MOPAC Manual

Amsterdam Modeling Suite 2026.1

www.scm.com

Apr 02, 2026

CONTENTS

1 Introduction 1
  1.1 What's new in MOPAC 2019 . . . 1

2 AMS driver's tasks and properties 3
  2.1 Geometry, System definition . . . 3
  2.2 Tasks: exploring the PES . . . 3
  2.3 Properties in the AMS driver . . . 4

3 Input keywords 5
  3.1 Model Hamiltonian . . . 5
  3.2 Solvation . . . 6
  3.3 Properties . . . 8
  3.4 Technical settings . . . 9
  3.5 Extra keywords . . . 11

4 References 13

5 Examples 15
  5.1 Example: GeoOpt+Frequencies of different O2 spin states . . . 15
  5.2 Example: Polarizability and hyperpolarizabilities . . . 17
  5.3 Example: Phonons . . . 17
  5.4 Example: Geometry optimization of polyethylene . . . 19
  5.5 Example: External electric field . . . 20
  5.6 Example: Camp-King Converger . . . 22
  5.7 Example: pKa prediction (PLAMS) . . . 23

6 KF output files 25
  6.1 Accessing KF files . . . 25
  6.2 Sections and variables on mopac.rkf . . . 26
    6.2.1 AMSResults . . . 26
    6.2.2 BZcell(primitive cell) . . . 32
    6.2.3 DOS_Phonons . . . 34
    6.2.4 General . . . 35
    6.2.5 KFDefinitions . . . 37
    6.2.6 kspace(primitive cell) . . . 37
    6.2.7 Low Frequency Correction . . . 38
    6.2.8 Mobile Block Hessian . . . 39
    6.2.9 Molecule . . . 40
    6.2.10 MoleculeSuperCell . . . 44
    6.2.11 phonon_curves . . . 48
    6.2.12 Phonons . . . 51

6.2.13 Plot 52
6.2.14 Properties 53
6.2.15 SCF 54
6.2.16 Thermodynamics 55
6.2.17 Vibrations 61

7 FAQ **69**
7.1 What's the difference between MOPAC in AMS2019 and in previous versions (AMS2018)? 69
7.2 How to use keywords from the MOPAC manual? 69
7.2.1 Keywords used in MOPAC2012 69
7.3 Which properties from MOPAC can be visualized with the GUI? 69

Index **71**

CHAPTER

ONE

INTRODUCTION

MOPAC [*1* (page 13)] is a general-purpose semiempirical quantum chemistry engine for the study of molecular and periodic structures. A good trade-off between speed and accuracy is achieved through a minimal basis and parameterization against experimental data, with parameters for most elements.

As of the 2026.1 release of the Amsterdam Modeling Suite, MOPAC has become an engine in the new AMS driver setup. If you have not done so yet, we highly recommend you to first read the General section of the AMS Manual. In practice the inclusion of MOPAC into AMS means that MOPAC can now be used for many applications that were previously not supported:

- Linear transit and PES scan
- Constrained geometry optimizations
- Molecular dynamics simulations
- Lattice optimization (also under pressure)
- Elastic tensor and related properties (e.g. Bulk modulus)
- Phonon calculations
- …

Please refer to the AMS manual for a complete overview.

1.1 What's new in MOPAC 2019

- MOPAC has been fully integrated as an Engine in the Amsterdam Modeling Suite; this significantly speeds up the execution of MOPAC via AMS.
- Parallel binaries.

New input options (also available via the Graphical User Interface):

- Calculation of *pKa* (page 8)
- *COSMO* (page 6): all solvents available in ADF/Band are now also available in MOPAC.
- *Static polarizability tensor* (page 8)
- Localized orbitals (Natural Bond Orbitals)
- *SCF options* (page 9): Camp-King converger, …

CHAPTER

TWO

AMS DRIVER'S TASKS AND PROPERTIES

MOPAC is an engine used by the AMS driver. While MOPAC's specific options and properties are described in this manual, the definition of the system, the selection of the task and certain (PES-related) properties are documented in the AMS driver's manual.

In this page you will find useful links to the relevant sections of the AMS driver's Manual.

2.1 Geometry, System definition

The definition of the system, i.e. the atom types and atomic coordinates (and optionally, the systems' net charge, the lattice vector, the input bond orders, external homogeneous electric field, atomic masses for isotopes) are part of the AMS driver input. See the System definition section of the AMS manual.

2.2 Tasks: exploring the PES

The job of the AMS driver is to handle all changes in the simulated system's geometry, e.g. during a geometry optimization or molecular dynamics calculation, using energy and forces calculated by the engine.

These are the tasks available in the AMS driver:

- Single Point
- Geometry Optimization
- Transition State Search
- IRC (Intrinsic Reaction Coordinate)
- PESScan (Potential Energy Surface Scan, including linear transit)
- NEB (Nudged Elastic Band)
- Vibrational Analysis
- Molecular Dynamics
- GCMC (Grand Canonical Monte Carlo)

2.3 Properties in the AMS driver

The following properties can be requested to the MOPAC engine in the AMS driver's input:

- Bond orders
- Atomic charges
- Dipole Moment
- Dipole Gradients
- Elastic tensor
- Nuclear Gradients / Forces
- Hessian
- Infrared (IR) spectra / Normal Modes
- Thermodynamic properties
- PES point character
- Phonons
- Stress tensor
- Elastic tensor
- VCD (Vibrational Circular Dichroism)

CHAPTER

THREE

INPUT KEYWORDS

This manual documents the input for the MOPAC engine used together with the AMS driver. If you are not yet familiar with the AMS driver setup, we highly recommend reading the introductory section in the AMS manual.

The MOPAC engine is selected and configured in the AMS input with

```
Engine MOPAC
   ... keywords documented in this manual ...
EndEngine
```

This page documents all keywords of the MOPAC engine input, basically the contents of the `Engine MOPAC` block in the AMS input file.

General remarks on the input syntax can be found in the AMS manual.

See also:

The *Examples* (page 15) section of this manual contains several example calculations

3.1 Model Hamiltonian

The most important keyword in the MOPAC engine input is the model selection:

`Model`

Type
Multiple Choice

Default value
PM7

Options
[AM1, MNDO, MNDOD, PM3, RM1, PM6, PM6-D3, PM6-DH+, PM6-DH2, PM6-DH2X, PM6-D3H4X, PM7]

GUI name
Method

Description
Selects the model Hamiltonian to use in the calculation.

AM1: Use the AM1 Hamiltonian.

MNDO: Use the MNDO Hamiltonian.

MNDOD: Use the MNDO-d Hamiltonian.

RM1: Use the RM1 Hamiltonian.

PM3: Use the MNDO-PM3 Hamiltonian.

PM6: Use the PM6 Hamiltonian.

PM6-D3: Use the PM6 Hamiltonian with Grimme's D3 corrections for dispersion.

PM6-DH+: Use the PM6 Hamiltonian with corrections for dispersion and hydrogen-bonding.

PM6-DH2: Use the PM6 Hamiltonian with corrections for dispersion and hydrogen-bonding.

PM6-DH2X: Use PM6 with corrections for dispersion and hydrogen and halogen bonding.

PM6-D3H4: Use PM6 with Rezac and Hobza's D3H4 correction.

PM6-D3H4X: Use PM6 with Brahmkshatriya, et al.'s D3H4X correction.

PM7: Use the PM7 Hamiltonian.

PM7-TS: Use the PM7-TS Hamiltonian (only for barrier heights)

The default PM7 model [*2* (page 13)] is the latest parametrization for MOPAC and should be the most accurate for most calculations.

`Sparkles`

Type
Bool

Default value
No

Description
Represent lanthanides by their fully ionized 3+ sparkles. That is, they have no basis set, and therefore cannot have a charge different from +3. When using sparkles, the geometries of the lanthanides are reproduced with good accuracy, but the heats of formation and electronic properties are not accurate.

