convenient to closely monitor a tricky optimization, e.g. excited state optimizations going through conical intersections, etc. ...

MaxIterations**Type**

Integer

Value Rangevalue ≥ 0 **GUI name**

Maximum number of iterations

Description

The maximum number of geometry iterations allowed to converge to the desired structure.

MaxRestarts**Type**

Integer

Default value

0

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small displacement along the imaginary vibrational mode. In case the restarted optimization again does not find a minimum, this can happen multiple times in succession. This keyword sets the maximum number of restarts. The default value is 0, so the automatic restarting is disabled by default.

Method**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Quasi-Newton, FIRE, L-BFGS, ConjugateGradients, Dimer]

GUI name

Optimization method

Description

Select the optimization algorithm employed for the geometry relaxation. Currently supported are:

the Hessian-based Quasi-Newton-type BFGS algorithm,

the fast inertial relaxation method (FIRE),

the limited-memory BFGS method,

and the conjugate gradients method. The default is to choose an appropriate method automatically based on the engine's speed, the system size and the supported optimization options.

OptimizeLattice**Type**

Bool

Default value

No

Description

Whether to also optimize the lattice for periodic structures. This is currently supported with the Quasi-Newton, FIRE, and L-BFGS optimizers.

PretendConverged**Type**

Bool

Default value

No

Description

Normally a non-converged geometry optimization is considered an error. If this keyword is set to True, the optimizer will only produce a warning and still claim that the optimization is

converged. (This is mostly useful for scripting applications, where one might want to consider non-converged optimizations still successful jobs.)

Quasi-Newton**Type**

Block

Description

Configures details of the Quasi-Newton geometry optimizer.

MaxGDIISVectors**Type**

Integer

Default value

0

Description

Sets the maximum number of GDIIS vectors. Setting this to a number >0 enables the GDIIS method.

Step**Type**

Block

Description**TrustRadius****Type**

Float

Description

Initial value of the trust radius.

VaryTrustRadius**Type**

Bool

Description

Whether to allow the trust radius to change during optimization. By default True during energy minimization and False during transition state search.

UpdateTSVectorEveryStep**Type**

Bool

Default value

Yes

GUI name

Update TSRC vector every step

Description

Whether to update the TS reaction coordinate at each step with the current eigenvector.

RestartDisplacement**Type**

Float

Default value

0.05

Unit

Angstrom

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small displacement along the imaginary vibrational mode. This keywords sets the size of the displacement for the furthest moving atom.

Keep**Type**

Multiple Choice

Default value

None

Options

[None, all]

Description

Keep all the output files of the geometry optimizations. If set to 'all', must be used in combination with `Output%KeepWorkDir`.

MaxConvergenceTime**Type**

Multiple Choice

Default value

Default

Options

[Default, High]

Description

The number of iterations for the geometry optimization, based on the number of atoms in the system. The default value is the general AMS value for geometry optimization. Often, for conformer generation, it needs to be set higher.

MaxOptimizations**Type**

Integer

Default value

1000

Description

Set a maximum to the number of geometries accepted for optimization at once, per AMSWorker (so should be multiplied by the number of cores used). If not set, the disc size requirements can become too large.

OptimizationMethod**Type**

Multiple Choice

Default value

Quasi-Newton

Options

[Auto, Quasi-Newton, FIRE, L-BFGS, ConjugateGradients, Dimer]

Description

Select the optimization algorithm employed for the geometry relaxation. Currently supported are:

the Hessian-based Quasi-Newton-type BFGS algorithm,

the fast inertial relaxation method (FIRE),

the limited-memory BFGS method,

and the conjugate gradients method. The default is Quasi-Newton, which gives the most reliable results for conformers. The Auto method leaves it to the AMS GeometryOptimization task to select a method.

UseAMSWorker**Type**

Bool

Default value

Yes

Description

Whether the set of optimizations should be run via the AMSWorkerPool or via regular AM-SJobs.

WriteGeometries**Type**

Block

Description

Determines if and where optimized geometries will be written to file during optimization. When the AMSWorker is used, then the write interval depends on MaxOptimizations. If enabled, must be used in combination with Output%KeepWorkDir.

Dirname**Type**

String

Default value

conf_tmpdir

Description

The name of the folder that should contain the optimized geometries.

Enabled**Type**

Bool

Default value

No

Description

Enables or disables the periodic writing of optimized geometries to file

10.4 Trajectory Analysis

`analysis` is a standalone program that performs analysis of molecular dynamics trajectories created with AMS. It can produce histograms and radial distribution functions. It is used under the hood in [AMSmovie](#) (**MD Properties** menu bar).

10.4.1 New in Trajectory Analysis-2025

- Ionic conductivity
- Mean square displacement (including ionic conductivity) can be computed for molecular centers of mass

10.4.2 New in Trajectory Analysis-2023

- Intra- and inter-molecular radial distribution functions
- Properties used to compute autocorrelation or mean square displacement can be stored to file
- Manual selection of coordinate unwrapping for autocorrelation or mean square displacement functions

10.4.3 Quickstart Guide

This example computes the oxygen-oxygen radial distribution function of a MD simulation using the analysis utility program:

```
#!/bin/sh

"$AMSBIN/analysis" << eor

Task RadialDistribution

TrajectoryInfo
  Trajectory
    KFFilename ams.results/ams.rkf
    Range 1 1000 2
  End
End

RadialDistribution
  NBins 1000
  AtomsFrom
    Element O
  End
  AtomsTo
    Element O
  End
End
eor
```

10.4.4 Trajectory selection

The analysis program reads one or more trajectory files (filename.rkf) from an AMS molecular dynamics (MD) or a Grand Canonical Monte Carlo (GCMC) simulation. The file information is supplied in the `TrajectoryInfo` input block. In this block, a separate `Trajectory` subblock needs to be supplied for each trajectory file. The `Trajectory` subblock contains a mandatory keyword `KFFilename`, and an optional keyword `Range`. The latter contains the initial frame to be read, the final frame to be read, and optionally the stepsize. By default all frames on the trajectory file are read.

```
TrajectoryInfo
  NBlocksToCompare integer
  Trajectory
    KFFilename string
    Range integer_list
    StepSize integer
  End
End
```

TrajectoryInfo

Type

Block

Description

All the info regarding the reading of the trajectory files.

NBlocksToCompare

Type

Integer

Default value

1

Description

Get an error estimate by comparing histograms for NBlocks time blocks of the trajectory.

Trajectory

Type

Block

Recurring

True

Description

All info regarding the reading of a single trajectory file.

KFFilename

Type

String

Default value

ams.rkf

Description

The name of the AMS trajectory file.

Range

Type

Integer List

Description

One or two values: start frame, and optionally end frame. By default the first and last frame are read.

StepSize**Type**

Integer

Default value

1

Description

The step size at which frames are read from the RKF (default 1, every frame is read).

10.4.5 Atom Selection

All tasks in the analysis program allow a subset of atoms to be selected, to be used for the analysis. They can be selected using element names (**Element**), region names (**Region**) or atom indices (**Atom**). In case **Region** is used, the user can overwrite the regions set in the original simulation in a new **System** block. The elements in the system block do have to match the elements in the original simulation.

Note: In the case of a **MolecularGun** simulation, the input regions will be overwritten with the regions from the trajectory file as soon as atoms are added to or removed from the system.

```
#!/bin/sh

"$AMSBIN/analysis" << eor

Task RadialDistribution

System
  Atoms
    O      -0.7113934019      1.6140994026      -0.7075521284 region=Oxygen
    H      -0.0425480297      2.3360319223      -0.8302756333
    H      -0.5643276324      1.0450643077      -1.4556911815
    O      1.1006174021      -7.1200763645      -3.6017659954 region=Oxygen
    H      1.4647560784      -6.1967314913      -3.5522086623
    H      1.8229257860      7.2170281524      -3.7443300356
  End
End

TrajectoryInfo
  Trajectory
    KFFilename ams.results/ams.rkf
    Range 1 1000 2
  End
End

RadialDistribution
  NBins 1000
  AtomsFrom
    Region Oxygen
  End
  AtomsTo
```

(continues on next page)

(continued from previous page)

```

      Region Oxygen
    End
End
eor

```

10.4.6 Convergence Check

The histogram and radialdistribution tasks provide an option to obtain information on the equilibration of the simulation. If the optional keyword `NBlocksToCompare` in the `TrajectoryInfo` block is set to a value N higher than 1, the trajectory is divided into N blocks, and the analysis results for each block are compared. The variation in the analysis result is provided as a standard deviation.

10.4.7 Radial Distribution Function (RDF)

The Analysis tool computes radial distribution functions $g(r)$ if the `Task` keyword is set to `RadialDistribution`.

```

Task [RadialDistribution | Histogram | AutoCorrelation | MeanSquareDisplacement | ↵
↵AverageBinPlot]

```

Task

Type

Multiple Choice

Options

[RadialDistribution, Histogram, AutoCorrelation, MeanSquareDisplacement, AverageBinPlot]

Description

The analysis task.

Further details on the radial distribution functions are then set in the `RadialDistribution` block. If more than one `RadialDistribution` block is present in the input, more than one radial distribution function will be computed. The result is printed to output as text, as well as stored in a binary file (plot.kf).

Description

A radial distribution function $g(r)$, or pair correlation function, is a density of distances between particles, relative to the average distance density. The x -axis variable represents a distance r , while the y -axis represents the relative density of that distance. For a complete homogeneous system of particles the $g(r)$ values for the distances between all particles equals 1 everywhere.

Two sets of atoms S_{from} and S_{to} , of length n_{from} and n_{to} respectively, are specified with the keywords `AtomsFrom` and `AtomsTo` in the `RadialDistribution` block. As a result the program computes $n_{\text{from}} * n_{\text{to}}$ distances r_{ij}^s between atom i in S_{from} and atom j in S_{to} for each trajectory frame s out of a total of n_{frames} frames.

A normalized histogram is then computed from these distances, resulting in a function $N(r)$.

$$N(r) = \frac{1}{n_{\text{frames}}} \sum_{s=1}^{n_{\text{frames}}} \sum_{i=1}^{n_{\text{from}}} \sum_{j=1}^{n_{\text{to}}} \delta(r_{ij}^s - r).$$

This histogram is converted to a density, by dividing all values $N(r)$ with the volume $V(r) = 4\pi r^2 dr$ of a sphere-slice at radius r with thickness dr .

The density is further converted to a relative density by dividing with the total density of the system $\rho_{\text{tot}} = \frac{n_{\text{from}} * n_{\text{to}}}{V_{\text{tot}}}$, yielding the final radial distribution function $g(r)$.

$$g(r) = \frac{N(r)}{V(r) * \rho_{\text{tot}}}$$

Atom Selecion

The variables n_{from} and n_{to} determining the atom pairs that contribute to the radial distribution can be selected in the `AtomsFrom` and `AtomsTo` blocks. Both can be selected using element names (`Element`), region names (`Region`) or atom indices (`Atom`).

```
RadialDistribution
  AtomsFrom
    Atom integer
    Element string
    Region string
  End
End
```

RadialDistribution

Type

Block

Recurring

True

Description

All input related to radial distribution functions.

AtomsFrom

Type

Block

Description

Atom numbers or elements for the first set of atoms in the radial distribution.

Atom

Type

Integer

Default value

0

Recurring

True

Description

Atom number.

Element

Type

String

Recurring

True

Description

Element Symbol Atom.

Region

Type

String

Recurring

True

Description

Region name.

RadialDistribution**Type**

Block

Recurring

True

Description

All input related to radial distribution functions.

AtomsTo**Type**

Block

Description

Atom numbers or elements for the second set of atoms in the radial distribution.

Atom**Type**

Integer

Default value

0

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

Options

Non-periodic systems The above equation assumes that the volume V_{tot} of the system is a well-defined quantity. This assumption is correct for systems with 3D periodicity, where the V_{tot} is defined as the volume of the periodic cell. In such a system the value of r can be no larger than r_{max} , the radius of the largest sphere that can be placed inside the periodic cell.

If a system is non-periodic in one or more direction, then the program still computes a $g(r)$, only if the radius r_{max} is supplied by the user with the `Range` keyword in the `RadialDistribution` block. The radius is the second value supplied.

```
RadialDistribution
  Range float_list
End
```

RadialDistribution

Type

Block

Recurring

True

Description

All input related to radial distribution functions.

Range

Type

Float List

Description

Either one, two, or three real values. If one it is the stepsize. If two, it is the minimum value and the maximum value. If three, it is the minimum value, the maximum value, and the stepsize. The stepsize overrides NBins.

In this case the volume V_{tot} is assumed to be the volume of a sphere with radius r_{max} .

NPT simulations The above equation further assumes that the volume V_{tot} is constant throughout the simulation. The $g(r)$ of the trajectory from an NPT simulation can still be computed, and in this case V_{tot} is the average value of the volume of the periodic cell.

Simulations with varying numbers of atoms The above equation also assumes that n_{from} and n_{to} remain constant throughout the simulation. However, in a Molecular Gun simulation particles can be added to the system, and in a GCMC simulation particles can be both added and removed from the system. Nonetheless, the program still computes a $g(r)$ in these situations.

If the `AtomsFrom` and `AtomsTo` blocks contain element names (supplied with the recurring `Element` keyword), then every time atoms are added to or removed from the system, the sets of atoms \mathbb{S}_{from} and \mathbb{S}_{to} are re-evaluated.

If the `AtomsFrom` and `AtomsTo` blocks contain atom numbers (supplied with the recurring `Atom` keyword), these numbers are updated in the sets \mathbb{S}_{from} and \mathbb{S}_{to} every time atoms are added to or removed from the system. If one of the atoms from the set disappears, the number of distances contributing to the $g(r)$ decreases.

Note: Currently, the values of n_{from} and n_{to} in the normalization factor are taken from the last frame of the simulation.

Warning: If multiple trajectories are supplied, and the number of atoms changes between the end of one trajectory and the beginning of another, this may result in an error in the atom numbers used by the program internally.

Inter- or intra-molecular atom-pairs By default, all $n_{\text{from}} * n_{\text{to}}$ distances are included. Sometimes it can be convenient to view exclusively the distances between atoms within a molecule, or those between different molecules. This can be controlled with the keyword `DistanceTypeSelection` in the `RadialDistribution` block.

```
RadialDistribution
  DistanceTypeSelection [All | InterMolecular | IntraMolecular]
End
```

RadialDistribution**Type**

Block

Recurring

True

Description

All input related to radial distribution functions.

DistanceTypeSelection**Type**

Multiple Choice

Default value

All

Options

[All, InterMolecular, IntraMolecular]

Description

Select only a certain type of interatomic distances.

10.4.8 Histogram

The Analysis program computes histograms if the Task keyword is set to Histogram.

```
Task [RadialDistribution | Histogram | AutoCorrelation | MeanSquareDisplacement | ↵
↵AverageBinPlot]
```

Task**Type**

Multiple Choice

Options

[RadialDistribution, Histogram, AutoCorrelation, MeanSquareDisplacement, AverageBinPlot]

Description

The analysis task.

Further details on the histogram need to be specified in the Histogram block. If more than one Histogram block is present in the input, more than one histogram will be computed. The result is printed to output as text, as well as stored in a binary file (plot.kf). By default the histogram contains the number of occurrences of a certain value, but the normalized occurrence is provided if the keyword Normalized in the Histogram block is specified.

```
Histogram
  Normalized Yes/No
End
```

Histogram**Normalized**

Type

Bool

Default value

No

Description

Give the normalized histogram.

Histograms can be computed for every quantity stored on the molecular dynamics trajectory file (ams.rkf) in the section History or MDHistory. Examples are scalar quantities like `PotentialEnergy`, `KineticEnergy`, `TotalEnergy`, `Temperature`. In the histogram block, this quantity is selected with the keyword `Variable` in the `Axis` subblock. If more than one `Axis` subblock is present, the dimensionality of the histogram is increased: Three `Axis` subblocks result in a 3D histogram.

It is also possible to select quantities that have scalar or vector values for each atom, such as `Coords` or `Velocities`. If this is combined with multiple axes, then the number of values along all axes must be the same. It is, for example, not possible to plot the atom velocities along one axis, and the temperature per frame along the other. A subset of atoms for which the atom-wise quantity should be selected can be provided in the subblock `Atoms`. The block can contain element names (recurring keyword `Element`), region names (recurring keyword `Region`), or individual atom numbers (recurring keyword `Atom`).

```
Atoms
  Atom integer
  Element string
  Region string
End
```

Atoms**Type**

Block

Description

Relevant if variable has a value per atom (e.g. `Coords`, `Velocities`). This block specifies indices or elements for the set of atoms for which the variable is to be read. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom index.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name

A subset of vector elements can be provided with the subblock `VecElements`. By default all vector elements found on the RKF file will be used.

```
VecElements
  Index integer
End
```

VecElements**Type**

Block

Description

A set of indices referring to a subset of a vector. If the variable to be plotted has non-scalar values per step, then this block allows the selection of a subset of vector elements (e.g. 1 and 2 for the x and y values). Can be used in combination with the `Atoms` block.

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

For each histogram axis, the number of bins can be selected with the `NBins` keyword in the `Axis` block, in which case the range of values along each axis is automatically determined. The default `NBins` value is 100.

Alternatively, a range and a stepsize can be selected with the keyword `Range` in the `Axis` subblock. The keyword `Range` can contain one, two, or three values: 1: Only a stepsize. 2: A smallest value and a largest value. 3: A smallest value, a largest value, and the stepsize.

```
Histogram
  Axes
    Axis
      Atoms
        Atom integer
        Element string
        Region string
      End
      NBins integer
      Range float_list
      Variable string
      VecElements
        Index integer
      End
    End
  End
End
```

Histogram

Type

Block

Recurring

True

Description

All input related to histograms.

Axes**Type**

Block

Description

Specifications for the histogram axes.

Axis**Type**

Block

Recurring

True

Description

Specifications for a single histogram axis.

Atoms**Type**

Block

Description

Relevant if variable has a value per atom (e.g. Coords, Velocities). This block specifies indices or elements for the set of atoms for which the variable is to be read. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom index.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name

NBins**Type**

Integer

Default value

100

Description

The number of bins along the histogram axis.

Range**Type**

Float List

Description

Either one, two, or three real values. If one it is the stepsize. If two, it is the minimum value and the maximum value. If three, it is the minimum value, the maximum value, and the stepsize. The stepsize overrides NBins.

Variable**Type**

String

Description

The quantity along the histogram axis.

VecElements**Type**

Block

Description

A set of indices referring to a subset of a vector. If the variable to be plotted has non-scalar values per step, then this block allows the selection of a subset of vector elements (e.g. 1 and 2 for the x and y values). Can be used in combination with the Atoms block.

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

10.4.9 Autocorrelation Functions

The Analysis program computes autocorrelation functions (ACF) if the Task keyword is set to AutoCorrelation.

Task [RadialDistribution | Histogram | AutoCorrelation | MeanSquareDisplacement | ↵
↵AverageBinPlot]

Task

Type

Multiple Choice

Options

[RadialDistribution, Histogram, AutoCorrelation, MeanSquareDisplacement, AverageBinPlot]

Description

The analysis task.

Further details need to be specified in the AutoCorrelation block. If more than one AutoCorrelation block is present in the input, more than one ACF will be computed. The result is printed to output as text, as well as stored in a binary file (plot.kf).

Description

An autocorrelation function $C(t)$ describes the average correlation (overlap) of a (vector) property **A** with itself as a function of time.

$$C(t) = \langle \mathbf{A}(0) \cdot \mathbf{A}(t) \rangle$$

The average runs over all time-intervals $(t_0, t_0 + t)$, $(t_1, t_1 + t)$, ..., $(t_N, t_N + t)$, with $t_N = t_n - t_m$. Here n is the total number of frames read from the trajectory (as determined in the block TrajectoryInfo), and m is the number of discrete t values for which $C(t)$ is computed. The value m can be set with the keywords MaxCorrelationTime or MaxFrame. This generally defaults to half the total number of frames read. Only when the pressure tensor is used (e.g. in the computation of the viscosity), the default value is smaller, at 10% of the total number of frames read. As stated above, the default average runs over the same number of time intervals for each value of t . If UseAllValues is selected, the average runs over all available time intervals for each value of t , which is large for small t , and smallest (N) for t_m .

The normalized autocorrelation function $c(t)$ describes the decorrelation of the property with time, and always starts at 1.0 at $t = 0$.

$$c(t) = \frac{\langle \mathbf{A}(0) \cdot \mathbf{A}(t) \rangle}{\langle \mathbf{A}(0) \cdot \mathbf{A}(0) \rangle}$$

In most cases short timescale fluctuations are important, so frequent storage of the desired property is required (when preparing the molecular dynamics simulation, set the SamplingFreq keyword in the Trajectory block of the MolecularDynamics settings low, preferably to 1).

Setting SamplingFreq to 1 results in the storage of a lot of superfluous information, which takes both time and disc space. For a few selected properties this can be circumvented with the MolecularDynamics%BinLog block. Properties requested in this block will be stored every step, even if SamplingFreq is set to a higher value. The trajectory analysis tool can take advantage of this for the computation of some autocorrelation functions, through the Property keyword.

A power spectrum is automatically computed by Fourier transform of the autocorrelation function, and provides information on the frequencies of the signal. When the selected property is the dipole moment time derivative, the power spectrum matches the IR spectrum.

Properties

Autocorrelation functions can be computed for different simulation properties:

- 1) Dipole moments from coordinates and atomic charges
- 2) Dipole moment derivatives from velocities and atomic charges
- 3) Velocities
- 4) The pressure tensor
- 5) Diffusion coefficient (equivalent to selecting *Velocities*).
- 6) Viscosity (equivalent to selecting *PressureTensor*, if only the off-diagonal elements are selected using *VecElements*)
- 7) *DipoleMomentFromBinLog* (approximately equivalent to *DipoleMomentFromCharges* if available)
- 8) *ViscosityFromBinlog* (equivalent to *Viscosity* if available)

```
AutoCorrelation
  Property [Velocities | DipoleMomentFromCharges | DiffusionCoefficient | ↵
↵DipoleDerivativeFromCharges |
      PressureTensor | Viscosity | DipoleMomentFromBinLog | ↵
↵ViscosityFromBinLog]
End
```

AutoCorrelation

Type

Block

Recurring

True

Description

All input related to auto correlation functions.

Property

Type

Multiple Choice

Default value

DipoleDerivativeFromCharges

Options

[Velocities, DipoleMomentFromCharges, DiffusionCoefficient, DipoleDerivativeFromCharges, PressureTensor, Viscosity, DipoleMomentFromBinLog, ViscosityFromBinLog]

Description

Compute the ACF either from velocities (from rkf), the dipole moment (from coordinates and atomic charges in rkf), the dipole moment derivative (from velocities and atomic charges in rkf), from the pressure tensor (from rkf), or from values specified in input. Selecting *DiffusionCoefficient* is equivalent to selecting *Velocities*. The default, *DipoleDerivativeFromCharges*, results in the computation of an IR spectrum. *DipoleMomentFromBinLog* and *ViscosityFromBinLog* allow the relevant properties (dipole moment and pressure tensor respectively) to be read from the BinLog section of the trajectory file. In the BinLog section requested properties are stored every step (even if *SamplingFreq* was set to a higher number than 1) but only if this was specifically requested at the start of the MD simulation.

When selecting options 7) or 8) the program attempts to read the dipole moment or the pressure tensor respectively from the BinLog section of the ams.rkf file. This will fail if the data is not stored there. The data will only be stored there if the user selected `MolecularDynamics%BinLog%DipoleMoment` and `MolecularDynamics%BinLog%PressureTensor` when setting up the MD simulation.

Some of the properties for which an autocorrelation function can be computed are simply read as is from the trajectory RKF file, but others are quite complex. For example, the dipole moment is a property obtained by reading the coordinates and the atomic charges, multiplying each atomic position vector with the corresponding charge, and then summing over all atoms. With the keyword `WritePropertyToKF` in the `AutoCorrelation` block, the user can choose to store not only the autocorrelation function, but also the property used to produce it.

```
AutoCorrelation
  WritePropertyToKF Yes/No
End
```

AutoCorrelation

Type

Block

Recurring

True

Description

All input related to auto correlation functions.

WritePropertyToKF

Type

Bool

Default value

No

Description

Write the selected property to the KF files for every requested frame

If a property involves atomic coordinates, then this generally means atomic coordinates wrapped into the periodic box at every time step. As a result, coordinates can make seemingly big jumps from one side of the box to another between two consecutive frames. The code is able to unwrap the coordinates, so that all atoms follow a continuous trajectory. For the `DipoleMoment` property, the wrapped coordinates will be used by default, but an experienced user can determine whether the wrapped or unwrapped coordinates are used with the keyword `UnwrapCoordinates` in the `AutoCorrelation` block.

```
AutoCorrelation
  UnwrapCoordinates [Auto | Yes | No]
End
```

AutoCorrelation

Type

Block

Recurring

True

Description

All input related to auto correlation functions.

UnwrapCoordinates

Type

Multiple Choice

Default value

Auto

Options

[Auto, Yes, No]

Description

If the coordinates are involved in the requested property, those coordinates are wrapped into the box at each time step. If set to true, this keyword unwraps those coordinates so that the trajectory is continuous. If not provided the code uses automatic defaults.

Options

With the keywords `MaxCorrelationTime` (in fs) or `MaxFrame` the number of values m in the autocorrelation function ($t = [0, t_1, t_2, \dots, t_m]$) can be set. When both are set, `MaxCorrelationTime` takes precedent. In most cases the default value is half of the total number of frames n read from the trajectory. Only when the selected Property is 'PressureTensor', 'Viscosity' or 'ViscosityFromBinLog', the default is 10% of the trajectory.

A subset of atoms for which the property **A** should be selected/computed can be provided in the block `Atoms`. The block can contain element names (recurring keyword `Element`), region names (recurring keyword `Region`), or individual atom numbers (recurring keyword `Atom`).

```
AutoCorrelation
  Atoms
    Atom integer
    Element string
    Region string
  End
End
```

AutoCorrelation

Type

Block

Recurring

True

Description

All input related to auto correlation functions.

Atoms**Type**

Block

Description

Relevant if Property is set to Velocities, DipoleMomentFromCharges, DipoleDerivativeFromCharges, or DiffusionCoefficient. Atom numbers or elements for the set of atoms for which the property is read/computed. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

A subset of vector elements can be provided with the subblock `VecElements`. By default all vector elements found on the RKF file will be used.

```
AutoCorrelation
  VecElements
    Index integer
  End
End
```

AutoCorrelation**Type**

Block

Recurring

True

Description

All input related to auto correlation functions.

VecElements**Type**

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property Velocities, the Atoms block allows the selection of a subset of atoms, while the VecElements block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

Keywords

The `AutoCorrelation` block holds the following keywords.

```
AutoCorrelation
  Atoms
    Atom integer
    Element string
    Region string
  End
  DataReading [Auto | AtOnce | BlockWise]
  MaxCorrelationTime float
  MaxFrame integer
  NPointsHighestFreq integer
  PerElement Yes/No
  Property [Velocities | DipoleMomentFromCharges | DiffusionCoefficient | ↵
↵DipoleDerivativeFromCharges |
    PressureTensor | Viscosity | DipoleMomentFromBinLog | ↵
↵ViscosityFromBinLog]
  UnwrapCoordinates [Auto | Yes | No]
  UseAllValues Yes/No
  UseTimeDerivative
    Enabled Yes/No
  End
  VecElements
    Index integer
  End
  WritePropertyToKF Yes/No
End
```

AutoCorrelation**Atoms****Type**

Block

Description

Relevant if `Property` is set to `Velocities`, `DipoleMomentFromCharges`, `DipoleDerivativeFromCharges`, or `DiffusionCoefficient`. Atom numbers or elements for the set of atoms for which the property is read/computed. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom number.

Element

Type

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

DataReading**Type**

Multiple Choice

Default value

Auto

Options

[Auto, AtOnce, BlockWise]

Description

The KF data can be read in and handled once, or blockwise. The former is memory intensive, but mostly faster. If Auto is selected, the data is read at once if it is less than 1 GB, and blockwise if it is more.

MaxCorrelationTime**Type**

Float

Description

The maximum correlation time in fs. The default is half the simulation time, except when the PressureTensor is read from the ams.rkf, in which case it is 10 percent of the simulation time. The PressureTensor is read when the Properties PressureTensor, Viscosity, or ViscosityFromBinLog are selected.

MaxFrame**Type**

Integer

Description

The maximum number of frames for which the autocorrelation function will be computed. The default is half of the number of provided frames. Determines the same settings as MaxCorrelationTime. If both are set, MaxCorrelationTime will take precedence.

NPointsHighestFreq**Type**

Integer

Default value

4

Description

The number of points (timesteps) used for the highest frequency displayed in spectrum. This determines up to which frequency the spectrum is displayed. If the spacing between time-steps used for the ACF is 1 fs, then by default the maximum frequency displayed is 0.25 fs⁻¹ (or 8339 cm⁻¹). This corresponds to a (default) value of NPointsHighestFreq of 4. A higher number selected here, will result in a lower maximum frequency returned by the program. The lowest possible value (spectrum up to highest possible frequency) is 2.

PerElement**Type**

Bool

Default value

No

Description

Compute ACF for all elements in the system. Any other settings in the block will be used.

Property**Type**

Multiple Choice

Default value

DipoleDerivativeFromCharges

Options

[Velocities, DipoleMomentFromCharges, DiffusionCoefficient, DipoleDerivativeFromCharges, PressureTensor, Viscosity, DipoleMomentFromBinLog, ViscosityFromBinLog]

Description

Compute the ACF either from velocities (from rkf), the dipole moment (from coordinates and atomic charges in rkf), the dipole moment derivative (from velocities and atomic charges in rkf), from the pressure tensor (from rkf), or from values specified in input. Selecting DiffusionCoefficient is equivalent to selecting Velocities. The default, DipoleDerivativeFromCharges, results in the computation of an IR spectrum. DipoleMomentFromBinLog and ViscosityFromBinLog allow the relevant properties (dipole moment and pressure tensor respectively) to be read from the BinLog section of the trajectory file. In the BinLog section requested properties are stored every step (even if SamplingFreq was set to a higher number than 1) but only if this was specifically requested at the start of the MD simulation.

UnwrapCoordinates**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Yes, No]

Description

If the coordinates are involved in the requested property, those coordinates are wrapped into the box at each time step. If set to true, this keyword unwraps those coordinates so that the trajectory is continuous. If not provided the code uses automatic defaults.

UseAllValues**Type**

Bool

Default value

No

Description

By default the same number of values are used for each t-step in the ACF. This has the advantage that all values in the ACF are equally reliable, but it does mean that for the smaller timesteps much of the data is not used. To switch this off and use all data, UseAllValues can be set to true

UseTimeDerivative**Type**

Block

Description

Possibly use the time derivative of the selected property (e.g. velocity or dipole moments).

Enabled**Type**

Bool

Default value

No

Description

Enable the use of the time derivative of the property.

VecElements**Type**

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property Velocities, the Atoms block allows the selection of a subset of atoms, while the VecElements block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

WritePropertyToKF**Type**

Bool

Default value


No

Description

Write the selected property to the KF files for every requested frame

10.4.10 Mean Square Displacement

The Analysis program computes mean square displacements (MSD) if the Task keyword is set to MeanSquareDisplacement.

Task [RadialDistribution | Histogram | AutoCorrelation | MeanSquareDisplacement |  AverageBinPlot]

Task

Type

Multiple Choice

Options

[RadialDistribution, Histogram, AutoCorrelation, MeanSquareDisplacement, AverageBinPlot]

Description

The analysis task.

Further details need to be specified in the MeanSquareDisplacement block. If more than one MeanSquareDisplacement block is present in the input, more than one MSD will be computed. The result is printed to output as text, as well as stored in a binary file (plot.kf).

Description

The mean square displacement $MSD(t)$ describes the average change of a (vector) property **A** over time. This property is usually an atom coordinate vector, but the implementation is entirely general.

$$MSD(t) = \langle \frac{1}{k} \sum_i^k ||\mathbf{A}_i(t') - \mathbf{A}_i(t)||^2 \rangle_{t'}$$

Here \mathbf{A}_i is usually the coordinate vector for atom i out of a set of k atoms. The average runs over all values of t' , or the time-intervals $(t_0, t_0 + t)$, $(t_1, t_1 + t)$, ..., $(t_N, t_N + t)$, with $t_N = t_n - t_m$. Here n is the total number of frames read from the trajectory, and m is the number of discrete t values for which $MSD(t)$ is computed. The value m can be set with the keywords MaxCorrelationTime or MaxFrame. It defaults to half the total number of frames read from the trajectory. As stated above, the default average runs over the same number of time intervals for each value of t . If UseAllValues is selected, the average runs over all available time intervals for each value of t , which is large for small t , and smallest (N) for t_m .

Slope

The most common use of the mean square displacement is for the computation of the diffusion coefficient, in which case the selected property is the set of atom coordinates (Coords). The diffusion coefficient is proportionate to the slope of this mean square displacement function, and therefore this slope β is automatically computed using linear regression.

$$\beta = \frac{\sum_i (t_i - \bar{t})(MSD_i - \overline{MSD})}{\sum_i (t_i - \bar{t})^2}$$

Here i runs over all contributions in the linear regime of the MSD graph.

The function $MSD(t)$ becomes linear only after an initial time interval, and the user can set this initial time with the keyword StartTimeSlope. If not provided, this start time is automatically determined. This is done by computing the slope from t_0 to t_m for increasing t_0 (from 0 to $\frac{1}{2}t_m$). At small t_0 values we expect the linear fit to keep improving, so the correlation coefficient r_{xy} should increase. At the switching point, where the MSD becomes linear, r_{xy} will no longer improve, so we select the t_0 that results in a smaller r_{xy} than the previous one.

$$r_{xy} = \frac{\sum_i (t_i - \bar{t})(MSD_i - \overline{MSD})}{\sqrt{\sum_i (t_i - \bar{t})^2 \sum_i (MSD_i - \overline{MSD})^2}}$$

To allow the user to determine if the linear regime has been sufficiently sampled, the slope of $MSD(t)$ as a function of an end time t_e is computed as well. If the slope does not converge to a stable value, the user should select a larger value of t_m or continue the molecular dynamics simulation for a longer time.

The convergence plot of the slope takes the MSD data, and computes slope values over intervals (t_0, t_e) , where t_e increases from t_0 to t_m . If the slope is to represent the diffusion coefficient (see [DiffusionFromMSD](#) (page 451)), then the slope $\beta(t_e)$ of the MSD up to t_e is divided by the number of dimensions (usually 3), and multiplied by a factor $\frac{1}{2}$.

$$D(t_e) = \frac{1}{2d}\beta(t_e)$$

Properties

The only property **A** of real relevance for the computation of the mean square displacement is a set of coordinates. Nonetheless, the `Property` keyword accepts four different options.

- 1) Coordinates
- 2) Diffusion coefficient
- 3) Ionic conductivity

Option 2) is equivalent to option 1). Option 3) uses charge weighted coordinates as the property **A**.

```
MeanSquareDisplacement
  Property [Coords | DiffusionCoefficient | Conductivity]
End
```

MeanSquareDisplacement

Type

Block

Recurring

True

Description

All input related to mean square displacement functions.

Property

Type

Multiple Choice

Default value

Coords

Options

[Coords, DiffusionCoefficient, Conductivity]

Description

Compute the MSD from the property selected here (from rkf). Selecting DiffusionCoefficient is equivalent to selecting the property Coords.

When read from file, the atomic coordinates will be wrapped inside the periodic box at every time step. As a result, coordinates can make seemingly big jumps from one side of the box to another between two consecutive frames. By default, the coordinates are unwrapped before the computation of the mean square displacement, so that all atoms follow a continuous trajectory. For the experienced users, the option exist to manually determine whether the wrapped or unwrapped coordinates are used with the keyword `UnwrapCoordinates` in the `MeanSquareDisplacement` block.

```
MeanSquareDisplacement
  UnwrapCoordinates [Auto | Yes | No]
End
```

MeanSquareDisplacement**Type**

Block

Recurring

True

Description

All input related to mean square displacement functions.

UnwrapCoordinates**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Yes, No]

Description

If the coordinates are involved in the requested property, those coordinates are wrapped into the box at each time step. If set to true, this keyword unwraps those coordinates so that the trajectory is continuous. If not provided the code uses automatic defaults.

Since the coordinates used to compute the mean square displacement often differ from the coordinates as read from the trajectory (they will be unwrapped, so that the trajectory becomes continuous). With the keyword `WritePropertyToKF` in the `MeanSquareDisplacement` block, the user can choose to store not only the mean square displacement results, but also the property used to produce it.

```
MeanSquareDisplacement
  WritePropertyToKF Yes/No
End
```

MeanSquareDisplacement**Type**

Block

Recurring

True

Description

All input related to mean square displacement functions.

WritePropertyToKF**Type**

Bool

Default value

No

Description

Write the selected property to the KF files for every requested frame

Options

A subset of atoms for which the property **A** should be selected/computed can be provided in the block `Atoms`. The block can contain element names (recurring keyword `Element`), region names (recurring keyword `Region`), or individual atom numbers (recurring keyword `Atom`).

```
MeanSquareDisplacement
  Atoms
    Atom integer
    Element string
    Region string
  End
End
```

MeanSquareDisplacement

Type

Block

Recurring

True

Description

All input related to mean square displacement functions.

Atoms

Type

Block

Description

Relevant if Property is set to any quantity that is available per atom (Coords, DiffusionCoefficient). Atom numbers or elements for the set of atoms for which the property is read/computed are provided here. By default all atoms are used.

Atom

Type

Integer

Recurring

True

Description

Atom number.

Element

Type

String

Recurring

True

Description

Element Symbol Atom.

Region

Type

String

Recurring

True

Description

Region name.

A subset of vector elements can be provided with the subblock `VecElements`. By default all vector elements found on the RKF file will be used.

```
MeanSquareDisplacement
  VecElements
    Index integer
  End
End
```

MeanSquareDisplacement**Type**

Block

Recurring

True

Description

All input related to mean square displacement functions.

VecElements**Type**

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property `Coords`, the `Atoms` block allows the selection of a subset of atoms, while the `VecElements` block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

The center of mass of each selected molecule can be used in the MSD computation. In that case, the selected atoms will be divided into groups belonging to different molecules, and the center of mass of each group is then computed. This will only work if the molecules remain stable throughout the simulation, which can only really be guaranteed if the `ForceField` or the `GFNFF` engines are used. If the bonding does change, the computation will abort by default. If, however, `UseMolecularCentersOfMass%CheckForBondChanges` is set to `No`, then any bond changes will be ignored, and the computation continues as if the molecules did not change.

Atomic masses used for the molecular centers of mass can be altered by providing a system block in the input. The elements in the input system block need to match the elements in the molecular dynamics simulation. If no system block is present in the input, the system block from the molecular dynamics simulation will be used.

```
#!/bin/sh
```

(continues on next page)

(continued from previous page)

```

"$AMSBIN/analysis" << eor

Task MeanSquareDisplacement

System
  Atoms
    O      -0.7113934019      1.6140994026      -0.7075521284 mass=15.999
    H      -0.0425480297      2.3360319223      -0.8302756333 mass=2.014
    H      -0.5643276324      1.0450643077      -1.4556911815 mass=2.014
    O      1.1006174021      -7.1200763645      -3.6017659954 mass=15.999
    H      1.4647560784      -6.1967314913      -3.5522086623 mass=2.014
    H      1.8229257860      7.2170281524      -3.7443300356 mass=2.014
  End
End

TrajectoryInfo
  Trajectory
    KFFilename ams.results/ams.rkf
  End
End

MeanSquareDisplacement
  Property Coords
  UseMolecularCentersOfMass
    Enabled Yes
  End
End
eor

```

```

MeanSquareDisplacement
  UseMolecularCentersOfMass
    CheckForBondChanges Yes/No
    Enabled Yes/No
  End
End

```

MeanSquareDisplacement**Type**

Block

Recurring

True

Description

All input related to mean square displacement functions.

UseMolecularCentersOfMass**Type**

Block

Description

Specifications on using the center of mass coordinates per molecule instead of coordinates per atom. Only relevant for properties that rely on coordinates: Coords, DiffusionCoefficient, and Conductivity.

CheckForBondChanges

Type

Bool

Default value

Yes

Description

Check for each frame if the bonds still correspond to the input bonds. If they changed, an error is thrown. If this is set to false, it is assumed that the molecules stay in tact.

Enabled**Type**

Bool

Default value

No

Description

Use the center of mass coordinates per molecule. Only relevant for properties that rely on coordinates: Coords, DiffusionCoefficient, and Conductivity. The bonds from the MD input system are used.

Keyword

The MeanSquareDisplacement block holds the following keywords.

```
MeanSquareDisplacement
  Atoms
    Atom integer
    Element string
    Region string
  End
  DataReading [Auto | AtOnce | BlockWise]
  MaxCorrelationTime float
  MaxFrame integer
  PerElement Yes/No
  Property [Coords | DiffusionCoefficient | Conductivity]
  StartTimeSlope float
  UnwrapCoordinates [Auto | Yes | No]
  UseAllValues Yes/No
  UseMolecularCentersOfMass
    CheckForBondChanges Yes/No
    Enabled Yes/No
  End
  VecElements
    Index integer
  End
  WritePropertyToKF Yes/No
End
```

MeanSquareDisplacement**Atoms****Type**

Block

Description

Relevant if Property is set to any quantity that is available per atom (Coords, DiffusionCoefficient). Atom numbers or elements for the set of atoms for which the property is read/computed are provided here. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

DataReading**Type**

Multiple Choice

Default value

Auto

Options

[Auto, AtOnce, BlockWise]

Description

The KF data can be read in and handled once, or blockwise. The former is memory intensive, but mostly faster. If Auto is selected, the data is read at once if it is less than 1 GB, and blockwise if it is more.

MaxCorrelationTime**Type**

Float

Description

The maximum correlation time in fs. The default is half the simulation time.

MaxFrame

Type

Integer

Description

The maximum number of frames for which the mean square displacement function will be computed. The default is half of the number of provided frames. Determines the same settings as MaxCorrelationTime. If both are set, MaxCorrelationTime will take precedence.

PerElement**Type**

Bool

Default value

No

Description

Compute MSD for all elements in the system. Any other settings in this block will be used.

Property**Type**

Multiple Choice

Default value

Coords

Options

[Coords, DiffusionCoefficient, Conductivity]

Description

Compute the MSD from the property selected here (from rkf). Selecting DiffusionCoefficient is equivalent to selecting the property Coords.

StartTimeSlope**Type**

Float

Default value

0.0

Description

The MSD has a nonlinear regime at short timescales, and a linear regime at long timescales. To determine the slope, the starting point for the linear regime has to be determined. This keyword sets the starting time in fs. If set to zero, the starttime will be automatically determined.

UnwrapCoordinates**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Yes, No]

Description

If the coordinates are involved in the requested property, those coordinates are wrapped into the box at each time step. If set to true, this keyword unwraps those coordinates so that the trajectory is continuous. If not provided the code uses automatic defaults.

UseAllValues

Type

Bool

Default value

No

Description

By default the same number of values are used for each t-step in the MSD. This has the advantage that all values in the MSD are equally reliable, but it does mean that for the smaller timesteps much of the data is not used. To switch this off and use all data, UseAllValues can be set to true

UseMolecularCentersOfMass**Type**

Block

Description

Specifications on using the center of mass coordinates per molecule instead of coordinates per atom. Only relevant for properties that rely on coordinates: Coords, DiffusionCoefficient, and Conductivity.

CheckForBondChanges**Type**

Bool

Default value

Yes

Description

Check for each frame if the bonds still correspond to the input bonds. If they changed, an error is thrown. If this is set to false, it is assumed that the molecules stay in tact.

Enabled**Type**

Bool

Default value

No

Description

Use the center of mass coordinates per molecule. Only relevant for properties that rely on coordinates: Coords, DiffusionCoefficient, and Conductivity. The bonds from the MD input system are used.

VecElements**Type**

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property Coords, the Atoms block allows the selection of a subset of atoms, while the VecElements block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

WritePropertyToKF**Type**

Bool

Default value

No

Description

Write the selected property to the KF files for every requested frame

10.4.11 AverageBinPlot

The Analysis program can plot two arbitrary properties, present on the RKF file, against each other averaged over each bin if the Task keyword is set to AverageBinPlot.

```
Task [RadialDistribution | Histogram | AutoCorrelation | MeanSquareDisplacement | ↵
↵AverageBinPlot]
```

Task**Type**

Multiple Choice

Options

[RadialDistribution, Histogram, AutoCorrelation, MeanSquareDisplacement, AverageBinPlot]

Description

The analysis task.

Further details need to be specified in the AverageBinPlot block. If more than one AverageBinPlot block is present in the input, more than one AverageBinPlot will be computed. The result is printed to output as text, as well as stored in a binary file (analysis.kf).

```
AverageBinPlot
  Atoms
    Atom integer
    Element string
    Region string
  End
  Nbins integer
  Property
    Axis float_list
    Name [FrictionCoefficient | Viscosity | Velocities | EngineGradients]
  End
  XProperty
    Name [Time | Coords]
    VecElements
      Index integer
    End
  End
End
```

AverageBinPlot

Atoms**Type**

Block

Description

Relevant if Properties are atom dependent. Atom numbers or elements for the set of atoms for which the property is read/computed. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

Nbins**Type**

Integer

Default value

10

Description

Number of bins that are plotted

Property**Type**

Block

Description

Property to be plotted along the Y-axis

Axis**Type**

Float List

Description

If defined the dot_product along this axis will be taken. Otherwise, the length of the property vector will be used.

Name**Type**

Multiple Choice

Options

[FrictionCoefficient, Viscosity, Velocities, EngineGradients]

Description

Name of the property

XProperty**Type**

Block

Description

Property to be plotted along the Y-axis

Name**Type**

Multiple Choice

Default value

Time

Options

[Time, Coords]

Description

Timestep used for the plotting

VecElements**Type**

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property Velocities, the Atoms block allows the selection of a subset of atoms, while the VecElements block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index**Type**

Integer

Default value

3

Description

Element of the x_property, in case it is a vector (For Coords: 1 for X, 2 for Y, 3 for Z).

10.4.12 Viscosity

The viscosity can be computed from an equilibrated Molecular Dynamics run as the integral over the autocorrelation function of the pressure tensor off-diagonal elements.

$$\eta = \frac{V}{k_B T} \int_{t=0}^{t=t_{max}} \langle P_{ij}(0) \cdot P_{ij}(t) \rangle_{i \neq j} dt$$

The viscosity is computed if the task `AutoCorrelation` is selected, and if in the `AutoCorrelation` block *Viscosity* is selected as the `Property`.

```
$AMSBIN/analysis <<eor
  Task AutoCorrelation
  AutoCorrelation
    Property Viscosity
  End
eor
```

AutoCorrelation

Type

Block

Recurring

True

Description

All input related to auto correlation functions.

Property

Type

Multiple Choice

Default value

DipoleDerivativeFromCharges

Options

[Velocities, DipoleMomentFromCharges, DiffusionCoefficient, DipoleDerivativeFromCharges, PressureTensor, Viscosity, DipoleMomentFromBinLog, ViscosityFromBinLog]

Description

Compute the ACF either from velocities (from rkf), the dipole moment (from coordinates and atomic charges in rkf), the dipole moment derivative (from velocities and atomic charges in rkf), from the pressure tensor (from rkf), or from values specified in input. Selecting `DiffusionCoefficient` is equivalent to selecting `Velocities`. The default, `DipoleDerivativeFromCharges`, results in the computation of an IR spectrum. `DipoleMomentFromBinLog` and `ViscosityFromBinLog` allow the relevant properties (dipole moment and pressure tensor respectively) to be read from the `BinLog` section of the trajectory file. In the `BinLog` section requested properties are stored every step (even if `SamplingFreq` was set to a higher number than 1) but only if this was specifically requested at the start of the MD simulation.

Again, a subset of atoms can be selected with the subblock `Atoms`.

The value of the viscosity is written to the output, as well as to the KF file.

If the pressure tensor was written to the `BinLog` section during the MD simulation, then the viscosity can be computed with the keyword `AutoCorrelation%Property` set to *ViscosityFromBinLog*. This is equivalent to setting `AutoCorrelation%Property` to *Viscosity*, but since values are written to the `BinLog` section at every step, there will be more data available.

10.4.13 Diffusion Coefficient

The diffusion coefficient can be computed from a molecular dynamics trajectory in two ways.

1. As the integral over the velocity autocorrelation function.
2. As the slope of the mean square displacement of the atomic coordinates.

The latter is more commonly used, as the former requires trajectory information to be stored at very short time intervals. Note that the obtained values for the diffusion coefficients correspond to the temperature of the molecular dynamics simulation.

From Velocity Autocorrelation

The diffusion coefficient can be defined as an integral over the velocity autocorrelation function.

$$D = \frac{1}{d} \int_{t=0}^{t=t_{max}} \langle \mathbf{v}(0) \cdot \mathbf{v}(t) \rangle dt$$

The factor $\frac{1}{d}$ corrects for the dimension of the system, which we assume to equal the length of the provided vector \mathbf{v} . The dimension d equals 3, unless specified otherwise in the subblock `VecElements`. In a system that is periodic in less than 3 dimensions, it may make sense to provide only the vector elements along the periodic dimensions. By default, however, all vector elements provided are used.

The diffusion coefficient is computed if the task `AutoCorrelation` is selected, and if in the `AutoCorrelation` block `DiffusionCoefficient` is selected as the `Property`. The result is completely equivalent to selecting the task `AutoCorrelation` with `Velocities` as the `Property` keyword.

```
$AMSBIN/analysis <<eor
  Task AutoCorrelation
  AutoCorrelation
    Property DiffusionCoefficient
  End
eor
```

AutoCorrelation

Type

Block

Recurring

True

Description

All input related to auto correlation functions.

Property

Type

Multiple Choice

Default value

DipoleDerivativeFromCharges

Options

[Velocities, DipoleMomentFromCharges, DiffusionCoefficient, DipoleDerivativeFromCharges, PressureTensor, Viscosity, DipoleMomentFromBinLog, ViscosityFromBinLog]

Description

Compute the ACF either from velocities (from rkf), the dipole moment (from coordinates and atomic charges in rkf), the dipole moment derivative (from velocities and atomic charges in rkf),

from the pressure tensor (from rkf), or from values specified in input. Selecting DiffusionCoefficient is equivalent to selecting Velocities. The default, DipoleDerivativeFromCharges, results in the computation of an IR spectrum. DipoleMomentFromBinLog and ViscosityFromBinLog allow the relevant properties (dipole moment and pressure tensor respectively) to be read from the BinLog section of the trajectory file. In the BinLog section requested properties are stored every step (even if SamplingFreq was set to a higher number than 1) but only if this was specifically requested at the start of the MD simulation.

From Mean Square Displacement

The mean square displacement becomes linear with time, after an initial time interval. We can therefore define the linear part of the function as follows.

$$MSD(t) = \langle (\mathbf{r}(0) - \mathbf{r}(t))^2 \rangle = at + b,$$

with a as the slope of the function. The diffusion coefficient is proportional to the slope a .

$$D = \frac{1}{2d}a$$

Here, d is the dimensionality of the system, or the length of the provided vector \mathbf{r} . The dimension d equals 3, unless specified otherwise in the subblock VecElements. In a system that is periodic in less than 3 dimensions, it may make sense to provide only the vector elements along the periodic dimensions. By default, however, all vector elements provided are used.

The diffusion coefficient is computed if the task MeanSquareDisplacement is selected, and if in the MeanSquareDisplacement block DiffusionCoefficient is selected as the Property. The result is completely equivalent to selecting the task MeanSquareDisplacement with Coords as the Property keyword.

```
$AMSBIN/analysis <<eor
  Task MeanSquareDisplacement
  MeanSquareDisplacement
    Property DiffusionCoefficient
  End
eor
```

MeanSquareDisplacement

Type

Block

Recurring

True

Description

All input related to mean square displacement functions.

Property

Type

Multiple Choice

Default value

Coords

Options

[Coords, DiffusionCoefficient, Conductivity]

Description

Compute the MSD from the property selected here (from rkf). Selecting DiffusionCoefficient is equivalent to selecting the property Coords.

In both cases, a subset of atoms can be selected with the subblock `Atoms`, and a subset of vector elements (in this case elements 1=X, 2=Y, 3=Z for the Cartesian coordinates) can be selected with the subblock `VecElements`.

The value of the diffusion coefficient is written to the output, as well as to the KF file.

10.4.14 Ionic Conductivity

Ionic conductivity can be computed from atomic charge weighted diffusion coefficients. In this case, we compute the diffusion coefficients from the mean square displacement. It can be computed by selecting the `Property Conductivity` in the `MeanSquareDisplacement` block.

```
$AMSBIN/analysis <<eor
  Task MeanSquareDisplacement
  MeanSquareDisplacement
    Property Conductivity
  End
eor
```

The set of atoms for which the conductivity is to be computed can be selected as described in the Mean Square Displacement section. If `UseMolecularCentersOfMass` is enabled, then the ionic conductivity is computed per molecule.

Atomic charges can be provided in a system block in the input. The elements in the input system block need to match the elements in the molecular dynamics simulation. If no system block is present in the input, the system block from the molecular dynamics simulation will be used. If no atomic charges are present there either, then the code will attempt to read atomic charges from the `ams.rkf` file.

Note: The ionic conductivity should be computed only for simulations with constant volume and using constant atomic charges. By default, only the ForceField engine keeps the atomic charges constant. Any other engine (e.g. REAXFF) varies the atomic charges throughout the simulation. In those cases it is highly recommended to provide atomic charges in an input system block.

```
#!/bin/sh

"$AMSBIN/analysis" << eor

Task MeanSquareDisplacement

System
  Atoms
    O      -0.7113934019      1.6140994026      -0.7075521284 analysis.
↪charge=-0.8
    H      -0.0425480297      2.3360319223      -0.8302756333 analysis.
↪charge=0.4
    H      -0.5643276324      1.0450643077      -1.4556911815 analysis.
↪charge=0.4
    O      1.1006174021      -7.1200763645      -3.6017659954 analysis.
↪charge=-0.8
    H      1.4647560784      -6.1967314913      -3.5522086623 analysis.
↪charge=0.4
    H      1.8229257860      7.2170281524      -3.7443300356 analysis.
↪charge=0.4
  End
End

TrajectoryInfo
```

(continues on next page)

(continued from previous page)

```

Trajectory
  KFFilename ams.results/ams.rkf
End
End

MeanSquareDisplacement
  Property Conductivity
  Atoms
    Element O
  End
End
eor

```

MeanSquareDisplacement**Type**

Block

Recurring

True

Description

All input related to mean square displacement functions.

Property**Type**

Multiple Choice

Default value

Coords

Options

[Coords, DiffusionCoefficient, Conductivity]

Description

Compute the MSD from the property selected here (from rkf). Selecting DiffusionCoefficient is equivalent to selecting the property Coords.

The ionic conductivity is computed from the mean square displacement, using coordinates multiplied by the charges (atomic or molecular).

$$MSD(t) = \frac{1}{N} \sum_i^N \langle (q_i(0)\mathbf{r}_i(0) - q_i(t)\mathbf{r}_i(t))^2 \rangle$$

Here N is the number of atoms/molecules for which the conductivity is computed, and i runs over those atoms/molecules. The average runs over all time-intervals of length t throughout the simulation. If the charges q_i remain constant throughout the simulation, then they can be moved outside the average.

$$MSD(t) = \frac{1}{N} \sum_i^N q_i^2 \langle (\mathbf{r}_i(0) - \mathbf{r}_i(t))^2 \rangle$$

The charge weighted diffusion coefficient is defined as the time derivative of the MSD, which is assumed constant.

$$D_q = \lim_{t \rightarrow \infty} \frac{1}{2dN} \frac{d \sum_i^N q_i^2 \langle (\mathbf{r}_i(0) - \mathbf{r}_i(t))^2 \rangle}{dt}$$

Here d is the dimensionality of the systems (usually $d = 3$).

The ionic conductivity can now be defined as

$$\sigma = \frac{N}{Vk_B T} D_q$$

Here, V is the volume of the system, k_B is the Boltzmann constant, and T is the average temperature.

10.5 VCD Analysis: VCDtools

The auxiliary program *VCDtools* provides insight regarding the origin of the VCD intensity of a given normal mode. This is done by analyzing and visualizing the various contributions to the total Electric (E) and Magnetic (M) Dipole Transition Moments (DTM). *e.g.*, the contributions from atoms, electrons, nuclei and molecular fragments. The *VCDtools* program is accessible through the AMS-GUI and allows one to:

- 1) Scale the size of the atoms with the magnitude of their atomic EDTM or MDTM contributions.
- 2) Visualize the various contributions to the EDTMs and MDTMs by arrows.
- 3) Perform a General Coupled Oscillator (GCO) analysis to evaluate the interactions between the different parts of a molecule.
- 4) Decompose the total EDTM and MDTM into nuclear and electronic contributions to gain insight into effects induced in the VCD spectra by charge transfer.
- 5) Perform a VCD robustness analysis.

See also:

Tutorial: [Analysis of the VCD spectrum of Oxirane with VCDtools](#)

VCDtools requires only the standard result file *adf.rkf* from an VCD calculation using the *ADF* engine. Once this file is opened with *AMSSpectra* the *VCDtools* program will be accessible through the standard user interface of *AMSSpectra*.

VCDtools produces one ASCII file, *VCDtools.out* where all output from the program is printed. This file will be located in the *.results* folder in the directory of the opened *adf.rkf* result file. Additionally, *VCDtools* will run on the background every time a new normal mode is selected. It then computes the atomic EDTM and MDTM which can be displayed inside *AMSSpectra* in various ways.

The following references should be cited when publishing results obtained with *VCDtools* Refs.¹²³.

10.5.1 General Theory

The VCD intensity associated with the fundamental vibrational transition for a given normal mode j is given by the rotational strength (RS):

$$R_{01}(j) = -i\vec{E}_{01}^{tot}(j) \cdot \vec{M}_{10}^{tot}(j)$$

where $\vec{E}_{01}^{tot}(j)$ and $\vec{M}_{10}^{tot}(j)$ are the total EDTM and MDTM of normal mode j , i is the unit imaginary number.

When the harmonic approximation the total EDTM and MDTM can be written as sums of atomic contributions:

$$\begin{aligned}\vec{E}_{01}^{tot}(j) &= \sum_{\lambda=1}^N \vec{E}_{01}^{\lambda}(j) \\ \vec{M}_{10}^{tot}(j) &= \sum_{\lambda=1}^N \vec{M}_{10}^{\lambda}(j)\end{aligned}$$

where N is the total number of atoms in the molecule and λ runs over all atoms.

¹ V.P. Nicu, *Revisiting an old concept: the coupled oscillator model for VCD. Part 1: the generalised coupled oscillator mechanism and its intrinsic connection to the strength of VCD signals*, *Physical Chemistry Chemical Physics* 18, 21202 (2016) (<https://doi.org/10.1039/C6CP01282E>)

² V.P. Nicu, J. Neugebauer and E.J. Baerends, *Effects of Complex Formation on Vibrational Circular Dichroism Spectra*, *Journal of Physical Chemistry A* 112, 6978 (2008) (<https://doi.org/10.1021/jp710201q>)

³ M.A.J. Koenis, O. Visser, L. Visscher, W.J. Buma, V.P. Nicu, *GUI Implementation of VCDtools, A Program to Analyze Computed Vibrational Circular Dichroism Spectra*, *J. Chem. Inf. Model* 60, 259 (2020) (<https://pubs.acs.org/doi/abs/10.1021/acs.jcim.9b00956>)

10.5.2 General Coupled Oscillator Analysis

The GCO analysis computes the contribution to total rotational strength from the interaction of two molecular fragments. This information provides important insight regarding the source of the VCD intensity and the robustness of the computed VCD sign. In the following, a brief summary of the GCO theory originally published in Ref. [Page 454, 1](#) is given.

Following the original coupled oscillator VCD mechanism, the atoms of a molecule are grouped into three fragments: **A**, **B**, and **R**. The fragments **A** and **B** represent the important Coupling Oscillator (CO) fragments, while fragment **R** is the ancillary fragment containing the rest of the atoms. As a result, the total RS can be decomposed in three components:

$$R_{01}(j) = R_{01}^{\text{GCO}}(j) + R_{01}^{\text{IF}}(j) + R_{01}^{\text{R}}(j)$$

The first contribution $R_{01}^{\text{GCO}}(j)$, is the GCO contribution to the RS and is given by the interaction between the EDTM and MDTM of fragments **A** and **B**:

$$R_{01}^{\text{GCO}}(j) = -i \cdot \left[\vec{E}_{01}^{\text{A}}(j) \cdot \vec{M}_{10}^{\text{B}}(j) + \vec{E}_{01}^{\text{B}}(j) \cdot \vec{M}_{10}^{\text{A}}(j) \right]$$

The $R_{01}^{\text{IF}}(j)$ term represents the contribution from the individual fragments (IF) **A** and **B** without any interaction with other fragments:

$$R_{01}^{\text{IF}}(j) = -i \cdot \left[\vec{E}_{01}^{\text{A}}(j) \cdot \vec{M}_{10}^{\text{A}}(j) + \vec{E}_{01}^{\text{B}}(j) \cdot \vec{M}_{10}^{\text{B}}(j) \right]$$

and the $R_{01}^{\text{R}}(j)$ contribution contains all contributions from fragment **R**:

$$R_{01}^{\text{R}}(j) = -i \cdot \left[\vec{E}_{01}^{\text{A}}(j) \cdot \vec{M}_{10}^{\text{R}}(j) + \vec{E}_{01}^{\text{R}}(j) \cdot \vec{M}_{10}^{\text{A}}(j) \right] \\ - i \cdot \left[\vec{E}_{01}^{\text{B}}(j) \cdot \vec{M}_{10}^{\text{R}}(j) + \vec{E}_{01}^{\text{R}}(j) \cdot \vec{M}_{10}^{\text{B}}(j) + \vec{E}_{01}^{\text{R}}(j) \cdot \vec{M}_{10}^{\text{R}}(j) \right]$$

In the above equations $\vec{E}_{01}^{\text{X}}(j)$ and $\vec{M}_{10}^{\text{X}}(j)$, with $\text{X} = (\text{A}, \text{B}, \text{R})$, are the EDTM and MDTM associated with the three fragments.

Using the origin dependency of the MDTM, the GCO contribution can be rewritten in a form similar to the original CO term:

$$R_{01}^{\text{GCO}}(j) = \frac{\pi \nu_j}{c} \cdot \vec{Y}^{\text{GCO}}(j) \cdot \left[\vec{E}_{01}^{\text{A}}(j) \times \vec{E}_{01}^{\text{B}}(j) \right]$$

where $\vec{Y}^{\text{GCO}}(j)$ is the general coupled oscillator vector.

As shown in ref. [Page 454, 1](#) most normal modes of a molecule (i.e. not just the carbonyl stretching modes) can be interpreted in terms of the GCO mechanism. That is, for most normal modes one can define the GCO fragments **A** and **B** fragments in such a way that the $R_{01}^{\text{GCO}}(j)$ term represent the dominant contributions to the rotational strengths. This second expression for $R_{01}^{\text{GCO}}(j)$ does not depend on the MDTM (which is origin dependent). As such, it allows one to interpret the VCD intensity of a mode in terms of interacting EDTMs that are associated with the various moieties of a molecule and their relative orientation.

The identification of the GCO fragments (which are normal mode dependent) is not always trivial, especially, in molecules without symmetry. Consequently, *VCDtools* offers several options for dividing the molecule into fragments. In *AMSSpectra* these fragments are referred as 'Regions' in analogy to the regions in the *AMSinput* program. Under the **Regions**-menu there are many options to set, alter and save the fragments. Additionally, *VCDtools* is able to make a guess for fragments **A** and **B** as will be discussed in more detail below.

10.5.3 Available options

Beside the standard calculation and visualization of the atomic contributions to the normal mode (NM) motion, EDTM and MDTM, *VCDtools* offers three more advanced tools that can be utilized when analyzing VCD spectra inside *AMSSpectra*:

- GCO Analysis
- Guess GCO Fragments
- NM Localization on Regions

The first option, “*GCO Analysis*”, uses the above equations to decompose the RS in its different contributions. Printing both the values and important angles between the vectors. Before this option can be run two regions inside the molecule should be defined as fragments **A** and **B**. It is important that these regions do not contain the same atoms. Also since the decomposition is different for each NM, a specific NM should be selected.

The second option, “*Guess Fragments*”, guesses which atoms belong the fragments **A** and **B** for a selected normal mode. In doing so the $R_{01}^{\text{GCO}}(j)$ is maximized while keeping $R_{01}^{\text{R}}(j)$ low. Additionally, it ensures that the fragments are localized on a part of the molecule.

The third option, “*NM Localization on Regions*”, computes the percentage of the mass-weighted normal mode motion that is located on the atoms in the selected regions. One or multiple regions can be computed at the same time and the localizations are determined for all modes within a selected frequency window.

EXAMPLES

11.1 Geometry optimization

11.1.1 Example: Simple geometry optimization

Download `GO_formaldehyde_noSCC.run`

```
#!/bin/sh

$AMSBIN/ams << EOF

Task GeometryOptimization

System
  Atoms [Bohr]
    C   0.0   0.0   -1.0
    O   0.0   0.0   1.247
    H   0.0  -1.738  -2.097
    H   0.0   1.738  -2.097
  End
End

Engine DFTB
  ResourcesDir Dresden
  Model DFTB0
  DispersionCorrection Auto
EndEngine

EOF
```

11.1.2 Example: Two-stage geometry optimization with initial Hessian

Download `2StepGO.run`

```
#!/bin/sh

# Preoptimization with DFTB and calculation of the Hessian
# =====
#
# We will reuse the geometry optimized at the DFTB level as a starting point for
# the DFT geometry optimization. We will also calculate the real Hessian with
```

(continues on next page)

(continued from previous page)

```
# DFTB and use that as the initial Hessian for the Quasi-Newton based
# optimization at the DFT level. DFTB is so fast compared to DFT, that all of
# this is basically instantaneous. Our goal here is really just to reduce the
# number of steps in the DFT geometry optimization. If we save just a single
# step there, the initial DFTB calculation will already have paid for itself ...
```

```
AMS_JOBNAME=dftb_preopt $AMSBIN/ams << EOF
```

```
# Specify the system geometry: Aspirin
```

```
System
```

```
  Atoms
```

```
    C      0.000000  0.000000  0.000000
    C      1.402231  0.000000  0.000000
    C      2.091015  1.220378  0.000000
    C      1.373539  2.425321  0.004387
    C     -0.034554  2.451759  0.016301
    C     -0.711248  1.213529  0.005497
    O     -0.709522  3.637718  0.019949
    C     -2.141910  1.166077 -0.004384
    O     -2.727881  2.161939 -0.690916
    C     -0.730162  4.530447  1.037168
    C     -0.066705  4.031914  2.307663
    H     -0.531323 -0.967191 -0.007490
    H      1.959047 -0.952181 -0.004252
    H      3.194073  1.231720 -0.005862
    H      1.933090  3.376356 -0.002746
    O     -2.795018  0.309504  0.548870
    H     -2.174822  2.832497 -1.125018
    O     -1.263773  5.613383  0.944221
    H     -0.337334  4.693941  3.161150
    H      1.041646  4.053111  2.214199
    H     -0.405932  3.005321  2.572927
```

```
  End
```

```
End
```

```
# Do a geometry optimization.
```

```
Task GeometryOptimization
```

```
# Also compute the Hessian at the optimized geometry.
```

```
Properties
```

```
  Hessian True
```

```
End
```

```
# Settings for the DFTB engine:
```

```
Engine DFTB
```

```
  Model DFTB3
```

```
  ResourcesDir DFTB.org/3ob-3-1
```

```
EndEngine
```

```
EOF
```

```
# Geometry optimization with DFT
```

```
# =====
```

```
AMS_JOBNAME=dft_opt $AMSBIN/ams << EOF
```

(continues on next page)

(continued from previous page)

```

# Start from the geometry that is already optimized at the DFTB level.
LoadSystem
  File dftb_preopt.results/dftb.rkf
End
# (equivalent to loading the system from dftb_preopt.results/ams.rkf)

Task GeometryOptimization

GeometryOptimization
  InitialHessian
    # Load the DFTB Hessian as the initial Hessian for the
    # Quasi-Newton based optimizer.
    Type FromFile
    File dftb_preopt.results
  End
End

# Settings for the BAND engine:
Engine BAND
  Basis Type=TZP
  XC GGA=PBE
EndEngine

EOF

```

11.1.3 Example: Periodic lattice optimization under pressure

Download Diamond_under_pressure.run

```

#!/bin/sh

# Calculate the phonon dispersion curves for diamond under pressure.

# Loop over pressure values (in GPa):
for P in -40 0 40 160 ; do
AMS_JOBNAME=pressure_$P $AMSBIN/ams << EOF

  Task GeometryOptimization

  System
    Atoms
      C -0.44625 -0.44625 -0.44625
      C 0.44625 0.44625 0.44625
    End
    Lattice
      0.0 1.785 1.785
      1.785 0.0 1.785
      1.785 1.785 0.0
    End
  End

  GeometryOptimization
    OptimizeLattice Yes
    Convergence Gradients=1e-5 StressEnergyPerAtom=1E-5
    Method Quasi-Newton
  End
End

```

(continues on next page)

(continued from previous page)

```

End

EngineAddons
  Pressure $P
End

Properties
  # Request the calculation of phonons at the optimized geometry.
  Phonons Yes
End

NumericalPhonons
  SuperCell
    2 0 0
    0 2 0
    0 0 2
  End
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/mio-1-1
  KSpace
    Type Symmetric
    Symmetric KInteg=5
  End
  Technical AnalyticalStressTensor=False
EndEngine

EOF
done

```

11.1.4 Example: Phase Transition Due To External Nonuniform Stress

Download `Diamond_from_pressure.run`

```

#!/bin/sh

# Starting from hexagonal graphite we optimize the crystal
# under the influence of a non-uniform stress tensor.
# The system will undergo a phase transition to a diamond structure
AMS_JOBNAME=Graphite2Diamond $AMSBIN/ams << eor

Task GeometryOptimization
EngineAddons
  ExternalStress
    StressTensorVoigt 0 0 0.01 0 -0.0003 0
  End
End
System
  Atoms
    C 0.0000 0.0000 1.6507
    C 0.0000 0.0000 -1.6507
    C 0.0000 -1.4225 1.6507
    C 0.0000 1.4225 -1.6507

```

(continues on next page)

(continued from previous page)

```

End
Lattice
    1.2319  -2.1338  0.0000
    1.2319   2.1338  0.0000
    0.0000   0.0000  6.6029
End
End
GeometryOptimization
    MaxIterations 300
    OptimizeLattice Yes
    Convergence
        Energy 1.0e-6
        Gradients 1.0e-5
        Step 1.0e-4
    End
End
UseSymmetry No
Engine DFTB
    Model SCC-DFTB
    ResourcesDir DFTB.org/3ob-3-1
    DispersionCorrection D3-BJ
    KSpace
        Quality Basic
    End
EndEngine

eor

# The diamond structure is still compressed under the influence of the external
↪ stress.
# We now disable this term and relax the unit cell with the same DFTB model
AMS_JOBNAME=DiamondRelaxation $AMSBIN/ams << eor

Task GeometryOptimization
LoadSystem
    File Graphite2Diamond.results/ams.rkf
End
GeometryOptimization
    MaxIterations 300
    OptimizeLattice Yes
    Convergence
        Energy 1.0e-6
        Gradients 1.0e-5
        Step 1.0e-4
    End
End
UseSymmetry No
LoadEngine Graphite2Diamond.results/dftb.rkf

eor

```

11.1.5 Example: Boron nitride optimization under external stress

Download ExternalStress_BN.run

```
#!/bin/sh

# Some (fairly arbitrary) external stress in atomic units:
external_stress='0.001 0.0003 -0.0007 0.0005 0.0004 0.001'

# The external stress applied here will result in pretty large deformations.
# Since the external stress energy term becomes more and more approximate the
# larger the deformation from the initial cell is, after the first optimization,
# one should run a second geometry optimization starting from the converged
# results of the first one.

# It is always good practice to validate the optimization under external stress
# by computing the stress tensor of the final optimized geometry: the stress
# tensor at the optimized geometry should match the external stress applied during
# the optimization, but with reversed signs.

AMS_JOBNAME=first_go $AMSBIN/ams << eor

Task GeometryOptimization

System
  Atoms
    B 0.0 0.0 0.0
    N 0.905 0.905 0.905
  End
  Lattice
    0.0 1.81 1.81
    1.81 0.0 1.81
    1.81 1.81 0.0
  End
  SuperCell 2 2 2
End

GeometryOptimization
  OptimizeLattice Yes
  # The convergence threshold should be smaller than the applied external stress:
  Convergence Gradients=1.0E-4
End

# We want the external stress to break the symmetry:
UseSymmetry No

EngineAddons
  ExternalStress
    StressTensorVoigt $external_stress
  End
End

Engine ForceField
EndEngine
eor

# The second geometry optimization starting from the results of the first one:
```

(continues on next page)

(continued from previous page)

```

AMS_JOBNAME=second_go $AMSBIN/ams << eor

Task GeometryOptimization

LoadSystem
  File first_go.results/ams.rkf
End

GeometryOptimization
  OptimizeLattice Yes
  Convergence Gradients=1.0E-5
End

UseSymmetry No

EngineAddons
  ExternalStress
    StressTensorVoigt $external_stress
  End
End

LoadEngine first_go.results/forcefield.rkf
eor

# In the final calculation we simply compute the stress tensor of the final
# geometry, which should be the opposite of the external stress tensor applied
# during the optimization.

AMS_JOBNAME=sp $AMSBIN/ams << eor

Task SinglePoint

LoadSystem
  File second_go.results/ams.rkf
End

UseSymmetry No

Properties StressTensor=Yes

LoadEngine first_go.results/forcefield.rkf
eor

echo "Applied external stress tensor (order: xx,yy,zz,yz,xz,xy):"
echo $external_stress
echo "Resulting final stress tensor:"
$AMSBIN/amsreport sp.results/forcefield.rkf -r "AMSResults%StressTensor#12.6f##3"

```

11.1.6 Example: Graphene optimization under external stress

Download ExternalStress_graphene.run

```
#!/bin/sh

# Here we perform an optimization under external stress using the
# UpdateReferenceCell option. When using this option the energy during
# the geometry optimization will not necessary go down, but the final
# stress will match the ExternalStress with much hihger accuracy.

AMS_JOBNAME=go $AMSBIN/ams << eor

Task GeometryOptimization

System
  Atoms
    C 0.0          0.0          0.0
    C 1.23647204352 0.7138774661 0.0
  End
  Lattice
    1.23647204352 -2.1416324015 0.0
    1.23647204352  2.1416324015 0.0
  End
End

UseSymmetry no

EngineAddons
  ExternalStress
    StressTensorVoigt 0.01 0.02 0.003
    UpdateReferenceCell Yes
  End
End

GeometryOptimization
  OptimizeLattice Yes
  Convergence Gradients=1.0E-4
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/pbc-0-3
  KSpace Quality=Normal
EndEngine
eor

# Here we compute the stress tensor at the optimized geometry to validate
# the optimization under stress

AMS_JOBNAME=sp $AMSBIN/ams << eor

Task SinglePoint

LoadSystem
  File go.results/ams.rkf
End
```

(continues on next page)

(continued from previous page)

```

UseSymmetry no

Properties
  StressTensor Yes
End

LoadEngine go.results/dftb.rkf
eor

echo "Resulting final stress tensor:"
$AMSBIN/amsreport sp.results/dftb.rkf -r "AMSResults%StressTensor#12.6f##2"

```

11.1.7 Example: Constrained optimizations

Download constraints.run

```

#!/bin/sh

# This example demonstrates the setup of all different types of constraints.
# Note that all constraints types can be combined with each other, as long as
# the resulting set of constraints actually makes sense. (It must of course be
# possible to satisfy all of them at the same time. AMS is not able to check
# that and you might get really surprising results if that is not the case ...)

# 1. Angle constraints
# =====

AMS_JOBNAME=angle "$AMSBIN/ams" << EOF

  Task GeometryOptimization

  GeometryOptimization
    Convergence Step=1.0e-3
  End

  System
    Atoms
      O   0.001356   0.000999   0.000000
      H   0.994442  -0.037855   0.000000
      H  -0.298554   0.948531   0.000000
    End
  End

  Constraints
    # Fix the H--O--H angle to 125 degrees.
    Angle  3 1 2 125.0
  End

  Engine DFTB
    Model SCC-DFTB
    ResourcesDir Dresden
    DispersionCorrection Auto
  EndEngine

```

(continues on next page)

(continued from previous page)

```

EOF

# 2. Distance constraints
# =====

AMS_JOBNAME=dist "$AMSBIN/ams" << EOF

    Task GeometryOptimization

    GeometryOptimization
        Convergence Step=1.0e-3
    End

    System
        Atoms
            O    0.001356    0.000999    0.000000
            H    0.994442   -0.037855    0.000000
            H   -0.298554    0.948531    0.000000
        End
        BondOrders
            1 2 1.0
            1 3 1.0
        End
    End

    Constraints
        # Fix the OH bond distances to 1.03 Angstrom, for which bonds need to be
        ↪defined in the System block
        All bonds O H to 1.03
        # Alternatively you can list the distances one by one as follows
        # Distance  1 2 1.03
        # Distance  1 3 1.03
    End

    Engine DFTB
        Model SCC-DFTB
        ResourcesDir Dresden
        DispersionCorrection Auto
    EndEngine

EOF

# 3. Dihedral angle constraint
# =====

AMS_JOBNAME=dihed "$AMSBIN/ams" << EOF

    Task GeometryOptimization

    GeometryOptimization
        Convergence Step=1.0e-3
    End

    System
        Atoms

```

(continues on next page)

(continued from previous page)

```

      C   -0.004115   -0.000021   0.000023
      C   1.535711    0.000022   0.000008
      H   -0.399693    1.027812  -0.000082
      H   -0.399745   -0.513934   0.890139
      H   -0.399612   -0.513952  -0.890156
      H   1.931188    0.514066   0.890140
      H   1.931432    0.513819  -0.890121
      H   1.931281   -1.027824   0.000244
    End
  End

  Constraints
    # Fix the dihedral angle H(6)--C(2)--C(1)--H(3) to 20 degrees.
    Dihedral  6 2 1 3 20.00
  End

  Engine DFTB
    Model SCC-DFTB
    ResourcesDir Dresden
    DispersionCorrection Auto
  EndEngine

EOF

# 4a. Fixed atom constraint (Atoms keyword)
# =====

AMS_JOBNAME=atom "$AMSBIN/ams" << EOF

  Task GeometryOptimization

  GeometryOptimization
    Convergence Energy=1.0e-6 Gradients=1.0e-4 Step=1.0e-3
    CoordinateType Cartesian
  End

  System
    Atoms
      C   -0.2460249052   -1.70363153    0.0005128649944
      O   1.152833576    -1.81594932   -0.0004409224206
      C   1.489235475     0.61782051   10.0004771689226
      O   0.5700116914     0.627761615   10.0005491194077
    End
  End

  Constraints
    # Fix atom 1 and 2 at their initial positions.
    Atom 1
    Atom 2
  End

  Engine DFTB
    Model SCC-DFTB
    ResourcesDir DFTB.org/mio-1-1
  EndEngine

```

(continues on next page)

(continued from previous page)

```

EOF

# 4b. Fixed atom constraint (AtomList keyword)
# =====

AMS_JOBNAME=atomlist "$AMSBIN/ams" << EOF

  Task GeometryOptimization

  GeometryOptimization
    Convergence Energy=1.0e-6 Gradients=1.0e-4 Step=1.0e-3
    CoordinateType Cartesian
  End

  System
    Atoms
      C   -0.2460249052   -1.70363153    0.0005128649944
      O    1.152833576    -1.81594932   -0.0004409224206
      C    1.489235475     0.61782051   10.0004771689226
      O    0.5700116914     0.627761615   10.0005491194077
    End
  End

  Constraints
    # Fix atom 1 and 2 at their initial positions.
    AtomList 1:2
  End

  Engine DFTB
    Model SCC-DFTB
    ResourcesDir DFTB.org/mio-1-1
  EndEngine