`UnpairedElectrons`

Type
Integer

GUI name
Spin polarization

Description
If this key is present, a spin-unrestricted calculation with the specified number of unpaired electrons is performed. If this key is not present the number of unpaired electrons is determined automatically (0 for systems with an even number of electrons, 1 for radicals), and a restricted or unrestricted calculation is performed accordingly.

3.2 Solvation

Solvation effects can be included via the COSMO model.

```
Solvation
   Enabled Yes/No
   NSPA [...]
   Solvent
      Eps float
      Name [...]
```

(continues on next page)

(continued from previous page)

```
      Rad float
   End
End
```

`Solvation`

Type
Block

Description
Options for the COSMO (Conductor like Screening Model) solvation model.

`Enabled`

Type
Bool

Default value
No

GUI name
Use COSMO

Description
Use the Conductor like Screening Model (COSMO) to include solvent effects.

`NSPA`

Type
Multiple Choice

Default value
42

Options
[12, 32, 42, 92, 122, 162, 252, 272, 362, 482, 492, 642, 752]

GUI name
NSPA

Description
Maximum number of COSMO surface points per atom.

`Solvent`

Type
Block

Description
Solvent details

`Eps`

Type
Float

GUI name
Dielectric constant

Description
User-defined dielectric constant of the solvent (overrides the Eps value of the solvent defined in ‘Name’)

Name

Type
Multiple Choice

Default value
Water

Options
[CRS, AceticAcid, Acetone, Acetonitrile, Ammonia, Aniline, Benzene, BenzylAlcohol, Bromoform, Butanol, isoButanol, tertButanol, CarbonDisulfide, CarbonTetrachloride, Chloroform, Cyclohexane, Cyclohexanone, Dichlorobenzene, DiethylEther, Dioxane, DMFA, DMSO, Ethanol, EthylAcetate, Dichloroethane, EthyleneGlycol, Formamide, FormicAcid, Glycerol, HexamethylPhosphoramide, Hexane, Hydrazine, Methanol, MethylEthylKetone, Dichloromethane, Methylformamide, Methypyrrolidinone, Nitrobenzene, Nitrogen, Nitromethane, PhosphorylChloride, IsoPropanol, Pyridine, Sulfolane, Tetrahydrofuran, Toluene, Triethylamine, TrifluoroaceticAcid, Water]

GUI name
Solvent

Description
Name of a pre-defined solvent. A solvent is characterized by the dielectric constant (Eps) and the solvent radius (Rad).

Rad

Type
Float

Unit
Angstrom

GUI name
Radius

Description
User-defined radius of the solvent molecule (overrides the Rad value of the solvent defined in 'Name').

3.3 Properties

```
Properties
   StaticPolarizability Yes/No
   pKa Yes/No
End
```

Properties

Type
Block

Description
MOPAC can calculate various properties of the simulated system. This block configures which properties will be calculated.

StaticPolarizability

Type
Bool

Default value
No

Description
Calculate the static polarizability. An electric field gradient is applied to the system, and the response is calculated. The dipole and polarizability are calculated two different ways, from the change in heat of formation and from the change in dipole. A measure of the imprecision of the calculation can be obtained by comparing the two quantities.

`pKa`

Type
Bool

Default value
No

GUI name
pKa

Description
If requested, the pKa of hydrogen atoms attached to oxygen atoms is calculated and printed.

The calculation of Natural Bond Orbitals can be requested with the following keyword:

```
CalcLocalOrbitals Yes/No
```

`CalcLocalOrbitals`

Type
Bool

Default value
No

Description
Compute and print the localized orbitals, also known as Natural Bond Orbitals (NBO). This is equivalent to the LOCAL mopac keyword.

The calculation of bond orders can be requested in the AMS Properties block.

3.4 Technical settings

```
SCF
   CampKingConverger Yes/No
   ConvergenceThreshold float
   MaxIterations integer
End
```

`SCF`

Type
Block

Description
Options for the self-consistent field procedure.

`CampKingConverger`

Type
Bool

Default value
No

GUI name
Use Camp-King

Description
Use the Camp-King SCF converger. This is a very powerful, but CPU intensive, SCF converger.

`ConvergenceThreshold`

Type
Float

Default value
0.0001

Unit
kcal/mol

Description
If the difference in energy between two successive SCF iterations is smaller than this value, the SCF procedure is considered converged.

`MaxIterations`

Type
Integer

Default value
2000

Description
Maximum number of SCF iterations.

With the MOZYME method the standard SCF procedure is replaced with a localized molecular orbital (LMO) method. This can speed-up the calculation of large molecules. Although a job that uses the MOZYME technique should give results that are the same as conventional SCF calculations, in practice there are differences. Most of these differences are small, but in some jobs the differences between MOZYME and conventional SCF calculations can be significant. Use with care.

`Mozyme`

Type
Bool

Default value
No

Description
Replace the standard SCF procedure with a localized molecular orbital (LMO) method. The time required for an SCF cycle when Mozyme is used scales linearly with system size.

3.5 Extra keywords

Finally it is possible to pass any other keywords directly to the MOPAC program [*1* (page 13)]. The full list of keywords can be found on the standalone MOPAC manual (http://openmopac.net/manual/index.html).

```
Keywords string
```

`Keywords`

Type
String

Description
A string containing all the desired custom MOPAC keywords. Basically for anything not directly supported through AMS.

These keywords are just literally passed through to MOPAC program which the AMS MOPAC engine wraps, without any checking in AMS. One should therefore be very careful with this, as it is very easy to set up completely non-sensical calculations in this way.

Note: The following keywords have been either removed or renamed in our version of MOPAC and they should not be used in the `Keywords` key: 0SCF, 1SCF, A0, ADD, AIDER, AIGIN, AIGOUT, ALT_A, ALT_R, ANGSTROMS, AUTOSYM, BANANA, BAR, BCC, BFGS, BIGCYCLES, BIRADICAL, CHAINS, COMPARE, CVB, DDMAX, DDMIN, DFORCE, DFP, DMAX, DRC, ECHO, EF, FLEPO, FORCE, FREQCY, GNORM, H, HTML, INT, IONIZE, IRC, ISOTOPE, KINETIC, LBFGS, LET, LOCATE, MODE, NOCOMMENTS, NOOPT, NORESEQ, NOSWAP, NOTER, NOTHIEL, NOTXT, OPT, P, PDB, PDBOUT, POINT, POINT1, POINT2, RABBIT, RECALC, RMAX, RMIN, SIGMA, SLOG, SMOOTH, SNAP, START_RES, STEP, STEP1, STEP2, SYBYL, T, THERMO, THREADS, TIMES, TRANS, TS, VELOCITY, X, XENO, XYZ,, AM1, LOCAL, BONDS, CHARGE, UHF, CAMP, KING, ITRY, EPS, FIELD, pKa, STATIC, CYCLES, PRESSURE, SPARKLE.

CHAPTER

FOUR

REFERENCES

The MOPAC engine in the 2026.1 of the Amsterdam Modeling Suite is a modified version of the standalone MOPAC2016 program developed by Dr. Jimmy Stewart.

1. AMS 2026.1 MOPAC: MOPAC Engine based on the MOPAC2016 source code (James J.P. http://openmopac.net)
2. James J.P. Stewart, *Optimization of parameters for semiempirical methods VI: more modifications to the NDDO approximations and re-optimization of parameters*, J. Mol. Modeling 19, 1-32 (2013) (https://doi.org/10.1007/s00894-012-1667-x)

A full list of references for the MOPAC package can be found on the official MOPAC references page (http://openmopac.net/Manual/references.html).

CHAPTER

FIVE

EXAMPLES

The *$AMSHOME/examples/mopac* directory contains many different example files, covering various MOPAC options. This is a selection of relevant examples.