```

```

EOF

# 4c. Fixed atom constraint (FixedRegion keyword)
# =====

AMS_JOBNAME=region "$AMSBIN/ams" << EOF

  Task GeometryOptimization

  GeometryOptimization
    Convergence Energy=1.0e-6 Gradients=1.0e-4 Step=1.0e-3
    CoordinateType Cartesian
  End

  System
    Atoms
      C   -0.2460249052   -1.70363153    0.0005128649944   region=fixed
      O    1.152833576    -1.81594932   -0.0004409224206   region=fixed
      C    1.489235475     0.61782051   10.0004771689226
      O    0.5700116914     0.627761615   10.0005491194077
    End
  End

```

(continues on next page)

(continued from previous page)

```

Constraints
  # Fix all atoms in region "fixed"
  FixedRegion fixed
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/mio-1-1
EndEngine

EOF

# 4d. Fixed atom constraint (overlapping combination)
# =====

AMS_JOBNAME=combination "$AMSBIN/ams" << EOF

Task GeometryOptimization

GeometryOptimization
  Convergence Energy=1.0e-6 Gradients=1.0e-4 Step=1.0e-3
  CoordinateType Cartesian
End

System
  Atoms
    C   -0.2460249052   -1.70363153    0.0005128649944   region=fixed
    O    1.152833576    -1.81594932   -0.0004409224206   region=fixed
    C    1.489235475     0.61782051   10.0004771689226
    O    0.5700116914     0.627761615   10.0005491194077
  End
End

Constraints
  Atom 1
  AtomList 1 2
  FixedRegion fixed
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/mio-1-1
EndEngine

EOF

# 5. Fixed coordinate constraint
# =====

AMS_JOBNAME=coord "$AMSBIN/ams" << EOF

Task GeometryOptimization

GeometryOptimization

```

(continues on next page)

(continued from previous page)

```

Convergence Energy=1.0e-6 Gradients=1.0e-4 Step=1.0e-3
CoordinateType Cartesian
End

System
  Atoms
    C  -0.2460249052  -1.70363153   0.0005128649944
    O   1.152833576   -1.81594932  -0.0004409224206
    C   1.489235475    0.61782051  10.0004771689226
    O   0.5700116914   0.627761615  10.0005491194077
  End
End

Constraints
  # Fix the x-coordinate of all atoms.
  Coordinate 1 x
  Coordinate 2 x
  Coordinate 3 x
  Coordinate 4 x
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/mio-1-1
EndEngine

```

EOF

```

# 6. Fixed atom constraint (in periodic system)
# =====

```

```

AMS_JOBNAME=pbcatom "$AMSBIN/ams" << EOF

```

```

Task GeometryOptimization

GeometryOptimization
  Convergence Step=1.0e-3
End

System
  Atoms
    C  -1.23  -0.710140830   0.0
    C  -1.23  -0.710140830   3.8
    C   0.0    0.0           0.4
    C   0.0   -1.42028166   3.355
  End

  Lattice
    1.23  -2.130422493309719  0.0
    1.23   2.130422493309719  0.0
  End
End

Constraints
  # Fix atom 1 and 3 at their initial positions.
  Atom 1

```

(continues on next page)

(continued from previous page)

```

    Atom 3
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/mio-1-1
  KSpace Quality=GammaOnly
EndEngine

EOF

# 7. Block constraints (with listing the atoms in a block)
# =====

AMS_JOBNAME=block_list "$AMSBIN/ams" << EOF

Task GeometryOptimization

System
  Atoms
    C   0.5584839616765542   0.5023705181144142  -0.4625483159356394
    C   1.07173137896726    0.2125484528111251  -1.892767990599312
    C   1.699248504588085   -1.006061067555322   -2.191856791501442
    C   2.242484629452111   -1.236470028363516   -3.455616615521399
    C   2.18874580207099    -0.2444337131062739  -4.435483595049287
    C   1.604409798904145    0.9866950282217637  -4.135465239465763
    C   1.061086793296828    1.217355116664161   -2.871773146851866
    H   1.763625603740592   -1.780903563899969   -1.431707209662057
    H   2.716038261390732   -2.190869049673275   -3.672115451399807
    H   2.611833078693977   -0.4241619800888815  -5.420308290235123
    H   1.578029796368043    1.774138556616255   -4.884624561698751
    H   0.6247213391616491    2.187200330357715   -2.64521108544713
    C   1.303528070245188    -0.1416812092038768  0.7303699949711653
    C   0.8164830922475474   -1.314631142230651   1.326337082260565
    C   1.531799364672407    -1.947399963062604   2.342825210379356
    C   2.757684862125068    -1.432061688813837   2.765634667957531
    C   3.271640455523863    -0.2897364031184506  2.150731553729188
    C   2.556535912403799    0.3432056352653093   1.134221563049466
    H  -0.128925843064934    -1.7366201913903     0.9939642396630857
    H   1.133600273086767    -2.849990046242235   2.799740694330775
    H   3.31486005979636    -1.925049398411132   3.557912279830031
    H   4.236604921323707    0.1064455961800578   2.457138367063388
    H   2.976510069814392    1.222131876866508     0.6510413538003352
    C  -0.930165749820548    0.9153412637395284   -0.5420710991631585
    C  -1.791729737216814    0.6892660986048864   0.5418285200469819
    C  -3.111373625199894    1.139542032267652     0.5090625363459357
    C  -3.586568528476239    1.843983986018719    -0.5977864609101087
    C  -2.726152821786783    2.111108432452229    -1.663369105880468
    C  -1.406454626777386    1.660929752085611    -1.63085383469072
    H  -1.428888457076976    0.1571120160719108   1.417905619994904
    H  -3.76723983501283     0.9462006794587581   1.35432032282366
    H  -4.614972346570283    2.194578435055282    -0.6233521468909432
    H  -3.080200905921361    2.678981846821393    -2.520207901691867
    H  -0.7413545301831963    1.891248563160919    -2.459672151335554
    C   1.235557647765805    1.735720249011045     0.1803884343948648
    C   1.377191890012647    1.826646222422494     1.573181692925026

```

(continues on next page)

(continued from previous page)

```

      C   1.905898822116255    2.975086608901246    2.16214311213053
      C   2.280792642899383    4.061906342938987    1.371311861877147
      C   2.105006642447361    3.998471351380415   -0.0115253875199488
      C   1.576317094651283    2.850163227898022   -0.6007264381779673
      H   1.072424817958776    0.9937816064904853    2.202306496283991
      H   2.017471491684088    3.023369029562452    3.242524256706377
      H   2.693031233132915    4.956641734238467    1.830324484771476
      H   2.372569859099136    4.8485771293401     -0.6342066225733602
      H   1.427765851939196    2.820397327218896   -1.677480576376967
    End
  End

  GeometryOptimization
    Convergence
      Energy 1.0e-6
      Gradients 1.0e-4
      Step 1.0e-4
    End
  End

  Constraints
    # Create blocks from the 4 phenyl groups by specifying the atom indices
    # explicitly. (The indices follow the order in the System%Atoms block,
    # where we happen to have the atoms belonging to the different phenyl
    # groups consecutive.)
    BlockAtoms 2 3 4 5 6 7 8 9 10 11 12
    BlockAtoms 13 14 15 16 17 18 19 20 21 22 23
    BlockAtoms 24 25 26 27 28 29 30 31 32 33 34
    BlockAtoms 35 36 37 38 39 40 41 42 43 44 45
  End

  Engine DFTB
    Model DFTB3
    ResourcesDir DFTB.org/3ob-3-1
    DispersionCorrection D3-BJ
  EndEngine

EOF

# 8. Block constraints (with named blocks)
# =====

AMS_JOBNAME=block_names "$AMSBIN/ams" << EOF

  Task GeometryOptimization

  System
    Atoms
      C   0.5584839616765542    0.5023705181144142   -0.4625483159356394
      C   1.07173137896726     0.2125484528111251   -1.892767990599312
    ↪region=phenyl1
      C   1.699248504588085   -1.006061067555322   -2.191856791501442
    ↪region=phenyl1
      C   2.242484629452111   -1.236470028363516   -3.455616615521399
    ↪region=phenyl1
      C   2.18874580207099    -0.2444337131062739   -4.435483595049287

```

(continues on next page)

(continued from previous page)

↪region=phenyl1				
C	1.604409798904145	0.9866950282217637	-4.135465239465763	└
↪region=phenyl1				
C	1.061086793296828	1.217355116664161	-2.871773146851866	└
↪region=phenyl1				
H	1.763625603740592	-1.780903563899969	-1.431707209662057	└
↪region=phenyl1				
H	2.716038261390732	-2.190869049673275	-3.672115451399807	└
↪region=phenyl1				
H	2.611833078693977	-0.4241619800888815	-5.420308290235123	└
↪region=phenyl1				
H	1.578029796368043	1.774138556616255	-4.884624561698751	└
↪region=phenyl1				
H	0.6247213391616491	2.187200330357715	-2.64521108544713	└
↪region=phenyl1				
C	1.303528070245188	-0.1416812092038768	0.7303699949711653	└
↪region=phenyl2				
C	0.8164830922475474	-1.314631142230651	1.326337082260565	└
↪region=phenyl2				
C	1.531799364672407	-1.947399963062604	2.342825210379356	└
↪region=phenyl2				
C	2.757684862125068	-1.432061688813837	2.765634667957531	└
↪region=phenyl2				
C	3.271640455523863	-0.2897364031184506	2.150731553729188	└
↪region=phenyl2				
C	2.556535912403799	0.3432056352653093	1.134221563049466	└
↪region=phenyl2				
H	-0.128925843064934	-1.7366201913903	0.9939642396630857	└
↪region=phenyl2				
H	1.133600273086767	-2.849990046242235	2.799740694330775	└
↪region=phenyl2				
H	3.31486005979636	-1.925049398411132	3.557912279830031	└
↪region=phenyl2				
H	4.236604921323707	0.1064455961800578	2.457138367063388	└
↪region=phenyl2				
H	2.976510069814392	1.222131876866508	0.6510413538003352	└
↪region=phenyl2				
C	-0.930165749820548	0.9153412637395284	-0.5420710991631585	└
↪region=phenyl3				
C	-1.791729737216814	0.6892660986048864	0.5418285200469819	└
↪region=phenyl3				
C	-3.111373625199894	1.139542032267652	0.5090625363459357	└
↪region=phenyl3				
C	-3.586568528476239	1.843983986018719	-0.5977864609101087	└
↪region=phenyl3				
C	-2.726152821786783	2.111108432452229	-1.663369105880468	└
↪region=phenyl3				
C	-1.406454626777386	1.660929752085611	-1.63085383469072	└
↪region=phenyl3				
H	-1.428888457076976	0.1571120160719108	1.417905619994904	└
↪region=phenyl3				
H	-3.76723983501283	0.9462006794587581	1.35432032282366	└
↪region=phenyl3				
H	-4.614972346570283	2.194578435055282	-0.6233521468909432	└
↪region=phenyl3				
H	-3.080200905921361	2.678981846821393	-2.520207901691867	└
↪region=phenyl3				

(continues on next page)

(continued from previous page)

```

      H  -0.7413545301831963  1.891248563160919  -2.459672151335554  _
↪region=phenyl3
      C  1.235557647765805  1.735720249011045  0.1803884343948648  _
↪region=phenyl4
      C  1.377191890012647  1.826646222422494  1.573181692925026  _
↪region=phenyl4
      C  1.905898822116255  2.975086608901246  2.16214311213053  _
↪region=phenyl4
      C  2.280792642899383  4.061906342938987  1.371311861877147  _
↪region=phenyl4
      C  2.105006642447361  3.998471351380415  -0.0115253875199488  _
↪region=phenyl4
      C  1.576317094651283  2.850163227898022  -0.6007264381779673  _
↪region=phenyl4
      H  1.072424817958776  0.9937816064904853  2.202306496283991  _
↪region=phenyl4
      H  2.017471491684088  3.023369029562452  3.242524256706377  _
↪region=phenyl4
      H  2.693031233132915  4.956641734238467  1.830324484771476  _
↪region=phenyl4
      H  2.372569859099136  4.8485771293401  -0.6342066225733602  _
↪region=phenyl4
      H  1.427765851939196  2.820397327218896  -1.677480576376967  _
↪region=phenyl4
      End
    End

    GeometryOptimization
      Convergence
        Energy 1.0e-6
        Gradients 1.0e-4
        Step 1.0e-4
      End
    End

    Constraints
      # Use the region from System%Atoms to set up the block constraints.
      Block phenyl1
      Block phenyl2
      Block phenyl3
      Block phenyl4
    End

    Engine DFTB
      Model DFTB3
      ResourcesDir DFTB.org/3ob-3-1
      DispersionCorrection D3-BJ
    EndEngine

EOF

# 9. Frozen strain components
# =====

AMS_JOBNAME=freezestrain "$AMSBIN/ams" << EOF

```

(continues on next page)

(continued from previous page)

Task GeometryOptimization

GeometryOptimization

OptimizeLattice Yes

Convergence Step=1.0e-3

End

Constraints

Keeps first two lattice vectors orthogonal to the third. Also fixes the
 # length of the third vector, keeping the graphene layer compressed.

FreezeStrain xz yz zz

End

System

Atoms

```

C 1.332002504889882e-05 -0.0005830055256093706 -8.209389319526933e-06
C -1.22799350000696 -0.7102112812520209 2.281155685325205e-06
C -0.0006872840163290542 -0.0003386565731411325 -1.981477647175959
C 1.2274512359848 0.7092866246929653 -1.981478017299119
C 2.455989750017203 -0.000767672446473915 -5.638209535859324e-06
C 1.227983749989149 -0.7105220051279582 3.556077144406634e-06
C 2.45553905980411 -0.0003697961984884611 -1.981476578954899
C 3.68349483597652 0.7093774139714127 -1.981475303736415
C 4.912014119974971 -0.0004697689000645081 8.202057640607653e-06
C 3.68401303002027 -0.7103327188132248 -6.644074866545941e-06
C 4.911561265976663 -0.0002732185613776612 -1.98147535090646
C -3.685503114025999 0.7094747213946447 -1.98147447813657
C -2.457004890026731 -0.0008782302621621878 8.760751751649826e-06
C -3.684994169978904 -0.7103491590560944 -6.913500704937906e-06
C -2.45740142402999 -0.0002120088132086839 -1.981473170030486
C -1.229200584026242 0.709517932531879 -1.98147439816519
C 1.227980230018157 2.127401471357515 -5.950364005944094e-06
C 9.469984377119545e-06 1.417970232416515 5.120417805695729e-06
C 1.227229005981529 2.127790824745807 -1.981476944534885
C 2.45544009594217 2.837313001498961 -1.981464045820237
C 3.683977240012926 2.127396400995821 -4.237131224100653e-06
C 2.456019429974761 1.41770041892015 8.271514976735398e-06
C 3.683520895940616 2.127826615636785 -1.981463536474189
C 4.911484545967099 2.837408990674362 -1.981472216079415
C 6.140019459971655 2.127636216669431 9.289406940173374e-06
C 4.912011129977858 1.417969521782559 7.256699431696856e-06
C 6.139527915931508 2.12792209328836 -1.981460550680031
C -2.457504644023984 2.837506078460876 -1.981475136785154
C -1.229001220032881 2.127025640069692 1.077705178964691e-05
C -2.457025360024441 1.417788944250494 8.010947395781608e-06
C -1.229428944072945 2.128012192586653 -1.981459091806229
C -0.001217694074323372 2.837543459113209 -1.981458639351295
C 2.455982410005773 4.255441598373883 -1.892083560740779e-06
C 1.228003499971814 3.545886142064043 9.237737681677788e-06
C 2.455221785970465 4.255792992279458 -1.981473318340598
C -1.228386974045185 -3.547700260767117 -1.981468190394571
C 4.911976899993052 4.255411828501257 2.27723146438149e-06
C 3.684014579960917 3.545723396055813 1.280915829951697e-05
C 4.911520375955087 4.255828023455356 -1.981468278811
C 1.227655395958869 -3.547614761418386 -1.98146951906497
C -2.457018900008975 4.255512695928259 2.943041159330732e-06
C 6.140026009993287 3.545891232143294 2.20060806891485e-06

```

(continues on next page)

(continued from previous page)

```

C 7.367526315913146 4.255927240986645 -1.981454533470139
C -6.141340994042006 -3.547511474074143 -1.981469232026619
C -0.001002500050462096 4.255387679251578 1.654017565004685e-05
C -1.228981830007242 3.545851372434187 2.37503105142233e-06
C -0.00142595404759982 4.256013860539822 -1.981467396982039
C -3.685044664049419 -3.547477052980626 -1.981466802486946
C -1.227808819999351 -2.12938224692705 -2.127801149456805e-07
C -2.455832350038186 -2.838708610558109 1.251523005803983e-05
C -1.228620264037983 -2.129205540950233 -1.981470550283798
C -0.0004192140835714842 -1.419477849521901 -1.981455609514733
C 1.228193719957573 -2.129406616582517 1.390527520593389e-05
C 0.0001477699611014405 -2.839255138681037 1.274825905530347e-05
C 1.227684425953123 -2.129163359695275 -1.981467635427455
C 2.455626385910209 -1.419413169537493 -1.981453570671329
C -6.140842350045955 -2.129322170430451 1.506226938282425e-05
C 2.456153479955113 -2.839200210115958 1.47101339575357e-05
C 3.683689305925244 -2.129064842197087 -1.981458500475127
C -4.913374384079035 -1.419316522220884 -1.981457095053834
C -3.684843340052955 -2.129350560151249 1.735547419229382e-05
C -4.912808430047692 -2.839071261955975 1.563048016823986e-05
C -3.685268534086676 -2.128978772839927 -1.98145459141338
C -2.45712732409351 -1.419353221465499 -1.981452351546793
C -2.996192032925579e-05 -0.000699242152149526 3.962939886687566
C -1.228003971875175 -0.7103778453492622 3.962925088122617
C -0.000700355908038296 -0.0003332148789394825 1.981452859744668
C 1.227439704045832 0.7092909964964251 1.981467977918086
C 2.455972288079895 -0.000591484995550912 3.962939812711258
C 1.228003298120663 -0.7104549535647978 3.962926452096982
C 2.455542044092204 -0.0003647244468015716 1.981452779807835
C 3.683482354040657 0.7093820057289575 1.981469673526022
C 4.912000348117807 -0.0004686671456845799 3.96292738718769
C 3.683997778083127 -0.7103249572456309 3.962938753391991
C 4.911548884078128 -0.0002680168299863262 1.981457393146928
C -3.685515265933381 0.7094801331342701 1.981461165904004
C -2.457014661901717 -0.0008761383582568633 3.962933787286889
C -3.685007531928472 -0.7103425373777408 3.962942555784025
C -2.457412705931413 -0.0002065770405354501 1.981460520958165
C -1.229211335932976 0.7095228343414065 1.98146103281522
C 1.227967348110556 2.127386193561453 3.962929764672974
C -4.281883530521391e-06 1.417919455416095 3.96292782690195
C 1.227218624090811 2.127796526544795 1.981453236878594
C 2.455427384033809 2.83731889320433 1.981471917810995
C 3.683966438104317 2.127406352683383 3.962931809362267
C 2.456006178071116 1.417740879775001 3.962942690498066
C 3.683508864080636 2.127831077384122 1.981456571921596
C 4.911472874041424 2.837414422441095 1.981469423240322
C 6.140006188100173 2.127640908364754 3.96293316778528
C 4.912000948062944 1.417975193600739 3.962945368689221
C 6.139515384034993 2.127928054994523 1.981471530520008
C -2.457516635928298 2.837511270211092 1.981459501137365
C -1.229012451945974 2.127104470095207 3.962948291371187
C -2.457033631907664 1.417792836176979 3.962935736248542
C -1.229441855971066 2.128017364296635 1.981473516125039
C -0.001231065982844282 2.83754798082405 1.981477377543141
C 2.455970918062962 4.255446730162459 3.962945363741043
C 1.227990978068837 3.545888883850297 3.962943437699987
C 2.455207564067417 4.255798083939164 1.981460905296909

```

(continues on next page)

(continued from previous page)

```

C -1.228397585936881 -3.54769452897796 1.981462310609218
C 4.911964488060532 4.255420650159205 3.962946159605713
C 3.684002968100828 3.545727757822979 3.962932953064215
C 4.911507194030889 4.25583213519047 1.981472875202606
C 1.22764468406568 -3.547609799614339 1.981461473160161
C -2.457027591902453 4.255510887977889 3.962934029953071
C 6.140011718057854 3.545898453762834 3.962947035832434
C 7.367513404018803 4.25593209270268 1.981476837527609
C -6.141352815949912 -3.547506482310592 1.981466583645511
C -0.001013701916043441 4.255385511194097 3.962938483249906
C -1.228995721948309 3.545866003895579 3.962949056786192
C -0.001438315946372004 4.256019262267359 1.98146542401001
C -3.68505504595736 -3.547472521146422 1.981469025022036
C -1.227817711946263 -2.129397354545356 3.962948386486201
C -2.45584760189824 -2.838693919188594 3.962932647309199
C -1.22863031596877 -2.129200899095368 1.981472763926178
C -0.0004312259786934947 -1.419471387806733 1.981476015973903
C 1.228183428085925 -2.129384525179524 3.962937836939656
C 0.0001342880538246494 -2.839225227582055 3.962948356816577
C 1.227671624025677 -2.129158527960262 1.981474583888431
C 2.455614294015247 -1.419407657803005 1.981478001551132
C -6.140855951953014 -2.129314868796188 3.962950599781267
C 2.456139408049645 -2.839212628030558 3.962949726809414
C 3.683677514018658 -2.12905939044384 1.98147688315868
C -4.913386125957436 -1.419311390469788 1.981469050050141
C -3.68485488196171 -2.129348928295441 3.96295344930467
C -4.91282016191732 -2.839063750265042 3.962938899273355
C -3.685281025994509 -2.12897353111013 1.981481200585353
C -2.457141595966529 -1.419347949824808 1.981472029890663

End
Lattice
  9.825000579999999 0.0 0.0
  4.91182904 8.51302256 0.0
  0.0 0.0 8.0

End
End

Engine DFTB
  Model DFTB
  ResourcesDir DFTB.org/mio-1-1
  KSpace
  Quality GammaOnly
End
EndEngine

EOF

# 10. Equalized strain components
# =====

AMS_JOBNAME=equalstrain "$AMSBIN/ams" << EOF

Task GeometryOptimization

GeometryOptimization
  OptimizeLattice Yes

```

(continues on next page)

(continued from previous page)

```
Convergence Step=1.0e-3
End

Constraints
  # Keep the cell cubic, but allow the size of the cube to vary.
  FreezeStrain xy xz yz
  EqualStrain  xx yy zz
End

System
  Atoms
    C -0.132285 3.230196 3.399625
    H 0.67231 2.571995 3.747816
    H -0.546925 3.782407 4.25108
    H -0.921872 2.627955 2.935193
    H 0.267346 3.938428 2.664409
    C 2.647972 3.79511 0.161215
    H 2.745753 2.707187 0.254494
    H 2.108302 4.189352 1.030219
    H 2.093026 4.03506 -0.753279
    H 3.644808 4.248843 0.113424
    C -3.290954 -3.607704 -3.419879
    H -4.099867 -4.000479 -4.046956
    H -2.386445 -3.482146 -4.026458
    H -3.088346 -4.309774 -2.602634
    H -3.589157 -2.638419 -3.003466
    C -3.900392 1.971446 -2.092972
    H -2.821972 1.97396 -1.895882
    H -4.303924 2.977719 -1.930439
    H -4.394183 1.265761 -1.414725
    H -4.081488 1.668344 -3.130841
    C -3.143958 -3.520015 3.393796
    H -3.128547 -4.088022 2.456262
    H -3.994325 -3.845525 4.004334
    H -3.24151 -2.450891 3.171766
    H -2.211449 -3.695621 3.94282
    C -0.31406 -0.626145 3.522914
    H -0.044022 0.228271 4.154493
    H -1.353702 -0.912447 3.720437
    H 0.346904 -1.471041 3.749005
    H -0.205421 -0.349364 2.467723
    C 3.411151 -3.454122 0.161835
    H 2.877462 -2.569463 0.528433
    H 4.211866 -3.141886 -0.518755
    H 2.711585 -4.107617 -0.372385
    H 3.843691 -3.997522 1.010048
    H -3.283653 -0.451758 -4.172013
    H -1.922139 0.6502520000000001 -3.802207
    H -2.586463 -0.412172 -2.523601
    C -2.360175 -0.332772 -3.593334
    H -1.648446 -1.117408 -3.875514
    C 3.046249 -3.33059 3.76859
    H 2.414628 -3.18136 2.88506
    H 2.465863 -3.831302 4.55235
    H 3.39517 -2.358701 4.136729
    H 3.909333 -3.950995 3.500222
    C -3.086408 3.73574 0.4638
```

(continues on next page)

(continued from previous page)

```

H -2.559805 3.990117 -0.463465
H -2.394025 3.813016 1.310247
H -3.469203 2.710618 0.397221
H -3.922599 4.429208 0.611196
C 3.736451 0.338903 -0.234383
H 4.139844 -0.659226 -0.441391
H 4.286082 0.789262 0.6004350000000001
H 2.675343 0.256879 0.028602
H 3.844535 0.968696 -1.125179
C -0.953217 3.761489 -3.029722
H -0.738671 2.687271 -2.986546
H -2.017033 3.913231 -3.24677
H -0.349572 4.223228 -3.819817
H -0.707592 4.222226 -2.065757
C 3.438238 3.368005 3.536049
H 3.718968 3.030104 2.531632
H 4.305113 3.831685 4.021198
H 3.102844 2.509703 4.129906
H 2.62603 4.10053 3.461459
C -0.093351 2.447961 0.147782
H 0.412783 2.191741 -0.790311
H -1.100739 2.015519 0.149423
H -0.163522 3.53817 0.239205
H 0.478074 2.046413 0.99281
C -0.067378999999999999 -1.067744 -0.644773
H 0.831493 -1.69444 -0.611303
H -0.920384 -1.630791 -0.248288
H -0.271712 -0.77696 -1.681851
H 0.091087 -0.168785 -0.037648
C -3.13266 0.095347 1.684164
H -2.468956 -0.758832 1.506078
H -3.797646 -0.127993 2.526632
H -3.731911 0.285949 0.786162
H -2.532126 0.982263 1.917783
C -3.650862 -2.700373 -0.074687
H -4.155919 -2.467829 0.87013
H -2.740814 -3.276506 0.129574
H -3.385824 -1.767849 -0.58654
H -4.32089 -3.289309 -0.711913
C 3.803884 3.754796 -3.348637
H 3.946713 2.667857 -3.350306
H 2.769436 3.987861 -3.626912
H 4.48894 4.214494 -4.07059
H 4.010448 4.148971 -2.34674
C 2.868209 0.11231 2.894284
H 2.317604 0.914041 2.388417
H 2.406222 -0.090550000000000001 3.867554
H 3.909847 0.420825 3.041412
H 2.839161 -0.795073999999999999 2.279753
C -0.320765 -3.560008 1.887422
H -0.965068 -2.68183 2.011986
H 0.02585 -3.901247 2.869895
H 0.543227 -3.294315 1.26713
H -0.88707 -4.36264 1.400678
C 2.415398 -1.437717 -2.776235
H 1.964383 -1.676188 -3.746573
H 2.445411 -2.340763 -2.155392

```

(continues on next page)

(continued from previous page)

```

H 1.816728 -0.668767 -2.274091
H 3.43507 -1.065149 -2.928883
C -3.625996 2.934989 3.78523
H -4.070333 2.734452 2.803299
H -3.043299 2.064066 4.107404
H -4.421782 3.131813 4.513121
H -2.968572 3.809626 3.717096
C 1.422335 1.538945 -3.931672
H 0.608488 0.8054 -3.894205
H 2.3282 1.060135 -4.321529
H 1.616409 1.921293 -2.92272
H 1.136242 2.368954 -4.588236
C 0.028875 -3.521123 -2.677443
H 0.240436 -2.624091 -3.271089
H -0.857142 -3.347281 -2.055678
H 0.888225 -3.744967 -2.034598
H -0.156019 -4.368152 -3.348409

End
Lattice
10.0 0.0 0.0
0.0 10.0 0.0
0.0 0.0 10.0

End
End
Engine DFTB
Model DFTB
ResourcesDir DFTB.org/mio-1-1
KSpace
Quality GammaOnly
End
EndEngine
EOF

```

11.2 Transition state search

11.2.1 Example: TS search starting from initial Hessian

Download COChainFreqTS.run

```

#!/bin/sh

# This example demonstrates in the first step how to calculate the Hessian.
# The second run uses the pre-calculated Hessian and performs a transition
# state search along the frequency mode with the smallest frequency.

# First run: Calculate Hessian
# =====

AMS_JOBNAME=hessian $AMSBIN/ams << EOF

```

(continues on next page)

(continued from previous page)

```

Task SinglePoint

Properties
  Hessian True
End

System
  Atoms
    C  0.0  0.0  0.0
    O  1.5  0.5  0.0
  End
  Lattice
    3.2  0.0  0.0
  End
End

Engine Band
  Basis Type=DZP
  KSpace Quality=Good
EndEngine

EOF

# Second run: TS search with initial Hessian
# =====

AMS_JOBNAME=TS $AMSBIN/ams << EOF

Task TransitionStateSearch

System
  Atoms
    C  0.0  0.0  0.0
    O  1.5  0.5  0.0
  End
  Lattice
    3.2  0.0  0.0
  End
End

GeometryOptimization
  Convergence Gradients=1.0e-4
  InitialHessian
    # Load the pre-calculated Hessian as the initial Hessian for the
    # transition state search using the Quasi-Newton based optimizer.
    Type FromFile
    File hessian.results/band.rkf
  End
End

Properties
  # Also calculate normal modes in the end, so we can see if we actually
  # found a transition state.
  NormalModes True
End

```

(continues on next page)

(continued from previous page)

```

Engine Band
  Basis Type=DZP
  KSpace Quality=Good
EndEngine

EOF

```

11.2.2 Example: PES scan and TS search for H2 on graphene

Download PESScan_and_TS_H2_on_Graphene.run

```

#!/bin/sh

# First we do a 2D PES scan varying the z-coordinate of the two hydrogen atoms
# In this example we will keep the graphene slab fixed. From a physical/chemical
# standpoint this is not a good approximation. The graphene slab is
# intentionally not perfectly symmetric.

AMS_JOBNAME=PESScan $AMSBIN/ams << EOF

Task PESScan

System
  Atoms
    H 0.0 1.53633037 1.1
    H 0.0 -0.11341359 1.1
    C 0.001 1.42028166 0.0
    C 1.230 2.13042249 0.0
    C 1.230 -0.71014083 0.0
    C 2.460 0.00000000 0.0
    C 2.460 1.42028167 0.0
    C 0.000 0.00000000 0.0
  End
  Lattice
    3.69 -2.13042249 0.0
    0.00 4.26084499 0.0
  End
End

PESScan
  ScanCoordinate
    nPoints 10
    Coordinate 1 Z 1.1 2.0
  End
  ScanCoordinate
    nPoints 10
    Coordinate 2 Z 1.1 2.0
  End
End

GeometryOptimization
  Convergence Step=1.0e-3
End

Constraints

```

(continues on next page)

(continued from previous page)

```

Task SinglePoint

System
  # Load the geometry we just saved.
  GeometryFile initial_geometry_for_TS.xyz
End

Properties
  # Calculate the Hessian (implied when calculating normal modes) ...
  NormalModes True
  # ... but only the part related to the hydrogen atoms.
  SelectedRegionForHessian H2
End

Engine DFTB
  Model DFTB
  ResourcesDir DFTB.org/3ob-3-1
  DispersionCorrection D3-BJ
  KSpace
    Type Symmetric
    Symmetric KInteg=3
  End
EndEngine

EOF

echo "Extract the frequencies from the kf file using amsreport:"
$AMSBIN/amsreport Hessian.results/dftb.rkf -r "Vibrations%Frequencies[cm-1]##1"

# Do a transition state search using the initial Hessian just computed (the
# Graphene slab is constrained). Also compute the final Hessian for the
# hydrogen atoms to validate the TS.

AMS_JOBNAME=TS $AMSBIN/ams << EOF

Task TransitionStateSearch

System
  # Load the geometry we just saved.
  GeometryFile initial_geometry_for_TS.xyz
End

GeometryOptimization
  Quasi-Newton
    Step TrustRadius=0.05
  End
  Convergence Gradients=1.0e-4
  InitialHessian
    # Load previously calculated Hessian as initial Hessian for a
    # transition state search with the Quasi-Newton optimizer.
    Type FromFile
    File Hessian.results/dftb.rkf
  End
End

```

(continues on next page)

(continued from previous page)

```
TransitionStateSearch
  # Follow the mode with the smallest frequency.
  ModeToFollow 1
  # (This is also the default, we wouldn't need to specify this.)
End

Constraints
  # Fix the entire graphene slab.
  Atom 3
  Atom 4
  Atom 5
  Atom 6
  Atom 7
  Atom 8
End

Properties
  NormalModes Yes
  SelectedRegionForHessian H2
End

Engine DFTB
  Model DFTB
  ResourcesDir DFTB.org/3ob-3-1
  DispersionCorrection D3-BJ
  KSpace
    Type Symmetric
    Symmetric KInteg=3
  End
EndEngine

EOF

echo "Extract energy from the rkf file using amsreport:"
$AMSBIN/amsreport TS.results/dftb.rkf -r "AMSResults%Energy"
```

11.3 Nudged Elastic Band (NEB)

11.3.1 Nudged Elastic Band (NEB) Examples

Here are a few examples showing how the NEB method can be used to obtain the the path and transition state of a reaction.

See also:

Nudged Elastic Band (NEB) Documentation (page 83)

HCN isomerization reaction with NEB

Download `NEB_HCN.run`

H2 dissociation on graphene

Download `NEB_H2_on_graphene.run`

Running multiple NEB calculations using PLAMS

Download `NEB.plms`

This example should be executed using [PLAMS](#).

See also:

[PLAMS documentation](#) and [tutorial](#)

11.4 Intrinsic reaction coordinate (IRC)

11.4.1 Example: IRC for HCN

Download `IRC_HCN.run`

```
#!/bin/sh

# == IRC scan of the reaction path ==

# The IRC calculation is split in two steps to illustrate the Restart feature.

# In the first calculation only a few points are computed along the so-called
# 'forward' path. The definition of which is 'forward' and which is
# 'backward' depends on the sign of the largest component of the normal mode
# corresponding to the reaction coordinate.

# The RKF file from this partial IRC scan serves as restart file
# for the next calculations that will continue the IRC scan.

# The 'MaxPoints' key in the IRC block is used to limit the number of IRC
# points to compute.

AMS_JOBNAME=irc1 $AMSBIN/ams << eor

Task IRC
System
  Atoms
    C      0.000000000000      0.000000000000      0.000000000000
    N      0.000000000000      0.000000000000     -1.182644220000
    H     -1.103250760411      0.000000000000     -0.322462130000
  End
End
```

(continues on next page)

(continued from previous page)

```

IRC
  MaxPoints 5
  Direction Forward
  CoordinateType Cartesian
  InitialHessian
    Type Calculate
  End
End

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-3-1
EndEngine

eor

# In the second IRC run, the IRC scan is finished. We start with the RKF file
# from the previous run and omit the MaxPoints from the settings, which means
# that the default 100 will be used. Note that the 100 also includes any points
# computed in the previous calculation. The program starts on
# the forward path, continuing where the first calculation had stopped,
# and completes it. Since we set the Direction to Both
# then AMS proceeds to the backward path. After both paths are finished a summary
# of the path characteristics is printed at the end of the output file.

AMS_JOBNAME=irc2 $AMSBIN/ams << eor

Task IRC
System
  Atoms
    C      0.000000000000      0.000000000000      0.000000000000
    N      0.000000000000      0.000000000000     -1.182644220000
    H     -1.103250760411      0.000000000000     -0.322462130000
  End
End

IRC
  Restart
    File irc1.results/ams.rkf
  End
  ! Change options from the ones found in the restart file
  ! (MaxIRCPoints and MaxPoints will be reset to defaults automatically)
  Direction Both
End

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-3-1
EndEngine

eor

```

11.4.2 Example: TS and IRC for Claisen reaction

Download TS_and_IRC_Claissen.run

```

#!/bin/sh

# Transition State Search (TS search) followed by Intrinsic Reaction
# Coordinates (IRC) for two similar Claisen rearrangement reactions.

# =====
# Claisen rearrangement from C=CCC1C=CC=CC1=O to C=CCOc1ccccc1
# =====

AMS_JOBNAME=TS_molecule $AMSBIN/ams << eor

Task TransitionStateSearch

System
  Atoms
    C -1.6622561642524 -1.4933421191817 0.6484353677288
    C -2.6070283916282 -1.7718977641902 -0.3933564306530
    C -2.7546368861548 -3.0534770331072 -0.8757259422474
    C -1.9443405437492 -4.1131780924428 -0.3870796280948
    C -1.0139402388630 -3.8827189276564 0.5979937257975
    C -0.7543606665660 -2.5518266272325 1.0788971265869
    H -3.2590342954832 -0.9734836183880 -0.7410179986476
    H -3.5074724102535 -3.2690301027846 -1.6324229738243
    H -2.1044351565280 -5.1220091436618 -0.7615296311573
    H -0.4323033724363 -4.7107876462166 1.0021729248699
    H -0.3533176373037 -2.4841001370767 2.0927436180830
    O -1.5058234397159 -0.2844936546832 1.1029538316409
    C 0.1375150486634 0.3928947854321 0.4626789880484
    C 1.0578648498087 -0.6180364119671 0.7737587143345
    C 0.8861173890663 -1.8991002496105 0.2125497351161
    H 0.5725481135101 -1.9591100578644 -0.8292382400922
    H -0.2171264706145 0.4859594211539 -0.5641152540806
    H 0.1902646842718 1.3359879065177 1.0025369222419
    H 1.5779002347488 -0.5540482019233 1.7307360489935
    H 1.6031168776346 -2.6724262749099 0.4842127285858
  End
End

Properties NormalModes=Yes

GeometryOptimization
  InitialHessian Type=Calculate
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/3ob-3-1
EndEngine
eor

AMS_JOBNAME=IRC_molecule $AMSBIN/ams << eor

```

(continues on next page)

(continued from previous page)

```

Task IRC

IRC
  MaxIterations 1000
  InitialHessian
    Type FromFile
    File TS_molecule.results
  End
end
GeometryOptimization
  Convergence
    Gradients 2e-4
  End
End

LoadSystem
  File TS_molecule.results/ams.rkf
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/3ob-3-1
EndEngine
eor

# =====
# Claisen rearrangement for a (periodic) polymer containing the same aromatic
# ring of the previous calculation (from C=CCC1C=CC=CC1=O to C=CCOc1cccc1)
# =====

AMS_JOBNAME=TS_polymer $AMSBIN/ams << eor

Task TransitionStateSearch

System
  Atoms
    C  9.4367476128766   1.6156795441351   0.8542644025030
    C  8.6813349262903   0.7302865575002   0.0170374963868
    C  9.3238583586638  -0.1438417574104  -0.8314773024306
    C  10.741095802442  -0.1538813927018  -0.9297572682512
    C  11.499909556383   0.6929842950328  -0.1576524435776
    C  10.884234857485   1.6861365363275   0.6817028097694
    C  7.1928255650616   0.6847044785103   0.0090333093191
    H  8.7472521390038  -0.8479982636647  -1.4293934603813
    H  11.224348014063  -0.8771406786637  -1.5831686891584
    C  12.930798908912   0.8113031163622  -0.1971191730819
    H  11.474366091679   2.0120169601893   1.5412690799783
    O  8.8401614254340   2.4539245100299   1.6503819438045
    C  9.0152184164720   4.1939147755043   0.9354292967913
    C  10.386800460742   4.3070474573655   0.6675488559614
    C  10.983233806221   3.4166811386158  -0.2473786535066
    H  10.425317882010   3.1661173633779  -1.1491528320010
    H  8.3301833043316   4.0024301166114   0.1091130317749
    H  8.5863952766270   4.7863410214548   1.7409077918798
    H  11.023307058306   4.7274400682699   1.4475913648158
    H  12.053474297431   3.5124067242738  -0.4244212846926

```

(continues on next page)

(continued from previous page)

```

      C 13.897191373368 -0.0308805104165 -0.7206407708622
      C 15.267346128343  0.2274543686942 -0.4348415320624
      C 16.387786908031 -0.4341710477973 -0.8962646879782
      C  6.2048264493065 -0.0924013581064 -0.6266734325417
      C  4.8694123304551  0.1638639419185 -0.3442585289372
      C  3.7074226634814 -0.4584344973285 -0.8689206409329
      C  2.4653410147781  0.0102007864139 -0.4848760767909
      H 16.298210512111 -1.2787892169088 -1.5797017846452
      H  3.7984002771976 -1.2806821291979 -1.5787594353897
      H  6.8240750625775  1.4435007199212  0.7047818340481
      H 13.314194383001  1.6434188966421  0.3984353613768
      H 13.626424975736 -0.8770823766787 -1.3528414377622
      H 15.451469431625  1.0743039058568  0.2322503424471
      H  2.4641684561885  0.8477050600186  0.2182734314177
      H  6.4819803798672 -0.8741436365939 -1.3339335062636
      H  4.6807561767470  0.9698264886906  0.3703766590249
    End
    Lattice
      15.210 0.0 0.0
    End
  End

Properties
  NormalModes Yes
End

GeometryOptimization
  Method Quasi-Newton
  InitialHessian Type=Calculate
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/3ob-3-1
EndEngine
eor

AMS_JOBNAME=IRC_polymer $AMSBIN/ams << eor

Task IRC

IRC
  MaxIterations 1000
  InitialHessian
    Type FromFile
    File TS_polymer.results
  End
  Direction Forward
end

LoadSystem
  File TS_polymer.results/ams.rkf
End

Engine DFTB
  Model SCC-DFTB

```

(continues on next page)

(continued from previous page)

```
ResourcesDir DFTB.org/3ob-3-1
EndEngine
eor
```

11.5 PES scan

11.5.1 Example: Linear transit

Download LinearTransit.run

```
#!/bin/sh

echo "====="
echo "HCN isomerization"
echo "====="
echo

AMS_JOBNAME=HCN_isomerization $AMSBIN/ams << EOF

Task PESScan
# (Linear transit is just a PES scan with 1 scan coordinate.)

System
  Atoms
    C      0.00000000    0.00000000    1.04219000
    H      0.00000000    0.00000000   -0.03324000
    N      0.00000000    0.00000000    2.20064000
  End
End

PESScan
  ScanCoordinate
    nPoints 25
    Angle  2 1 3 180.0 0.0
  End
End

Engine DFTB
  Model DFTB0
  ResourcesDir DFTB.org/mio-1-1
EndEngine

EOF

echo
echo "====="
echo "Water angle transit"
echo "====="
echo
```

(continues on next page)

(continued from previous page)

```
AMS_JOBNAME=water_angle $AMSBIN/ams << EOF
```

```
Task PESScan
```

```
System
```

```
Atoms
```

O	0.00000000	0.00000000	0.59372000
H	0.00000000	0.76544000	-0.00836000
H	0.00000000	-0.76544000	-0.00836000

```
End
```

```
End
```

```
PESScan
```

```
ScanCoordinate
```

```
nPoints 25
```

```
Angle 2 1 3 80.0 180.0
```

```
End
```

```
End
```

```
GeometryOptimization
```

```
! Delocalized coordinates currently have a problem with linear systems.
```

```
! So we will use cartesian coordinates here.
```

```
CoordinateType Cartesian
```

```
End
```

```
Engine DFTB
```

```
Model DFTB0
```

```
ResourcesDir DFTB.org/mio-1-1
```

```
EndEngine
```

```
EOF
```

```
echo
```

```
echo "=====
```

```
echo "Hydrocarbon reaction"
```

```
echo "=====
```

```
echo
```

```
AMS_JOBNAME=hydcarb $AMSBIN/ams << EOF
```

```
Task PESScan
```

```
System
```

```
Atoms
```

C	0.14667300	-0.21503500	0.40053800
C	1.45297400	-0.07836900	0.12424400
C	2.23119700	1.15868100	0.12912100
C	1.78331500	2.39701500	0.38779700
H	-0.48348000	0.63110600	0.67664100
H	-0.33261900	-1.19332100	0.35411600
H	2.01546300	-0.97840100	-0.14506700
H	3.29046200	1.03872500	-0.12139700
H	2.45728900	3.25301000	0.35150400
H	0.74193400	2.60120700	0.64028800

(continues on next page)

(continued from previous page)

```

      C      -0.75086900      1.37782400      -2.43303700
      C      -0.05392100      2.51281000      -2.41769100
      H      -1.78964800      1.33942600      -2.09651100
      H      -0.30849400      0.43896500      -2.76734700
      H      -0.49177100      3.45043100      -2.06789100
      H       0.98633900      2.54913500      -2.74329400
    End
  End

  PESScan
    ScanCoordinate
      nPoints 25
      Distance 1 11 3.36 1.538
      Distance 4 12 3.36 1.538
    End
  End

  Engine DFTB
    Model DFTB0
    ResourcesDir DFTB.org/mio-1-1
  EndEngine

EOF

echo
echo "=====
echo "Retinal trans -> 11-cis isomerization"
echo "=====
echo

AMS_JOBNAME=retinal_transcis $AMSBIN/ams << EOF

Task PESScan

System
  Atoms
    H      -2.10968473      -1.58238733      0.78224517
    C      -2.10306857      -0.54058322      0.46363503
    C      -0.89436995      0.04807217      0.25528247
    H      -0.85555481      1.05432693      -0.15803658
    C      0.38987539      -0.58661182      0.49038464
    C      1.53213446      0.09657801      0.14394773
    H      1.40518949      1.08783970      -0.29205231
    H      3.05232192      -1.34477492      0.72115301
    C      2.88311454      -0.36358433      0.28105432
    C      3.96024700      0.37378345      -0.12385974
    H      3.77965758      1.35231793      -0.56821856
    C      5.34627719      -0.04025647      -0.02249097
    C      6.32191717      0.80135945      -0.49190463
    H      6.00090638      1.74979100      -0.92101391
    C      -4.46825064      -0.90426552      -0.39585925
    C      -5.87277429      -0.25303564      -0.45007491
    C      -3.41139545      0.06493448      0.19516310
    C      -3.67932839      1.38221399      0.41656971
    C      -5.81598497      1.19032366      -0.92660753

```

(continues on next page)

(continued from previous page)

```

C      -5.00049358      2.01922634      0.05561242
C      -4.58391145      -2.18782901      0.46346394
C      -4.01729542      -1.30039402     -1.82272212
C      -2.72429960      2.32303313      1.10290124
C       0.40919453      -1.96244629      1.09501374
C       5.64155973      -1.38034133      0.59419110
C       7.76996060      0.56699126     -0.48750226
O       8.57693167      1.36615612     -0.92976322
H      -6.51997817      -0.84904979     -1.10100203
H      -6.32039371      -0.28079023      0.54871092
H      -5.36159995      1.23817633     -1.92112092
H      -6.82595442      1.60207678     -1.01946858
H      -5.58216571      2.18390764      0.97424181
H      -4.81292271      3.01993001     -0.35246294
H      -4.74166770      -1.94289144      1.51126095
H      -5.43008715      -2.78247632      0.12572479
H      -3.69644845      -2.81116549      0.38705593
H      -3.02900804      -1.75403268     -1.79820003
H      -4.71056940      -2.01489741     -2.26202914
H      -3.97070839      -0.42860260     -2.47090348
H      -2.16469005      2.92261100      0.38111736
H      -3.27791517      3.02297911      1.72885233
H      -2.00470188      1.79865198      1.72726573
H      -0.13689001      -1.97717074      2.03825359
H      -0.07664772      -2.68134154      0.43362393
H       1.41837401      -2.31391556      1.28591185
H       5.15278730      -2.17622743      0.03222328
H       6.70436647      -1.59729505      0.62729622
H       5.25700064      -1.42489613      1.61313095
H       8.12614442      -0.41441814     -0.04549414
End
End

PESScan
  ScanCoordinate
    nPoints 25
    Dihedral  6 9 10 12 180 0
    Dihedral  8 9 10 11 180 0
  End
End

Engine DFTB
  Model DFTB0
  ResourcesDir DFTB.org/mio-1-1
EndEngine
EOF

```


11.5.2 Example: 2D PES scan

Download PESScan.run

```
#!/bin/sh

echo "====="
echo "Ethane torsion"
echo "====="
echo

AMS_JOBNAME=ethane_torsion $AMSBIN/ams << EOF

Task PESScan

System
  Atoms
    C  0.0      0.0      0.76576
    C  0.0      0.0     -0.76576
    H -0.88668938 0.51193036  1.16677
    H  0.88668938 0.51193036  1.16677
    H  0.0      -1.02386071  1.16677
    H  0.0       1.02386071 -1.16677
    H -0.88668938 -0.51193036 -1.16677
    H  0.88668938 -0.51193036 -1.16677
  End
End

PESScan
  # First scan coordinate: C--C bond distance
  ScanCoordinate
    nPoints 5
    Distance 1 2 1.3 1.7
  End
  # Second scan coordinate: One of the H--C--C--H dihedral angles (others will
  ↳ follow naturally)
  ScanCoordinate
    nPoints 21
    Dihedral 3 1 2 6 60.0 0.0
  End
End

GeometryOptimization
  Convergence Step=1.0e-3
End

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-3-1
  DispersionCorrection D3-BJ
EndEngine

EOF

echo "====="
echo "Ethene torsion"
```

(continues on next page)

(continued from previous page)

```

echo "=====
echo

AMS_JOBNAME=ethene_torsion $AMSBIN/ams << EOF

Task PESScan

System
  Atoms
    C  0.0  0.0      0.66687
    C  0.0  0.0     -0.66687
    H  0.0  0.92974 -1.23912
    H  0.0  0.92974  1.23912
    H  0.0 -0.92974  1.23912
    H  0.0 -0.92974 -1.23912
  End
End

PESScan
  # First scan coordinate: C--C bond distance
  ScanCoordinate
    nPoints 5
    Distance 1 2 1.1 1.8
  End
  # Second scan coordinate: Two of the H--C--C--H dihedrals
  ScanCoordinate
    nPoints 21
    Dihedral 4 1 2 3 0.0 60.0
    Dihedral 5 1 2 6 0.0 60.0
  End
End

GeometryOptimization
  Convergence Step=1.0e-3
End

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-3-1
  DispersionCorrection D3-BJ
EndEngine

```

```

EOF

```

```

# Below are more technical examples, demonstrating the PES scan gap filling.
# The QUASINANO2015 parameter set shows some discontinuities in the PES,
# which leads to problems with convergence. The first job leaves the
# non-converged steps as is while the second job instructs AMS to
# attempt a second optimization for non-converged point starting from
# a different initial geometry.

```

```

echo "=====
echo "Ethane gap filling test (1/2)"
echo "=====
echo

```

(continues on next page)

(continued from previous page)

```
AMS_JOBNAME=ethane_nofillgaps $AMSBIN/ams << EOF
```

```
Task PESScan
```

```
System
```

```
Atoms
```

```

C -2.333834610464788 -2.268837915270455 -0.2417723425321957
C -0.8081611038872945 -2.334371994724881 -0.04271045326758349
H -0.2505615773096904 -1.473443563856088 -0.38077110593546
H -0.3249814761083244 -3.235478579439597 -0.3904810245975267
H -0.583247370537557 -2.349691649662279 1.013499336841977
H -2.817014238243758 -1.367731330555738 0.1059982287977475
H -2.891434137042391 -3.129766346139247 0.09628831013568076
H -2.558748343814525 -2.253518260333056 -1.297982132641757
```

```
End
```

```
End
```

```
GeometryOptimization
```

```
CoordinateType Cartesian
```

```
Convergence Step=1.0e-3
```

```
End
```

```
PESScan
```

```
FillUnconvergedGaps False
```

```
CalcPropertiesAtPESPoints True
```

```
ScanCoordinate
```

```
nPoints 10
```

```
Distance 1 2 1.4 1.7
```

```
End
```

```
ScanCoordinate
```

```
nPoints 10
```

```
Distance 7 1 1.0 1.2
```

```
Dihedral 7 1 2 3 60.0 180.0
```

```
End
```

```
End
```

```
Engine DFTB
```

```
Model SCC-DFTB
```

```
ResourcesDir QUASINANO2015
```

```
EndEngine
```

```
EOF
```

```
echo "=====
```

```
echo "Ethane gap filling test (2/2)"
```

```
echo "=====
```

```
echo
```

```
AMS_JOBNAME=ethane_fillgaps $AMSBIN/ams << EOF
```

```
Task PESScan
```

```
System
```

```
Atoms
```

```

C -2.333834610464788 -2.268837915270455 -0.2417723425321957
```

(continues on next page)

(continued from previous page)

```

      C  -0.8081611038872945  -2.334371994724881  -0.04271045326758349
      H  -0.2505615773096904  -1.473443563856088  -0.38077110593546
      H  -0.3249814761083244  -3.235478579439597  -0.3904810245975267
      H  -0.583247370537557   -2.349691649662279   1.013499336841977
      H  -2.817014238243758   -1.367731330555738   0.1059982287977475
      H  -2.891434137042391   -3.129766346139247   0.09628831013568076
      H  -2.558748343814525   -2.253518260333056  -1.297982132641757
    End
  End

  GeometryOptimization
    CoordinateType Cartesian
    Convergence Step=1.0e-3
  End

  PESScan
    FillUnconvergedGaps True
    CalcPropertiesAtPESPoints True
    ScanCoordinate
      nPoints 10
      Distance 1 2 1.4 1.7
    End
    ScanCoordinate
      nPoints 10
      Distance 7 1 1.0 1.2
      Dihedral 7 1 2 3 60.0 180.0
    End
  End
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir QUASINANO2015
EndEngine

EOF

```

11.5.3 Example: PES scan for lattice degrees of freedom

Download `PESScan_lattice.run`

```

#!/bin/sh

AMS_JOBNAME=aramide_twist_stretch "$AMSBIN/ams" << eor

Task PESScan

PESScan
  ScanCoordinate
    nPoints 11
    Dihedral 28 2 19 25 0 125
    Dihedral 28 2 19 17 180.0 305
  End
  ScanCoordinate
    nPoints 5

```

(continues on next page)

(continued from previous page)

```

        FromLattice
          13.05 0.0 0.0
        End
        ToLattice
          13.25 0.0 0.0
        End
      End
    End
  End

System
  Atoms
    C 0.9216531199085832 -0.5454515169645626 -0.2340697410375409
    N 7.445957794299515 -0.01681964756070911 0.06597315465223283
    H 3.355037052904425 -1.688703702390754 -0.8816252752735284
    H 9.500951967451224 -1.431847829691195 -0.7477789274312976
    H 5.95132493510865 1.984563817105728 1.200973659205526
    H 12.85473854569853 1.727718149230879 1.067133628354558
    H 3.505268840981867 2.045685273157995 1.235853514670207
    H 10.46827545285599 2.196297117037482 1.332327137819838
    H 5.801094666247189 -1.749824820538445 -0.9165045876907203
    H 11.88742341712797 -1.900439221973683 -1.012979952512364
    O 1.177299854143639 -1.578051954645684 -0.8209290818960512
    H 7.76304211724438 -0.8491313767219947 -0.4061010828807728
    H 1.543350207091979 1.145009665092428 0.7254639267720714
    O 8.129085729296452 1.87393928672803 1.140288046892258
    C 5.295013835525856 -0.9149565254466551 -0.4430653247614506
    C 11.56877839022646 -1.000603430504764 -0.4978381858642764
    C 3.910823555244332 -0.8828087052616261 -0.4247213344032906
    C 10.22040985505902 -0.7280213735372861 -0.3433304227374022
    C 6.047229053946179 0.1077887625542279 0.1368149617061501
    C 12.54387637027615 -0.133469498676877 5.429319732480154e-05
    C 3.259131208138235 0.1880726458368187 0.1825325811252197
    C 9.811866044761443 0.4293238039083616 0.3192924938584955
    C 5.395538021020841 1.178670009771766 0.7440678521670584
    C 12.13529814513992 1.023878258864663 0.662679127884124
    C 4.011348680350846 1.210817677954059 0.762411994086449
    C 10.78693136481616 1.296462367210578 0.8171881914852902
    N 1.860406855610308 0.3126927007099983 0.2533844179661697
    C 8.384662974014494 0.8413381653458467 0.553433102061532
  End
  Lattice
    13.04933362476033 0.0 0.0
  End
End

Engine DFTB
EndEngine

eor

AMS_JOBNAME=graphene_shear "$AMSBIN/ams" << eor

Task PESScan

PESScan
  ScanCoordinate

```

(continues on next page)

(continued from previous page)

```

      nPoints 21
      FromLattice
        4.92000000    0.00000000    0.00000000
        0.00000000    8.52169000    0.00000000
      End
      ToLattice
        4.92000000    0.00000000    0.00000000
        4.92000000    8.52169000    0.00000000
      End
    End
  End
End

GeometryOptimization
  Convergence
    Energy 1.0e-6
    Gradients 1.0e-4
    Step 1.0e-3
  End
End

System
  Atoms
    C    -1.23000000    -0.71014083    0.00000000
    C    -1.23000000     3.55070417    0.00000000
    C     1.23000000    -0.71014083    0.00000000
    C     1.23000000     3.55070417    0.00000000
    C    -2.46000000    -4.26084500    0.00000000
    C    -2.46000000     0.00000000    0.00000000
    C     0.00000000    -4.26084500    0.00000000
    C     0.00000000     0.00000000    0.00000000
    C    -2.46000000    -2.84056333    0.00000000
    C    -2.46000000     1.42028167    0.00000000
    C     0.00000000    -2.84056333    0.00000000
    C     0.00000000     1.42028167    0.00000000
    C    -1.23000000    -2.13042250    0.00000000
    C    -1.23000000     2.13042250    0.00000000
    C     1.23000000    -2.13042250    0.00000000
    C     1.23000000     2.13042250    0.00000000
  End
  Lattice
    4.92000000    0.00000000    0.00000000
    0.00000000    8.52169000    0.00000000
  End
End

Engine ForceField
EndEngine

eor

AMS_JOBNAME=kevlar_compression "$AMSBIN/ams" << eor

Task PESScan

PESScan
  ScanCoordinate

```

(continues on next page)

(continued from previous page)

```

nPoints 20
FromLattice
  VEC1      4.46344907      0.00000000      0.00000000
  VEC2      0.00000000     15.00000000      0.00000000
  VEC3     -4.15619754      0.00000000      7.88460768
End
ToLattice
  VEC1      4.46344907      0.00000000      0.00000000
  VEC2      0.00000000     10.00000000      0.00000000
  VEC3     -4.15619754      0.00000000      7.88460768
End
End
End

GeometryOptimization
  Convergence
    Energy 1.0e-6
    Gradients 1.0e-4
    Step 1.0e-3
  End
End

System
  Atoms
    H      -2.34574255     10.46947046      6.67609071
    H      -0.21566118      9.39070300      5.31032622
    H       1.86323468      3.63186532      1.36675944
    H       0.91346507      8.66345932      2.72468894
    H      -1.16691150      4.36193098      6.67034725
    O       0.68270622      0.35139274      3.28609533
    O      -1.39626502     12.67182069      7.22653591
    O      -0.38377339      7.08567440      4.63239262
    O       1.69448120      5.93755213      0.69230693
    N      -1.77963457      0.94576156      4.16255748
    N       0.29699209     12.07879149      0.22096612
    N       2.09175597      6.51059740      3.71380310
    N       0.01425402      6.51383083      7.65821075
    C       1.78077337     13.06266669      3.74496897
    C      -0.29869254     -0.03890117      7.68614442
    C       2.15256370     11.62041659      3.86515960
    C       0.07290835      1.40360669      7.80683608
    C      -1.25820404     11.13456872      4.63597235
    C       0.81898860      1.88918756      0.69223960
    C      -1.03413359      9.76886805      4.71156187
    C       1.04382335      3.25459780      0.76864405
    C      -1.84321511      8.85293722      4.03709183
    C       0.23429106      4.17122113      0.09564705
    C       1.56385944      9.33860183      3.26763920
    C      -0.51592344      3.68621895      7.21191051
    C       1.34444793     10.70471794      3.18555782
    C      -0.73592207      2.32019944      7.12929096
    C      -1.95926167      2.34708714      4.02068710
    C       0.11835470     10.67722153      0.08003469
    C       1.48642930      3.00363604      4.70160260
    C       3.56353841     10.02077981      0.76017378
    C       1.37833294      4.38413148      4.62212487
    C       3.45526706      8.64027066      0.68061721

```

(continues on next page)

(continued from previous page)

```

C      2.27121884      5.11078244      3.84153940
C      0.19337160      7.91355330      7.78646448
C     -1.17724825      4.45615536      3.15531864
C     -3.25414982      8.56800675      7.10151848
C     -1.05881000      3.07587289      3.25293559
C     -3.13673047      9.94838835      7.19779875
C     -1.47236454      7.40647135      4.16074741
C      0.60580388      5.61750577      0.22026971
H     -0.81007305      0.64728338      4.20563108
H      1.26623048     12.37812387      0.26423466
H      1.12367626      6.80447595      3.64285507
H     -0.95355823      6.21973701      7.58479288
H     -0.61732754     11.80995952      5.18500113
H      1.46048855      1.21353041      1.24041623
H      0.52432261     11.08730485      2.59181576
H     -1.55764558      1.93835499      6.53737820
H      0.78524131      2.43214154      5.29441455
H      2.86125712     10.59243593      1.35153830
H      0.59209290      4.90748377      5.15056713
H      2.66790223      8.11715076      1.20764332
H     -0.48947479      5.02445608      2.54346024
H     -2.56564408      7.99935355      6.49077972
H     -0.26677134      2.55599700      2.73158401
End
Lattice
  VEC1      4.46344907      0.00000000      0.00000000
  VEC2      0.00000000     13.02426461      0.00000000
  VEC3     -4.15619754      0.00000000      7.88460768
End
End

Engine ForceField
EndEngine

eor

AMS_JOBNAME=diamond_volumescan "$AMSBIN/ams" << eor

Task PESScan

PESScan
  ScanCoordinate
    nPoints 5
    CellVolumeScalingRange 0.8 1.2
  End
End
System
  Atoms
    C 3.12375 3.12375 3.12375
    C 0.44625 0.44625 0.44625
    C 3.12375 1.33875 1.33875
    C 0.44625 2.23125 2.23125
    C 1.33875 3.12375 1.33875
    C 2.23125 0.44625 2.23125
    C 1.33875 1.33875 3.12375
    C 2.23125 2.23125 0.44625

```

(continues on next page)

(continued from previous page)

```

End
Lattice
  3.57 0.0 0.0
  0.0 3.57 0.0
  0.0 0.0 3.57
End
BondOrders
  1 2 1.0 1 1 1
  1 4 1.0 1 0 0
  1 6 1.0 0 1 0
  1 8 1.0 0 0 1
  2 3 1.0 -1 0 0
  2 5 1.0 0 -1 0
  2 7 1.0 0 0 -1
  3 4 1.0 1 0 0
  3 6 1.0
  3 8 1.0
  4 5 1.0
  4 7 1.0
  5 6 1.0 0 1 0
  5 8 1.0
  6 7 1.0
  7 8 1.0 0 0 1
End
End
Engine DFTB
EndEngine
eor

```

11.6 PES Exploration

11.6.1 Example: Basin Hopping for Ar 13 cluster

Download BasinHopping_Ar13.run

```

#!/bin/sh

# This example shows how the Basin Hopping procedure can be used to find the
# global minimum (and several local minima) of an Argon-13 cluster.
# The LennardJones engine is used for computing energy and gradients.

"$AMSBIN/ams" << EOF

Task PESExploration

System
  Atoms
    Ar      3.19796865788947      1.15934987330252      -1.33239534934505
    Ar      0.77363272043336      0.86005279907579      4.44840472306942
    Ar      6.16557782618638      1.42715657126078      1.00292335487038
    Ar      1.31188973509015      -2.14075518617674      -0.96878076867831

```

(continues on next page)

(continued from previous page)

```

      Ar      -2.41087887202771      -1.38658115989663      -0.47288216008194
      Ar      -0.21272211859964      -2.33529094743978       2.53956949757235
      Ar       0.68146295080450       4.04228262113743      -1.08313269600570
      Ar      -0.30271346683735       0.77172238432013      -2.86122291101125
      Ar      -2.82554186881390       0.46651994777473       2.87593685953916
      Ar       3.25381032839757      -0.76862912920592       2.01893019694943
      Ar      -2.88251373018356       2.43351471156095      -0.54320785580135
      Ar       0.18478571590318       0.93399697546301       0.81886912458718
      Ar       2.90785012175754       3.08435653882379       2.01057298433565
End
End

PESExploration
  Job BasinHopping
  RandomSeed 10
  Temperature 50.0
  NumExpeditions 5
  NumExplorers 5
  FiniteDifference 1e-7

  BasinHopping
    Displacement 5.0
    Steps 50
    DisplacementDistribution gaussian
    WriteUnique T
  End

  Optimizer
    ConvergedForce 1e-8
    MaxIterations 1000
  End

  StructureComparison
    DistanceDifference 0.5
    NeighborCutoff 7.6
  End
End

#Engine ForceField
#EndEngine
Engine LennardJones
  Rmin 3.81749342630018000 # sigma=3.401*angs, rmin=2** (1.0/6.0)*3.401*angs
  Eps 0.000369915709507261 # eps=116.81*K
  Cutoff 15.00 # 15*angs
EndEngine
EOF

```

11.6.2 Example: PES Exploration, Process Search for alanine with PLAMS

Download ProcessSearch_alanine.plms

```
from tools.plams_test_utils import validate_energy_landscape

config.log.stdout = 0

alanine = from_smiles("CC(C(=O)O)N")

# Let's run a process search job on Alanine:

engine_sett = Settings()
engine_sett.input.ForceField = Settings()

sett = Settings()
sett.input.ams.Task = "PESExploration"
sett.input.ams.PESExploration.Job = "ProcessSearch"
sett.input.ams.PESExploration.NumExpeditions = 32
sett.input.ams.PESExploration.NumExplorers = 16

job = AMSJob(name="alanine_process_search", molecule=alanine, settings=sett + engine_
↳sett)
results = job.run()
energy_landscape = results.get_energy_landscape()
print(energy_landscape)

# Let's run single point calculations on the found states to verify the
# correctness of the results:

validate_energy_landscape(energy_landscape, engine_sett)
```

11.6.3 Example: PES Exploration, Binding Sites for O on Pt 111

Download BindingSites_O+Pt111.run

```
#!/bin/sh

export OMP_NUM_THREADS=1 # Reduce ReaxFF engine noise

AMS_JOBNAME=process_search "$AMSBIN/ams" << EOF

Task PESExploration

System
  Atoms
    O      5.54003937      1.60484089      8.92740946      region=adsorbate
    Pt     8.31521714      4.80239136      7.78926430      region=surface
    Pt     5.54298786      0.00160612      7.79128800      region=surface
    Pt     6.92910250      2.40199874      7.79027615      region=surface
    Pt    11.08707571      4.80244586      7.78854885      region=surface
    Pt     8.31484643      0.00166062      7.79057254      region=surface
    Pt     9.70096096      2.40178486      7.79068672      region=surface
    Pt     5.54335868      4.80260524      7.78885374      region=surface
    Pt     2.77112940      0.00182001      7.79087743      region=surface
    Pt     4.15724404      2.40221263      7.78986559      region=surface
```

(continues on next page)

(continued from previous page)

```

Pt      9.70154331      5.60130339      5.52585419      region=surface
Pt      6.92931403      0.80051815      5.52787789      region=surface
Pt      8.31542867      3.20091077      5.52686604      region=surface
Pt      4.15782639      5.60173116      5.52503306      region=surface
Pt      1.38559711      0.80094593      5.52705675      region=surface
Pt      2.77171175      3.20133854      5.52604491      region=surface
Pt      6.92968485      5.60151727      5.52544363      region=surface
Pt      4.15745557      0.80073204      5.52746732      region=surface
Pt      5.54357021      3.20112466      5.52645547      region=surface
Pt      6.92989639      4.00003668      3.26303536      region=surface
Pt      8.31601103      6.40042930      3.26202352      region=surface
Pt      5.54378174      1.59964406      3.26404721      region=surface
Pt      9.70175484      3.99982280      3.26344593      region=surface
Pt      11.08786948      6.40021541      3.26243408      region=surface
Pt      8.31564020      1.59943018      3.26445778      region=surface
Pt      4.15803792      4.00025057      3.26262480      region=surface
Pt      5.54415256      6.40064318      3.26161295      region=surface
Pt      2.77192328      1.59985795      3.26363664      region=surface
Pt      1.38639089      2.39871548      1.00094199      region=surface
Pt      5.54436409      4.79916260      0.99921469      region=surface
Pt      6.93024583      7.19969029      1.00043291      region=surface
Pt      4.15824945      2.39876998      1.00022653      region=surface
Pt      8.31622256      4.79894871      0.99962525      region=surface
Pt      9.70210429      7.19947641      1.00084348      region=surface
Pt      6.93010792      2.39855609      1.00063710      region=surface
Pt      2.77250564      4.79937648      0.99880412      region=surface
Pt      4.15838737      7.19990418      1.00002234      region=surface
End
Lattice
      8.31557575      0.00000000      0.00000000
      4.15778787      7.20149984      0.00000000
End
Constraints
  FixedRegion surface
End
PESExploration
  Job ProcessSearch

  RandomSeed 100
  NumExpeditions 50
  NumExplorers 5
  DynamicSeedStates T

  Temperature 100.0

  Optimizer
    ConvergedForce 0.0001
  End

  SaddleSearch
    MaxEnergy 2.0
  End

  StructureComparison

```

(continues on next page)

(continued from previous page)

```

        DistanceDifference 0.1
        NeighborCutoff 10.0
        EnergyDifference 0.01
    End
End

Engine ReaxFF
    ForceField CHONSFPtClNi.ff
    Charges Solver=Direct # Reduce ReaxFF engine noise
EndEngine

EOF

AMS_JOBNAME=binding_sites "$AMSBIN/ams" << EOF

Task PESExploration

System
    Atoms
        O      5.54003937      1.60484089      8.92740946      region=adsorbate
        Pt      8.31521714      4.80239136      7.78926430      region=surface
        Pt      5.54298786      0.00160612      7.79128800      region=surface
        Pt      6.92910250      2.40199874      7.79027615      region=surface
        Pt      11.08707571      4.80244586      7.78854885      region=surface
        Pt      8.31484643      0.00166062      7.79057254      region=surface
        Pt      9.70096096      2.40178486      7.79068672      region=surface
        Pt      5.54335868      4.80260524      7.78885374      region=surface
        Pt      2.77112940      0.00182001      7.79087743      region=surface
        Pt      4.15724404      2.40221263      7.78986559      region=surface
        Pt      9.70154331      5.60130339      5.52585419      region=surface
        Pt      6.92931403      0.80051815      5.52787789      region=surface
        Pt      8.31542867      3.20091077      5.52686604      region=surface
        Pt      4.15782639      5.60173116      5.52503306      region=surface
        Pt      1.38559711      0.80094593      5.52705675      region=surface
        Pt      2.77171175      3.20133854      5.52604491      region=surface
        Pt      6.92968485      5.60151727      5.52544363      region=surface
        Pt      4.15745557      0.80073204      5.52746732      region=surface
        Pt      5.54357021      3.20112466      5.52645547      region=surface
        Pt      6.92989639      4.00003668      3.26303536      region=surface
        Pt      8.31601103      6.40042930      3.26202352      region=surface
        Pt      5.54378174      1.59964406      3.26404721      region=surface
        Pt      9.70175484      3.99982280      3.26344593      region=surface
        Pt      11.08786948      6.40021541      3.26243408      region=surface
        Pt      8.31564020      1.59943018      3.26445778      region=surface
        Pt      4.15803792      4.00025057      3.26262480      region=surface
        Pt      5.54415256      6.40064318      3.26161295      region=surface
        Pt      2.77192328      1.59985795      3.26363664      region=surface
        Pt      1.38639089      2.39871548      1.00094199      region=surface
        Pt      5.54436409      4.79916260      0.99921469      region=surface
        Pt      6.93024583      7.19969029      1.00043291      region=surface
        Pt      4.15824945      2.39876998      1.00022653      region=surface
        Pt      8.31622256      4.79894871      0.99962525      region=surface
        Pt      9.70210429      7.19947641      1.00084348      region=surface
        Pt      6.93010792      2.39855609      1.00063710      region=surface
        Pt      2.77250564      4.79937648      0.99880412      region=surface
        Pt      4.15838737      7.19990418      1.00002234      region=surface

```

(continues on next page)

(continued from previous page)

```

End
Lattice
    8.31557575    0.00000000
    4.15778787    7.20149984
End
End

Constraints
    FixedRegion surface
End

PESExploration
    Job BindingSites
    LoadEnergyLandscape
        Path process_search.results/ams.rkf
    End

    StatesAlignment
        ReferenceRegion surface
    End

    StructureComparison
        DistanceDifference 0.1
        NeighborCutoff 3.5
        EnergyDifference 0.1
    End
End

Engine ReaxFF
    ForceField CHONSFpPtClNi.ff
    Charges Solver=Direct # Reduce ReaxFF engine noise
EndEngine

EOF

```

11.6.4 Example: Importing conformers into PES explorations

Download `import_into_pesexp.run`

```

#!/bin/sh

# Step 1: Run the conformer tool to quickly get a set of minima on the PES
# -----

$AMSBIN/conformers << EOF

Generator
    RDKit
        InitialNConformers 100
    End
End

System
    Atoms

```

(continues on next page)

(continued from previous page)

```

N -1.912873733399416 2.022897940092991 -0.1378294583620503
C -0.871664156578182 3.02960778940094 0.06584379364356192
C 0.5013812000791015 2.352115382193892 0.0410146984615521
O 0.812167345172201 1.847076794564962 -1.240836870209463
H -1.850257440384896 1.675835972092315 -1.088966428216949
H -2.82010628990089 2.459772455658122 -0.02412199180453132
H -1.024866752021569 3.492408399687411 1.045420209976512
H -0.9134720948542179 3.809309747393572 -0.7093004343600406
H 0.5270488401868264 1.547182041972351 0.7934160458635944
H 1.275876827761384 3.088526288396493 0.2854207086154862
H 0.2041922851100986 1.132993293447435 -1.463777763138591
End
BondOrders
  1 2 1.0
  1 5 1.0
  1 6 1.0
  2 3 1.0
  2 7 1.0
  2 8 1.0
  3 4 1.0
  3 9 1.0
  3 10 1.0
  4 11 1.0
End
End
Engine DFTB
  Model DFTB0
  ResourcesDir DFTB.org/mio-1-1
EndEngine

RNGSeed 20200512 # Just to make the test output deterministic ...

EOF

# Step 2: Import energy landscape by running a LandscapeRefinement on it
# -----
#
# The energy landscape that is produced by the conformers tool does not contain
# the engine output files of the individual states, which contain e.g. the
# normal modes. It can be used as input to the PESExploration ->
# LandscapeRefinement job, but not for the other PESExploration jobs (e.g.
# ProcessSearch). You should therefore always run the conformers output through
# a quick LandscapeRefinement before proceeding to other PES exploration jobs.
#
# At this point you can also trim down the number of minima loaded from the set
# of conformers. In the example below, we only load the 10 lowest energy
# conformers.

AMS_JOBNAME=refinement $AMSBIN/ams << EOF

Task PESExploration

PESExploration
  Job LandscapeRefinement

```

(continues on next page)

(continued from previous page)

```

    LoadEnergyLandscape
      Path conformers.results/conformers.rkf
      KeepOnly 1:10
    End
  End
End

Engine DFTB
  Model DFTB0
  ResourcesDir DFTB.org/mio-1-1
EndEngine

RNGSeed 20200512 # Just to make the test output deterministic ...

EOF

# Step 3: Run a process search around the lowest energy conformer
# -----
#
# We load all states we got from the LandscapeRefinement and run a single
# expedition around the lowest energy conformer. This will give us a set of
# transition states connecting it to other conformers, but possibly also other
# isomers.

AMS_JOBNAME=process_search $AMSBIN/ams << EOF

Task PESExploration

PESExploration
  Job ProcessSearch
  LoadEnergyLandscape
    Path refinement.results
    SeedStates 1
  End
  DynamicSeedStates False
  NumExpeditions 1
  NumExplorers 64
End

LoadSystem
  File refinement.results/ams.rkf
End

Engine DFTB
  Model DFTB0
  ResourcesDir DFTB.org/mio-1-1
EndEngine

RNGSeed 20200512 # Just to make the test output deterministic ...

EOF

# Note that DFTB0 wrongly predicts an energetically favorable proton transfer
# from the OH group to the NH2 group. This would not have happened if we had
# used SCC-DFTB, but it is a nice example to show off the input for these jobs
# and that the PES exploration can discover also isomers, while the conformer

```

(continues on next page)

(continued from previous page)

```
# tool should only produce conformers.
```

11.7 Molecular dynamics

11.7.1 Example: Simple MD for H2

Download MD_hydrogen_longrun.run

```
#!/bin/sh

$AMSBIN/ams << eor

Task MolecularDynamics

MolecularDynamics
  nSteps 1000
  TimeStep 0.1
  InitialVelocities Type=zero
  Preserve Momentum=False AngularMomentum=False
  Thermostat Type=none
  Trajectory SamplingFreq=100
End

System
  Atoms [Bohr]
    H -2.0 0.0 0.0
    H 2.0 0.0 0.0
  End
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir Dresden
  Occupation Strategy=Fermi Temperature=5
  Repulsion
    forcePolynomial true
  End
  DispersionCorrection Auto
EndEngine

eor
```

11.7.2 Example: MD for a box of water

Download H2O_nreac.run

11.7.3 Example: Lattice deformations in MD

Download MD_Deformation.run

```
#!/bin/sh

# This example consists of three MD runs exercising the Deformation feature in MD:
#   1. Simple deformations of a 2D slab
#   2. Complex sequence of deformations of a 3D box including multiple simultaneous
↪deformations
#   3. Combination of a uniaxial strain with barostatting the remaining directions

# This run first oscillates the lengths of the lattice vectors of the original non-
↪orthogonal
# lattice of an argon slab to 20 and 12 Angstrom respectively and back to the
↪original values,
# with a period of 125 fs. Afterwards, it linearly morphs the lattice to a 5 by 5
↪Angstrom square.
AMS_JOBNAME=Ar-2D "$AMSBIN/ams" <<EOR
# Set the random number seed to ensure reproducible results for testing. Omit this in
↪normal runs.
RNGSeed -2056449389 -1942239902 -422783941 -563204803 439708673 85320298 1328648683 -
↪688288369

Task MolecularDynamics

MolecularDynamics
  nSteps 1000
  TimeStep 0.25
  CalcPressure True
  InitialVelocities Type=Zero
  Trajectory SamplingFreq=10
  Deformation
    StopStep 500
    Type Cosine
    TargetLength 20 12
    Period 125
  End
  Deformation
    StartStep 500
    Type Linear
    TargetLattice
      5 0 0
      0 5 0
    End
  End
  BinLog
    Step
    Time
    PressureTensor
    Pressure
    Volume
    Density
  End
End

System
  Atoms
```

(continues on next page)

(continued from previous page)

```

        Ar 0.5 0.6 0.7
        Ar 3.0 3.2 3.4
    End
    Lattice
        10.0 2.0 0.0
        1.0 10.0 0.0
    End
End

Engine LennardJones
    Rmin 3.753962043231411
    Eps 3.98e-4
    Cutoff 12.0
EndEngine
EOR

# Print the evolution of the lattice parameters in the "fort.59" format
# of old standalone reaxff for testing purposes.
echo "BEGIN fort59 nobarostat"
"$AMSBIN/amspython" "$AMSHOME/scripting/standalone/reaxff-ams/fort59.py" Ar-2D.results
echo "END fort59"

# This run applies a sequence of deformations to a box of Lennard-Jones argon atoms.
AMS_JOBNAME=Ar-box "$AMSBIN/ams" <<EOR
# Set the random number seed to ensure reproducible results for testing. Omit this in
↳normal runs.
RNGSeed 1450250278 2029462689 827312839 1518230169 1408825437 1769469410 -406759494 -
↳1006638501

Task MolecularDynamics

MolecularDynamics
    nSteps 10000
    InitialVelocities Type=Random Temperature=300
    Thermostat Type=NHC Temperature=300 Tau=100
    Timestep 1
    Trajectory SamplingFreq=100
    CalcPressure True
    # Compress the a and b vectors to 20 Angstrom and stretch the c vector to 40.
↳Angstrom over 2000 steps.
    Deformation
        StartStep 0
        StopStep 2000
        Type Linear
        TargetLength 20 20 40
    End
    # Oscillate the c vector (z axis) from 40 to 30 Angstroms and back again with a
↳period of 1000 fs.
    Deformation
        StartStep 2000
        StopStep 3500
        Type Cosine
        TargetLattice
            20.0 0.0 0.0
            0.0 20.0 0.0
            0.0 0.0 30.0

```

(continues on next page)

(continued from previous page)

```

    End
    Period 1000
End
# Shear the box by tilting the c vector 10 A away from the z axis in the xz plane.
Deformation
    StartStep 3500
    StopStep 4000
    Type Linear
    TargetLattice
        20.0  0.0  0.0
        0.0  20.0  0.0
        10.0  0.0  30.0
    End
End
# The following two blocks apply simultaneous cosine and sine oscillations to the
# x and y components of the c vector. The amplitude of both oscillations is the
↪ same,
# indirectly defined by the maximum velocity (velocity at the inflection point).
# The c vector thus traces out a cone around the z axis over 2000 fs.
Deformation
    StartStep 4000
    StopStep 8000
    Type Cosine
    LatticeVelocity
        0.0  0.0  0.0
        0.0  0.0  0.0
        -3e-2 0.0  0.0
    End
    Period 2000
End
Deformation
    StartStep 4000
    StopStep 8000
    Type Sine
    LatticeVelocity
        0.0  0.0  0.0
        0.0  0.0  0.0
        0.0 3e-2  0.0
    End
    Period 2000
End
# This un-shears the box again, returning the c vector to the z axis.
Deformation
    StartStep 8000
    StopStep 8500
    Type Linear
    TargetLattice
        20.0  0.0  0.0
        0.0  20.0  0.0
        0.0  0.0  30.0
    End
End
# Finally, apply an exponential ("true") strain by compressing the c vector by 1/
↪ 1000th every step.
Deformation
    StartStep 8500
    Type Exponential

```

(continues on next page)

(continued from previous page)

```

        LengthRate 0.0 0.0 -1e-3
    End
End

System
    Atoms
        Ar 22.97333 5.533091 19.78068
        Ar 5.255777 10.75370 3.164254
        Ar 8.594419 0.957110 14.00514
        Ar 8.407438 7.306515 2.762099
        Ar 5.335001 18.10475 11.25000
        Ar 13.84870 24.00523 1.485310
        Ar 18.65495 8.812906 24.08580
        Ar 23.50126 5.372569 24.77962
        Ar 20.32336 23.84153 3.669808
        Ar 7.692337 5.389691 10.57909
        Ar 2.407753 7.779618 0.755645
        Ar 1.520220 15.74803 13.75000
        Ar 2.921397 15.34413 20.41981
        Ar 1.660913 18.10666 3.895263
        Ar 19.54458 18.76246 20.66279
        Ar 3.853819 4.537228 14.15265
        Ar 17.57174 15.47554 17.14501
        Ar 8.601399 2.836638 1.872337
        Ar 2.786004 3.601185 22.17957
        Ar 0.047572 18.11282 24.78959
        Ar 0.172965 0.035326 0.223982
        Ar 6.604983 7.254174 21.83257
        Ar 16.63254 15.02432 7.992034
        Ar 7.941898 9.130224 17.89523
        Ar 18.26063 20.70600 5.643385
        Ar 22.63932 15.20789 23.33429
        Ar 1.138343 21.37810 13.97294
        Ar 0.461841 18.99572 6.401096
        Ar 22.67542 22.89935 9.344785
        Ar 7.215648 18.28717 3.411308
    End
    Lattice
        25.0 0.0 0.0
        0.0 25.0 0.0
        0.0 0.0 25.0
    end
end

Engine LennardJones
    Eps 3.67e-4
    RMin 3.4
    Cutoff 12.0
EndEngine
EOR

# Print the evolution of the lattice parameters in the "fort.59" format
# of old standalone reaxff for testing purposes.
echo "BEGIN fort59 nobarostat"
"$AMSBIN/amspython" "$AMSHOME/scripting/standalone/reaxff-ams/fort59.py" Ar-box.
↪results | cut -c 1-86
echo "END fort59"

```

(continues on next page)

(continued from previous page)

```

# This run simulates an experiment to measure the mechanical properties of an argon.
↳box
# by stretching the lattice in the z direction by 100 ppm per timestep (1 fs) while.
↳simultaneously
# keeping the x and y axes relaxed using a barostat.
AMS_JOBNAME=Ar-box-barostat "$AMSBIN/ams" <<EOR
# Set the random number seed to ensure reproducible results for testing. Omit this in.
↳normal runs.
RNGSeed 877404503 1621112889 -559440914 -1857103174 -1082101960 -1842609385 -
↳845245172 -279655974

Task MolecularDynamics

MolecularDynamics
  nSteps 8000
  InitialVelocities Type=Random Temperature=300
  Thermostat Type=NHC Temperature=300 Tau=100
  Barostat Type=MTK Pressure=1e7 Tau=1000 Scale=XY Equal=XY
  Timestep 1
  Trajectory SamplingFreq=100
  Deformation
    Type Exponential
    StrainRate
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 1e-4
  End
End
End

System
  Atoms
    Ar 22.97333 5.533091 19.78068
    Ar 5.255777 10.75370 3.164254
    Ar 8.594419 0.957110 14.00514
    Ar 8.407438 7.306515 2.762099
    Ar 5.335001 18.10475 11.25000
    Ar 13.84870 24.00523 1.485310
    Ar 18.65495 8.812906 24.08580
    Ar 23.50126 5.372569 24.77962
    Ar 20.32336 23.84153 3.669808
    Ar 7.692337 5.389691 10.57909
    Ar 2.407753 7.779618 0.755645
    Ar 1.520220 15.74803 13.75000
    Ar 2.921397 15.34413 20.41981
    Ar 1.660913 18.10666 3.895263
    Ar 19.54458 18.76246 20.66279
    Ar 3.853819 4.537228 14.15265
    Ar 17.57174 15.47554 17.14501
    Ar 8.601399 2.836638 1.872337
    Ar 2.786004 3.601185 22.17957
    Ar 0.047572 18.11282 24.78959
    Ar 0.172965 0.035326 0.223982
    Ar 6.604983 7.254174 21.83257
    Ar 16.63254 15.02432 7.992034

```

(continues on next page)

(continued from previous page)

```

      Ar  7.941898  9.130224  17.89523
      Ar 18.26063  20.70600  5.643385
      Ar 22.63932  15.20789  23.33429
      Ar  1.138343  21.37810  13.97294
      Ar  0.461841  18.99572  6.401096
      Ar 22.67542  22.89935  9.344785
      Ar  7.215648  18.28717  3.411308
End
Lattice
      25.0  0.0  0.0
      0.0  25.0  0.0
      0.0   0.0 25.0
end
end

Engine LennardJones
  Eps 3.67e-4
  RMin 3.4
  Cutoff 12.0
EndEngine
EOR

# Print the evolution of the lattice parameters in the "fort.59" format
# of old standalone reaxff for testing purposes.
echo "BEGIN fort59 barostat"
"$AMSBIN/amspython" "$AMSHOME/scripting/standalone/reaxff-ams/fort59.py" Ar-box-
↪barostat.results | cut -c 1-86
echo "END fort59"

```

11.7.4 Example: PLUMED spherical constraint on Ion-Water cluster

Download IonWaterClusterPlumed.run

```

#!/bin/sh

#####
# A molecular dynamics simulation of a sodium ion in a small water cluster.
# The oxygen atoms of the water molecules are constrained to be within 0.45 nm of the
↪central ion.
#
# The constraint is a harmonic function  $U(r)=\kappa*(r-0.45)**2$ .
# The constrained value  $r$  is the maximum distance out of the set of Na-O distances.
# It is a continuous function that approximates the maximum out of a set of values.
# The parameter beta determines the gradient of the function.
# Choosing a small beta value results in a better match between the actual
# and approximate maximum values, but also in a less continuous function.
#####

$AMSBIN/ams << eor

Task MolecularDynamics

MolecularDynamics
  NSteps 1000
  TimeStep 0.5

```

(continues on next page)

(continued from previous page)

```

InitialVelocities
  Temperature 300
End
Thermostat
  Type Berendsen
  Temperature 300
  Tau 100
End
Trajectory
  SamplingFreq 1
End
Plumed
  Input
GROUP ATOMS=2,5,8,11,14,17,20,23,26 LABEL=oxygen
DISTANCES GROUPA=1 GROUPB=oxygen MAX={BETA=0.05} LABEL=wallcv
UPPER_WALLS ARG=wallcv.max AT=0.45 KAPPA=5000.0
PRINT ARG=wallcv.max STRIDE=1 FILE=COLVAR
  End
End
End

system
  Atoms
      Na      0.0000000000      0.0000000000      0.0000000000
      O      1.7119252402     -2.0080034855      1.0155335456
      H      2.5731328659     -2.4678020432      1.2672765294
      H      1.4480244399     -2.5303870208      0.1729165635
      O      0.6300542373      0.5761441595      2.8226448718
      H      1.2391579321     -0.0089304214      2.3349735783
      H      0.9338206205      1.4729694268      2.4878846070
      O      0.1647045543      2.0208702482     -2.3249601218
      H     -0.7307215954      2.0625495942     -2.7797813945
      H      0.2097075116      1.0635963204     -2.1125352562
      O     -1.7025066874     -1.0780652110     -3.0700032660
      H     -2.3616744611     -0.4381207399     -3.4654803274
      H     -1.8354437418     -1.8757732529     -3.6203721171
      O      3.0658874271      0.7135749531     -2.1674092521
      H      3.9052706085      1.2362759268     -2.1017219618
      H      2.5231505825      1.3717470880     -2.6845066156
      O     -2.8805081861      2.6302885978     -0.3532568156
      H     -3.8762635503      2.6712519969     -0.4669367452
      H     -2.6280204633      2.2152595735     -1.2369080650
      O     -3.5076002124     -1.7860735453     -0.4319799526
      H     -3.8213733752     -1.9705946207     -1.3583130388
      H     -2.8954116001     -1.0294541930     -0.5563055520
      O      2.9868579456      2.5941122968      0.9208652453
      H      2.2983746780      3.1353478114      0.4089961955
      H      3.1905202107      1.8321429488      0.2760998864
      O     -2.9007338617      1.4515370844      2.5612078673
      H     -2.1648641984      1.4189029227      1.8879229015
      H     -3.6017400767      1.9414536506      2.1086226997
  End
  BondOrders
    20 21 1.0
    20 22 1.0
    26 27 1.0
    26 28 1.0

```

(continues on next page)

(continued from previous page)

```

23 24 1.0
23 25 1.0
2 3 1.0
2 4 1.0
11 12 1.0
11 13 1.0
17 18 1.0
17 19 1.0
14 15 1.0
14 16 1.0
5 6 1.0
5 7 1.0
8 9 1.0
8 10 1.0
End
Charge 1.0
End

Engine ForceField
  Type UFF
EndEngine

RNGSeed 1

eor

#####
# The constrained values are stored in the file named COLVAR.
# Below we use Python to read the COLVAR file, and print the average, maximum, and
# ↪ minimum value.
#####

$AMSBIN/amspython << eor
infile = open('COLVAR')
lines = infile.readlines()
infile.close()

values = [float(line.split()[1]) for line in lines[1:]]
print()
print('Average value constraint: ',sum(values)/len(values))
print('Minimum value constraint: ',min(values))
print('Maximum value constraint: ',max(values))
eor

```

11.7.5 Example: MM/MM with constrained ion-water cluster in periodic water box

Download IonWaterHybridSolvationPlumed.run

```

#!/bin/sh

#####
# A molecular dynamics simulation of MgCl2 dissolved in a small periodic water box.
# The setup mimics a multi-scale (QM/MM) simulation,
# but with both sides described by the UFF force field.
#

```

(continues on next page)

(continued from previous page)

```

# In an actual QM/MM simulation it is crucial that
# the QM and MM water molecules do not diffuse into one another.
# We accomplish this by constraining the water molecules according to the FIRES_
↳procedure
# (doi: 10.1021/ct300091w).
# The constraint f({r_O}) is a function of the set of distances of
# all oxygen atoms to the central ion.
# {r_O}={r_QM}+{r_MM}
# f({r_O}) = min({r_MM}) - max({r_QM}).
# This function is constrained to be greater than zero using a half-harmonic_
↳potential.
# U(f)=kappa*(f)**2 for f<0
# U(f)=0 for f>=0
#
# The min() and max() functions are continuous functions that
# approximate respectively the minimum and the maximum out of a set of values.
# The parameter beta determines the continuity of the function.
# Choosing a beta value further from 1
# (larger and smaller for the min() and max() functions respectively)
# results in a better match between the actual
# and approximate minimum/maximum values, but also in a less continuous function.
#####

$AMSBIN/ams << eor

MolecularDynamics
  InitialVelocities
    Temperature 300.0
  End
  NSteps 200
  Plumed
    Input
GROUP  ATOMS=27,36,63,93,102,138,198,213,276 LABEL=qm
GROUP  ATOMS=3,6,9,12,15,18,21,24,30,33,39,42,45,48,51,54,57,60,66,69,72,75,78,81,84,
↳87,90,96,99,105,108,111,114,117,120,123,126,129,132,135,141,144,147,150,153,156,159,
↳162,165,168,171,174,177,180,183,186,189,192,195,201,204,207,210,216,219,222,225,228,
↳231,234,237,240,243,246,249,252,255,258,261,264,267,270,273,279,282,285,288,291,294,
↳297,300,303,306 LABEL=mm
DISTANCES GROUPA=1 GROUPB=qm MAX={BETA=0.005} LABEL=maxcv
DISTANCES GROUPA=1 GROUPB=mm MIN={BETA=100.0} LABEL=mincv
COMBINE ARG=mincv.min,maxcv.max COEFFICIENTS=1,-1 PERIODIC=NO LABEL=cv
LOWER_WALLS ARG=cv AT=0. KAPPA=5000.0
PRINT ARG=maxcv.max,mincv.min,cv STRIDE=1 FILE=COLVAR
    End
  End
  Thermostat
    Tau 100.0
    Temperature 300.0
    Type Berendsen
  End
  TimeStep 0.5
  Trajectory
    SamplingFreq 1
  End
End

Task MolecularDynamics

```

(continues on next page)

(continued from previous page)

system
Atoms

Mg	5.7200350720	-2.9328800664	0.7897499036	region=qm
H	1.8099054637	-4.4220924299	4.8703260963	region=mm
O	7.1694504052	-8.8515558080	-4.2753237457	region=mm
H	-7.1267761804	-8.3665468365	-4.8452690067	region=mm
H	6.7102946636	-8.0997193116	-3.7685132509	region=mm
O	2.7088914434	1.0610825031	4.6923243683	region=mm
H	3.3054833375	0.4921267903	5.2402434951	region=mm
H	1.8233363292	0.6530077196	4.7305392172	region=mm
O	-1.4764148218	-6.878850509	-2.6003412213	region=mm
H	-2.0005170531	-7.6500792651	-2.8620779463	region=mm
H	-2.1885584491	-6.3289735405	-2.1092174532	region=mm
O	3.8478147995	-7.4795251060	-6.6736084141	region=mm
H	4.3989126832	-6.7783144126	-7.0250045541	region=mm
H	2.9555533860	-7.0654023511	-6.8252343194	region=mm
O	-4.7471764043	-8.6077953930	-2.2377385210	region=mm
H	-5.5092008380	-8.5969300931	-2.8802846966	region=mm
H	-5.2576468285	-8.9527935768	-1.4219979311	region=mm
O	-4.7772922849	-4.8474333611	-2.1581380915	region=mm
H	-4.8239240982	-4.7790577136	-3.1075173037	region=mm
H	-5.0338948567	-5.7924530112	-2.0117434690	region=mm
O	2.4515102733	0.8469564304	-3.2719985082	region=mm
H	2.3774951038	-0.1397894447	-3.4984484181	region=mm
H	1.8468114255	0.8917317291	-2.4779777927	region=mm
O	-3.3840665349	-4.0437270402	4.9817876227	region=mm
H	-4.0089551919	-4.0145592434	5.6903827994	region=mm
H	-2.8889478457	-3.1965528132	5.0436088929	region=mm
O	3.3018776783	-3.8565105302	0.1699379171	region=qm
H	2.8411614398	-2.9527284257	0.2152811292	region=qm
H	3.3278682045	-4.1059629877	-0.8090519740	region=qm
O	-0.3029454571	0.3223547569	3.0869293987	region=mm
H	0.2531700096	0.3756415457	2.2625874250	region=mm
H	-0.6800920306	-0.5897179149	2.9611527639	region=mm
O	-5.3484100952	1.5806960236	0.2989946199	region=mm
H	-4.7381235203	2.3283676440	0.2109523156	region=mm
H	-6.1446741550	2.0700615293	0.6143220874	region=mm
O	-5.9927569909	-4.5445068094	0.7252872361	region=qm
H	-6.1165384480	-5.5153290275	0.4900339104	region=qm
H	-6.2159199095	-4.0455250579	-0.1412140829	region=qm
O	-0.9099199619	-1.8881647338	5.2995851295	region=mm
H	-0.2497687678	-2.4792433544	4.9371760594	region=mm
H	-0.3480512141	-1.3260802954	5.9009948331	region=mm
O	3.6215242959	2.1814508127	1.6874751538	region=mm
H	4.5165073769	2.5415115300	1.9221949954	region=mm
H	3.4064165275	1.6332209215	2.4834093017	region=mm
O	5.1833051040	-1.2908175195	6.4042504661	region=mm
H	5.0111822889	-1.5205203085	7.3487966849	region=mm
H	5.1785123947	-0.3057599544	6.5543696690	region=mm
O	0.4603933279	-1.2441515414	-0.2033532097	region=mm
H	0.7445254512	-0.6039397934	-0.9167026803	region=mm
H	-0.5688268735	-1.2243571151	-0.2572139023	region=mm
O	-5.4968287382	-8.9609194175	1.2499552393	region=mm
H	-5.1571901876	-8.8988514057	2.1941505053	region=mm
H	-6.4529456019	5.8448385545	1.3244900622	region=mm
O	-5.0649038431	-1.5925489571	5.4791046507	region=mm

(continues on next page)

(continued from previous page)

H	-5.4190296195	-0.6625809652	5.5607319002	region=mm
H	-4.6322521004	-1.5417498356	4.5302003438	region=mm
O	0.3677337844	-8.5810760971	5.8032571443	region=mm
H	-0.1086349973	5.8640829165	5.1172774803	region=mm
H	0.3027415357	-7.6611857343	5.5038587712	region=mm
O	1.5717316961	-8.1886314470	-4.1347530444	region=mm
H	0.6160556952	-7.9037370683	-4.2849423181	region=mm
H	1.5976518950	-8.8465245437	-3.3851806828	region=mm
O	6.0759336250	-1.6447775457	-1.9957055354	region=qm
H	5.0826000738	-1.6154040577	-2.1716448087	region=qm
H	6.2225213454	-0.9240536977	-1.3190154430	region=qm
O	7.0411997183	-6.3306028762	2.6673264477	region=mm
H	7.2603015022	-6.4603061816	3.6331777655	region=mm
H	6.8554828937	-5.4038722790	2.3804719901	region=mm
O	5.4891620764	1.7626392649	5.4506129348	region=mm
H	5.1390437356	2.4184645147	6.1374986521	region=mm
H	5.0532883869	2.1669956638	4.6417895438	region=mm
O	-4.2394186994	0.8168239717	3.3776299556	region=mm
H	-5.1876861457	0.9406435176	3.6467381795	region=mm
H	-4.3280295844	0.9981858500	2.4035538318	region=mm
O	1.6209849652	-6.0322658389	-1.7020131060	region=mm
H	1.9033980535	-6.6148231926	-0.9312205097	region=mm
H	0.6074913843	-6.2050854204	-1.6495119106	region=mm
O	2.3347829393	-2.1734816277	6.0621168212	region=mm
H	2.3757060915	-2.0614469403	7.0710797559	region=mm
H	3.1815618956	-2.6859727964	5.8159286113	region=mm
O	2.6526456824	-4.5252677481	-6.7120333812	region=mm
H	2.8020328735	-4.0500353548	-5.8761570391	region=mm
H	3.5420462524	-4.3070798418	-7.1561049884	region=mm
O	-5.1041720806	-2.2728098033	-4.3516727544	region=mm
H	-4.3561610138	-2.4109430620	-4.9834336502	region=mm
H	-5.2187295786	-1.2775902805	-4.3031347846	region=mm
O	-6.7183990792	-5.6026203305	5.5769211366	region=mm
H	-6.2276316883	-4.8397127501	6.0169261502	region=mm
H	-6.8644442921	-6.2360671517	6.3313904040	region=mm
O	-0.2997249631	-2.3248611096	-6.0652301268	region=mm
H	0.4480748249	-1.7420695578	-5.7844640285	region=mm
H	-0.2345271923	-3.0579113415	-5.3963109343	region=mm
O	4.4557229323	-4.0321409485	3.4229510135	region=qm
H	4.8228557987	-3.1175556198	3.2353382785	region=qm
H	3.6117875931	-4.0360335014	2.8114358246	region=qm
O	0.0246131293	0.7982683816	6.6958688191	region=mm
H	-0.7298836601	0.7814998411	7.3663632337	region=mm
H	0.8452573517	1.1002106587	7.2018710420	region=mm
O	0.5132549970	-3.4758845845	2.4755554365	region=mm
H	0.7192096520	-4.4568118056	2.3329907178	region=mm
H	-0.4531120524	-3.3351695953	2.3773299504	region=mm
O	3.7669484301	-0.0561634528	-0.5323383617	region=qm
H	3.3380366522	-0.6568105740	0.1454732327	region=qm
H	3.3744123064	0.8333772867	-0.3424611610	region=qm
O	-3.0203045076	4.5213182503	-0.5452112002	region=mm
H	-3.2392487263	4.4060241441	-1.4939704819	region=mm
H	-3.1245969727	5.4811933760	-0.4039600780	region=mm
O	-6.7794338138	0.4373660034	-3.4123297388	region=mm
H	-7.3506733030	1.2324326950	-3.2821835254	region=mm
H	-7.0467839805	-0.1662989127	-2.6296147182	region=mm
O	-7.2132673344	-6.9058350093	-0.9498762639	region=mm

(continues on next page)

(continued from previous page)

H	-7.4423414731	-7.4598580979	-1.7108459608	region=mm
H	-7.2921175047	-7.5034525041	-0.1967566459	region=mm
O	-3.4217194872	-7.0004057995	0.2176912260	region=mm
H	-4.4126912325	-6.9974370339	0.2326646720	region=mm
H	-3.2536051030	-6.8635041363	1.1841799205	region=mm
O	5.5910174783	2.1763450404	-2.4932345279	region=mm
H	4.6852021649	2.5693050423	-2.2779105192	region=mm
H	5.3066397633	1.4214774177	-3.1080658731	region=mm
O	-2.4065506310	-7.2932513323	5.7623954624	region=mm
H	-2.7119429920	-7.7054547466	6.5983438564	region=mm
H	-2.6278433895	-6.2775197492	5.8969552032	region=mm
O	3.6640031160	1.8922352246	-6.2296569654	region=mm
H	2.8532764044	2.2115544890	-5.7709024750	region=mm
H	4.3856544177	2.5355822019	-5.8712770223	region=mm
O	-2.3745450348	-8.5657562688	2.7265414945	region=mm
H	-2.1735583778	5.4660643939	2.7300346193	region=mm
H	-2.9009994050	-8.4965918575	3.5413436006	region=mm
O	5.8804342009	4.5354240094	7.1325312387	region=mm
H	5.0131973513	4.8751670246	7.4157239493	region=mm
H	5.9002289371	4.7710058802	6.1556779744	region=mm
O	2.7885658031	-0.8611341888	-5.9360278211	region=mm
H	2.1091805613	-0.1354528711	-6.0003389022	region=mm
H	3.5887596075	-0.3300691963	-5.6526108069	region=mm
O	-7.3714532064	-2.7223839233	-7.2784923252	region=mm
H	-7.1791818261	-2.2678828420	6.8553882382	region=mm
H	-6.4996545488	-2.5177748518	-6.7900553649	region=mm
O	-7.0331348372	-0.9204735428	0.6063230068	region=qm
H	-6.0921468583	-0.5254758737	0.4836642534	region=qm
H	-6.8874090945	-1.8611486055	0.3995170375	region=qm
O	-0.1957946562	-7.2810928977	0.5399018314	region=mm
H	-1.0417531010	-7.4140228543	0.0205629395	region=mm
H	-0.5066511890	-7.1774442433	1.4825039769	region=mm
O	1.8600464739	4.3473997665	-7.1437429628	region=mm
H	1.3996822124	5.0181731916	7.3159348797	region=mm
H	2.3374212908	4.9612352379	-6.5405671664	region=mm
O	-2.6582165538	3.6810647486	6.5096614907	region=mm
H	-1.9090302118	3.2846493549	5.9694982028	region=mm
H	-2.8320470394	4.5572534342	6.1597032487	region=mm
O	-0.9179119719	-7.7641827452	-6.4041108369	region=mm
H	0.0071023946	-7.5564900267	-6.8187063304	region=mm
H	-1.0509730649	-8.7349187187	-6.5113229941	region=mm
O	1.7179882871	4.6867540529	1.8322922929	region=mm
H	1.5287720026	4.0244442843	2.5787414767	region=mm
H	1.2098413206	5.5102132633	2.1367024110	region=mm
O	0.2315235157	5.3955193196	-1.3941386642	region=mm
H	0.9154934397	5.3968616552	-0.6625282811	region=mm
H	-0.6278735028	5.3026017629	-0.9672366771	region=mm
O	-2.6986887017	0.3541447860	5.9851919711	region=mm
H	-3.1402066242	1.0834994773	5.5167535098	region=mm
H	-2.0336367686	0.0007013596	5.3230337832	region=mm
O	7.2680955491	-7.7923878013	-7.0442418879	region=mm
H	-7.4069606746	-8.7072187093	-7.0814921365	region=mm
H	6.3283830672	-7.7710456593	-7.4966591209	region=mm
O	-6.3515739115	1.6766983710	5.5666144799	region=mm
H	-6.9810189288	1.3458684925	6.2898485162	region=mm
H	-6.7899460804	2.4073839912	5.1839051961	region=mm
O	-2.8567695695	-5.4944242037	-4.6438699205	region=mm

(continues on next page)

(continued from previous page)

H	-2.1548980650	-4.9062145314	-4.9951434742	region=mm
H	-2.5217388050	-6.4735195066	-4.8292651223	region=mm
O	-6.7725880456	-5.3434185943	-4.7758904121	region=mm
H	-6.7069455824	-4.3534970892	-4.8167677162	region=mm
H	-6.7009918221	-5.5232802389	-5.7915052246	region=mm
O	-5.9314369718	-0.0938404833	-6.7661002883	region=mm
H	-5.0003399758	0.1037854112	-6.5155498992	region=mm
H	-6.3730841010	0.7190796050	-6.4056830813	region=mm
O	-1.2158144042	3.0933733381	1.7465428819	region=mm
H	-2.0208343618	3.5624274404	1.2900489033	region=mm
H	-1.6275528675	2.3857289407	2.3167156253	region=mm
O	-1.2518029287	-5.6724338313	3.1188234197	region=mm
H	-0.9850206033	-5.8319014903	4.0612631213	region=mm
H	-2.2290327112	-5.7688269115	3.2007651071	region=mm
O	-4.9043214995	5.6519654141	-5.4508370598	region=mm
H	-4.2942510297	-8.7887223286	-4.9708446990	region=mm
H	-4.4193283168	4.7703067795	-5.4563625441	region=mm
O	-2.5247612340	0.5732653081	0.0472599717	region=mm
H	-3.3924069268	0.1521030415	-0.1584794862	region=mm
H	-2.3892053288	0.2230100537	0.9885150093	region=mm
O	0.3117073158	3.2255479828	4.5437006801	region=mm
H	0.3906039249	2.9422839706	5.4734024052	region=mm
H	0.3838958897	2.3769700045	4.0139213002	region=mm
O	-2.9873643647	-0.4272529780	-4.2245927340	region=mm
H	-3.3997184963	0.2439643116	-3.6677294861	region=mm
H	-2.3084400234	0.1141270972	-4.6000318007	region=mm
O	-3.6653412109	2.2731181202	-6.1284820314	region=mm
H	-2.7056660816	2.1477099050	-6.3724225625	region=mm
H	-4.1288630092	2.2931913038	-7.0006965934	region=mm
O	4.6410116320	-6.7264988994	0.4616167168	region=qm
H	4.4101166082	-6.1645564534	-0.2934777526	region=qm
H	4.6730614520	-6.0060191761	1.1335922366	region=qm
O	4.3976523416	-7.8892228919	-2.5423896153	region=mm
H	5.0311201352	-7.2835094543	-2.0152830256	region=mm
H	3.5722763512	-7.3542989528	-2.6482867334	region=mm
O	7.0792006923	0.1209931002	3.4993386680	region=mm
H	6.8558318086	-0.2426661313	4.4284878799	region=mm
H	7.0270810501	-0.7659916797	2.9870911033	region=mm
O	6.6224221857	5.1402997862	-1.3156095086	region=mm
H	6.4201527371	4.7330581484	-2.1687204168	region=mm
H	6.3243291309	4.4143694297	-0.6709013361	region=mm
O	5.2683337184	-3.5100535099	-5.2215001164	region=mm
H	5.8186543832	-3.6850578037	-5.9995996723	region=mm
H	5.6184071998	-2.7108787147	-4.7924321672	region=mm
O	-7.6586220330	-3.0339696605	3.7287215092	region=qm
H	7.2504138494	-3.8120801294	4.3111784975	region=qm
H	-6.7267376092	-3.0922749899	3.3936627288	region=qm
O	5.0926753309	-4.7195394906	6.3232718697	region=mm
H	5.0070618688	-5.3058931194	5.5083656859	region=mm
H	5.7226193237	-3.9642093025	6.0597369209	region=mm
O	4.9347540809	-7.5589961366	5.0766107941	region=mm
H	5.0190229119	-7.1138138521	4.1998727672	region=mm
H	4.0768338888	-8.0217682098	4.9892390834	region=mm
O	-1.3396252121	-4.8600195445	7.1634896373	region=mm
H	-1.4362228840	-3.8887325443	6.7864428723	region=mm
H	-0.3747219308	-4.9651044260	7.3167935631	region=mm
O	-0.2275547923	1.9526060878	-5.5664997661	region=mm

(continues on next page)

(continued from previous page)

H	-0.1338166809	1.7406093835	-4.5841281712	region=mm
H	-0.2484315595	2.9461522919	-5.5385698031	region=mm
O	-4.6688925072	-6.6804128275	3.4621502759	region=mm
H	-5.5204451119	-6.1841221817	3.2712262969	region=mm
H	-4.5417671440	-6.4984155373	4.3985513305	region=mm
O	-1.7613304313	4.8508364057	-4.1901358774	region=mm
H	-0.9490659842	4.6732275830	-3.6375923368	region=mm
H	-2.1008600242	3.9220171023	-4.3346962477	region=mm
O	1.2813806306	3.6942770054	-3.7171813351	region=mm
H	2.1905963115	3.2609579576	-3.8200540205	region=mm
H	0.9663947506	3.5064245036	-2.7884181141	region=mm
O	-1.6491575074	-4.0249870766	0.4625752262	region=mm
H	-2.3793757109	-4.6115566902	0.1880014661	region=mm
H	-0.8619995170	-4.5482091047	0.1181706395	region=mm
O	3.4345393490	4.4656758215	-0.8654721765	region=mm
H	3.3499869346	4.8246344996	0.0656770670	region=mm
H	3.9194874435	5.1825851342	-1.3432740979	region=mm
O	6.0129279769	-0.1231584466	-5.7786542866	region=mm
H	6.9162672527	-0.4111451674	-5.5767296519	region=mm
H	6.1760997496	0.8392847423	-5.9209507268	region=mm
O	-3.9277443861	3.8877115252	3.4225768700	region=mm
H	-3.8512216972	3.1029288422	4.0808853225	region=mm
H	-4.9243236650	3.8119809069	3.1676946984	region=mm
O	-2.0152094098	-3.5752136096	-2.8775430094	region=mm
H	-2.8511202240	-3.1700568061	-3.2842415136	region=mm
H	-2.4420915432	-4.0420947596	-2.0928745724	region=mm
O	-4.5676885067	-5.7821342384	-7.1960812502	region=mm
H	-4.0359502299	-5.0251773009	-6.7947097709	region=mm
H	-4.3397042788	-6.5601909701	-6.6311674198	region=mm
O	2.3837305449	-7.0019574341	2.8932940349	region=mm
H	1.5843595021	-6.5819302317	3.2063062063	region=mm
H	2.5110710808	-6.4985063424	2.0690005115	region=mm
O	4.3480000521	4.6575725755	-4.8007895809	region=mm
H	3.6918814150	5.3893598759	-4.6122940939	region=mm
H	5.1810645873	5.1658353534	-5.0129492914	region=mm
O	0.9105420244	2.0399129451	-0.2709605351	region=mm
H	0.4502824090	2.6330572019	0.3791216068	region=mm
H	1.7544905017	2.6157672486	-0.3305092813	region=mm
O	6.3707479202	1.9225951795	0.5271756371	region=mm
H	6.1104155087	1.4272322940	-0.2591354170	region=mm
H	6.3108235294	1.1855175540	1.1809221336	region=mm
O	4.9793112969	5.7720777956	2.5371259504	region=mm
H	5.4410948138	-8.6373822889	1.8401865423	region=mm
H	4.0732573198	-8.8547465508	2.6355542548	region=mm
O	-1.5069942574	2.1175389224	-2.6413269953	region=mm
H	-2.2772337306	2.2001260822	-2.0840108235	region=mm
H	-0.8183840708	1.8070130896	-1.9384258703	region=mm
O	-3.1778696685	-2.2850690637	-7.0670100441	region=mm
H	-2.4447728292	-1.6521612371	-6.8766103444	region=mm
H	-3.6202120836	-1.7154370846	7.2497112917	region=mm
O	2.6192086546	-1.2244922235	2.4518195104	region=qm
H	1.8351686207	-1.4137948393	1.8609661449	region=qm
H	2.5964459046	-1.8062454655	3.2887419185	region=qm
O	-7.3321456538	2.9732498807	-5.9518392574	region=mm
H	-6.6099217284	3.5423516492	-5.5423303270	region=mm
H	6.8861073462	3.5653655129	-6.0522063293	region=mm
O	3.5659080357	4.5187085880	5.0152611102	region=mm

(continues on next page)

(continued from previous page)

H	3.5206873771	4.2674900037	4.0721000327	region=mm
H	2.7215092257	4.1357642662	5.4053293417	region=mm
O	0.0671456473	-1.3384369103	-3.1666600421	region=mm
H	-0.8744551748	-1.6446509153	-3.3302330139	region=mm
H	0.5025521594	-2.2006073634	-2.8295670561	region=mm
O	-4.7331769248	-1.9735610859	-1.1710931100	region=mm
H	-4.1695543501	-2.8050647365	-1.1464150457	region=mm
H	-5.3822945516	-2.0965866117	-1.9311157594	region=mm
O	2.7575468577	-3.0372722381	-2.9953973324	region=mm
H	3.3068657454	-3.8611606289	-3.0419899298	region=mm
H	2.3763274374	-2.9947684277	-3.9059578245	region=mm
O	0.5580418210	-4.9464761608	-4.2804053084	region=mm
H	0.0324857624	-4.8267051729	-3.4837664784	region=mm
H	1.0961942044	-5.7302405505	-4.1506399120	region=mm
O	-7.2757328853	4.1642190639	3.0300970932	region=mm
H	6.9121825713	4.5448659465	3.3081894379	region=mm
H	-7.4863485302	3.2969249860	2.5402164796	region=mm
O	-4.1054878322	-1.9430177652	2.0669797525	region=mm
H	-3.6008873069	-2.6082159778	2.6226459577	region=mm
H	-4.6226134371	-2.6261274449	1.5495079114	region=mm
O	5.3816120650	-5.1804053954	-2.7240970290	region=mm
H	5.9660533434	-4.3648474016	-2.5314171173	region=mm
H	5.7143244302	-5.4848887180	-3.6381896932	region=mm
O	1.3959632716	-5.2927847809	5.0843716744	region=mm
H	1.7762736990	-5.4289533682	5.9772747488	region=mm
Cl	-5.4680212411	3.4130263189	-2.7136231556	region=mm
Cl	-5.9988494351	5.4081934954	5.7964537819	region=mm

End

BondOrders

2 306 1.0
 3 4 1.0
 3 5 1.0
 6 7 1.0
 6 8 1.0
 9 10 1.0
 9 11 1.0
 12 13 1.0
 12 14 1.0
 15 16 1.0
 15 17 1.0
 18 19 1.0
 18 20 1.0
 21 22 1.0
 21 23 1.0
 24 25 1.0
 24 26 1.0
 27 28 1.0
 27 29 1.0
 30 31 1.0
 30 32 1.0
 33 34 1.0
 33 35 1.0
 36 37 1.0
 36 38 1.0
 39 40 1.0
 39 41 1.0
 42 43 1.0

(continues on next page)

(continued from previous page)

```
42 44 1.0
45 46 1.0
45 47 1.0
48 49 1.0
48 50 1.0
51 52 1.0
51 53 1.0
54 55 1.0
54 56 1.0
57 58 1.0
57 59 1.0
60 61 1.0
60 62 1.0
63 64 1.0
63 65 1.0
66 67 1.0
66 68 1.0
69 70 1.0
69 71 1.0
72 73 1.0
72 74 1.0
75 76 1.0
75 77 1.0
78 79 1.0
78 80 1.0
81 82 1.0
81 83 1.0
84 85 1.0
84 86 1.0
87 88 1.0
87 89 1.0
90 91 1.0
90 92 1.0
93 94 1.0
93 95 1.0
96 97 1.0
96 98 1.0
99 100 1.0
99 101 1.0
102 103 1.0
102 104 1.0
105 106 1.0
105 107 1.0
108 109 1.0
108 110 1.0
111 112 1.0
111 113 1.0
114 115 1.0
114 116 1.0
117 118 1.0
117 119 1.0
120 121 1.0
120 122 1.0
123 124 1.0
123 125 1.0
126 127 1.0
126 128 1.0
```

(continues on next page)

(continued from previous page)

```
129 130 1.0
129 131 1.0
132 133 1.0
132 134 1.0
135 136 1.0
135 137 1.0
138 139 1.0
138 140 1.0
141 142 1.0
141 143 1.0
144 145 1.0
144 146 1.0
147 148 1.0
147 149 1.0
150 151 1.0
150 152 1.0
153 154 1.0
153 155 1.0
156 157 1.0
156 158 1.0
159 160 1.0
159 161 1.0
162 163 1.0
162 164 1.0
165 166 1.0
165 167 1.0
168 169 1.0
168 170 1.0
171 172 1.0
171 173 1.0
174 175 1.0
174 176 1.0
177 178 1.0
177 179 1.0
180 181 1.0
180 182 1.0
183 184 1.0
183 185 1.0
186 187 1.0
186 188 1.0
189 190 1.0
189 191 1.0
192 193 1.0
192 194 1.0
195 196 1.0
195 197 1.0
198 199 1.0
198 200 1.0
201 202 1.0
201 203 1.0
204 205 1.0
204 206 1.0
207 208 1.0
207 209 1.0
210 211 1.0
210 212 1.0
213 214 1.0
```

(continues on next page)

(continued from previous page)

213	215	1.0
216	217	1.0
216	218	1.0
219	220	1.0
219	221	1.0
222	223	1.0
222	224	1.0
225	226	1.0
225	227	1.0
228	229	1.0
228	230	1.0
231	232	1.0
231	233	1.0
234	235	1.0
234	236	1.0
237	238	1.0
237	239	1.0
240	241	1.0
240	242	1.0
243	244	1.0
243	245	1.0
246	247	1.0
246	248	1.0
249	250	1.0
249	251	1.0
252	253	1.0
252	254	1.0
255	256	1.0
255	257	1.0
258	259	1.0
258	260	1.0
261	262	1.0
261	263	1.0
264	265	1.0
264	266	1.0
267	268	1.0
267	269	1.0
270	271	1.0
270	272	1.0
273	274	1.0
273	275	1.0
276	277	1.0
276	278	1.0
279	280	1.0
279	281	1.0
282	283	1.0
282	284	1.0
285	286	1.0
285	287	1.0
288	289	1.0
288	290	1.0
291	292	1.0
291	293	1.0
294	295	1.0
294	296	1.0
297	298	1.0
297	299	1.0

(continues on next page)

(continued from previous page)

```

300 301 1.0
300 302 1.0
303 304 1.0
303 305 1.0
306 307 1.0
End
Lattice
15.0000000000 0.0000000000 0.0000000000
0.0000000000 15.0000000000 0.0000000000
0.0000000000 0.0000000000 15.0000000000
End
End

Engine Hybrid
Engine ForceField ForceField1
Type UFF
EndEngine

Engine ForceField ForceField2
Type UFF
EndEngine

QMMM
Embedding Mechanical
mmEngineID ForceField2
qmEngineID ForceField1
qmRegion qm
End

GuessAttributesOnce No
EndEngine

RNGSeed 1
eor

#####
# The constrained values are stored in the file named COLVAR.
# Below we use Python to read the COLVAR file, and print the averages for max({r_QM})
↪and min({r_MM}).
#####

$AMSBIN/amspython << eor
infile = open('COLVAR')
lines = infile.readlines()
infile.close()

maxdist = [float(line.split()[1]) for line in lines[1:]]
mindist = [float(line.split()[2]) for line in lines[1:]]
print()
print('Average of max QM O-Eu distance: ',sum(maxdist)/len(maxdist))
print('Average of min MM O-Eu distance: ',sum(mindist)/len(mindist))
eor

```

11.7.6 Example: Measuring the friction coefficient using NEMD

Download MD_friction.run

```
#!/bin/sh

# This example measures the friction coefficient between two diamond slabs.
#
# The top slab (B) is being pressed onto slab A by a -Z force while being pulled in
↳the +X direction
# with a velocity of 200 m/s. The normal force used in this test is very high (5 nN
↳per atom in B,
# corresponding to a pressure around 20 GPa) to make the friction coefficient
↳measurably nonzero.

"$AMSBIN/ams" << eor
# Set the RNG seed to make results reproducible for regression testing, delete this
↳line for real simulations
RNGSeed 1354700531 -1828372117 1981811283 531897225 -1777170443 -138867001 572543905 -
↳1932870903

Task MolecularDynamics
MolecularDynamics
  NSteps 10000
  TimeStep 2
  Trajectory
    SamplingFreq 10
    WriteEngineGradients Yes
    WriteVelocities No
    WriteCharges No
    WriteBonds No
    WriteMolecules No
  End
  InitialVelocities
    Temperature 300
  End
  Thermostat
    Type NHC
    Temperature 300
    Tau 100
    Region free
  End
  ApplyVelocity
    Velocity 0.002 0 0
    Components XY
    Region B
  End
  ApplyForce
    Force 0.0 0.0 -160e-9 [Newton]
    Region B
  End
End
System
  Atoms
    C 1.32194823 1.26633023 1.67762819 region=A,free
    C 1.32194823 1.26633023 9.42762819 region=B,free
    C 1.32194823 3.79899068 1.67762819 region=A,free
    C 1.32194823 3.79899068 9.42762819 region=B,free
```

(continues on next page)

(continued from previous page)

```

C 3.9658447 1.26633023 1.67762819 region=A, free
C 3.9658447 1.26633023 9.42762819 region=B, free
C 3.9658447 3.79899068 1.67762819 region=A, free
C 3.9658447 3.79899068 9.42762819 region=B, free
C 0.0 1.26633023 0.78253111 region=A, free
C 0.0 1.26633023 8.532531110000001 region=B, free
C 0.0 3.79899068 0.78253111 region=A, free
C 0.0 3.79899068 8.532531110000001 region=B, free
C 2.64389647 1.26633023 0.78253111 region=A, free
C 2.64389647 1.26633023 8.532531110000001 region=B, free
C 2.64389647 3.79899068 0.78253111 region=A, free
C 2.64389647 3.79899068 8.532531110000001 region=B, free
C 0.0 0.0 3.48183649 region=A, free
C 0.0 0.0 11.23183649 region=B, free
C 0.0 2.53266045 3.48183649 region=A, free
C 0.0 2.53266045 11.23183649 region=B, free
C 2.64389647 0.0 3.48183649 region=A, free
C 2.64389647 0.0 11.23183649 region=B, free
C 2.64389647 2.53266045 3.48183649 region=A, free
C 2.64389647 2.53266045 11.23183649 region=B, free
C 1.32194823 0.0 6.07181582 region=A, free
C 1.32194823 0.0 13.82181582 region=B, free
C 1.32194823 2.53266045 6.07181582 region=A, free
C 1.32194823 2.53266045 13.82181582 region=B, free
C 3.9658447 0.0 6.07181582 region=A, free
C 3.9658447 0.0 13.82181582 region=B, free
C 3.9658447 2.53266045 6.07181582 region=A, free
C 3.9658447 2.53266045 13.82181582 region=B, free
C 1.32194823 1.26633023 5.28928471 region=A, free
C 1.32194823 1.26633023 13.03928471 region=B, free
C 1.32194823 3.79899068 5.28928471 region=A, free
C 1.32194823 3.79899068 13.03928471 region=B, free
C 3.9658447 1.26633023 5.28928471 region=A, free
C 3.9658447 1.26633023 13.03928471 region=B, free
C 3.9658447 3.79899068 5.28928471 region=A, free
C 3.9658447 3.79899068 13.03928471 region=B, free
C 0.0 1.26633023 4.39418763 region=A, free
C 0.0 1.26633023 12.14418763 region=B, free
C 0.0 3.79899068 4.39418763 region=A, free
C 0.0 3.79899068 12.14418763 region=B, free
C 2.64389647 1.26633023 4.39418763 region=A, free
C 2.64389647 1.26633023 12.14418763 region=B, free
C 2.64389647 3.79899068 4.39418763 region=A, free
C 2.64389647 3.79899068 12.14418763 region=B, free
C 0.0 0.0 0.0 region=A, fixed
C 0.0 0.0 7.75 region=B, free
C 0.0 2.53266045 0.0 region=A, fixed
C 0.0 2.53266045 7.75 region=B, free
C 2.64389647 0.0 0.0 region=A, fixed
C 2.64389647 0.0 7.75 region=B, free
C 2.64389647 2.53266045 0.0 region=A, fixed
C 2.64389647 2.53266045 7.75 region=B, free
C 1.32194823 0.0 2.58997933 region=A, free
C 1.32194823 0.0 10.33997933 region=B, free
C 1.32194823 2.53266045 2.58997933 region=A, free
C 1.32194823 2.53266045 10.33997933 region=B, free
C 3.9658447 0.0 2.58997933 region=A, free

```

(continues on next page)

(continued from previous page)

```
C 3.9658447 0.0 10.33997933 region=B,free
C 3.9658447 2.53266045 2.58997933 region=A,free
C 3.9658447 2.53266045 10.33997933 region=B,free

End
Lattice
  5.28779292 0.0 0.0
  0.0 5.06532092 0.0

End
BondOrders
  1 9 1.0
  1 13 1.0
  2 10 1.0
  2 14 1.0
  3 11 1.0
  3 15 1.0
  4 12 1.0
  4 16 1.0
  5 9 1.0 1 0
  5 13 1.0
  6 10 1.0 1 0
  6 14 1.0
  7 11 1.0 1 0
  7 15 1.0
  8 12 1.0 1 0
  8 16 1.0
  33 25 1.0
  33 27 1.0
  33 41 1.0
  33 45 1.0
  34 26 1.0
  34 28 1.0
  34 42 1.0
  34 46 1.0
  35 25 1.0 0 1
  35 27 1.0
  35 43 1.0
  35 47 1.0
  36 26 1.0 0 1
  36 28 1.0
  36 44 1.0
  36 48 1.0
  37 29 1.0
  37 31 1.0
  37 41 1.0 1 0
  37 45 1.0
  38 30 1.0
  38 32 1.0
  38 42 1.0 1 0
  38 46 1.0
  39 29 1.0 0 1
  39 31 1.0
  39 43 1.0 1 0
  39 47 1.0
  40 30 1.0 0 1
  40 32 1.0
  40 44 1.0 1 0
  40 48 1.0
```

(continues on next page)

(continued from previous page)

```
41 17 1.0
41 19 1.0
42 18 1.0
42 20 1.0
43 17 1.0 0 1
43 19 1.0
44 18 1.0 0 1
44 20 1.0
45 21 1.0
45 23 1.0
46 22 1.0
46 24 1.0
47 21 1.0 0 1
47 23 1.0
48 22 1.0 0 1
48 24 1.0
49 9 1.0
49 11 1.0 0 -1
50 10 1.0
50 12 1.0 0 -1
51 9 1.0
51 11 1.0
52 10 1.0
52 12 1.0
53 13 1.0
53 15 1.0 0 -1
54 14 1.0
54 16 1.0 0 -1
55 13 1.0
55 15 1.0
56 14 1.0
56 16 1.0
57 1 1.0
57 3 1.0 0 -1
57 17 1.0
57 21 1.0
58 2 1.0
58 4 1.0 0 -1
58 18 1.0
58 22 1.0
59 1 1.0
59 3 1.0
59 19 1.0
59 23 1.0
60 2 1.0
60 4 1.0
60 20 1.0
60 24 1.0
61 5 1.0
61 7 1.0 0 -1
61 17 1.0 1 0
61 21 1.0
62 6 1.0
62 8 1.0 0 -1
62 18 1.0 1 0
62 22 1.0
63 5 1.0
```

(continues on next page)

(continued from previous page)

```

        63 7 1.0
        63 19 1.0 1 0
        63 23 1.0
        64 6 1.0
        64 8 1.0
        64 20 1.0 1 0
        64 24 1.0
    End
End
Constraints
    FixedRegion fixed
End

Engine ForceField
EndEngine
eor

"$AMSBIN/amspython" "${TEST_DIRECTORY:-.}/friction_coeff.py" --block 200 ams.results/
echo "END OF TEST"

```

11.7.7 Example: Measuring the viscosity using NEMD

Download MD_shear_viscosity.run

```

#!/bin/sh

# This example performs a NEMD simulation of a water box under a shear acceleration.
↪profile.
# The acceleration profile is cosine-shaped and oriented along the Z axis (so that
↪atoms with Z=0 are
# pulled in one direction, atoms in the middle of the box are pulled the other way).
↪The flow direction
# is aligned with the X axis.
#
# This periodic perturbation induces a similarly cosine-shaped flow velocity profile.
↪in the liquid.
# The steady-state average amplitude of the flow velocity is then used to calculate
↪the viscosity.

"$AMSBIN/ams" << eor
# Set the RNG seed to make results reproducible for regression testing, delete this
↪line for real simulations
RNGSeed 836841470 1678420146 -1718043284 -369569254 -2053894812 -1628712594 -
↪240676301 -356441007

Task MolecularDynamics
MolecularDynamics
    NSteps 24000
    TimeStep 1
    InitialVelocities
        Type Random
        Temperature 298.15
    End
    Thermostat
        Type NHC

```

(continues on next page)

(continued from previous page)

```

        Temperature 298.15
        Tau 1000
    End
    CosineShear
        Enabled Yes
        Acceleration 0.000004522108063030400
    End
End
System
    Atoms
        O 0.8244444545793177 4.544838021389278 7.080687151044827 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 0.3547827848924093 4.719510963847915 7.916536224904227 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 1.355721739229628 5.344893665481732 6.975525637919818 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        O 4.824211318207558 0.8069313216022225 7.759825120988862 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 4.514558713823999 1.280197293066311 8.558923240175481 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 4.033391063195659 0.7201928228115005 7.187127164561153 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        O 2.934572969015879 0.626831742380179 5.865967452739235 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 2.151893964459041 1.137329352693095 6.027000900498897 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 3.263433084806719 1.048186940192297 5.040531567587069 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        O 1.206430947463354 1.856002602957112 9.003428097539595 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 0.7495045796559563 2.685732546074114 9.186271372016726 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 0.5732201719526651 1.426252646404271 8.428676649867203 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        O 4.00858752820043 4.029996794081304 5.583951652401872 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 4.6767739512274 4.347838022636921 6.162932810282044 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 3.285915055183219 4.655218377978755 5.758534605518392 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        O 3.563629518129562 1.371347494469859 12.8855840322333 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 3.213171669104573 0.4819670684498465 12.77900015824002 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 3.044964414139814 1.734726370517266 13.59851543041306 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        O 0.3789251709796683 1.127496658856301 13.2115287736376 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 0.6813483419435356 1.239300575078584 14.13801389632943 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 5.369672396356584 1.083579289595597 13.29812675901156 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        O 5.466225740442832 5.993496838594383 3.826556322251799 ForceField.Type=OW
        ↪ForceField.Charge=-0.834
        H 4.698727888102344 0.3975271277797332 3.345862980059084 ForceField.Type=HW
        ↪ForceField.Charge=0.417
        H 5.678590795097077 0.7684784866437412 4.409847532491263 ForceField.Type=HW

```

(continues on next page)

(continued from previous page)

```

↪ForceField.Charge=0.417
  O 5.308923379504553 4.019057754213824 12.30890772203438 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 4.947834409182614 3.130476283899112 12.54309710913415 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 4.507522419442757 4.539019535791192 12.2645840765018 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 0.5246189538877805 2.11444774753437 5.633847854203101 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 0.4798849442749721 2.981151461335371 6.079301327936341 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 0.1746402018766592 1.514431334250879 6.300570842670576 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 0.7532540279197091 1.724451179192218 15.61527324613631 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 1.102056945497837 1.316928430428979 16.4265184790613 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 0.7972987223637659 2.664620455755369 15.82837324135009 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 0.9018681569836331 3.656882816889424 3.402291999454496 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 0.4026827408969129 4.459281159084593 3.630496065021554 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 0.5526952071471827 3.010307298769798 4.02681327048169 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 4.863403687774274 0.9028609469360367 0.558021938995828 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 3.996919626672331 1.319885981099747 0.7021352725814224 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 4.689645967466681 5.893899093581502 0.6975454946677974 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 0.612542233973695 4.471658272146462 15.20199530885226 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 5.757700386271314 4.628928071051244 14.72292292770029 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 0.4490343750054829 4.454474545873454 16.17738658790725 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 0.6366518329200573 4.752909677710202 9.952302751289087 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 0.9300822912913463 5.41930018832904 10.60529619335514 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 0.09536209451509015 4.185550544321785 10.53226705454537 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 1.973521991893519 4.136851000321193 0.9323878175561867 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 1.958801293505274 4.982561965771234 0.471890391343053 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 1.500116599864132 4.339560036883479 1.75645305944565 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 4.912922003545703 4.048464241294365 1.122473368577996 ForceField.Type=OW_
↪ForceField.Charge=-0.834
  H 5.258380821455368 3.682506383808921 1.973193063451997 ForceField.Type=HW_
↪ForceField.Charge=0.417
  H 3.961611234368922 3.899191033445874 1.229424459523123 ForceField.Type=HW_
↪ForceField.Charge=0.417
  O 3.68652628868048 1.595323244715648 10.09294194440881 ForceField.Type=OW_
↪ForceField.Charge=-0.834

```

(continues on next page)

(continued from previous page)

```

      H 2.716738755606992 1.65284789480996 9.785054765941782 ForceField.Type=HW
↪ForceField.Charge=0.417
      H 3.717820588318913 1.493830202223642 11.06393026159224 ForceField.Type=HW
↪ForceField.Charge=0.417
      O 2.322791158566589 4.525494429167161 13.13712396271713 ForceField.Type=OW
↪ForceField.Charge=-0.834
      H 1.684244052546178 4.292584907055148 12.46669482040438 ForceField.Type=HW
↪ForceField.Charge=0.417
      H 1.713022179274242 4.656194947137037 13.91280050486365 ForceField.Type=HW
↪ForceField.Charge=0.417
      O 1.531573866397418 1.053168936091418 0.3718291838441352 ForceField.Type=OW
↪ForceField.Charge=-0.834
      H 1.522293261003647 2.011561261078512 0.5486474226338094 ForceField.Type=HW
↪ForceField.Charge=0.417
      H 0.5646086101553838 0.8818604575421908 0.3306918436047358 ForceField.Type=HW
↪ForceField.Charge=0.417
      O 3.418411885135323 2.370515151746976 3.6606774188027 ForceField.Type=OW
↪ForceField.Charge=-0.834
      H 2.581438388703135 2.63052383259613 3.242613350177665 ForceField.Type=HW
↪ForceField.Charge=0.417
      H 3.469669459117304 3.023705121589598 4.375706151452582 ForceField.Type=HW
↪ForceField.Charge=0.417
    End
  Lattice
    5.952918007318143 0.0 0.0
    0.0 5.952918007318143 0.0
    0.0 0.0 17.85875402195457
  End
  BondOrders
    1 2 1.0
    1 3 1.0
    4 5 1.0
    4 6 1.0
    7 8 1.0
    7 9 1.0
    10 11 1.0
    10 12 1.0
    13 14 1.0
    13 15 1.0
    16 17 1.0
    16 18 1.0
    19 20 1.0
    19 21 1.0 -1 0 0
    22 23 1.0 0 1 0
    22 24 1.0 0 1 0
    25 26 1.0
    25 27 1.0
    28 29 1.0
    28 30 1.0
    31 32 1.0
    31 33 1.0
    34 35 1.0
    34 36 1.0
    37 38 1.0
    37 39 1.0 0 -1 0
    40 41 1.0 -1 0 0
    40 42 1.0

```

(continues on next page)

(continued from previous page)

```

43 44 1.0
43 45 1.0
46 47 1.0
46 48 1.0
49 50 1.0
49 51 1.0
52 53 1.0
52 54 1.0
55 56 1.0
55 57 1.0
58 59 1.0
58 60 1.0
61 62 1.0
61 63 1.0
End
End

Engine ForceField
  Type Amber95
EndEngine
eor

# The following script extracts the induced velocity amplitude and calculates the_
↪corresponding viscosity
#
# The --test argument is only useful for regression testing
"$AMSBIN/amspython" "${TEST_DIRECTORY:-.}/analyze_cosine_shear.py" --test --from 41 --
↪block 100 ams.results/
echo "END OF TEST"

```

11.8 Trajectory replay

11.8.1 Example: Trajectory replays for PES scans, NEB and MD calculations

Download `Replay.run`

```

#!/bin/sh

# 1. Replay of PES scans
# =====

AMS_JOBNAME=pesscan "$AMSBIN/ams" << eor

Task PESScan

PESScan
  ScanCoordinate
    nPoints 6
    Dihedral 5 1 2 6 0.0 50
    Dihedral 4 1 2 3 0.0 50
  End
  ScanCoordinate

```

(continues on next page)

(continued from previous page)

```

        nPoints 7
        Distance 1 2 1.2 1.5
    End
End

System
    Atoms
        C 0.0 -0.0 0.66687
        C 0.0 -0.0 -0.33313
        H 0.0 0.92974 -0.90538
        H 0.0 0.92974 1.23912
        H 0.0 -0.92974 1.23912
        H 0.0 -0.92974 -0.90538
    End
End

Engine ForceField
EndEngine

eor

AMS_JOBNAME=replay_pesscan "$AMSBIN/ams" << eor

Task Replay
Replay
    File pesscan.results
    StoreAllResultFiles Yes
End

Properties
    Gradients True
End

Engine DFTB
EndEngine

eor

echo "Start extracting data from ams.rkf ..."
echo "Number of frames on file:"
$AMSBIN/amsreport replay_pesscan.results/ams.rkf -r 'History%nEntries'
frame=1
while [ $frame -le 42 ]; do
    echo "Frame: $frame"
    $AMSBIN/amsreport replay_pesscan.results/ams.rkf -r "History%Energy($frame) "
    $AMSBIN/amsreport replay_pesscan.results/ams.rkf -r "History%Gradients($frame) "
    frame=$((expr $frame + 1))
done
echo "Data in PESScan section:"
$AMSBIN/amsreport replay_pesscan.results/ams.rkf -r 'PESScan%PESCoords'
$AMSBIN/amsreport replay_pesscan.results/ams.rkf -r 'PESScan%PES'
$AMSBIN/amsreport replay_pesscan.results/ams.rkf -r 'PESScan%HistoryIndices'
echo "Data extracted."

```

(continues on next page)

(continued from previous page)

```

# 2. Replay of NEB calculations
# =====

AMS_JOBNAME=neb "$AMSBIN/ams" << eor

Task NEB

NEB
  Images 16
End

GeometryOptimization
  Convergence
    Energy 1.0e-4
    Gradients 1.0e-2
  End
End

System
  Atoms
    C -0.04837097189972882 0.002828754545985427 1.099927275387085
    C -0.7231327687579711 -1.264630955508361 1.591341080262239
    C -0.7174118987979032 1.275587203448819 1.585489846719994
    H -1.758221225827696 -1.294861009536393 1.215612210647893
    H -0.7536609466556244 -1.286949469754588 2.689737906089842
    H 0.02700743800283175 0.0001348498227195367 0.004547638456250216
    H -0.7481941882345999 1.302956664081095 2.683766567353197
    H -0.1880537663322776 2.167332127899618 1.228056629315197
    H -1.752232117403696 1.308927696210993 1.209290912745431
    H -0.1979764470101184 -2.160385916691095 1.237731888775011
    Cl 1.710726893919094 4.005547128448699e-05 1.649098041095717
  End
End
System final
  Atoms
    C -0.8173023574499557 -0.01408167289928656 0.8544841004283144
    C -1.086263680920393 -1.14724517989661 1.518039806717775
    C -0.8990221290639472 1.362120508644236 1.43231282384412
    H -1.018446188967536 -2.121649215821082 1.03173453221026
    H -1.404982357822256 -1.129945575549278 2.563696578967125
    H -0.5085933307712909 -0.08344191227560969 -0.1948268157146379
    H -1.210244367762068 1.342173382164807 2.485525533439326
    H 0.07413947140246446 1.873805664554979 1.364278897383801
    H -1.616521025856608 1.9769362830809 0.8657976784252567
    H 1.044986335480545 -0.9543604705303219 1.922869894354733
    Cl 2.292729632733364 -1.053331811482705 2.350686966791855
  End
End

Engine DFTB
EndEngine

eor

```

(continues on next page)

(continued from previous page)

```

AMS_JOBNAME=replay_neb "$AMSBIN/ams" << eor

Task Replay
Replay File=neb.results/ams.rkf

Engine ADF
  NumericalQuality Basic
EndEngine

eor

echo "Start extracting data from ams.rkf ..."
echo "Number of frames on file:"
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'History%nEntries'
frame=1
while [ $frame -le 16 ]; do
  echo "Frame: $frame"
  $AMSBIN/amsreport replay_neb.results/ams.rkf -r "History%Energy($frame)"
  frame=$((expr $frame + 1))
done
echo "Data in NEB section:"
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%nebIterations'
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%historyIndex@dim'
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%historyIndex'
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%energy'
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%highestIndex'
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%LeftBarrier'
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%RightBarrier'
$AMSBIN/amsreport replay_neb.results/ams.rkf -r 'NEB%ReactionEnergy'
echo "Data extracted."

# 3. Replay of generic history (e.g. MD)
# =====

AMS_JOBNAME=md "$AMSBIN/ams" << eor

Task MolecularDynamics

MolecularDynamics
  nSteps 1000
  InitialVelocities Type=Zero
  Trajectory SamplingFreq=10
End

System
  Atoms
    O      5.84400000    0.31970000    0.89540000
    H      1.17000000    0.12750000   -2.13090000
    H     -0.61000000   -1.28280000   -1.84810000
    H     -0.57930000   -0.70680000    1.15800000
    C     -3.12520000    0.09550000    0.94430000
    C     -3.19980000   -1.42460000    1.23360000
    C     -2.05280000    0.23760000   -0.14210000
    C     -1.27100000    1.53130000   -0.13370000
    C     -0.13080000    1.46490000   -1.14670000

```

(continues on next page)

(continued from previous page)

```

C      -2.05030000    -2.08490000     0.42320000
C      -1.08120000    -0.91390000     0.20520000
C       0.72590000     0.20890000    -1.09890000
C      -0.07070000    -1.07000000    -0.90020000
C       1.91880000     0.37840000    -0.17010000
C       2.85070000     1.44940000    -0.76420000
C       4.06090000     1.66070000     0.14280000
C       0.79940000    -2.27110000    -0.66800000
C       2.70610000    -0.90160000    -0.05510000
C       4.70630000     0.33980000     0.46480000
C       2.06470000    -2.18950000    -0.28730000
C       4.02370000    -0.86790000     0.26480000
C       1.49560000     0.82380000     1.22970000
C      -2.72010000     0.03450000    -1.50220000
C      -4.44990000     0.59620000     0.43260000
O      -5.21170000    -0.16370000    -0.11570000
C      -4.82510000     2.04560000     0.60460000
H      -2.82710000     0.64000000     1.83720000
H      -3.05600000    -1.60950000     2.29900000
H      -4.16120000    -1.82340000     0.91220000
H      -1.95080000     2.35390000    -0.41170000
H      -0.90510000     1.74180000     0.87040000
H      -0.54560000     1.56290000    -2.16020000
H       0.51310000     2.34310000    -1.00320000
H      -1.56740000    -2.86790000     1.01220000
H      -2.40770000    -2.48600000    -0.52040000
H       3.18420000     1.11610000    -1.74960000
H       2.30420000     2.38610000    -0.86930000
H       4.79290000     2.28810000    -0.38330000
H       3.77550000     2.17310000     1.05860000
H       0.36740000    -3.25490000    -0.82120000
H       2.63690000    -3.09890000    -0.14660000
H       4.55990000    -1.80060000     0.37040000
H       0.83020000     0.07700000     1.66290000
H       2.37890000     0.93230000     1.85900000
H       0.97580000     1.77970000     1.16520000
H      -3.42010000     0.84890000    -1.68870000
H      -1.95890000     0.02380000    -2.28240000
H      -3.25700000    -0.91410000    -1.50550000
H      -4.48270000     2.61460000    -0.25970000
H      -5.90820000     2.13270000     0.69120000
H      -4.35580000     2.43770000     1.50690000
End
End

Engine ReaxFF
  ForceField CHON-2019.ff
EndEngine

eor

AMS_JOBNAME=replay_md "$AMSBIN/ams" << eor

Task Replay
Replay
  File md.results/

```

(continues on next page)

(continued from previous page)

```

    Frames 1:101:10
End

Properties
    Gradients True
    BondOrders True
End

Engine DFTB
EndEngine

eor

echo "Start extracting data from ams.rkf ..."
echo "Number of frames on file:"
$AMSBIN/amsreport replay_md.results/ams.rkf -r 'History%nEntries'
frame=1
while [ $frame -le 11 ]; do
    echo "Frame: $frame"
    $AMSBIN/amsreport replay_md.results/ams.rkf -r "History%Energy($frame) "
    $AMSBIN/amsreport replay_md.results/ams.rkf -r "History%Bonds.Index($frame) "
    $AMSBIN/amsreport replay_md.results/ams.rkf -r "History%Bonds.Atoms($frame) "
    $AMSBIN/amsreport replay_md.results/ams.rkf -r "History%Bonds.Orders($frame) #8.2f"
    frame=$((expr $frame + 1))
done
echo "Data extracted."