5.1 Example: GeoOpt+Frequencies of different O2 spin states

Download GOFREQ_unrestricted.run

```
#!/bin/sh

# Neutral O2 singlet state
# ========================

AMS_JOBNAME=O2_singlet $AMSBIN/ams << EOF

Task GeometryOptimization

Properties
   NormalModes Yes
End

System
   Atoms
      O 1.5 0.0 0.0
      O 0.0 0.0 0.0
   End
End

Engine MOPAC
EndEngine
EOF

echo "O2 bond distance (singlet)"
$AMSBIN/amsreport O2_singlet.results/ams.rkf distance#1#2


# O2+ doublet state
# =================

AMS_JOBNAME=O2+_doublet $AMSBIN/ams << EOF

Task GeometryOptimization

```

(continues on next page)

(continued from previous page)

```
Properties
   NormalModes Yes
End

System
   Atoms
      O 1.5 0.0 0.0
      O 0.0 0.0 0.0
   End
   Charge 1
End

Engine MOPAC
   UnpairedElectrons 1
EndEngine
EOF

echo "O2 bond distance (doublet, charged)"
$AMSBIN/amsreport O2+_doublet.results/ams.rkf distance#1#2


# Neutral O2 triplet state
# ========================

AMS_JOBNAME=O2_triplet $AMSBIN/ams << EOF

Task GeometryOptimization

Properties
   NormalModes Yes
End

System
   Atoms
      O 1.5 0.0 0.0
      O 0.0 0.0 0.0
   End
End

Engine MOPAC
   UnpairedElectrons 2
EndEngine
EOF

echo "O2 bond distance (triplet)"
$AMSBIN/amsreport O2_triplet.results/ams.rkf distance#1#2
```

5.2 Example: Polarizability and hyperpolarizabilities

Download Polar.run

```
#! /bin/sh

# Compute polarizability and first and second hyperpolarizabilities.
# The string in the 'Keywords' key is passed to the input-parsing routines of MOPAC.

$AMSBIN/ams << eor

Task SinglePoint

System
    Atoms
        C -0.917657604523966  0.464763072607994 -0.042272407464148
        C  0.599132389604762  0.488150975335481  0.042272407810247
        H -1.336541780023175  1.363372335927188  0.457720688164060
        H -1.308637306012442 -0.446333757344598  0.457720688143968
        H -1.234937187765967  0.459870835772842 -1.106331392792046
        H  0.990112088660506  1.399247806016238 -0.457720688423546
        H  1.018016566995508 -0.410458286745563 -0.457720688426743
        H  0.916411973169395  0.493043222972654  1.106331392988198
    End
End

Engine MOPAC
    Keywords POLAR(E=(1.0))
EndEngine

eor

# The 'polar' results are printed to the mopac.out file, which is located in the ams
# results folder (and not to standard output)

cat ams.results/mopac.out
```

5.3 Example: Phonons

Download phonons.run

```
#! /bin/sh

# Phonons for polyphenylene vinylene (PPV)
# ========================================

AMS_JOBNAME=PPV $AMSBIN/ams << eor

Task SinglePoint

System
   Atoms
      C  1.432420914962878  -1.133348744664622  -0.6391103371334507
      C  0.075602182675705  -0.946866493711738  -0.5497084115413023
      C  2.345587368530869  -0.191932196525464  -0.0965381875924778
```

(continues on next page)

(continued from previous page)

```
      C -0.466207830009865   0.191351632533680   0.0976709467922905
      C  1.803663911626683   0.948320770238396   0.5481842048314089
      C  0.446862721780109   1.134635005787038   0.6370714302545314
      C -1.855533046352049   0.415640484802555   0.2316022019049204
      C -2.841044836424757  -0.419157153044205  -0.2271278521017774
      H -0.602199468183589  -1.681633760082688  -0.9836845375123017
      H  2.480073119105696   1.685566870120453   0.9806344160825713
      H  0.050338193748088   2.021718778887199   1.1315059772026770
      H  1.827043768886768  -2.019275515588153  -1.1372628449390670
      H -2.553512025749108  -1.341888903294209  -0.7454241111668017
      H -2.143094970839948   1.336869222541756   0.7521871009187797
   End
   Lattice
      6.575588248161897 0.0 0.0
   End
End

Properties
   Phonons Yes
End

NumericalPhonons
   SuperCell
      3
   End
End

Engine MOPAC
   SCF
      ConvergenceThreshold 1.0E-5
   End
EndEngine

eor

# Phonons for Boron-Nitrade slab (2x2 super cell)
# ===============================================

AMS_JOBNAME=BN $AMSBIN/ams << eor

Task SinglePoint

System
   Atoms
      N  1.275622848015759  -0.736481194060720  0.0
      N  2.551245696034436   1.472962389682135  0.0
      B -2.551245696034436  -1.472962389682135  0.0
      B -1.275622848015759   0.736481194060720  0.0
      B  0.0                -1.472962389679606  0.0
      B  1.275622848017218   0.736481194063248  0.0
      N -1.275622848017218  -0.736481194063248  0.0
      N  0.0                 1.472962389679606  0.0
   End
   Lattice
      5.102491392075644 0.0 0.0
      2.551245696042202 4.418887167494105 0.0
   End
```

(continues on next page)

(continued from previous page)

```
End

Properties
   Phonons Yes
End

NumericalPhonons
   SuperCell
      2 0
      0 2
   End
End

Engine MOPAC
   SCF
      ConvergenceThreshold 1.0E-5
   End
EndEngine

eor
```

5.4 Example: Geometry optimization of polyethylene

Download GO_polyethylene.run

```
#! /bin/sh

# Geometry optimization of a slighly distorted polyethelene chain (6 units in the␣
↪unit cell)

$AMSBIN/ams << eor

Task GeometryOptimization

GeometryOptimization
   Convergence
      Gradients 1.0e-4
   End
End

System
   Atoms
      C -5.686966610289906 -0.00173661090043054 -0.4355683776313619
      C 1.895723638480955 -0.00173661090043054 -0.4355683776313619
      C -3.159403194032952 -0.00173661090043054 -0.4355683776313619
      C 4.491312517927723 -0.0863455367929557 -0.474315563245167
      C -0.6414620718677587 0.2951925083203292 -0.3915990966867868
      C 6.950850470994863 -0.00173661090043054 -0.4355683776313619
      H -6.951201432748922 0.8860020896101368 1.098388839692907
      H 0.7283521430793004 0.9062923240105974 0.9236806626313948
      H -4.047903951160414 0.9426765116296983 0.8853722637672539
      H 3.145873269393606 0.7752976020042145 1.050585933807339
      H -1.902858714187983 1.074510344152748 1.180825231795906
      H 5.579937435062504 1.017854159367372 1.025095354070417
```

(continues on next page)

(continued from previous page)

```
      H -6.950939675307238 -0.8793662426450884 1.105233273612651
      H 0.6317505734636235 -0.8793662426450884 1.105233273612651
      H -4.423376259050285 -0.8793662426450884 1.105233273612651
      H 3.146135026835287 -0.9900707302510107 1.057430367727084
      H -2.067352365692016 -0.7586675287504774 1.334377669481547
      H 5.686877405977529 -0.8793662426450884 1.105233273612651
      H -5.686618534103184 0.8797167676702464 -1.103339585577878
      H 1.790283468854915 0.878947797439763 -1.127416231785004
      H -3.15905511784623 0.8797167676702464 -1.103339585577878
      H 4.410456168039341 0.7690122800643241 -1.151142491463446
      H -0.576790284167599 1.020121306579756 -1.135070326918629
      H 7.127011768776353 0.7534682953709397 -1.016196632797457
      H -5.571852371888105 -0.783856089153288 -1.124998626807626
      H 1.895778201220154 -0.8852857607466311 -1.100407519984987
      H -3.159348631293752 -0.8852857607466311 -1.100407519984987
      H 4.410162654591847 -0.9959902483525568 -1.148210425870549
      H -0.6344641535070402 -0.6484916142655238 -1.015540900330991
      H 6.950905033734066 -0.8852857607466311 -1.100407519984987
      C -6.950812943854352 0.0006697570117673826 0.4356933698886703
      C 0.7242710564399106 0.03708203634208995 0.4116378321176493
      C -4.428926336438604 -0.0461213910675544 0.3956424425613723
      C 3.38068654505114 -0.01625773059919498 0.3275387816426286
      C -1.921486943590773 0.2660741237064986 0.6146828354694926
      C 5.687004137430418 0.0006697570117673826 0.4356933698886703
   End
   Lattice
      15.16538049754172 0.0 0.0
   End
End

Engine MOPAC
EndEngine

eor
```