```

11.9 Vibrational analysis

11.9.1 Example: Mode Refinement

Download VATools_dydrogesterone.run

```

#!/bin/sh

# This example shows a mode refinement of the band associated with the
# C=O and C=C stretch modes in the dydrogesterone molecule. This was one
# of the example calculations in the original paper on mode refinement:
#
#   J. Phys. Chem. Lett., 2018, 9 (23), pp 6878-6882

# Step 1: Get DFTB modes at the optimized DFT geometry
# -----

AMS_JOBNAME=FREQ_DFTB $AMSBIN/ams << EOF

System
  Atoms
    # Dydrogesterone geometry already optimized with DFT (BP86/TZP).
    C 0.179402320119871 1.1462568499773749 -1.34045805553897
    C -1.0397129548582973 1.4038864822738149 -0.42742864655449175

```

(continues on next page)

(continued from previous page)

```

C -1.9187723039939633 0.15407663507305838 -0.2887082902723144
C 5.26339021966562 -0.0803229279006518 0.15901574737297627
C 4.387948120190181 -1.2500901532169464 0.1985138671042922
C 3.0305058270434313 -1.2006020082109579 0.06687676611264043
C 2.2733069275281843 0.12162323554336309 -0.08656132560038762
C 3.224495049036027 1.1562588842598234 -0.7427294996476081
C 4.589580248129175 1.2673670000240371 -0.055599205726051454
C -2.593712600512111 -0.1967276203097069 -1.6287069723510241
C -2.011183672049868 -2.0904848406306247 0.6637924766824627
C -3.24333170765593 -1.3040476226855777 1.1949612943873085
C -2.9910039148987044 0.19289628304945772 0.8647395097635705
C 1.8832501895313913 0.6066618903839885 1.3319251022509289
C -1.021215985785471 -0.9883577256376227 0.2595406535945511
C 0.950572917185341 -2.499497885394989 -0.17845795742396
C 2.2618422613458535 -2.429295894019406 0.12998062227671012
C 0.14931427176016024 -1.3398555308999998 -0.6810919840203331
C 1.0424571714734645 -0.11148243146945544 -1.0381151732514977
C -4.261014768367461 0.9709543041281233 0.5302962418813911
C -4.2770282131623745 2.4457823410117943 0.8829249061723259
O -5.23153866013212 0.4348316478674291 0.006835058476534351
O 6.486629619384615 -0.1787206146841789 0.3101830744372818
H 4.496419281348679 1.7413060958817603 0.9361407723186962
H 4.889108126023183 -2.2103967123783246 0.34783394165927906
H -2.5053803890738937 0.6948315694295174 1.7185731459759712
H 2.747947397670055 2.147017249296297 -0.7608107343500633
H 3.3788188981072294 0.8618896159833123 -1.7937886227370627
H 5.278515387629924 1.9033945262275014 -0.6280820487414189
H -0.313881863508176 -1.6818474813482587 -1.6263008186833716
H 2.7970964757389787 -3.3338033858599037 0.4312075853898926
H 0.4355882565839679 -3.460258453470549 -0.09628540747796212
H 2.7788926695047853 0.7904363143118734 1.939504792825026
H 1.3142558486724543 1.5445870277271823 1.281760102768146
H 1.2817550339094792 -0.13839730232383732 1.865618408504196
H 1.519857325493626 -0.37989780969443004 -1.9963570252289853
H -3.35526145791249 2.9466219355670638 0.5538746959825623
H -0.5844412872674496 -0.5966200898063014 1.193600037988863
H 0.8128688006525261 2.043605920072621 -1.3496094336049678
H -0.18858319204633375 1.0503960522723468 -2.3730012879550473
H -1.6112343483079186 2.2461900431605373 -0.8495116990527544
H -0.710640496667605 1.7174589936204205 0.5760379282354502
H -1.5937002785491352 -2.7672883134736264 1.4205298630102559
H -2.2836000112559893 -2.709635208733019 -0.2046201598348845
H -3.3761548431942434 -1.4386051747789588 2.276540650869438
H -4.174084527825135 -1.628156544022731 0.7132317870749941
H -1.860827847389362 -0.4325042173503644 -2.4102132115597854
H -3.2757258446436426 -1.0505046678063983 -1.5382984747892279
H -3.1932088604642455 0.6502553019056425 -1.9892714020438331
H -4.314908973244162 2.55066722711135 1.9790476680048945
H -5.155478941054673 2.9311753898910595 0.44480850527661253
End
End

Task SinglePoint

Properties
NormalModes Yes
End

```

(continues on next page)

(continued from previous page)

```

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-freq-1-2
EndEngine

EOF

# Step 2: Mode refinement of the DFTB C=O and C=C stretch bands at the DFT level
# -----

AMS_JOBNAME=ModeRefinement $AMSBIN/ams << EOF

LoadSystem
  File FREQ_DFTB.results/dftb.rkf
End

Task VibrationalAnalysis

VibrationalAnalysis
  Type ModeRefinement
  NormalModes
    ModeFile FREQ_DFTB.results/dftb.rkf
    ModeSelect
      FreqRange 1500 1800
    End
  End
End

Engine BAND
  # Settings from the paper:
  XC
    GGA BP86
  End
  Basis
    Type DZP
  End
  Relativity Level=None
  # Just to make this test run faster:
  NumericalQuality Basic
  Convergence Criterion=1e-6 # results quite sensitive to this
  SCF Method=DIIS # slightly faster because mix is inherited with restarts, not
↳so for the MultiStepper.
EndEngine

EOF

```

11.9.2 Example: Mode Tracking

Download VATools_cyclohexanone.run

```
#!/bin/sh

# This example demonstrates the usage of the AMS vibrational analysis tools
# on the cyclohexanone molecule.

# 1. Optimization with DFT
# -----

AMS_JOBNAME=DFT $AMSBIN/ams << EOF

  System
    Atoms
      C      0.000000    0.000000    0.000000
      C      0.978606    1.950589   -1.251328
      C      0.978606    1.950589    1.251328
      C      0.812569    2.806732    0.000000
      C     -0.039622    0.802151    1.272926
      C     -0.039622    0.802151   -1.272926
      O      0.062619   -1.251283    0.000000
      H      0.123306    0.119861    2.122758
      H     -1.060012    1.229063    1.372200
      H      0.871524    2.560541    2.167037
      H      2.001065    1.521810    1.263746
      H     -0.196785    3.268312    0.000000
      H      1.549313    3.631809    0.000000
      H      0.871524    2.560541   -2.167037
      H      2.001065    1.521810   -1.263746
      H      0.123306    0.119861   -2.122758
      H     -1.060012    1.229063   -1.372200
    End
  End

  Task GeometryOptimization

  GeometryOptimization
    Convergence Gradients=1.0e-4
  End

  Engine BAND
  EndEngine

EOF

# 2. Obtain DFTB hessian and modes as preconditioner and guess
# -----

AMS_JOBNAME=DFTB $AMSBIN/ams << EOF

  LoadSystem
    File DFT.results/ams.rkf
  End
```

(continues on next page)

(continued from previous page)

```

Task GeometryOptimization

GeometryOptimization
  Convergence Gradients=1.0e-4
End

Properties
  NormalModes Yes
End

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-freq-1-2
EndEngine

EOF

# 3. ModeScanning of DFTB C=O stretch mode with DFT
# -----

AMS_JOBNAME=ModeScanning $AMSBIN/ams << EOF

  LoadSystem
    File DFTB.results/ams.rkf
  End
  LoadEngine DFT.results/band.rkf

Task VibrationalAnalysis

VibrationalAnalysis
  Type ModeScanning
  NormalModes
    ModeFile DFTB.results/dftb.rkf
    ModeSelect
      HighIR 1 # This should select the C=O stretch
    End
  End
End

EOF

# 4. Mode tracking with DFT starting from DFTB C=O stretch mode
# -----

AMS_JOBNAME=ModeTracking $AMSBIN/ams << EOF

  LoadSystem
    File DFT.results/ams.rkf
  End
  LoadEngine DFT.results/band.rkf

Task VibrationalAnalysis

VibrationalAnalysis
  Type ModeTracking

```

(continues on next page)

(continued from previous page)

```

ModeTracking
  HessianGuess File
  HessianPath DFTB.results
End
NormalModes
  ModeInputFormat File
  ModeFile DFTB.results/dftb.rkf
  ModeSelect
    HighIR 1 # This should select the C=O stretch
  End
End
End
EOF

# 5. Mode tracking with DFT starting from a pure C=O stretch
# -----

AMS_JOBNAME=ModeTracking_COSTretch $AMSBIN/ams << EOF

LoadSystem
  File DFT.results/ams.rkf
End
LoadEngine DFT.results/band.rkf

Task VibrationalAnalysis

VibrationalAnalysis
  Type ModeTracking
  NormalModes
    ModeInputFormat Inline
    ModeInline
      0.0 0.0 0.7071 # This is the C attached to the O
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 -0.7071 # This is the O
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
      0.0 0.0 0.0
    End
    ModeSelect
      HighIR 1 # This should select the C=O stretch
    End
  End
ModeTracking
  HessianGuess File

```

(continues on next page)

(continued from previous page)

```

HessianPath DFTB.results
TrackingMethod OverlapPrevious
#      ^-- Probably better than the default.
#      Our initial mode is not particularly close yet ...
→
End
End
EOF

```

11.9.3 Example: Vibronic-Structure Tracking

Download VST_pyrene.run

```

#!/bin/sh

# This example demonstrates the usage of vibronic-structure tracking
# on the pyrene molecule. Excited state calculations are performed within AMS and
# at the DFTB level

# 1. Ground state optimization with DFTB
#-----

AMS_JOBNAME=GroundState $AMSBIN/ams << eor

System
  Atoms
    C      -0.01654981      1.36506537      0.00802668
    C      0.00281590      -0.06808050      0.00051978
    C      1.19835953      2.08859192      0.00127327
    C      -1.25056871      2.05539572      0.02157814
    C      1.23682878      -0.75840671     -0.01375673
    C      -1.21209870     -0.79160461      0.00671225
    C      2.44308841      1.35898462     -0.01256813
    C      1.16450680      3.48665407      0.00772934
    C      -2.47512291      1.29231475      0.02824995
    C      -1.25452039      3.45386389      0.02759561
    C      2.46138768      0.00467142     -0.01984509
    C      1.24076149     -2.15686407     -0.02209661
    C      -2.45682219     -0.06199957      0.02121310
    C      -1.17826305     -2.18965609     -0.00177638
    C      -0.05432386      4.15935390      0.02049748
    C      0.04055488     -2.86234799     -0.01620562
    H      3.36596464      1.93801392     -0.01731234
    H      2.09583666      4.04986513      0.00239299
    H      -3.41330596      1.84611463      0.03902307
    H      -2.20073438      3.99162601      0.03767421
    H      3.39957176     -0.54912830     -0.03053168
    H      2.18696724     -2.69462025     -0.03320657
    H      -3.37969722     -0.64102946      0.02611688
    H      -2.10959908     -2.75286422      0.00274662
    H      -0.06904859      5.24837501      0.02503968
    H      0.05526108     -3.95135857     -0.02284989

  End
End

```

(continues on next page)

(continued from previous page)

```

Task GeometryOptimization

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-freq-1-2
EndEngine

eor

# 2. Get lowest singlet-singlet excitation of pyrene and
# compute the excited state gradient at the optimized GS geometry
#-----

AMS_JOBNAME=ExcitedState $AMSBIN/ams << eor

Task SinglePoint
LoadSystem
  File ./GroundState.results/dftb.rkf
End

Properties
  Gradients yes
End

Engine DFTB
  Model DFTB3
  ResourcesDir DFTB.org/3ob-freq-1-2
  Properties
    Excitations
      TDDFTB
        Calc Singlet
        Lowest 1
      End
      TDDFTBGradients
        Excitation 1
      End
    End
  End
EndEngine

eor

# 3. Vibronic-structure tracking for this excitation
# -----

AMS_JOBNAME=VibronicStructure $AMSBIN/ams << eor

LoadSystem
  File ./GroundState.results/dftb.rkf
End

Task VibrationalAnalysis

VibrationalAnalysis
  Type VibronicStructureTracking
  ExcitationSettings

```

(continues on next page)

(continued from previous page)

```

        ExcitationInputFormat File
        ExcitationFile ./ExcitedState.results/dftb.rkf
        Singlet
        A 1
    End
End
AbsorptionSpectrum
    AbsorptionRange -500.0 4000.0
    LineWidth 100
    FrequencyGridPoints 181
End
End

Engine DFTB
    Model DFTB3
    ResourcesDir DFTB.org/3ob-freq-1-2
EndEngine

eor

# 4. Restart the VST run from the previous one
# We have changed our mind and we want
# the peaks to be a little sharper
# -----

AMS_JOBNAME=VibronicStructureRestart $AMSBIN/ams << eor

LoadSystem
    File ./GroundState.results/dftb.rkf
End

Task VibrationalAnalysis

VibrationalAnalysis
    Type VibronicStructureTracking
    ExcitationSettings
        ExcitationInputFormat File
        ExcitationFile ./ExcitedState.results/dftb.rkf
        Singlet
        A 1
    End
End
VSTRestartFile ./VibronicStructure.results/ams.rkf
AbsorptionSpectrum
    AbsorptionRange -500.0 4000.0
    LineWidth 50
    FrequencyGridPoints 181
End
End

Engine DFTB
    Model DFTB3
    ResourcesDir DFTB.org/3ob-freq-1-2
EndEngine

eor

```

11.10 PES point properties

11.10.1 Example: Phonons for graphene

Download Phonons_Graphene.run

```
#!/bin/sh

AMS_JOBNAME=graphene $AMSBIN/ams << EOF

Task GeometryOptimization

GeometryOptimization
!   CoordinateType Cartesian
    OptimizeLattice True
    Convergence Gradients=1.0e-5
    Method Quasi-Newton
End

Properties
    Phonons True
End

NumericalPhonons
    SuperCell
        2 0
        0 2
    End
End

System
    Atoms
        C      0.000000000    -0.000000000    0.00000
        C      0.000000000    -1.420281662    0.00000
    End

    Lattice
        1.230000000    -2.130422493    0.000000000
        1.230000000     2.130422493    0.000000000
    End
End

Engine DFTB
    ResourcesDir Dresden
    Model DFTB0
    KSpace
        Type Symmetric
        Symmetric KInteg=9
    End
    Technical AnalyticalStressTensor=False # Not yet supported with symmetric k-
↪space grid ...
EndEngine

EOF

echo ""
echo "Begin TOC of result file"
```

(continues on next page)

(continued from previous page)

```
$AMSBIN/dmpkf -n 1 graphene.results/dftb.rkf --toc
echo "End TOC of result file"
```

11.10.2 Example: Phonons with isotopes

Download Phonons_Isotopes.run

```
#!/bin/sh

# =====
# Phonons with default nuclear masses:
# =====

AMS_JOBNAME=defmasses $AMSBIN/ams << EOF

Task SinglePoint

Properties
  Phonons True
End

NumericalPhonons
  StepSize 0.01
  SuperCell
    4
  End
End

System
  Atoms
    C -2.42906152 -0.3445528299 -0.1353492062
    C -1.146891508 -1.134644249 0.1353492061
    H -2.429062041 0.004468895147 -1.185797304
    H -2.429062011 0.5753101439 0.4803683017
    H -1.146891017 -2.054507222 -0.4803683019
    H -1.146890987 -1.483665974 1.185797304
  End

  Lattice
    2.564338467 0.0 0.0
  End
End

Engine DFTB
  ResourcesDir QUASINANO2015
  Model DFTB0
  KSpace
    Type Symmetric
    Symmetric KInteg=9
  End
EndEngine
```

(continues on next page)

(continued from previous page)

```

EOF

echo ""
echo "Begin TOC of result file"
$AMSBIN/dmpkf -n 1 defmasses.results/dftb.rkf --toc
echo "End TOC of result file"

# =====
# Phonons with two deuterium atoms:
# =====

AMS_JOBNAME=usermasses $AMSBIN/ams << EOF

  Task SinglePoint

  Properties
    Phonons true
  End

  NumericalPhonons
    StepSize 0.01
    SuperCell
      4
    End
  End

  System
    Atoms
      C   -2.42906152   -0.3445528299   -0.1353492062
      C   -1.146891508   -1.134644249     0.1353492061
      H   -2.429062041    0.004468895147   -1.185797304
      H   -2.429062011    0.5753101439    0.4803683017
      H   -1.146891017   -2.054507222     -0.4803683019   mass=2.014
      H   -1.146890987   -1.483665974     1.185797304     mass=2.014
    End
    Lattice
      2.564338467 0.0 0.0
    End
  End

  Engine DFTB
    ResourcesDir QUASINANO2015
    Model DFTB0
    KSpace
      Type Symmetric
      Symmetric KInteg=9
    End
  EndEngine

EOF

echo ""
echo "Begin TOC of result file"
$AMSBIN/dmpkf -n 1 usermasses.results/dftb.rkf --toc
echo "End TOC of result file"

```

11.10.3 Example: User-defined Brillouin zone for phonon dispersion

Download Phonons_UserBZPath.run

```
#!/bin/sh

# This example shows how to specify a user-defined path through the
# Brillouin zone (BZ) when computing the phonon dispersion curves.

# Note: when computing the phonons, you should first run a geometry
# optimization (including relaxation of lattice vectors).
# Here we use a pre-optimized structure.

AMS_JOBNAME=BoronNitrade $AMSBIN/ams << eor

Task SinglePoint

System
  Atoms
    B 0.0          0.0          0.0
    N 0.92538708336681 0.92538708336681 0.92538708336681
  End
  Lattice
    0.0          1.8507741667336 1.8507741667336
    1.8507741667336 0.0          1.8507741667336
    1.8507741667336 1.8507741667336 0.0
  End
End

Properties
  Phonons Yes
End

NumericalPhonons
  SuperCell
    2 0 0
    0 2 0
    0 0 2
  End

  # Disable the automatically generated path through the BZ

  AutomaticBZPath No

  BZPath
    # In each 'Path' block you should define the vertices of the path
    # in fractional coordinates (with respect to the reciprocal lattice
    # vectors)

    # First segment: Gamma-X-W-K
    Path
      0.0  0.0  0.0  G
      0.5  0.0  0.5  X
      0.5  0.25 0.75  W
      0.375 0.375 0.75  K
    End

    # Second segment: Gamma-SomeRandomPoint
```

(continues on next page)

(continued from previous page)

```

        Path
            0.0  0.0  0.0 G
            0.1  0.2  0.3 RandomPoint
        End
    End
End

Engine DFTB
    Model SCC-DFTB
    ResourcesDir DFTB.org/matsci-0-3
    KSpace Quality=Basic
EndEngine

eor

echo 'Content of phonon_curves section'
$AMSBIN/dmpkf BoronNitrade.results/dftb.rkf 'phonon_curves'
echo 'End content of phonon_curves section'

```

11.10.4 Example: Elastic tensor

Download ElasticTensor.run

```

#!/bin/sh

# === Diamond ===

AMS_JOBNAME=Diamond $AMSBIN/ams << EOF

    Task GeometryOptimization

    Properties
        ElasticTensor Yes
    End

    System
        Atoms
            C  0.44625  0.44625  2.23125
            C  2.23125  2.23125  2.23125
            C -2.23125 -2.23125 -2.23125
            C -0.44625 -0.44625 -2.23125
            C -0.44625 -2.23125 -0.44625
            C  1.33875 -0.44625 -0.44625
            C -2.23125 -0.44625 -0.44625
            C -0.44625  1.33875 -0.44625
            C -0.44625 -0.44625  1.33875
            C  1.33875  1.33875  1.33875
            C -1.33875 -1.33875 -1.33875
            C  0.44625  0.44625 -1.33875
            C  0.44625 -1.33875  0.44625
            C  2.23125  0.44625  0.44625
            C -1.33875  0.44625  0.44625
            C  0.44625  2.23125  0.44625
        End
    End

```

(continues on next page)

(continued from previous page)

```

    Lattice
      0.0  3.57 3.57
      3.57 0.0  3.57
      3.57 3.57 0.0
    End
  End

  GeometryOptimization
    OptimizeLattice Yes
    Convergence Quality=Good
  End

  Symmetry Tolerance=1e-6

  Engine DFTB
    Model DFTB
    ResourcesDir DFTB.org/mio-1-1
    KSpace
      Type Symmetric
      Symmetric KInteg=3
    End
    Technical AnalyticalStressTensor=False # Not yet supported with symmetric k-
↪space grid.
  EndEngine

EOF

# === Boron-Nitride sheet ===

# 3x3 super-cell, default k-space sampling

AMS_JOBNAME=BN_sheet $AMSBIN/ams << EOF

  Task GeometryOptimization

  Properties
    ElasticTensor Yes
  End

  System
    Atoms
      N  3.76095075  0.723795  0.0
      N  5.01460112  2.89518114 0.0
      B -3.76095112 -2.17138614 0.0
      B -2.50730075  0.0        0.0
      B -1.25365038  2.17138614 0.0
      B -1.25365037 -2.17138614 0.0
      B  0.0         0.0        0.0
      B  1.25365037  2.17138614 0.0
      B  1.25365038 -2.17138614 0.0
      B  2.50730075  0.0        0.0
      B  3.76095112  2.17138614 0.0
      N -2.50730112 -1.44759114 0.0
      N -1.25365075  0.723795  0.0
      N -3.8e-07     2.89518114 0.0
      N -3.7e-07     -1.44759114 0.0

```

(continues on next page)

(continued from previous page)

```

      N  1.25365      0.723795      0.0
      N  2.50730037   2.89518114      0.0
      N  2.50730038  -1.44759114      0.0
    End
  Lattice
    7.52190225 0.0
    3.76095111 6.51415842
  End
End

GeometryOptimization
  OptimizeLattice Yes
  Convergence Quality=Good
End

Engine DFTB
  Model SCC-DFTB
  ResourcesDir DFTB.org/matsci-0-3
EndEngine

EOF

# === Polyoxyethylene ===
# primitive cell with k-space sampling

AMS_JOBNAME=Polyoxyethylene $AMSBIN/ams << EOF

  Task GeometryOptimization

  Properties
    ElasticTensor Yes
  End

  ElasticTensor
    StrainStepSize 0.002
  End

  System
    Atoms
      C  -0.279368361  -0.125344097  -0.026221791
      O   0.840592835  -0.919621431  -0.193214154
      H  -0.279527057   0.337014408   0.997733792
      H  -0.281697417   0.707951120  -0.778297849
    End
    Lattice
      2.240292981
    End
  End

  GeometryOptimization
    OptimizeLattice Yes
    Convergence Quality=Good
  End

  Engine DFTB

```

(continues on next page)

(continued from previous page)

```

Model SCC-DFTB
ResourcesDir DFTB.org/3ob-3-1
KSpace
    Type Symmetric
    Symmetric KInteg=5
End
Technical AnalyticalStressTensor=False # Not yet supported with symmetric k-
↳space grid.
EndEngine

EOF

# Note: the elastic tensor is also printed to standard output.

echo ""
echo "Extract the elastic tensor of Diamond from the rkf file:"
$AMSBIN/amsreport Diamond.results/dftb.rkf -r "AMSResults%ElasticTensor#12.4f##6"

echo ""
echo "Extract the elastic tensor of Boron-Nitride from the rkf file:"
$AMSBIN/amsreport BN_sheet.results/dftb.rkf -r "AMSResults%ElasticTensor#12.4f##3"

echo ""
echo "Extract the elastic tensor of Polyoxyethylene from the rkf file:"
$AMSBIN/amsreport Polyoxyethylene.results/dftb.rkf -r "AMSResults%ElasticTensor#12.4f#
↳#1"

```

11.11 Pipe interface

11.11.1 Example: ASE calculator as a pipe worker

Download Pipe_ASEWorker.run

```

#!/bin/sh
# This example uses a Python pipe worker based on ASE to evaluate the Lennard-Jones_
↳potential.
# First we do a single-point calculation including the gradients and the stress_
↳tensor and then
# a short MD run similar to the Pipe_AMSonAMS example.

export SCM_DISABLE_MPI=1

AMS_JOBNAME=SP "$AMSBIN/ams" << eor
RNGSeed -538839488 972444872 -468448621 535232319 -953628259 777353319 -1036072_
↳387155213

Task SinglePoint

Properties
    Gradients True
    StressTensor True
End

```

(continues on next page)

(continued from previous page)

```

System
  Atoms
    Ar    0.03374714   -0.02579634   -0.03864485
    Ar    -2.47202773  -2.57754100   -0.02505375
    Ar    -2.54465030  -0.00587811   -2.49637860
    Ar    -4.98237672  -2.60930954   -2.49495517
    Ar    -0.03221746  -2.51967081   -2.47883073
    Ar    -2.52675554  -5.12905072   -2.48152312
    Ar    -2.54445364  -2.45619404   -5.02601013
    Ar    -4.95645981  -5.14903666   -4.95752059
  End
  Lattice
    0.00000000    5.00000000    5.00000000
    5.00000000    0.00000000    5.00000000
    5.00000000    5.20000000    0.00000000
  End
End

Engine Pipe
  WorkerCommand "$AMSBIN/amspython" "$TEST_DIRECTORY/ASE-LJ-worker.py"
EndEngine

eor

AMS_JOBNAME=MD "$AMSBIN/ams" << eor
RNGSeed -538839488 972444872 -468448621 535232319 -953628259 777353319 -1036072_
↪387155213

Task MolecularDynamics

MolecularDynamics
  nSteps 200
  TimeStep 5.0
  Thermostat Type=NHC Temperature=298.15 Tau=100
  Trajectory SamplingFreq=20
  InitialVelocities Type=random Temperature=200
End

System
  Atoms
    Ar 0.0 0.0 0.0
    Ar 1.605 0.9266471820493496 2.605
  End
  Lattice
    3.21 0.0 0.0
    1.605 2.779941546148048 0.0
    0.0 0.0 5.21
  End
  SuperCell 4 4 4
End

Engine Pipe
  WorkerCommand "$AMSBIN/amspython" "$TEST_DIRECTORY/ASE-LJ-worker.py"
EndEngine

eor

```

Download ASE-LJ-worker.py

```
from scm.amspipe import ASEPipeWorker
from ase.calculators.lj import LennardJones

calculator = LennardJones()
calculator.parameters.epsilon = 0.0102996
calculator.parameters.sigma = 3.4
calculator.parameters.rc = 12.0

engine = ASEPipeWorker(calculator=calculator)
engine.run()
```

11.11.2 Example: AMS as a pipe worker

Download Pipe_AMSonAMS.run

```
#!/bin/sh
# This example runs two separate AMS processes communicating over AMSPipe. One of
# them is the PipeWorker
# calculating Lennard-Jones energies and the other as the PipeMaster drives the MD
# simulation.

export SCM_DISABLE_MPI=1
export NSCM_AMSEXTERNAL=1

cat - > worker.in << eor
Task Pipe

Engine LennardJones
    Rmin 3.81637
    Eps 3.785e-4
    Cutoff 12.0
EndEngine
eor

"$AMSBIN/ams" << eor
RNGSeed -538839488 972444872 -468448621 535232319 -953628259 777353319 -1036072
387155213

Task MolecularDynamics

MolecularDynamics
    nSteps 200
    TimeStep 5.0
    Thermostat Type=NHC Temperature=298.15 Tau=100
    Trajectory SamplingFreq=20
    InitialVelocities Type=random Temperature=200
End

System
    Atoms
        Ar 0.0 0.0 0.0
        Ar 1.605 0.9266471820493496 2.605
    End
    Lattice
```

(continues on next page)

(continued from previous page)

```
        3.21 0.0 0.0
        1.605 2.779941546148048 0.0
        0.0 0.0 5.21
    End
    SuperCell 4 4 4
End

Engine Pipe
    WorkerCommand "$AMSBIN/ams" < ../../worker.in > worker.out 2>&1
EndEngine
eor

echo "Worker output:"
cat ams.results/*/worker.out
```


APPENDICES

12.1 Extended XYZ file format

The `.xyz` file format is a simple text based format for molecular geometries. `.xyz` files have the number of atoms in the first line, followed by a comment line, followed by one line per atom, specifying the element as well as the x, y, and z coordinates of this atom.

However, the standard `.xyz` file format does not include lattice vectors. AMS therefore uses an extended `.xyz` file format which is also suitable for periodic systems. In this extended format the lattice vectors are specified at the end of the `.xyz` file via the keys `VEC1`, `VEC2` and `VEC3`. For 1D periodic systems (chains) only `VEC1` is needed. For 2D periodic systems (slabs) only `VEC1` and `VEC2` are needed. An example extended `.xyz` for graphene looks like this:

```
2
C      0.0    0.0    0.0
C      1.23   0.71014 0.0
VEC1   2.46   0.0    0.0
VEC2   1.23   2.13042 0.0
```

Note that the extended `.xyz` format is also understood by the AMS GUI for importing and exporting geometries from/to `.xyz` files.

12.2 Developer options

```
Print
  Timers [None | Normal | Detail | TooMuchDetail]
End
```

Print

Type

Block

Description

This block controls the printing of additional information to stdout.

Timers

Type

Multiple Choice

Default value

None

Options

[None, Normal, Detail, TooMuchDetail]

Description

Printing timing details to see how much time is spend in which part of the code.

```
EngineDebugging
  AlwaysClaimSuccess Yes/No
  CheckInAndOutput Yes/No
  ForceContinousPES Yes/No
  IgnoreGradientsRequest Yes/No
  IgnorePreviousResults Yes/No
  IgnoreStressTensorRequest Yes/No
  NeverQuiet Yes/No
  RandomFailureChance float
  RandomNoiseInEnergy float
  RandomNoiseInGradients float
  RandomStopChance float
End
```

EngineDebugging**Type**

Block

Description

This block contains some options useful for debugging the computational engines.

AlwaysClaimSuccess**Type**

Bool

Default value

No

Description

If an engine fails, pretend that it worked. This can be useful when you know that an SCF might fail.

CheckInAndOutput**Type**

Bool

Default value

No

Description

Enables some additional checks on the input and output of an engine, e.g. for NaN values.

ForceContinousPES**Type**

Bool

Default value

No

Description

If this option is set, the engine will always run in continuous PES mode. For many engines this disables the use of symmetry, as this one always leads to a discontinuous PES around the symmetric points: Basically there is jump in the PES at the point where the symmetry detection

starts classifying the system as symmetric. Normally the continuous PES mode of the engine (often disabling the symmetry) is only used when doing numerical derivatives, but this flag forces the engine to continuously run in this mode.

IgnoreGradientsRequest

Type

Bool

Default value

No

Description

If this option is set, the engine will not do analytical gradients if asked for it, so that gradients will have to be evaluated numerically by AMS.

IgnorePreviousResults

Type

Bool

Default value

No

Description

If this option is set, the engine will not receive information from previous calculations. Typically this information is used to restart the self consistent procedure of the engine.

IgnoreStressTensorRequest

Type

Bool

Default value

No

Description

If this option is set, the engine will not calculate an analytical stress tensor if asked for it, so that the stress tensor will have to be evaluated numerically by AMS.

NeverQuiet

Type

Bool

Default value

No

Description

Makes the engine ignore the request to work quietly.

RandomFailureChance

Type

Float

Default value

0.0

Description

Makes the engine randomly report failures, even though the results are actually fine. Useful for testing error handling on the application level.

RandomNoiseInEnergy

Type

Float

Default value

0.0

Unit

Hartree

Description

Adds a random noise to the energy returned by the engine. The random contribution is drawn from $[-r,r]$ where r is the value of this keyword.

RandomNoiseInGradients**Type**

Float

Default value

0.0

Unit

Hartree/Angstrom

Description

Adds a random noise to the gradients returned by the engine. A random number in the range $[-r,r]$ (where r is the value of this keyword) is drawn and added separately to each component of the gradient.

RandomStopChance**Type**

Float

Default value

0.0

Description

Makes the engine randomly stop. Can be used to simulate crashes.

12.3 Symmetry

12.3.1 Schönfliess symbols and symmetry labels

A survey of all molecular point groups that are recognized by AMS is given below. The table contains the Schönfliess symbols together with the names of the subspecies of the irreducible representations as they are used by AMS to label normal modes.

Table 12.1: Schönfliess symbols and the labels of the irreducible representations.

Point Group	Schönfliess Symbol in AMS	Irreducible representations
C ₁	NOSYM	A
C _{∞v}	C(LIN)	Sigma Pi
D _{∞h}	D(LIN)	Sigma.g Sigma.u Pi.g Pi.u
I	I	A T1 T2 G H

continues on next page

Table 12.1 – continued from previous page

I_h	$I(H)$	A.g A.u T1.g T1.u T2.g T2.u G.g G.u H.g H.u
O	O	A1 A2 E T1 T2
O_h	$O(H)$	A1.g A1.u A2.g A2.u E.g E.u T1.g T1.u T2.g T2.u
T	T	A E T
T_h	$T(H)$	A.g A.u E.g E.u T.g T.u
T_d	$T(D)$	A1 A2 E T1 T2
C_i	$C(I)$	A.g A.u
S_n	$S(N)$	n=4: A B E
		n=6: A.g A.u E.g E.u
		n=8: A B E1 E2 E3
C_n	$C(N)$	n=2: A B
		n=4: A B E
		even n: A B E1 E2 ...
		n=3: A E
		odd n: A E1 E2 ...
C_{nv}	$C(NV)$	n=2: A1 A2 B1 B2
		n=4: A1 A2 B1 B2 E
		even n: A1 A2 B1 B2 E1 E2 ...
		n=3: A1 A2 E
		odd n: A1 A2 E1 E2 ...
C_s	$C(S)$	A' A''
C_{nh}	$C(NH)$	n=2: A.g A.u B.g B.u
		n=4: A.g A.u B.g B.u E.g E.u
		even n: A.g A.u B.g B.u E1.g E1.u E2.g E2.u ...
		n=3: A' A'' E' E''
		odd n: A' A'' E1' E1'' E2' E2'' ...
D_n	$D(N)$	n=2: A B1 B2 B3
		n=4: A1 A2 B1 B2 E
		even n: A1 A2 B1 B2 E1 E2 ...
		n=3: A1 A2 E
		odd n: A1 A2 E1 E2 ...
D_{nd}	$D(ND)$	n=2: A1 A2 B1 B2 E
		even n: A1 A2 B1 B2 E1 E2 ...
		n=3: A1.g A1.u A2.g A2.u E.g E.u
		odd n: A1.g A1.u A2.g A2.u E1.g E1.u E2.g E2.u ...
D_{nh}	$D(NH)$	n=2: A.g A.u B1.g B1.u B2.g B2.u B3.g B3.u
		n=4: A.g A.u B1.g B1.u B2.g B2.u B3.g B3.u E.g E.u
		even n: A.g A.u B1.g B1.u B2.g B2.u B3.g B3.u E1.g E1.u E2.g E2.u ...
		n=3: A1' A1'' A2' A2'' E' E''
		odd n: A1' A1'' A2' A2'' E1' E1'' E2' E2'' ...

The symmetry labeling may depend on the choice of coordinate system. For instance, B1 and B2 representations in C_{nv} are interchanged when you rotate the system by 90 degrees around the z-axis so that x-axis becomes y-axis and vice-versa (apart from sign).

12.3.2 Molecular orientation requirements

In order that AMS recognizes the (sub)symmetry of a molecule, the molecule has to have a specific orientation in space.

- The origin is a fixed point of the symmetry group.
- The z-axis is the main rotation axis.
- xy is the σ_h -plane (axial groups, C(s)).
- The x-axis is a C_2 axis (D symmetries).
- The xz-plane is a σ_v -plane (C_{nv} symmetries).
- In T_d , O, and O_h the z-axis is a fourfold axis (S_4 or C_4) and the (111)-direction is a threefold axis.
- In T and T_h the z-axis is a C_2 axis and the (111)-direction is a threefold axis.
- In I and I_h the z-axis is a C_5 axis and the x-axis is a C_2 axis

If the system is symmetrized (and no symmetry is given in the System block key) the molecular structure is rotranslated into this standard orientation.

Molecular orientation conventions

The molecular requirements for AMS may not fully determine the orientation of the molecule, like for D_{2h} molecules. See Mulliken¹ for standard conventions on the molecular orientation and notation. A few of these conventions are listed below:

- in planar C_{2v} molecules the x axis is perpendicular to the plane of the molecule.
- in planar D_{2h} molecules the x axis is perpendicular to the plane of the molecule and the z-axis passes through the greatest number of atoms. If the last condition is not decisive, the z-axis should pass through the greatest number of bonds.
- in planar D_{4h} and D_{6h} molecules the C_2 -axis passes through the greatest number of atoms. If the last condition is not decisive, the C_2 -axis should pass through the greatest number of bonds.

¹ R.S Mulliken, *Report on Notation for the Spectra of Polyatomic Molecules*, Journal of Chemical Physics 23, 1997 (1955) (<https://doi.org/10.1063/1.1740655>)

REQUIRED CITATIONS

13.1 General references

When you publish results in the scientific literature that were obtained through the AMS driver program, you are required to include a reference to the program package with the appropriate release number:

AMS 2025.1, SCM, Theoretical Chemistry, Vrije Universiteit, Amsterdam, The Netherlands, <http://www.scm.com>. Optionally, you may add the following list of authors and contributors: R. Rüger, M. Franchini, T. Trnka, A. Yakovlev, E. van Lenthe, P. Philipsen, T. van Vuren, B. Klumpers, T. Soini

The engine used for a particular calculation might require you to include other references. Please refer to the specific *engine manuals* (page 317) for required citations.

In addition to these general references, certain AMS features require additional citations, in case you have used them. An overview of these is given in the *Feature references* section below.

Note: If you have used a modified (by yourself, for instance) version of the code, you should mention in the citation that a modified version has been used.

13.2 Feature references

13.2.1 Frequencies, IR Intensities, Raman, VCD

Mode tracking

M. Reiher, J. Neugebauer, *A mode-selective quantum chemical method for tracking molecular vibrations applied to functionalized carbon nanotubes*, *Journal of Chemical Physics* 118, 1634 (2003) (<https://doi.org/10.1063/1.1523908>)

M. Reiher, J. Neugebauer, *Convergence characteristics and efficiency of mode-tracking calculations on pre-selected molecular vibrations*, *Physical Chemistry Chemical Physics* 6, 4621 (2004) (<http://dx.doi.org/10.1039/B406134A>)

C. Herrmann, M. Reiher, J. Neugebauer, *Finding a needle in a haystack: direct determination of vibrational signatures in complex systems*, *New Journal of Chemistry* 31, 818 (2007) (<http://dx.doi.org/10.1039/B618769M>)

Mode tracking based on intensities

M. Reiher, J. Neugebauer, *A mode-selective quantum chemical method for tracking molecular vibrations applied to functionalized carbon nanotubes*, *Journal of Chemical Physics* 118, 1634 (2003) (<https://doi.org/10.1063/1.1523908>)

S. Luber, J. Neugebauer, M. Reiher, *Intensity tracking for theoretical infrared spectroscopy of large molecules*, *Journal of Chemical Physics* 130, 064105 (2009) (<https://doi.org/10.1063/1.3069834>)

Mode refinement

T.Q. Teodoro, M.A.J. Koenis, S.E. Galembeck, V.P. Nicu, W.J. Buma, L. Visscher, *A frequency range selection method for vibrational spectra*, *J. Phys. Chem. Lett.*, 9 (23), 6878 (2018) (<https://doi.org/10.1021/acs.jpcllett.8b02963>)

Mobile Block Hessian (MBH)

A. Ghysels, D. Van Neck, V. Van Speybroeck, T. Verstraelen and M. Waroquier, *Vibrational Modes in partially optimized molecular systems* *Journal of Chemical Physics* 126, 224102 (2007) (<https://doi.org/10.1063/1.2737444>)

Raman scattering

S.J.A. van Gisbergen, J.G. Snijders and E.J. Baerends, *Application of time-dependent density functional response theory to Raman scattering*, *Chemical Physics Letters* 259, 599 (1996) ([https://doi.org/10.1016/0009-2614\(96\)00858-5](https://doi.org/10.1016/0009-2614(96)00858-5))

Resonance Raman: excited-state finite lifetime

L. Jensen, L. Zhao, J. Autschbach and G.C. Schatz, *Theory and method for calculating resonance Raman scattering from resonance polarizability derivatives*, *Journal of Chemical Physics* 123, 174110 (2005) (<https://doi.org/10.1063/1.2046670>)

Resonance Raman: excited-state gradient

J. Neugebauer, E.J. Baerends, E. Efremov, F. Ariese and C. Gooijer, *Combined Theoretical and Experimental Deep-UV Resonance Raman Studies of Substituted Pyrenes*, *Journal of Physical Chemistry A* 109, 2100 (2005) (<https://doi.org/10.1021/jp045360d>)

VROA: (Resonance) vibrational Raman optical activity

L. Jensen, J. Autschbach, M. Krykunov, and G.C. Schatz, *Resonance vibrational Raman optical activity: A time-dependent density functional theory approach*, *Journal of Chemical Physics* 127, 134101 (2007) (<https://doi.org/10.1063/1.2768533>)

Vibrational Circular Dichroism (VCD)

V.P. Nicu J. Neugebauer S.K. Wolff and E.J. Baerends, *A vibrational circular dichroism implementation within a Slater-type-orbital based density functional framework and its application to hexa- and hepta-helicenes*, *Theoretical Chemical Accounts* 119, 245 (2008) (<https://doi.org/10.1007/s00214-006-0234-x>)

VCD analysis: VCDtools

V.P. Nicu, J. Neugebauer and E.J. Baerends, *Effects of Complex Formation on Vibrational Circular Dichroism Spectra*, *Journal of Physical Chemistry A* 112, 6978 (2008) (<https://doi.org/10.1021/jp710201q>)

M.A.J. Koenis, O. Visser, L. Visscher, W.J. Buma, V.P. Nicu, *GUI Implementation of VCDtools, A Program to Analyze Computed Vibrational Circular Dichroism Spectra*, *J. Chem. Inf. Model* 60, 259 (2020) (<https://pubs.acs.org/doi/abs/10.1021/acs.jcim.9b00956>)

V.P. Nicu, *Revisiting an old concept: the coupled oscillator model for VCD. Part 1: the generalised coupled oscillator mechanism and its intrinsic connection to the strength of VCD signals*, *Physical Chemistry Chemical Physics* 18, 21202 (2016) (<https://doi.org/10.1039/C6CP01282E>).

Franck-Condon factors

J.S. Seldenthuis, H.S.J. van der Zant, M.A. Ratner and J.M. Thijssen, *Vibrational Excitations in Weakly Coupled Single-Molecule Junctions: A Computational Analysis*, *ACS Nano* 2, 1445 (2008) (<https://doi.org/10.1021/nn800170h>)

13.2.2 PES Exploration

Samuel T Chill, Matthew Welborn, Rye Terrell, Liang Zhang, Jean-Claude Berthet, Andreas Pedersen, Hannes Jónsson and Graeme Henkelman *EON: software for long time simulations of atomic scale systems*, *Modelling Simul. Mater. Sci. Eng.* 22 055002 (2014) (<https://doi.org/10.1088/0965-0393/22/5/055002>)

EXTERNAL PROGRAMS AND LIBRARIES

Third party software used in the 2025.1 version of the Amsterdam Modeling Suite can be found in the file titled “third-party-software.txt” in the root of your AMS installation.

KEYWORDS

15.1 Links to manual entries

ams:

- *BondOrders* (page 311)
- *Constraints* (page 56)
- *ElasticTensor* (page 243)
- *Engine* (page 317)
- *EngineAddons* (page 332)
- *EngineDebugging* (page 566)
- *EngineRestart* (page 72)
- *ExitCondition* (page 189)
- *GCMC* (page 195)
- *GeometryOptimization* (page 50)
- *IRC* (page 95)
- *LoadEngine* (page 325)
- *LoadSystem* (page 41)
- *MolecularDynamics* (page 101)
- *Molecules* (page 312)
- *NEB* (page 86)
- *NormalModes* (page 249)
- *NumericalDifferentiation* (page 245)
- *NumericalPhonons* (page 274)
- *PESExploration* (page 205)
- *PESPointCharacter* (page 240)
- *PESScan* (page 79)
- *Print* (page 565)
- *Properties* (page 9)
- *Raman* (page 278)
- *Replay* (page 192)
- *Restrains* (page 334)
- *Symmetry* (page 36)
- *System* (page 31)
- *System > ElectrostaticEmbedding* (page 40)
- *Task* (page 9)
- *Thermo* (page 270)
- *TransitionStateSearch* (page 73)
- *UseSymmetry* (page 36)
- *VibrationalAnalysis > NormalModes > ModeSelect* (page 266)

ams_interactive:

- *Stop* (page 14)
- *StopProperties* (page 15)

amsbatch:

- *SystemsFromConformers* (page 341)
- *SystemsFromSMILES* (page 340)
- *SystemsFromTrajectory* (page 341)

analysis:

- *AutoCorrelation* (page 432)
- *AverageBinPlot* (page 446)
- *Histogram* (page 422)
- *Histogram > Axes > Axis > Atoms* (page 423)
- *Histogram > Axes > Axis > VecElements* (page 424)
- *MeanSquareDisplacement* (page 442)
- *RadialDistribution* (page 421)
- *Task* (page 418)
- *TrajectoryInfo* (page 416)

conformers:

- *Equivalence* (page 348)
- *Expander* (page 380)
- *Filter* (page 349)
- *Generator* (page 347)
- *GeometryOptimization* (page 399)
- *InputConformersSet* (page 346)
- *InputMaxConfs* (page 346)
- *InputMaxEnergy* (page 346)
- *Output* (page 353)
- *RNGSeed* (page 353)
- *Task* (page 345)

pipe:

- *WorkerCommand* (page 28)

15.2 Summary of all keywords

15.2.1 ams

BondOrders**Type**

Block

Description

Configures details regarding the calculation/guessing of bond orders. To request the calculation of bond orders, use the 'Properties%BondOrders' key.

Method**Type**

Multiple Choice

Default value

EngineWithGuessFallback

Options

[Engine, Guess, EngineWithGuessFallback]

Description

How to compute the bond orders when they are requested via the 'Properties%BondOrders' key.

'Engine': let the engine compute the bond orders. The specific method used to compute the bond orders depends on the engine selected, and it may be configurable in the engine's input. Note: the calculation may stop if the engine cannot compute bond orders.

'Guess': Use a bond guessing algorithm based on the system's geometry. This is the same algorithm that is used by the Graphical User Interface to guess bonds.

'EngineWithGuessFallback': let the engine compute the bond orders (same as in 'Engine' option) but if the engine did not produce any bond orders, use the bond guessing algorithm as a fallback option.

Constraints**Type**

Block

Description

The Constraints block allows geometry optimizations and potential energy surface scans with constraints. The constraints do not have to be satisfied at the start of the calculation.

All**Type**

String

Recurring

True

Description

Fix multiple distances using one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 as well as the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then any bond between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk "*" denoting any atom. If the distance is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at the start of the simulation can be constrained, which means that the bonds may need to be specified in the System block.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

Angle**Type**

String

Recurring

True

Description

Fix the angle between three atoms. Three atom indices followed by an angle in degrees.

Atom**Type**

Integer

Recurring

True

Description

Fix the position of an atom. Just one integer referring to the index of the atom in the [System%Atoms] block.

AtomList

Type

Integer List

Recurring

True

Description

Fix positions of the specified atoms. A list of integers referring to indices of atoms in the [System%Atoms] block.

Block**Type**

String

Recurring

True

Description

Name of the region to constrain as a rigid block. Regions are specified in the System%Atoms block.

BlockAtoms**Type**

Integer List

Recurring

True

Description

List of atom indices for a block constraint, where the internal degrees of freedom are frozen.

Coordinate**Type**

String

Recurring

True

Description

Fix a particular coordinate of an atom. Atom index followed by (x|y|z).

DifDist**Type**

String

Recurring

True

Description

Four atom indices i j k l followed by the distance in Angstrom. This will constrain the difference $R(ij)-R(kl)$ at the given value.

Dihedral**Type**

String

Recurring

True

Description

Fix the dihedral angle between four atoms. Four atom indices followed by an angle in degrees.

Distance**Type**

String

Recurring

True

Description

Fix the distance between two atoms. Two atom indices followed by the distance in Angstrom.

EqualStrain**Type**

String

Description

Exclusively for lattice optimizations:

Accepts a set of strain components [xx, xy, xz, yy, yz, zz] which are to be kept equal.

The applied strain will be determined by the average of the corresponding stress tensors components.

In AMSinput just check the corresponding check buttons.

FixedRegion**Type**

String

Recurring

True

Description

Fix positions of all atoms in a region.

FreezeStrain**Type**

String

Description

Exclusively for lattice optimizations:

Freezes any lattice deformation corresponding to a particular component of the strain tensor.

Accepts a set of strain components [xx, xy, xz, yy, yz, zz] to be frozen.

In AMSinput just check the corresponding check buttons.

SumDist**Type**

String

Recurring

True

Description

Four atom indices i j k l followed by the distance in Angstrom. This will constrain the sum $R(ij)+R(kl)$ at the given value.

ElasticTensor**Type**

Block

Description

Options for numerical evaluation of the elastic tensor.

ConvergenceQuality**Type**

Multiple Choice

Default value

Good

Options

[Normal, Good, VeryGood]

GUI name

Convergence

Description

The tightness of the convergence of the geometry optimizations for each strain deformation. This should not be set higher than the overall convergence quality of the preceding geometry optimization configured by the `GeometryOptimization%Convergence%Quality` keyword.

Parallel**Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup

Type

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

StrainStepSize**Type**

Float

Default value

0.001

Description

Step size (relative) of strain deformations used for computing the elastic tensor numerically.

Engine**Type**

Block

Description

The input for the computational engine. The header of the block determines the type of the engine.

EngineAddons**Type**

Block

Description

This block configures all the engine add-ons.

AtomEnergies**Type**

Non-standard block

Description

Add an element-dependent energy per atom. On each line, give the chemical element followed by the energy (in atomic units).

D3Dispersion**Type**

Block

Description

This block configures the add-on that adds the Grimme D3 dispersion correction to the engine's energy, gradients, and stress tensor.

Damping**Type**

Multiple Choice

Default value

BJ

Options

[BJ, Zero]

Description

Type of damping: BJ (Becke-Johnson) or Zero. BJ is recommended for most applications.

Enabled**Type**

Bool

Default value

No

GUI name

D3 dispersion

Description

Enables the D3 dispersion correction addon.

Functional**Type**

String

Default value

PBE

Description

Use the D3 parameterization by Grimme for a given xc-functional. Accepts the same values as the `-func` command line option of the official `dftd3` program. Note: the naming convention is different from elsewhere in the AMS suite. For example, BLYP should be called b-lyp.

a1**Type**

Float

Description

The a1 parameter. Only used if Damping is set to BJ. If set, it overwrites the a1 value for the chosen functional.

a2**Type**

Float

Description

The a2 parameter. Only used if Damping is set to BJ. If set, it overwrites the a2 value for the chosen functional.

s6**Type**

Float

Description

The s6 parameter, global scaling parameter. If set, it overwrites the s6 value for the chosen functional.

s8**Type**

Float

Description

The s8 parameter. If set, it overwrites the s8 value for the chosen functional.

sr6**Type**

Float

Description

The sr6 parameter. Only used if Damping is set to Zero. If set, it overwrites the sr6 value for the chosen functional.

D4Dispersion**Type**

Block

Description

This block configures the addon that adds the Grimme D4(EEQ) dispersion correction to the engine's energy, gradients, stress tensor and Hessian.

Enabled**Type**

Bool

Default value

No

GUI name

D4 dispersion

Description

Enables the D4 dispersion correction addon.

Functional**Type**

Multiple Choice

Default value

PBE

Options

[HF, BLYP, BPBE, BP86, BPW, LB94, MPWLYP, MPWPW91, OLYP, OPBE, PBE, RPBE, REVPBE, PW86PBE, RPW86PBE, PW91, PW91P86, XLYP, B97, TPSS, REVTPSS, SCAN, B1LYP, B3LYP, BHLYP, B1P86, B3P86, B1PW91, B3PW91, O3LYP, REVPBE0, REVPBE38, PBE0, PWP1, PW1PW, MPW1PW91, MPW1LYP, PW6B95, TPSSH, TPSS0, X3LYP, M06L, M06, OMEGAB97, OMEGAB97X, CAM-B3LYP, LC-BLYP, LH07TSVWN, LH07SSVWN, LH12CTSSIRPW92, LH12CTSSIFPW92, LH14TCALPBE, B2PLYP, B2GPPLYP, MPW2PLYP, PWPB95, DSDBLYP, DSDPBE, DSDPBEB95, DSDPBEP86, DSDSVWN, DODBLYP, DODPBE, DODPBEB95, DODPBEP86, DODSVWN, PBE02, PBE0DH, DFTB(3ob), DFTB(mio), DFTB(pbc), DFTB(matsci), DFTB(ob2), B1B95, MPWB1K, REVTPSSH, GLYP, REVPBE0DH, REVTPSS0, REVDSDPBEP86, REVDSDPBEPBE, REVDSDBLYP, REVDODPBEP86, B97M, OMEGAB97M, R2SCAN]

Description

Use the D4 parameterization by Grimme for a given xc-functional.

Verbosity**Type**

Multiple Choice

Default value

Silent

Options

[Silent, Normal, Verbose, VeryVerbose]

Description

Controls the verbosity of the dftd4 code. Equivalent to the `–silent` and `–verbose` command line switches of the official dftd4 program.

a1**Type**

Float

Description

The a1 parameter, see D4 article. The physically reasonable range for a1 is [0.0,1.0]. If set, it overwrites the a1 value for the chosen functional.

a2**Type**

Float

Description

The a2 parameter, see D4 article. The physically reasonable range for a2 is [0.0,7.0]. If set, it overwrites the a2 value for the chosen functional.

s6**Type**

Float

Description

The s6 parameter, see D4 article. The physically reasonable range for s6 is [0.0,1.0]. If set, it overwrites the s6 value for the chosen functional.

s8**Type**

Float

Description

The s8 parameter, see D4 article. The physically reasonable range for s8 is [0.0,3.0]. If set, it overwrites the s8 value for the chosen functional.

s9**Type**

Float

Description

The s9 parameter, see D4 article. If set, it overwrites the s9 value for the chosen functional.

ExternalEngine**Type**

Block

Description

External engine as an addon

Execute

Type
String

GUI name
Execute

Description
execute command

ExternalStress

Type
Block

Description
This block configures the addon that adds external stress term to the engine's energy and stress tensor.

StressTensorVoigt

Type
Float List

Unit
Hartree/Bohr³

GUI name
External stress tensor

Description
The elements of the external stress tensor in Voigt notation. One should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

UpdateReferenceCell

Type
Bool

Default value
No

Description
Whether or not the reference cell should be updated every time the system changes (see documentation).

PipeEngine

Type
Block

Description
Pipe engine as an addon

WorkerCommand

Type
String

GUI name
Worker command

Description
pipe worker command

Pressure**Type**

Float

Default value

0.0

Unit

GPa

Description

Add a hydrostatic pressure term to the engine's energy and stress tensor. Can only be used for 3D periodic boundary conditions.

Repulsion**Type**

Block

Description

This block configures an addon that adds a repulsive Weeks-Chandler-Andersen potential to all atom pairs.

Enabled**Type**

Bool

Default value

No

GUI name

Repulsion

Description

Enables the repulsive Weeks-Chandler-Andersen potential addon.

When enabled, all atom pairs will experience repulsion $E = 4 \cdot \epsilon \cdot \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 + \frac{1}{4} \right)$ at the distances shorter than about $1.12 \cdot \sigma$.

Epsilon**Type**

Float

Default value

0.01

Unit

Hartree

Description

The epsilon parameter in the potential equation. It is equal to the amount of energy added at $r = \sigma$.

HydrogenSigmaScale**Type**

Float

Default value

0.75

Unit

Angstrom

Description

The sigma parameter for a pair of atoms where one of them is hydrogen is scaled with the given factor. For H-H interactions the sigma is scaled with this value squared.

Sigma**Type**

Float

Default value

0.55

Unit

Angstrom

Description

The sigma parameter in the potential equation. The potential is exactly zero at the distances larger than about $1.12 \cdot \text{sigma}$

SkinLength**Type**

Float

Default value

2.0

Unit

Angstrom

Description

Technical parameter specifying skin length for the neighbor list generation. A larger value increases the neighbor list cutoff (and cost) but reduces the frequency it needs to be re-created.

WallPotential**Type**

Block

Description

This block configures the addon that adds a spherical wall potential to the engine's energy and gradients.

Enabled**Type**

Bool

Default value

No

Description

Enables the wall potential addon. When enabled, a spherical wall of radius [Radius] around the origin will be added. The force due to the potential will decay exponentially inside the wall, will be close to [Prefactor*Gradient] outside and exactly half of that at the wall.

Gradient**Type**

Float

Default value

10.0

Unit

1/Angstrom

Description

The radial gradient outside the sphere.

Prefactor**Type**

Float

Default value

0.01

Unit

Hartree

Description

The multiplier for the overall strength of the potential.

Radius**Type**

Float

Default value

30.0

Unit

Angstrom

Value Range

value > 0

Description

The radius of the sphere, wherein the potential is close to zero.

EngineDebugging**Type**

Block

Description

This block contains some options useful for debugging the computational engines.

AlwaysClaimSuccess**Type**

Bool

Default value

No

Description

If an engine fails, pretend that it worked. This can be useful when you know that an SCF might fail.

CheckInAndOutput**Type**

Bool

Default value

No

Description

Enables some additional checks on the input and output of an engine, e.g. for NaN values.

ForceContinuousPES**Type**

Bool

Default value

No

Description

If this option is set, the engine will always run in continuous PES mode. For many engines this disables the use of symmetry, as this one always leads to a discontinuous PES around the symmetric points: Basically there is jump in the PES at the point where the symmetry detection starts classifying the system as symmetric. Normally the continuous PES mode of the engine (often disabling the symmetry) is only used when doing numerical derivatives, but this flag forces the engine to continuously run in this mode.

IgnoreGradientsRequest**Type**

Bool

Default value

No

Description

If this option is set, the engine will not do analytical gradients if asked for it, so that gradients will have to be evaluated numerically by AMS.

IgnorePreviousResults**Type**

Bool

Default value

No

Description

If this option is set, the engine will not receive information from previous calculations. Typically this information is used to restart the self consistent procedure of the engine.

IgnoreStressTensorRequest**Type**

Bool

Default value

No

Description

If this option is set, the engine will not calculate an analytical stress tensor if asked for it, so that the stress tensor will have to be evaluated numerically by AMS.

NeverQuiet**Type**

Bool

Default value

No

Description

Makes the engine ignore the request to work quietly.

RandomFailureChance**Type**

Float

Default value

0.0

Description

Makes the engine randomly report failures, even though the results are actually fine. Useful for testing error handling on the application level.

RandomNoiseInEnergy**Type**

Float

Default value

0.0

Unit

Hartree

Description

Adds a random noise to the energy returned by the engine. The random contribution is drawn from $[-r, r]$ where r is the value of this keyword.

RandomNoiseInGradients**Type**

Float

Default value

0.0

Unit

Hartree/Angstrom

Description

Adds a random noise to the gradients returned by the engine. A random number in the range $[-r, r]$ (where r is the value of this keyword) is drawn and added separately to each component of the gradient.

RandomStopChance**Type**

Float

Default value

0.0

Description

Makes the engine randomly stop. Can be used to simulate crashes.

EngineRestart**Type**

String

Description

The path to the file from which to restart the engine.

Should be a proper engine result file (like adf.rkf, band.rkf etc), or the name of the results directory containing it.

ExitCondition**Type**

Block

Recurring

True

Description

If any of the specified exitconditions are met, the AMS driver will exit cleanly.

AtomsTooClose**Type**

Block

Description

If any pair of atoms is closer than the specified minimum value, the program will exit cleanly.

MinimumDistance**Type**

Float

Default value

0.7

Unit

Angstrom

Description

Two atoms closer than this threshold value are considered too close.

PairCalculation**Type**

Multiple Choice

Default value

NeighborList

Options

[NeighborList, DistanceMatrix]

Description

Two atoms closer than this threshold value are considered too close.

EngineEnergyUncertainty**Type**

Block

Description

If the engine reports an uncertainty that is too high, the program will exit cleanly.

MaxUncertainty**Type**

Float

Default value

0.001

Unit

Hartree

Description

Threshold for Engine Energy Uncertainty divided by Normalization (by default the number of atoms)

Normalization**Type**

Float

Value Rangevalue \geq 0.0**Description**

Divide the reported Engine Energy Uncertainty by this normalization. Will divide by the number of atoms if unset.

EngineGradientsUncertainty**Type**

Block

Description

If the engine reports an uncertainty in the magnitude of the nuclear gradient of any atom that is too high, the program will exit cleanly.

MaxUncertainty**Type**

Float

Default value

0.01580221

Unit

Hartree/Angstrom

Description

Threshold for Engine Gradients Uncertainty.

Type**Type**

Multiple Choice

Default value

AtomsTooClose

Options

[AtomsTooClose, EngineEnergyUncertainty, EngineGradientsUncertainty]

Description

The type of exitcondition specified

FallbackSolveAfterEngineFailure**Type**

Bool

Default value

Yes

Description

If the engine fails to Solve, try to re-run the Solve without restarting the engine from the previous results. This generally decreases the engine failure rate. Only relevant certain tasks, such as GeometryOptimization, MolecularDynamics, Replay, IRC.

GCMC**Type**

Block

Description

This block controls the Grand Canonical Monte Carlo (GCMC) task.

By default, molecules are added at random positions in the simulation box. The initial position is controlled by

AccessibleVolume**Type**

Float

Default value

0.0

Description

Volume available to GCMC, in cubic Angstroms. AccessibleVolume should be specified for “Accessible” and “FreeAccessible” [VolumeOption].

Box**Type**

Block

Description

Boundaries of the insertion space, i.e. coordinates of the origin of an inserted molecule (coordinates of an atom of the inserted system may fall outside the box).

For a periodic dimension it is given as a fraction of the simulation box (the full 0 to 1 range by default). For a non-periodic dimension it represents absolute Cartesian coordinates in Angstrom (the system’s bounding box extended by the MaxDistance value by default).

Amax**Type**

Float

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Description

Coordinate of the upper bound along the third axis.

Cmin**Type**

Float

Description

Coordinate of the lower bound along the third axis.

Ensemble**Type**

Multiple Choice

Default value

Mu-VT

Options

[Mu-VT, Mu-PT]

Description

Select the MC ensemble: Mu-VT for fixed volume or Mu-PT for variable volume. When the Mu-PT ensemble is selected the [Pressure] and [VolumeChangeMax] should also be specified.

Iterations**Type**

Integer

GUI name

Number of GCMC iterations

Description

Number of GCMC moves.

MapAtomsToOriginalCell**Type**

Bool

Default value

Yes

Description

Keeps the atom (mostly) in the original cell by mapping them back before the geometry optimizations.

MaxDistance**Type**

Float

Default value

3.0

Unit

Angstrom

GUI name

Add molecules within

Description

The max distance to other atoms of the system when adding the molecule.

MinDistance**Type**

Float

Default value

0.3

Unit

Angstrom

GUI name

Add molecules not closer than

Description

Keep the minimal distance to other atoms of the system when adding the molecule.

Molecule**Type**

Block

Recurring

True

GUI name

Molecules

Description

This block defines the molecule (or atom) that can be inserted/moved/deleted with the MC method. The coordinates should form a reasonable structure. The MC code uses these coordinates during the insertion step by giving them a random rotation, followed by a random translation to generate a random position of the molecule inside the box. Currently, there is no check to make sure all atoms of the molecule stay inside the simulation box. The program does check that the MaxDistance/MinDistance conditions are satisfied.

ChemicalPotential**Type**

Float

Unit

Hartree

Description

Chemical potential of the molecule (or atom) reservoir.

It is used when calculating the Boltzmann accept/reject criteria after a MC move is executed. This value can be derived from first principles using statistical mechanics, or equivalently, it can be determined from thermochemical tables available in literature sources.

For example, the proper chemical potential for a GCMC simulation in which single oxygen atoms are exchanged with a reservoir of O₂ gas, should equal 1/2 the chemical potential of O₂ at the temperature and pressure of the reservoir:

$$\text{cmpot} = \mu_{\text{O}}(T,P) = 1/2 * \mu_{\text{O}_2}(T,P) = 1/2 * [\mu_{\text{ref}}(T,P_{\text{ref}}) + kT * \log(P/P_{\text{ref}}) - E_{\text{diss}}]$$

where the reference chemical potential [$\mu_{\text{ref}}(T,P_{\text{ref}})$] is the experimentally determined chemical potential of O₂ at T and P_{ref}; $kT * \log(P/P_{\text{ref}})$ is the pressure correction to the free energy, and E_{diss} is the dissociation energy of the O₂ molecule.

NoAddRemove**Type**

Bool

Default value

No

GUI name

Fix molecule

Description

Set to True to tell the GCMC code to keep the number of molecules/atoms of this type fixed.

It will thus disable Insert/Delete moves on this type, meaning it can only do a displacement move, or volume change move (for an NPT ensemble).

SystemName**Type**

String

GUI name

Molecule

Description

String ID of a named [System] to be inserted. The lattice specified with this System, if any, is ignored and the main system's lattice is used instead.

NonAccessibleVolume**Type**

Float

Default value

0.0

GUI name

Non-accessible volume

Description

Volume not available to GCMC, in cubic Angstroms. NonAccessibleVolume may be specified for the "Free" [VolumeOption] to reduce the accessible volume.

NumAttempts**Type**

Integer

Default value

1000

GUI name

Max tries

Description

Try inserting/moving the selected molecule up to the specified number of times or until all constraints are satisfied. If all attempts fail a message will be printed and the simulation will stop. If the MaxDistance-MinDistance interval is small this number may have to be large.

Pressure**Type**

Float

Default value

0.0

Unit

Pascal

Description

Pressure used to calculate the energy correction in the Mu-PT ensemble. Set it to zero for incompressible solid systems unless at very high pressures.

Removables**Type**

Non-standard block

Description

The Removables can be used to specify a list of molecules that can be removed or moved during this GCMC calculation. Molecules are specified one per line in the format following format:

MoleculeName atom1 atom2 ...

The MoleculeName must match a name specified in one of the [Molecule] blocks. The atom indices refer to the whole input System and the number of atoms must match that in the specified Molecule. A suitable Removables block is written to the standard output after each accepted MC move. If you do so then you should also replace the initial atomic coordinates with the ones found in the same file. If a [Restart] key is present then the Removables block is ignored.

Restart**Type**

String

Description

Name of an RKF restart file. Upon restart, the information about the GCMC input parameters, the initial system (atomic coordinates, lattice, charge, etc.) and the MC molecules (both already inserted and to be inserted) are read from the restart file. The global GCMC input parameters and the MC Molecules can be modified from input. Any parameter not specified in the input will use its value from the restart file (i.e. not the default value). Molecules found in the restart file do not have to be present as named Systems in the input, however if there is a System present that matches the name of a molecule from restart then the System's geometry will replace that found in the restart file. It is also possible to specify new Molecules in the input, which will be added to the pool of the MC molecules from restart.

SwapAtoms

Type

Block

Description

Experimental: Occasionally swap the coordinates of a random pair of atoms from two regions.

Probability**Type**

Float

Default value

0.0

Description

Probability of performing a swap move instead of any other GCMC move in a single GCMC iteration.

Regions**Type**

String

Description

Names of two regions to swap between (separated by a space).

Temperature**Type**

Float

Default value

300.0

Unit

Kelvin

Description

Temperature of the simulation. Increase the temperature to improve the chance of accepting steps that result in a higher energy.

UseGCPreFactor**Type**

Bool

Default value

Yes

GUI name

Use GC prefactor

Description

Use the GC pre-exponential factor for probability.

VolumeChangeMax**Type**

Float

Default value

0.05

Description

Fractional value by which logarithm of the volume is allowed to change at each step. The new volume is then calculated as $V_{\text{new}} = \exp(\text{random}(-1:1) * \text{VolumeChangeMax}) * V_{\text{old}}$

VolumeOption**Type**

Multiple Choice

Default value

Free

Options

[Free, Total, Accessible, FreeAccessible]

GUI name

Volume method

Description

Specifies the method to calculate the volume used to calculate the GC pre-exponential factor and the energy correction in the Mu-PT ensemble:

Free: $V = \text{totalVolume} - \text{occupiedVolume} - \text{NonAccessibleVolume}$;

Total: $V = \text{totalVolume}$;

Accessible: $V = \text{AccessibleVolume}$;

FreeAccessible: $V = \text{AccessibleVolume} - \text{occupiedVolume}$.

The AccessibleVolume and NonAccessibleVolume are specified in the input, the occupiedVolume is calculated as a sum of atomic volumes.

GeometryOptimization**Type**

Block

Description

Configures details of the geometry optimization and transition state searches.

CalcPropertiesOnlyIfConverged**Type**

Bool

Default value

Yes

Description

Compute the properties requested in the 'Properties' block, e.g. Frequencies or Phonons, only if the optimization (or transition state search) converged. If False, the properties will be computed even if the optimization did not converge.

Convergence**Type**

Block

Description

Convergence is monitored for up to 4 quantities: the energy change, the Cartesian gradients, the Cartesian step size, and for lattice optimizations the stress energy per atom. Convergence criteria can be specified separately for each of these items.

Energy

Type

Float

Default value

1e-05

Unit

Hartree

Value Range

value > 0

GUI name

Energy convergence

Description

The criterion for changes in the energy. The energy is considered converged when the change in energy is smaller than this threshold times the number of atoms.

Gradients**Type**

Float

Default value

0.001

Unit

Hartree/Angstrom

Value Range

value > 0

GUI name

Gradient convergence

Description

Threshold for nuclear gradients.

Quality**Type**

Multiple Choice

Default value

Custom

Options

[VeryBasic, Basic, Normal, Good, VeryGood, Custom]

GUI name

Convergence

Description

A quick way to change convergence thresholds: 'Good' will reduce all thresholds by an order of magnitude from their default value. 'VeryGood' will tighten them by two orders of magnitude. 'Basic' and 'VeryBasic' will increase the thresholds by one or two orders of magnitude respectively.

Step**Type**

Float

Default value

0.01

Unit

Angstrom

Value Range

value > 0

GUI name

Step convergence

Description

The maximum Cartesian step allowed for a converged geometry.

StressEnergyPerAtom**Type**

Float

Default value

0.0005

Unit

Hartree

Value Range

value > 0

Description

Threshold used when optimizing the lattice vectors. The stress is considered ‘converged’ when the maximum value of $\text{stress_tensor} * \text{cell_volume} / \text{number_of_atoms}$ is smaller than this threshold (for 2D and 1D systems, the cell_volume is replaced by the cell_area and cell_length respectively).

CoordinateType**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Delocalized, Cartesian]

GUI name

Optimization space

Description

Select the type of coordinates in which to perform the optimization. ‘Auto’ automatically selects the most appropriate CoordinateType for a given Method.

If ‘Auto’ is selected, Delocalized coordinates will be used for the Quasi-Newton method, while Cartesian coordinates will be used for all other methods.

Dimer**Type**

Block

Description

Options for the Dimer method for transition state search.

AngleThreshold**Type**

Float

Default value

1.0

Unit

Degree

Description

The rotation is considered converged when the the rotation angle falls below the specified threshold.

DimerDelta**Type**

Float

Default value

0.01

Unit

Angstrom

Description

Euclidian distance between the midpoint and the endpoint.

ExtrapolateForces**Type**

Bool

Default value

Yes

Description

Set to false to call engine to calculate forces at the extrapolated rotation angle instead of extrapolating them.

LBFGSMaxVectors**Type**

Integer

Default value

10

Description

Max number of vectors for the L-BFGS algorithm to save.

MaxRotationIterations**Type**

Integer

Default value

10

Description

Maximum number of rotation iterations for a single translation step.

Region

Type

String

Default value

*

Description

Include only atoms of the specified region(s) in the rotations, which allows searching for a transition state involving selected atoms only.

RotationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during rotation iterations.

TranslationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during translation iterations.

EngineAutomations**Type**

Block

Description

The optimizer can change some settings of the engine, based for instance on the error.

The idea is to allow the engine to be a bit quicker at the start, and more accurate towards the end.

Automations are always engine specific.

Enabled**Type**

Bool

Default value

Yes

Description

Whether or not automations are enabled at all.

Gradient

Type

Block

Recurring

True

Description

A gradient-based automation.

FinalValue**Type**

Float

Description

This value will be used whenever the gradient is less than GradientLow

HighGradient**Type**

Float

Default value

1.0

Unit

Hartree/Angstrom

Description

Defines a large gradient. When the actual gradient is between GradientHigh and GradientLow a linear interpolation scheme is used for kT (on a log scale).

InitialValue**Type**

Float

Description

This value will be used at the first geometry, and whenever the gradient is higher than GradientHigh

LowGradient**Type**

Float

Default value

1.0

Unit

Hartree/Angstrom

Description

Defines a small gradient, see GradientHigh

UseLogInterpolation**Type**

Bool

Default value

Yes

Description

Whether to use interpolation on a log (y) scale or not

Variable**Type**

String

Default value**Description**

variable to be tweaked for the engine.

Iteration**Type**

Block

Recurring

True

Description

Geometry step based automation.

FinalValue**Type**

Float

Description**FirstIteration****Type**

Integer

Default value

1

Description

When the actual gradient is between the first and last iteration, a linear interpolation is used.

InitialValue**Type**

Float

Description

This value will be used when the iteration number is smaller or equal to FirstIteration

LastIteration**Type**

Integer

Default value

10

Description

Where the automation should reach the FinalValue

UseLogInterpolation**Type**

Bool

Default value

Yes

Description

Whether to use interpolation on a log (y) scale or not

Variable**Type**

String

Default value**Description**

variable to be tweaked for the engine.

FIRE**Type**

Block

Description

This block configures the details of the FIRE optimizer. The keywords name correspond the the symbols used in the article describing the method, see PRL 97, 170201 (2006).

AllowOverallRotation**Type**

Bool

Default value

Yes

Description

Whether or not the system is allowed to freely rotate during the optimization. This is relevant when optimizing structures in the presence of external fields.

AllowOverallTranslation**Type**

Bool

Default value

No

Description

Whether or not the system is allowed to translate during the optimization. This is relevant when optimizing structures in the presence of external fields.

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

Map the atoms to the central cell at each geometry step.

NMin**Type**

Integer

Default value

5

Description

Number of steps after stopping before increasing the time step again.

alphaStart**Type**

Float

Default value

0.1

Description

Steering coefficient.

dtMax**Type**

Float

Default value

1.0

Unit

Femtoseconds

Description

Maximum time step used for the integration. For ReaxFF and APPLE&P, this value is reduced by 50%.

dtStart**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

Initial time step for the integration.

fAlpha**Type**

Float

Default value

0.99

Description

Reduction factor for the steering coefficient.

fDec**Type**

Float

Default value

0.5

Description

Reduction factor for reducing the time step in case of uphill movement.

fInc**Type**

Float

Default value

1.1

Description

Growth factor for the integration time step.

strainMass**Type**

Float

Default value

0.5

Description

Fictitious relative mass of the lattice degrees of freedom. This controls the stiffness of the lattice degrees of freedom relative to the atomic degrees of freedom, with smaller values resulting in a more aggressive optimization of the lattice.

HessianFree**Type**

Block

Description

Configures details of the Hessian-free (conjugate gradients or L-BFGS) geometry optimizer.

Step**Type**

Block

Description**MaxCartesianStep****Type**

Float

Default value

0.1

Unit

Angstrom

Description

Limit on a single Cartesian component of the step.

MinRadius**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Minimum value for the trust radius.

TrialStep**Type**

Float

Default value

0.0005

Unit

Angstrom

Description

Length of the finite-difference step when determining curvature. Should be smaller than the step convergence criterion.

TrustRadius**Type**

Float

Default value

0.2

Unit

Angstrom

Description

Initial value of the trust radius.

InitialHessian**Type**

Block

Description

Options for initial model Hessian when optimizing systems with the Quasi-Newton method.

File**Type**

String

GUI name

Initial Hessian from

Description

KF file containing the initial Hessian (or the results dir. containing it). This can be used to load a Hessian calculated in a previously with the [Properties%Hessian] keyword.

Type**Type**

Multiple Choice

Default value

Auto

Options

[Auto, UnitMatrix, Swart, FromFile, Calculate, CalculateWithFastEngine]

GUI name

Initial Hessian

Description

Select the type of initial Hessian. Auto: let the program pick an initial model Hessian. Unit-Matrix: simplest initial model Hessian, just a unit matrix in the optimization coordinates. Swart: model Hessian from M. Swart. FromFile: load the Hessian from the results of a previous calculation (see InitialHessian%File). Calculate: compute the initial Hessian (this may be computationally expensive and it is mostly recommended for TransitionStateSearch calculations). CalculateWithFastEngine: compute the initial Hessian with a faster engine.

KeepIntermediateResults**Type**

Bool

Default value

No

Description

Whether the full engine result files of all intermediate steps are stored on disk. By default only the last step is kept, and only if the geometry optimization converged. This can easily lead to huge amounts of data being stored on disk, but it can sometimes be convenient to closely monitor a tricky optimization, e.g. excited state optimizations going through conical intersections, etc.

...

MaxIterations**Type**

Integer

Value Rangevalue ≥ 0 **GUI name**

Maximum number of iterations

Description

The maximum number of geometry iterations allowed to converge to the desired structure.

MaxRestarts**Type**

Integer

Default value

0

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small displacement along the imaginary vibrational mode. In case the restarted optimization again does not find a minimum, this can happen multiple times in succession. This keyword sets the maximum number of restarts. The default value is 0, so the automatic restarting is disabled by default.

Method**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Quasi-Newton, FIRE, L-BFGS, ConjugateGradients, Dimer]

GUI name

Optimization method

Description

Select the optimization algorithm employed for the geometry relaxation. Currently supported are:

the Hessian-based Quasi-Newton-type BFGS algorithm,

the fast inertial relaxation method (FIRE),

the limited-memory BFGS method,

and the conjugate gradients method. The default is to choose an appropriate method automatically based on the engine's speed, the system size and the supported optimization options.

OptimizeLattice**Type**

Bool

Default value

No

Description

Whether to also optimize the lattice for periodic structures. This is currently supported with the Quasi-Newton, FIRE, and L-BFGS optimizers.

PretendConverged**Type**

Bool

Default value

No

Description

Normally a non-converged geometry optimization is considered an error. If this keyword is set to True, the optimizer will only produce a warning and still claim that the optimization is converged. (This is mostly useful for scripting applications, where one might want to consider non-converged optimizations still successful jobs.)

Quasi-Newton**Type**

Block

Description

Configures details of the Quasi-Newton geometry optimizer.

MaxGDIISVectors**Type**

Integer

Default value

0

Description

Sets the maximum number of GDIIS vectors. Setting this to a number >0 enables the GDIIS method.

Step**Type**

Block

Description**TrustRadius****Type**

Float

Description

Initial value of the trust radius.

VaryTrustRadius**Type**

Bool

Description

Whether to allow the trust radius to change during optimization. By default True during energy minimization and False during transition state search.

UpdateTSVectorEveryStep**Type**

Bool

Default value

Yes

GUI name

Update TSRC vector every step

Description

Whether to update the TS reaction coordinate at each step with the current eigenvector.

RestartDisplacement**Type**

Float

Default value

0.05

Unit

Angstrom

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small displacement along the imaginary vibrational mode. This keywords sets the size of the displacement for the furthest moving atom.

IRC**Type**

Block

Description

Configures details of the Intrinsic Reaction Coordinate optimization.

Convergence

Type

Block

Description

Convergence at each given point is monitored for two items: the Cartesian gradient and the calculated step size. Convergence criteria can be specified separately for each of these items. The same criteria are used both in the inner IRC loop and when performing energy minimization at the path ends.

Gradients**Type**

Float

Default value

0.001

Unit

Hartree/Angstrom

GUI name

Gradient convergence

Description

Convergence criterion for the max component of the residual energy gradient.

Step**Type**

Float

Default value

0.001

Unit

Angstrom

GUI name

Step convergence

Description

Convergence criterion for the max component of the step in the optimization coordinates.

CoordinateType**Type**

Multiple Choice

Default value

Cartesian

Options

[Cartesian, Delocalized]

GUI name

Coordinates used for optimization

Description

Select the type of coordinates in which to perform the optimization. Note that the Delocalized option should be considered experimental.

Direction

Type

Multiple Choice

Default value

Both

Options

[Both, Forward, Backward]

Description

Select direction of the IRC path. The difference between the Forward and the Backward directions is determined by the sign of the largest component of the vibrational normal mode corresponding to the reaction coordinate at the transition state geometry. The Forward path correspond to the positive sign of the component. If Both is selected then first the Forward path is computed followed by the Backward one.

InitialHessian**Type**

Block

Description

Options for initial Hessian at the transition state. The first eigenvalue of the initial Hessian defines direction of the first forward or backward step. This block is ignored when restarting from a previous IRC calculation because the initial Hessian found in the restart file is used.

File**Type**

String

GUI name

File

Description

If 'Type' is set to 'FromFile' then in this key you should specify the RKF file containing the initial Hessian (or the ams results dir. containing it). This can be used to load a Hessian calculated previously with the 'Properties%Hessian' keyword. If you want to also use this file for the initial geometry then also specify it in a 'LoadSystem' block.

Type**Type**

Multiple Choice

Default value

Calculate

Options

[Calculate, FromFile]

GUI name

Initial Hessian

Description

Calculate the exact Hessian for the input geometry or load it from the results of a previous calculation.

KeepConvergedResults**Type**

Bool

Default value

Yes

Description

Keep the binary RKF result file for every converged IRC point. These files may contain more information than the main ams.rkf result file.

MaxIRCSteps**Type**

Integer

GUI name

Maximum IRC steps

Description

Soft limit on the number of IRC points to compute in each direction. After the specified number of IRC steps the program will switch to energy minimization and complete the path. This option should be used when you are interested only in the reaction path area near the transition state. Note that even if the soft limit has been hit and the calculation has completed, the IRC can still be restarted with a 'RedoBackward' or 'RedoForward' option.

MaxIterations**Type**

Integer

Default value

300

GUI name

Maximum iterations

Description

The maximum number of geometry iterations allowed to converge the inner IRC loop. If optimization does not converge within the specified number of steps, the calculation is aborted.

MaxPoints**Type**

Integer

Default value

100

GUI name

Maximum points

Description

Hard limit on the number of IRC points to compute in each direction. After the specified number of IRC steps the program will stop with the current direction and switch to the next one. If both 'MaxPoints' and 'MaxIRCSteps' are set to the same value then 'MaxPoints' takes precedence, therefore this option should be used to set a limit on the number of IRC steps if you intend to use the results later for a restart.

MinEnergyProfile**Type**

Bool

Default value

No

GUI name

Minimum energy profile

Description

Calculate minimum energy profile (i.e. no mass-weighting) instead of the IRC.

MinPathLength**Type**

Float

Default value

0.1

Unit

Angstrom

Description

Minimum length of the path required before switching to energy minimization. Use this to overcome a small kink or a shoulder on the path.

Restart**Type**

Block

Description

Restart options. Upon restart, the information about the IRC input parameters and the initial system (atomic coordinates, lattice, charge, etc.) is read from the restart file. The IRC input parameters can be modified from input. Except for 'MaxPoints' and 'Direction' all parameters not specified in the input will use their values from the restart file. The 'MaxPoints' and 'Direction' will be reset to their respective default values if not specified in the input. By default, the IRC calculation will continue from the point where it left off. However, the 'RedoForward' and/or 'RedoBackward' option can be used to enforce recalculation of a part of the reaction path, for example, using a different 'Step' value.

File**Type**

String

GUI name

Restart

Description

Name of an RKF restart file generated by a previous IRC calculation. Do not use this key to provide an RKF file generated by a TransitionStateSearch or a SinglePoint calculation, use the 'LoadSystem' block instead.

RedoBackward**Type**

Integer

Default value

0

Description

IRC step number to start recalculating the backward path from. By default, if the backward path has not been completed then start after the last completed step. If the backward path has been completed and the 'RedoBackward' is omitted then no point on the backward path will be recomputed.

RedoForward**Type**

Integer

Default value

0

Description

IRC step number to start recalculating the forward path from. By default, if the forward path has not been completed then start after the last completed step. If the forward path has been completed and the 'RedoForward' is omitted then no point on the forward path will be recomputed.

Step**Type**

Float

Default value

0.2

GUI name

Step size

Description

IRC step size in mass-weighted coordinates, $\sqrt{\text{amu}} \cdot \text{bohr}$. One may have to increase this value when heavy atoms are involved in the reaction, or decrease it if the reactant or products are very close to the transition state.

LoadEngine**Type**

String

Description

The path to the file from which to load the engine configuration. Replaces the Engine block.

LoadSystem**Type**

Block

Recurring

True

Description

Block that controls reading the chemical system from a KF file instead of the [System] block.

File**Type**

String

Description

The path of the KF file from which to load the system. It may also be the results directory containing it.

Section**Type**

String

Default value

Molecule

Description

The section on the KF file from which to load the system.

Log**Type**

Non-standard block

Description

Configures the debugging loggers. Syntax: 'Level LoggerName'. Possible Levels: All, Debug, Info, Warning, Error, Fatal.

MolecularDynamics**Type**

Block

Description

Configures molecular dynamics (with the velocity-Verlet algorithm) with and without thermostats. This block allows to specify the details of the molecular dynamics calculation.

AddMolecules**Type**

Block

Recurring

True

GUI name

Add molecules

Description

This block controls adding molecules to the system (a.k.a. the Molecule Gun). Multiple occurrences of this block are possible.

By default, molecules are added at random positions in the simulation box with velocity matching the current system temperature. The initial position can be modified using one of the following keywords: Coords, CoordsBox, FractionalCoords, FractionalCoordsBox. The Coords and FractionalCoords keys can optionally be accompanied by CoordsSigma or FractionalCoordsSigma, respectively.

AtomTemperature**Type**

Float

Default value

0.0

Unit

Kelvin

Description

Add random velocity corresponding to the specified temperature to individual atoms of the molecule. This only affects rotational and internal degrees of freedom, not the net translational velocity of the inserted molecule as set by the other options.

ContactDistance

Type

Float

Default value

0.0

Unit

Angstrom

Description

Translate the bullet along the velocity vector until it comes within ContactDistance of any other atom.

Coords**Type**

Float List

Unit

Angstrom

Description

Place molecules at or around the specified Cartesian coordinates.

This setting takes precedence over other ways to specify initial coordinates of the molecule: [CoordsBox], [FractionalCoords], and [FractionalCoordsBox].

CoordsBox**Type**

Float List

Unit

Angstrom

Description

Place molecules at random locations inside the specified box in Cartesian coordinates.

Coordinates of the box corners are specified as: Xmin, Xmax, Ymin, Ymax, Zmin, Zmax.

This setting is ignored if Coords is used.

In AMSinput, if this field is not empty it will be used instead of the default Coords.

CoordsSigma**Type**

Float List

Unit

Angstrom

Description

Sigma values (one per Cartesian axis) for a Gauss distribution of the initial coordinates. Can only be used together with Coords.

DeviationAngle**Type**

Float

Default value

0.0

Unit

Degree

Description

Randomly tilt the shooting direction up to this angle away from the VelocityDirection vector.

Energy**Type**

Float

Unit

Hartree

Description

Initial kinetic energy of the molecule in the shooting direction.

EnergySigma**Type**

Float

Default value

0.0

Unit

Hartree

Description

Sigma value for the Gauss distribution of the initial kinetic energy around the specified value.
Should only be used together with Energy.

FractionalCoords**Type**

Float List

Description

Place molecules at or around the specified fractional coordinates in the main system's lattice. For non-periodic dimensions a Cartesian value in Angstrom is expected. This setting is ignored if [Coords] or [CoordsBox] is used.

FractionalCoordsBox**Type**

Float List

Description

Place molecules at random locations inside the box specified as fractional coordinates in the main system's lattice.

Coordinates of the box corners are specified as: Xmin, Xmax, Ymin, Ymax, Zmin, Zmax.

For non-periodic dimensions the Cartesian value in Angstrom is expected.

This setting is ignored if [Coords], [CoordsBox], or [FractionalCoords] is used.

FractionalCoordsSigma**Type**

Float List

Description

Sigma values (one per axis) for a Gauss distribution of the initial coordinates. For non-periodic dimensions the Cartesian value in Angstrom is expected. Can only be used together with FractionalCoords.

Frequency**Type**

Integer

Default value

0

Description

A molecule is added every [Frequency] steps after the StartStep.

There is never a molecule added at step 0.

MinDistance**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Keep the minimal distance to other atoms of the system when adding the molecule.

MoleFraction**Type**

Float

Description

Defines a mixture to be deposited using one AddMolecules block per component.

AMS will randomly alternate between any guns that have MoleFraction set. These need to all have the same settings for StartStep, StopStep and Frequency. Any additional AddMolecules blocks without MoleFraction will remain completely independent.

NumAttempts**Type**

Integer

Default value

10

Description

Try adding the molecule up to the specified number of times or until the MinDistance constraint is satisfied. If all attempts fail a message will be printed and the simulation will continue normally.

Rotate**Type**

Bool

Default value

No

Description

Rotate the molecule randomly before adding it to the system.

StartStep**Type**

Integer

Default value

0

Description

Step number when the first molecule should be added. After that, molecules are added every Frequency steps.

For example, if StartStep=99 and Frequency=100 then a molecule will be added at steps 99, 199, 299, etc...

No molecule will be added at step 0, so if StartStep=0 the first molecule is added at the step number equal to [Frequency].

StopStep**Type**

Integer

Description

Do not add this molecule after the specified step.

System**Type**

String

Description

String ID of the [System] that will be added with this 'gun'.

The lattice specified with this System is ignored and the main system's lattice is used instead.

AMSinput adds the system at the coordinates of the System (thus setting Coords to the center of the System).

Temperature**Type**

Float

Unit

Kelvin

Description

Initial energy of the molecule in the shooting direction will correspond to the given temperature.

TemperatureSigma**Type**

Float

Default value

0.0

Unit

Kelvin

Description

Sigma value for the Gauss distribution of the initial temperature the specified value. Should only be used together with Temperature.

Velocity**Type**

Float

Unit

Angstrom/fs

Description

Initial velocity of the molecule in the shooting direction.

VelocityDirection**Type**

Float List

Description

Velocity direction vector for aimed shooting. It will be random if not specified.

In AMSinput add one or two atoms (which may be dummies).

One atom: use vector from center of the system to add to that atom.

Two atoms: use vector from the first to the second atom.

VelocitySigma**Type**

Float

Default value

0.0

Unit

Angstrom/fs

Description

Sigma value for the Gauss distribution of the initial velocity around the specified value. Should only be used together with Velocity.

ApplyForce**Type**

Block

Recurring

True

Description

The ApplyForce keyword can be used to apply an arbitrary constant force to a certain (subgroups of) atoms in the system

Force**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Hartree/Bohr

Description

Defines the constant force vector

PerAtom**Type**

Bool

Default value

No

Description

If enabled, the Force vector is applied separately to every atom in the selected Region, so that the total net force on the Region equals the value of Force times the number of atoms in Region. This was the behavior of ApplyForce in AMS2022. By default, with PerAtom disabled, the Force vector defines the total net force on the Region, so the force applied to each atom equals the value of Force divided by the number of atoms in Region.

Region**Type**

String

Recurring

True

Description

Apply the constant force to all atoms in this region.

ApplyVelocity**Type**

Block

Recurring

True

Description

The ApplyVelocity keyword can be used to move an arbitrary group of atoms in the system with a constant net velocity

Components**Type**

Multiple Choice

Default value

XY

Options

[X, Y, Z, XY, YZ, XZ, XYZ]

Description

Select which components of the Velocity vector are used to set the corresponding components of the net velocity of the specified set of atoms. Any other components of Velocity are ignored and the motion of the selected atoms in those directions is unaffected by ApplyVelocity.

Region**Type**

String

Recurring

True

Description

Applies the defined velocity to all atoms in this region.

Velocity**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Angstrom/fs

Recurring

False

Description

The constant velocity that will be applied to the specified atoms.

Barostat**Type**

Block

Description

This block allows to specify the use of a barostat during the simulation.

BulkModulus**Type**

Float

Default value

2200000000.0

Unit

Pascal

Description

An estimate of the bulk modulus (inverse compressibility) of the system for the Berendsen barostat.

This is only used to make Tau correspond to the true observed relaxation time constant. Values are commonly on the order of 10-100 GPa (1e10 to 1e11) for solids and 1 GPa (1e9) for liquids (2.2e9 for water). Use 1e9 to match the behavior of standalone ReaxFF.

ConstantVolume**Type**

Bool

Default value

No

Description

Keep the volume constant while allowing the box shape to change.

This is currently supported only by the MTK barostat.

Duration

Type

Integer List

Description

Specifies how many steps should a transition from a particular pressure to the next one in sequence take.

Equal**Type**

Multiple Choice

Default value

None

Options

[None, XYZ, XY, YZ, XZ]

Description

Enforce equal scaling of the selected set of dimensions. They will be barostatted as one dimension according to the average pressure over the components.

Pressure**Type**

Float List

Unit

Pascal

Description

Specifies the target pressure.

You can specify multiple pressures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one p to the next p (using a linear ramp).

Scale**Type**

Multiple Choice

Default value

XYZ

Options

[XYZ, Shape, X, Y, Z, XY, YZ, XZ]

Description

Dimensions that should be scaled by the barostat to maintain pressure. Selecting Shape means that all three dimensions and also all the cell angles are allowed to change.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

Specifies the time constant of the barostat.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, MTK]

GUI name

Barostat

Description

Selects the type of the barostat.

BinLog**Type**

Block

Description

This block controls writing the BinLog section in ams.rkf, which contains the selected MD state scalars and tensors from every MD step. No per-atom data is written. If you need the per-atom data then you can set the sampling frequency to 1 and get the required data from the MDHistory section. The tensors are written per component, that is, the pressure tensor is written as six variables: PressureTensor_xx, PressureTensor_yy, etc.. To reduce the file size, all data is written in blocks.

BiasEnergy**Type**

Bool

Default value

No

Description

Write the CVHD bias energy at every time step.

BoostFactor**Type**

Bool

Default value

No

Description

Write the CVHD boost factor at every time step.

ConservedEnergy**Type**

Bool

Default value

No

Description

Write the conserved energy value at every time step.

Density**Type**

Bool

Default value

No

Description

Write the density at every time step.

DipoleMoment**Type**

Bool

Default value

No

GUI name

Dipole moment binlog

Description

Write the dipole moment at every time step. Each component of the tensor is written in its own variable.

Hypertime**Type**

Bool

Default value

No

Description

Write the CVHD hypertime at every time step.

KineticEnergy**Type**

Bool

Default value

No

Description

Write the kinetic energy value at every time step.

MaxBiasEnergy**Type**

Bool

Default value

No

Description

Write the max CVHD bias energy at every time step.

MaxBoostFactor

Type

Bool

Default value

No

Description

Write the max CVHD boost factor at every time step.

PotentialEnergy**Type**

Bool

Default value

No

Description

Write the potential energy value at every time step.

Pressure**Type**

Bool

Default value

No

Description

Write the pressure at every time step.

PressureTensor**Type**

Bool

Default value

No

GUI name

Pressure tensor binlog

Description

Write the pressure tensor in Voigt notation at every time step. Each component of the tensor is written in its own variable.

Step**Type**

Bool

Default value

No

Description

Write the step index during the simulation at every time step.

Temperature**Type**

Bool

Default value

No

Description

Write the temperature at every time step.

Time**Type**

Bool

Default value

No

GUI name

Time (fs) binlog

Description

Write the simulation time (fs) at every time step.

TotalEnergy**Type**

Bool

Default value

No

Description

Write the total energy value at every time step.

Volume**Type**

Bool

Default value

No

Description

Write the simulation cell volume, area or length, depending on the system periodicity, at every time step.

BondBoost**Type**

Block

Recurring

True

Description

Forced reaction (bond boost) definitions. Multiple BondBoost blocks may be specified, which will be treated independently.

Chain**Type**

Block

Description

Specifications of a chain of atoms. When a chain is detected the distance restraints will be activated. No other chain of this type will be detected while any restraints for this chain is active.

AtomNames

Type

String

Description

Atom names specifying the chain. An atom name can optionally be followed by '@' and a region name, in this case only atoms of this type from the given region will be matched. A leading '@' followed by a number indicates that this position in the chain must be occupied by the atom found earlier at the specified position in the chain. For example "O H N C @1" indicates that the last atom in the chain of the five atoms must be the first oxygen, thus defining a 4-membered ring. This is the only way to define a ring because implicit rings will not be detected. For example, "O H N C O" does not include rings.

MaxDistances**Type**

Float List

Unit

Angstrom

Description

Maximum distances for each pair of atoms in the chain. The number of distances must be one less than the number of AtomNames.

MinDistances**Type**

Float List

Unit

Angstrom

Description

Minimum distances for each pair of atoms in the chain. The number of distances must be one less than the number of AtomNames.

DistanceRestraint**Type**

String

Recurring

True

Description

Specify two atom indices followed by the target distance in Angstrom, the first parameter and, optionally, the profile type and the second parameter.

For periodic systems, restraints follow the minimum image convention.

Each atom index indicates a position of the corresponding atom in the AtomNames key. Currently recognized restraint profile types: Harmonic (default), Hyperbolic, Erf, GaussianWell. The profile name can optionally be prefixed by "OneSided-", thus making this particular restraint one-sided. A one-sided restraint will only push or pull the distance towards its target value depending on the sign of the difference between the initial and the target distance. For example, if the initial distance is larger than the target one then the restraint will only push the value down. This is especially useful with moving restraints because then the restraint will effectively disappear as soon as the system crosses a barrier along the reaction path. By default, the restraints are two-sided, which means they will try to keep the distance between the two specified atoms near the (possibly moving) target value. For moving restraints, this

means that after crossing a reaction barrier the system will slide slowly towards products held back by the restraints.

The first parameter is the force constant for the Harmonic, Hyperbolic, and Erf profiles, or well depth for GaussianWell. The second parameter is the asymptotic force value $F(\text{Inf})$ for Hyperbolic and Erf profiles, or the factor before x^2 in the exponent for GaussianWell.

Note: the GaussianWell restraint should be used with the Moving flag.

Moving**Type**

Bool

Default value

No

GUI name

Move restraint

Description

Move the restraints created with this BondBoost. The restraint value will start at the current coordinate's value and will move towards the optimum during the restraint's lifetime. The increment is calculated from the initial deviation and the [NSteps] parameter.

This feature should be used with the GaussianWell restraint types.

NSteps**Type**

Integer

GUI name

Boost lifetime

Description

Number of steps the restraints will remain active until removed. Atoms participating in one reaction are not available for the given number of steps.

NumInstances**Type**

Integer

Default value

1

GUI name

Number of instances

Description

Number of reactions of this type taking place simultaneously.

Units**Type**

Multiple Choice

Default value

Default

Options

[Default, MD]

GUI name

Restr. parameter units

Description

Change energy, force and force constant units in the DistanceRestraint key from the default (atomic units) to those often used in the MD community (based on kcal/mol and Angstrom). Units for the optimum distances are not affected.

CRESTMTD**Type**

Block

GUI name

CREST_MTD

Description

Input for CREST metadynamics simulation.

AddEnergy**Type**

Bool

Default value

No

Description

Add the bias energy to the potential energy (to match the gradients)

GaussianScaling**Type**

Block

Description

Options for gradual introduction of the Gaussians

ScaleGaussians**Type**

Bool

Default value

Yes

Description

Introduce the Gaussians gradually, using a scaling function

ScalingSlope**Type**

Float

Default value

0.03

Description

Slope of the scaling function for the Gaussians with respect to time

Height**Type**

Float

Unit

Hartree

Description

The height of the Gaussians added

NGaussiansMax**Type**

Integer

Description

Maximum number of Gaussians stored

NSteps**Type**

Integer

Description

Interval of Gaussian placement

Region**Type**

String

Default value

*

Description

Restrict the range of atoms for RMSD calculation to the specified region.

RestartFile**Type**

String

Description

Filename for file from which to read data on Gaussians placed previously.

Width**Type**

Float

Unit

Bohr

Description

The width of the Gaussians added in terms of the RMSD

CVHD**Type**

Block

Recurring

True

GUI name

CVHD

Description

Input for the Collective Variable-driven HyperDynamics (CVHD).

Bias**Type**

Block

Description

The bias is built from a series of Gaussian peaks deposited on the collective variable axis every [Frequency] steps during MD. Each peak is characterized by its (possibly damped) height and the RMS width (standard deviation).

DampingTemp**Type**

Float

Default value

0.0

Unit

Kelvin

GUI name

Bias damping T

Description

During well-tempered hyperdynamics the height of the added bias is scaled down with an $\exp(-E/kT)$ factor [PhysRevLett 100, 020603 (2008)], where E is the current value of the bias at the given CV value and T is the damping temperature DampingTemp. If DampingTemp is zero then no damping is applied.

Delta**Type**

Float

Description

Standard deviation parameter of the Gaussian bias peak.

Height**Type**

Float

Unit

Hartree

Description

Height of the Gaussian bias peak.

ColVarBB**Type**

Block

Recurring

True

GUI name

Collective Variable

Description

Description of a bond-breaking collective variable (CV) as described in [Bal & Neyts, JCTC, 11 (2015)]. A collective variable may consist of multiple ColVar blocks.

at1**Type**

Block

Description

Specifies the first bonded atom in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection of bonded atoms to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the first atom of the bond. The name must be as it appears in the System block. That is, if the atom name contains an extension (e.g C.1) then the full name including the extension must be used here.

at2**Type**

Block

Description

Specifies the second bonded atom in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection of bonded atoms to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the second atom of the bond. The value is allowed to be the same as [at1], in which case bonds between atoms of the same type will be included.

cutoff**Type**

Float

Default value

0.3

GUI name

Bond order cutoff

Description

Bond order cutoff. Bonds with BO below this value are ignored when creating the initial bond list for the CV. The bond list does not change during lifetime of the variable even if some bond orders drop below the cutoff.

p**Type**

Integer

Default value

6

GUI name

Exponent p

Description

Exponent value p used to calculate the p-norm for this CV.

rmax**Type**

Float

Unit

Angstrom

GUI name

R max

Description

Max bond distance parameter Rmax used for calculating the CV. It should be close to the transition-state distance for the corresponding bond.

rmin**Type**

Float

Unit

Angstrom

GUI name

R min

Description

Min bond distance parameter Rmin used for calculating the CV. It should be close to equilibrium distance for the corresponding bond.

ColVarBM**Type**

Block

Recurring

True

GUI name

Collective Variable

Description

Description of a bond-making collective variable (CV). A collective variable may consist of multiple ColVar blocks.

at1**Type**

Block

Description

Specifies selection criteria for the first atom of a pair in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the first atom of the pair. The name must be as it appears in the System block. That is, if the atom name contains an extension (e.g C.1) then the full name including the extension must be used here.

at2**Type**

Block

Description

Specifies selection criteria for the second atom of a pair in the collective variable.

region**Type**

String

Default value

*

Description

Restrict the selection to a specific region. If this is not set, atoms anywhere in the system will be selected.

symbol**Type**

String

Description

Atom type name of the second atom of the pair. The value is allowed to be the same as [at1], in which case pairs of atoms of the same type will be included.

cutoff**Type**

Float

Default value

0.3

GUI name

Bond order cutoff

Description

Bond order cutoff. Bonds with BO above this value are ignored when creating the initial atom-pair list for the CV. The list does not change during lifetime of the variable even if some bond orders rise above the cutoff.

p**Type**

Integer

Default value

6

GUI name

Exponent p

Description

Exponent value p used to calculate the p-norm for this CV.

rmax**Type**

Float

Unit

Angstrom

GUI name

R max

Description

Max bond distance parameter Rmax used for calculating the CV. It should be much larger than the corresponding Rmin.

rmin**Type**

Float

Unit

Angstrom

GUI name

R min

Description

Min bond distance parameter Rmin used for calculating the CV. It should be close to the transition-state distance for the corresponding bond.

Frequency**Type**

Integer

Description

Frequency of adding a new bias peak, in steps.

New bias is deposited every [Frequency] steps after [StartStep] if the following conditions are satisfied:

the current CV value is less than 0.9 (to avoid creating barriers at the transition state),

the step number is greater than or equal to [StartStep], and

the step number is less than or equal to [StopStep].

MaxEvents**Type**

Integer

Default value

0

Description

Max number of events to allow during dynamics. When this number is reached no new bias will be added for this input block.

StartStep**Type**

Integer

Description

If this key is specified, the first bias will be deposited at this step. Otherwise, the first bias peak is added at the step number equal to the Frequency parameter. The bias is never deposited at step 0.

StopStep**Type**

Integer

Description

No bias will be deposited after the specified step. The already deposited bias will continue to be applied until the reaction event occurs. After that no new CVHD will be started. By default, the CVHD runs for the whole duration of the MD calculation.

WaitSteps**Type**

Integer

Description

If the CV value becomes equal to 1 and remains at this value for this many steps then the reaction event is considered having taken place. After this, the collective variable will be reset and the bias will be removed.

CalcPressure**Type**

Bool

Default value

No

GUI name

Calculate pressure

Description

Calculate the pressure in periodic systems.

This may be computationally expensive for some engines that require numerical differentiation.

Some other engines can calculate the pressure for negligible additional cost and will always do so, even if this option is disabled.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps.

Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

CopyRestartTrajectory**Type**

Bool

Default value

No

Description

If the keyword Restart is present, the content of the restartfile is copied to the ams.rkf file.

CosineShear

Type

Block

Description

Apply an external acceleration to all atoms of a fluid using a periodic (cosine) function along a selected coordinate axis. This induces a periodic shear flow profile which can be used to determine the viscosity.

Acceleration**Type**

Float

Default value

5e-06

UnitAngstrom/fs²**Description**

Amplitude of the applied cosine shear acceleration profile. The default value should be a rough first guess for water and it needs to be adjusted by experimentation for other systems.

Enabled**Type**

Bool

Default value

No

GUI name

Enable cosine shear

Description

Apply a cosine shear acceleration profile for a NEMD calculation of viscosity.

FlowDirection**Type**

Float List

Default value

[1.0, 0.0, 0.0]

Description

The direction in which to apply the shear acceleration, in Cartesian coordinates. The magnitude of this vector is ignored (AMS will normalize it internally). FlowDirection has to be perpendicular to ProfileAxis.

ProfileAxis**Type**

Multiple Choice

Default value

Z

Options

[X, Y, Z]

Description

The Cartesian coordinate axis along which the cosine wave runs

Deformation**Type**

Block

Recurring

True

Description

Deform the periodic lattice of the system during the simulation.

LatticeVelocity**Type**

Non-standard block

Description

Velocity of individual lattice vector components in Angstrom/fs. The format is identical to the System%Lattice block. For Type Sine and Cosine, this defines the maximum velocity (at the inflection point).

LengthRate**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Description

Relative rate of change of each lattice vector per step.

LengthVelocity**Type**

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Angstrom/fs

Description

Change the length of each lattice vector with this velocity. With Type=Exponential, Length-Velocity is divided by the current lattice vector lengths on StartStep to determine a Length-Rate, which is then applied on all subsequent steps. For Type Sine and Cosine, this defines the maximum velocity (at the inflection point).

Period**Type**

Float

Unit

Femtoseconds

Description

Period of oscillation for Type Sine and Cosine.

ScaleAtoms**Type**

Bool

Default value

Yes

Description

Scale the atomic positions together with the lattice vectors. Disable this to deform only the lattice, keeping the coordinates of atoms unchanged.

StartStep**Type**

Integer

Default value

1

Description

First step at which the deformation will be applied.

StopStep**Type**

Integer

Default value

0

Description

Last step at which the deformation will be applied. If unset or zero, nSteps will be used instead.

StrainRate**Type**

Non-standard block

Description

Strain rate matrix to be applied on every step. The format is identical to the System%Lattice block.

TargetDensity**Type**

Float

Unit

g/cm3

Description

Target density of the system to be achieved by StopStep by isotropically rescaling the lattice.

This can only be used for 3D-periodic systems. Note that the selected Type determines how the lengths of the lattice vectors will evolve (linearly or as a sine/cosine), the evolution of the volume or density is thus necessarily non-linear.

TargetLattice**Type**

Non-standard block

Description

Target lattice vectors to be achieved by StopStep. The format is identical to the System%Lattice block.

TargetLength

Type

Float List

Default value

[0.0, 0.0, 0.0]

Unit

Angstrom

Description

Target lengths of each lattice vector to be achieved by StopStep. The number of values should equal the periodicity of the system. If a value is zero, the corresponding lattice vector will not be modified.

Type**Type**

Multiple Choice

Default value

Linear

Options

[Linear, Exponential, Sine, Cosine]

Description

Function defining the time dependence of the deformed lattice parameters.

Linear increments the lattice parameters by the same absolute amount every timestep. Exponential multiplies the lattice parameters by the same factor every timestep. Only StrainRate, LengthRate, and LengthVelocity are supported for Type=Exponential. Sine deforms the system from the starting lattice to TargetLattice/TargetLength and then by the same amount to the opposite direction, while Cosine deforms the system from the starting lattice to the target and back.

Gravity**Type**

Block

Description

Apply a constant acceleration in -z.

Acceleration**Type**

Float

Default value

0.0

UnitAngstrom/fs²**Description**

Magnitude of the applied acceleration.

HeatExchange**Type**

Block

Recurring

True

GUI name

Heat exchange

Description

Input for the heat-exchange non-equilibrium MD (T-NEMD).

HeatingRate**Type**

Float

Unit

Hartree/fs

Description

Rate at which the energy is added to the Source and removed from the Sink. A heating rate of 1 Hartree/fs equals to about 0.00436 Watt of power being transferred through the system.

Method**Type**

Multiple Choice

Default value

Simple

Options

[Simple, HEX, eHEX]

Description

Heat exchange method used.

Simple: kinetic energy of the atoms of the source and sink regions is modified irrespective of that of the center of mass (CoM) of the region (recommended for solids).

HEX: kinetic energy of the atoms of these regions is modified keeping that of the corresponding CoM constant.

eHEX: an enhanced version of HEX that conserves the total energy better (recommended for gases and liquids).

Sink**Type**

Block

Description

Defines the heat sink region (where the heat will be removed).

AtomList**Type**

Integer List

GUI name

Sink region

Description

The atoms that are part of the sink.

This key is ignored if the [Box] block or [Region] key is present.

Box

Type

Block

Description

Part of the simulation box (in fractional cell coordinates) defining the heat sink. If this block is specified, then by default, the whole box in each of the three dimensions is used, which usually does not make much sense. Normally, you will want to set the bounds along one of the axes.

Amax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the third axis.

Cmin

Type

Float

Default value

0.0

Description

Coordinate of the lower bound along the third axis.

Region**Type**

String

GUI name

Sink region

Description

The region that is the sink.

This key is ignored if the [Box] block is present.

Source**Type**

Block

Description

Defines the heat source region (where the heat will be added).

AtomList**Type**

Integer List

GUI name

Source region

Description

The atoms that are part of the source.

This key is ignored if the [Box] block or [Region] key is present.

Box**Type**

Block

Description

Part of the simulation box (in fractional cell coordinates) defining the heat source. If this block is specified, then by default, the whole box in each of the three dimensions is used, which usually does not make much sense. Normally, you will want to set the bounds along one of the axes. This block is mutually exclusive with the FirstAtom/LastAtom setting.

Amax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Default value

1.0

Description

Coordinate of the upper bound along the third axis.

Cmin**Type**

Float

Default value

0.0

Description

Coordinate of the lower bound along the third axis.

Region**Type**

String

GUI name

Source region

Description

The region that is the source.

This key is ignored if the [Box] block is present.

StartStep**Type**

Integer

Default value

0

Description

Index of the MD step at which the heat exchange will start.

StopStep**Type**

Integer

Description

Index of the MD step at which the heat exchange will stop.

InitialVelocities**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod**Type**

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature

Type

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type**Type**

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values**Type**

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

MovingRestraints**Type**

Block

Recurring

True

Description

Define a set of moving restraints.

Change**Type**

Multiple Choice

Default value

Linear

Options

[Linear, Sine, Cosine]

GUI name

Move type

Description

Type of function defining how the target restraint value will change over time:

Linear - linearly between the StartValue and EndValue.

Sine - oscillating around StartValue with an amplitude equal to the difference between EndValue and StartValue.

Cosine - oscillating between StartValue and EndValue.

Distance**Type**

Block

Recurring

True

Description

Define a distance restraint between pair of atoms. For linear-type

Atom1**Type**

Integer

Description

First atom of the distance restraint.

Atom2**Type**

Integer

Description

Second atom of the distance restraint.

EndValue**Type**

Float

Unit

Angstrom

Description

Linear: final target distance.

Sine: target distance at 1/4 of the period.

Cosine: target distance at 1/2 of the period.

StartValue

Type

Float

Unit

Angstrom

Description

Initial target distance.

Erf

Type

Block

Description

Define parameters for the Int(erf) restraint potential $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi}))$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce.

ForceConstant

Type

Float

Default value

0.5

Unit

Hartree/Bohr²

GUI name

Erf force constant

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce

Type

Float

Default value

0.05

Unit

Hartree/Bohr

GUI name

Erf F(Inf)

Description

Asymptotic value of the force at the infinity.

GaussianWell

Type

Block

Description

Define parameters in the Gaussian well restraint potential $V = -\text{WellDepth} * \exp(-\text{Sigma} * (r - r_0)^2)$.

Sigma**Type**

Float

Default value

1.0

Unit1/Bohr²**GUI name**

Gaussian well sigma

Description

Sigma parameter in the potential expression.

WellDepth**Type**

Float

Default value

1.0

Unit

Hartree

GUI name

Gaussian well depth

Description

WellDepth parameter in the potential expression.

Harmonic**Type**

Block

DescriptionDefine parameters for the harmonic potential $V=0.5*FC*(r-r_0)^2$.**ForceConstant****Type**

Float

Default value

0.5

UnitHartree/Bohr²**GUI name**

Harmonic force constant

Description

The FC parameter of the harmonic potential.

Hyperbolic**Type**

Block

Description

Define parameters for the hyperbolic restraint potential $V = \alpha * (\sqrt{1 + \beta * x^2} - 1)$.
 The alpha and beta parameters are computed from the user-defined ForceConstant and Max-Force: $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²**GUI name**

Hyperbolic force constant

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

GUI name

Hyperbolic F(Inf)

Description

Asymptotic value of the force at the infinity.

Name**Type**

String

GUI name

Name

Description

Optional name to be used for plotting.

Period**Type**

Float

Default value

0.0

Unit

Femtoseconds

Description

Period of oscillation for Sine and Cosine change types.

RestraintType

Type

Multiple Choice

Default value

None

Options

[None, Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Restraint type

Description

Select type of the moving restraint profile. The force for Hyperbolic and Erf is bounded by a user-defined value, the latter converging to it faster than the former. The GaussianWell has a finite depth so it is suitable for cases when crossing a high reaction barrier is not desirable.

StartStep**Type**

Integer

Default value

1

GUI name

Start step

Description

First step number at which the restraints will be applied.

StopStep**Type**

Integer

Default value

0

GUI name

End step

Description

Last step number at which the restraints will be applied.

NSteps**Type**

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

Plumed**Type**

Block

Description

Input for PLUMED (version 2.5.0). The parallel option is still experimental.

Input**Type**

Non-standard block

Description

Input for PLUMED. Contents of this block is passed to PLUMED as is.

Parallel**Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

Preserve**Type**

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and 3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass**Type**

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum**Type**

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

System**Type**

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities**Type**

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

ReactionBoost**Type**

Block

GUI name

Reaction Boost

Description

Define a series of transitions between different states of the system.

Each transition is defined by a TargetSystem and by a set of restraints that force the transition.

BondBreakingRestrains**Type**

Block

Description

Define parameters for moving restraints that are added for pairs of atoms that become disconnected during the transition.

It is intended to make sure the corresponding bonds get broken, although this may not always be required because forming other bonds will likely get these bonds broken.

Erf**Type**

Block

Description

Define parameters for the Int(erf) potential $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi}))$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce.

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

GaussianWell**Type**

Block

DescriptionDefine parameters in the Gaussian well potential $V = -\text{WellDepth} \cdot \exp(-\text{Sigma} \cdot (r - r_0)^2)$.**Sigma****Type**

Float

Default value

1.0

Unit1/Bohr²**Description**

Sigma parameter in the potential expression.

WellDepth**Type**

Float

Default value

1.0

Unit

Hartree

Description

WellDepth parameter in the potential expression.

Harmonic**Type**

Block

DescriptionDefine parameters for the harmonic potential $V = 0.5 \cdot \text{FC} \cdot (r - r_0)^2$.**ForceConstant****Type**

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The FC parameter of the harmonic potential.

Hyperbolic**Type**

Block

Description

Define parameters for the hyperbolic potential $V = \alpha * (\sqrt{1 + \beta * x^2} - 1)$. The α and β parameters are computed from the user-defined ForceConstant and MaxForce: $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

Taper**Type**

Block

Description**Enabled****Type**

Bool

Default value

No

GUI name

Tapering

Description

Enable tapering of the restraint potential and force between the given range of bond distances. A 7-th order tapering function on the actual (not target!) distance will be used. The MaxDistance must be greater than MinDistance.

MaxDistance**Type**

Float

Default value

0.0

Unit

Angstrom

GUI name

End tapering at

Description

Bond length at which the restraint potential and force decays to zero.

MinDistance**Type**

Float

Default value

0.0

Unit

Angstrom

GUI name

Start tapering at

Description

Bond length at which the restraint potential and force will start decaying to zero.

Type**Type**

Multiple Choice

Default value

Erf

Options

[None, Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Bond breaking restraints

Description

Select type of the moving restraint profile.

Harmonic: $V=0.5*FC*(r-r_0)^2$

Hyperbolic: $V=\alpha*(\sqrt{1 + \beta*x^2} - 1)$

Erf: $V = \alpha*(\beta*x*\text{erf}(\beta*x) + (\exp(-(\beta*x)^2) - 1)/\sqrt{\pi}))$

GaussianWell: $V=\text{WellDepth}*(1-\exp(-\text{Sigma}*(r-r_0)^2))$

Here $\beta=\text{ForceConstant}/\text{MaxForce}$, $\alpha=\text{MaxForce}/\beta$.

The force for Hyperbolic and Erf is bounded by a user-defined value, the latter converging to it faster than the former.

The GaussianWell has a finite depth so it is suitable for cases when crossing a high reaction barrier is not desirable.

Moving restraints are added for pairs of atoms that become disconnected during the transition.

It is intended to make sure the corresponding bonds get broken, although this may not always be required because forming other bonds will likely get these bonds broken.

BondMakingRestrains

Type

Block

Description

Define parameters for moving restraints that are added for pairs of atoms that become connected during the transition.

It is intended to make sure the bonds are created as required.

Erf

Type

Block

Description

Define parameters for the Int(erf) potential $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi}))$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce.

ForceConstant

Type

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce

Type

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

GaussianWell

Type

Block

Description

Define parameters in the Gaussian well potential $V = -\text{WellDepth} \cdot \exp(-\text{Sigma} \cdot (r-r_0)^2)$.

Sigma**Type**

Float

Default value

1.0

Unit

1/Bohr²

Description

Sigma parameter in the potential expression.

WellDepth**Type**

Float

Default value

1.0

Unit

Hartree

Description

WellDepth parameter in the potential expression.

Harmonic**Type**

Block

Description

Define parameters for the harmonic potential $V = 0.5 \cdot \text{FC} \cdot (r-r_0)^2$.

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The FC parameter of the harmonic potential.

Hyperbolic**Type**

Block

Description

Define parameters for the hyperbolic potential $V = \alpha \cdot (\sqrt{1 + \beta \cdot x^2} - 1)$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce: $\beta = \text{ForceConstant}/\text{MaxForce}$, $\alpha = \text{MaxForce}/\beta$

ForceConstant

Type

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

Type**Type**

Multiple Choice

Default value

Erf

Options

[None, Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Bond making restraints

Description

Select type of the moving restraint profile.

Harmonic: $V = 0.5 \cdot FC \cdot (r - r_0)^2$

Hyperbolic: $V = \alpha \cdot (\sqrt{1 + \beta \cdot x^2} - 1)$

Erf: $V = \alpha \cdot (\beta \cdot x \cdot \text{erf}(\beta \cdot x) + (\exp(-(\beta \cdot x)^2) - 1) / \sqrt{\pi}))$

GaussianWell: $V = -\text{WellDepth} \cdot \exp(-\text{Sigma} \cdot (r - r_0)^2)$

Here $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$.

The force for Hyperbolic and Erf is bounded by a user-defined value.

The GaussianWell has a finite depth so it is suitable for cases when crossing a high reaction barrier is not desirable.

Moving restraints are added for pairs of atoms that become connected during the transition.

It is intended to make sure the bonds are created as required.

BondedRestraints**Type**

Block

Description

Define parameters for bonded restraints. A bonded restraint is added for each pair of atoms that are bonded both in the current and in the final state.

It is intended to make sure they remain bonded during simulation.

Harmonic**Type**

Block

Description

Define parameters for the harmonic potential $V=0.5*FC*(r-r_0)^2$.

ForceConstant**Type**

Float

Default value

0.5

Unit

Hartree/Bohr²

Description

The FC parameter of the harmonic potential.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Harmonic]

GUI name

Bonded restraints

Description

Select type of the bonded restraints:

Harmonic: $V=0.5*FC*(r-r_0)^2$

A bonded restraint is added for each pair of atoms that are bonded both in the current and in the final state.

It is intended to make sure they remain bonded during simulation.

Change**Type**

Multiple Choice

Default value

TargetCoordinate

Options

[TargetCoordinate, Force, LogForce]

GUI name

Move type

Description

Select what to change during dynamics.

By default, once the restraints are switched on, AMS will change the restraint's target coordinate towards its final value.

If the [Force] or [LogForce] option is selected then the target coordinate is set to its final value immediately and instead the restraint force is gradually scaled from 0 to 1. The scaling is either linear (Force) or logarithmic (LogForce).

InitialFraction**Type**

Float

Default value

0.0

Description

Initial fraction of the boost variable.

At the first boosting step, the restraint's target value (or force or log(force)) is equal to InitialFraction + 1/NSteps.

InterEquilibrationSteps**Type**

Integer

Default value

0

Description

Number of equilibration steps after reaching a target before setting up restraints for the next one.

MinBondChange**Type**

Float

Default value

1.0

Unit

Bohr

Description

Minimal change in the distance for an individual restraint to be considered bond-breaking/making vs bonded.

MinBondStrength**Type**

Float

Default value

0.5

Description

Minimum strength (usually ranges from 0 to 1) for a bond to be considered.

NSteps

Type

Integer

Default value

500

GUI name

Steps per target

Description

Number of steps per target the restraints should be active for.

NonBondedRestrains**Type**

Block

Description

Define parameters for non-bonded restraints. A non-bonded restraint is added for each pair of atoms that are bonded neither in the current nor in the final state.

It is intended to keep them from forming a bond unintentionally. They are represented by a repulsive potential

Exponential**Type**

Block

Description

Define parameters for the repulsive potential $V = \text{Epsilon} \cdot \exp(-\text{Sigma} \cdot r)$.

Epsilon**Type**

Float

Default value

1.0

Unit

Hartree

Description

Epsilon parameter in the repulsive potential expression.

Sigma**Type**

Float

Default value

1.0

Unit

1/Bohr

Description

Sigma parameter in the repulsive potential expression.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Exponential]

GUI name

Non-bonded restraints

Description

Select type of the non-bonded restraints:

Exponential: $V = \text{Epsilon} * \exp(-\text{Sigma} * r)$

A non-bonded restraint is added for each pair of atoms that are bonded neither in the current nor in the final state.

It is intended to keep them from forming a bond unintentionally. They are represented by a repulsive potential.

PreEquilibrationSteps**Type**

Integer

Default value

0

Description

Number of steps before enabling the first set of restraints.

RMSDRestraint**Type**

Block

GUI name

RMSD restraint

Description

Define a static restraint that pulls each atom to its position in the target system, but in contrast to the individual restraints, the force for this one depends on the total mass-weighted root-mean-squared distance (RMSD) between the two structures.

Erf**Type**

Block

Description

Define parameters for the Int(erf) potential $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi}))$. The alpha and beta parameters are computed from the user-defined ForceConstant and MaxForce.

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²

Description

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

GaussianWell**Type**

Block

Description

Define parameters in the Gaussian well potential $V = -\text{WellDepth} \cdot \exp(-\text{Sigma} \cdot (r - r_0)^2)$.

Sigma**Type**

Float

Default value

1.0

Unit

1/Bohr²

Description

Sigma parameter in the potential expression.

WellDepth**Type**

Float

Default value

1.0

Unit

Hartree

Description

WellDepth parameter in the potential expression.

Harmonic**Type**

Block

Description

Define parameters for the harmonic potential $V = 0.5 \cdot \text{FC} \cdot (r - r_0)^2$.

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The FC parameter of the harmonic potential.

Hyperbolic**Type**

Block

Description

Define parameters for the hyperbolic potential $V = \alpha * (\sqrt{1 + \beta * x^2} - 1)$. The α and β parameters are computed from the user-defined ForceConstant and MaxForce: $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$

ForceConstant**Type**

Float

Default value

0.5

UnitHartree/Bohr²**Description**

The force constant (second derivative of the potential) at the optimum point.

MaxForce**Type**

Float

Default value

0.05

Unit

Hartree/Bohr

Description

Asymptotic value of the force at the infinity.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Type

Description

Select type of the RMSD restraint profile:

Harmonic: $V = 0.5 * FC * (r - r_0)^2$

Hyperbolic: $V = \alpha * (\sqrt{1 + \beta * x^2} - 1)$

Erf: $V = \alpha * (\beta * x * \text{erf}(\beta * x) + (\exp(-(\beta * x)^2) - 1) / \sqrt{\pi})$,

GaussianWell: $V = -\text{WellDepth} * \exp(-\text{Sigma} * (r - r_0)^2)$

Here $\beta = \text{ForceConstant} / \text{MaxForce}$, $\alpha = \text{MaxForce} / \beta$.

The Harmonic profile can be problematic at large deviations as it may result in large forces. The force for Hyperbolic and Erf is bounded by a user-defined value. The GaussianWell has a finite depth so it is suitable for cases when crossing a high reaction barrier is not desirable.

Region

Type

String

Default value

*

GUI name

Region

Description

Region to which the restraints should be limited.

TargetSystem

Type

String

Recurring

True

GUI name

Target system

Description

The target system's name for this transition. Multiple targets can be specified to request multiple transitions in one simulation.

Note that only the lattice and the atomic coordinates of the target system are used and other properties (bonds, charge, etc.) are ignored. The target system's lattice is used only to determine connections and it cannot be restrained.

Type

Type

Multiple Choice

Default value

None

Options

[None, Pair, RMSD]

GUI name

Restraint set type

Description

Reaction Boost uses a series of transitions between different states of the system.

Each transition is defined by a TargetSystem and by a set of restraints that force the transition.

Select the type of the restraint set:

-None: no Reaction Boost

- Pair: use pair restraints
- RMSD: use RMSD restraints.

Pair restraints are defined per atom pair while the RMSD defines one collective restraint for all atoms and is thus suitable for very large systems.

The pair restraints are further divided into four sub-types: bonding, non-bonding, bond-breaking and bond-making. The sub-type of restraints for each pair is determined automatically depending on whether the two atoms are bonded in the initial/final state.

Parameters of the pair restraints are defined by [NonBondedRestrains], [BondedRestrains], [BondBreakingRestrains] and [BondMakingRestrains] blocks, while those of the RMSD restraint by the [RMSDRestraint] block.

Reactor

Type

Block

Recurring

True

Description

Define one phase of the nanoreactor. A reactor is a region of space surrounded by an elastic wall. Atoms inside the region are not affected. Atoms outside it will be pushed back with force depending on the [ForceConstant] and the [MassScaled] flag.

ForceConstant

Type

Float

GUI name

Reactor force constant

Description

Force constant of the reactor wall in Hartree/Bohr² (or Hartree/Bohr²/Dalton if [MassScaled] is true).

MassScaled

Type

Bool

Default value

Yes

GUI name

Scale force by mass

Description

If this flag is disabled the force on an atom outside of the reactor depends only on the atomic coordinates and the force constant. Otherwise, the force is also multiplied by the mass of the atom. This means that atoms at the same distance from the wall will receive the same accelerate due to the wall potential.

NSteps

Type

Integer

GUI name

Reactor lifetime

Description

Number of steps for which the reactor will remain active until disabled. The next reactor will be activated immediately after this. After the last reactor is disabled the cycle will repeat.

Radius**Type**

Float

Unit

Angstrom

GUI name

Reactor radius

Description

Radius of the reactor sphere.

ReflectiveWall**Type**

Block

Recurring

True

Description

Apply a reflective wall in space

Axis**Type**

Float List

Unit

Angstrom

Description

Defines the normal vector perpendicular to the plane of the reflective wall. Any particle moving in this direction will be reflected back.

Region**Type**

String

Recurring

True

Description

Apply the reflective wall to all atoms in this region.

Threshold**Type**

Float

Unit

Angstrom

Description

Defines the threshold value determining the position of the reflective wall. If the dot product

of a position of a particle with Axis exceeds Threshold, the particle will be reflected. This means that the plane of the wall passes through a point given by Axis times Threshold.

Remap

Type

Block

Description

Control periodic remapping (backtranslation) of atoms into the PBC box.

Range

Type

Multiple Choice

Default value

Auto

Options

[Auto, ZeroToOne, PlusMinusHalf, AroundCenter]

Description

Select the range of fractional coordinates to which atoms are remapped.

Auto: Use PlusMinusHalf if the geometrical center of the starting positions of all atoms lies between -0.25 and 0.25 in fractional coordinates in all periodic directions, ZeroToOne otherwise.

PlusMinusHalf: Remap atoms to lie between -0.5 and 0.5 in fractional coordinates.

ZeroToOne: Remap atoms to lie between 0 and 1 in fractional coordinates.

AroundCenter: Remap atoms to lie within 0.5 in fractional coordinates around the geometrical center of the starting positions of all atoms.

Type

Type

Multiple Choice

Default value

Atoms

Options

[None, Atoms]

Description

Select the method used to remap atoms into the unit cell.

None: Disable remapping completely.

Atoms: Remap any atoms that leave the unit cell.

RemoveMolecules

Type

Block

Recurring

True

GUI name

Remove molecules

Description

This block controls removal of molecules from the system. Multiple occurrences of this block are possible.

Formula**Type**

String

Description

Molecular formula of the molecules that should be removed from the system.

The order of elements in the formula is very important and the correct order is: C, H, all other elements in the strictly alphabetic order.

Element names are case-sensitive, spaces in the formula are not allowed. Digit '1' must be omitted.

Valid formula examples: C2H6O, H2O, O2S. Invalid formula examples: C2H5OH, H2O1, OH, SO2. Invalid formulas are silently ignored.

Use * to remove any molecule, which must be combined with SinkBox or SafeBox.

Frequency**Type**

Integer

Default value

0

Description

The specified molecules are removed every so many steps after the StartStep. There is never a molecule removed at step 0.

SafeBox**Type**

Block

Description

Part of the simulation box where molecules may not be removed. Only one of the SinkBox or SafeBox blocks may be present. If this block is present the molecule will not be removed if any of its atoms is within the box. For a periodic dimension it is given as a fraction of the simulation box (the full 0 to 1 range by default). For a non-periodic dimension it represents absolute Cartesian coordinates in Angstrom.

Amax**Type**

Float

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Description

Coordinate of the lower bound along the first axis.

Bmax

Type

Float

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Description

Coordinate of the upper bound along the third axis.

Cmin**Type**

Float

Description

Coordinate of the lower bound along the third axis.

FractionalCoordsBox**Type**

Float List

GUI name

Safe box

Description

Do not remove molecules that are (partly) inside the safe box.

Borders of the safe box specified as: Amin, Amax, Bmin, Bmax, Cmin, Cmax.

For periodic dimensions fractional coordinates between 0 and 1 and for non-periodic dimensions Cartesian values in Angstrom are expected.

SinkBox**Type**

Block

Description

Part of the simulation box where matching molecules will be removed. By default, molecules matching the formula will be removed regardless of their location. If this block is present then such a molecule will only be removed if any of its atoms is within the box. For a periodic dimension it is given as a fraction of the simulation box (the full 0 to 1 range by default). For a non-periodic dimension it represents absolute Cartesian coordinates in Angstrom.

Amax**Type**

Float

Description

Coordinate of the upper bound along the first axis.

Amin**Type**

Float

Description

Coordinate of the lower bound along the first axis.

Bmax**Type**

Float

Description

Coordinate of the upper bound along the second axis.

Bmin**Type**

Float

Description

Coordinate of the lower bound along the second axis.

Cmax**Type**

Float

Description

Coordinate of the upper bound along the third axis.

Cmin**Type**

Float

Description

Coordinate of the lower bound along the third axis.

FractionalCoordsBox**Type**

Float List

GUI name

Sink box

Description

Remove molecules that are (partly) inside the sink box.

Borders of the sink box specified as: Amin, Amax, Bmin, Bmax, Cmin, Cmax.

For periodic dimensions fractional coordinates between 0 and 1 and for non-periodic dimensions Cartesian values in Angstrom are expected.

StartStep**Type**

Integer

Default value

0

Description

Step number when molecules are removed for the first time. After that, molecules are removed every [Frequency] steps.

For example, if StartStep=99 and Frequency=100 then molecules will be removed at steps 99, 199, 299, etc...

No molecule will be removed at step 0, so if StartStep=0 the first molecules are removed at the step number equal to [Frequency].

StopStep**Type**

Integer

Description

Do not remove the specified molecules after this step.

ReplicaExchange**Type**

Block

Description

This block is used for (temperature) Replica Exchange MD (Parallel Tempering) simulations.

AllowWrongResults**Type**

Bool

Default value

No

Description

Allow combining Replica Exchange with other features when the combination is known to produce physically incorrect results.

EWMALength**Type**

Integer

Default value

10

Description

Length of the exponentially weighted moving average used to smooth swap probabilities for monitoring.

This value is equal to the inverse of the EWMA mixing factor.

SwapFrequency**Type**

Integer

Default value

100

Description

Attempt an exchange every N steps.

TemperatureFactors

Type

Float List

Description

This is the ratio of the temperatures of two successive replicas.

The first value sets the temperature of the second replica with respect to the first replica, the second value sets the temperature of the third replica with respect to the second one, and so on. If there are fewer values than nReplicas, the last value of TemperatureFactor is used for all the remaining replicas.

Temperatures**Type**

Float List

Description

List of temperatures for all replicas except for the first one.

This is mutually exclusive with TemperatureFactors. Exactly nReplicas-1 temperature values need to be specified, in increasing order. The temperature of the first replica is given by [Thermostat%Temperature].

nReplicas**Type**

Integer

Default value

1

GUI name

Number of replicas

Description

Number of replicas to run in parallel.

Restart**Type**

String

GUI name

Restart from

Description

The path to the ams.rkf file from which to restart the simulation.

Shake**Type**

Block

Description

Parameters of the Shake/Rattle algorithm.

All**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, "All triangles H C H" in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2**Type**

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV**Type**

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations

Type

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates**Type**

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Thermostat**Type**

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply**Type**

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)
- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type

Type

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

TimeStep**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

Trajectory**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

EngineResultsFreq**Type**

Integer

GUI name

Engine results frequency

Description

Write MDStep*.rkf files with engine-specific results once every N steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of Checkpoint%Frequency.

ExitConditionFreq**Type**

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every N steps. By default this is done every SamplingFreq steps.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N steps. By default this is done every SamplingFreq steps.

SamplingFreq**Type**

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds**Type**

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges**Type**

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

fbMC**Type**

Block

Recurring

True

GUI name

fbMC

Description

This block sets up force bias Monte Carlo interleaved with the molecular dynamics simulation.

Frequency**Type**

Integer

Default value

1

Description

Run the fbMC procedure every Frequency MD steps.

MassRoot**Type**

Float

Default value

2.0

Description

Inverse of the exponent used to mass-weight fbMC steps.

MolecularMoves**Type**

Block

Description

Move molecules as rigid bodies in addition to normal atomic moves.

Enabled**Type**

Bool

Default value

No

GUI name

Enable molecular moves

Description

Enable moving molecules as rigid bodies based on net forces and torques. Ordinary per-atom displacements will then be based on residual atomic forces.

RotationStepAngle**Type**

Float

Default value

0.1

Unit

Radian

Description

Maximum allowed angle of rotation of each molecule in one fbMC step.

TranslationStepLength**Type**

Float

Default value

0.1

Unit

Angstrom

Description

Maximum allowed displacement of each molecule in each Cartesian coordinate in one fbMC step.

NSteps

Type

Integer

GUI name

Number of steps

Description

Number of fbMC steps to perform on every invocation of the procedure.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N fbMC steps. This defaults to the PrintFreq set in the Trajectory block. Setting this to zero disables printing fbMC steps.

StartStep**Type**

Integer

Default value

1

Description

First step at which the fbMC procedure may run.

StepLength**Type**

Float

Default value

0.1

Unit

Angstrom

Description

Maximum allowed displacement of the lightest atom in the system in each Cartesian coordinate in one fbMC step.

StopStep**Type**

Integer

Default value

0

Description

Last step at which the fbMC procedure may run. If unset or zero, there is no limit.

Temperature**Type**

Float

Unit

Kelvin

Description

Temperature used for fbMC.

Molecules**Type**

Block

Description

Configures details of the molecular composition analysis enabled by the Properties%Molecules block.

AdsorptionSupportRegion**Type**

String

GUI name

Adsorption support region

Description

Select region that will represent a support for adsorption analysis. Adsorbed molecules will receive an '(ads)' suffix after name of the element bonded to the support. Such elements will be listed separate from atoms of the same element not bonded to the support, for example, HOH(ads) for a water molecule bonded to a surface via one of its H atoms.

BondOrderCutoff**Type**

Float

Default value

0.5

Description

Bond order cutoff for analysis of the molecular composition. Bonds with bond order smaller than this value are neglected when determining the molecular composition.

NEB**Type**

Block

Description

Configures details of the Nudged Elastic Band optimization.

Climbing**Type**

Bool

Default value

Yes

GUI name

Climb highest image to TS

Description

Use the climbing image algorithm to drive the highest image to the transition state.

ClimbingThreshold**Type**

Float

Default value

0.0

Unit

Hartree/Bohr

GUI name

CI force threshold

Description

Climbing image force threshold. If ClimbingThreshold > 0 and the max perpendicular force component is above the threshold then no climbing is performed at this step. This entry can be used to get a better approximation for the reaction path before starting the search for the transition state. A typical value is 0.01 Hartree/Bohr.

Images**Type**

Integer

Default value

8

GUI name

Number of images

Description

Number of NEB images (not counting the chain ends). Using more images will result in a smoother reaction path and can help with convergence problems, but it will also increase the computation time.

InterpolateInternal**Type**

Bool

Default value

Yes

GUI name

Interpolate in Internal coordinates

Description

The initial NEB image geometries are calculated by interpolating between the initial and the final state. By default, for non-periodic systems the interpolation is performed in internal coordinates but the user can choose to do it in the Cartesian ones. For periodic systems the interpolation is always done in Cartesian coordinates. If PreOptimizeWithIDPP is set then the path may be further refined using the image-dependent pair potential (IDPP).

InterpolateShortest**Type**

Bool

Default value

Yes

GUI name

Interpolate across cell boundary

Description

Allow interpolation across periodic cell boundaries. Set to false if an atom is intended to move more than half across the simulation box during reaction.

Iterations**Type**

Integer

GUI name

Maximum number of iterations

Description

Maximum number of NEB iterations. The default value depends on the number of degrees of freedom (number of images, atoms, periodic dimensions).

Jacobian**Type**

Float

GUI name

Jacobian value

Description

Scaling factor used to convert the lattice strain to a NEB coordinate value. Default value: $\sqrt{N} \cdot (V/N)^{1/d}$, where V - lattice volume (area for 2D, length for 1D), N - number of atoms, and d - number of periodic dimensions.

LoadPath**Type**

Block

Description

Provide details about the trajectory to get the initial NEB path from. PESScan and NEB trajectories are supported. Only the last geometry for each point on the trajectory is considered.

File**Type**

String

GUI name

Initial path file

Description

Provide an ams.rkf file to load the initial path from. All geometries of this calculation, including initial and final, will be taken from the History section of the file.

Note that for a PESScan it should be a 1D path.

Geometries**Type**

Integer List

GUI name

Raw geometry indices

Description

Raw indices of the geometries from the History section. By default the last geometry of each path point is used.

Points**Type**

Integer List

GUI name

Path points

Description

By default the whole path is used, which may sometimes be not desirable. For example when a PESScan revealed multiple barriers. In this case one can specify indices of the path points to be used. The last geometry of the specified path point will be loaded.

MapAtomsToCell**Type**

Bool

Default value

Yes

GUI name

Map atoms to cell

Description

Translate atoms to the $[-0.5, 0.5]$ cell before every step. This option cannot be disabled for SS-NEB.

OldTangent**Type**

Bool

Default value

No

GUI name

Use old tangent

Description

Turn on the old central difference tangent.

OptimizeEnds**Type**

Bool

Default value

Yes

GUI name

Optimize reactants/products

Description

Start the NEB with optimization of the reactant and product geometries.

OptimizeLattice**Type**

Bool

Default value

No

GUI name

Optimize lattice

Description

Turn on the solid-state NEB (SS-NEB).

Parallel**Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

PreOptimizeWithIDPP**Type**

Bool

Default value

No

GUI name

Use IDPP

Description

(Experimental)

When there is only initial and final system available, the image-dependent pair potential (IDPP, doi: 10.1063/1.4878664) can be used to determine the initial NEB path by interpolating all interatomic distances between the two points and optimizing intermediate images towards them. The optimization starts from the geometries obtained using the selected interpolation options.

ReOptimizeEnds**Type**

Bool

Default value

No

GUI name

Re-optimize reactants/products

Description

Re-optimize reactant and product geometries upon restart.

Restart**Type**

String

GUI name

Restart from

Description

Provide an ams.rkf file from a previous NEB calculation to restart from. It can be an unfinished NEB calculation or one performed with different engine parameters.

Skewness**Type**

Float

Default value

1.0

GUI name

Skewness

Description

Degree of how much images are shifted towards or away from the TS, which may help tackle problems with a long reaction path (for example involving a loose adsorption complex) without needing too many images. A value greater than 1 will make sure that images are concentrated near the transition state. The optimal value depends on the path length, the number of images (larger [Skewness] may be needed for a longer path and fewer images). Technically [Skewness] is equal to the ratio between the optimized distances to the lower and the higher neighbor image on the path.

Spring**Type**

Float

Default value

1.0

UnitHartree/Bohr²**GUI name**

Spring value

Description

Spring force constant in atomic units.

NormalModes

Type

Block

Description

Configures details of a normal modes calculation.

BlockDisplacements**Type**

Block

Description

Configures details of a Block Normal Modes (a.k.a. Mobile Block Hessian, or MBH) calculation.

AngularDisplacement**Type**

Float

Default value

0.5

Unit

Degree

Description

Relative step size for rotational degrees of freedom during Block Normal Modes finite difference calculations. It will be scaled with the characteristic block size.

BlockAtoms**Type**

Integer List

Recurring

True

Description

List of atoms belonging to a block. You can have multiple BlockAtoms.

BlockRegion**Type**

String

Recurring

True

Description

The region to be considered a block. You can have multiple BlockRegions, also in combination with BlockAtoms.

Parallel**Type**

Block

Description

Configuration for how the individual displacements are calculated in parallel.

nCoresPerGroup**Type**

Integer

Description

Number of cores in each working group.

nGroups**Type**

Integer

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Cores per task

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

RadialDisplacement**Type**

Float

Default value

0.005

Unit

Angstrom

Description

Step size for translational degrees of freedom during Block Normal Modes finite difference calculations.

Displacements**Type**

Multiple Choice

Default value

Cartesian

Options

[Cartesian, Symmetric, Block]

GUI name

Displacements

Description

Type of displacements.

In case of symmetric displacements it is possible to choose only the modes that have non-zero IR or Raman intensity.

Block displacements take rigid blocks into account.

Hessian**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Analytical, Numerical]

Description

Default Auto means that if possible by the engine the Hessian will be calculated analytically, else the Hessian will be calculated numerically by AMS.

ReScanFreqRange**Type**

Float List

Default value

[-10000000.0, 10.0]

Unit

cm-1

Recurring

True

GUI name

Re-scan range

Description

Specifies a frequency range within which all modes will be scanned. 2 numbers: an upper and a lower bound.

ReScanModes**Type**

Bool

Default value

Yes

GUI name

Re-scan modes

Description

Whether or not to scan imaginary modes after normal modes calculation has concluded.

SymmetricDisplacements**Type**

Block

Description

Configures details of the calculation of the frequencies and normal modes of vibration in symmetric displacements.

Type**Type**

Multiple Choice

Default value

All

Options

[All, Infrared, Raman, InfraredAndRaman]

GUI name

Symm Frequencies

Description

For symmetric molecules it is possible to choose only the modes that have non-zero IR or Raman intensity (or either of them) by symmetry.

In order to calculate the Raman intensities the Raman property must be requested.

NumericalDifferentiation**Type**

Block

Description

Define options for numerical differentiations, that is the numerical calculation of gradients, Hessian and the stress tensor for periodic systems.

NuclearStepSize**Type**

Float

Default value

0.005

Unit

Bohr

Description

Step size for numerical nuclear gradient calculation.

Parallel**Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

StrainStepSize**Type**

Float

Default value

0.001

Description

Step size (relative) for numerical stress tensor calculation.

NumericalPhonons**Type**

Block

Description

Configures details of a numerical phonons calculation.

AutomaticBZPath**Type**

Bool

Default value

Yes

GUI name

Automatic BZ path

Description

If True, compute the phonon dispersion curve for the standard path through the Brillouin zone.
If False, you must specify your custom path in the [BZPath] block.

BZPath**Type**

Block

Description

If [NumericalPhonons%AutomaticBZPath] is false, the phonon dispersion curve will be computed for the user-defined path in the [BZPath] block. You should define the vertices of your path in fractional coordinates (with respect to the reciprocal lattice vectors) in the [Path] sub-block. If you want to make a jump in your path (i.e. have a discontinuous path), you need to specify a new [Path] sub-block.

Path**Type**

Non-standard block

Recurring

True

Description

A section of a k space path. This block should contain multiple lines, and in each line you should specify one vertex of the path in fractional coordinates. Optionally, you can add text labels for your vertices at the end of each line.

BornEffCharge**Type**

Float

Default value

0.0

Description

Input option to give the Born effective charges of the species.

DielectricConst**Type**

Float

Default value

1.0

Description

Input option to give the static dielectric constant of the species.

DoubleSided**Type**

Bool

Default value

Yes

Description

By default a two-sided (or quadratic) numerical differentiation of the nuclear gradients is used. Using a single-sided (or linear) numerical differentiation is computationally faster but much less accurate. Note: In older versions of the program only the single-sided option was available.

Interpolation**Type**

Integer

Default value

100

Description

Use interpolation to generate smooth phonon plots.

KPathFinderConvention**Type**

Multiple Choice

Default value

Setyawan-Curtarolo

Options

[Setyawan-Curtarolo, Hinuma]

Description

This option determines how the path through the Brillouin zone is generated when using the automatic k-point mode.

Available options:

- **Setyawan-Curtarolo** (default for 1D and 2D lattices): Uses our built-in KPath program to find a path through high-symmetry points based on the method by Setyawan and Curtarolo (<https://doi.org/10.1016/j.commatsci.2010.05.010>). For 2D lattices, the path is derived from the intersection of the 3D Brillouin zone with a plane. For 1D lattices, the path is simply Gamma-Z.
- **Hinuma**: Uses the external SeeKPath utility to generate the k-path (<https://github.com/giovannipizzi/seekpath> and <https://doi.org/10.1016/j.commatsci.2016.10.015>).

NDosEnergies**Type**

Integer

Default value

1000

Description

Nr. of energies used to calculate the phonon DOS used to integrate thermodynamic properties. For fast compute engines this may become time limiting and smaller values can be tried.

Parallel**Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

StepSize**Type**

Float

Default value

0.04

Unit

Angstrom

Description

Step size to be taken to obtain the force constants (second derivative) from the analytical gradients numerically.

SuperCell**Type**

Non-standard block

Description

Used for the phonon run. The super lattice is expressed in the lattice vectors. Most people will find a diagonal matrix easiest to understand.

PESExploration**Type**

Block

Description

Configures details of the automated PES exploration methods.

BasinHopping**Type**

Block

Description

Configures the details of the Basin Hopping subtask.

DisplaceAtomsInRegion**Type**

String

Default value**Description**

If you specify a region name here, only the atoms belonging to this region will be displaced during the basin hopping procedure. For more details on regions, see the documentation on the System definition.

Displacement**Type**

Float

Default value

0.5

Unit

Angstrom

Description

Displacement in each degree of freedom.

MainSystemAsSeed**Type**

Bool

Default value

No

Description

If true, only the main system will be used as a seed state. The main system is not added to the database.

PESPointCharacterization**Type**

Bool

Default value

Yes

Description

If true, a PES point characterization based on a vibrational analysis is carried out to confirm each detected state is an actual local minimum (no imaginary frequencies). Conversely, if this option is false, the PES point characterization is avoided, which will assume that all located states are local minima (zero gradients). Enabling this option is very useful for large systems. It circumvents the need for computing and diagonalizing the Hessian matrix, a typically expensive computational process.

PushApartDistance**Type**

Float

Default value

0.4

Unit

Angstrom

Description

Push atoms apart until no atoms are closer than this distance. This criterion is enforced for the initial structure and all those generated by random displacements.

RotateNonListedAtoms**Type**

Bool

Default value

No

Description

If true, each iteration randomly rotates all atoms as a rigid body, excluding those listed in

[BasinHopping%DisplaceListedAtoms], [BasinHopping%DisplaceListedTypes], or [BasinHopping%DisplaceAtomsInRegion].

Steps**Type**

Integer

Default value

20

Description

Number of displace & optimize Monte-Carlo steps to take.

BindingSites**Type**

Block

Description

Options related to the calculation of binding sites.

Calculate**Type**

Bool

Default value

No

Description

Calculate binding sites at the end of a job. Not needed for Binding Sites job.

DistanceDifference**Type**

Float

Default value

-1.0

Unit

Angstrom

Description

If the distance between two mapped binding-sites is larger than this threshold, the binding-sites are considered different. If not specified, its value will set equal to [PESExploration%StructureComparison%DistanceDifference]

MaxCoordinationShellsForLabels**Type**

Integer

Default value

3

Description

The binding site labels are given based on the coordination numbers of shells in the reference region, using the following format: N<int><int>..., e.g., the label 'N334' means 3 atoms in the first coordination shell, 3 in the second one, and 4 in the third one. This parameter controls the maximum number of shells to include.

NeighborCutoff

Type

Float

Default value

-1.0

Unit

Angstrom

Description

Atoms within this distance of each other are considered neighbors for the calculation of the binding sites. If not specified, its value will set equal to [PESExploration%StructureComparison%NeighborCutoff]

ReferenceRegion**Type**

String

Default value**Description**

Defines the region that is considered as the reference for binding sites detection. Binding sites are projected on this region using the geometry from the reference system. If not specified, its value will set equal to [PESExploration%StatesAlignment%ReferenceRegion]

CalculateEnergyReferences**Type**

Bool

Default value

No

Description

Calculates the energy references.

CalculateFragments**Type**

Bool

Default value

No

Description

Must be used together with an adsorbent set as the StatesAlignment%ReferenceRegion. Runs a final calculation of the adsorbate and adsorbent (marked by the ReferenceRegion) individually. The fragmented state is included in the energy landscape.

Debug**Type**

Block

Description

???

DynamicSeedStates**Type**

Bool

Default value

Yes

Description

Whether subsequent expeditions may start from states discovered by previous expeditions. This should lead to a more comprehensive exploration of the potential energy surface. Disabling this will focus the PES exploration around the initial seed states.

Dynamics**Type**

Block

Description

???

Andersen**Type**

Block

Description

???

Alpha**Type**

Float

Default value

1.0

Description

???

CollisionPeriod**Type**

Float

Default value

100.0

Description

???

Langevin**Type**

Block

Description

???

Friction**Type**

Float

Default value

0.01

Description

???

Nose

Type

Block

Description

???

Mass**Type**

Float

Default value

1.0

Description

???

Thermostat**Type**

Multiple Choice

Default value

none

Options

[andersen, nose_hoover, langevin, none]

Description

???

Time**Type**

Float

Default value

1000.0

Description

???

TimeStep**Type**

Float

Default value

1.0

Description

???

FiniteDifference**Type**

Float

Default value

0.0026458861

Unit

Angstrom

Description

The finite difference distance to use for Dimer, Hessian, Lanczos, and optimization methods.

Hessian**Type**

Block

Description

???

AtomList**Type**

String

Default value

all

Description

???

ZeroFreqValue**Type**

Float

Default value

1e-06

Description

???

Job**Type**

Multiple Choice

Options

[ProcessSearch, BasinHopping, SaddleSearch, LandscapeRefinement, BindingSites]

Description

Specify the PES exploration job to perform.

LandscapeRefinement**Type**

Block

Description

Configures details of the energy landscape refinement job.

CalculateOnlyEnergies**Type**

Bool

Default value

No

Description

If true, the states' geometry is not optimized, and the final PES point characterization is ignored [PESExploration%LandscapeRefinement%IgnoreFinalPESPointCharacter]. Only energy values are updated using the specified engine. Furthermore, normal modes and associated properties are copied from the previous calculation to avoid the typically high computa-

tional effort of the Hessian matrix calculation. Enabling this option implies that `RunInitialSinglePoints='F'`, `IgnoreFinalPESPointCharacter='T'`.

IgnoreFinalPESPointCharacter

Type

Bool

Default value

No

Description

At the end of the energy landscape refinement job, each state is assigned a PES point character (MIN or TS) based on its vibrational frequencies before being included in the final database. States are only added if the PES point character after refinement remains unchanged. However, states are added without verifying if this option is `true`. Nonetheless, vibrational frequencies are calculated and stored for future analysis. This option is especially useful when using computationally demanding engines. Because in those cases, precision and computational effort must be balanced, resulting in significant vibrational frequencies inaccuracies.

IgnoreFinalPESPointCharacterForFragments

Type

Bool

Default value

No

GUI name

... for fragments

Description

Same as `LandscapeRefinement%IgnoreFinalPESPointCharacter` but regarding the Fragments calculations, see option `LandscapeRefinement%CalculateFragments`.

RelaxFromSaddlePoint

Type

Bool

Default value

No

Description

Relaxes the saddle point geometries following the imaginary mode to get both reactants and products.

RunInitialSinglePoints

Type

Bool

Default value

Yes

Description

If it is `true`, just after loading the energy landscape to refine, the single energy point computations are disabled. Be aware that if you enable this, the output file's 'Initial Energy Landscape' section will display incorrect states' energy values. If the engine requires too much processing power, this option can help you save a small amount of time.

TransitionStateSearchMethod**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Dimer]

Description

Sets the method to refine transition states.

LoadEnergyLandscape**Type**

Block

Description

Options related to the loading of an Energy Landscape from a previous calculation.

GenerateSymmetryImages**Type**

Bool

Default value

No

Description

By activating this option, after loading the energy landscape, it will create the complete set of symmetry-related copies by using the symmetry operators of the reference structure. Be aware that rkf result files of the generated symmetry images are copies from the parent structures but only atomic coordinates are updated.

KeepOnly**Type**

Integer List

GUI name

List of states to keep

Description

Upon loading the Energy Landscape, only keep the states specified here. The states should be specified via a list of integers referring to the indices of the states you want to keep.

Path**Type**

String

GUI name

Load energy landscape from

Description

AMS results folder to load an energy landscape from. In the text input file, you may alternatively specify a .con file in the native EON format.

Remove**Type**

Integer List

GUI name

List of states to remove

Description

Upon loading the Energy Landscape, remove (i.e. do not load) the states specified here. The states should be specified via a list of integers referring to the indices of the states you want to remove (i.e. the states you don't want to load).

RemoveWithNoBindingSites**Type**

Bool

Default value

No

Description

Upon loading the Energy Landscape, it removes states with no associated binding sites. Associated transition states are also removed. This is an advantageous option to remove physisorbed states automatically. Notice that it requires that the previous calculation was executed, enabling the option [BindingSites%Calculate].

SeedStates**Type**

Integer List

GUI name

List of seed states

Description

By default when you start a new PES Exploration from a loaded Energy Landscape, expeditions can start from any of the loaded minima. By using this input option, you can instruct the program to only use some of the states as 'expedition starting point'. The states that serve as 'expedition starting points' should be specified via a list of integers referring to the indices of the states.

LoadInitialSystems**Type**

Block

Description

Load initial systems from different sources.

FromKFHistory**Type**

Block

Recurring

True

Description

Load initial systems from the History section of kf files.

Path**Type**

String

GUI name

Load initial systems from

Description

AMS results folder to load the initial systems from. In the text input file, you may alternatively specify a `.kf` file. By default, the `ams.kf` file will be used.

TargetPESPointCharacter**Type**

Multiple Choice

Default value

MIN

Options

[MIN, TS]

Description

Selects the PES point character for all loaded systems.

NegativeEigenvalueTolerance**Type**

Float

Default value

-0.0005

Unit

Hartree/Bohr²

Description

The threshold in Hessian eigenvalue below which a mode is considered imaginary, i.e. indicating a transition state. This is a small negative number, as very small negative eigenvalues may be due to numerical noise on an essentially flat PES and do not indicate true transition states. We need a more flexible value for this parameter in PESExploration because the high computational cost of the task typically forces us to reduce the engine precision, which increases the noise in the vibrational frequencies evaluation. [PESPointCharacter%NegativeEigenvalueTolerance] is overridden by this parameter.

NudgedElasticBand**Type**

Block

Description

Options for the Nudged Elastic Band (NEB) method.

ClimbingImageMethod**Type**

Bool

Default value

Yes

Description

Use the climbing image algorithm to drive the highest image to the transition state.

ConvergedForce**Type**

Float

Default value

-1.0

Unit

eV/Angstrom

Description

Convergence threshold for nuclear gradients. Note: Special value of -1.0 means using the same convergence criterion as the PES explorer's geometry optimizer.

Images**Type**

Integer

Default value

5

Description

Number of NEB images between the two endpoints.

MaxIterations**Type**

Integer

Default value

500

Description

Maximum number of NEB iterations.

OldTangent**Type**

Bool

Default value

No

Description

Use the old central difference tangent.

Spring**Type**

Float

Default value

5.0

UniteV/Ang²**Description**

Spring force constant.

NumExpeditions**Type**

Integer

Default value

1

Description

Sets the number of subsequent expeditions our job will consist of. Larger values result in a more

comprehensive exploration of the potential energy surface, but will take more computational time.

NumExplorers**Type**

Integer

Default value

1

Description

Sets the number of independent PES explorers dispatched as part of each expedition. Larger values will result in a more comprehensive exploration of the potential energy surface, but will take more computational time. By default an appropriate number of explorers are executed in parallel.

OptTSMethod**Type**

Multiple Choice

Default value

SaddleSearch

Options

[SaddleSearch, NudgedElasticBand]

Description

When the full set of states in the energy landscape are optimized (see PESExploration%Job = GeometryOptimization), transition states can be optimized using either SaddleSearch or NudgedElasticBand methods. SaddleSearch uses information only from the current geometry of the TS; contrary, NudgedElasticBand ignores the current geometry and runs a Nudged-Elastic-Band calculation trying to connect the associated reactants and products if they are available.

Optimizer**Type**

Block

Description

Configures the details of the geometry optimizers used by the PES explorers.

ConvergedForce**Type**

Float

Default value

0.005

Unit

eV/Angstrom

Description

Convergence threshold for nuclear gradients.

MaxIterations**Type**

Integer

Default value

400

Description

Maximum number of iterations allowed for optimizations.

Method**Type**

Multiple Choice

Default value

CG

Options

[CG, QM, LBFGS, FIRE, SD]

Description

Select the method for geometry optimizations.

Parallel**Type**

Block

Description

Options for double parallelization, which allows to split the available processor cores into groups working through all the available tasks in parallel, resulting in a better parallel performance. The keys in this block determine how to split the available processor cores into groups working in parallel.

nCoresPerGroup**Type**

Integer

GUI name

Cores per group

Description

Number of cores in each working group.

nGroups**Type**

Integer

GUI name

Number of groups

Description

Total number of processor groups. This is the number of tasks that will be executed in parallel.

nNodesPerGroup**Type**

Integer

GUI name

Nodes per group

Description

Number of nodes in each group. This option should only be used on homogeneous compute clusters, where all used compute nodes have the same number of processor cores.

ParallelReplica**Type**

Block

Description

???

DephaseLoopMax**Type**

Integer

Default value

5

Description

???

DephaseLoopStop**Type**

Bool

Default value

No

Description

???

DephaseTime**Type**

Float

Default value

1000.0

Description

???

PostTransitionTime**Type**

Float

Default value

1000.0

Description

???

RefineTransition**Type**

Bool

Default value

Yes

Description

???

StateCheckInterval

Type

Float

Default value

1000.0

Description

???

StateSaveInterval**Type**

Float

Default value

-1.0

Description

???

StopAfterTransition**Type**

Bool

Default value

No

Description

???

Prefactor**Type**

Block

Description

???

Rate**Type**

Multiple Choice

Default value

HTST

Options

[HTST, QQHTST, None]

Description

Calculates reaction rate pre-exponential factors via: HTST (Harmonic Transition State Theory), QQHTST (quasi-quantum HTST), or None (disable calculation).

ProcessSearch**Type**

Block

Description

Input options specific to the process search procedure.

MinimizationOffset

Type

Float

Default value

0.2

Description

After a saddle is found, images are placed on either side of the saddle along the mode and minimized to ensure that the saddle is connected to the original minimum and to locate the product state. MinimizationOffset is the distance those images are displaced from the saddle.

RandomSeed**Type**

Integer

Description

Number used to initialize both the EON clients random number generators as well as the AMS global RNG. The latter is normally initialized with the RNGSeed keyword at the root level. Should be used by developers only. May or may not help to make more reproducible regression tests ...

SaddleSearch**Type**

Block

Description

Configuration for the Saddle Search procedure (used in SaddleSearch and ProcessSearch Jobs).

ConvergedForce**Type**

Float

Default value

-1.0

Unit

eV/Angstrom

Description

Convergence threshold for nuclear gradients. Note: Special value of -1.0 means using the same convergence criterion as the PES explorer's geometry optimizer.

DisplaceAlongNormalModesActiveModes**Type**

String

Default value**GUI name**

Displace active modes

Description

Sets the active modes to be used when the option [SaddleSearch%DisplaceAlongNormalModesWeight] is enabled. e.g. 1,2,3,5. By default, all normal modes are considered active.

DisplaceAlongNormalModesWeight**Type**

Float

Default value

0.0

GUI name

Displace modes weight

Description

The probability of generating a displacement resulting in a random linear combination of the normal modes specified in [SaddleSearch%DisplaceAlongNormalModesActiveModes]. This parameter is a numeric value that should fall within the interval [0.0, 1.0].

DisplaceAtomsInRegion**Type**

String

Default value**Description**

A string corresponding to the name of a region. When performing the initial random displacement, only displace atoms in the specified region.

DisplaceAtomsInRegionWeight**Type**

Float

Default value

0.0

Description

The probability of generating a displacement involving only atoms from the region specified in [SaddleSearch%DisplaceAtomsInRegion]. This parameter is a numeric value that should fall within the interval [0.0, 1.0].

DisplaceListedAtoms**Type**

String

Default value**Description**

Sets the active atoms to be used when the option [SaddleSearch%DisplaceAlongNormalModesWeight] is enabled. e.g. 1,2,3,5. By default, all normal modes are considered active.

DisplaceListedTypes**Type**

String

Default value**Description**

???

DisplaceMagnitude**Type**

Float

Default value

0.1

Unit

Angstrom

Description

The standard deviation of the Gaussian displacement in each degree of freedom for the selected atoms.

DisplaceRadius**Type**

Float

Default value

4.0

Unit

Angstrom

Description

Atoms within this distance of the epicenter will be displaced.

MaxEnergy**Type**

Float

Default value

20.0

Unit

eV

Description

The energy (relative to the starting point of the saddle search) at which a saddle search explorer considers the search bad and terminates it.

MaxIterations**Type**

Integer

Default value

400

Description

Maximum number of iterations for each saddle search run.

MinEnergyBarrier**Type**

Float

Default value

0.001

Unit

eV

Description

Minimum energy barrier to accept a new transition state.

MinModeMethod**Type**

Multiple Choice

Default value

dimer

Options

[dimer, lanczos]

Description

The minimum-mode following method to use.

RelaxFromSaddlePoint**Type**

Bool

Default value

No

Description

Relaxes the saddle point geometries following the imaginary mode to get both reactants and products.

ZeroModeAbortCurvature**Type**

Float

Default value

0.01

UniteV/Angstrom²**Description**

The threshold in the frequency below which the minimum mode is considered zero. The calculation is aborted if the negative mode becomes zero.

SelectedListedAtomsForPESPointCharacter**Type**

String

Default value**GUI name**

PESPoint character for atoms

Description

Uses the Hessian matrix elements only for the listed atoms to determine the PES point character of a located state during the exploration. If not specified, the full Hessian is used.

SelectedRegionForPESPointCharacter**Type**

String

Default value**GUI name**

PESPoint character for region

Description

Uses the Hessian matrix elements only for the atoms in a particular region to determine the PES point character of a located state during the exploration. If not specified, the full Hessian is used.

StatesAlignment**Type**

Block

Description

Configures details of how the energy landscape configurations are aligned respect to the main chemical system [System].

DistanceDifference**Type**

Float

Default value

-1.0

Unit

Angstrom

Description

If the distance between two mapped atoms is larger than this threshold, the configuration is considered not aligned. If not specified, its value will set equal to [PESExploration%StructureComparison%DistanceDifference]

ReferenceRegion**Type**

String

Default value**Description**

Defines the region that is considered as the reference for alignments. Atoms outside this region are ignored in the alignments.

StructureComparison**Type**

Block

Description

Settings for structure comparison.

CheckRotation**Type**

Bool

Description

Rotates the system optimally before comparing structures. The default is to do this only for molecular systems when there are no fixed atom constraints.

CheckSymmetry**Type**

Bool

Default value

No

Description

Considers that two systems are equal if they are equivalent by symmetry.

DistanceDifference**Type**

Float

Default value

0.1

Unit

Angstrom

Description

If the distance between two mapped atoms is larger than this threshold, the two configurations are considered different structures.

EnergyDifference**Type**

Float

Default value

0.01

Unit

eV

Description

If the energy difference between two configurations is larger than this threshold, the two configurations are considered to be different structures.

IndistinguishableAtoms**Type**

Bool

Default value

Yes

Description

If yes, the order of the atoms does not affect the structural comparison. Atoms of the same element are then indistinguishable.

NeighborCutoff**Type**

Float

Default value

3.3

Unit

Angstrom

Description

Atoms within this distance of each other are considered neighbors.

RemoveTranslation**Type**

Bool

Description

Translates the system optimally before comparing structures. The default is to do this only when there are no fixed atom constraints.

Temperature**Type**

Float

Default value

300.0

Unit

Kelvin

Description

The temperature that the job will run at. This may be used in different ways depending on the job, e.g. acceptance probabilities for Monte-Carlo based jobs, thermostating for dynamics based jobs, kinetic prefactors for jobs that find transition states. Some jobs may not use this temperature at all.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteHistory**Type**

Multiple Choice

Default value

Converged

Options

[None, Converged, All]

Description

When to write the molecular geometry (and possibly other properties) to the history on the ams.rkf file. The default is to only write the converged geometries to the history. Can be changed to write no frames at all to the history, or write all frames (should only be used when testing because of the performance impact). Note that for parallel calculations, only the first group of processes writes to ams.rkf.

PESPointCharacter**Type**

Block

Description

Options for the characterization of PES points.

Displacement**Type**

Float

Default value

0.04

Description

Controls the size of the displacements used for numerical differentiation: The displaced geometries are calculated by taking the original coordinates and adding the mass-weighted mode times the reduced mass of the mode times the value of this keyword.

NegativeEigenvalueTolerance**Type**

Float

Default value

-0.0001

UnitHartree/Bohr²**Description**

The threshold in Hessian eigenvalue below which a mode is considered imaginary, i.e. indicating a transition state. This is a small negative number, as very small negative eigenvalues may be due to numerical noise on an essentially flat PES and do not indicate true transition states.

NumberOfModes**Type**

Integer

Default value

2

Description

The number of (lowest) eigenvalues that should be checked.

Tolerance**Type**

Float

Default value

0.016

Description

Convergence tolerance for residual in iterative Davidson diagonalization.

PESScan**Type**

Block

Description

Configures the details of the potential energy surface scanning task.

CalcPropertiesAtPESPoints**Type**

Bool

Default value

No

Description

Whether to perform an additional calculation with properties on all the sampled points of the PES. If this option is enabled AMS will produce a separate engine output file for every sampled PES point.

FillUnconvergedGaps**Type**

Bool

Default value

Yes

Description

After the initial pass over the PES, restart the unconverged points from converged neighboring points.

ScanCoordinate**Type**

Block

Recurring

True

Description

Specifies a coordinate along which the potential energy surface is scanned. If this block contains multiple entries, these coordinates will be varied and scanned together as if they were one. Note that there can be only one ScanCoordinate containing a lattice scan in any PES scan job.

Angle**Type**

String

Recurring

True

Description

Scan the angle between three atoms. Three atom indices followed by two real numbers delimiting the transit range in degrees.

CellVolumeRange**Type**

Float List

Unit

Angstrom^3

Description

Two numbers for the initial and final cell volume. The cell is scaled isotropically between these values. Can not be used together with any other coordinate within the same ScanCoordinate block.

CellVolumeScalingRange**Type**

Float List

Description

Two scaling factors for the initial and final cell volume. A value of '0.9 1.1' would result in an isotropic scaling between 90% and 110% of the cell volume of the input system. Can not be used together with any other coordinate within the same ScanCoordinate block.

Coordinate**Type**

String

Recurring

True

Description

Scan a particular coordinate of an atom. Atom index followed by (xlylz) followed by two real numbers delimiting the transit range.

DifDist**Type**

String

Recurring

True

Description

Scan the difference distance between two pairs of atoms, R(12)-R(34). Four atom indices followed by two real numbers delimiting the transit range in Angstrom.

Dihedral**Type**

String

Recurring

True

Description

Scan the dihedral angle between four atoms. Four atom indices followed by two real numbers delimiting the transit angle in degrees.

Distance**Type**

String

Recurring

True

Description

Scan the distance between two atoms. Two atom indices followed by two real numbers delimiting the transit distance in Angstrom.

FromLattice**Type**

Non-standard block

Description

Up to three lattice vectors to start the scan at. Has to be used in combination with the ToLattice keyword and no other coordinate within the same ScanCoordinate block. Unit can be specified in the header. Default unit is Angstrom.

FromStrainVoigt**Type**

Float List

Description

The elements of the initial lattice strain in Voigt notation. One should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems. Has to be used in combination with the ToStrainVoigt keyword and no other coordinate within the same ScanCoordinate block.

LatticeARange

Type

Float List

Unit

Angstrom

Description

Scans the length of the first lattice vector. Can be combined with the LatticeBRange and LatticeCRange keywords, but no other coordinates within the same ScanCoordinate..

LatticeBRange**Type**

Float List

Unit

Angstrom

Description

Scans the length of the second lattice vector. Can be combined with the LatticeARange and LatticeCRange keyword, but no other coordinates within the same ScanCoordinate..