5.5 Example: External electric field

Download EField.run

```
#! /bin/sh

# Induce a dipole moment in benzene by applying a field orthogonal to the ring

for EField in 0 0.051422 0.51422 5.1422 ; do # which is 0.001 0.01 0.1 in atomic units

AMS_JOBNAME=benzene_$EField $AMSBIN/ams << eor

Task SinglePoint
System
    Atoms
        C   2.09820318  1.21139817   0.0
        C -0.69940106  1.21139817   0.0
        C   1.39880212  0.0          0.0
        C   1.39880212  2.42279634   0.0
```

(continues on next page)

(continued from previous page)

```
        C  0.0         2.42279634   0.0
        C  0.0         0.0          0.0
        H  3.18949204  1.21139817   0.0
        H  1.94444655  3.36788021   0.0
        H -0.54564443  3.36788021   0.0
        H -1.79068992  1.21139817   0.0
        H -0.54564443 -0.94508387   0.0
        H  1.94444655 -0.94508387   0.0
    End
    ElectrostaticEmbedding
        ElectricField 0.0 0.0 $EField
    End
End

Engine MOPAC
EndEngine

eor

done

# If I apply an electric field of 1 [a.u.] (51.42 Volt/Angstrom = 1 a.u.) on a system␣
↪with charge 1,
# I expect the net force to be equal to the 1 [a.u.]

AMS_JOBNAME=OH_plus $AMSBIN/ams << eor

Task SinglePoint
System
    Atoms
        O 0.0 0.0 0.0
        H 1.0 0.0 0.0
    End
    Charge 1
    ElectrostaticEmbedding
        ElectricField 0.0 51.422 0.0
    End
End

Properties
    Gradients Yes
End

Engine MOPAC
EndEngine

eor
```

5.6 Example: Camp-King Converger

Download CampKingConverger.run

```
#! /bin/sh

# Single point calculation using the non-default Camp-King converger.
# This is a very powerful, but CPU intensive, SCF converger.

$AMSBIN/ams << eor

Task SinglePoint
System
   Atoms
      Au  0.991939 -1.013256  6.087687
      N   0.671226 -0.526321  4.067029
      Au  1.387933 -1.619200  8.613660
      C  -0.555388 -0.148486  3.616932
      C   1.681804 -0.577717  3.158165
      Au -1.113240 -0.959652  8.002939
      Au  3.551662 -1.763674  7.076475
      C  -0.799200  0.178830  2.295071
      H  -1.346696 -0.123715  4.362730
      C   1.503026 -0.266909  1.821998
      H   2.653410 -0.874661  3.546413
      C   0.236703  0.129690  1.334288
      H  -1.814620  0.448943  2.011007
      H   2.368652 -0.310806  1.163512
      C   0.011948  0.467735 -0.077072
      C   0.874402  0.017077 -1.100014
      C  -1.079433  1.261872 -0.491478
      C   0.629560  0.357920 -2.422535
      H   1.729669 -0.619555 -0.876597
      C  -1.259607  1.557051 -1.835225
      H  -1.784097  1.673119  0.230431
      N  -0.422938  1.118535 -2.804801
      H   1.284502  0.027292 -3.228959
      H  -2.097962  2.162456 -2.180355
      Au -0.765534  1.615397 -4.922645
      Au -1.186659  2.214533 -7.501957
      Au -3.056147  2.893410 -5.586159
      Au  1.119984  0.909275 -6.730463
      Br -1.580087  2.774299 -9.904465
   End
   Charge -1
End

Engine MOPAC
   SCF
      ConvergenceThreshold 1.0E-8
      CampKingConverger Yes
   End
EndEngine

eor
```

5.7 Example: pKa prediction (PLAMS)

This example should be executed using PLAMS.

`Download pKa.py`

```
from scm.plams.interfaces.molecule.rdkit import from_smiles
import numpy as np
import multiprocessing

# In this example we compute pKa (acid dissociation constant) using MOPAC for a set of
# molecules. The molecules are defined using smiles strings, and are converted to xyz
# structures using the plams-rdkit interface.

# Important note: the predicted pKa strongly depend on the molecule's conformer.
# Here we use the lowest conformer predicted by rdkit's UFF.
# The difference between the values computed here and the results on the
# MOPAC website (ref_mopac_pKa) is due to different conformers

# Data taken from the online MOPAC manual: http://openmopac.net/manual/ (only a sub␣
↪set)
data_tmp = [
    # Molecule name                 smiles                            exp_pKa   ref_
↪mopac_pKa (from mopac's website)
    ["1-Naphthoic_acid", "C1=CC=C2C(=C1)C=CC=C2C(=O)O", 3.69, 4.35],
    ["2,2,2-Trichloroethanol", "C(C(Cl)(Cl)Cl)O", 12.02, 12.22],
    ["2,2,2-Trifluoroethanol", "C(C(F)(F)F)O", 12.40, 12.27],
    ["2,2-Dimethylpropionic_acid", "CC(C)(C)C(=O)O", 5.03, 5.23],
    ["2,3,4,6-Tetrachlorophenol", "C1=C(C(=C(C(=C1Cl)Cl)Cl)O)Cl", 7.10, 6.08],
    ["Acetic_acid", "CC(=O)O", 4.76, 5.00],
    ["Acrylic_acid", "C=CC(=O)O", 4.25, 4.65],
    ["Benzoid_acid", "C1=CC=C(C=C1)C(=O)O", 4.20, 4.30],
    ["Citric_acid", "C(C(=O)O)C(CC(=O)O)(C(=O)O)O", 3.13, 2.56],
    ["Ethanol", "CCO", 16.00, 16.37],
    ["Formic_acid", "C(=O)O", 3.77, 3.77],
    ["Glycine", "C(C(=O)O)N", 2.35, 2.53],
    ["Isoleucine", "CCC(C)C(C(=O)O)N", 2.32, 2.48],
    ["Methanol", "CO", 15.54, 15.23],
    ["o-Nitrophenol", "C1=CC=C(C(=C1)[N+](=O)[O-])O", 7.17, 7.52],
    ["Pentachlorophenol", "C1(=C(C(=C(C(=C1Cl)Cl)Cl)Cl)Cl)O", 4.90, 5.55],
    ["Phenol", "C1=CC=C(C=C1)O", 10.00, 9.71],
    ["Pyruvic_acid", "CC(=O)C(=O)O", 2.50, 2.85],
    ["T-Butanol", "CC(C)(C)O", 17.00, 16.25],
    ["Terephthalic_acid", "C1=CC(=CC=C1C(=O)O)C(=O)O", 3.51, 3.59],
    ["Valine", "CC(C)C(C(=O)O)N", 2.29, 2.61],
    ["Water", "O", 15.74, 15.75],
]

# Turn data_tmp into a dictionary:
systems = [{"name": d[0], "smiles": d[1], "exp_pKa": d[2], "ref_mopac_pKa": d[3]} for␣
↪d in data_tmp]

# Create the molecules from the smiles using rdkit:
molecules = []
for system in systems:
    # Compute 30 conformers, optimize with UFF and pick the lowest in energy.
    mol = from_smiles(system["smiles"], nconfs=30, forcefield="uff")[0]
```