LatticeCRange**Type**

Float List

Unit

Angstrom

Description

Scans the length of the third lattice vector. Can be combined with the LatticeARange and LatticeBRange keyword, but no other coordinates within the same ScanCoordinate..

SumDist**Type**

String

Recurring

True

Description

Scan the sum of distances between two pairs of atoms, R(12)+R(34). Four atom indices followed by two real numbers delimiting the transit range in Angstrom.

ToLattice**Type**

Non-standard block

Description

Up to three lattice vectors to end the scan at. Unit can be specified in the header. Default unit is Angstrom.

ToStrainVoigt**Type**

Float List

Description

The elements of the final lattice strain in Voigt notation. One should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

nPoints**Type**

Integer

Default value

10

Description

The number of points along the scanned coordinate. Must be greater or equal 2.

Phonons**Type**

Block

Description

Configures the phonons calculation.

BandStructure**Type**

Bool

Default value

Yes

GUI name

Calculate phonons band structure

Description

Calculates and saves the phonon band structure for visualization. Further configuration options for analytical calculations may be available in the engine-specific settings (Phonons%BandStructure).

DOS**Type**

Bool

Default value

Yes

GUI name

Calculate phonon DOS

Description

Calculates and saves the phonon density of states (DOS) for visualization. Further configuration options for analytical calculations may be available in the engine-specific settings (Phonons%DOS).

Method**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Analytical, Numerical]

Description

Determines how phonons are calculated. `Auto` selects the analytical method if supported by

the engine, otherwise it defaults to numerical calculation. Configure numerical parameters in the `NumericalPhonons` section. Engine-specific options for analytical phonon calculations may be available in `Engine%Phonons`.

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

Timers**Type**

Multiple Choice

Default value

None

Options

[None, Normal, Detail, TooMuchDetail]

Description

Printing timing details to see how much time is spend in which part of the code.

Properties**Type**

Block

Description

Configures which AMS level properties to calculate for `SinglePoint` calculations or other important geometries (e.g. at the end of an optimization).

BondOrders**Type**

Bool

Default value

No

Description

Requests the engine to calculate bond orders.

For MM engines these might just be the defined bond orders that go into the force-field, while for QM engines, this might trigger a bond order analysis based on the electronic structure. For engines that do not have a bond order analysis method, a bond guessing algorithm will be used. See also the input options in the `BondOrders` block.

Charges**Type**

Bool

Default value

No

Description

Requests the engine to calculate the atomic charges.

DipoleGradients

Type

Bool

Default value

No

Description

Requests the engine to calculate the nuclear gradients of the electric dipole moment of the molecule. This can only be requested for non-periodic systems.

DipoleMoment**Type**

Bool

Default value

No

Description

Requests the engine to calculate the electric dipole moment of the molecule. This can only be requested for non-periodic systems.

ElasticTensor**Type**

Bool

Default value

No

Description

Calculate the elastic tensor.

GSES**Type**

Bool

Default value

No

Description

Requests the engine to calculate the gradients of ground to excited state properties.

Gradients**Type**

Bool

Default value

No

GUI name

Nuclear gradients

Description

Calculate the nuclear gradients.

Hessian**Type**

Bool

Default value

No

Description

Whether or not to calculate the Hessian.

Molecules**Type**

Bool

Default value

No

Description

Requests an analysis of the molecular components of a system, based on the bond orders calculated by the engine.

NormalModes**Type**

Bool

Default value

No

GUI name

Frequencies

Description

Calculate the frequencies and normal modes of vibration, as well as the corresponding IR intensities, if the engine supports these calculations natively or can calculate dipole moments.

OrbitalsInfo**Type**

Bool

Default value

No

Description

Basic molecular orbitals information: orbital energies, occupations, HOMO, LUMO and HOMO-LUMO gap.

Other**Type**

Bool

Default value

Yes

Description

Other (engine specific) properties. Details are configured in the engine block.

PESPointCharacter**Type**

Bool

Default value

No

GUI name

Characterize PES point

Description

Determine whether the sampled PES point is a minimum or saddle point. Note that for large systems this does not entail the calculation of the full Hessian and can therefore be used to quickly confirm the success of a geometry optimization or transition state search.

Phonons**Type**

Bool

Default value

No

Description

Calculate the phonons (for periodic systems).

Polarizability**Type**

Bool

Default value

No

Description

Requests the engine to calculate the polarizability tensor of the system.

Raman**Type**

Bool

Default value

No

Description

Requests calculation of Raman intensities for vibrational normal modes.

SelectedRegionForHessian**Type**

String

GUI name

Hessian only for

Description

Compute the Hessian matrix elements only for the atoms in a particular region. If not specified, the Hessian will be computed for all atoms.

StressTensor**Type**

Bool

Default value

No

GUI name

Stress tensor

Description

Calculate the stress tensor.

VCD

Type

Bool

Default value

No

Description

Requests calculation of VCD for vibrational normal modes.

VROA**Type**

Bool

Default value

No

Description

Requests calculation of VROA for vibrational normal modes.

Raman**Type**

Block

Description

Configures details of the Raman or VROA calculation.

FreqRange**Type**

Float List

Unit

cm-1

Recurring

True

GUI name

Frequency range

Description

Specifies a frequency range within which all modes will be scanned. 2 numbers: an upper and a lower bound.

IncidentFrequency**Type**

Float

Default value

0.0

Unit

eV

Description

Frequency of incident light.

LifeTime**Type**

Float

Default value

0.0

Unit

hartree

Description

Specify the resonance peak width (damping) in Hartree units. Typically the lifetime of the excited states is approximated with a common phenomenological damping parameter. Values are best obtained by fitting absorption data for the molecule, however, the values do not vary a lot between similar molecules, so it is not hard to estimate values. A typical value is 0.004 Hartree.

Replay**Type**

Block

Description

Configures the details of the Replay task.

File**Type**

String

GUI name

Restart from

Description

Provide an ams.rkf file (or a .results folder) from a previously run job to replay. The file needs to contain a History section.

Frames**Type**

Integer List

Description

List of frames from the History section to recompute.

If not specified the recomputed frames are determined automatically based on the task of the job that is being replayed: PES scans and NEB calculations will only have the converged points replayed, while all other tasks will have all frames recomputed.

Specifying the frames to recompute in the input is probably only useful when replaying trajectories from MolecularDynamics calculations.

StoreAllResultFiles**Type**

Bool

Default value

No

Description

If this option is enabled AMS will produce a separate engine output file for every replayed frame.

While basic properties like energy, gradients, stress tensor, etc. are stored anyway on the History section in the AMS driver output file (if they were requested in the Properties block), engine

specific properties (e.g. excitations energies from ADF) will only be available if the full result files are stored.

Restraints

Type

Block

Description

The Restraints block allows to add soft constraints to the system. A restraint is a potential energy function (a spring) attached to a certain coordinate, for example, an interatomic distance, with its minimum at the specified optimal value. A restraint is defined using one or two parameters: the ForceConstant and, for some types, the F(Inf) value. The ForceConstant parameter corresponds to second derivative of the restraint potential energy $d^2V(x)/dx^2$ for any x (harmonic restraints) or only at $x=0$ (other restraints). Here, x is a deviation from the restraint's optimal value.

Angle

Type

String

Recurring

True

Description

Specify three atom indices $i\ j\ k$ followed by an angle in degrees and, optionally, by the ForceConstant (default is 0.3 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the i - j - k angle at the given value. For periodic systems this restraint follows the minimum image convention.

DifDist

Type

String

Recurring

True

Description

Specify four atom indices $i\ j\ k\ l$ followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the difference $R(ij)-R(kl)$ at the given value. For periodic systems this restraint follows the minimum image convention.

Dihedral

Type

String

Recurring

True

Description

Specify four atom indices $i\ j\ k\ l$ followed by an angle in degrees and, optionally, by the ForceConstant (default is 0.1 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the i - j - k - l dihedral angle at the given value. For periodic systems this restraint follows the minimum image convention.

Distance

Type

String

Recurring

True

Description

Specify two atom indices followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the distance between the two specified atoms at the given value. For periodic systems this restraint follows the minimum image convention.

FInfinity**Type**

Float

Default value

1.0

GUI name

Default F(inf)

Description

Specify the default asymptotic value for the restraint force for the Hyperbolic and Erf profiles, in Hartree/Bohr or Hartree/radian.

A per-restraint value can be specified after the profile type on the corresponding restraint line.

Profile**Type**

Multiple Choice

Default value

Harmonic

Options

[Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Default restraint profile

Description

Select the default type of restraint profile.

The harmonic profile is most suitable for geometry optimizations but may result in very large forces that can be problematic in molecular dynamic.

For MD simulations the Hyperbolic or Erf may be more suitable because the restraint force is bounded by a user-defined value.

A per-restraint profile type can be specified after the ForceConstant value on the corresponding restraint line.

SumDist**Type**

String

Recurring

True

Description

Specify four atom indices i j k l followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try

to keep the sum $R(ij)+R(kl)$ at the given value. For periodic systems this restraint follows the minimum image convention.

Units**Type**

Multiple Choice

Default value

Default

Options

[Default, MD]

GUI name

Units

Description

Change units for energy, force and force constant values from the default (atomic units) to those often used in the MD community (based on kcal/mol and Angstrom). Units for the optimal distances are not affected and are always Angstrom.

RigidMotions**Type**

Block

Description

Specify which rigid motions of the total system are allowed. An external field is not considered part of the system. Normally the automatic option is doing what you want. However this feature can be used as a means of geometry constraint.

AllowRotations**Type**

Multiple Choice

Default value

Auto

Options

[Auto, None, All, X, Y, Z, XY, XZ, YZ]

Description

Which overall rotations of the system are allowed

AllowTranslations**Type**

Multiple Choice

Default value

Auto

Options

[Auto, None, All, X, Y, Z, XY, XZ, YZ]

Description

Which overall transitions of the system are allowed

Tolerance**Type**

Float

Default value

1e-06

Description

Tolerance for detecting linear molecules. A large value means larger deviation from linearity is permitted.

RNGSeed**Type**

Integer List

Description

Initial seed for the (pseudo)random number generator. This should be omitted in most calculations to avoid introducing bias into the results. If this is unset, the generator will be seeded randomly from external sources of entropy. If you want to exactly reproduce an older calculation, set this to the numbers printed in its output.

SCMatrix**Type**

Block

Description

Technical settings for programs using the AMT matrix system. Currently this is only used by DFTB

DistributedMatrix**Type**

Block

Description

Technical settings for Distributed matrices

ColBlockSize**Type**

Integer

Default value

64

Description

See comment of RowBlockSize.

RowBlockSize**Type**

Integer

Default value

64

Description

The matrix is divided into blocks of size RowBlockSize x ColBlockSize. The smaller the blocks the better the distribution, but at the expense of increased communication overhead

Type**Type**

Multiple Choice

Default value

Elpa

Options

[Auto, Reference, ScaLapack, Elpa]

Description

Determines which implementation is used to support the AbstractMatrixType.

Symmetry**Type**

Block

Description

Specifying details about the details of symmetry detection and usage.

SymmetrizeTolerance**Type**

Float

Default value

0.05

Description

Tolerance used to detect symmetry in case symmetrize is requested.

Tolerance**Type**

Float

Default value

1e-07

Description

Tolerance used to detect symmetry in the system.

System**Type**

Block

Recurring

True

Description

Specification of the chemical system. For some applications more than one system may be present in the input. In this case, all systems except one must have a non-empty string ID specified after the System keyword. The system without an ID is considered the main one.

AllowCloseAtoms**Type**

Bool

Default value

No

Description

If AllowCloseAtoms is set to False, the AMS driver will stop with an error if it detects almost-coinciding atomic coordinates. If set to True, the AMS driver will try to carry on with the calculation.

Atoms

Type

Non-standard block

Description

The atom types and coordinates. Unit can be specified in the header. Default unit is Angstrom.

BondOrders**Type**

Non-standard block

Description

Defined bond orders. Each line should contain two atom indices, followed by the bond order (1, 1.5, 2, 3 for single, aromatic, double and triple bonds) and (optionally) the cell shifts for periodic systems. May be used by MM engines and for defining constraints. If the system is periodic and none of the bonds have the cell shift defined then AMS will attempt to determine them following the minimum image convention.

Charge**Type**

Float

Default value

0.0

GUI name

Total charge

Description

The system's total charge in atomic units.

ElectrostaticEmbedding**Type**

Block

Description

Container for electrostatic embedding options, which can be combined.

ElectricField**Type**

Float List

Unit

V/Angstrom

Description

External homogeneous electric field with three Cartesian components: ex, ey, ez, the default unit being V/Å.

In atomic units: $\text{Hartree}/(e \text{ bohr}) = 51.422 \text{ V/Angstrom}$; the relation to SI units is: $1 \text{ Hartree}/(e \text{ bohr}) = 5.14 \dots \text{e}11 \text{ V/m}$.

Supported by the engines adf, band, dftb and mopac.

For periodic systems the field may only have nonzero components orthogonal to the direction(s) of periodicity (i.e. for 1D periodic system the x-component of the electric field should be zero, while for 2D periodic systems both the x and y components should be zero. This options cannot be used for 3D periodic systems.

MultipolePotential

Type

Block

Description

External point charges (and dipoles).

ChargeModel**Type**

Multiple Choice

Default value

Point

Options

[Point, Gaussian]

Description

A multipole may be represented by a point (with a singular potential at its location) or by a spherical Gaussian distribution.

ChargeWidth**Type**

Float

Default value

-1.0

Description

The width parameter in a.u. in case a Gaussian charge model is chosen. A negative value means that the width will be chosen automatically.

Coordinates**Type**

Non-standard block

Description

Positions and values of the multipoles, one per line. Each line has the following format:

x y z q, or

x y z q μ_x μ_y μ_z .

Here x, y, z are the coordinates in Å, q is the charge (in atomic units of charge) and μ_x , μ_y , μ_z are the (optional) dipole moment components (in atomic units, i.e. e*Bohr).

Periodic systems are not supported.

FractionalCoords**Type**

Bool

Default value

No

Description

Whether the atomic coordinates in the Atoms block are given in fractional coordinates of the lattice vectors. Requires the presence of the Lattice block.

GeometryFile

Type

String

Description

Read the geometry from a file (instead of from Atoms and Lattice blocks). Supported formats: .xyz

GuessBonds**Type**

Bool

Default value

No

Description

Whether or not UFF bonds should be guessed.

Lattice**Type**

Non-standard block

Description

Up to three lattice vectors. Unit can be specified in the header. Default unit is Angstrom.

LatticeStrain**Type**

Float List

Description

Deform the input system by the specified strain. The strain elements are in Voigt notation, so one should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

LoadForceFieldAtomTypes**Type**

Block

Description

This is a mechanism to set the ForceField.Type attribute in the input. This information is currently only used by the ForceField engine.

File**Type**

String

Description

Name of the (kf) file. It needs to be the result of a forcefield calculation.

LoadForceFieldCharges**Type**

Block

Recurring

True

Description

This is a mechanism to set the ForceField.Charge attribute in the input. This information is currently only used by the ForceField engine.

CheckGeometryRMSD**Type**

Bool

Default value

No

Description

Whether the geometry RMSD test should be performed, see MaxGeometryRMSD. Otherwise only basic tests are performed, such as number and atom types. Not doing the RMSD test allows you to load molecular charges in a periodic system.

File**Type**

String

Description

Name of the (kf) file

MaxGeometryRMSD**Type**

Float

Default value

0.1

Unit

Angstrom

Description

The geometry of the charge producing calculation is compared to the one of the region, and need to be the same within this tolerance.

Region**Type**

String

Default value

*

Description

Region for which the charges should be loaded

Section**Type**

String

Default value

AMSResults

Description

Section name of the kf file

Variable**Type**

String

Default value

Charges

Description

Variable name of the kf file

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

For periodic systems the atoms will be moved to the central cell.

ModifyAlternativeElements**Type**

Bool

Default value

No

Description

When using alternative elements (using the `nuclear_charge` attribute) set the element to the nearest integer Z . If you specify an H atom with a `nuclear_charge` of 2.9 it is replaced by a Li atom with the same nuclear charge.

PerturbCoordinates**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Perturb the atomic coordinates by adding random numbers between `[-PerturbCoordinates, PerturbCoordinates]` to each Cartesian component. This can be useful if you want to break the symmetry of your system (e.g. for a geometry optimization).

PerturbLattice**Type**

Float

Default value

0.0

Description

Perturb the lattice vectors by applying random strain with matrix elements between `[-PerturbLattice, PerturbLattice]`. This can be useful if you want to deviate from an ideal symmetric geometry, for example if you look for a phase change due to high pressure.

RandomizeAtomOrder**Type**

Bool

Default value

No

Description

Whether or not the order of the atoms should be randomly changed. Intended for some technical testing purposes only. Does not work with bond information.

Region**Type**

Block

Recurring

True

Description

Properties for each region specified in the Atoms block.

Properties**Type**

Non-standard block

Description

Properties for each region specified in the Atoms block.

ShiftCoordinates**Type**

Float List

Unit

Bohr

Description

Translate the atoms by the specified shift (three numbers).

SuperCell**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems). The integer numbers represent the diagonal elements of the supercell transformation; you should specify as many numbers as lattice vectors (i.e. 1 number for 1D, 2 numbers for 2D and 3 numbers for 3D periodic systems).

SuperCellTrafo**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems) $\vec{a}'_i = \sum_j T_{ij} \vec{a}_j$. The integer numbers represent the supercell transformation T_{ij} : 1 number for 1D PBC, 4 numbers for 2D PBC corresponding to a 2x2 matrix (order: (1,1),(1,2),(2,1),(2,2)) and 9 numbers for 3D PBC corresponding to a 3x3 matrix (order: (1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)).

Symmetrize**Type**

Bool

Default value

No

Description

Whether to symmetrize the input structure. This might also rototranslate the structure into a standard orientation. This will symmetrize the atomic coordinates to machine precision. Useful if the system is almost symmetric or to rototranslate a symmetric molecule into a standard orientation.

Symmetry**Type**

Multiple Choice

Default value

AUTO

Options

[AUTO, NOSYM, C(LIN), D(LIN), C(I), C(S), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(2V), C(3V), C(4V), C(5V), C(6V), C(7V), C(8V), C(2H), C(3H), C(4H), C(5H), C(6H), C(7H), C(8H), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(2D), D(3D), D(4D), D(5D), D(6D), D(7D), D(8D), D(2H), D(3H), D(4H), D(5H), D(6H), D(7H), D(8H), I, I(H), O, O(H), T, T(D), T(H), S(4), S(6), S(8)]

Description

Use (sub)symmetry with this Schoenflies symbol. Can only be used for molecules. Orientation should be correct for the (sub)symmetry. If used icw Symmetrize, the symmetrization will not reorient the molecule.

Task**Type**

Multiple Choice

Options

[GCMC, GeometryOptimization, IRC, MolecularDynamics, NEB, PESExploration, PESScan, Replay, SinglePoint, TransitionStateSearch, VibrationalAnalysis]

Description

Specify the computational task to perform:

- Single Point: keep geometry as is
- Geometry Optimization: minimize energy
- Transition State: search for a transition state
- IRC: intrinsic reaction coordinate
- PES Scan: scan the potential energy surface
- NEB: Nudged elastic band for reaction path optimization
- Vibrational Analysis: perform one of the analysis types selected on the options page
- Molecular Dynamics: perform MD simulation
- GCMC: Grand Canonical Monte Carlo simulation
- PES Exploration: automated potential energy surface exploration
- Replay: recompute frames from the trajectory of a previously run job

Thermo**Type**

Block

Description

Options for thermodynamic properties (assuming an ideal gas). The properties are computed for all specified temperatures.

LowFrequencyCorrector**Type**

Block

Description

Options for the dampener-powered free rotor interpolator that corrects thermodynamic quantities for low frequencies. See DOI:10.1021/jp509921r and DOI:10.1002/chem.201200497.

Alpha**Type**

Float

Default value

4.0

Description

The exponent term used in the dampener.

Frequency**Type**

Float

Default value

100.0

Unit

cm-1

Description

The frequency around which the dampener interpolates between harmonic oscillator and free rotor quantities.

MomentOfInertia**Type**

Float

Default value

1e-44

Unit

kg m²

GUI name

Averaging Moment of Inertia

Description

The moment of inertia used to restrict entropy results for very small frequencies (generally around less than 1 cm-1).

Pressure**Type**

Float

Default value

1.0

Unit

atm

Description

The pressure at which the thermodynamic properties are computed.

Temperatures**Type**

Float List

Default value

[298.15]

Unit

Kelvin

Value Rangevalue ≥ 0 **Description**

List of temperatures at which the thermodynamic properties will be calculated.

TransitionStateSearch**Type**

Block

Description

Configures some details of the transition state search.

ModeToFollow**Type**

Integer

Default value

1

Description

In case of Transition State Search, here you can specify the index of the normal mode to follow (1 is the mode with the lowest frequency).

ReactionCoordinate**Type**

Block

Description

Specify components of the transition state reaction coordinate (TSRC) as a linear combination of internal coordinates (distances or angles).

Angle**Type**

String

Recurring

True

Description

The TSRC contains the valence angle between the given atoms. Three atom indices followed by the weight.

Block

Type

String

Recurring

True

Description

Name of the region. Only atoms of the region will be included in the TSRC. It is useful when computing the reaction coordinate from the initial Hessian, in which case only part of the Hessian will be analyzed.

BlockAtoms**Type**

Integer List

Value Range

value > 0

Recurring

True

Description

List of atom indices. Only the listed atoms will be included in the TSRC. It is useful when computing the reaction coordinate from the initial Hessian, in which case only part of the Hessian will be analyzed.

Coordinate**Type**

String

Recurring

True

Description

The TSRC contains Cartesian displacement of an atom: atom index followed by [x|y|z] and the weight.

Dihedral**Type**

String

Recurring

True

Description

The TSRC contains the dihedral angle between the given atoms. Four atom indices followed by the weight.

Distance**Type**

String

Recurring

True

Description

The TSRC contains the distance between the given atoms. Two atom indices followed by the weight.

UseSymmetry

Type

Bool

Default value

Yes

Description

Whether to use the system's symmetry in AMS. Symmetry is recognized within a tolerance as given in the Symmetry key.

VibrationalAnalysis**Type**

Block

Description

Input data for all vibrational analysis utilities in the AMS driver.

AbsorptionSpectrum**Type**

Block

Description

Settings related to the integration of the spectrum for vibronic tasks.

AbsorptionRange**Type**

Float List

Default value

[-200.0, 4000.0]

Unit

cm-1

Recurring

True

Description

Specifies frequency range of the vibronic absorption spectrum to compute. 2 numbers: an upper and a lower bound.

FrequencyGridPoints**Type**

Integer

Default value

400

Description

Number of grid points to use for the spectrum

LineWidth**Type**

Float

Default value

200.0

Unit

cm-1

Description

Lorentzian line-width.

SpectrumOffset**Type**

Multiple Choice

Default value

relative

Options

[absolute, relative]

Description

Specifies whether provided frequency range are absolute frequencies or frequencies relative to computed 0-0 excitation energy.

Displacement**Type**

Float

Unit

Bohr

Description

Step size for finite difference calculations.

ExcitationSettings**Type**

Block

Description

Block that contains settings related to the excitation for vibronic tasks.

EnergyInline**Type**

Float

Unit

hartree

Description

Vertical excitation energy, used when [ExcitationInfo] = [Inline].

ExcitationFile**Type**

String

Description

Path to a .rkf/.t21 file containing the excited state information (gradients, transition dipoles and energies).

ExcitationInputFormat**Type**

Multiple Choice

Default value

File

Options

[File, Inline]

Description

Select how the application should retrieve the excited state information (energy, gradient).

GradientInline**Type**

Non-standard block

Description

Excited state gradient at ground state equilibrium geometry, used when [ExcitationInfo] = [Inline].

Singlet**Type**

Non-standard block

Description

Symmetry labels + integer indices of desired singlet transitions (VG-FC absorption spectra support only 1 at a time)

Triplet**Type**

Non-standard block

Description

Symmetry labels + integer indices of desired triplet transitions (VG-FC absorption spectra support only 1 at a time)

ModeTracking**Type**

Block

Description

Input data for Mode Tracking.

HessianGuess**Type**

Multiple Choice

Default value

CalculateWithFastEngine

Options

[Unit, File, CalculateWithFastEngine]

GUI name

Guess Hessian

Description

Sets how to obtain the guess for the Hessian used in the preconditioner (if one is to be used).

HessianInline**Type**

Non-standard block

Description

Initial guess for the (non-mass-weighted) Hessian in a $3N \times 3N$ block, used when [HessianGuess] = [Inline].

HessianPath**Type**

String

Description

Path to a .rkf file containing the initial guess for the Hessian, used when [HessianGuess] = [File]. It may also be the name of the results folder containing the engine file.

ToleranceForBasis**Type**

Float

Default value

0.0001

Description

Convergence tolerance for the contribution of the newest basis vector to the tracked mode.

ToleranceForNorm**Type**

Float

Default value

0.0005

Description

Convergence tolerance for residual RMS value.

ToleranceForResidual**Type**

Float

Default value

0.0005

Description

Convergence tolerance for the maximum component of the residual vector.

ToleranceForSpectrum**Type**

Float

Default value

0.01

Description

Convergence tolerance for the spectrum in Vibronic Structure Tracking.

TrackingMethod**Type**

Multiple Choice

Default value

OverlapInitial

Options

[OverlapInitial, DifferenceInitial, FreqInitial, IRInitial, OverlapPrevious, DifferencePrevious, FreqPrevious, IRPrevious, HighestFreq, HighestIR, LowestFreq, LowestResidual]

Description

Set the tracking method that will be used. Vibronic Structure Tracking uses Largest Displacement.

UpdateMethod**Type**

Multiple Choice

Options

[JD, D, I]

Description

Chooses the method for expanding the Krylov subspace: (I) No preconditioner (VST default), (D) Davidson or (JD) vdVorst-Sleijpen variant of Jacobi-Davidson (Mode tracking default).

NormalModes**Type**

Block

Description

All input related to processing of normal modes. Not available for vibronic structure tracking (as no modes are required there).

MassWeightInlineMode**Type**

Bool

Default value

Yes

Description

MODE TRACKING ONLY: The supplied modes must be mass-weighted. This tells the program to mass-weight the supplied modes in case this has not yet been done. (True means the supplied modes will be mass-weighted by the program, e.g. the supplied modes are non-mass-weighted.)

ModeFile**Type**

String

Description

Path to a .rkf or .t21 file containing the modes which are to be scanned. Which modes will be scanned is selected using the criteria from the [ModeSelect] block.) This key is optional for Resonance Raman and Vibronic Structure. These methods can also calculate the modes using the engine.

ModeInline**Type**

Non-standard block

Recurring

True

Description

MODE TRACKING ONLY: Coordinates of the mode which will be tracked in a N x 3 block (same as for atoms), used when [ModeInputFormat] = [Inline]. Rows must be ordered in the same way as in the [System%Atoms] block. Mode Tracking only.

ModeInputFormat**Type**

Multiple Choice

Default value

File

Options

[File, Inline, Hessian]

GUI name

Tracked mode source

Description

Set how the initial guesses for the modes are supplied. Only mode tracking supports the Inline and Hessian options.

ModeSelect**Type**

Block

Description

Pick which modes to read from file.

DisplacementBound**Type**

Float

Description

Vibronic Structure (Refinement), Resonance Raman: Select all modes with a dimensionless oscillator displacement greater than the specified value.

FreqAndIRRange**Type**

Float List

Recurring

True

Description

Specifies a combined frequency and IR intensity range within which all modes will be selected. First 2 numbers are the frequency range in cm⁻¹, last 2 numbers are the IR intensity range in km/mol.

FreqRange**Type**

Float List

Unit

cm⁻¹

Recurring

True

Description

Specifies a frequency range within which all modes will be selected. 2 numbers: an upper and a lower bound. Calculating all modes higher than some frequency can be achieved by making the upper bound very large.

Full**Type**

Bool

Default value

No

GUI name

All modes

Description

Select all modes. This only make sense for Mode Scanning calculations.

HighFreq**Type**

Integer

GUI name

High frequencies

Description

Select the N modes with the highest frequencies.

HighIR**Type**

Integer

GUI name

High IR

Description

Select the N modes with the largest IR intensities.

IRRange**Type**

Float List

Unit

km/mol

Recurring

True

Description

Specifies an IR intensity range within which all modes will be selected. 2 numbers: an upper and a lower bound.

ImFreq**Type**

Bool

Default value

No

GUI name

All imaginary frequencies

Description

Select all modes with imaginary frequencies.

LargestDisplacement**Type**

Integer

Description

Vibronic Structure (Refinement), Resonance Raman: Select the N modes with the largest VG-FC displacement.

LowFreq**Type**

Integer

GUI name

Low frequencies

Description

Select the N modes with the lowest frequencies. Includes imaginary modes which are recorded with negative frequencies.

LowFreqNoIm**Type**

Integer

GUI name

Low positive frequencies

Description

Select the N modes with the lowest non-negative frequencies. Imaginary modes have negative frequencies and are thus omitted here.

LowIR**Type**

Integer

GUI name

Low IR

Description

Select the N modes with the smallest IR intensities.

ModeNumber**Type**

Integer List

GUI name

Mode numbers

Description

Indices of the modes to select.

ScanModes**Type**

Bool

Default value

No

GUI name

Scan after refining

Description

Supported by: Mode Tracking, Mode Refinement, Vibronic Structure Refinement: If enabled an additional displacement will be performed along the new modes at the end of the calculation to obtain refined frequencies and IR intensities. Equivalent to running the output file of the mode tracking calculation through the AMS ModeScanning task.

ResonanceRaman**Type**

Block

Description

Block that contains settings for the calculation of Resonance Raman calculations

IncidentFrequency**Type**

Float

Unit

cm-1

Description

Frequency of incident light. Also used to determine most important excitation in case more than one is provided.

LifeTime**Type**

Float

Default value

0.00045

Unit

hartree

Description

Lifetime of Raman excited state.

RamanOrder**Type**

Integer

Default value

2

Description

Order up to which to compute Raman transitions

RamanRange**Type**

Float List

Default value

[0.0, 2000.0]

Unit

cm-1

Recurring

True

Description

Specifies frequency range of the Raman spectrum to compute. 2 numbers: an upper and a lower bound.

Type**Type**

Multiple Choice

Options

[ModeScanning, ModeTracking, ModeRefinement, VibronicStructure, VibronicStructure-Tracking, VibronicStructureRefinement, ResonanceRaman]

Description

Specifies the type of vibrational analysis that should be performed

VSTRestartFile**Type**

String

Description

Path to a .rkf file containing restart information for VST.

15.2.2 analysis

AutoCorrelation**Type**

Block

Recurring

True

Description

All input related to auto correlation functions.

Atoms**Type**

Block

Description

Relevant if Property is set to Velocities, DipoleMomentFromCharges, DipoleDerivativeFromCharges, or DiffusionCoefficient. Atom numbers or elements for the set of atoms for which the property is read/computed. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

DataReading**Type**

Multiple Choice

Default value

Auto

Options

[Auto, AtOnce, BlockWise]

Description

The KF data can be read in and handled once, or blockwise. The former is memory intensive, but mostly faster. If Auto is selected, the data is read at once if it is less than 1 GB, and blockwise if it is more.

MaxCorrelationTime**Type**

Float

Description

The maximum correlation time in fs. The default is half the simulation time, except when the PressureTensor is read from the ams.rkf, in which case it is 10 percent of the simulation time. The PressureTensor is read when the Properties PressureTensor, Viscosity, or ViscosityFromBinLog are selected.

MaxFrame**Type**

Integer

Description

The maximum number of frames for which the autocorrelation function will be computed. The default is half of the number of provided frames. Determines the same settings as MaxCorrelationTime. If both are set, MaxCorrelationTime will take precedence.

NPointsHighestFreq

Type

Integer

Default value

4

Description

The number of points (timesteps) used for the highest frequency displayed in spectrum. This determines up to which frequency the spectrum is displayed. If the spacing between time-steps used for the ACF is 1 fs, then by default the maximum frequency displayed is 0.25 fs⁻¹ (or 8339 cm⁻¹). This corresponds to a (default) value of NPointsHighestFreq of 4. A higher number selected here, will result in a lower maximum frequency returned by the program. The lowest possible value (spectrum up to highest possible frequency) is 2.

PerElement**Type**

Bool

Default value

No

Description

Compute ACF for all elements in the system. Any other settings in the block will be used.

Property**Type**

Multiple Choice

Default value

DipoleDerivativeFromCharges

Options

[Velocities, DipoleMomentFromCharges, DiffusionCoefficient, DipoleDerivativeFromCharges, PressureTensor, Viscosity, DipoleMomentFromBinLog, ViscosityFromBinLog]

Description

Compute the ACF either from velocities (from rkf), the dipole moment (from coordinates and atomic charges in rkf), the dipole moment derivative (from velocities and atomic charges in rkf), from the pressure tensor (from rkf), or from values specified in input. Selecting DiffusionCoefficient is equivalent to selecting Velocities. The default, DipoleDerivativeFromCharges, results in the computation of an IR spectrum. DipoleMomentFromBinLog and ViscosityFromBinLog allow the relevant properties (dipole moment and pressure tensor respectively) to be read from the BinLog section of the trajectory file. In the BinLog section requested properties are stored every step (even if SamplingFreq was set to a higher number than 1) but only if this was specifically requested at the start of the MD simulation.

UnwrapCoordinates**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Yes, No]

Description

If the coordinates are involved in the requested property, those coordinates are wrapped into

the box at each time step. If set to true, this keyword unwraps those coordinates so that the trajectory is continuous. If not provided the code uses automatic defaults.

UseAllValues

Type

Bool

Default value

No

Description

By default the same number of values are used for each t-step in the ACF. This has the advantage that all values in the ACF are equally reliable, but it does mean that for the smaller timesteps much of the data is not used. To switch this off and use all data, UseAllValues can be set to true

UseTimeDerivative

Type

Block

Description

Possibly use the time derivative of the selected property (e.g. velocity or dipole moments).

Enabled

Type

Bool

Default value

No

Description

Enable the use of the time derivative of the property.

VecElements

Type

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property Velocities, the Atoms block allows the selection of a subset of atoms, while the VecElements block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index

Type

Integer

Recurring

True

Description

Element of the property vector.

WritePropertyToKF

Type

Bool

Default value

No

Description

Write the selected property to the KF files for every requested frame

AverageBinPlot**Type**

Block

Recurring

True

Description

All input related to velocity profile

Atoms**Type**

Block

Description

Relevant if Properties are atom dependent. Atom numbers or elements for the set of atoms for which the property is read/computed. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

Nbins**Type**

Integer

Default value

10

Description

Number of bins that are plotted

Property**Type**

Block

Description

Property to be plotted along the Y-axis

Axis**Type**

Float List

Description

If defined the dot_product along this axis will be taken. Otherwise, the length of the property vector will be used.

Name**Type**

Multiple Choice

Options

[FrictionCoefficient, Viscosity, Velocities, EngineGradients]

Description

Name of the property

XProperty**Type**

Block

Description

Property to be plotted along the Y-axis

Name**Type**

Multiple Choice

Default value

Time

Options

[Time, Coords]

Description

Timestep used for the plotting

VecElements**Type**

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property Velocities, the Atoms block

allows the selection of a subset of atoms, while the VecElements block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index**Type**

Integer

Default value

3

Description

Element of the x_property, in case it is a vector (For Coords: 1 for X, 2 for Y, 3 for Z).

Histogram**Type**

Block

Recurring

True

Description

All input related to histograms.

Axes**Type**

Block

Description

Specifications for the histogram axes.

Axis**Type**

Block

Recurring

True

Description

Specifications for a single histogram axis.

Atoms**Type**

Block

Description

Relevant if variable has a value per atom (e.g. Coords, Velocities). This block specifies indices or elements for the set of atoms for which the variable is to be read. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom index.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name

NBins**Type**

Integer

Default value

100

Description

The number of bins along the histogram axis.

Range**Type**

Float List

Description

Either one, two, or three real values. If one it is the stepsize. If two, it is the minimum value and the maximum value. If three, it is the minimum value, the maximum value, and the stepsize. The stepsize overrides NBins.

Variable**Type**

String

Description

The quantity along the histogram axis.

VecElements**Type**

Block

Description

A set of indices referring to a subset of a vector. If the variable to be plotted has non-scalar values per step, then this block allows the selection of a subset of vector elements (e.g. 1 and 2 for the x and y values). Can be used in combination with the Atoms block.

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

KeepRemainder**Type**

Bool

Default value

No

Description

Place the values that fall outside the range in an extra bin (on the right).

Normalized**Type**

Bool

Default value

No

Description

Give the normalized histogram.

LoadSystem**Type**

Block

Recurring

True

Description

Block that controls reading the chemical system from a KF file instead of the [System] block.

File**Type**

String

Description

The path of the KF file from which to load the system. It may also be the results directory containing it.

Section**Type**

String

Default value

Molecule

Description

The section on the KF file from which to load the system.

MeanSquareDisplacement**Type**

Block

Recurring

True

Description

All input related to mean square displacement functions.

Atoms**Type**

Block

Description

Relevant if Property is set to any quantity that is available per atom (Coords, DiffusionCoefficient). Atom numbers or elements for the set of atoms for which the property is read/computed are provided here. By default all atoms are used.

Atom**Type**

Integer

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

DataReading**Type**

Multiple Choice

Default value

Auto

Options

[Auto, AtOnce, BlockWise]

Description

The KF data can be read in and handled once, or blockwise. The former is memory intensive, but mostly faster. If Auto is selected, the data is read at once if it is less than 1 GB, and blockwise if it is more.

MaxCorrelationTime**Type**

Float

Description

The maximum correlation time in fs. The default is half the simulation time.

MaxFrame**Type**

Integer

Description

The maximum number of frames for which the mean square displacement function will be computed. The default is half of the number of provided frames. Determines the same settings as MaxCorrelationTime. If both are set, MaxCorrelationTime will take precedence.

PerElement**Type**

Bool

Default value

No

Description

Compute MSD for all elements in the system. Any other settings in this block will be used.

Property**Type**

Multiple Choice

Default value

Coords

Options

[Coords, DiffusionCoefficient, Conductivity]

Description

Compute the MSD from the property selected here (from rkf). Selecting DiffusionCoefficient is equivalent to selecting the property Coords.

StartTimeSlope**Type**

Float

Default value

0.0

Description

The MSD has a nonlinear regime at short timescales, and a linear regime at long timescales. To determine the slope, the starting point for the linear regime has to be determined. This keyword sets the starting time in fs. If set to zero, the starttime will be automatically determined.

UnwrapCoordinates**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Yes, No]

Description

If the coordinates are involved in the requested property, those coordinates are wrapped into the box at each time step. If set to true, this keyword unwraps those coordinates so that the trajectory is continuous. If not provided the code uses automatic defaults.

UseAllValues**Type**

Bool

Default value

No

Description

By default the same number of values are used for each t-step in the MSD. This has the advantage that all values in the MSD are equally reliable, but it does mean that for the smaller timesteps much of the data is not used. To switch this off and use all data, UseAllValues can be set to true

UseMolecularCentersOfMass**Type**

Block

Description

Specifications on using the center of mass coordinates per molecule instead of coordinates per atom. Only relevant for properties that rely on coordinates: Coords, DiffusionCoefficient, and Conductivity.

CheckForBondChanges**Type**

Bool

Default value

Yes

Description

Check for each frame if the bonds still correspond to the input bonds. If they changed, an error is thrown. If this is set to false, it is assumed that the molecules stay in tact.

Enabled**Type**

Bool

Default value

No

Description

Use the center of mass coordinates per molecule. Only relevant for properties that rely on coordinates: Coords, DiffusionCoefficient, and Conductivity. The bonds from the MD input system are used.

VecElements**Type**

Block

Description

A set of indices referring to a subset of the property vector. Works in combination with the atoms block. For example, in combination with the property Coords, the Atoms block allows the selection of a subset of atoms, while the VecElements block allows the selection of a subset of vector elements (e.g. 1 and 2 for the elements x and y).

Index**Type**

Integer

Recurring

True

Description

Element of the property vector.

WritePropertyToKF**Type**

Bool

Default value

No

Description

Write the selected property to the KF files for every requested frame

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

Timers**Type**

Multiple Choice

Default value

None

Options

[None, Normal, Detail, TooMuchDetail]

Description

Printing timing details to see how much time is spend in which part of the code.

RadialDistribution**Type**

Block

Recurring

True

Description

All input related to radial distribution functions.

AtomsFrom**Type**

Block

Description

Atom numbers or elements for the first set of atoms in the radial distribution.

Atom**Type**

Integer

Default value

0

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

AtomsTo**Type**

Block

Description

Atom numbers or elements for the second set of atoms in the radial distribution.

Atom**Type**

Integer

Default value

0

Recurring

True

Description

Atom number.

Element**Type**

String

Recurring

True

Description

Element Symbol Atom.

Region**Type**

String

Recurring

True

Description

Region name.

DistanceTypeSelection**Type**

Multiple Choice

Default value

All

Options

[All, InterMolecular, IntraMolecular]

Description

Select only a certain type of interatomic distances.

KeepRemainder**Type**

Bool

Default value

No

Description

Place the values that fall outside the range in an extra bin (on the right).

NBins**Type**

Integer

Default value

1000

Description

The number of bins in the histogram.

PairwisePerElement**Type**

Bool

Default value

No

Description

Compute RDF for all element pairs (for all atoms in the system). Any other settings in the block will be used.

Range**Type**

Float List

Description

Either one, two, or three real values. If one it is the stepsize. If two, it is the minimum value and the maximum value. If three, it is the minimum value, the maximum value, and the stepsize. The stepsize overrides NBins.

System**Type**

Block

Recurring

True

Description

Specification of the chemical system. For some applications more than one system may be present in the input. In this case, all systems except one must have a non-empty string ID specified after the System keyword. The system without an ID is considered the main one.

AllowCloseAtoms**Type**

Bool

Default value

No

Description

If AllowCloseAtoms is set to False, the AMS driver will stop with an error if it detects almost-coinciding atomic coordinates. If set to True, the AMS driver will try to carry on with the calculation.

Atoms**Type**

Non-standard block

Description

The atom types and coordinates. Unit can be specified in the header. Default unit is Angstrom.

BondOrders**Type**

Non-standard block

Description

Defined bond orders. Each line should contain two atom indices, followed by the bond order (1, 1.5, 2, 3 for single, aromatic, double and triple bonds) and (optionally) the cell shifts for periodic systems. May be used by MM engines and for defining constraints. If the system is periodic and none of the bonds have the cell shift defined then AMS will attempt to determine them following the minimum image convention.

Charge**Type**

Float

Default value

0.0

GUI name

Total charge

Description

The system's total charge in atomic units.

ElectrostaticEmbedding**Type**

Block

Description

Container for electrostatic embedding options, which can be combined.

ElectricField**Type**

Float List

Unit

V/Angstrom

Description

External homogeneous electric field with three Cartesian components: ex, ey, ez, the default unit being V/Å.

In atomic units: Hartree/(e bohr) = 51.422 V/Angstrom; the relation to SI units is: 1 Hartree/(e bohr) = 5.14 ... e11 V/m.

Supported by the engines adf, band, dftb and mopac.

For periodic systems the field may only have nonzero components orthogonal to the direction(s) of periodicity (i.e. for 1D periodic system the x-component of the electric field should be zero, while for 2D periodic systems both the x and y components should be zero. This options cannot be used for 3D periodic systems.

MultipolePotential**Type**

Block

Description

External point charges (and dipoles).

ChargeModel**Type**

Multiple Choice

Default value

Point

Options

[Point, Gaussian]

Description

A multipole may be represented by a point (with a singular potential at its location) or by a spherical Gaussian distribution.

ChargeWidth**Type**

Float

Default value

-1.0

Description

The width parameter in a.u. in case a Gaussian charge model is chosen. A negative value means that the width will be chosen automatically.

Coordinates**Type**

Non-standard block

Description

Positions and values of the multipoles, one per line. Each line has the following format:

x y z q, or

x y z q μ_x μ_y μ_z .

Here x, y, z are the coordinates in Å, q is the charge (in atomic units of charge) and μ_x , μ_y , μ_z are the (optional) dipole moment components (in atomic units, i.e. e*Bohr).

Periodic systems are not supported.

FractionalCoords**Type**

Bool

Default value

No

Description

Whether the atomic coordinates in the Atoms block are given in fractional coordinates of the lattice vectors. Requires the presence of the Lattice block.

GeometryFile**Type**

String

Description

Read the geometry from a file (instead of from Atoms and Lattice blocks). Supported formats: .xyz

GuessBonds**Type**

Bool

Default value

No

Description

Whether or not UFF bonds should be guessed.

Lattice**Type**

Non-standard block

Description

Up to three lattice vectors. Unit can be specified in the header. Default unit is Angstrom.

LatticeStrain

Type

Float List

Description

Deform the input system by the specified strain. The strain elements are in Voigt notation, so one should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

LoadForceFieldAtomTypes**Type**

Block

Description

This is a mechanism to set the ForceField.Type attribute in the input. This information is currently only used by the ForceField engine.

File**Type**

String

Description

Name of the (kf) file. It needs to be the result of a forcefield calculation.

LoadForceFieldCharges**Type**

Block

Recurring

True

Description

This is a mechanism to set the ForceField.Charge attribute in the input. This information is currently only used by the ForceField engine.

CheckGeometryRMSD**Type**

Bool

Default value

No

Description

Whether the geometry RMSD test should be performed, see MaxGeometryRMSD. Otherwise only basic tests are performed, such as number and atom types. Not doing the RMSD test allows you to load molecular charges in a periodic system.

File**Type**

String

Description

Name of the (kf) file

MaxGeometryRMSD**Type**

Float

Default value

0.1

Unit

Angstrom

Description

The geometry of the charge producing calculation is compared to the one of the region, and need to be the same within this tolerance.

Region**Type**

String

Default value

*

Description

Region for which the charges should be loaded

Section**Type**

String

Default value

AMSResults

Description

Section name of the kf file

Variable**Type**

String

Default value

Charges

Description

Variable name of the kf file

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

For periodic systems the atoms will be moved to the central cell.

ModifyAlternativeElements**Type**

Bool

Default value

No

Description

When using alternative elements (using the nuclear_charge attribute) set the element to the

nearest integer Z. If you specify an H atom with a nuclear_charge of 2.9 it is replaced by a Li atom with the same nuclear charge.

PerturbCoordinates**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Perturb the atomic coordinates by adding random numbers between [-PerturbCoordinates,PerturbCoordinates] to each Cartesian component. This can be useful if you want to break the symmetry of your system (e.g. for a geometry optimization).

PerturbLattice**Type**

Float

Default value

0.0

Description

Perturb the lattice vectors by applying random strain with matrix elements between [-PerturbLattice,PerturbLattice]. This can be useful if you want to deviate from an ideal symmetric geometry, for example if you look for a phase change due to high pressure.

RandomizeAtomOrder**Type**

Bool

Default value

No

Description

Whether or not the order of the atoms should be randomly changed. Intended for some technical testing purposes only. Does not work with bond information.

Region**Type**

Block

Recurring

True

Description

Properties for each region specified in the Atoms block.

Properties**Type**

Non-standard block

Description

Properties for each region specified in the Atoms block.

ShiftCoordinates

Type

Float List

Unit

Bohr

Description

Translate the atoms by the specified shift (three numbers).

SuperCell**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems). The integer numbers represent the diagonal elements of the supercell transformation; you should specify as many numbers as lattice vectors (i.e. 1 number for 1D, 2 numbers for 2D and 3 numbers for 3D periodic systems).

SuperCellTrafo**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems) $\vec{a}'_i = \sum_j T_{ij} \vec{a}_j$. The integer numbers represent the supercell transformation T_{ij} : 1 number for 1D PBC, 4 numbers for 2D PBC corresponding to a 2x2 matrix (order: (1,1),(1,2),(2,1),(2,2)) and 9 numbers for 3D PBC corresponding to a 3x3 matrix (order: (1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)).

Symmetrize**Type**

Bool

Default value

No

Description

Whether to symmetrize the input structure. This might also rototranslate the structure into a standard orientation. This will symmetrize the atomic coordinates to machine precision. Useful if the system is almost symmetric or to rototranslate a symmetric molecule into a standard orientation.

Symmetry**Type**

Multiple Choice

Default value

AUTO

Options

[AUTO, NOSYM, C(LIN), D(LIN), C(I), C(S), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(2V), C(3V), C(4V), C(5V), C(6V), C(7V), C(8V), C(2H), C(3H), C(4H), C(5H), C(6H), C(7H), C(8H), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(2D), D(3D), D(4D), D(5D), D(6D), D(7D), D(8D), D(2H), D(3H), D(4H), D(5H), D(6H), D(7H), D(8H), I, I(H), O, O(H), T, T(D), T(H), S(4), S(6), S(8)]

Description

Use (sub)symmetry with this Schoenflies symbol. Can only be used for molecules. Orientation

should be correct for the (sub)symmetry. If used icw Symmetrize, the symmetrization will not reorient the molecule.

Task**Type**

Multiple Choice

Options

[RadialDistribution, Histogram, AutoCorrelation, MeanSquareDisplacement, AverageBinPlot]

Description

The analysis task.

TrajectoryInfo**Type**

Block

Description

All the info regarding the reading of the trajectory files.

NBlocksToCompare**Type**

Integer

Default value

1

Description

Get an error estimate by comparing histograms for NBlocks time blocks of the trajectory.

Trajectory**Type**

Block

Recurring

True

Description

All info regarding the reading of a single trajectory file.

KFFilename**Type**

String

Default value

ams.rkf

Description

The name of the AMS trajectory file.

Range**Type**

Integer List

Description

One or two values: start frame, and optionally end frame. By default the first and last frame are read.

StepSize

Type

Integer

Default value

1

Description

The step size at which frames are read from the RKF (default 1, every frame is read).

15.2.3 chemtrayzer2**Analysis****Type**

Block

Description

Statistical post-detection analysis, includes reaction coefficients calculation.

PerformAnalysis**Type**

Bool

Default value

Yes

Description

Determine the reaction rate coefficients and statistical errors for the detected reactions.

RateConfidence**Type**

Float

Default value

0.9

Description

Upper and lower bounds to the rate coefficients will be calculated for this confidence ($0 < \text{confidence} < 1$), assuming a Poisson distribution of the number of reactive events. A value of 0.9 means that the kinetics of 90% of events of one reaction can be described by a coefficient between the bounds.

MoleculeIdentifier**Type**

Block

Description

Settings for the subgraph identification of molecules and reactions.

MaxDepth**Type**

Integer

Default value

2

Description

The maximum number of layers the algorithm goes along bonds, starting from one atom when

generating hashes for one atom. The entire molecule hash is built from the atom hashes, so this setting influences the identification of atom neighborhoods.

UseBondOrders**Type**

Bool

Default value

No

Description

Consider bond orders in the identifier.

UseHs**Type**

Bool

Default value

No

GUI name

Use Hs

Description

Consider number of hydrogens of atoms in the identifier.

UseRings**Type**

Bool

Default value

Yes

Description

Consider ring membership of atoms in the identifier.

WindowDepth**Type**

Integer

Default value

5

Description

The maximum number of layers the algorithm goes along bonds starting from the reactive atoms when generating hashes for the entire molecule. With this setting, the identifier can be limited to only a part of the molecule.

Output**Type**

Block

Description

Settings for program output and output file generation.

CreateLegacyOutput**Type**

Bool

Default value

No

Description

Whether to save the reactions, species, and rates as 'reac.reac.tab', 'reac.spec.tab', and 'reac.rate.tab' in the same format as ChemTraYzer 1.

ShowReactionGraph**Type**

Bool

Default value

No

Description

Whether or not to show the reaction graph at the end of the calculation. Requires the python library matplotlib to be installed.

WriteEventsPerTime**Type**

Bool

Default value

No

Description

Write two .csv files that contain the number of reactions in every frame (reaction_events_per_time.csv) and the number of bond changes in every frame(bond_change_events_per_time.csv)

WriteKF**Type**

Bool

Default value

No

Description

Whether to write output to KF

WriteMolPopulation**Type**

Bool

Default value

No

Description

Write two .csv files: (1) mol_statistics.csv, which contains basic population statistics (counts, averages) for each unique species over the entire trajectory; and (2) mol_population.csv, which provides the count of each unique species in every frame.

WriteReactions**Type**

Bool

Default value

Yes

Description

Write two .csv files that contain information about (1) all unique reactions (reactions.csv); and (2) all individual reaction events (reaction_events.csv).

WriteXYZFiles**Type**

Bool

Default value

No

Description

Write XYZ files (geometries) for detected species and XYZ movies for detected reactions into a subfolder named 'xyz'.

PrintDebug**Type**

Bool

Default value

No

Description

Print extra debug information to the terminal.

ReactionDetection**Type**

Block

Description

Parameters for the the reaction detection algorithm.

BondBreakingThreshold**Type**

Float

Default value

0.3

Description

The bond-order threshold for bond breaking. If the bond order of a bond goes below this value, the bond is considered broken.

BondFormationThreshold**Type**

Float

Default value

0.8

Description

The bond-order threshold for bond formation. If the bond order between two atoms goes above this value, then this will be considered to be a new bond.

InitialBondThreshold**Type**

Float

Description

The bond-order threshold for determining the connectivity for the first frame of the simulation.
If not specified, the value in BondFormationThreshold will be used instead.

TStable**Type**

Float

Default value

10.0

Unit

fs

GUI name

T stable

Description

The minimum time for a molecule to be considered stable.

Trajectory**Type**

Block

Description

Info regarding the trajectory to analyze.

FinalFrame**Type**

Integer

Default value

-1

Description

Last frame of the trajectory to analyze.

FirstFrame**Type**

Integer

Default value

1

Description

First frame of the trajectory to analyze.

Path**Type**

String

Description

The path to ams results dir of an AMS calculation. This folder must contain a ams.rkf file.

15.2.4 conformers

Constraints

Type

Block

Description

The Constraints block allows geometry optimizations and potential energy surface scans with constraints. The constraints do not have to be satisfied at the start of the calculation.

All**Type**

String

Recurring

True

Description

Fix multiple distances using one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 as well as the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then any bond between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk "*" denoting any atom. If the distance is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at the start of the simulation can be constrained, which means that the bonds may need to be specified in the System block.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

Angle**Type**

String

Recurring

True

Description

Fix the angle between three atoms. Three atom indices followed by an angle in degrees.

Atom**Type**

Integer

Recurring

True

Description

Fix the position of an atom. Just one integer referring to the index of the atom in the [System%Atoms] block.

AtomList**Type**

Integer List

Recurring

True

Description

Fix positions of the specified atoms. A list of integers referring to indices of atoms in the [System%Atoms] block.

Block**Type**

String

Recurring

True

Description

Name of the region to constrain as a rigid block. Regions are specified in the System%Atoms block.

BlockAtoms**Type**

Integer List

Recurring

True

Description

List of atom indices for a block constraint, where the internal degrees of freedom are frozen.

Coordinate**Type**

String

Recurring

True

Description

Fix a particular coordinate of an atom. Atom index followed by (x|y|z).

DifDist**Type**

String

Recurring

True

Description

Four atom indices i j k l followed by the distance in Angstrom. This will constrain the difference $R(ij)-R(kl)$ at the given value.

Dihedral**Type**

String

Recurring

True

Description

Fix the dihedral angle between four atoms. Four atom indices followed by an angle in degrees.

Distance**Type**

String

Recurring

True

Description

Fix the distance between two atoms. Two atom indices followed by the distance in Angstrom.

EqualStrain**Type**

String

Description

Exclusively for lattice optimizations:

Accepts a set of strain components [xx, xy, xz, yy, yz, zz] which are to be kept equal.

The applied strain will be determined by the average of the corresponding stress tensors components.

In AMSinput just check the corresponding check buttons.

FixedRegion**Type**

String

Recurring

True

Description

Fix positions of all atoms in a region.

FreezeStrain**Type**

String

Description

Exclusively for lattice optimizations:

Freezes any lattice deformation corresponding to a particular component of the strain tensor.

Accepts a set of strain components [xx, xy, xz, yy, yz, zz] to be frozen.

In AMSinput just check the corresponding check buttons.

SumDist**Type**

String

Recurring

True

Description

Four atom indices i j k l followed by the distance in Angstrom. This will constrain the sum $R(ij)+R(kl)$ at the given value.

Engine**Type**

Block

Description

The input for the computational engine used to compute energy and forces.

EngineAddons**Type**

Block

Description

This block configures all the engine add-ons.

AtomEnergies**Type**

Non-standard block

Description

Add an element-dependent energy per atom. On each line, give the chemical element followed by the energy (in atomic units).

D3Dispersion**Type**

Block

Description

This block configures the add-on that adds the Grimme D3 dispersion correction to the engine's energy, gradients, and stress tensor.

Damping**Type**

Multiple Choice

Default value

BJ

Options

[BJ, Zero]

Description

Type of damping: BJ (Becke-Johnson) or Zero. BJ is recommended for most applications.

Enabled**Type**

Bool

Default value

No

GUI name

D3 dispersion

Description

Enables the D3 dispersion correction addon.

Functional**Type**

String

Default value

PBE

Description

Use the D3 parameterization by Grimme for a given xc-functional. Accepts the same values as the `-func` command line option of the official `dftd3` program. Note: the naming convention is different from elsewhere in the AMS suite. For example, BLYP should be called `b-lyp`.

a1**Type**

Float

Description

The `a1` parameter. Only used if Damping is set to BJ. If set, it overwrites the `a1` value for the chosen functional.

a2**Type**

Float

Description

The `a2` parameter. Only used if Damping is set to BJ. If set, it overwrites the `a2` value for the chosen functional.

s6**Type**

Float

Description

The `s6` parameter, global scaling parameter. If set, it overwrites the `s6` value for the chosen functional.

s8**Type**

Float

Description

The `s8` parameter. If set, it overwrites the `s8` value for the chosen functional.

sr6**Type**

Float

Description

The `sr6` parameter. Only used if Damping is set to Zero. If set, it overwrites the `sr6` value for the chosen functional.

D4Dispersion**Type**

Block

Description

This block configures the addon that adds the Grimme D4(EEQ) dispersion correction to the engine's energy, gradients, stress tensor and Hessian.

Enabled**Type**

Bool

Default value

No

GUI name

D4 dispersion

Description

Enables the D4 dispersion correction addon.

Functional**Type**

Multiple Choice

Default value

PBE

Options

[HF, BLYP, BPBE, BP86, BPW, LB94, MPWLYP, MPWPW91, OLYP, OPBE, PBE, RPBE, REVPBE, PW86PBE, RPW86PBE, PW91, PW91P86, XLYP, B97, TPSS, REVTPSS, SCAN, B1LYP, B3LYP, BHLYP, B1P86, B3P86, B1PW91, B3PW91, O3LYP, REVPBE0, REVPBE38, PBE0, PWP1, PW1PW, MPW1PW91, MPW1LYP, PW6B95, TPSSH, TPSS0, X3LYP, M06L, M06, OMEGAB97, OMEGAB97X, CAM-B3LYP, LC-BLYP, LH07TSVWN, LH07SSVWN, LH12CTSSIRPW92, LH12CTSSIFPW92, LH14TCALPBE, B2PLYP, B2GPPLYP, MPW2PLYP, PWPB95, DSDBLYP, DSDPBE, DSDPBEB95, DSDPBEP86, DSDSVWN, DODBLYP, DODPBE, DODPBEB95, DODPBEP86, DODSVWN, PBE02, PBE0DH, DFTB(3ob), DFTB(mio), DFTB(pbc), DFTB(matsci), DFTB(ob2), B1B95, MPWB1K, REVTPSSH, GLYP, REVPBE0DH, REVTPSS0, REVDSDPBEP86, REVDSDPBEPBE, REVDSDBLYP, REVDODPBEP86, B97M, OMEGAB97M, R2SCAN]

Description

Use the D4 parameterization by Grimme for a given xc-functional.

Verbosity**Type**

Multiple Choice

Default value

Silent

Options

[Silent, Normal, Verbose, VeryVerbose]

Description

Controls the verbosity of the dftd4 code. Equivalent to the `-silent` and `-verbose` command line switches of the official dftd4 program.

a1**Type**

Float

Description

The a1 parameter, see D4 article. The physically reasonable range for a1 is [0.0,1.0]. If set, it overwrites the a1 value for the chosen functional.

a2**Type**

Float

Description

The a2 parameter, see D4 article. The physically reasonable range for a2 is [0.0,7.0]. If set, it overwrites the a2 value for the chosen functional.

s6**Type**

Float

Description

The s6 parameter, see D4 article. The physically reasonable range for s6 is [0.0,1.0]. If set, it overwrites the s6 value for the chosen functional.

s8**Type**

Float

Description

The s8 parameter, see D4 article. The physically reasonable range for s8 is [0.0,3.0]. If set, it overwrites the s8 value for the chosen functional.

s9**Type**

Float

Description

The s9 parameter, see D4 article. If set, it overwrites the s9 value for the chosen functional.

ExternalEngine**Type**

Block

Description

External engine as an add-on

Execute**Type**

String

GUI name

Execute

Description

execute command

ExternalStress**Type**

Block

Description

This block configures the addon that adds external stress term to the engine's energy and stress tensor.

StressTensorVoigt**Type**

Float List

Unit

Hartree/Bohr³

GUI name

External stress tensor

Description

The elements of the external stress tensor in Voigt notation. One should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

UpdateReferenceCell**Type**

Bool

Default value

No

Description

Whether or not the reference cell should be updated every time the system changes (see documentation).

PipeEngine**Type**

Block

Description

Pipe engine as an addon

WorkerCommand**Type**

String

GUI name

Worker command

Description

pipe worker command

Pressure**Type**

Float

Default value

0.0

Unit

GPa

Description

Add a hydrostatic pressure term to the engine's energy and stress tensor. Can only be used for 3D periodic boundary conditions.

Repulsion**Type**

Block

Description

This block configures an addon that adds a repulsive Weeks-Chandler-Andersen potential to all atom pairs.

Enabled**Type**

Bool

Default value

No

GUI name

Repulsion

Description

Enables the repulsive Weeks-Chandler-Andersen potential addon.

When enabled, all atom pairs will experience repulsion $E = 4 * \epsilon * ((\sigma/r)^{12} - (\sigma/r)^6 + 1/4)$ at the distances shorter than about $1.12 * \sigma$.

Epsilon**Type**

Float

Default value

0.01

Unit

Hartree

Description

The epsilon parameter in the potential equation. It is equal to the amount of energy added at $r=\sigma$.

HydrogenSigmaScale**Type**

Float

Default value

0.75

Unit

Angstrom

Description

The sigma parameter for a pair of atoms where one of them is hydrogen is scaled with the given factor. For H-H interactions the sigma is scaled with this value squared.

Sigma**Type**

Float

Default value

0.55

Unit

Angstrom

Description

The sigma parameter in the potential equation. The potential is exactly zero at the distances larger than about $1.12 \cdot \sigma$

SkinLength**Type**

Float

Default value

2.0

Unit

Angstrom

Description

Technical parameter specifying skin length for the neighbor list generation. A larger value increases the neighbor list cutoff (and cost) but reduces the frequency it needs to be re-created.

WallPotential**Type**

Block

Description

This block configures the addon that adds a spherical wall potential to the engine's energy and gradients.

Enabled**Type**

Bool

Default value

No

Description

Enables the wall potential addon. When enabled, a spherical wall of radius [Radius] around the origin will be added. The force due to the potential will decay exponentially inside the wall, will be close to [Prefactor*Gradient] outside and exactly half of that at the wall.

Gradient**Type**

Float

Default value

10.0

Unit

1/Angstrom

Description

The radial gradient outside the sphere.

Prefactor

Type

Float

Default value

0.01

Unit

Hartree

Description

The multiplier for the overall strength of the potential.

Radius**Type**

Float

Default value

30.0

Unit

Angstrom

Value Range

value > 0

Description

The radius of the sphere, wherein the potential is close to zero.

Equivalence**Type**

Block

Description

Options for the procedure determining whether two structures are equivalent or distinct conformers.

AMS**Type**

Block

Description

Options for the AMS method of checking equivalence. This method uses the atomic distance matrices and the torsion angles between heavy atoms to determine if conformer candidates are duplicates.

DihedralThreshold**Type**

Float

Default value

30.0

Description

Maximum difference a dihedral can have for a conformer to be considered a duplicate.

DistanceThreshold**Type**

Float

Default value

0.1

Description

Maximum difference a distance between two atoms can have for a conformer to be considered a duplicate.

EnergyThreshold**Type**

Float

Default value

0.2

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

IrrelevantAtoms**Type**

Integer List

Description

To detect equivalence, only a subset of atoms is used. The atoms that are excluded from equivalence comparison should be specified here. By default only non-hydrogen atoms will be used for the comparison. Numbering starts at 0.

AcceptAll**Type**

Bool

Default value

No

Description

If set to True, add any candidate to the set without checks of connectivity changes, stereo- or cis/trans isomerization, or duplication.

AcceptIsomers**Type**

Bool

Default value

No

Description

If set to True, perform all checks of a new conformer candidate, except for the stereo- or cis/trans isomerization check.

CREST**Type**

Block

Description

Options for the CREST method of checking equivalence. This method uses the rotational constants of conformer candidates to determine if they are duplicates.

EnergyThreshold

Type

Float

Default value

0.05

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

RMSDThreshold**Type**

Float

Default value

0.125

Description

Threshold for the RMSD between two conformers that determines if they are duplicates or rotamers (according to the CREST rotamer definition.)

ScaledRotationalConstantSettings**Type**

Block

Description

By default, the equivalence of two geometries is determined mainly by comparing the rotational constants, and weighing the difference based on the average anisotropy of the two systems. This procedure has several settings that can be user defined.

RotationalConstantThreshold**Type**

Float

Default value

0.003

Description

Threshold for the difference in rotational constants that determines if two geometries are duplicates. The threshold is weighed by the anisotropy of the systems. Note: in the grimme code they use 0.01 as `bconst_threshold`, but this leads to a lot of misclassifications (i.e. different conformers are classified as equivalent rotamers) So, here we use a smaller default value.

CheckForDuplicates**Type**

Bool

Default value

Yes

Description

If set to True, check any new conformer candidate for duplication, and only accept unique conformers. If set to False, accept duplicates into the set.

LogDuplicates

Type

Bool

Default value

No

Description

If set to True, print information on equivalence to logfile and standard output.

Method**Type**

Multiple Choice

Default value

CREST

Options

[AMS, TFD, RMSD, CREST]

GUI name

Equivalence method

Description

Method used to determine (and filter out) equivalent conformers.

The CREST equivalence method relies on rotational constants comparisons. For this reason, conformers with the same rotational constants (such as mirror images) will be considered equivalent conformers.

The AMS equivalence method uses a distance matrix and dihedrals to compare conformers. This equivalence method can be computationally expensive for large molecules.

The TFD equivalence method uses the Torsion Fingerprint Difference as implemented in RDKit.

The RMSD equivalence method uses the RDKit GetBestRMS implementation.

RMSD**Type**

Block

Description

Options for the RMSD method of checking equivalence. This method uses the RDKit implementation of GetBestRMS, which enumerates over atomic permutations for pairs of geometries to detect duplicates based on the RMSD value.

EnergyThreshold**Type**

Float

Default value

0.05

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

RMSDThreshold

Type

Float

Default value

0.125

Description

Threshold on the RMSD difference to determine if two geometries represent the same conformer. This value is in Angstrom.

Reorder**Type**

Bool

Default value

Yes

Description

Reorder conformers based on energy, whenever a new conformer is added.

TFD**Type**

Block

Description

Options for the TFD method of checking equivalence. This method uses the Torsion Finger Print method to determine if two conformer candidates are duplicates.

EnergyThreshold**Type**

Float

Default value

0.05

Unit

kcal/mol

Description

The energy difference beyond which two conformers are always considered distinct.

TFDThreshold**Type**

Float

Default value

0.05

Description

Threshold on the torsion fingerprint difference to determine if two geometries represent the same conformer. This value is unit-less.

Expander**Type**

Block

Description

Options for the conformer expander. The Expander expands an existing conformer set, by adding new conformers to it. The new conformers are generated from the original conformers in the

set. Unlike the generators, the outcome of an expander is therefore very dependent on the input conformations.

The GC generator uses a genetic algorithm to create a combinatorial expansion by combining local substructures.

The MD expander start MD simulations from the conformers in the set, and extracts snapshots to create new conformers. Both these expanders are part of the CREST generator.

GC

Type

Block

Description

Options for the genetic algorithm for combinatorial expansion of conformer geometries. This generator only works if a non-zero set of conformers is already provided.

MaxGCenergy

Type

Float

Default value

6.0

Description

The maximum energy (relative to the lowest in the set) of the conformers we are going to use for expansion. The default is 6.0 kcal/mol, but if MaxEnergy was set, then that value is used.

Parallel

Type

Bool

Default value

Yes

Description

Determines if the combinatorial expansion of conformers should be performed in parallel or not (default: True).

RMSDThreshold

Type

Float

Default value

0.25

Description

Newly generated geometries are only considered unique if their RMSD from all other newly generated geometries is larger than this threshold.

MD

Type

Block

Description

Produces conformers by running a set of MD simulations at different elevated temperatures, extracting snapshots, and optimizing those.

MolecularDynamics

Type

Block

Description

Configures molecular dynamics (with the velocity-Verlet algorithm) with and without thermostats. This block allows to specify the details of the molecular dynamics calculation. Default values will be ignored.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps. Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

InitialVelocities**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod

Type

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature**Type**

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type**Type**

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values**Type**

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

NSteps**Type**

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

Preserve**Type**

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and 3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass**Type**

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum**Type**

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

System**Type**

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities**Type**

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

Restart**Type**

String

GUI name

Restart from

Description

The path to the ams.rkf file from which to restart the simulation.

Shake

Type

Block

Description

Parameters of the Shake/Rattle algorithm.

A11**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, "All triangles H C H" in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2**Type**

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV**Type**

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations**Type**

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates**Type**

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Thermostat**Type**

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply**Type**

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)

- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

TimeStep**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

Trajectory**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

EngineResultsFreq**Type**

Integer

GUI name

Engine results frequency

Description

Write MDStep*.rkf files with engine-specific results once every N steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of Checkpoint%Frequency.

ExitConditionFreq

Type

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every N steps. By default this is done every SamplingFreq steps.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N steps. By default this is done every SamplingFreq steps.

SamplingFreq**Type**

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds**Type**

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges

Type

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

Ngeoms**Type**

Integer

Default value

4

Description

At each temperature, MD simulations are started from Ngeoms different starting geometries. The starting geometries are extracted from the provided conformer set. If the conformer set is empty, then no more than a single geometry per temperature can be provided, limiting the total number of MD simulations.

Temperatures**Type**

Float List

Default value

[400.0, 500.0]

Unit

Kelvin

Description

The list of different temperatures at which MD simulations are run.

UseShake**Type**

Bool

Default value

No

Description

Constrain all -H bonds with shake. If turned on, the MD timestep is automatically increased.

MaxEnergy**Type**

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers. If the relative energy of a conformer with respect to the lowest conformer is larger than this value, the conformer will be discarded.

Method**Type**

Multiple Choice

Default value

GC

Options

[MD, GC]

GUI name

Generator method

Description

Method used to generate the conformers.

Preoptimization**Type**

Block

Description

If this block is enabled geometries will be preoptimized. After preoptimization the high energy conformers will be discarded, and then from the remaining set the unoptimized geometries will be optimized at higher level. This is to prevent the preoptimizer from collapsing different conformers into a single false minimum. As a result, preoptimization is only useful if MaxEnergy is chosen low.

Enable**Type**

Bool

Default value

No

Description

Perform preoptimization at a low level of accuracy.

Engine**Type**

Block

Description

The engine specifics to be used for preoptimization.

PreoptFactor**Type**

Integer

Default value

2

Description

This factor is multiplied with MaxEnergy, to determine which high energy conformers can be discarded after preoptimization.

RDKitETKDG**Type**

Block

Description

Settings for the call to RDKits ETKDG conformer generator tool.

BestRMSDThreshold**Type**

Float

Default value

-1.0

Description

After ETKDG conformer generation by RDKit, RDKit can be used to remove duplicates via the BestRMS algorithm. This filter does exactly the same as the RMSD equivalence detector in the Equivalence block.

Forcefield**Type**

Multiple Choice

Default value

None

Options

[None, UFF, MMFF]

Description

The name of the RDKit forcefield to use for geometry optimization at the end of ETKDG conformer generation (by default no geometry optimization is performed). Using the RDKit internal optimization may make the subsequent geometry optimizations with AMS faster.

Parallel**Type**

Bool

Default value

No

Description

Experimental: Parallelize the RDKit generation step by calling the RDKit conformer generation method in parallel from multiple processes.

RMSDThreshold**Type**

Float

Default value

-1.0

Description

Root Mean Square deviation threshold for removing similar/equivalent conformations during the RDKit ETKDG procedure. By default there is no pruning (value: -1).

UseExpTorsionAnglePrefs**Type**

String

Default value

default

Description

Impose experimental torsion angle preferences in RDKit ETKDG conformer generation. By default the RDKit version determines whether or not to switch this on.

RNGSeed**Type**

Integer

Description

Initial guesses for conformers can be randomly generated by RDKit, using the ETKDG algorithm. For reproducibility, a random number seed can be provided here.

Filter**Type**

Block

Description

Options for the conformer filtering.

MaxEnergy**Type**

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers. If the relative energy of a conformer with respect to the lowest conformer is larger than this value, the conformer will be discarded.

RemoveNonMinima**Type**

Bool

Default value

No

Description

For the final set of conformers, explicitly check that the geometry corresponds to a local minimum, and remove it if it does not. Note: this will run a PES point characterization, which can be computationally expensive!

Generator**Type**

Block

Description

Options for the conformer generator.

ANNEALING**Type**

Block

Description

Options for the annealing generator. This generator creates conformers by performing a simulated annealing simulation.

MolecularDynamics**Type**

Block

Description

Configures molecular dynamics (with the velocity-Verlet algorithm) with and without thermostats. This block allows to specify the details of the molecular dynamics calculation. Default values will be ignored.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps. Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

InitialVelocities**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod**Type**

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature

Type

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type

Type

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values

Type

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

NSteps

Type

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

Preserve**Type**

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and 3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass**Type**

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum**Type**

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

System**Type**

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities**Type**

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

Restart**Type**

String

GUI name

Restart from

Description

The path to the ams.rkf file from which to restart the simulation.

Shake**Type**

Block

Description

Parameters of the Shake/Rattle algorithm.

All**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, "All triangles H C H" in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2**Type**

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV**Type**

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations**Type**

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates**Type**

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Thermostat**Type**

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply**Type**

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)
- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

TimeStep**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

Trajectory**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

EngineResultsFreq**Type**

Integer

GUI name

Engine results frequency

Description

Write MDStep*.rkf files with engine-specific results once every N steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of Checkpoint%Frequency.

ExitConditionFreq**Type**

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every N steps. By default this is done every SamplingFreq steps.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N steps. By default this is done every SamplingFreq steps.

SamplingFreq**Type**

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds**Type**

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges**Type**

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

Temperatures**Type**

Float List

Default value

[298.0, 798.0]

Unit

Kelvin

Description

The minimum and maximum temperature of the annealing simulation. The simulation will start at the highest temperature, and cool down to the lowest.

UseShake**Type**

Bool

Default value

No

Description

Constrain all -H bonds with shake. If turned on, the MD timestep is automatically increased.

CREST**Type**

Block

Description

Options for the CREST generator. The CREST generator performs a set of metadynamics simulations (using the METADYNAMICS generator), then a set of MD simulations (using the MD expander), and finally it does a combinatorial expansion of the generated conformers using the GC expander. This sequence is repeated in an iterative fashion until the lowest energy conformer no longer changes.

ConvergenceQualityCrude**Type**

Multiple Choice

Default value

None

Options

[Normal, Good, VeryGood, Excellent, None]

Description

The tightness of the convergence of the crude geometry pre-optimizations. If set to none it will be selected two levels below ConvergenceQuality.

ConvergenceQualityTight**Type**

Multiple Choice

Default value

None

Options

[Normal, Good, VeryGood, Excellent, None]

Description

The tightness of the convergence of the final geometry optimizations. If set to none it will be selected the same as ConvergenceQuality.

GCStep**Type**

Bool

Default value

Yes

Description

Whether or not to include the combinatorial expansion of the conformers using the GC Generator. For big systems this step can be very time consuming. By default it is set to True.

NCycles**Type**

Integer

Default value

10

Description

The maximum number of CREST cycles (by default the number is 10). If the lowest conformer energy converges before then, Crest exits.

UseShake**Type**

Bool

Default value

Yes

Description

Whether or not SHAKE should be turned on in the MD and Metadynamics simulations. If this is turned on, the MD timestep is automatically increased (from 2 to 5 fs).

METADYNAMICS**Type**

Block

Description

Produces conformers by running a set of CREST-RMSD metadynamics simulations with different biases, extracting snapshots, and optimizing those.

ConvergenceQualityCrude**Type**

Multiple Choice

Default value

none

Options

[normal, good, verygood, excellent, none]

Description

The tightness of the convergence of the crude geometry pre-optimizations. If set to none it will be selected two levels below ConvergenceQuality.

MolecularDynamics**Type**

Block

Description

Configures molecular dynamics (with the velocity-Verlet algorithm) with and without thermostats. This block allows to specify the details of the molecular dynamics calculation. Default values will be ignored.

Checkpoint**Type**

Block

Description

Sets the frequency for storing the entire MD state necessary for restarting the calculation.

Frequency**Type**

Integer

Default value

1000

GUI name

Checkpoint frequency

Description

Write the MD state to the Molecule and MDResults sections on ams.rkf once every N steps. Setting this to zero disables checkpointing during the simulation, but one checkpoint at the very end of the simulation will always be written.

WriteProperties**Type**

Bool

Default value

No

Description

Honor top-level Properties input block when writing engine results files.

DEPRECATED: This input option will be removed in AMS2026 and will be considered always enabled. If you rely on it being disabled, please contact support@scm.com.

InitialVelocities**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

File**Type**

String

Description

AMS RKF file containing the initial velocities.

RandomVelocitiesMethod**Type**

Multiple Choice

Default value

Exact

Options

[Exact, Boltzmann, Gromacs]

GUI name

Velocity randomization method

Description

Specifies how are random velocities generated. Three methods are available.

Exact: Velocities are scaled to exactly match set random velocities temperature.

Boltzmann: Velocities are not scaled and sample Maxwell-Boltzmann distribution. However, the distribution is not corrected for constraints.

Gromacs: Velocities are scaled to match set random velocities temperature, but removal of net momentum is performed only after the scaling. Resulting kinetic energy is lower based on how much net momentum the system had.

Temperature

Type

Float

Unit

Kelvin

GUI name

Initial temperature

Description

Sets the temperature for the Maxwell-Boltzmann distribution when the type of the initial velocities is set to random, in which case specifying this key is mandatory.

AMSinput will use the first temperature of the first thermostat as default.

Type

Type

Multiple Choice

Default value

Random

Options

[Zero, Random, FromFile, Input]

GUI name

Initial velocities

Description

Specifies the initial velocities to assign to the atoms. Three methods to assign velocities are available.

Zero: All atom are at rest at the beginning of the calculation.

Random: Initial atom velocities follow a Maxwell-Boltzmann distribution for the temperature given by the [MolecularDynamics%InitialVelocities%Temperature] keyword.

FromFile: Load the velocities from a previous ams result file.

Input: Atom's velocities are set to the values specified in the [MolecularDynamics%InitialVelocities%Values] block, which can be accessed via the Expert AMS panel in AMSinput.

Values

Type

Non-standard block

Description

This block specifies the velocity of each atom, in Angstrom/fs, when [MolecularDynamics%InitialVelocities%Type] is set to Input. Each row must contain three floating point values (corresponding to the x,y,z component of the velocity vector) and a number of rows equal to the number of atoms must be present, given in the same order as the [System%Atoms] block.