(continues on next page)

(continued from previous page)

```
    mol.properties.name = system["name"]
    mol.properties.exp_pKa = system["exp_pKa"]
    mol.properties.ref_mopac_pKa = system["ref_mopac_pKa"]

    molecules.append(mol)

# MOPAC input:
s = Settings()
s.runscript.nproc = 1  # serial calculation
s.input.ams.Task = "GeometryOptimization"
s.input.ams.GeometryOptimization.Convergence.Step = 1.0e-3
s.input.ams.GeometryOptimization.Convergence.Gradients = 1.0e-5
s.input.mopac.model = "PM6"
s.input.mopac.properties.pKa = "Yes"

# Set up and run jobs:
jobs = MultiJob(children=[AMSJob(name=mol.properties.name, molecule=mol, settings=s)␣
↪for mol in molecules])
jr = JobRunner(parallel=True, maxjobs=multiprocessing.cpu_count())  # run jobs in␣
↪parallel
jobs.run(jobrunner=jr)

# Collect results:
for i, mol in enumerate(molecules):
    pKaValues = jobs.children[i].results.readrkf("Properties", "pKaValues", file=
↪"mopac")
    mol.properties.calc_pKa = np.mean(pKaValues)  # If there is more than one pKa,␣
↪take the average value

# Print results in a table:
print("Results:\n")
print("| {:28} | {:8} | {:8} | {:8} | {:8} |".format("Molecule", "exp pKa", "calc pKa
↪", "ref", "calc-exp"))
for mol in molecules:
    print(
        "| {:28} | {:>8.2f} | {:>8.4f} | {:>8.2f} | {:>8.2f} |".format(
            mol.properties.name,
            mol.properties.exp_pKa,
            mol.properties.calc_pKa,
            mol.properties.ref_mopac_pKa,
            mol.properties.calc_pKa - mol.properties.exp_pKa,
        )
    )
print("")

errors = [mol.properties.calc_pKa - mol.properties.exp_pKa for mol in molecules]

print("Mean signed error  : {:4.2f}".format(np.mean(errors)))
print("Mean unsigned error: {:4.2f}".format(np.mean([abs(e) for e in errors])))
print("Root mean square error: {:4.2f}".format(np.sqrt(np.mean([e**2 for e in␣
↪errors]))))
print("Done")
```

CHAPTER

SIX

KF OUTPUT FILES

6.1 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.

6.2 Sections and variables on mopac.rkf

6.2.1 AMSResults

KF Section: AMSResults

Content: Generic results of the MOPAC Engine evaluation.

`AMSResults%Bonds`

Type
subsection

Description
Bond info

`AMSResults%Bonds.Atoms`

Type
archived_int_array

Description
?

`AMSResults%Bonds.CellShifts`

Type
archived_int_array

Description
?

`AMSResults%Bonds.description`

Type
string

Description
A string containing a description of how the bond orders were calculated / where they come from

`AMSResults%Bonds.hasCellShifts`

Type
bool

Description
Whether there are cell shifts (relevant only in case of periodic boundary conditions)

`AMSResults%Bonds.Index`

Type
archived_int_array

Description
index(i) points to the first element of Atoms, Orders, and CellShifts belonging to bonds from atom 'i'. Index(1) is always 1, Index(nAtoms+1) is always nBonds + 1

`AMSResults%Bonds.nLattVec`

Type
int

Description
Number of lattice vectors (0:molecule, 1:chain, 2:slab, 3:bulk). This determines how the lattice displacements for bonds are interpreted.

AMSResults%Bonds.Orders

Type
archived_float_array

Description
The bond orders.

AMSResults%BulkModulus

Type
float

Description
The Bulk modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

AMSResults%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]). The method used to compute these atomic charges depends on the engine.

Unit
e

Shape
[Molecule%nAtoms]

AMSResults%DipoleGradients

Type
float_array

Description
Derivative of the dipole moment with respect to nuclear displacements.

Shape
[3, 3, Molecule%nAtoms]

AMSResults%DipoleMoment

Type
float_array

Description
Dipole moment vector (x,y,z)

Unit
e*bohr

Shape
[3]

AMSResults%ElasticTensor

Type
float_array

Description
The elastic tensor in Voigt notation (6x6 matrix for 3D periodic systems, 3x3 matrix for 2D periodic systems, 1x1 matrix for 1D periodic systems).

Unit
hartree/bohr^nLatticeVectors

Shape
[:, :]

`AMSResults%Energy`

Type
float

Description
The energy computed by the engine.

Unit
hartree

`AMSResults%fractionalOccupation`

Type
bool

Description
Whether of not we have fractionally occupied orbitals (i.e. not all occupations are integer numbers).

`AMSResults%Gradients`

Type
float_array

Description
The nuclear gradients.

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

`AMSResults%Hessian`

Type
float_array

Description
The Hessian matrix

Unit
hartree/bohr^2

Shape
[3*Molecule%nAtoms, 3*Molecule%nAtoms]

`AMSResults%HOMOEnergy`

Type
float_array

Description
Molecular Orbital Info: energy of the HOMO.

Unit
hartree

Shape
[nSpin]

`AMSResults%HOMOIndex`

Type
int_array

Description
Molecular Orbital Info: index in the arrays orbitalEnergies and orbitalOccupations corresponding to the HOMO.

Shape
[nSpin]

`AMSResults%HOMOLUMOGap`

Type
float_array

Description
Molecular Orbital Info: HOMO-LUMO gap per spin.

Unit
hartree

Shape
[nSpin]

`AMSResults%LUMOEnergy`

Type
float_array

Description
Molecular Orbital Info: energy of the LUMO.

Unit
hartree

Shape
[nSpin]

`AMSResults%LUMOIndex`

Type
int_array

Description
Molecular Orbital Info: index in the arrays orbitalEnergies and orbitalOccupations corresponding to the LUMO.

Shape
[nSpin]

`AMSResults%Molecules`

Type
subsection

Description
Molecules

`AMSResults%Molecules.AtCount`

Type
archived_int_array

Description
shape=(nMolType), Summary: number of atoms per formula.

`AMSResults%Molecules.Atoms`

Type
archived_int_array

Description
shape=(nAtoms), atoms(index(i):index(i+1)-1) = atom indices of molecule i

`AMSResults%Molecules.Count`

Type
archived_int_array

Description
Mol count per formula.

`AMSResults%Molecules.Formulas`

Type
string

Description
Summary: unique molecule formulas

`AMSResults%Molecules.Index`

Type
archived_int_array

Description
shape=(nMol+1), index(i) = index of the first atom of molecule i in array atoms(:)

`AMSResults%Molecules.Type`

Type
archived_int_array

Description
shape=(nMol), type of the molecule, reference to the summary arrays below

`AMSResults%nOrbitals`

Type
int

Description
Molecular Orbital Info: number of orbitals.

`AMSResults%nSpin`

Type
int

Description
Molecular Orbital Info: number spins (1: spin-restricted or spin-orbit coupling, 2: spin unrestricted).

`AMSResults%orbitalEnergies`

Type
float_array

Description
Molecular Orbital Info: the orbital energies.

Unit
hartree

Shape
[nOrbitals, nSpin]

`AMSResults%orbitalOccupations`

Type
float_array

Description
Molecular Orbital Info: the orbital occupation numbers. For spin restricted calculations, the value will be between 0 and 2. For spin unrestricted or spin-orbit coupling the values will be between 0 and 1.

Shape
[nOrbitals, nSpin]

`AMSResults%PESPointCharacter`

Type
string

Description
The character of a PES point.

Possible values
['local minimum', 'transition state', 'stationary point with >1 negative frequencies', 'non-stationary point']

`AMSResults%PoissonRatio`

Type
float

Description
The Poisson ratio

`AMSResults%ShearModulus`

Type
float

Description
The Shear modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

`AMSResults%SmallestHOMOLUMOGap`

Type
float

Description
Molecular Orbital Info: the smallest HOMO-LUMO gap irrespective of spin (i.e. min(LUMO) - max(HOMO)).

Unit
hartree

`AMSResults%StressTensor`

Type
float_array

Description
The clamped-ion stress tensor in Cartesian notation.

Unit
hartree/bohr^nLatticeVectors

Shape
[:, :]

`AMSResults%YoungModulus`

Type
float

Description
The Young modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

6.2.2 BZcell(primitive cell)

KF Section: BZcell(primitive cell)
Content: The Brillouin zone of the primitive cell.

`BZcell(primitive cell)%boundaries`

Type
float_array

Description
Normal vectors for the boundaries.

Shape
[ndim, nboundaries]

`BZcell(primitive cell)%distances`

Type
float_array

Description
Distance to the boundaries.

Shape
[nboundaries]

`BZcell(primitive cell)%idVerticesPerBound`

Type
int_array

Description
The indices of the vertices per bound.

Shape
[nvertices, nboundaries]

`BZcell(primitive cell)%latticeVectors`

Type
float_array

Description
The lattice vectors.

Shape
[3, :]

`BZcell(primitive cell)%nboundaries`

Type
int

Description
The nr. of boundaries for the cell.

`BZcell(primitive cell)%ndim`

Type
int

Description
The nr. of lattice vectors spanning the Wigner-Seitz cell.

`BZcell(primitive cell)%numVerticesPerBound`

Type
int_array

Description
The nr. of vertices per bound.

Shape
[nboundaries]

`BZcell(primitive cell)%nvertices`

Type
int

Description
The nr. of vertices of the cell.

`BZcell(primitive cell)%vertices`

Type
float_array

Description
The vertices of the bounds.

Unit
a.u.

Shape
[ndim, nvertices]

6.2.3 DOS_Phonons

KF Section: DOS_Phonons
Content: Phonon Density of States

`DOS_Phonons%DeltaE`

Type
float

Description
The energy difference between sampled DOS energies. When there is no DOS at all a certain energy range can be skipped.

Unit
hartree

`DOS_Phonons%Energies`

Type
float_array

Description
The energies at which the DOS is sampled.

Unit
hartree

Shape
[nEnergies]

`DOS_Phonons%Fermi Energy`

Type
float

Description
The fermi energy.

Unit
hartree

`DOS_Phonons%IntegrateDeltaE`

Type
bool

Description
If enabled it means that the DOS is integrated over intervals of DeltaE. Sharp delta function like peaks cannot be missed this way.

`DOS_Phonons%nEnergies`

Type
int

Description
The nr. of energies to use to sample the DOS.

`DOS_Phonons%nSpin`

Type
int

Description
The number of spin components for the DOS.

Possible values
[1, 2]

`DOS_Phonons%Total DOS`

Type
float_array

Description
The total DOS.

Shape
[nEnergies, nSpin]

6.2.4 General

KF Section: General

Content: General information about the MOPAC calculation.

`General%account`

Type
string

Description
Name of the account from the license

`General%engine input`

Type
string

Description
The text input of the engine.

`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%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%version`

Type
int

Description
Version number?

6.2.5 KFDefinitions

KF Section: KFDefinitions

Content: The definitions of the data on this file

`KFDefinitions%json`

Type
string

Description
The definitions of the data on this file in json.

6.2.6 kspace(primitive cell)

KF Section: kspace(primitive cell)

Content: should not be here!!!

`kspace(primitive cell)%avec`

Type
float_array

Description
The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit
bohr

Shape
[3, :]

`kspace(primitive cell)%bvec`

Type
float_array

Description
The inverse lattice stored as a 3x3 matrix. Only the ndimk,ndimk part has meaning.

Unit
1/bohr

Shape
[ndim, ndim]

`kspace(primitive cell)%kt`

Type
int

Description
The total number of k-points used by the k-space to sample the unique wedge of the Brillouin zone.

`kspace(primitive cell)%kuniqu`

Type
int

Description
The number of symmetry unique k-points where an explicit diagonalization is needed. Smaller or equal to kt.

`kspace(primitive cell)%ndim`

Type
int

Description
The nr. of lattice vectors.

`kspace(primitive cell)%ndimk`

Type
int

Description
The nr. of dimensions used in the k-space integration.

`kspace(primitive cell)%xyzpt`

Type
float_array

Description
The coordinates of the k-points.

Unit
1/bohr

Shape
[ndimk, kt]

6.2.7 Low Frequency Correction

KF Section: Low Frequency Correction
Content: Configuration for the Head-Gordon Dampener-powered Free Rotor Interpolation.

`Low Frequency Correction%Alpha`

Type
float

Description
Exponent term for the Head-Gordon dampener.

`Low Frequency Correction%Frequency`

Type
float

Description
Frequency around which interpolation happens, in 1/cm.

`Low Frequency Correction%Moment of Inertia`

Type
float

Description
Used to make sure frequencies of less than ca. 1 1/cm don't overestimate entropy, in kg m^2.

6.2.8 Mobile Block Hessian

KF Section: Mobile Block Hessian

Content: Mobile Block Hessian.

`Mobile Block Hessian%Coordinates Internal`

Type
float_array

Description
?

`Mobile Block Hessian%Free Atom Indexes Input`

Type
int_array

Description
?

`Mobile Block Hessian%Frequencies in atomic units`

Type
float_array

Description
?

`Mobile Block Hessian%Frequencies in wavenumbers`

Type
float_array

Description
?

`Mobile Block Hessian%Input Cartesian Normal Modes`

Type
float_array

Description
?

`Mobile Block Hessian%Input Indexes of Block #`

Type
int_array

Description
?

`Mobile Block Hessian%Intensities in km/mol`

Type
float_array

Description
?

`Mobile Block Hessian%MBH Curvatures`

Type
float_array

Description
?

`Mobile Block Hessian%Number of Blocks`

Type
int

Description
Number of blocks.

`Mobile Block Hessian%Sizes of Blocks`

Type
int_array

Description
Sizes of the blocks.

Shape
[Number of Blocks]

6.2.9 Molecule

KF Section: Molecule

Content: The input molecule of the calculation.

`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
UNUSED IN AMS>=2026. 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
UNUSED IN AMS>=2026. 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
UNUSED IN AMS>=2026. 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

6.2.10 MoleculeSuperCell

KF Section: MoleculeSuperCell
Content: The system used for the numerical phonon super cell calculation.

`MoleculeSuperCell%AtomicNumbers`

Type
int_array

Description
Atomic number ‘Z’ of the atoms in the system

Shape
[nAtoms]

`MoleculeSuperCell%AtomMasses`

Type
float_array

Description
Masses of the atoms

Unit
a.u.

Values range
[0, ‘\infinity’]

Shape
[nAtoms]

`MoleculeSuperCell%AtomSymbols`

Type
string

Description
The atom's symbols (e.g. 'C' for carbon)

Shape
[nAtoms]

`MoleculeSuperCell%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

`MoleculeSuperCell%Charge`

Type
float

Description
Net charge of the system

Unit
e

`MoleculeSuperCell%Coords`

Type
float_array

Description
Coordinates of the nuclei (x,y,z)

Unit
bohr

Shape
[3, nAtoms]

`MoleculeSuperCell%eeAttachTo`

Type
int_array

Description
UNUSED IN AMS>=2026. A multipole may be attached to an atom. This influences the energy gradient.

`MoleculeSuperCell%eeChargeWidth`

Type
float

Description
If charge broadening was used for external charges, this represents the width of the charge distribution.