NSteps

Type

Integer

Default value

1000

GUI name

Number of steps

Description

The number of steps to be taken in the MD simulation.

Preserve**Type**

Block

Description

Periodically remove numerical drift accumulated during the simulation to preserve different whole-system parameters.

AngularMomentum**Type**

Bool

Default value

Yes

GUI name

: Angular momentum

Description

Remove overall angular momentum of the system. This option is ignored for 2D and 3D-periodic systems, and disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

CenterOfMass**Type**

Bool

Default value

No

GUI name

: Center of mass

Description

Translate the system to keep its center of mass at the coordinate origin. This option is not very useful for 3D-periodic systems.

Momentum**Type**

Bool

Default value

Yes

GUI name

Preserve: Total momentum

Description

Remove overall (linear) momentum of the system. This is disabled by default for systems which are not translationally invariant (for example when frozen atoms are present).

Print**Type**

Block

Description

This block controls the printing of additional information to stdout.

System**Type**

Bool

Default value

No

Description

Print the chemical system before and after the simulation.

Velocities**Type**

Bool

Default value

No

Description

Print the atomic velocities before and after the simulation.

Restart**Type**

String

GUI name

Restart from

Description

The path to the ams.rkf file from which to restart the simulation.

Shake**Type**

Block

Description

Parameters of the Shake/Rattle algorithm.

All**Type**

String

Recurring

True

GUI name

Constrain all

Description

Constraint description in one the following formats:

All [bondOrder] bonds at1 at2 [to distance]

All triangles at1 at2 at3

The first option constrains all bonds between atoms at1 at2 to a certain length, while the second - bonds at1-at2 and at2-at3 and the angle between them.

The [bondOrder] can be a number or a string such as single, double, triple or aromatic. If it's omitted then all bonds between specified atoms will be constrained. Atom names are case-sensitive and they must be as they are in the Atoms block, or an asterisk '*' denoting any atom. The distance, if present, must be in Angstrom. If it is omitted then the bond length from the initial geometry is used.

Important: only the bonds present in the system at certain points of the simulation (at the start or right after adding/removing atoms) can be constrained, which means that the bonds may need to be specified in the System block.

Warning: the triangles constraint should be used with care because each constrained bond or angle means removing one degree of freedom from the dynamics. When there are too many constraints (for example, "All triangles H C H" in methane) some of them may be linearly dependent, which will lead to an error in the temperature computation.

Valid examples:

All single bonds C C to 1.4

All bonds O H to 0.98

All bonds O H

All bonds H *

All triangles H * H

ConvergeR2**Type**

Float

Default value

1e-08

Description

Convergence criterion on the max squared difference, in atomic units.

ConvergeRV**Type**

Float

Default value

1e-08

Description

Convergence criterion on the orthogonality of the constraint and the relative atomic velocity, in atomic units.

Iterations**Type**

Integer

Default value

100

Description

Number of iterations.

ShakeInitialCoordinates**Type**

Bool

Default value

Yes

Description

Apply constraints before computing the first energy and gradients.

Thermostat**Type**

Block

Recurring

True

Description

This block allows to specify the use of a thermostat during the simulation. Depending on the selected thermostat type, different additional options may be needed to characterize the specific thermostat' behavior.

BerendsenApply**Type**

Multiple Choice

Default value

Global

Options

[Local, Global]

GUI name

Apply Berendsen

Description

Select how to apply the scaling correction for the Berendsen thermostat:

- per-atom-velocity (Local)
- on the molecular system as a whole (Global).

ChainLength**Type**

Integer

Default value

10

GUI name

NHC chain length

Description

Number of individual thermostats forming the NHC thermostat

Duration**Type**

Integer List

GUI name

Duration(s)

Description

Specifies how many steps should a transition from a particular temperature to the next one in sequence take.

Region**Type**

String

Default value

*

Description

The identifier of the region to thermostat. The default '*' applies the thermostat to the entire system. The value can be a plain region name, or a region expression, e.g. '*-myregion' to thermostat all atoms that are not in myregion, or 'regionA+regionB' to thermostat the union of the 'regionA' and 'regionB'. Note that if multiple thermostats are used, their regions may not overlap.

Tau**Type**

Float

Unit

Femtoseconds

GUI name

Damping constant

Description

The time constant of the thermostat.

Temperature**Type**

Float List

Unit

Kelvin

GUI name

Temperature(s)

Description

The target temperature of the thermostat.

You can specify multiple temperatures (separated by spaces). In that case the Duration field specifies how many steps to use for the transition from one T to the next T (using a linear ramp). For NHC thermostat, the temperature may not be zero.

Type**Type**

Multiple Choice

Default value

None

Options

[None, Berendsen, NHC]

GUI name

Thermostat

Description

Selects the type of the thermostat.

TimeStep**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

The time difference per step.

Trajectory**Type**

Block

Description

Sets the frequency for printing to stdout and storing the molecular configuration on the .rkf file.

EngineResultsFreq**Type**

Integer

GUI name

Engine results frequency

Description

Write MDStep*.rkf files with engine-specific results once every N steps. Setting this to zero disables writing engine results files except for one file at the end of the simulation. If unset or negative, defaults to the value of Checkpoint%Frequency.

ExitConditionFreq**Type**

Integer

GUI name

Exit condition frequency

Description

Check the exit conditions every N steps. By default this is done every SamplingFreq steps.

PrintFreq**Type**

Integer

GUI name

Printing frequency

Description

Print current thermodynamic properties to the output every N steps. By default this is done every SamplingFreq steps.

SamplingFreq**Type**

Integer

Default value

100

GUI name

Sample frequency

Description

Write the molecular geometry (and possibly other properties) to the ams.rkf file once every N steps.

TProfileGridPoints**Type**

Integer

Default value

0

Description

Number of points in the temperature profile. If TProfileGridPoints > 0, a temperature profile along each of the three lattice axes will be written to the .rkf file. The temperature at a given profile point is calculated as the total temperature of all atoms inside the corresponding slice of the simulation box, time-averaged over all MD steps since the previous snapshot. By default, no profile is generated.

WriteBonds**Type**

Bool

Default value

Yes

Description

Write detected bonds to the .rkf file.

WriteCharges**Type**

Bool

Default value

Yes

Description

Write current atomic point charges (if available) to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze charges.

WriteCoordinates**Type**

Bool

Default value

Yes

Description

Write atomic coordinates to the .rkf file.

WriteEngineGradients**Type**

Bool

Default value

No

Description

Write atomic gradients (negative of the atomic forces, as calculated by the engine) to the History section of ams.rkf.

WriteMolecules**Type**

Bool

Default value

Yes

Description

Write the results of molecule analysis to the .rkf file.

WriteVelocities**Type**

Bool

Default value

Yes

Description

Write velocities to the .rkf file. Disable this to reduce trajectory size if you do not need to analyze the velocities.

NCycles**Type**

Integer

Default value

5

Description

The maximum number of cycles of metadynamics simulations. The generator stops when either this number is reached, or the conformer set is stable.

NWidthsHeights**Type**

Integer List

Default value

[4, 3]

Description

The number of different Gaussian widths and heights respectively used in the metadynamics

simulations. By default 4 different widths are used and 3 different heights, resulting in 12 different metadynamics simulations.

SimulationSuccessFraction**Type**

Float

Default value

0.4

Description

The fraction of planned metadynamics steps that has to have succeeded in order for the metadynamics iteration to be considered a success.

UseShake**Type**

Bool

Default value

No

Description

Constrain all -H bonds with shake. If turned on, the MD timestep is automatically increased.

MaxEnergy**Type**

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers. If the relative energy of a conformer with respect to the lowest conformer is larger than this value, the conformer will be discarded.

Method**Type**

Multiple Choice

Default value

RDKit

Options

[RDKit, CREST, ANNEALING]

GUI name

Generator method

Description

Method used to generate the conformers.

The RDKit generator is based on random distance matrix method. This is the recommended (and default) method.

The CREST Generator uses a multi-step workflow with meta-dynamics simulations to explore the conformers space of a molecule. This can be a powerful conformer search method, but it is generally computationally expensive compared to the RDKit generator.

The ANNEALING generator performs a simulated annealing simulation to explore conformer space.

Preoptimization**Type**

Block

Description

If this block is enabled geometries will be preoptimized. After preoptimization the high energy conformers will be discarded, and then from the remaining set the unoptimized geometries will be optimized at higher level. This is to prevent the preoptimizer from collapsing different conformers into a single false minimum. As a result, preoptimization is only useful if MaxEnergy is chosen low.

Enable**Type**

Bool

Default value

No

Description

Perform preoptimization at a low level of accuracy.

Engine**Type**

Block

Description

The engine specifics to be used for preoptimization.

PreoptFactor**Type**

Integer

Default value

2

Description

This factor is multiplied with MaxEnergy, to determine which high energy conformers can be discarded after preoptimization.

RDKit**Type**

Block

Description

Options for the RDKit generator. This generator produces initial guesses for conformers using the RDKit ETKDG method, followed by AMS geometry optimizations.

InitialNConformers**Type**

Integer

GUI name

Initial no. of conformers

Description

Number of geometries initially created by RDKit, before AMS geometry optimization and filtering. If not set, the number will be automatically set, based on the number of rotational bonds.

MaxConfs**Type**

Integer

Default value

5000

Description

If InitialNConformers is not set, then the number of conformers will be automatically set, with a maximum of MaxConfs.

MinConfs**Type**

Integer

Default value

10

Description

If InitialNConformers is not set, then the number of conformers will be automatically set, with a minimum of MinConfs.

NconfsEstimationFactor**Type**

Integer

Default value

100

Description

If InitialNConformers is not set, then the number of conformers will be automatically set based on the number of rotational bonds. The resulting number is then multiplied by this factor (default: 100), to ensure that enough conformers will be created.

RDKitETKDG**Type**

Block

Description

Settings for the call to RDKit's ETKDG conformer generator tool.

BestRMSDThreshold**Type**

Float

Default value

-1.0

Description

After ETKDG conformer generation by RDKit, RDKit can be used to remove duplicates via the BestRMS algorithm. This filter does exactly the same as the RMSD equivalence detector in the Equivalence block.

ConstrainedAtoms**Type**

Integer List

Description

The indices of the atoms to constrain during ETKDG conformer generation.

Forcefield**Type**

Multiple Choice

Default value

None

Options

[None, UFF, MMFF]

Description

The name of the RDKit forcefield to use for geometry optimization at the end of ETKDG conformer generation (by default no geometry optimization is performed). Using the RDKit internal optimization may make the subsequent geometry optimizations with AMS faster.

Parallel**Type**

Bool

Default value

No

Description

Experimental: Parallelize the RDKit generation step by calling the RDKit conformer generation method in parallel from multiple processes.

RMSDThreshold**Type**

Float

Default value

-1.0

Description

Root Mean Square deviation threshold for removing similar/equivalent conformations during the RDKit ETKDG procedure. By default there is no pruning (value: -1).

UseExpTorsionAnglePrefs**Type**

String

Default value

default

Description

Impose experimental torsion angle preferences in RDKit ETKDG conformer generation. By default the RDKit version determines whether or not to switch this on.

RNGSeed**Type**

Integer

Description

Initial guesses for conformers can be randomly generated by RDKit, using the ETKDG algorithm. For reproducibility, a random number seed can be provided here.

TORSION**Type**

Block

Description

Options for the TorsionGenerator, which generates geometries by enumerative rotation around rotatable bonds. This is the slowest of all generator, and quickly becomes infeasible for large systems. Currently does not work for systems with interconnected rings.

Dtheta**Type**

Float

Default value

60.0

Description

The angle over which the bonds are rotated, in order to create a new conformer.

GeometryOptimization**Type**

Block

Description

Some options / details regarding the optimization procedure.

ConvergenceQuality**Type**

Multiple Choice

Default value

VeryGood

Options

[Normal, Good, VeryGood, Excellent]

GUI name

Convergence

Description

The tightness of the convergence of the geometry optimizations. Lower quality levels may lead to badly converged geometries being classified as distinct conformers.

GeometryOptimization**Type**

Block

Description

Configures details of the geometry optimization and transition state searches.

CalcPropertiesOnlyIfConverged**Type**

Bool

Default value

Yes

Description

Compute the properties requested in the 'Properties' block, e.g. Frequencies or Phonons,

only if the optimization (or transition state search) converged. If False, the properties will be computed even if the optimization did not converge.

Convergence

Type

Block

Description

Convergence is monitored for up to 4 quantities: the energy change, the Cartesian gradients, the Cartesian step size, and for lattice optimizations the stress energy per atom. Convergence criteria can be specified separately for each of these items.

Energy

Type

Float

Default value

1e-05

Unit

Hartree

Value Range

value > 0

GUI name

Energy convergence

Description

The criterion for changes in the energy. The energy is considered converged when the change in energy is smaller than this threshold times the number of atoms.

Gradients

Type

Float

Default value

0.001

Unit

Hartree/Angstrom

Value Range

value > 0

GUI name

Gradient convergence

Description

Threshold for nuclear gradients.

Quality

Type

Multiple Choice

Default value

Custom

Options

[VeryBasic, Basic, Normal, Good, VeryGood, Custom]

GUI name

Convergence

Description

A quick way to change convergence thresholds: 'Good' will reduce all thresholds by an order of magnitude from their default value. 'VeryGood' will tighten them by two orders of magnitude. 'Basic' and 'VeryBasic' will increase the thresholds by one or two orders of magnitude respectively.

Step**Type**

Float

Default value

0.01

Unit

Angstrom

Value Range

value > 0

GUI name

Step convergence

Description

The maximum Cartesian step allowed for a converged geometry.

StressEnergyPerAtom**Type**

Float

Default value

0.0005

Unit

Hartree

Value Range

value > 0

Description

Threshold used when optimizing the lattice vectors. The stress is considered 'converged' when the maximum value of $\text{stress_tensor} * \text{cell_volume} / \text{number_of_atoms}$ is smaller than this threshold (for 2D and 1D systems, the cell_volume is replaced by the cell_area and cell_length respectively).

CoordinateType**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Delocalized, Cartesian]

GUI name

Optimization space

Description

Select the type of coordinates in which to perform the optimization. 'Auto' automatically selects the most appropriate CoordinateType for a given Method.

If 'Auto' is selected, Delocalized coordinates will be used for the Quasi-Newton method, while Cartesian coordinates will be used for all other methods.

Dimer**Type**

Block

Description

Options for the Dimer method for transition state search.

AngleThreshold**Type**

Float

Default value

1.0

Unit

Degree

Description

The rotation is considered converged when the the rotation angle falls below the specified threshold.

DimerDelta**Type**

Float

Default value

0.01

Unit

Angstrom

Description

Euclidian distance between the midpoint and the endpoint.

ExtrapolateForces**Type**

Bool

Default value

Yes

Description

Set to false to call engine to calculate forces at the extrapolated rotation angle instead of extrapolating them.

LBFGSMaxVectors**Type**

Integer

Default value

10

Description

Max number of vectors for the L-BFGS algorithm to save.

MaxRotationIterations**Type**

Integer

Default value

10

Description

Maximum number of rotation iterations for a single translation step.

Region**Type**

String

Default value

*

Description

Include only atoms of the specified region(s) in the rotations, which allows searching for a transition state involving selected atoms only.

RotationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during rotation iterations.

TranslationTrustRadius**Type**

Float

Default value

0.1

Unit

Angstrom

Description

L-BFGS trust radius during translation iterations.

EngineAutomations**Type**

Block

Description

The optimizer can change some settings of the engine, based for instance on the error.

The idea is to allow the engine to be a bit quicker at the start, and more accurate towards the end.

Automations are always engine specific.

Enabled

Type

Bool

Default value

Yes

Description

Whether or not automations are enabled at all.

Gradient

Type

Block

Recurring

True

Description

A gradient-based automation.

FinalValue

Type

Float

Description

This value will be used whenever the gradient is less than GradientLow

HighGradient

Type

Float

Default value

1.0

Unit

Hartree/Angstrom

Description

Defines a large gradient. When the actual gradient is between GradientHigh and GradientLow a linear interpolation scheme is used for kT (on a log scale).

InitialValue

Type

Float

Description

This value will be used at the first geometry, and whenever the gradient is higher than GradientHigh

LowGradient

Type

Float

Default value

1.0

Unit

Hartree/Angstrom

Description

Defines a small gradient, see GradientHigh

UseLogInterpolation**Type**

Bool

Default value

Yes

Description

Whether to use interpolation on a log (y) scale or not

Variable**Type**

String

Default value**Description**

variable to be tweaked for the engine.

Iteration**Type**

Block

Recurring

True

Description

Geometry step based automation.

FinalValue**Type**

Float

Description**FirstIteration****Type**

Integer

Default value

1

Description

When the actual gradient is between the first and last iteration, a linear interpolation is used.

InitialValue**Type**

Float

Description

This value will be used when the iteration number is smaller or equal to FirstIteration

LastIteration

Type

Integer

Default value

10

Description

Where the automation should reach the FinalValue

UseLogInterpolation**Type**

Bool

Default value

Yes

Description

Whether to use interpolation on a log (y) scale or not

Variable**Type**

String

Default value**Description**

variable to be tweaked for the engine.

FIRE**Type**

Block

Description

This block configures the details of the FIRE optimizer. The keywords name correspond the the symbols used in the article describing the method, see PRL 97, 170201 (2006).

AllowOverallRotation**Type**

Bool

Default value

Yes

Description

Whether or not the system is allowed to freely rotate during the optimization. This is relevant when optimizing structures in the presence of external fields.

AllowOverallTranslation**Type**

Bool

Default value

No

Description

Whether or not the system is allowed to translate during the optimization. This is relevant when optimizing structures in the presence of external fields.

MapAtomsToUnitCell

Type

Bool

Default value

No

Description

Map the atoms to the central cell at each geometry step.

NMin**Type**

Integer

Default value

5

Description

Number of steps after stopping before increasing the time step again.

alphaStart**Type**

Float

Default value

0.1

Description

Steering coefficient.

dtMax**Type**

Float

Default value

1.0

Unit

Femtoseconds

Description

Maximum time step used for the integration. For ReaxFF and APPLE&P, this value is reduced by 50%.

dtStart**Type**

Float

Default value

0.25

Unit

Femtoseconds

Description

Initial time step for the integration.

fAlpha**Type**

Float

Default value

0.99

Description

Reduction factor for the steering coefficient.

fDec**Type**

Float

Default value

0.5

Description

Reduction factor for reducing the time step in case of uphill movement.

fInc**Type**

Float

Default value

1.1

Description

Growth factor for the integration time step.

strainMass**Type**

Float

Default value

0.5

Description

Fictitious relative mass of the lattice degrees of freedom. This controls the stiffness of the lattice degrees of freedom relative to the atomic degrees of freedom, with smaller values resulting in a more aggressive optimization of the lattice.

HessianFree**Type**

Block

Description

Configures details of the Hessian-free (conjugate gradients or L-BFGS) geometry optimizer.

Step**Type**

Block

Description**MaxCartesianStep****Type**

Float

Default value

0.1

Unit

Angstrom

Description

Limit on a single Cartesian component of the step.

MinRadius**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Minimum value for the trust radius.

TrialStep**Type**

Float

Default value

0.0005

Unit

Angstrom

Description

Length of the finite-difference step when determining curvature. Should be smaller than the step convergence criterion.

TrustRadius**Type**

Float

Default value

0.2

Unit

Angstrom

Description

Initial value of the trust radius.

InitialHessian**Type**

Block

Description

Options for initial model Hessian when optimizing systems with the Quasi-Newton method.

File**Type**

String

GUI name

Initial Hessian from

Description

KF file containing the initial Hessian (or the results dir. containing it). This can be used to load a Hessian calculated in a previously with the [Properties%Hessian] keyword.

Type**Type**

Multiple Choice

Default value

Auto

Options

[Auto, UnitMatrix, Swart, FromFile, Calculate, CalculateWithFastEngine]

GUI name

Initial Hessian

Description

Select the type of initial Hessian. Auto: let the program pick an initial model Hessian. UnitMatrix: simplest initial model Hessian, just a unit matrix in the optimization coordinates. Swart: model Hessian from M. Swart. FromFile: load the Hessian from the results of a previous calculation (see InitialHessian%File). Calculate: compute the initial Hessian (this may be computationally expensive and it is mostly recommended for TransitionStateSearch calculations). CalculateWithFastEngine: compute the initial Hessian with a faster engine.

KeepIntermediateResults**Type**

Bool

Default value

No

Description

Whether the full engine result files of all intermediate steps are stored on disk. By default only the last step is kept, and only if the geometry optimization converged. This can easily lead to huge amounts of data being stored on disk, but it can sometimes be convenient to closely monitor a tricky optimization, e.g. excited state optimizations going through conical intersections, etc. ...

MaxIterations**Type**

Integer

Value Range

value ≥ 0

GUI name

Maximum number of iterations

Description

The maximum number of geometry iterations allowed to converge to the desired structure.

MaxRestarts**Type**

Integer

Default value

0

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small displacement along the imaginary vibrational mode. In case the restarted optimization again does not find a minimum, this can happen multiple times in succession. This keyword sets the maximum number of restarts. The default value is 0, so the automatic restarting is disabled by default.

Method**Type**

Multiple Choice

Default value

Auto

Options

[Auto, Quasi-Newton, FIRE, L-BFGS, ConjugateGradients, Dimer]

GUI name

Optimization method

Description

Select the optimization algorithm employed for the geometry relaxation. Currently supported are:

the Hessian-based Quasi-Newton-type BFGS algorithm,

the fast inertial relaxation method (FIRE),

the limited-memory BFGS method,

and the conjugate gradients method. The default is to choose an appropriate method automatically based on the engine's speed, the system size and the supported optimization options.

OptimizeLattice**Type**

Bool

Default value

No

Description

Whether to also optimize the lattice for periodic structures. This is currently supported with the Quasi-Newton, FIRE, and L-BFGS optimizers.

PretendConverged**Type**

Bool

Default value

No

Description

Normally a non-converged geometry optimization is considered an error. If this keyword is set to True, the optimizer will only produce a warning and still claim that the optimization is converged. (This is mostly useful for scripting applications, where one might want to consider non-converged optimizations still successful jobs.)

Quasi-Newton

Type

Block

Description

Configures details of the Quasi-Newton geometry optimizer.

MaxGDIISVectors**Type**

Integer

Default value

0

Description

Sets the maximum number of GDIIS vectors. Setting this to a number >0 enables the GDIIS method.

Step**Type**

Block

Description**TrustRadius****Type**

Float

Description

Initial value of the trust radius.

VaryTrustRadius**Type**

Bool

Description

Whether to allow the trust radius to change during optimization. By default True during energy minimization and False during transition state search.

UpdateTSVectorEveryStep**Type**

Bool

Default value

Yes

GUI name

Update TSRC vector every step

Description

Whether to update the TS reaction coordinate at each step with the current eigenvector.

RestartDisplacement**Type**

Float

Default value

0.05

Unit

Angstrom

Description

If a geometry optimization of a system with no symmetry operators (or with explicitly disabled symmetry: `UseSymmetry False`) and enabled PES point characterization converges to a transition state (or higher order saddle point), it can be restarted automatically after a small displacement along the imaginary vibrational mode. This keywords sets the size of the displacement for the furthest moving atom.

Keep**Type**

Multiple Choice

Default value

None

Options

[None, all]

Description

Keep all the output files of the geometry optimizations. If set to 'all', must be used in combination with `Output%KeepWorkDir`.

MaxConvergenceTime**Type**

Multiple Choice

Default value

Default

Options

[Default, High]

Description

The number of iterations for the geometry optimization, based on the number of atoms in the system. The default value is the general AMS value for geometry optimization. Often, for conformer generation, it needs to be set higher.

MaxOptimizations**Type**

Integer

Default value

1000

Description

Set a maximum to the number of geometries accepted for optimization at once, per `AMSWorker` (so should be multiplied by the number of cores used). If not set, the disc size requirements can become too large.

OptimizationMethod**Type**

Multiple Choice

Default value

Quasi-Newton

Options

[Auto, Quasi-Newton, FIRE, L-BFGS, ConjugateGradients, Dimer]

Description

Select the optimization algorithm employed for the geometry relaxation. Currently supported are:

the Hessian-based Quasi-Newton-type BFGS algorithm,

the fast inertial relaxation method (FIRE),

the limited-memory BFGS method,

and the conjugate gradients method. The default is Quasi-Newton, which gives the most reliable results for conformers. The Auto method leaves it to the AMS GeometryOptimization task to select a method.

UseAMSWorker**Type**

Bool

Default value

Yes

Description

Whether the set of optimizations should be run via the AMSWorkerPool or via regular AM-SJobs.

WriteGeometries**Type**

Block

Description

Determines if and where optimized geometries will be written to file during optimization. When the AMSWorker is used, then the write interval depends on MaxOptimizations. If enabled, must be used in combination with Output%KeepWorkDir.

Dirname**Type**

String

Default value

conf_tmpdir

Description

The name of the folder that should contain the optimized geometries.

Enabled**Type**

Bool

Default value

No

Description

Enables or disables the periodic writing of optimized geometries to file

InputConformersSet**Type**

String

Recurring

True

Description

The path to a file containing a set of conformers.

The file should be either an 'conformers.rkf' file (i.e. the results file of conformers), a concatenated .xyz file, a .sdf file, or even .dcd file. In the case of a .dcd file, an additional System block is required. If the specified file does not contain energies in the AMS conformers format (as produced by the conformers tool), all conformer energies will be set to zero.

You can specify multiple input conformers sets by including the InputConformersSet keyword multiple times.

InputMaxConfs**Type**

Integer

Description

The maximum number of conformers to carry forward when loading conformer sets. If this input is not specified, this limit will not be imposed.

InputMaxEnergy**Type**

Float

Unit

kcal/mol

Description

Threshold for filtering out high-energy conformers when loading conformers sets using the Input-ConformerSet keyword. Conformers with an larger relative energy will not be loaded.

LoadSystem**Type**

Block

Recurring

True

Description

Block that controls reading the chemical system from a KF file instead of the [System] block.

File**Type**

String

Description

The path of the KF file from which to load the system. It may also be the results directory containing it.

Section**Type**

String

Default value

Molecule

Description

The section on the KF file from which to load the system.

Output**Type**

Block

Description

Options regarding the output and result files.

KeepWorkDir**Type**

Bool

Default value

No

Description

Do not remove the working directories after the conformer generation is finished.

rkf**Type**

Bool

Default value

Yes

Description

Save the final conformers in .rkf format. The file 'conformers.rkf' will be located in the results directory. You can visualize this file using the AMSMovie GUI module.

sdf**Type**

Bool

Default value

Yes

Description

Save the final conformers in .sdf format. The file 'conformers.sdf' will be located in the results directory.

xyz**Type**

Bool

Default value

Yes

Description

Save the final conformers in .xyz format. The file 'conformers.xyz' will be located in the results directory.

Restraints**Type**

Block

Description

The Restraints block allows to add soft constraints to the system. A restraint is a potential energy

function (a spring) attached to a certain coordinate, for example, an interatomic distance, with its minimum at the specified optimal value. A restraint is defined using one or two parameters: the ForceConstant and, for some types, the F(Inf) value. The ForceConstant parameter corresponds to second derivative of the restraint potential energy $d^2V(x)/dx^2$ for any x (harmonic restraints) or only at $x=0$ (other restraints). Here, x is a deviation from the restraint's optimal value.

Angle**Type**

String

Recurring

True

Description

Specify three atom indices $i\ j\ k$ followed by an angle in degrees and, optionally, by the ForceConstant (default is 0.3 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the i - j - k angle at the given value. For periodic systems this restraint follows the minimum image convention.

DifDist**Type**

String

Recurring

True

Description

Specify four atom indices $i\ j\ k\ l$ followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the difference $R(ij)-R(kl)$ at the given value. For periodic systems this restraint follows the minimum image convention.

Dihedral**Type**

String

Recurring

True

Description

Specify four atom indices $i\ j\ k\ l$ followed by an angle in degrees and, optionally, by the ForceConstant (default is 0.1 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the i - j - k - l dihedral angle at the given value. For periodic systems this restraint follows the minimum image convention.

Distance**Type**

String

Recurring

True

Description

Specify two atom indices followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the distance between the two specified atoms at the given value. For periodic systems this restraint follows the minimum image convention.

FInfinity**Type**

Float

Default value

1.0

GUI name

Default F(inf)

Description

Specify the default asymptotic value for the restraint force for the Hyperbolic and Erf profiles, in Hartree/Bohr or Hartree/radian.

A per-restraint value can be specified after the profile type on the corresponding restraint line.

Profile**Type**

Multiple Choice

Default value

Harmonic

Options

[Harmonic, Hyperbolic, Erf, GaussianWell]

GUI name

Default restraint profile

Description

Select the default type of restraint profile.

The harmonic profile is most suitable for geometry optimizations but may result in very large forces that can be problematic in molecular dynamic.

For MD simulations the Hyperbolic or Erf may be more suitable because the restraint force is bounded by a user-defined value.

A per-restraint profile type can be specified after the ForceConstant value on the corresponding restraint line.

SumDist**Type**

String

Recurring

True

Description

Specify four atom indices i j k l followed by the distance in Angstrom and, optionally, by the ForceConstant (default is 1.0 in a.u.), profile type and F(Inf) (in a.u.). This restraint will try to keep the sum $R(ij)+R(kl)$ at the given value. For periodic systems this restraint follows the minimum image convention.

Units**Type**

Multiple Choice

Default value

Default

Options

[Default, MD]

GUI name

Units

Description

Change units for energy, force and force constant values from the default (atomic units) to those often used in the MD community (based on kcal/mol and Angstrom). Units for the optimal distances are not affected and are always Angstrom.

RNGSeed**Type**

Integer

Description

Initial seed for the (pseudo)random number generator. If this is unset, the generator will be seeded randomly from external sources of entropy and the generated conformers will be non-deterministic.

System**Type**

Block

Recurring

True

Description

Specification of the chemical system. For some applications more than one system may be present in the input. In this case, all systems except one must have a non-empty string ID specified after the System keyword. The system without an ID is considered the main one.

AllowCloseAtoms**Type**

Bool

Default value

No

Description

If AllowCloseAtoms is set to False, the AMS driver will stop with an error if it detects almost-coinciding atomic coordinates. If set to True, the AMS driver will try to carry on with the calculation.

Atoms**Type**

Non-standard block

Description

The atom types and coordinates. Unit can be specified in the header. Default unit is Angstrom.

BondOrders**Type**

Non-standard block

Description

Defined bond orders. Each line should contain two atom indices, followed by the bond order (1, 1.5, 2, 3 for single, aromatic, double and triple bonds) and (optionally) the cell shifts for periodic systems. May be used by MM engines and for defining constraints. If the system is

periodic and none of the bonds have the cell shift defined then AMS will attempt to determine them following the minimum image convention.

Charge

Type

Float

Default value

0.0

GUI name

Total charge

Description

The system's total charge in atomic units.

ElectrostaticEmbedding

Type

Block

Description

Container for electrostatic embedding options, which can be combined.

ElectricField

Type

Float List

Unit

V/Angstrom

Description

External homogeneous electric field with three Cartesian components: ex, ey, ez, the default unit being V/Å.

In atomic units: Hartree/(e bohr) = 51.422 V/Angstrom; the relation to SI units is: 1 Hartree/(e bohr) = 5.14 ... e11 V/m.

Supported by the engines adf, band, dftb and mopac.

For periodic systems the field may only have nonzero components orthogonal to the direction(s) of periodicity (i.e. for 1D periodic system the x-component of the electric field should be zero, while for 2D periodic systems both the x and y components should be zero. This options cannot be used for 3D periodic systems.

MultipolePotential

Type

Block

Description

External point charges (and dipoles).

ChargeModel

Type

Multiple Choice

Default value

Point

Options

[Point, Gaussian]

Description

A multipole may be represented by a point (with a singular potential at its location) or by a spherical Gaussian distribution.

ChargeWidth**Type**

Float

Default value

-1.0

Description

The width parameter in a.u. in case a Gaussian charge model is chosen. A negative value means that the width will be chosen automatically.

Coordinates**Type**

Non-standard block

Description

Positions and values of the multipoles, one per line. Each line has the following format:

x y z q, or

x y z q μ_x μ_y μ_z .

Here x, y, z are the coordinates in Å, q is the charge (in atomic units of charge) and μ_x , μ_y , μ_z are the (optional) dipole moment components (in atomic units, i.e. e*Bohr).

Periodic systems are not supported.

FractionalCoords**Type**

Bool

Default value

No

Description

Whether the atomic coordinates in the Atoms block are given in fractional coordinates of the lattice vectors. Requires the presence of the Lattice block.

GeometryFile**Type**

String

Description

Read the geometry from a file (instead of from Atoms and Lattice blocks). Supported formats: .xyz

GuessBonds**Type**

Bool

Default value

No

Description

Whether or not UFF bonds should be guessed.

Lattice**Type**

Non-standard block

Description

Up to three lattice vectors. Unit can be specified in the header. Default unit is Angstrom.

LatticeStrain**Type**

Float List

Description

Deform the input system by the specified strain. The strain elements are in Voigt notation, so one should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

LoadForceFieldAtomTypes**Type**

Block

Description

This is a mechanism to set the ForceField.Type attribute in the input. This information is currently only used by the ForceField engine.

File**Type**

String

Description

Name of the (kf) file. It needs to be the result of a forcefield calculation.

LoadForceFieldCharges**Type**

Block

Recurring

True

Description

This is a mechanism to set the ForceField.Charge attribute in the input. This information is currently only used by the ForceField engine.

CheckGeometryRMSD**Type**

Bool

Default value

No

Description

Whether the geometry RMSD test should be performed, see MaxGeometryRMSD. Otherwise only basic tests are performed, such as number and atom types. Not doing the RMSD test allows you to load molecular charges in a periodic system.

File**Type**

String

Description

Name of the (kf) file

MaxGeometryRMSD**Type**

Float

Default value

0.1

Unit

Angstrom

Description

The geometry of the charge producing calculation is compared to the one of the region, and need to be the same within this tolerance.

Region**Type**

String

Default value

*

Description

Region for which the charges should be loaded

Section**Type**

String

Default value

AMSResults

Description

Section name of the kf file

Variable**Type**

String

Default value

Charges

Description

Variable name of the kf file

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

For periodic systems the atoms will be moved to the central cell.

ModifyAlternativeElements

Type

Bool

Default value

No

Description

When using alternative elements (using the `nuclear_charge` attribute) set the element to the nearest integer Z. If you specify an H atom with a `nuclear_charge` of 2.9 it is replaced by a Li atom with the same nuclear charge.

PerturbCoordinates**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Perturb the atomic coordinates by adding random numbers between `[-PerturbCoordinates,PerturbCoordinates]` to each Cartesian component. This can be useful if you want to break the symmetry of your system (e.g. for a geometry optimization).

PerturbLattice**Type**

Float

Default value

0.0

Description

Perturb the lattice vectors by applying random strain with matrix elements between `[-PerturbLattice,PerturbLattice]`. This can be useful if you want to deviate from an ideal symmetric geometry, for example if you look for a phase change due to high pressure.

RandomizeAtomOrder**Type**

Bool

Default value

No

Description

Whether or not the order of the atoms should be randomly changed. Intended for some technical testing purposes only. Does not work with bond information.

Region**Type**

Block

Recurring

True

Description

Properties for each region specified in the Atoms block.

Properties

Type

Non-standard block

Description

Properties for each region specified in the Atoms block.

ShiftCoordinates**Type**

Float List

Unit

Bohr

Description

Translate the atoms by the specified shift (three numbers).

SuperCell**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems). The integer numbers represent the diagonal elements of the supercell transformation; you should specify as many numbers as lattice vectors (i.e. 1 number for 1D, 2 numbers for 2D and 3 numbers for 3D periodic systems).

SuperCellTrafo**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems) $\vec{a}'_i = \sum_j T_{ij} \vec{a}_j$. The integer numbers represent the supercell transformation T_{ij} : 1 number for 1D PBC, 4 numbers for 2D PBC corresponding to a 2x2 matrix (order: (1,1),(1,2),(2,1),(2,2)) and 9 numbers for 3D PBC corresponding to a 3x3 matrix (order: (1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)).

Symmetrize**Type**

Bool

Default value

No

Description

Whether to symmetrize the input structure. This might also rototranslate the structure into a standard orientation. This will symmetrize the atomic coordinates to machine precision. Useful if the system is almost symmetric or to rototranslate a symmetric molecule into a standard orientation.

Symmetry**Type**

Multiple Choice

Default value

AUTO

Options

[AUTO, NOSYM, C(LIN), D(LIN), C(I), C(S), C(2), C(3), C(4), C(5), C(6), C(7), C(8),

C(2V), C(3V), C(4V), C(5V), C(6V), C(7V), C(8V), C(2H), C(3H), C(4H), C(5H), C(6H), C(7H), C(8H), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(2D), D(3D), D(4D), D(5D), D(6D), D(7D), D(8D), D(2H), D(3H), D(4H), D(5H), D(6H), D(7H), D(8H), I, I(H), O, O(H), T, T(D), T(H), S(4), S(6), S(8)]

Description

Use (sub)symmetry with this Schoenflies symbol. Can only be used for molecules. Orientation should be correct for the (sub)symmetry. If used icw Symmetrize, the symmetrization will not reorient the molecule.

Task**Type**

Multiple Choice

Default value

Generate

Options

[Generate, Optimize, Filter, Score, Expand]

Description

The task to be performed by the Conformers tool.

‘Generate’: given a molecule, generate a set of conformers. Note: this task will automatically optimize, filter and score the conformers.

‘Optimize’: given a previously generated set of conformers, optimize, filter and score the structures using the specified engine.

‘Filter’: given one or more previously generated set of conformers, merge them into a single conformer set and filter out duplicate conformers. Note: this will not optimize or re-score the conformers.

‘Score’ given one or more previously generated set of conformers, re-score them by computing the energy using the specified engine. Note: this will only do a single point calculation, and will not optimize the structures.

In case of ‘Optimize’, ‘Filter’ and ‘Score’ you can specify the input conformer set(s) using the ‘InputConformersSet’ keyword.

15.2.5 fcf

See *FCF manual page* (page 289)

15.2.6 oled-deposition**Box****Type**

Block

Description

Specifications of the box into which the material is deposited.

Size**Type**

Float List

Default value

[60.0, 60.0, 120.0]

Unit

Angstrom

GUI name

Box size

Description

Specify the desired size of the box. The final deposited box may have a different size. The x- and y-axis are perpendicular to the direction of deposition, so these may be regarded as the width of the growing layer. The z-axis is the direction along which the deposition happens, so this determines the thickness of the deposited layer. Note that the x- and y-axis will be ignored if a custom substrate is used: the area of the box is then determined by the lattice of the substrate. The z-axis can still be freely chosen, but should be large enough that there is enough space for the substrate itself and to deposit more molecules on top of it.

Substrate**Type**

Multiple Choice

Default value

Graphene

Options

[Graphene, Custom]

Description

The substrate on which to grow the layer.

SubstrateSystem**Type**

String

GUI name

Custom substrate

Description

String ID of a named [System] to be used as a substrate. (This is only used when the Substrate key is set to Custom.)

Deposition**Type**

Block

Description

Specifies the details of how molecules are deposited.

ConstrainHXBonds**Type**

Bool

Default value

Yes

GUI name

Constrain H-* bonds

Description

Constrain the bond length for all H-* bonds (i.e. any bond to a hydrogen atom). Doing this allows choosing a larger time step. If this option is disabled, the TimeStep needs to be reduced manually.

Frequency**Type**

Integer

Default value

10000

Description

The frequency in MD steps at which new molecules will be added to the system.

NumMolecules**Type**

Integer

Description

The number of molecules that we will try to deposit. If not specified the number will be determined automatically such that the box becomes approximately full.

Temperature**Type**

Float

Default value

600.0

Description

The temperature at which the deposition happens.

TimeStep**Type**

Float

Default value

1.0

Unit

Femtoseconds

Description

The time difference per step.

LAMMPSOffload**Type**

Block

Description

Offload the calculation to LAMMPS via AMSPipe.

Enabled**Type**

Bool

Default value

No

Description

Enable offloading the force field evaluation to LAMMPS instead of handling it internally in AMS.

UseGPU**Type**

Bool

Default value

No

GUI name

Use GPU

Description

Accelerate LAMMPS calculations using a GPU. Requires a LAMMPS library built with the GPU package.

UseGPUForKSpace**Type**

Bool

Default value

Yes

Description

When UseGPU is enabled, also use the GPU to accelerate reciprocal space electrostatic interactions. Disabling this can improve performance on less powerful GPUs.

UseGPUForNeighbor**Type**

Bool

Default value

Yes

Description

When UseGPU is enabled, also use the GPU to accelerate neighbor searches. Disabling this can improve performance on less powerful GPUs.

UseOpenMP**Type**

Bool

Default value

No

GUI name

Use OpenMP

Description

Parallelize LAMMPS calculations using OpenMP threading. Requires a LAMMPS library built with the OMP package.

LoadSystem**Type**

Block

Recurring

True

Description

Block that controls reading the chemical system from a KF file instead of the [System] block.

File**Type**

String

Description

The path of the KF file from which to load the system. It may also be the results directory containing it.

Section**Type**

String

Default value

Molecule

Description

The section on the KF file from which to load the system.

Molecule**Type**

Block

Recurring

True

GUI name

Molecules

Description

Specification of the molecule to be deposited.

MoleFraction**Type**

Float

Default value

1.0

GUI name

Molar fraction

Description

The relative occurrence of the molecule with regard to other deposited species. Only relevant for mixed molecule depositions.

SystemName**Type**

String

GUI name

Molecule

Description

String ID of a named [System] to be inserted. The lattice specified with this System, if any, is ignored and the main system's lattice is used instead.

RestartWorkdir**Type**

String

Description

Uses the data from the working directory of a previously run deposition workflow for restarting. Under the hood this uses the normal rerun-prevention available in PLAMS: it may reuse results from old jobs instead of running them again.

System**Type**

Block

Recurring

True

Description

Specification of the chemical system. For some applications more than one system may be present in the input. In this case, all systems except one must have a non-empty string ID specified after the System keyword. The system without an ID is considered the main one.

AllowCloseAtoms**Type**

Bool

Default value

No

Description

If AllowCloseAtoms is set to False, the AMS driver will stop with an error if it detects almost-coinciding atomic coordinates. If set to True, the AMS driver will try to carry on with the calculation.

Atoms**Type**

Non-standard block

Description

The atom types and coordinates. Unit can be specified in the header. Default unit is Angstrom.

BondOrders**Type**

Non-standard block

Description

Defined bond orders. Each line should contain two atom indices, followed by the bond order (1, 1.5, 2, 3 for single, aromatic, double and triple bonds) and (optionally) the cell shifts for periodic systems. May be used by MM engines and for defining constraints. If the system is periodic and none of the bonds have the cell shift defined then AMS will attempt to determine them following the minimum image convention.

Charge

Type

Float

Default value

0.0

GUI name

Total charge

Description

The system's total charge in atomic units.

ElectrostaticEmbedding**Type**

Block

Description

Container for electrostatic embedding options, which can be combined.

ElectricField**Type**

Float List

Unit

V/Angstrom

Description

External homogeneous electric field with three Cartesian components: ex, ey, ez, the default unit being V/Å.

In atomic units: Hartree/(e bohr) = 51.422 V/Angstrom; the relation to SI units is: 1 Hartree/(e bohr) = 5.14 ... e11 V/m.

Supported by the engines adf, band, dftb and mopac.

For periodic systems the field may only have nonzero components orthogonal to the direction(s) of periodicity (i.e. for 1D periodic system the x-component of the electric field should be zero, while for 2D periodic systems both the x and y components should be zero. This options cannot be used for 3D periodic systems.

MultipolePotential**Type**

Block

Description

External point charges (and dipoles).

ChargeModel**Type**

Multiple Choice

Default value

Point

Options

[Point, Gaussian]

Description

A multipole may be represented by a point (with a singular potential at its location) or by a spherical Gaussian distribution.

ChargeWidth**Type**

Float

Default value

-1.0

Description

The width parameter in a.u. in case a Gaussian charge model is chosen. A negative value means that the width will be chosen automatically.

Coordinates**Type**

Non-standard block

Description

Positions and values of the multipoles, one per line. Each line has the following format:

x y z q, or

x y z q μ_x μ_y μ_z .

Here x, y, z are the coordinates in Å, q is the charge (in atomic units of charge) and μ_x , μ_y , μ_z are the (optional) dipole moment components (in atomic units, i.e. e*Bohr).

Periodic systems are not supported.

FractionalCoords**Type**

Bool

Default value

No

Description

Whether the atomic coordinates in the Atoms block are given in fractional coordinates of the lattice vectors. Requires the presence of the Lattice block.

GeometryFile**Type**

String

Description

Read the geometry from a file (instead of from Atoms and Lattice blocks). Supported formats: .xyz

GuessBonds**Type**

Bool

Default value

No

Description

Whether or not UFF bonds should be guessed.

Lattice**Type**

Non-standard block

Description

Up to three lattice vectors. Unit can be specified in the header. Default unit is Angstrom.

LatticeStrain**Type**

Float List

Description

Deform the input system by the specified strain. The strain elements are in Voigt notation, so one should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

LoadForceFieldAtomTypes**Type**

Block

Description

This is a mechanism to set the ForceField.Type attribute in the input. This information is currently only used by the ForceField engine.

File**Type**

String

Description

Name of the (kf) file. It needs to be the result of a forcefield calculation.

LoadForceFieldCharges**Type**

Block

Recurring

True

Description

This is a mechanism to set the ForceField.Charge attribute in the input. This information is currently only used by the ForceField engine.

CheckGeometryRMSD**Type**

Bool

Default value

No

Description

Whether the geometry RMSD test should be performed, see MaxGeometryRMSD. Otherwise only basic tests are performed, such as number and atom types. Not doing the RMSD test allows you to load molecular charges in a periodic system.

File**Type**

String

Description

Name of the (kf) file

MaxGeometryRMSD

Type

Float

Default value

0.1

Unit

Angstrom

Description

The geometry of the charge producing calculation is compared to the one of the region, and need to be the same within this tolerance.

Region**Type**

String

Default value

*

Description

Region for which the charges should be loaded

Section**Type**

String

Default value

AMSResults

Description

Section name of the kf file

Variable**Type**

String

Default value

Charges

Description

Variable name of the kf file

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

For periodic systems the atoms will be moved to the central cell.

ModifyAlternativeElements**Type**

Bool

Default value

No

Description

When using alternative elements (using the `nuclear_charge` attribute) set the element to the nearest integer Z . If you specify an H atom with a `nuclear_charge` of 2.9 it is replaced by a Li atom with the same nuclear charge.

PerturbCoordinates**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Perturb the atomic coordinates by adding random numbers between `[-PerturbCoordinates,PerturbCoordinates]` to each Cartesian component. This can be useful if you want to break the symmetry of your system (e.g. for a geometry optimization).

PerturbLattice**Type**

Float

Default value

0.0

Description

Perturb the lattice vectors by applying random strain with matrix elements between `[-PerturbLattice,PerturbLattice]`. This can be useful if you want to deviate from an ideal symmetric geometry, for example if you look for a phase change due to high pressure.

RandomizeAtomOrder**Type**

Bool

Default value

No

Description

Whether or not the order of the atoms should be randomly changed. Intended for some technical testing purposes only. Does not work with bond information.

Region**Type**

Block

Recurring

True

Description

Properties for each region specified in the Atoms block.

Properties**Type**

Non-standard block

Description

Properties for each region specified in the Atoms block.

ShiftCoordinates**Type**

Float List

Unit

Bohr

Description

Translate the atoms by the specified shift (three numbers).

SuperCell**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems). The integer numbers represent the diagonal elements of the supercell transformation; you should specify as many numbers as lattice vectors (i.e. 1 number for 1D, 2 numbers for 2D and 3 numbers for 3D periodic systems).

SuperCellTrafo**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems) $\vec{a}'_i = \sum_j T_{ij} \vec{a}_j$. The integer numbers represent the supercell transformation T_{ij} : 1 number for 1D PBC, 4 numbers for 2D PBC corresponding to a 2x2 matrix (order: (1,1),(1,2),(2,1),(2,2)) and 9 numbers for 3D PBC corresponding to a 3x3 matrix (order: (1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)).

Symmetrize**Type**

Bool

Default value

No

Description

Whether to symmetrize the input structure. This might also rototranslate the structure into a standard orientation. This will symmetrize the atomic coordinates to machine precision. Useful if the system is almost symmetric or to rototranslate a symmetric molecule into a standard orientation.

Symmetry**Type**

Multiple Choice

Default value

AUTO

Options

[AUTO, NOSYM, C(LIN), D(LIN), C(I), C(S), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(2V), C(3V), C(4V), C(5V), C(6V), C(7V), C(8V), C(2H), C(3H), C(4H), C(5H), C(6H), C(7H), C(8H), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(2D), D(3D), D(4D), D(5D), D(6D), D(7D), D(8D), D(2H), D(3H), D(4H), D(5H), D(6H), D(7H), D(8H), I, I(H), O, O(H), T, T(D), T(H), S(4), S(6), S(8)]

Description

Use (sub)symmetry with this Schoenflies symbol. Can only be used for molecules. Orientation should be correct for the (sub)symmetry. If used icw Symmetrize, the symmetrization will not reorient the molecule.

15.2.7 oled-properties**CoresPerJob****Type**

Integer

Default value

8

Description

The number of CPU cores used for each job in the workflow. Combined with the total number of cores used (set by the NSCM environment variable or the `-n` command line argument), this indirectly determines the number of simultaneously running jobs. The default value should usually be a good choice. When changing this value, make sure you are using all allocated cores by setting a value that divides the total number of cores, as well as the number of cores on each node.

Embedding**Type**

Block

Description

Configures details of how the environment is taken into account.

Charges**Type**

Multiple Choice

Default value

DFT

Options

[DFTB, DFT]

Description

Which atomic charges to use for the DRF embedding.

- DFTB: Use the self-consistent Mulliken charges from a quick DFTB calculation with the GFN1-xTB model.
- DFT: Use the MDC-D charges from a relatively quick DFT calculation using LDA and a DZP basis set.

Cutoff**Type**

Float

Default value

15.0

Unit

Angstrom

Description

The cutoff distance determining which molecules will be considered the environment of the central molecule. The maximum possible cutoff distance is half the length of the smallest lattice vector. The distance can be measured using different metrics, see the `Metric` keyword.

Metric**Type**

Multiple Choice

Default value

Atoms

Options

[CoM, Atoms, Atoms_noH]

Description

The metric used to calculate the distance between two molecules.

- CoM: use the distance between the centers of mass of the two molecules.
- Atoms: Use the distance between the two closest atoms of two molecules.
- Atoms_noH: Use the distance between the closest non-hydrogen atoms of the two molecules.

Type**Type**

Multiple Choice

Default value

DRF

Options

[None, DRF]

Description

The type of embedding used to simulate the molecular environment.

GW**Type**

Block

Description

Perform a GW calculation. G0W0 is the default for `GW%SelfConsistency`.

AdaptiveMixing**Type**

Float List

Description

Requests to use adaptive mixing instead of DIIS and sets the starting mixing parameter for mixing of Green's function in case of self-consistency.

Adaptive mixing is recommended in case a qsGW calculation does not converge with DIIS.

It is ignored in non-selfconsistent calculation and overwritten by DIIS when DIIS is also present.

AnalyticalIntegration**Type**

Block

Description

Use analytical integration to calculate the self-energy. Very slow, unless the system is very small but useful to check the accuracy of the frequency integration

Enabled**Type**

Bool

Default value

No

GUI name

analytical integration

Description

Enable the calculation of the GW quasi-particle energies via analytical integration.

SpectralFunctionResolution**Type**

Integer

Default value

800

Description

Number of points at which spectral function is evaluated.

TDA**Type**

Bool

Default value

No

Description

Solve the linear response equations in the Tamm-Dancoff approximation.

eta**Type**

Float

Default value

0.001

Description

Artificial (positive) broadening parameter for evaluation of self-energy in analytical integration.

Ideally should be as small as possible but this might lead to convergence issues in partially self-consistent approaches.

In this case, a value of up to 0.1 is possible.

Converge**Type**

Block

Description

Sets convergence criteria for the GW calculation in self-consistent case

Density**Type**

Float List

Default value

[1e-08, 1e-05]

Description

First Criterion for self-consistency procedure to terminate.

Criterion is the trace of the density matrix. Ignored in non-selfconsistent Calculation and in eigenvalue self-consistent GW

It is possible to run a qsGW calculation with an inner SCF loop which updates the static part of the elf-energy only. This can be useful to accelerate the convergence in case linear mixing is used. It is not recommended to use linear mixing, so it is also not recommended to use that inner loop as well. The second number in this list specifies the convergence criterion for the inner SCF loop.

HOMO**Type**

Float

Default value

0.003

Unit

eV

GUI name

HOMO energy convergence

Description

Criterion for self-consistency procedure to terminate.

The self-consistent GW calculation terminates, when the difference between the HOMO QP energies between 2 consecutive iterations is below this number.

The LUMO energy converged faster than the HOMO energy so when the HOMO energy is converged according to this criterion, the LUMO energy will be converged as well.

In non-selfconsistent Calculation, this criterion is ignored.

DIIS**Type**

Integer

Default value

10

Description

Requests to use DIIS. This is the Default. Number of expansion coefficients can be requested as well. Ignored in non-selfconsistent calculation

Enabled**Type**

Bool

Default value

No

GUI name

Calculate GW quasi-particle energies

Description

Enable the calculation of the GW quasi-particle energies.

LinearMixing**Type**

Float List

Description

Requests to use linear mixing instead of DIIS and sets the mixing parameter for linear mixing of Green's function in case of self-consistency.

It is ignored in non-selfconsistent calculation and overwritten by DIIS when DIIS is also present.

LinearizeQPequations**Type**

Bool

Default value

No

Description

Instead of solving the non-linear QP equations in a G0W0 (or evGW calculation) by bisection exactly, linearize them by first-order Taylor expansion.

This is not recommended since it does not save computational time when used together with analytical continuation (as implemented in AMS). It might however be useful for benchmarking or for validating results.

If the results of the linearization differ by a lot (for instance, more than 0.1 eV in frontier QP energies) from the non-linearized results, this might indicate that the GW calculation is not reliable.

Polarizability**Type**

Multiple Choice

Default value

RPA

Options

[RPA, BSE, G4W1, G4V1, TDHF]

Description

Sets the expression for the Polarizability used in the GW calculation.

RPA is the Default and amounts to a standard GW calculation.

BSE denotes screening in the Bethe-Salpeter-equation formalism.

PrintAllSolutions**Type**

Bool

Default value

No

Description

Print out all solutions for all requested states. Detects multiple solutions of the QP equations.

PrintSpectralFunction**Type**

Bool

Default value

No

Description

Plot the self-energy as a function of frequency. Automatically done in case of analytical continuation. However, this is expensive in the analytical integration formalism.

QPHamiltonian**Type**

Multiple Choice

Default value

KSF2

Options

[KSF1, KSF2, SRG, LQSGW]

Description

The quasi-particle Hamiltonian can be constructed in different ways.

KSF1 refers to the original construction by Kotani, Van Schilfgaarde and Faleev (KSF) which is also implemented in TURBOMOLE.

KSF2 refers to an alternative construction by KSF.

KSF1 is not recommended since it is numerically less stable than KSF2 in case analytical continuation is used (the default).

In case the analytical integration algorithm is used, only KSF1 is implemented. Therefore, make sure that KSF1 is specified. The results are typically very similar.

The QP energies at which the matrix elements are evaluated can be tweaked further, see the two subsequent keys: However, KSF2 is recommended since it typically leads to QP energies with the best agreement with experiment.

Ignored when not a quasi-particle self-consistent GW calculation is performed

ScissorShift**Type**

Bool

Default value

No

Description

Only calculate the HOMO and LUMO QP energies and shift the remaining QP energies by the same amount.

This is a rather crude approximation and not recommended.

It might again be useful for benchmarking purposes.

SelfConsistency**Type**

Multiple Choice

Default value

G0W0

Options

[G0W0, EVGW0, EVGW, QSGW0, QSGW]

Description

Sets the level of self-consistency in a GW calculation.

G0W0 calculates a one-shot, perturbative correction to the KS eigenvalues.

In evGW and evGW0, the quasi-particle energies are updated until self-consistency is reached.

evGW0 requests that the Green's function is evaluated self-consistently but not the screened interaction.

In qsGW, the density is updated as well, however, the self-energy is mapped to a static effective potential and the Dyson equation is solved by diagonalization instead of inversion. The results of a qsGW are independent of the choice of the underlying exchange-correlation functional and are usually the most accurate ones.

The same is done in qsGW0, but the screened interaction is not updated.

SelfEnergy**Type**

Multiple Choice

Default value

GW

Options

[HF, GW, G3W2, SOSEX, GWGamma, G3W2dynamic, GWGammaInf, GWGammaInfDyn]

Description

Controls the form of the self-energy.

GW is the default and corresponds to the standard GW calculation.

G3W2 is a GW calculation plus a perturbative second-order statically screened exchange correction (second order expansion in the self-energy). Note, that there the self-energy is always static.

nIterations**Type**

Integer List

Default value

[10]

GUI name

Number of iterations

Description

The maximum number of iterations within the (partially or fully) self-consistent GW calculation has to converge.

Ignored when Formalism is set to G0W0

nLowest**Type**

Integer

Default value

1

GUI name

N Lowest

Description

Number of lowest occupied QP levels to be evaluated, overwrites nStates'

nStates**Type**

Integer

Default value

5

GUI name

N states

Description

Number of Quasiparticle States to be printed to output.

The default is 5 states which in this case means that min(5, Number of particle states) occupied and min(5, Number of hole states) hole states are printed. The whole list of states can be printed by setting this parameter to -1'

LoadSystem**Type**

Block

Recurring

True

Description

Block that controls reading the chemical system from a KF file instead of the [System] block.

File**Type**

String

Description

The path of the KF file from which to load the system. It may also be the results directory containing it.

Section**Type**

String

Default value

Molecule

Description

The section on the KF file from which to load the system.

LogProgressEvery**Type**

Float

Default value

600.0

Unit

Seconds

Description

How often to print progress information to the logfile.

MBPT**Type**

Block

Description

Technical aspects of the MP2 algorithm.

Dependency**Type**

Bool

Default value

Yes

Description

If true, to improve numerical stability, almost linearly-dependent combination of basis functions are removed from the Green's function that are used in the MBPT equations. Disabling this key is strongly discouraged. Its value can however be changed. The key to adjust this value is `RiHartreeFock%DependencyThreshold`

ExcludeCore**Type**

Bool

Description

If active, excludes core states from the calculation of the optimal imaginary time and frequency grids.

The core states are still included in all parts of the calculations.

In case a frozen core calculation is performed, this key is ignored.

For MP2 and double hybrid calculation, it defaults to false. For RPA and GW calculations, it defaults to true.

FitSetQuality**Type**

Multiple Choice

Default value

Auto

Options

[Auto, VeryBasic, Basic, Normal, Good, VeryGood]

Description

Specifies the fit set to be used in the MBPT calculation.

'Normal' quality is generally sufficient for basis sets up to and including TZ2P.

For larger basis sets (or for benchmarking purposes) a 'VeryGood' fit set is recommended. Note that the `FitSetQuality` heavily influences the computational cost of the calculation.

If not specified or 'Auto', the `RiHartreeFock%FitSetQuality` is used.

Formalism**Type**

Multiple Choice

Default value

Auto

Options

[Auto, RI, LT, All]

Description

Specifies the formalism for the calculation of the MP2 correlation energy.

‘LT’ means Laplace Transformed MP2 (also referred to as AO-PARI-MP2),

‘RI’ means that a conventional RI-MP2 is carried out.

If ‘Auto’, LT will be used in case of DOD double hybrids and SOS MP2, and RI will be used in all other cases.

‘All’ means that both RI and LT formalisms are used in the calculation.

For a RPA or GW calculation, the formalism is always LT, irrespective of the formalism specified with this key.

FrequencyGridType**Type**

Multiple Choice

Default value

LeastSquare

Options

[LeastSquare, GaussLegendre]

Description

Use Gauss-legendre grid for imaginary frequency integration in RPA and GW calculations instead of the usually used Least-Square optimized ones. Has the advantage that it can be systematically converged and an arbitrary number of grid points can be used. Typically more grid points will be needed to get the same level of accuracy. However, the convergence of the results with the size of the grid can be more systematic. These grids can only be used when Formalism is set to RI.

IntegrationQuality**Type**

Multiple Choice

Options

[VeryBasic, Basic, Normal, Good, VeryGood]

Description

Specifies the integration quality to be used in the MBPT calculation. If not specified, the RI-HartreeFock%IntegrationQuality is used.

SigmaFunctionalParametrization**Type**

Multiple Choice

Default value

S1re

Options

[W1, W2, S1, S2, S1re]

Description

Only relevant if a sigma-functional calculation is performed. Possible choices for the parametrization of the sigma-functional. Not all options are supported for all functionals.

ThresholdQuality**Type**

Multiple Choice

Options

[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description

Controls the distances between atomic centers for which the product of two basis functions is not fitted any more. Especially for spatially extended, large systems, 'VERYBASIC' and 'BASIC' can lead to large computational savings, but the fit is also more approximate. If not specified, the RIHartreeFock%ThresholdQuality is used.

UseScaledZORA**Type**

Bool

Default value

Yes

Description

If true, use the scaled ZORA orbital energies instead of the ZORA orbital energies in the MBPT equations.

frozenscore**Type**

Bool

Default value

No

Description

Freeze core states in correlation part of MBPT calculation

nCore**Type**

Integer

Default value

0

GUI name

Number of core levels

Description

Number of core states which will be excluded from the correlated calculation.

Will be ignored if frozenscore is false.

In case nothing is specified, the number of core levels will be determined automatically.

Needs to be smaller than the number of occupied states.

nFreqIntegration

Type

Integer

Default value

32

GUI name

N freq points for G3W2 integration

Description

Number of imaginary frequency points for G3W2 integration

nFrequency**Type**

Integer

Default value

12

GUI name

N freq points

Description

Number of imaginary frequency points. This key is only relevant for RPA and GW and will be ignored if used in an AO-PARI-MP2 calculation. 12 Points is the default for a RPA calculation. It is technically possible to use a different number of imaginary frequency points than for imaginary time. The maximum number of points which can be requested for imaginary frequency integration is 42. Important note: The computation time and memory requirements roughly scale linearly with the number of imaginary frequency points. However, memory can be an issue for RPA and GW when the number of imaginary frequency points is high. In case a job crashes, it is advised to increase the number of nodes since the necessary memory distributes over all nodes.

nLambda**Type**

Integer

Default value

1

GUI name

Number of lambda points

Description

Size of coupling constant integration grid for SOSEX variants in RPA. Default is 4 points

nTime**Type**

Integer

GUI name

Number of time points

Description

Number of imaginary time points (only relevant in case the Laplace Transformed (LT) formalism is used).

In the many-body-perturbation theory module in ADF, the polarizability (or Kohn-Sham density response function) is evaluated in imaginary time to exploit sparsity in the AO basis. For MP2, this is often referred to as a Laplace transform. For MP2, 9 points are the default. This is

a safe choice, guaranteeing accuracies higher than 1 KJ/mol for most systems (For many simple organic systems, 6 points are sufficient for good accuracy).

Only for systems with a very small HOMO-LUMO gap or low-lying core states (heavy elements starting from the 4th row of the periodic table) more points might be necessary.

In principle, the same considerations apply for RPA and GW as well, however, the accuracy requirements are somewhat higher and 12 point are the default for RPA. In a GW calculation, the number of points is adjusted according to the numerical quality. Using less than 9 points is strongly discouraged except for the simplest molecules.

In ADF2019, it can happen that the algorithm determining the imaginary time grid does not converge. In this case, the usual reason is that the number of points is too small and more points need to be specified. Starting from AMS2020, this does not happen any more. In case the imaginary time grid does not converge, the number of points is automatically adjusted until it does.

The computation time of AO-PARI-MP2, RPA, and GW scales linearly with the number of imaginary time points.

useGreenXgrids

Type

Bool

Default value

No

Description

Use GreenX library to generate grid points. This is recommended for larger number of grid points (> 20). Up to 34 points can be requested.

NumAdditionalOrbitalEnergies

Type

Integer

Default value

1

Description

The number of additional orbital energies to write to the HDF5 file. A value of N means to write everything up to HOMO-N and LUMO+N.

NumExcitations

Type

Integer

Default value

1

Description

The number of excited states to calculate. By default the S_1 and T_1 states will be calculated. The calculation of excited states is currently only supported for systems with a closed-shell ground state.

NumSelectedMoleculesPerSpecies

Type

Integer

Description

Number of molecules per species to calculate properties for. Around 50 molecules per species

should be enough to estimate distribution means and standard deviations as input for the Bumblebee KMC code. Mutually exclusive with the `SelectedMolecules` keyword. If neither this key nor `SelectedMolecules` is present, all molecules will be selected.

OccupationSmearing

Type

Multiple Choice

Default value

Ions

Options

[None, Ions, All]

Description

Determines for which systems the electron smearing feature in ADF will be used. If enabled, the molecular orbital occupations will be smeared out with a 300K Fermi-Dirac distribution. This makes SCF convergence easier, as the occupation of energetically close orbitals does not jump when their energetic order flips. See the ADF manual for details. It is recommended to keep this option enabled for the ionic systems, which are more likely to suffer from difficult SCF convergence.

Relax

Type

Multiple Choice

Default value

All

Options

[None, Neutral, All]

Description

Which geometries to relax prior to taking the energy differences for the calculation of ionization potential and electron affinity. The relaxation is done at the DFTB level using the GFN1-xTB model Hamiltonian with electrostatic embedding in a UFF environment.

- None: Use the geometries directly from the input.
- Neutral: Relax the uncharged molecule and use its optimized geometry for the neutral as well as the ionic systems. This gives (approximately) the vertical ionization potential and electron affinity.
- All: Individually relax the neutral systems and the ions before calculating the total energies. This gives (approximately) the adiabatic ionization potential and electron affinity.

Restart

Type

String

Description

The HDF5 file from a previous calculation on the same morphology. Data already calculated on the restart file will just be copied over and not be recalculated.

SelectedMolecules

Type

Integer List

Description

Indices of the molecules to calculate properties for. Note that indexing starts at 0. Mutually exclusive with the NumSelectedMoleculesPerSpecies keyword. If neither this key nor NumSelectedMoleculesPerSpecies is present, all molecules will be selected.

StoreResultFiles**Type**

Multiple Choice

Default value

Failed

Options

[None, Failed, All]

Description

Whether to keep the full result files from all the individual jobs. By default the result files from all jobs for a particular molecule will be deleted after all relevant results have been extracted and stored on the HDF5 file. Note that keeping the full results for all molecules can easily require hundreds of gigabytes of storage space.

System**Type**

Block

Recurring

True

Description

Specification of the chemical system. For some applications more than one system may be present in the input. In this case, all systems except one must have a non-empty string ID specified after the System keyword. The system without an ID is considered the main one.

AllowCloseAtoms**Type**

Bool

Default value

No

Description

If AllowCloseAtoms is set to False, the AMS driver will stop with an error if it detects almost-coinciding atomic coordinates. If set to True, the AMS driver will try to carry on with the calculation.

Atoms**Type**

Non-standard block

Description

The atom types and coordinates. Unit can be specified in the header. Default unit is Angstrom.

BondOrders**Type**

Non-standard block

Description

Defined bond orders. Each line should contain two atom indices, followed by the bond order

(1, 1.5, 2, 3 for single, aromatic, double and triple bonds) and (optionally) the cell shifts for periodic systems. May be used by MM engines and for defining constraints. If the system is periodic and none of the bonds have the cell shift defined then AMS will attempt to determine them following the minimum image convention.

Charge**Type**

Float

Default value

0.0

GUI name

Total charge

Description

The system's total charge in atomic units.

ElectrostaticEmbedding**Type**

Block

Description

Container for electrostatic embedding options, which can be combined.

ElectricField**Type**

Float List

Unit

V/Angstrom

Description

External homogeneous electric field with three Cartesian components: ex, ey, ez, the default unit being V/Å.

In atomic units: $\text{Hartree}/(e \text{ bohr}) = 51.422 \text{ V/Angstrom}$; the relation to SI units is: $1 \text{ Hartree}/(e \text{ bohr}) = 5.14 \dots e11 \text{ V/m}$.

Supported by the engines adf, band, dftb and mopac.

For periodic systems the field may only have nonzero components orthogonal to the direction(s) of periodicity (i.e. for 1D periodic system the x-component of the electric field should be zero, while for 2D periodic systems both the x and y components should be zero. This options cannot be used for 3D periodic systems.

MultipolePotential**Type**

Block

Description

External point charges (and dipoles).

ChargeModel**Type**

Multiple Choice

Default value

Point

Options

[Point, Gaussian]

Description

A multipole may be represented by a point (with a singular potential at its location) or by a spherical Gaussian distribution.

ChargeWidth**Type**

Float

Default value

-1.0

Description

The width parameter in a.u. in case a Gaussian charge model is chosen. A negative value means that the width will be chosen automatically.

Coordinates**Type**

Non-standard block

Description

Positions and values of the multipoles, one per line. Each line has the following format:

x y z q, or

x y z q μ_x μ_y μ_z .

Here x, y, z are the coordinates in Å, q is the charge (in atomic units of charge) and μ_x , μ_y , μ_z are the (optional) dipole moment components (in atomic units, i.e. e*Bohr).

Periodic systems are not supported.

FractionalCoords**Type**

Bool

Default value

No

Description

Whether the atomic coordinates in the Atoms block are given in fractional coordinates of the lattice vectors. Requires the presence of the Lattice block.

GeometryFile**Type**

String

Description

Read the geometry from a file (instead of from Atoms and Lattice blocks). Supported formats: .xyz

GuessBonds**Type**

Bool

Default value

No

Description

Whether or not UFF bonds should be guessed.

Lattice**Type**

Non-standard block

Description

Up to three lattice vectors. Unit can be specified in the header. Default unit is Angstrom.

LatticeStrain**Type**

Float List

Description

Deform the input system by the specified strain. The strain elements are in Voigt notation, so one should specify 6 numbers for 3D periodic system (order: xx,yy,zz,yz,xz,xy), 3 numbers for 2D periodic systems (order: xx,yy,xy) or 1 number for 1D periodic systems.

LoadForceFieldAtomTypes**Type**

Block

Description

This is a mechanism to set the ForceField.Type attribute in the input. This information is currently only used by the ForceField engine.

File**Type**

String

Description

Name of the (kf) file. It needs to be the result of a forcefield calculation.

LoadForceFieldCharges**Type**

Block

Recurring

True

Description

This is a mechanism to set the ForceField.Charge attribute in the input. This information is currently only used by the ForceField engine.

CheckGeometryRMSD**Type**

Bool

Default value

No

Description

Whether the geometry RMSD test should be performed, see MaxGeometryRMSD. Otherwise only basic tests are performed, such as number and atom types. Not doing the RMSD test allows you to load molecular charges in a periodic system.

File

Type

String

Description

Name of the (kf) file

MaxGeometryRMSD**Type**

Float

Default value

0.1

Unit

Angstrom

Description

The geometry of the charge producing calculation is compared to the one of the region, and need to be the same within this tolerance.

Region**Type**

String

Default value

*

Description

Region for which the charges should be loaded

Section**Type**

String

Default value

AMSResults

Description

Section name of the kf file

Variable**Type**

String

Default value

Charges

Description

Variable name of the kf file

MapAtomsToUnitCell**Type**

Bool

Default value

No

Description

For periodic systems the atoms will be moved to the central cell.

ModifyAlternativeElements**Type**

Bool

Default value

No

Description

When using alternative elements (using the `nuclear_charge` attribute) set the element to the nearest integer *Z*. If you specify an H atom with a `nuclear_charge` of 2.9 it is replaced by a Li atom with the same nuclear charge.

PerturbCoordinates**Type**

Float

Default value

0.0

Unit

Angstrom

Description

Perturb the atomic coordinates by adding random numbers between `[-PerturbCoordinates,PerturbCoordinates]` to each Cartesian component. This can be useful if you want to break the symmetry of your system (e.g. for a geometry optimization).

PerturbLattice**Type**

Float

Default value

0.0

Description

Perturb the lattice vectors by applying random strain with matrix elements between `[-PerturbLattice,PerturbLattice]`. This can be useful if you want to deviate from an ideal symmetric geometry, for example if you look for a phase change due to high pressure.

RandomizeAtomOrder**Type**

Bool

Default value

No

Description

Whether or not the order of the atoms should be randomly changed. Intended for some technical testing purposes only. Does not work with bond information.

Region**Type**

Block

Recurring

True

Description

Properties for each region specified in the Atoms block.

Properties**Type**

Non-standard block

Description

Properties for each region specified in the Atoms block.

ShiftCoordinates**Type**

Float List

Unit

Bohr

Description

Translate the atoms by the specified shift (three numbers).

SuperCell**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems). The integer numbers represent the diagonal elements of the supercell transformation; you should specify as many numbers as lattice vectors (i.e. 1 number for 1D, 2 numbers for 2D and 3 numbers for 3D periodic systems).

SuperCellTrafo**Type**

Integer List

Description

Create a supercell of the input system (only possible for periodic systems) $\vec{a}'_i = \sum_j T_{ij} \vec{a}_j$. The integer numbers represent the supercell transformation T_{ij} : 1 number for 1D PBC, 4 numbers for 2D PBC corresponding to a 2x2 matrix (order: (1,1),(1,2),(2,1),(2,2)) and 9 numbers for 3D PBC corresponding to a 3x3 matrix (order: (1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)).

Symmetrize**Type**

Bool

Default value

No

Description

Whether to symmetrize the input structure. This might also rototranslate the structure into a standard orientation. This will symmetrize the atomic coordinates to machine precision. Useful if the system is almost symmetric or to rototranslate a symmetric molecule into a standard orientation.

Symmetry**Type**

Multiple Choice

Default value

AUTO

Options

[AUTO, NOSYM, C(LIN), D(LIN), C(I), C(S), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(2V), C(3V), C(4V), C(5V), C(6V), C(7V), C(8V), C(2H), C(3H), C(4H), C(5H), C(6H), C(7H), C(8H), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(2D), D(3D), D(4D), D(5D), D(6D), D(7D), D(8D), D(2H), D(3H), D(4H), D(5H), D(6H), D(7H), D(8H), I, I(H), O, O(H), T, T(D), T(H), S(4), S(6), S(8)]

Description

Use (sub)symmetry with this Schoenflies symbol. Can only be used for molecules. Orientation should be correct for the (sub)symmetry. If used icw Symmetrize, the symmetrization will not reorient the molecule.

TransferIntegrals**Type**

Block

Description

Configures the details of the calculation of electron and hole transfer integrals.

Exclude**Type**

Block

Description

Configures which dimers NOT to calculate transfer integrals for.

Cutoff**Type**

Float

Default value

4.0

Unit

Angstrom

GUI name

Exclude beyond

Description

Exclude dimers for which the distance is larger than this threshold. Acts as a quick pre-screening to reduce the number of dimers to calculate transfer integrals for.

Metric**Type**

Multiple Choice

Default value

Atoms

Options

[CoM, Atoms, Atoms_noH]

Description

The metric used to calculate the distance between two molecules.

- CoM: use the distance between the centers of mass of the two molecules.
- Atoms: Use the distance between the two closest atoms of two molecules.

- Atoms_noH: Use the distance between the closest non-hydrogen atoms of the two molecules.

Include**Type**

Block

Description

Configures which dimers transfer integrals are calculated for.

Cutoff**Type**

Float

Default value

4.0

Unit

Angstrom

GUI name

Include within

Description

Transfer integrals will be calculated for all molecule pairs within a cutoff distance from each other. This distance can be measured using different metrics, see the corresponding **Metric** keyword.

Metric**Type**

Multiple Choice

Default value

Atoms

Options

[CoM, Atoms, Atoms_noH]

Description

The metric used to calculate the distance between two molecules.

- CoM: use the distance between the centers of mass of the two molecules.
- Atoms: Use the distance between the two closest atoms of two molecules.
- Atoms_noH: Use the distance between the closest non-hydrogen atoms of the two molecules.

Type**Type**

Multiple Choice

Default value

Fast

Options

[None, Fast, Full]

Description

The method used for the calculation of the transfer integrals.

KF OUTPUT FILES

16.1 Accessing KF files

KF files are Direct Access binary files. KF stands for Keyed File: KF files are keyword oriented, which makes them easy to process by simple procedures. Internally all the data on KF files is organized into sections containing variables, so each datum on the file can be identified by the combination of section and variable.

All KF files can be opened using the [KFbrowser](#) GUI program:

```
$AMSBIN/kfbrowser path/to/ams.rkf
```

By default KFbrowser shows a just a curated summary of the results on the file, but you can make it show the raw section and variable structure by switching it to expert mode. To do this, click on **File → Expert Mode** or press **ctrl/cmd + e**.

KF files can be opened and read with [Command line tools](#).

For working with the data from KF files, it is often useful to be able to read them from Python. Using the [AMS Python Stack](#), this can easily be done with the [AKFReader](#) class:

```
>>> from scm.akfreader import AKFReader
>>> kf = AKFReader("path/to/ams.rkf")
>>> "Molecule%Coords" in kf
True
>>> kf.description("Molecule%Coords")
{
  '_type': 'float_array',
  '_shape': [3, 'nAtoms'],
  '_comment': 'Coordinates of the nuclei (x,y,z)',
  '_unit': 'Bohr'
}
>>> kf.read("Molecule%Coords")
array([[ -11.7770694 ,  -4.19739597,   0.04934546],
       [  -9.37471321,  -2.63234227,  -0.13448698],
       ...,
       [  10.09508738,  -1.06191208,   1.45286913],
       [  10.11689333,  -1.5080196 ,  -1.87916127]])
```

Tip: For a full overview of the available methods in AKFReader, see the [AKFReader API](#) documentation.

16.2 Sections and variables on ams.rkf

Note: The AMS driver creates an entire directory full of result files. Often there are multiple `.rkf` files in that directory. This page only documents the sections and variables in the `ams.rkf` file.

BindingSites

Section content: Information on binding sites for a adsorbate.

BindingSites%AdsorbateLabel

Type

string

Description

Label for the adsorbate.

BindingSites%AverEnergy

Type

float_array

Description

The average energy per site. The energy of all stationary states that at least has an atom attached to the given site contributes to this average.

Unit

hartree

Shape

[nSites]

BindingSites%Coords

Type

float_array

Description

The coordinates of the sites.

Unit

bohr

Shape

[3, nSites]

BindingSites%CoordsFrac

Type

float_array

Description

The fractional coordinates of the sites.

Shape

[3, nSites]

BindingSites%FromSites

Type

int_array

Description

Sites from where the connections start.

Shape

[nConnections]

BindingSites%Labels**Type**

lchar_string_array

Description

Arbitrary labels for the binding sites. They are assigned according to the number of atom neighbors.

BindingSites%LatticeDisplacements**Type**

int_array

Description

Links between neighboring sites across the periodic boundary conditions.

Shape

[:, nConnections]

BindingSites%nConnections**Type**

int

Description

The number of connections between sites.

BindingSites%nParentStates**Type**

int_array

Description

The number of stationary states contributing to the average and standard deviation of the energy. See AverEnergy and StdevEnergy.

Shape

[nSites]

BindingSites%nSites**Type**

int

Description

The number of sites.

BindingSites%ParentAtoms**Type**

int_array

Description

Atom id that is attached to the site in the corresponding parent state. See ParentStates.

Shape

[:,]

BindingSites%ParentStates**Type**

int_array

Description

Stationary states ids contributing to the average and standard deviation of the energy. See AverEnergy and StdevEnergy.

Shape

[:]

BindingSites%ReferenceRegionLabel**Type**

string

Description

Label for the reference region.

BindingSites%StdevEnergy**Type**

float_array

Description

Std. deviation for energy per site. The energy of all stationary states that at least has an atom attached to the given site contributes to this std. deviation.

Unit

hartree

Shape

[nSites]

BindingSites%ToSites**Type**

int_array

Description

Sites to where the connections end up.