`MoleculeSuperCell%eeEField`

Type
float_array

Description
The external homogeneous electric field.

Unit
hartree/(e*bohr)

Shape
[3]

`MoleculeSuperCell%eeLatticeVectors`

Type
float_array

Description
UNUSED IN AMS>=2026. The lattice vectors used for the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNLatticeVectors]

`MoleculeSuperCell%eeMulti`

Type
float_array

Description
The values of the external point- or multipole- charges.

Unit
a.u.

Shape
[eeNZlm, eeNMulti]

`MoleculeSuperCell%eeNLatticeVectors`

Type
int

Description
UNUSED IN AMS>=2026. The number of lattice vectors for the external point- or multipole- charges.

`MoleculeSuperCell%eeNMulti`

Type
int

Description
The number of external point- or multipole- charges.

`MoleculeSuperCell%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).

`MoleculeSuperCell%eeUseChargeBroadening`

Type
bool

Description
Whether or not the external charges are point-like or broadened.

`MoleculeSuperCell%eeXYZ`

Type
float_array

Description
The position of the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNMulti]

`MoleculeSuperCell%EngineAtomicInfo`

Type
string_fixed_length

Description
Atom-wise info possibly used by the engine.

`MoleculeSuperCell%fromAtoms`

Type
int_array

Description
Index of the first atom in a bond. See the bondOrders array

`MoleculeSuperCell%latticeDisplacements`

Type
int_array

Description
The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

`MoleculeSuperCell%LatticeVectors`

Type
float_array

Description
Lattice vectors

Unit
bohr

Shape
[3, nLatticeVectors]

`MoleculeSuperCell%nAtoms`

Type
int

Description
The number of atoms in the system

`MoleculeSuperCell%nAtomsTypes`

Type
int

Description
The number different of atoms types

`MoleculeSuperCell%nLatticeVectors`

Type
int

Description
Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values
[0, 1, 2, 3]

`MoleculeSuperCell%toAtoms`

Type
int_array

Description
Index of the second atom in a bond. See the bondOrders array

6.2.11 phonon_curves

KF Section: phonon_curves
Content: Phonon dispersion curves.

`phonon_curves%brav_type`

Type
string

Description
Type of the lattice.

`phonon_curves%Edge_#_bands`

Type
float_array

Description
The band energies

Shape
[nBands, nSpin, :]

`phonon_curves%Edge_#_direction`

Type
float_array

Description
Direction vector.

Shape
[nDimK]

`phonon_curves%Edge_#_kPoints`

Type
float_array

Description
Coordinates for points along the edge.

Shape
[nDimK, :]

`phonon_curves%Edge_#_labels`

Type
lchar_string_array

Description
Labels for begin and end point of the edge.

Shape
[2]

`phonon_curves%Edge_#_lGamma`

Type
bool

Description
Is gamma point?

`phonon_curves%Edge_#_nKPoints`

Type
int

Description
The nr. of k points along the edge.

`phonon_curves%Edge_#_vertices`

Type
float_array

Description
Begin and end point of the edge.

Shape
[nDimK, 2]

`phonon_curves%Edge_#_xFor1DPlotting`

Type
float_array

Description
x Coordinate for points along the edge.

Shape
[:]

phonon_curves%indexLowestBand

Type
int

Description
?

phonon_curves%nBands

Type
int

Description
Number of bands.

phonon_curves%nBas

Type
int

Description
Number of basis functions.

phonon_curves%nDimK

Type
int

Description
Dimension of the reciprocal space.

phonon_curves%nEdges

Type
int

Description
The number of edges. An edge is a line-segment through k-space. It has a begin and end point and possibly points in between.

phonon_curves%nEdgesInPath

Type
int

Description
A path is built up from a number of edges.

phonon_curves%nSpin

Type
int

Description
Number of spin components.

Possible values
[1, 2]

phonon_curves%path

Type
int_array

Description
If the (edge) index is negative it means that the vertices of the edge abs(index) are swapped e.g. path = (1,2,3,0,-3,-2,-1) goes though edges 1,2,3, then there's a jump, and then it goes back.

Shape
[nEdgesInPath]

phonon_curves%path_source

Type
string

Description
Source or program used to generate the path.

Possible values
['input', 'kpath', 'seekpath']

phonon_curves%path_type

Type
string

Description
?

6.2.12 Phonons

KF Section: Phonons

Content: Information on the numerical phonons (super cell) setup. NB: the reciprocal cell of the super cell is smaller than the reciprocal primitive cell.

Phonons%Modes

Type
float_array

Description
The normal modes with the translational symmetry of the super cell.

Shape
[3, nAtoms, 3, NumAtomsPrim, nK]

Phonons%nAtoms

Type
int

Description
Number of atoms in the super cell.

Phonons%nK

Type
int

Description
Number of gamma-points (of the super cell) that fit into the primitive reciprocal cell.

Phonons%NumAtomsPrim

Type
int

Description
Number of atoms in the primitive cell.

`Phonons%xyzKSuper`

Type
float_array

Description
The coordinates of the gamma points that fit into the primitive reciprocal cell.

Shape
[3, nK]

6.2.13 Plot

KF Section: Plot

Content: Generic section to store x-y plots.

`Plot%numPlots`

Type
int

Description
Number of plots.

`Plot%NumPoints(#)`

Type
int

Description
Number of x points for plot #.

`Plot%NumYSeries(#)`

Type
int

Description
Number of y series for plot #.

`Plot%Title(#)`

Type
string

Description
Title of plot #

`Plot%XLabel(#)`

Type
string

Description
X label for plot #.

`Plot%XUnit(#)`

Type
string

Description
X unit for plot #.

`Plot%XValues(#)`

Type
float_array

Description
X values for plot #.

Shape
[:]

`Plot%YLabel(#)`

Type
string

Description
Y label for plot #.

`Plot%YUnit(#)`

Type
string

Description
Y unit for plot #.

`Plot%YValues(#)`

Type
float_array

Description
Y values for plot #. Array has extra column NumYSeries.

6.2.14 Properties

KF Section: Properties
Content: Properties computed by the MOPAC engine.

`Properties%IonizationPotential`

Type
float

Description
The ionization potential.

Unit
hartree

`Properties%pKaAtomIndices`

Type
int_array

Description
The indices of the atoms for which the pKa values were computed. The corresponding pKa values are stored in the variable ‘pKaValues’.

`Properties%pKaValues`

Type
float_array

Description
The pKa values (i.e. the negative base-10 logarithm of the acid dissociation constant). pKa values are only computed for hydrogen atoms connected to oxygen atoms. See the variable ‘pKaAtomIndices’ to know which atoms the pKa values stored here correspond to.

`Properties%StatPolTensorDipole`

Type
float_array

Description
The static polarizability tensor (x,y,z) computed from by differentiating the dipole moment.

Unit
a.u.

Shape
[3, 3]

`Properties%StatPolTensorHeat`

Type
float_array

Description
The static polarizability tensor (x,y,z) computed from the heat of formation.

Unit
a.u.

Shape
[3, 3]

6.2.15 SCF

KF Section: SCF
Content: Results from the SCF of MOPAC. The MOPAC orbital energies and occupations are stored here.

`SCF%Eigenvalues_A`

Type
float_array

Description
In case of spin-restricted calculations: orbital energies. In case of spin-unrestricted calculations: the orbital energies for the spin-alpha orbitals.

Unit
hartree

`SCF%Eigenvalues_B`

Type
float_array

Description
The orbital energies for the spin-beta orbitals. Only present in case of spin-unrestricted calculations.

Unit
hartree

`SCF%Labels_A`

Type
lchar_string_array

Description
In case of spin-restricted calculations: the symmetry labels for the orbitals. In case of spin-unrestricted calculations: the symmetry labels for the spin-alpha orbitals.