Shape

[nConnections]

BinLog**Section content:****BinLog%Area (#)****Type**

float_array

Description

The area of the cell (only for 2D PBC). This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

bohr^2

BinLog%BiasEnergy (#)**Type**

float_array

Description

? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

BinLog%blockSize**Type**

int

Description

Explain the block-system... ?

BinLog%BoostFactor (#)**Type**

float_array

Description

The boost factor for hyper-dynamics. Expand? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

BinLog%ConservedEnergy (#)**Type**

float_array

Description

The conserved energy...? Some MD person, please fix. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

BinLog%currentEntryOpen**Type**

bool

Description

?

BinLog%Density (#)**Type**

float_array

Description

The density of the system (mass/simulation_cell_volume). Only for 3D PBC. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

dalton/bohr^3

BinLog%DipoleMoment_* (#)**Type**

float_array

Description

The dipolemoment of the full system. For periodic systems this value will often not make sense, because molecules are not mapped together

BinLog%Hypertime (#)

Type

float_array

Description

Hyper time for hyper-dynamics. Expand? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

femtosecond

BinLog%ItemName (#)**Type**

string

Description

?

BinLog%KineticEnergy (#)**Type**

float_array

Description

The kinetic energy of the system. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

BinLog%Length (#)**Type**

float_array

Description

The length of the cell (only for 1D PBC). This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

bohr

BinLog%MaxBiasEnergy (#)**Type**

float_array

Description

? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

BinLog%MaxBoostFactor (#)**Type**

float_array

Description

The maximum boost factor for hyper-dynamics. Expand? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

BinLog%nBlocks

Type
int

Description
Explain the block-system... ?

BinLog%nEntries

Type
int

Description
Number of MD history entries.

BinLog%PotentialEnergy (#)

Type
float_array

Description
The potential energy, i.e. the energy as computed by the engine. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit
hartree

BinLog%Pressure (#)

Type
float_array

Description
The pressure of the system (only for 3D PBC). This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit
hartree/bohr³

BinLog%PressureTensor_* (#)

Type
float_array

Description
?

BinLog%Step (#)

Type
int_array

Description
The step number of the MD calculation. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

BinLog%Temperature (#)

Type
float_array

Description
The temperature of the system, computed from the kinetic energy?. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

kelvin

BinLog%Time (#)**Type**

float_array

Description

The MD simulation time. This is a ‘blocked’ property. See the ‘blockSize’ and ‘nBlocks’ variables for more details.

Unit

femtosecond

BinLog%TotalEnergy (#)**Type**

float_array

Description

The total energy of the system: potential energy (as computed by the engine) + kinetic energy. This is a ‘blocked’ property. See the ‘blockSize’ and ‘nBlocks’ variables for more details.

Unit

hartree

BinLog%Volume (#)**Type**

float_array

Description

The volume of the cell (only for 3D PBC). This is a ‘blocked’ property. See the ‘blockSize’ and ‘nBlocks’ variables for more details.

Unitbohr³**ChemicalSystem(#)**

Section content: Various ‘versions’ of the chemical system. See also the ‘SystemVersionHistory’ section.

ChemicalSystem (#) %AtomicNumbers**Type**

int_array

Description

Atomic number ‘Z’ of the atoms in the system

Shape

[nAtoms]

ChemicalSystem (#) %AtomMasses**Type**

float_array

Description

Masses of the atoms

Unit

a.u.

Values range

[0, 'infinity']

Shape

[nAtoms]

ChemicalSystem (#) %AtomSymbols**Type**

string

Description

The atom's symbols (e.g. 'C' for carbon)

Shape

[nAtoms]

ChemicalSystem (#) %bondOrders**Type**

float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

ChemicalSystem (#) %Charge**Type**

float

Description

Net charge of the system

Unit

e

ChemicalSystem (#) %Coords**Type**

float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

Shape

[3, nAtoms]

ChemicalSystem (#) %eeAttachTo**Type**

int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

ChemicalSystem (#) %eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution.

ChemicalSystem(#)%eeEField**Type**

float_array

Description

The external homogeneous electric field.

Unit

hartree/(e*bohr)

Shape

[3]

ChemicalSystem(#)%eeLatticeVectors**Type**

float_array

Description

The lattice vectors used for the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNLatticeVectors]

ChemicalSystem(#)%eeMulti**Type**

float_array

Description

The values of the external point- or multipole- charges.

Unit

a.u.

Shape

[eeNZlm, eeNMulti]

ChemicalSystem(#)%eeNLatticeVectors**Type**

int

Description

The number of lattice vectors for the external point- or multipole- charges.

ChemicalSystem(#)%eeNMulti**Type**

int

Description

The number of external point- or multipole- charges.

ChemicalSystem(#)%eeNZlm

Type

int

Description

When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

ChemicalSystem(#)%eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened.

ChemicalSystem(#)%eeXYZ**Type**

float_array

Description

The position of the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNMulti]

ChemicalSystem(#)%EngineAtomicInfo**Type**

string_fixed_length

Description

Atom-wise info possibly used by the engine.

ChemicalSystem(#)%fromAtoms**Type**

int_array

Description

Index of the first atom in a bond. See the bondOrders array

ChemicalSystem(#)%latticeDisplacements**Type**

int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

ChemicalSystem(#)%LatticeVectors**Type**

float_array

Description

Lattice vectors

Unit

bohr

Shape

[3, nLatticeVectors]

ChemicalSystem(#)%nAtoms**Type**

int

Description

The number of atoms in the system

ChemicalSystem(#)%nAtomsTypes**Type**

int

Description

The number different of atoms types

ChemicalSystem(#)%nLatticeVectors**Type**

int

Description

Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values

[0, 1, 2, 3]

ChemicalSystem(#)%toAtoms**Type**

int_array

Description

Index of the second atom in a bond. See the bondOrders array

CrestMTDHistory

Section content: Data related to Crest MD.

CVHDBiasHistory

Section content: ?

CVHDBiasHistory%blockSize**Type**

int

Description

Explain the block-system... ?

CVHDBiasHistory%currentEntryOpen**Type**

bool

Description

?

CVHDBiasHistory%CVValue(#)**Type**

float_array

Description

?

CVHDBiasHistory%Height (#)**Type**

float_array

Description

?

CVHDBiasHistory%ItemName (#)**Type**

string

Description

?

CVHDBiasHistory%nBlocks**Type**

int

Description

Explain the block-system... ?

CVHDBiasHistory%nEntries**Type**

int

Description

Number of MD history entries.

CVHDBiasHistory%Step (#)**Type**

int_array

Description

The step number of the MD calculation. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

EnergyLandscape

Section content: Information on PES exploration.

EnergyLandscape%counts**Type**

int_array

Description

Number of times that the exploration job found each state.

Shape

[nStates]

EnergyLandscape%Edge (#)**Type**

subsection

Description

From acerxn ?

EnergyLandscape%Edge (#) %brokenAtomsFrom

Type

int_array

Description

From acerxn ?

EnergyLandscape%Edge (#) %brokenAtomsTo

Type

int_array

Description

From acerxn ?

EnergyLandscape%Edge (#) %formedAtomsFrom

Type

int_array

Description

From acerxn ?

EnergyLandscape%Edge (#) %formedAtomsTo

Type

int_array

Description

From acerxn ?

EnergyLandscape%edgesBegin

Type

int_array

Description

From acerxn ?

EnergyLandscape%edgesEnd

Type

int_array

Description

From acerxn ?

EnergyLandscape%energies

Type

float_array

Description

Energies of the stationary states.

Unit

hartree

Shape

[nStates]

EnergyLandscape%energyReferenceLabels

Type

string

Description

?

EnergyLandscape%energyReferenceValues**Type**

float_array

Description

?

EnergyLandscape%fileNames**Type**

ftl_string_array

Description

Filenames for the stationary state calculations.

Shape

[nStates]

EnergyLandscape%fragmentsEnergies**Type**

float_array

Description

?

EnergyLandscape%fragmentsFileNames**Type**

string

Description

?

EnergyLandscape%fragmentsRegions**Type**

string

Description

?

EnergyLandscape%fStatesAdsorptionPrefactors (#)**Type**

float_array

Description

?

EnergyLandscape%fStatesComposition (#)**Type**

int_array

Description

?

EnergyLandscape%fStatesConnections (#)**Type**

int_array

Description

?

EnergyLandscape%fStatesDesorptionPrefactors (#)**Type**

float_array

Description

?

EnergyLandscape%fStatesEnergy (#)**Type**

float

Description

?

EnergyLandscape%fStatesNConnections (#)**Type**

int

Description

?

EnergyLandscape%fStatesNFrags (#)**Type**

int

Description

?

EnergyLandscape%historyIndices**Type**

int_array

Description

Indices of the corresponding entries in the History section.

Shape

[nStates]

EnergyLandscape%isTS**Type**

bool_array

Description

Whether a state is a transition state or a minimum.

Shape

[nStates]

EnergyLandscape%names**Type**

string

Description

From acrxn ?

EnergyLandscape%nEdges

Type
int

Description
From acerxn ?

EnergyLandscape%nFragments

Type
int

Description
?

EnergyLandscape%nFStates

Type
int

Description
?

EnergyLandscape%nStates

Type
int

Description
The number of stationary states (points with vanishing gradient).

EnergyLandscape%prefactorsFromProduct

Type
float_array

Description
?

EnergyLandscape%prefactorsFromReactant

Type
float_array

Description
?

EnergyLandscape%prefactorsTemperature

Type
float

Description
?

EnergyLandscape%products

Type
int_array

Description
For stationary states that are a TS state this is the connected product.

Shape
[nStates]

EnergyLandscape%reactants

Type

int_array

Description

For stationary states that are a TS state this is the connected reactant.

Shape

[nStates]

EnergyLandscape%referenceRegionLabel**Type**

string

Description

?

EngineResults**Section content:** Description and names of engine result files.**EngineResults%Description (#)****Type**

string

Description

Description of the results.

EngineResults%Files (#)**Type**

string

Description

Filenames of the result files from the engine

EngineResults%nEntries**Type**

int

Description

Number of engine results files.

EngineResults%Title (#)**Type**

string

Description

Title of engine calculations.

GCMC**Section content:** Data related to the Gran Canonical Monte Carlo procedure**GCMC%AccessibleVolume****Type**

float

Description

input accessible volume

GCMC%AtomicNumbers

Type

int_array

Description

Atomic number 'Z' of the atoms in the system

Shape

[nAtoms]

GCMC%AtomMasses**Type**

float_array

Description

Masses of the atoms

Unit

a.u.

Values range

[0, 'infinity']

Shape

[nAtoms]

GCMC%AtomSymbols**Type**

string

Description

The atom's symbols (e.g. 'C' for carbon)

Shape

[nAtoms]

GCMC%bondOrders**Type**

float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

GCMC%Box**Type**

float_array

Description

Range of cell coordinates where atoms are allowed to be added (x,y,z min,max?)

Unit

bohr

Shape

[3, 2]

GCMC%ChangeVolume

Type

bool

Description

Allow volume changes

GCMC%Charge**Type**

float

Description

Net charge of the system

Unit

e

GCMC%Coords**Type**

float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

Shape

[3, nAtoms]

GCMC%eeAttachTo**Type**

int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

GCMC%eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution.

GCMC%eeEField**Type**

float_array

Description

The external homogeneous electric field.

Unit

hartree/(e*bohr)

Shape

[3]

GCMC%eeLatticeVectors

Type

float_array

Description

The lattice vectors used for the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNLatticeVectors]

GCMC%eeMulti**Type**

float_array

Description

The values of the external point- or multipole- charges.

Unit

a.u.

Shape

[eeNZlm, eeNMulti]

GCMC%eeNLatticeVectors**Type**

int

Description

The number of lattice vectors for the external point- or multipole- charges.

GCMC%eeNMulti**Type**

int

Description

The number of external point- or multipole- charges.

GCMC%eeNZlm**Type**

int

Description

When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

GCMC%eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened.

GCMC%eeXYZ**Type**

float_array

Description

The position of the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNMulti]

GCMC%EngineAtomicInfo**Type**

string_fixed_length

Description

Atom-wise info possibly used by the engine.

GCMC%fromAtoms**Type**

int_array

Description

Index of the first atom in a bond. See the bondOrders array

GCMC%HistoryAccepted**Type**

int_array

Description

result (1-accepted, 0-rejected)

Shape

[NIterMCtried]

GCMC%HistoryAMSEnergy**Type**

float_array

Description

energy (AMSEnergy)

Shape

[NIterMCtried]

GCMC%HistoryMCEnergy**Type**

float_array

Description

corrected MC energy (AMSEnergy - sum(ChemicalPotential))

Shape

[NIterMCtried]

GCMC%HistoryMoleculeIndex**Type**

int_array

Description

molecule index of the type

Shape

[NIterMCtried]

GCMC%HistoryMoleculeType**Type**

int_array

Description

molecule type

Shape

[NIterMCtried]

GCMC%HistoryMoveType**Type**

int_array

Description

action type

Shape

[NIterMCtried]

GCMC%HistoryVolume**Type**

float_array

Description

?

Shape

[NIterMCtried]

GCMC%InitialEnergy**Type**

float

Description

?

Unit

hartree

GCMC%Iterations**Type**

int

Description

Number of MC iterations before stopping

GCMC%latticeDisplacements**Type**

int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

GCMC%LatticeVectors

Type

float_array

Description

Lattice vectors

Unit

bohr

Shape

[3, nLatticeVectors]

GCMC%MapToOriginalCell**Type**

bool

Description

map atoms back to the original cell?

GCMC%MaxDistance**Type**

float

Description

Max radius for atom placement

Unit

bohr

GCMC%MinDistance**Type**

float

Description

Min radius for atom placement

Unit

bohr

GCMC%Mol#**Type**

subsection

Description

?

GCMC%Mol#%addedAtoms**Type**

archived_int_array

Description

Atom species indices

GCMC%Mol#%AtomicInfo**Type**

archived_string_array

Description**GCMC%Mol#%BondOrders**

Type

archived_float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

GCMC%Mol#%Charge**Type**

float

Description

Net charge of the system

Unit

e

GCMC%Mol#%chemPot**Type**

float

Description

?

GCMC%Mol#%eeAttachTo**Type**

archived_int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

GCMC%Mol#%eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution

GCMC%Mol#%eeEField**Type**

float_array

Description

The external homogeneous electric field

Unit

hartree/(e*bohr)

Shape

[3]

GCMC%Mol#%eeLatticeVectors**Type**

archived_float_array

Description

The lattice vectors used for the external point- or multipole- charges

Unit

bohr

GCMC%Mol#%eeMultipoles**Type**

archived_float_array

Description

The multiple charges.

Unit

bohr

GCMC%Mol#%eenMulti**Type**

int

Description

The number of multipoles.

GCMC%Mol#%eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened

GCMC%Mol#%eeXYZ**Type**

archived_float_array

Description

The position of the external point- or multipole- charges

Unit

bohr

GCMC%Mol#%FromAtoms**Type**

archived_int_array

Description

Index of the first atom in a bond. See the bondOrders array

GCMC%Mol#%hasLatticeDisplacements**Type**

bool

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

GCMC%Mol#%inserted**Type**

int_array

Description

?

GCMC%Mol#%kVectors**Type**

float_array

Description

Reciprocal lattice vectors (corresponding to the strained lattice vectors)

Unit

1/bohr

Shape

[3, 3]

GCMC%Mol#%LatticeDisplacements**Type**

archived_int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

GCMC%Mol#%name**Type**

string

Description

?

GCMC%Mol#%nAtoms**Type**

int

Description

The number of atoms in the system

GCMC%Mol#%nInsert**Type**

int

Description

?

GCMC%Mol#%noAR**Type**

bool

Description

?

GCMC%Mol#%nSpecies**Type**

int

Description

The number different of atoms types

GCMC%Mol#%nVectors**Type**

int

Description

Number of lattice vectors (0:molecule, 1:chain, 2:slab, 3:bulk).

GCMC%Mol#%removedAtoms**Type**

archived_int_array

Description

Atom species indices

GCMC%Mol#%sp#inputSymbol**Type**

string

Description

Symbol as specified in the input

GCMC%Mol#%sp#mass**Type**

float

Description

mass

GCMC%Mol#%sp#regions**Type**

archived_string_array

Description

regions

GCMC%Mol#%sp#symbol**Type**

string

Description

Element symbol

GCMC%Mol#%sp#Z**Type**

int

Description

Atomic number

GCMC%Mol#%SpIndices**Type**

archived_int_array

Description

Atom species indices

GCMC%Mol#%SPlen

Type
int

Description
Number of species

GCMC%Mol#%strain

Type
float_array

Description
The strain matrix.

Shape
[3, 3]

GCMC%Mol#%strainedVectors

Type
float_array

Description
Strained real space lattice vectors

Unit
bohr

Shape
[3, 3]

GCMC%Mol#%ToAtoms

Type
archived_int_array

Description
Index of the second atom in a bond. See the bondOrders array

GCMC%Mol#%unstrainedVectors

Type
float_array

Description
Real space lattice vectors (unstrained).

Unit
bohr

Shape
[3, 3]

GCMC%Mol#%version

Type
int

Description

GCMC%Mol#%xyzAtoms

Type
archived_float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

GCMC%MoleculeName (#)**Type**

string

Description

?

GCMC%MoveType (#)**Type**

string

Description

?

GCMC%nAtoms**Type**

int

Description

The number of atoms in the system

GCMC%nAtomsTypes**Type**

int

Description

The number different of atoms types

GCMC%NIterMCaccept**Type**

int

Description

Number of moves accepted

GCMC%NIterMCreject**Type**

int

Description

Number of moves rejected

GCMC%NIterMCtried**Type**

int

Description

Number of moves tried, aka iteration number

GCMC%nLatticeVectors**Type**

int

Description

Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values

[0, 1, 2, 3]

GCMC%NMCacceptAdd**Type**

int

Description

statistics about success rates of MC

GCMC%NMCacceptMove**Type**

int

Description

?

GCMC%NMCacceptRemove**Type**

int

Description

?

GCMC%NMCacceptVolume**Type**

int

Description

?

GCMC%NMCrejectAdd**Type**

int

Description

?

GCMC%NMCrejectMove**Type**

int

Description

?

GCMC%NMCrejectRemove**Type**

int

Description

?

GCMC%NMCrejectVolume**Type**

int

Description

?

GCMC%NMols**Type**

int

Description

?

GCMC%NonAccessibleVolume**Type**

float

Description

input non-accessible (vacuum) volume

GCMC%NumAttempts**Type**

int

Description

Maximum number of trial steps when inserting/moving a molecule

GCMC%Pressure**Type**

float

Description

NPT pressure

Unit

a.u.

GCMC%Temperature**Type**

float

Description

system temperature

Unit

kelvin

GCMC%toAtoms**Type**

int_array

Description

Index of the second atom in a bond. See the bondOrders array

GCMC%UseGCPreFactor**Type**

bool

Description

include GCE factors in probability?

GCMC%VolumeChangeMax

Type

float

Description

input max volume change factor for NPT MC

GCMC%VolumeOption**Type**

int

Description

input accessible volume? 0=no (use total cell volume) 1=yes 2=bulk

General**Section content:** General information about the AMS driver calculation.**General%account****Type**

string

Description

Name of the account from the license

General%CPUTime**Type**

float

Description

CPU time of the AMS calculation.

Unit

second

General%ElapsedTime**Type**

float

Description

Elapsed time of the AMS calculation.

Unit

second

General%engine**Type**

string

Description

The main engine of the calculation.

General%engine messages**Type**

string

Description

Message from the engine. In case the engine fails to solves, this may contains extra information on why.

General%file-ident

Type

string

Description

The file type identifier, e.g. RKF, RUNKF, TAPE21...

General%jobid**Type**

int

Description

Unique identifier for the job.

General%program**Type**

string

Description

The name of the program/engine that generated this kf file.

General%release**Type**

string

Description

The version of the program that generated this kf file (including svn revision number and date).

General%SysTime**Type**

float

Description

System time of the AMS calculation.

Unit

second

General%task**Type**

string

Description

The Task of the AMS driver (e.g. singlepoint, geometryoptimization, molecular dynamics...).

General%termination status**Type**

string

Description

The termination status. Possible values: 'NORMAL TERMINATION', 'NORMAL TERMINATION with warnings', 'NORMAL TERMINATION with errors', 'ERROR', 'IN PROGRESS'.

General%title**Type**

string

Description

Title of the calculation.

General%uid**Type**

string

Description

SCM User ID

General%user input**Type**

string

Description

The text input of the AMS calculation.

General%version**Type**

int

Description

Version number?

History

Section content: History of the system during the AMS calculation. What is stored here depends on the task of the AMS calculation. For example, for a GeometryOptimization this will contain the intermediate steps of the GO, while for a MoleculeDynamics calculation it will contain the MD frames.

History%Angle (#)**Type**

float

Description

IRC (Intrinsic Reaction Coordinate) angle((pivot->start),(pivot->coords)), indication of the path curvature.

Unit

degrees

History%ArcLength (#)**Type**

float

Description

IRC (Intrinsic Reaction Coordinate): length of arc(start->pivot->coords).

Unit

angstrom

History%Bonds**Type**

subsection

Description

?

History%Bonds%Atoms (#)

Type
int_array

Description
?

History%Bonds%CellShifts (#)

Type
int_array

Description
?

History%Bonds%Index (#)

Type
int_array

Description
?

History%Bonds%Orders (#)

Type
float_array

Description
?

History%Converged (#)

Type
bool

Description
Whether the entry corresponds to a converged structure. The meaning of ‘converged’ depends on the Task.

History%Coords (#)

Type
float_array

Description
Coordinates of the systems of a given entry.

Shape
[3, :]

History%currentEntryOpen

Type
bool

Description
Currently open entry?

History%Energy (#)

Type
float

Description
Energy of the system of a given entry.

Unit

hartree

History%EnergySubtype**Type**

string

Description

?

History%EngineEnergy (#)**Type**

float

Description

The energy as computed by the engine of a given entry.

Unit

hartree

History%EngineEnergyU (#)**Type**

float

Description

The uncertainty of the energy as computed by the engine of a given entry.

Unit

hartree

History%EngineGradients (#)**Type**

float_array

Description

The gradients as computed by the engine of a given entry.

Unit

hartree/bohr

Shape

[3, :]

History%EngineGradientsNormU (#)**Type**

float_array

Description

The uncertainty of the norm of the gradients (using the variance formula) as computed by the engine of a given entry.

Unit

hartree/bohr

Shape

[3, :]

History%EngineGradientsU (#)

Type

float_array

Description

The uncertainty of the gradients as computed by the engine of a given entry.

Unit

hartree/bohr

Shape

[3, :]

History%ExitConditionMsg**Type**

string

Description

Message from the exit condition on what condition was met.

History%Gradients (#)**Type**

float_array

Description

Nuclear gradients of a given entry

Unit

hartree/bohr

Shape

[3, :]

History%IRCDirection (#)**Type**

int

Description

IRC (Intrinsic Reaction Coordinate) direction of a given entry (1=forward, 2=backwards).

History%IRCGradMax (#)**Type**

float

Description

IRC (Intrinsic Reaction Coordinate) Max of the gradient determining the step in the inner loop.

Unit

hartree/bohr

History%IRCGradRms (#)**Type**

float

Description

IRC (Intrinsic Reaction Coordinate) Root mean square of the gradient determining the step in the inner loop.

Unit

hartree/bohr

History%IRCIteration (#)**Type**

int

Description

IRC (Intrinsic Reaction Coordinate) outer loop iteration number.

History%ItemName (#)**Type**

string

Description

Name of generic item on this section?

History%LatticeVectors (#)**Type**

float_array

Description

The lattice vectors of a given entry.

Unit

bohr

Shape

[3, :]

History%maxGrad (#)**Type**

float

Description

Maximum Cartesian component of the nuclear gradients of a given entry. This is generally used as a convergence criterion in geometry optimizations and similar tasks.

Unit

hartree/bohr

History%maxStep (#)**Type**

float

Description

Maximum difference in the Cartesian nuclear coordinates from the previous step. This is generally used as a convergence criterion in geometry optimizations and similar tasks.

Unit

bohr

History%MaxStressEnergyPerAtom (#)**Type**

float

Description

Maximum value of stress_tensor * cell_volume / number_of_atoms (for 2D and 1D systems, the cell_volume is replaced by the cell_area and cell_length respectively). This is the quantity used for determining whether a lattice optimization has converged.

History%MCMolecule (#)

Type
int

Description
Index of the randomly selected molecule/atom type to MCMove?

History%MCMove (#)

Type
int

Description
Index of monte carlo move of a given entry.

History%MCMoveType (#)

Type
string

Description
The type of monte carlo move. Can be one of the following: 'Insert', 'Delete', 'Displace', 'ChangeVolume'

History%Mols

Type
subsection

Description
Molecule detection info at various steps.

History%Mols%Atoms (#)

Type
int_array

Description
atoms(index(i):index(i+1)-1) = atom indices of molecule i

History%Mols%Index (#)

Type
int_array

Description
Size: Molecules%Num molecules. index(i) = index of the first atom of molecule i in array atoms(:)

History%Mols%Type (#)

Type
int_array

Description
Index indicating the type of the molecule ('Molecules%Molecule name #')

History%nEntries

Type
int

Description
Number of history entries.

History%nLatticeVectors (#)

Type
int

Description

The number of lattice vectors (i.e. the number of periodic boundary conditions) of a given entry.

History%OptIteration (#)

Type
int

Description

IRC (Intrinsic Reaction Coordinate) inner loop iteration number.

History%OrigStep (#)

Type
int

Description

Index of the original step for the Replay task.

History%PathLength (#)

Type
float

Description

IRC (Intrinsic Reaction Coordinate): sum of the arc lengths up to the latest point.

Unit
angstrom

History%rmsGrad (#)

Type
float

Description

Root mean square of the nuclear gradients of a given entry. This is generally used as a convergence criterion in geometry optimizations and similar tasks.

Unit
hartree/bohr

History%rmsStep (#)

Type
float

Description

Root mean square of the difference between the nuclear coordinates at this step and at the previous step. This is generally used as a convergence criterion in geometry optimizations and similar tasks.

History%Step (#)

Type
int

Description

The step number in a Molecular Dynamics calculation.

History%StressTensor (#)

Type

float_array

Description

The stress tensor of a given entry.

Unit

hartree/bohr^nLatticeVectors

Shape

[:, :]

History%SystemVersion (#)**Type**

int

Description

Index of the versioned-chemical system of a given frame.

IMDHORaman**Section content:** Resonance raman spectra using IMDHO**IMDHORaman%spectrum****Type**

int

Description

?

IMDHORaman%overtones**Type**

int_array

Description

?

IMDHORaman%ramanorder**Type**

int

Description

Maximum order of raman final state

IMDHORaman%spectrum**Type**

float_array

Description

?

InputMolecule**Section content:** The main input molecule of the calculation.**InputMolecule%AtomicNumbers****Type**

int_array

Description

Atomic number 'Z' of the atoms in the system

Shape

[nAtoms]

InputMolecule%AtomMasses**Type**

float_array

Description

Masses of the atoms

Unit

a.u.

Values range

[0, 'infinity']

Shape

[nAtoms]

InputMolecule%AtomSymbols**Type**

string

Description

The atom's symbols (e.g. 'C' for carbon)

Shape

[nAtoms]

InputMolecule%bondOrders**Type**

float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

InputMolecule%Charge**Type**

float

Description

Net charge of the system

Unit

e

InputMolecule%Coords**Type**

float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

Shape

[3, nAtoms]

InputMolecule%eeAttachTo**Type**

int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

InputMolecule%eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution.

InputMolecule%eeEField**Type**

float_array

Description

The external homogeneous electric field.

Unit

hartree/(e*bohr)

Shape

[3]

InputMolecule%eeLatticeVectors**Type**

float_array

Description

The lattice vectors used for the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNLatticeVectors]

InputMolecule%eeMulti**Type**

float_array

Description

The values of the external point- or multipole- charges.

Unit

a.u.

Shape

[eeNZIm, eeNMulti]

InputMolecule%eeNLatticeVectors

Type
int

Description

The number of lattice vectors for the external point- or multipole- charges.

InputMolecule%eeNMulti

Type
int

Description

The number of external point- or multipole- charges.

InputMolecule%eeNZlm

Type
int

Description

When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

InputMolecule%eeUseChargeBroadening

Type
bool

Description

Whether or not the external charges are point-like or broadened.

InputMolecule%eeXYZ

Type
float_array

Description

The position of the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNMulti]

InputMolecule%EngineAtomicInfo

Type
string_fixed_length

Description

Atom-wise info possibly used by the engine.

InputMolecule%fromAtoms

Type
int_array

Description

Index of the first atom in a bond. See the bondOrders array

InputMolecule%latticeDisplacements

Type

int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

InputMolecule%LatticeVectors**Type**

float_array

Description

Lattice vectors

Unit

bohr

Shape

[3, nLatticeVectors]

InputMolecule%nAtoms**Type**

int

Description

The number of atoms in the system

InputMolecule%nAtomsTypes**Type**

int

Description

The number different of atoms types

InputMolecule%nLatticeVectors**Type**

int

Description

Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values

[0, 1, 2, 3]

InputMolecule%toAtoms**Type**

int_array

Description

Index of the second atom in a bond. See the bondOrders array

InputMolecule(#)

Section content: Any auxiliary, named input molecules of the calculation.

InputMolecule (#) %AtomicNumbers**Type**

int_array

Description

Atomic number 'Z' of the atoms in the system

Shape

[nAtoms]

InputMolecule (#) %AtomMasses**Type**

float_array

Description

Masses of the atoms

Unit

a.u.

Values range

[0, 'infinity']

Shape

[nAtoms]

InputMolecule (#) %AtomSymbols**Type**

string

Description

The atom's symbols (e.g. 'C' for carbon)

Shape

[nAtoms]

InputMolecule (#) %bondOrders**Type**

float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

InputMolecule (#) %Charge**Type**

float

Description

Net charge of the system

Unit

e

InputMolecule (#) %Coords**Type**

float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

Shape

[3, nAtoms]

InputMolecule (#) %eeAttachTo**Type**

int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

InputMolecule (#) %eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution.

InputMolecule (#) %eeEField**Type**

float_array

Description

The external homogeneous electric field.

Unit

hartree/(e*bohr)

Shape

[3]

InputMolecule (#) %eeLatticeVectors**Type**

float_array

Description

The lattice vectors used for the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNLatticeVectors]

InputMolecule (#) %eeMulti**Type**

float_array

Description

The values of the external point- or multipole- charges.

Unit

a.u.

Shape

[eeNZIm, eeNMulti]

InputMolecule (#) %eeNLatticeVectors**Type**

int

Description

The number of lattice vectors for the external point- or multipole- charges.

InputMolecule (#) %eeNMulti**Type**

int

Description

The number of external point- or multipole- charges.

InputMolecule (#) %eeNZlm**Type**

int

Description

When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

InputMolecule (#) %eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened.

InputMolecule (#) %eeXYZ**Type**

float_array

Description

The position of the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNMulti]

InputMolecule (#) %EngineAtomicInfo**Type**

string_fixed_length

Description

Atom-wise info possibly used by the engine.

InputMolecule (#) %fromAtoms**Type**

int_array

Description

Index of the first atom in a bond. See the bondOrders array

InputMolecule (#) %latticeDisplacements

Type

int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

InputMolecule (#) %LatticeVectors**Type**

float_array

Description

Lattice vectors

Unit

bohr

Shape

[3, nLatticeVectors]

InputMolecule (#) %nAtoms**Type**

int

Description

The number of atoms in the system

InputMolecule (#) %nAtomsTypes**Type**

int

Description

The number different of atoms types

InputMolecule (#) %nLatticeVectors**Type**

int

Description

Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values

[0, 1, 2, 3]

InputMolecule (#) %toAtoms**Type**

int_array

Description

Index of the second atom in a bond. See the bondOrders array

InputMolecules

Section content: Section to store additional information about the auxiliary, named input molecules.

InputMolecules%Name (#)**Type**

string

Description

The name that was given to the molecule stored in the corresponding InputMolecule(*) section.
The name comes from the System block header in the AMS input file.

InputMolecules%numNamedMolecules**Type**

int

Description

The number of auxiliary, named input molecules.

IRC

Section content: Data regarding the IRC calculation.

IRC%AtomicNumbers**Type**

int_array

Description

Atomic number 'Z' of the atoms in the system

Shape

[nAtoms]

IRC%AtomMasses**Type**

float_array

Description

Masses of the atoms

Unit

a.u.

Values range

[0, 'infinity']

Shape

[nAtoms]

IRC%AtomSymbols**Type**

string

Description

The atom's symbols (e.g. 'C' for carbon)

Shape

[nAtoms]

IRC%barriers**Type**

float_array

Description

TS barrier energies (forward and backward?)

Unit

hartree

Shape
[2]

IRC%bondOrders

Type
float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

IRC%cen

Type
subsection

Description
?

IRC%cen%addedAtoms

Type
archived_int_array

Description
Atom species indices

IRC%cen%AtomicInfo

Type
archived_string_array

Description

IRC%cen%BondOrders

Type
archived_float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

IRC%cen%Charge

Type
float

Description
Net charge of the system

Unit
e

IRC%cen%eeAttachTo

Type
archived_int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

IRC%cen%eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution

IRC%cen%eeEField**Type**

float_array

Description

The external homogeneous electric field

Unit

hartree/(e*bohr)

Shape

[3]

IRC%cen%eeLatticeVectors**Type**

archived_float_array

Description

The lattice vectors used for the external point- or multipole- charges

Unit

bohr

IRC%cen%eeMultipoles**Type**

archived_float_array

Description

The multiple charges.

Unit

bohr

IRC%cen%eenMulti**Type**

int

Description

The number of multipoles.

IRC%cen%eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened

IRC%cen%eeXYZ

Type

archived_float_array

Description

The position of the external point- or multipole- charges

Unit

bohr

IRC%cen%FromAtoms**Type**

archived_int_array

Description

Index of the first atom in a bond. See the bondOrders array

IRC%cen%hasLatticeDisplacements**Type**

bool

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

IRC%cen%kVectors**Type**

float_array

Description

Reciprocal lattice vectors (corresponding to the strained lattice vectors)

Unit

1/bohr

Shape

[3, 3]

IRC%cen%LatticeDisplacements**Type**

archived_int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

IRC%cen%nAtoms**Type**

int

Description

The number of atoms in the system

IRC%cen%nSpecies**Type**

int

Description

The number different of atoms types

IRC%cen%nVectors**Type**

int

Description

Number of lattice vectors (0:molecule, 1:chain, 2:slab, 3:bulk).

IRC%cen%removedAtoms**Type**

archived_int_array

Description

Atom species indices

IRC%cen%sp#inputSymbol**Type**

string

Description

Symbol as specified in the input

IRC%cen%sp#mass**Type**

float

Description

mass

IRC%cen%sp#regions**Type**

archived_string_array

Description

regions

IRC%cen%sp#symbol**Type**

string

Description

Element symbol

IRC%cen%sp#Z**Type**

int

Description

Atomic number

IRC%cen%SpIndices**Type**

archived_int_array

Description

Atom species indices

IRC%cen%SPlen

Type

int

Description

Number of species

IRC%cen%strain**Type**

float_array

Description

The strain matrix.

Shape

[3, 3]

IRC%cen%strainedVectors**Type**

float_array

Description

Strained real space lattice vectors

Unit

bohr

Shape

[3, 3]

IRC%cen%ToAtoms**Type**

archived_int_array

Description

Index of the second atom in a bond. See the bondOrders array

IRC%cen%unstrainedVectors**Type**

float_array

Description

Real space lattice vectors (unstrained).

Unit

bohr

Shape

[3, 3]

IRC%cen%version**Type**

int

Description**IRC%cen%xyzAtoms****Type**

archived_float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

IRC%Charge**Type**

float

Description

Net charge of the system

Unit

e

IRC%conf**Type**

subsection

Description

Configuration data for the IRC procedure.

IRC%conf%ConvGrad**Type**

float

Description

Convergence criterion for gradient

Unit

hartree/bohr

IRC%conf%ConvStep**Type**

float

Description

Convergence criterion for step in optim coords

Unit

bohr

IRC%conf%CoordType**Type**

int

Description

0, 1 or 2, see IRC_OPTIM_COORDS

IRC%conf%Directions**Type**

int

Description

one of IRC_DIRECTION_* constants

IRC%conf%HessFile

Type

string

Description

File to get the Hessian from if hessianType=='fromfile'

IRC%conf%HesType**Type**

string

Description

The hessian type used in the IRC calculation: 'calculate', 'fromfile' or 'restart'

IRC%conf%IrcStep**Type**

float

Description

Step size

Unit

bohr

IRC%conf%keepResult**Type**

bool

Description

Keep rkf files from single point calculations for each converged path point

IRC%conf%MaxIRCSteps**Type**

int

Description

Max number of IRC points before switching to energy minimization

IRC%conf%MaxIter**Type**

int

Description

Max num steps in each geometry optimization (inner IRC loop)

IRC%conf%MaxPoints**Type**

int

Description

Max number of IRC points before switching to the next direction

IRC%conf%MinEnProf**Type**

bool

Description

! Minimum energy profile (i.e. no mass-weighting) instead of IRC?

IRC%Coords

Type

float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

Shape

[3, nAtoms]

IRC%curIRCStep**Type**

float

Description

ircConfig%ircStep, possibly reduced for a curved path

IRC%direction**Type**

int

Description

Current direction: 1 - forward, 2 - backward

IRC%directionDone**Type**

bool_array

Description

Flag to see which direction has been already done (for restart)

Shape

[2]

IRC%eeAttachTo**Type**

int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

IRC%eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution.

IRC%eeEField**Type**

float_array

Description

The external homogeneous electric field.

Unit

hartree/(e*bohr)

Shape

[3]

IRC%eeLatticeVectors**Type**

float_array

Description

The lattice vectors used for the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNLatticeVectors]

IRC%eeMulti**Type**

float_array

Description

The values of the external point- or multipole- charges.

Unit

a.u.

Shape

[eeNZlm, eeNMulti]

IRC%eeNLatticeVectors**Type**

int

Description

The number of lattice vectors for the external point- or multipole- charges.

IRC%eeNMulti**Type**

int

Description

The number of external point- or multipole- charges.

IRC%eeNZlm**Type**

int

Description

When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

IRC%eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened.

IRC%eeXYZ**Type**

float_array

Description

The position of the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNMulti]

IRC%EngineAtomicInfo**Type**

string_fixed_length

Description

Atom-wise info possibly used by the engine.

IRC%fromAtoms**Type**

int_array

Description

Index of the first atom in a bond. See the bondOrders array

IRC%gradCart**Type**

archived_float_array

Description

Cartesian gradients

Unit

hartree/bohr

IRC%hessCart**Type**

archived_float_array

Description

Current Hessian in Cartesian coords

Unit

hartree/bohr²

IRC%hessInit**Type**

archived_float_array

Description

Initial Hessian in Cartesian coords

Unit

hartree/bohr²

IRC%histEnergy**Type**

archived_float_array

Description

Energy history

Unit

hartree

IRC%histGradRms**Type**

archived_float_array

Description

Gradients RMS history

Unit

hartree/bohr

IRC%histPathLength**Type**

archived_float_array

Description

Path length history

IRC%histStatus**Type**

archived_int_array

Description

Status history

IRC%histXYZ**Type**

archived_float_array

Description

XYZ history

IRC%initialEnergy**Type**

float

Description

TS energy

Unit

hartree

IRC%initialGradRms**Type**

float

Description

TS RMS gradient

Unit

hartree/bohr

IRC%ircIteration**Type**

int

Description

outer loop iteration number

IRC%latticeDisplacements**Type**

int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

IRC%LatticeVectors**Type**

float_array

Description

Lattice vectors

Unit

bohr

Shape

[3, nLatticeVectors]

IRC%nAtoms**Type**

int

Description

The number of atoms in the system

IRC%nAtomsTypes**Type**

int

Description

The number different of atoms types

IRC%nLatticeVectors**Type**

int

Description

Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values

[0, 1, 2, 3]

IRC%optIteration**Type**

int

Description

inner loop iteration number

IRC%pathLength**Type**

float

Description

Sum of the arc lengths up to the latest point

IRC%sys**Type**

subsection

Description

?

IRC%sys%addedAtoms**Type**

archived_int_array

Description

Atom species indices

IRC%sys%AtomicInfo**Type**

archived_string_array

Description**IRC%sys%BondOrders****Type**

archived_float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

IRC%sys%Charge**Type**

float

Description

Net charge of the system

Unit

e

IRC%sys%eeAttachTo**Type**

archived_int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

IRC%sys%eeChargeWidth

Type

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution

IRC%sys%eeEField**Type**

float_array

Description

The external homogeneous electric field

Unit

hartree/(e*bohr)

Shape

[3]

IRC%sys%eeLatticeVectors**Type**

archived_float_array

Description

The lattice vectors used for the external point- or multipole- charges

Unit

bohr

IRC%sys%eeMultipoles**Type**

archived_float_array

Description

The multiple charges.

Unit

bohr

IRC%sys%eenMulti**Type**

int

Description

The number of multipoles.

IRC%sys%eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened

IRC%sys%eeXYZ**Type**

archived_float_array

Description

The position of the external point- or multipole- charges

Unit

bohr

IRC%sys%FromAtoms**Type**

archived_int_array

Description

Index of the first atom in a bond. See the bondOrders array

IRC%sys%hasLatticeDisplacements**Type**

bool

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

IRC%sys%kVectors**Type**

float_array

Description

Reciprocal lattice vectors (corresponding to the strained lattice vectors)

Unit

1/bohr

Shape

[3, 3]

IRC%sys%LatticeDisplacements**Type**

archived_int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

IRC%sys%nAtoms**Type**

int

Description

The number of atoms in the system

IRC%sys%nSpecies**Type**

int

Description

The number different of atoms types

IRC%sys%nVectors

Type
int

Description
Number of lattice vectors (0:molecule, 1:chain, 2:slab, 3:bulk).

IRC%sys%removedAtoms

Type
archived_int_array

Description
Atom species indices

IRC%sys%sp#inputSymbol

Type
string

Description
Symbol as specified in the input

IRC%sys%sp#mass

Type
float

Description
mass

IRC%sys%sp#regions

Type
archived_string_array

Description
regions

IRC%sys%sp#symbol

Type
string

Description
Element symbol

IRC%sys%sp#Z

Type
int

Description
Atomic number

IRC%sys%SpIndices

Type
archived_int_array

Description
Atom species indices

IRC%sys%SPlen

Type
int

Description

Number of species

IRC%sys%strain**Type**

float_array

Description

The strain matrix.

Shape

[3, 3]

IRC%sys%strainedVectors**Type**

float_array

Description

Strained real space lattice vectors

Unit

bohr

Shape

[3, 3]

IRC%sys%ToAtoms**Type**

archived_int_array

Description

Index of the second atom in a bond. See the bondOrders array

IRC%sys%unstrainedVectors**Type**

float_array

Description

Real space lattice vectors (unstrained).

Unit

bohr

Shape

[3, 3]

IRC%sys%version**Type**

int

Description**IRC%sys%xyzAtoms****Type**

archived_float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

IRC%toAtoms**Type**

int_array

Description

Index of the second atom in a bond. See the bondOrders array

KFDefinitions**Section content:** The definitions of the data on this file**KFDefinitions%json****Type**

string

Description

The definitions of the data on this file in json.

MDHistory**Section content:** History of a Molecular dynamics simulation.**MDHistory%Area (#)****Type**

float_array

Description

The area of the cell (only for 2D PBC). This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

bohr^2

MDHistory%BerBstatEnergy (#)**Type**

float_array

Description

?

Unit

hartree

MDHistory%BerTstat#Energy (#)**Type**

float_array

Description

?

Unit

hartree

MDHistory%BerTstat#Temperature (#)**Type**

float_array

Description

Local temperature of the thermostat region

Unit

kelvin

MDHistory%BiasEnergy (#)**Type**

float_array

Description

? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

MDHistory%blockSize**Type**

int

Description

Explain the block-system... ?

MDHistory%BondBoostEnergy (#)**Type**

float_array

Description

BondBoost energy, part of the potential energy

Unit

hartree

MDHistory%BoostFactor (#)**Type**

float_array

Description

The boost factor for hyper-dynamics. Expand? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

MDHistory%Charges (#)**Type**

float_array

Description

Net atomic charges as computed by the engine (for example, the Charges for a water molecule might be [-0.6, 0.3, 0.3]).

Unit

e

MDHistory%ConservedEnergy (#)**Type**

float_array

Description

The conserved energy...? Some MD person, please fix. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

MDHistory%CosineShearVelocity (#)**Type**

float_array

Description

?

MDHistory%currentEntryOpen**Type**

bool

Description

?

MDHistory%Density (#)**Type**

float_array

Description

The density of the system (mass/simulation_cell_volume). Only for 3D PBC. This is a ‘blocked’ property. See the ‘blockSize’ and ‘nBlocks’ variables for more details.

Unitdalton/bohr³**MDHistory%Eng*FrcRgn* (#)****Type**

float_array

Description

?

MDHistory%Engine*ForceRegion* (#)**Type**

float_array

Description

?

MDHistory%Hypertime (#)**Type**

float_array

Description

Hyper time for hyper-dynamics. Expand? This is a ‘blocked’ property. See the ‘blockSize’ and ‘nBlocks’ variables for more details.

Unit

femtosecond

MDHistory%ItemName (#)**Type**

string

Description

?

MDHistory%KineticEnergy (#)**Type**

float_array

Description

The kinetic energy of the system. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

MDHistory%Length (#)**Type**

float_array

Description

The length of the cell (only for 1D PBC). This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

bohr

MDHistory%MaxBiasEnergy (#)**Type**

float_array

Description

? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

MDHistory%MaxBoostFactor (#)**Type**

float_array

Description

The maximum boost factor for hyper-dynamics. Expand? This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

MDHistory%MeanCosineShearVelocity (#)**Type**

float_array

Description

?

MDHistory%MeanEng*FrcRgn* (#)**Type**

float_array

Description

?

MDHistory%MeanEngine*ForceRegion* (#)

Type
float_array

Description
?

MDHistory%MovingRestr*Energy (#)

Type
float_array

Description
Moving restraints energy, part of the potential energy

Unit
hartree

MDHistory%nBlocks

Type
int

Description
Explain the block-system... ?

MDHistory%nEntries

Type
int

Description
Number of MD history entries.

MDHistory%NHCTstat#Energy (#)

Type
float_array

Description
?

Unit
hartree

MDHistory%NHCTstat#Temperature (#)

Type
float_array

Description
Local temperature of the thermostat region

Unit
kelvin

MDHistory%NHTBstat#Energy (#)

Type
float_array

Description
?

Unit
hartree

MDHistory%NHTBstat#Temperature (#)**Type**

float_array

Description

Local temperature of the thermostat region

Unit

kelvin

MDHistory%Number of molecules (#)**Type**

int_array

Description

Number of molecules. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

MDHistory%PotentialEnergy (#)**Type**

float_array

Description

The potential energy, i.e. the energy as computed by the engine. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

hartree

MDHistory%Pressure (#)**Type**

float_array

Description

The pressure of the system (only for 3D PBC). This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unithartree/bohr³**MDHistory%PressureTensor (#)****Type**

float_array

Description

Pressure tensor in Voigt notation.

Unithartree/bohr³**MDHistory%ReactionBoostEnergy (#)****Type**

float_array

Description

ReactionBoost energy, part of the potential energy

Unit

hartree

MDHistory%ReactorEnergy (#)**Type**

float_array

Description

Reactor energy, part of the potential energy

Unit

hartree

MDHistory%StdevCosineShearVelocity (#)**Type**

float_array

Description

?

MDHistory%StdevEng*FrcRgn* (#)**Type**

float_array

Description

?

MDHistory%StdevEngine*ForceRegion* (#)**Type**

float_array

Description

?

MDHistory%Step (#)**Type**

int_array

Description

The step number of the MD calculation. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

MDHistory%Temperature (#)**Type**

float_array

Description

The temperature of the system, computed from the kinetic energy?. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

kelvin

MDHistory%TempProfile_a (#)**Type**

float_array

Description

The temperature profile...? Some MD person, please fix.

Unit

kelvin

MDHistory%TempProfile_b (#)**Type**

float_array

Description

The temperature profile...? Some MD person, please fix.

Unit

kelvin

MDHistory%TempProfile_c (#)**Type**

float_array

Description

The temperature profile...? Some MD person, please fix.

Unit

kelvin

MDHistory%Time (#)**Type**

float_array

Description

The MD simulation time. This is a ‘blocked’ property. See the ‘blockSize’ and ‘nBlocks’ variables for more details.

Unit

femtosecond

MDHistory%TotalEnergy (#)**Type**

float_array

Description

The total energy of the system: potential energy (as computed by the engine) + kinetic energy. This is a ‘blocked’ property. See the ‘blockSize’ and ‘nBlocks’ variables for more details.

Unit

hartree

MDHistory%Velocities (#)**Type**

float_array

Description

The velocity of the atoms.

Unit

bohr/femtosecond

Shape

[3, :]

MDHistory%Volume (#)

Type

float_array

Description

The volume of the cell (only for 3D PBC). This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

Unit

bohr^3

MDHookState

Section content: Data related to MD hooks

MDResults

Section content: Results of an MD calculation.

MDResults%EndStep**Type**

int

Description

Step number of the last step.

MDResults%EndTime [fs]**Type**

float

Description

The time at the end of the MD simulation.

Unit

femtosecond

MDResults%EndVelocities**Type**

float_array

Description

The atomic velocities at the last step of the MD simulation.

Unit

a.u.

Shape

[3, :]

MDResults%MaxConservedEnergy**Type**

float

Description

Maximum value of the conserved energy during the MD simulation.

Unit

hartree

MDResults%MaxKineticEnergy**Type**

float

Description

Maximum value of the kinetic energy during the MD simulation.

Unit

hartree

MDResults%MaxPotentialEnergy**Type**

float

Description

Maximum value of the potential energy during the MD simulation.

Unit

hartree

MDResults%MaxTemperature**Type**

float

Description

Maximum value of the temperature during the MD simulation.

Unit

kelvin

MDResults%MaxTotalEnergy**Type**

float

Description

Maximum value of the total energy during the MD simulation.

Unit

hartree

MDResults%MeanConservedEnergy**Type**

float

Description

Mean conserved energy during the MD simulation.

Unit

hartree

MDResults%MeanKineticEnergy**Type**

float

Description

Mean kinetic energy during the MD simulation.

Unit

hartree

MDResults%MeanPotentialEnergy**Type**

float

Description

Mean potential energy during the MD simulation. The potential energy is the energy computed by the engine.

Unit

hartree

MDResults%MeanTemperature**Type**

float

Description

Mean temperature during the MD simulation.

Unit

kelvin

MDResults%MeanTotalEnergy**Type**

float

Description

Mean total energy during the MD simulation (total energy = potential energy from engine + kinetic energy).

Unit

hartree

MDResults%MinConservedEnergy**Type**

float

Description

Minimum value of the conserved energy during the MD simulation.

Unit

hartree

MDResults%MinKineticEnergy**Type**

float

Description

Minimum value of the kinetic energy during the MD simulation.

Unit

hartree

MDResults%MinPotentialEnergy**Type**

float

Description

Minimum value of the potential energy during the MD simulation.

Unit

hartree

MDResults%MinTemperature

Type

float

Description

Minimum value of the temperature during the MD simulation.

Unit

kelvin

MDResults%MinTotalEnergy**Type**

float

Description

Minimum value of the total energy during the MD simulation.

Unit

hartree

MDResults%StartStep**Type**

int

Description

Step number of the first step.

MDResults%StartTime [fs]**Type**

float

Description

The time at the beginning of the MD simulation.

Unit

femtosecond

MDResults%StdDevConservedEnergy**Type**

float

Description

Standard deviation of the conserved energy during the MD simulation.

Unit

hartree

MDResults%StdDevKineticEnergy**Type**

float

Description

Standard deviation of the kinetic energy during the MD simulation.

Unit

hartree

MDResults%StdDevPotentialEnergy**Type**

float

Description

Standard deviation of the potential energy during the MD simulation.

Unit

hartree

MDResults%StdDevTemperature**Type**

float

Description

Standard deviation of the temperature during the MD simulation.

Unit

kelvin

MDResults%StdDevTotalEnergy**Type**

float

Description

Standard deviation of the total energy during the MD simulation.

Unit

hartree

Molecule

Section content: The final molecule in the calculation. What is stored here depends on the task of the AMS calculation. For example, for a GeometryOptimization this will contain the optimized molecule, while for a TransitionStateSearch this will contain the molecule at the TS geometry.

Molecule%AtomicNumbers**Type**

int_array

Description

Atomic number 'Z' of the atoms in the system

Shape

[nAtoms]

Molecule%AtomMasses**Type**

float_array

Description

Masses of the atoms

Unit

a.u.

Values range

[0, 'infinity']

Shape

[nAtoms]

Molecule%AtomSymbols**Type**

string

Description

The atom's symbols (e.g. 'C' for carbon)

Shape

[nAtoms]

Molecule%bondOrders**Type**

float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

Molecule%Charge**Type**

float

Description

Net charge of the system

Unit

e

Molecule%Coords**Type**

float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

Shape

[3, nAtoms]

Molecule%eeAttachTo**Type**

int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

Molecule%eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution.

Molecule%eeEField**Type**

float_array

Description

The external homogeneous electric field.

Unit

hartree/(e*bohr)

Shape

[3]

Molecule%eeLatticeVectors**Type**

float_array

Description

The lattice vectors used for the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNLatticeVectors]

Molecule%eeMulti**Type**

float_array

Description

The values of the external point- or multipole- charges.

Unit

a.u.

Shape

[eeNZlm, eeNMulti]

Molecule%eeNLatticeVectors**Type**

int

Description

The number of lattice vectors for the external point- or multipole- charges.

Molecule%eeNMulti**Type**

int

Description

The number of external point- or multipole- charges.

Molecule%eeNZlm**Type**

int

Description

When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

Molecule%eeUseChargeBroadening

Type

bool

Description

Whether or not the external charges are point-like or broadened.

Molecule%eeXYZ**Type**

float_array

Description

The position of the external point- or multipole- charges.

Unit

bohr

Shape

[3, eeNMulti]

Molecule%EngineAtomicInfo**Type**

string_fixed_length

Description

Atom-wise info possibly used by the engine.

Molecule%fromAtoms**Type**

int_array

Description

Index of the first atom in a bond. See the bondOrders array

Molecule%latticeDisplacements**Type**

int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

Molecule%LatticeVectors**Type**

float_array

Description

Lattice vectors

Unit

bohr

Shape

[3, nLatticeVectors]

Molecule%nAtoms**Type**

int

Description

The number of atoms in the system

Molecule%nAtomsTypes**Type**

int

Description

The number different of atoms types

Molecule%nLatticeVectors**Type**

int

Description

Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values

[0, 1, 2, 3]

Molecule%toAtoms**Type**

int_array

Description

Index of the second atom in a bond. See the bondOrders array

Molecules

Section content: Results of the molecules detection algorithms. Note: molecules with the same formula are considered to be the same molecule.

Molecules%Molecule name #**Type**

string

Description

The name of the molecules, i.e. their molecular formula.

Molecules%Num molecules**Type**

int

Description

Number distinct (i.e. with different molecular formula) molecules found.

NEB

Section content: Data related to the Nudge Elastic Band procedure.

NEB%climbing**Type**

bool

Description

Climb the highest image to the TS

NEB%climbingThresh**Type**

float

Description

Threshold on the max perpendicular force component for the climbing image

NEB%doubleNudge**Type**

string

Description

Henkelman: smoothly turns off the double nudging as the NEB converges; Trygubenko: JCP 120, 2082 (2005)

NEB%energy**Type**

float_array

Description

energy per image, including ends

Unit

hartree

NEB%fixed**Type**

bool_array

Description

Flag fixed atoms

NEB%forces**Type**

float_array

Description

forces per image, including stress forces (1st nVectors 'atoms' of each image)

Unit

hartree/bohr

NEB%highestIndex**Type**

int

Description

Index of the highest (or climbing) image

NEB%historyIndex**Type**

int_array

Description

(i,j) element contains history index of image i on iteration j

NEB%im#**Type**

subsection

Description

The molecule info of image #

NEB%im#%addedAtoms**Type**

archived_int_array

Description

Atom species indices

NEB%im#%AtomicInfo**Type**

archived_string_array

Description**NEB%im#%BondOrders****Type**

archived_float_array

Description

The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

NEB%im#%Charge**Type**

float

Description

Net charge of the system

Unit

e

NEB%im#%eeAttachTo**Type**

archived_int_array

Description

A multipole may be attached to an atom. This influences the energy gradient.

NEB%im#%eeChargeWidth**Type**

float

Description

If charge broadening was used for external charges, this represents the width of the charge distribution

NEB%im#%eeEField**Type**

float_array

Description

The external homogeneous electric field

Unit

hartree/(e*bohr)

Shape

[3]

NEB%im%%eeLatticeVectors**Type**

archived_float_array

Description

The lattice vectors used for the external point- or multipole- charges

Unit

bohr

NEB%im%%eeMultipoles**Type**

archived_float_array

Description

The multiple charges.

Unit

bohr

NEB%im%%eenMulti**Type**

int

Description

The number of multipoles.

NEB%im%%eeUseChargeBroadening**Type**

bool

Description

Whether or not the external charges are point-like or broadened

NEB%im%%eeXYZ**Type**

archived_float_array

Description

The position of the external point- or multipole- charges

Unit

bohr

NEB%im%%FromAtoms**Type**

archived_int_array

Description

Index of the first atom in a bond. See the bondOrders array

NEB%im%%hasLatticeDisplacements**Type**

bool

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

NEB%im##kVectors**Type**

float_array

Description

Reciprocal lattice vectors (corresponding to the strained lattice vectors)

Unit

1/bohr

Shape

[3, 3]

NEB%im##LatticeDisplacements**Type**

archived_int_array

Description

The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

NEB%im##nAtoms**Type**

int

Description

The number of atoms in the system

NEB%im##nSpecies**Type**

int

Description

The number different of atoms types

NEB%im##nVectors**Type**

int

Description

Number of lattice vectors (0:molecule, 1:chain, 2:slab, 3:bulk).

NEB%im##removedAtoms**Type**

archived_int_array

Description

Atom species indices

NEB%im##sp#inputSymbol**Type**

string

Description

Symbol as specified in the input

NEB%im#%sp#mass**Type**

float

Description

mass

NEB%im#%sp#regions**Type**

archived_string_array

Description

regions

NEB%im#%sp#symbol**Type**

string

Description

Element symbol

NEB%im#%sp#Z**Type**

int

Description

Atomic number

NEB%im#%SpIndices**Type**

archived_int_array

Description

Atom species indices

NEB%im#%SPlen**Type**

int

Description

Number of species

NEB%im#%strain**Type**

float_array

Description

The strain matrix.

Shape

[3, 3]

NEB%im#%strainedVectors**Type**

float_array

Description

Strained real space lattice vectors

Unit

bohr

Shape

[3, 3]

NEB%im##ToAtoms**Type**

archived_int_array

Description

Index of the second atom in a bond. See the bondOrders array

NEB%im##unstrainedVectors**Type**

float_array

Description

Real space lattice vectors (unstrained).

Unit

bohr

Shape

[3, 3]

NEB%im##version**Type**

int

Description**NEB%im##xyzAtoms****Type**

archived_float_array

Description

Coordinates of the nuclei (x,y,z)

Unit

bohr

NEB%interIntCoords**Type**

bool

Description

Interpolate in the internal coords instead of Cartesian

NEB%interpolateOption**Type**

int

Description

1=only dist, 2=1+linear angles, 3=1+val. angles, 4=1+dihedrals, 99=all coordinates

NEB%interShortest

Type

bool

Description

Interpolate across cell boundary if necessary

NEB%jacobian**Type**

float

Description

Scaling factor used to convert the lattice strain to a NEB coordinate value

NEB%LeftBarrier**Type**

float

Description

Left barrier energy

Unit

hartree

NEB%mapToOriginalCell**Type**

bool

Description

Map atoms to the [-0.5,0.5] cell

NEB%nebImages**Type**

int

Description

number of intermediate NEB images, without ends. With them, there will be nebImages+2 images

NEB%nebIterations**Type**

int

Description

Max number of iterations

NEB%nParallel**Type**

int

Description

Number of images to do in parallel

NEB%oldTangent**Type**

bool

Description

use old tangent

NEB%optimizeEnds**Type**

bool

Description

Start NEB with optimization of the products/reactants

NEB%optimizeLattice**Type**

bool

Description

Optimize lattice vectors

NEB%ReactionEnergy**Type**

float

Description

Reaction energy

Unit

hartree

NEB%RightBarrier**Type**

float

Description

Left barrier energy

Unit

hartree

NEB%spring**Type**

float

Description

spring force constant

Unit

a.u.

NEB%stressTensors**Type**

float_array

Description

stress tensor per image

NEB%success**Type**

int_array

Description

Single-point success flag (0 or 1)

PESScan

Section content: Data related to the Potential Energy Surface (PES) Scan procedure.

PESScan%GOConverged**Type**

bool_array

Description

Whether the (constrained) optimization at the various PES scan points converged.

Shape

[nPoints]

PESScan%HistoryIndices**Type**

int_array

Description

The indices of the frames in the History section corresponding to the PES point values.

Shape

[nPoints]

PESScan%HistoryPESPoints**Type**

int_array

Description

?

PESScan%nPoints**Type**

int

Description

The total number of scanned PES points. This is the product of all nPoints(#) values.

PESScan%nPoints (#)**Type**

int

Description

Number of points along the corresponding scan coordinate.

PESScan%nScanCoord**Type**

int

Description

Number of (independent) coordinates along which the PES scan is performed.

PESScan%PES**Type**

float_array

Description

The total energy at each particular PES point.

Unit

hartree

Shape

[nPoints]

PESScan%PESCoords**Type**

float_array

Description

The values of all coordinates for each particular PES point.

Shape

[:, nPoints]

PESScan%RangeEnd (#)**Type**

float_array

Description

The final value(s) for the corresponding scan coordinate.

PESScan%RangeStart (#)**Type**

float_array

Description

The starting value(s) for the corresponding scan coordinate.

PESScan%ScanCoord (#)**Type**

string_fixed_length

Description

A human readable description of the scan coordinate.

Replay**Section content:** Output of the Replay task of the AMS driver.**Replay%File****Type**

string

Description

The path to the file from which the trajectory was replayed.

Replay%Frames**Type**

int_array

Description

The indices of the frames in the history section that were replayed.

Replay%Type**Type**

string

Description

The type of job that was replayed, e.g. PESScan, NEB, IRC or Generic.

ReplicaExchangeHistory

Section content: ?

ReplicaExchangeHistory%AvgSwapProbability #-#(##)

Type

float_array

Description

?

ReplicaExchangeHistory%blockSize

Type

int

Description

Explain the block-system... ?

ReplicaExchangeHistory%currentEntryOpen

Type

bool

Description

?

ReplicaExchangeHistory%EnsembleOfSystem #(##)

Type

int_array

Description

?

ReplicaExchangeHistory%ItemName (##)

Type

string

Description

?

ReplicaExchangeHistory%MaxPotentialEnergy #(##)

Type

float_array

Description

?

ReplicaExchangeHistory%MDHistoryFrame (##)

Type

int_array

Description

?

ReplicaExchangeHistory%MeanPotentialEnergy #(##)

Type

float_array

Description

?

ReplicaExchangeHistory%MinPotentialEnergy #(#)**Type**

float_array

Description

?

ReplicaExchangeHistory%nBlocks**Type**

int

Description

Explain the block-system... ?

ReplicaExchangeHistory%nEntries**Type**

int

Description

Number of MD history entries.

ReplicaExchangeHistory%StdDevPotentialEnergy #(#)**Type**

float_array

Description

?

ReplicaExchangeHistory%Step (#)**Type**

int_array

Description

The step number of the MD calculation. This is a 'blocked' property. See the 'blockSize' and 'nBlocks' variables for more details.

ReplicaExchangeHistory%SystemInEnsemble #(#)**Type**

int_array

Description

?

ReplicaExchangeHistory%TemperatureOfSystem #(#)**Type**

float_array

Description

?

SystemVersionHistory**Section content:** ?**SystemVersionHistory%AddedAtoms (#)**

Type
int_array

Description
?

SystemVersionHistory%currentEntryOpen

Type
bool

Description
?

SystemVersionHistory%ItemName (#)

Type
string

Description
?

SystemVersionHistory%nEntries

Type
int

Description
?

SystemVersionHistory%RemovedAtoms (#)

Type
int_array

Description
?

SystemVersionHistory%SectionNum (#)

Type
int

Description
?

Timers

Section content: Information on the timers, as invoked by Print Timers=Normal.

Timers%Elapsed

Type
float_array

Description
Elapsed time for the master process

Shape
[NumTimers]

Timers%ElapsedAllTasks

Type
float_array

Description

?

Shape

[NumTasks, NumTimers]

Timers%MinMaxAvElapsed**Type**

float_array

Description

The minimum, maximum, and average elapsed time

Shape

[3, NumTimers]

Timers%MinMaxAvSystem**Type**

float_array

Description

The minimum, maximum, and average system time

Shape

[3, NumTimers]

Timers%MinMaxAvUser**Type**

float_array

Description

The minimum, maximum, and average user time

Shape

[3, NumTimers]

Timers%Names**Type**

string

Description

Names of the timed sections

Shape

[NumTimers]

Timers%NumCalls**Type**

int_array

Description

How often has a section been called for the master process

Shape

[NumTimers]

Timers%NumTasks**Type**

int

Description

The number of processes used to do the calculation (NSCM)

Timers%NumTimers**Type**

int

Description

The number of timed sections

Timers%PageFaultsAllTasks**Type**

int_array

Description

Page faults for all tasks

Shape

[NumTasks, NumTimers]

Timers%System**Type**

float_array

Description

System time on the master process

Shape

[NumTimers]

Timers%SystemAllTasks**Type**

float_array

Description

The system time for all tasks

Shape

[NumTasks, NumTimers]

Timers%User**Type**

float_array

Description

User time for the master process, what you see in the first table of the normal output

Shape

[NumTimers]

Timers%UserAllTasks**Type**

float_array

Description

The user time for all tasks

Shape

[NumTasks, NumTimers]

Vibrations

Section content: Data concerning the vibrational modes/spectra of the system.

Vibrations%ExcitedStateLifetime

Type

float

Description

Raman excited state lifetime.

Unit

hartree

Vibrations%ForceConstants

Type

float_array

Description

The force constants of the vibrations.

Unit

hartree/bohr²

Shape

[nNormalModes]

Vibrations%FreqBeforeScan [cm-1]

Type

float_array

Description

?

Shape

[nNormalModes]

Vibrations%Frequencies [cm-1]

Type

float_array

Description

The vibrational frequencies of the normal modes.

Unit

cm⁻¹

Shape

[nNormalModes]

Vibrations%Intensities [km/mol]

Type

float_array

Description

The intensity of the normal modes.

Unit

km/mol

Shape

[nNormalModes]

Vibrations%IRBeforeScan [km/mol]**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%IrReps**Type**

lchar_string_array

Description

Symmetry symbol of the normal mode.

Shape

[nNormalModes]

Vibrations%IrRepsBeforeScan**Type**

lchar_string_array

Description

?

Shape

[nNormalModes]

Vibrations%ModesNorm2**Type**

float_array

Description

Norms of the rigid motions.

Shape

[nNormalModes+nRigidModes]

Vibrations%ModesNorm2***Type**

float_array

Description

Norms of the rigid motions (for a given irrep...?).

Shape

[nNormalModes+nRigidModes]

Vibrations%nNormalModes**Type**

int

Description

Number of normal modes.

Vibrations%NoWeightNormalMode (#)**Type**

float_array

Description

?.

Shape

[3, Molecule%nAtoms]

Vibrations%NoWeightRigidMode (#)**Type**

float_array

Description

?

Shape

[3, Molecule%nAtoms]

Vibrations%nRigidModes**Type**

int

Description

Number of rigid modes.

Vibrations%nSemiRigidModes**Type**

int

Description

Number of semi-rigid modes.

Vibrations%PVDOS**Type**

float_array

Description

Partial vibrational density of states.

Values range

[0.0, 1.0]

Shape

[nNormalModes, Molecule%nAtoms]

Vibrations%RamanDepolRatioLin**Type**

float_array

Description

Raman depol ratio (lin).

Shape

[nNormalModes]

Vibrations%RamanDepolRatioLinBeforeScan

Type

float_array

Description

?

Shape

[nNormalModes]

Vibrations%RamanDepolRatioNat**Type**

float_array

Description

Raman depol ratio (nat).

Shape

[nNormalModes]

Vibrations%RamanDepolRatioNatBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%RamanIncidentFreq**Type**

float

Description

Raman incident light frequency.

Unit

hartree

Vibrations%RamanIntens [A^4/amu]**Type**

float_array

Description

Raman intensities

Unit

A^4/amu

Shape

[nNormalModes]

Vibrations%RamanIntensBeforeScan [A^4/amu]**Type**

float_array

Description

Raman intensities

UnitA⁴/amu**Shape**

[nNormalModes]

Vibrations%ReducedMasses**Type**

float_array

Description

The reduced masses of the normal modes.

Unit

a.u.

Values range

[0, 'infinity']

Shape

[nNormalModes]

Vibrations%RotationalStrength**Type**

float_array

Description

The rotational strength of the normal modes.

Shape

[nNormalModes]

Vibrations%ScannedModes**Type**

int_array

Description

?

Shape

[nNormalModes]

Vibrations%TransformationMatrix**Type**

float_array

Description

?

Shape

[3, Molecule%nAtoms, nNormalModes]

Vibrations%VROACIDBackward**Type**

float_array

Description

VROA Circular Intensity Differential: Backward scattering.

Unit 10^{-3} **Shape**

[nNormalModes]

Vibrations%VROACIDBackwardBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%VROACIDDePolarized**Type**

float_array

Description

VROA Circular Intensity Differential: Depolarized scattering.

Unit 10^{-3} **Shape**

[nNormalModes]

Vibrations%VROACIDDePolarizedBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%VROACIDForward**Type**

float_array

Description

VROA Circular Intensity Differential: Forward scattering.

Unit 10^{-3} **Shape**

[nNormalModes]

Vibrations%VROACIDForwardBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%VROACIDPolarized**Type**

float_array

Description

VROA Circular Intensity Differential: Polarized scattering.

Unit 10^{-3} **Shape**

[nNormalModes]

Vibrations%VROACIDPolarizedBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%VROADeltaBackward**Type**

float_array

Description

VROA Intensity: Backward scattering.

Unit $10^{-3} \text{ A}^4/\text{amu}$ **Shape**

[nNormalModes]

Vibrations%VROADeltaBackwardBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%VROADeltaDePolarized**Type**

float_array

Description

VROA Intensity: Depolarized scattering.

Unit $10^{-3} \text{ A}^4/\text{amu}$

Shape

[nNormalModes]

Vibrations%VROADeltaDePolarizedBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%VROADeltaForward**Type**

float_array

Description

VROA Intensity: Forward scattering.

Unit $10^{-3} \text{ A}^4/\text{amu}$ **Shape**

[nNormalModes]

Vibrations%VROADeltaForwardBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%VROADeltaPolarized**Type**

float_array

Description

VROA Intensity: Polarized scattering.

Unit $10^{-3} \text{ A}^4/\text{amu}$ **Shape**

[nNormalModes]

Vibrations%VROADeltaPolarizedBeforeScan**Type**

float_array

Description

?

Shape

[nNormalModes]

Vibrations%ZeroPointEnergy

Type

float

Description

Vibrational zero-point energy.

Unit

hartree

VibronicStructure**Section content:** Data related to the Vibronic Structure Tracking procedure.**VibronicStructure%nspectrum****Type**

int

Description

?

VibronicStructure%spectrum**Type**

float_array

Description

?

17.1 What is the difference between AMS and the AMS driver?

Amsterdam Modeling Suite (AMS) refers to the whole suite which includes the AMS driver for complex potential energy surface tasks with any underlying engine. The AMS bundle further includes the integrated GUI and the compute engines ADF, BAND, DFTB, MOPAC, ReaxFF, UFF, COSMO-RS.

17.2 What is the AMS driver?

It's a driver for running molecular dynamics or exploring the potential energy surface (finding transition states etc.)

It can be used with different modules: ADF, BAND, DFTB, MOPAC, ReaxFF and external engines. And as such you can switch easily between them, e.g. to transfer a Hessian from a low-lying method.

17.3 What will AMS do when one point fails to converge?

With AMS2018.103 it will exit with an error.

Starting with r69000, AMS geometry optimizations try to continue even if engine fails to solve for a geometry.

This can happen if there are for example SCF convergence problems in the engine. The energy/gradients might then not be quite correct, but it is probably safe to continue the optimization and hope that things are fine again for the next step. Note that this fix also applies to PES scans, transition state searches, and any other applications that use geometry optimizations internally (e.g. elastic property calculations).

A

Add molecules (*molecular dynamics*), 134
 Add-ons, 325
 ADF, 317
 Adiabatic Hessian Franck–Condon, 287
 AMS input file, 9
 AMS_JOBNAME, 17
 AMS_RESULTSDIR, 17
 AMS_SWITCH_LOGFILE_AND_STDOUT, 17
 AMSbatch, 339
 ams.log, 18
 AMSPipe, 24
 ams.rkf, 19
 ams.rkf output file, 907
 Applications, 45
 APT, 285
 Atom energies, 332
 Atomic charges, 307
 Atomic masses, 38
 Atomic polar tensor, 285
 AtomTyping, 342
 Autocorrelation function, 426
 Automated PES exploration, 202
 Available engines, 317
 AverageBinPlot, 446

B

BAND, 317
 Barostats, 118
 Basin Hopping, 218
 Binary output files, 18
 Binding Sites Detection, 229
 Block constraints, 58
 Block normal modes, 251
 Bond Boost method, 156
 Bond energy calculation, 47
 Bond guessing algorithm, 312
 Bond orders, 310
 Bulk modulus, 242

C

Cell optimization, 53

Charge, 38
 Compute clusters, 15
 Conformers, 343
 Conjugate gradients (*geometry optimizer*), 67
 Constrained optimization, 55
 Constraints, 55
 CREST, 343
 CREST metadynamics, 145
 CVHD, 147

D

D3 dispersion add-on, 326
 D4 dispersion add-on, 326
 D5 dispersion add-on, 326
 Developer options, 565
 DFTB, 317
 Diffusion coefficient, 449
 Dimer method, 68
 Dipole moment, 307
 Double group symmetry, 568
 Double parallelism, 20
 Driver level parallelism, 20

E

eHEX method (*NEMD*), 179
 Elastic properties, 242
 Elastic tensor, 242
 Electric field, 39
 Engine add-ons, 325
 Engine input, 317
 Engine output files, 19
 Engines, 315
 Enthalpy, 269
 Entropy, 269
 Environment variables, 17
 EON, 202
 EON CreatingAndExtendingEnergyLand-
 scape, 225
 EON Optimizer, 227
 Examples, 456
 Excited state optimizations, 100
 External engines, 320

F

fbMC, 174
FCF, 287
FCF module, 287
FIRE (*geometry optimizer*), 63
Fixed atoms, 56
Fluorescence, 287
Force bias Monte Carlo, 174
Forces, 237
Fractional coordinates, 33
Franck-Condon factors, 286, 287
Free rotor interpolation corrections, 269

G

GCMC, 193
Geometry, 31
Geometry constraints, 55
Geometry convergence, 52
Geometry optimization, 49
Geometry optimization methods, 59
Geometry relaxation, 49
Gibbs free energy, 269
Grand canonical Monte Carlo, 193

H

Heat capacity, 269
Heat exchange (*molecular dynamics*), 179
Hessian, 238
HEX method (*NEMD*), 179
Histogram (*molecular dynamics*), 422
HOMO-LUMO gap, 313

I

Imaginary modes, 249
IMDHO, 297
Independent mode displaced harmonic oscillator, 297
Initial Hessian, 62
Input file syntax, 9
Input for AMS, 5
Interface to external programs, 320
Internal energy, 269
Intrinsic reaction coordinate, 93
IR frequencies, 247
IRC, 93
Irreducible representation, 568
Isotopes, 38

J

Job name, 17

L

Landscape Refinement, 223

Lattice, 31
Lattice constraints, 58
Lattice deformations, 129
Lattice optimization, 53
Lattice vectors, 32
Lattice vibrations, 273
L-BFGS (*geometry optimizer*), 66
Lennard-Jones potential, 324
Linear transit, 76
Logfile `ams.log`, 18

M

MBH, 251
MD, 101
MD trajectory analysis, 414
Mean square displacement, 435
MEP, 93
Minimum energy profile, 93
MLPotential, 317
Mobile block Hessian, 251
Mode intensity tracking, 263
Mode refinement, 255
Mode scanning, 253
Mode selecting, 266
Mode tracking, 257
Molecular dynamics, 101
Molecular dynamics binary log, 125
Molecular dynamics checkpoint, 122
Molecular dynamics PBC remapping, 129
Molecular orbitals info, 307
Molecule detection, 310
Molecule gun (*molecular dynamics*), 134
Moments of inertia, 273
MOPAC, 317

N

Nanoreactor, 177
NEB, 83
NEMD, 178
NEMD Velocities and Forces, 184
NEMD viscosity, 186
Non-isotropic stress, 330
Normal modes, 247
Nuclear gradients, 237
Nudged elastic band, 83

O

Output files, 18
Output of AMS, 18

P

Partial vibrational density of states, 273
Partial vibrational spectra, 273

PES point characterization, 238
 PES point properties, 235
 PES scan, 76
 Phonons, 273
 Phosphorescence, 287
 Pipe interface, 24
 Pipe protocol, 24
 PLAMS, 14
 PLUMED library, 144
 Point charges, 39
 Poisson ratio, 242
 Polarizability, 307
 Pressure, 329
 Pressure (*molecular dynamics*), 120
 Process Search, 213
 Properties, 9
 PVDOS, 273
 Python, 14

Q

Quasi-Newton (*geometry optimizer*), 60

R

Radial distribution function, 418
 Raman, 278
 Raman-active vibrations, 250
 RDF, 418
 Reaction boost method, 159
 ReaxFF, 317
 Regions, 37
 Remove molecules (*molecular dynamics*), 140
 Required citations, 571
 Resonance Raman, 279
 Resonance VROA, 283
 Restart (*geometry*), 41, 71
 Restart (*molecular dynamics*), 113
 Restraints, 332
 Results directory, 16, 18
 Run types, 45
 Running AMS, 13

S

Saddle Search, 214
 Scan coordinate, 77
 ScanFreq, 249
 Schönflies symbol, 568
 SCM_TMPDIR, 16
 Scratch directory, 16
 Scripting, 14
 Shear modulus, 242
 Single point calculation, 47
 Starting directory, 16
 Strain constraints, 58
 Stress tensor, 241

Structure relaxation, 49
 Subspecies, 568
 Super cell, 34
 Symmetric displacements, 250
 Symmetrization, 35
 Symmetry, 35, 568
 Symmetry label, 568

T

Targeted MD, 159
 Task farming, 20
 Tasks, 8, 45
 Temperature (*molecular dynamics*), 120
 Temporary directory, 16
 Thermodynamics, 269
 Thermostat, 116
 T-NEMD, 179
 Trajectory Replay, 191
 Trajectory sampling, 121
 Transition state search, 72
 Tribology (*NEMD*), 184
 Two-level parallelism, 20

U

UFF, 317
 Utilities, 338

V

VCD, 285
 VCDtools module, 453
 Vertical gradient Franck-Condon, 297
 VG-FC, 297
 Vibrational circular dichroism, 285
 vibrational polarizabilities, 286
 Vibrational Raman optical activity, 283
 Vibrational spectroscopy, 246
 Vibrationally resolved electronic spectra, 286
 vibronic density of states, 287, 296
 Vibronic-structure excited state, 286
 Viscosity, 448
 Volume regimes, 129
 VROA, 283

W

Wall potential, 335
 WCA potential, 337

X

XYZ file format, 565

Y

Young's modulus, 242

Z

Z-matrix, [32](#)