`SCF%Labels_B`

Type
lchar_string_array

Description
The symmetry labels for the spin-beta orbitals. Only present in case of spin-unrestricted calculations.

`SCF%Occupations_A`

Type
float_array

Description
In case of spin-restricted calculations: the occupations of the orbitals (the value can be between 0 and 2). In case of spin-unrestricted calculations: the occupations for the spin-alpha orbitals (the value can be between 0 and 1).

`SCF%Occupations_B`

Type
float_array

Description
The occupations for the spin-beta orbitals (the value can be between 0 and 1). Only present in case of spin-unrestricted calculations.

6.2.16 Thermodynamics

KF Section: Thermodynamics
Content: Thermodynamic properties computed from normal modes.

`Thermodynamics%Enthalpy`

Type
float_array

Description
Enthalpy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Entropy rotational`

Type
float_array

Description
Rotational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Entropy total`

Type
float_array

Description
Total entropy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Entropy translational`

Type
float_array

Description
Translational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Entropy vibrational`

Type
float_array

Description
Vibrational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Gibbs free Energy`

Type
float_array

Description
Gibbs free energy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Heat Capacity rotational`

Type
float_array

Description
Rotational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Heat Capacity total`

Type
float_array

Description
Total heat capacity.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Heat Capacity translational`

Type
float_array

Description
Translational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Heat Capacity vibrational`

Type
float_array

Description
Vibrational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Inertia direction vectors`

Type
float_array

Description
Inertia direction vectors.

Shape
[3, 3]

`Thermodynamics%Internal Energy rotational`

Type
float_array

Description
Rotational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Internal Energy total`

Type
float_array

Description
Total internal energy.

Unit
a.u.

`Thermodynamics%Internal Energy translational`

Type
float_array

Description
Translational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Internal Energy vibrational`

Type
float_array

Description
Vibrational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%lowFreqEntropy`

Type
float_array

Description
Entropy contributions from low frequencies (see ‘lowFrequencies’).

Unit
a.u.

Shape
[nLowFrequencies]

`Thermodynamics%lowFreqHeatCapacity`

Type
float_array

Description
Heat capacity contributions from low frequencies (see ‘lowFrequencies’).

Unit
a.u.

Shape
[nLowFrequencies]

`Thermodynamics%lowFreqInternalEnergy`

Type
float_array

Description
Internal energy contributions from low frequencies (see ‘lowFrequencies’).

Unit
a.u.

Shape
[nLowFrequencies]

`Thermodynamics%lowFrequencies`

Type
float_array

Description
Frequencies below 20 cm^-1 (contributions from frequencies below 20 cm^-1 are not included in vibrational sums, and are saved separately to ‘lowFreqEntropy’, ‘lowFreqInternalEnergy’ and ‘lowFreqInternalEnergy’). Note: this does not apply to RRHO-corrected quantities.

Unit
cm^-1

Shape
[nLowFrequencies]

`Thermodynamics%Moments of inertia`

Type
float_array

Description
Moments of inertia.

Unit
a.u.

Shape
[3]

`Thermodynamics%nLowFrequencies`

Type
int

Description
Number of elements in the array lowFrequencies.

`Thermodynamics%nTemperatures`

Type
int

Description
Number of temperatures.

`Thermodynamics%Pressure`

Type
float

Description
Pressure used.

Unit
atm

`Thermodynamics%RRHOCorrectedHeatCapacity`

Type
float_array

Description
Heat capacity T*S corrected using the ‘low vibrational frequency free rotor interpolation corrections’.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%RRHOCorrectedInternalEnergy`

Type
float_array

Description
Internal energy T*S corrected using the ‘low vibrational frequency free rotor interpolation corrections’.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%RRHOCorrectedTS`

Type
float_array

Description
T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Temperature`

Type
float_array

Description
List of temperatures at which properties are calculated.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%TS`

Type
float_array

Description
T*S, i.e. temperature times entropy.

Unit
a.u.

Shape
[nTemperatures]

6.2.17 Vibrations

KF Section: Vibrations
Content: Information related to molecular vibrations.

`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/bohr^2

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%IrReps`

Type
lchar_string_array

Description
Symmetry symbol of the normal mode.

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%RamanDepolRatioNat`

Type
float_array

Description
Raman depol ratio (nat).

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%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%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%VROACIDDePolarized`

Type
float_array

Description
VROA Circular Intensity Differential: Depolarized scattering.

Unit
10¯3

Shape
[nNormalModes]

`Vibrations%VROACIDForward`

Type
float_array

Description
VROA Circular Intensity Differential: Forward scattering.

Unit
10¯3

Shape
[nNormalModes]

`Vibrations%VROACIDPolarized`

Type
float_array

Description
VROA Circular Intensity Differential: Polarized scattering.

Unit
10¯3

Shape
[nNormalModes]

`Vibrations%VROADeltaBackward`

Type
float_array

Description
VROA Intensity: Backward scattering.

Unit
10¯3 A^4/amu

Shape
[nNormalModes]

`Vibrations%VROADeltaDePolarized`

Type
float_array

Description
VROA Intensity: Depolarized scattering.

Unit
10¯3 A^4/amu

Shape
[nNormalModes]

`Vibrations%VROADeltaForward`

Type
float_array

Description
VROA Intensity: Forward scattering.

Unit
10¯3 A^4/amu

Shape
[nNormalModes]

`Vibrations%VROADeltaPolarized`

Type
float_array

Description
VROA Intensity: Polarized scattering.

Unit
10¯3 A^4/amu

Shape
[nNormalModes]

`Vibrations%ZeroPointEnergy`

Type
float

Description
Vibrational zero-point energy.

Unit
hartree

CHAPTER

SEVEN

FAQ

7.1 What's the difference between MOPAC in AMS2019 and in previous versions (AMS2018)?

In AMS2019 we have made a new MOPAC library, which is fully integrated as an Engine with the AMS driver and our GUI. This new MOPAC works much faster as a pre-optimizer and with any AMS driver functionality. This version of MOPAC is based on the original MOPAC code of Dr. Stewart and contains much but not all of the functionality. In AMS2019, MOPAC is included with the DFTB module. With AMS2018 it is possible to use the openmopac binaries, with all original functionality, and MOPAC could also be used as an external engine. The MOPAC binaries in AMS2018 were free of charge for academic groups with any other license. In AMS2018 there is no support for the new integrated MOPAC Engine.

Older ADF Modeling License suite licenses could also contain the external MOPAC binary and corresponding GUI support.

7.2 How to use keywords from the MOPAC manual?

In ADFInput in the Details → Run script tab, you can change the MOPAC input file before submitting the calculation. A list of MOPAC keywords (from http://openmopac.net/manual/allkeys.html):

7.2.1 Keywords used in MOPAC2012

7.3 Which properties from MOPAC can be visualized with the GUI?

Besides facilitating building molecules, our GUI can also visualize the progress of geometry optimizations, MOPAC charges, and IR modes. Geometry optimization can also be done interactively, e.g. to pre-optimize your molecule or periodic system.

INDEX

A

`AMS driver`, 1, 3
`Atomic charges`, 3
`Atoms`, 3

B

`Bond orders`, 3

C

`Charge`, 3
`Coordinates`, 3

D

`Dipole Gradients`, 3
`Dipole Moment`, 3

E

`Elastic tensor`, 3
`examples`, 13

G

`GCMC` (*Grand Canonical Monte Carlo*), 3
`Geometry`, 3
`Geometry Optimization`, 3

H

`Hessian`, 3
`Homogeneous Electric Field`, 3

I

`Infrared (IR) spectra / Normal Modes`, 3
`IRC` (*Intrinsic Reaction Coordinate*), 3
`Isotopes`, 3

L

`Lattice Vectors`, 3
`Linear Transit`, 3

M

`Molecular Dynamics`, 3
`Molecules detection`, 3

N

`NEB` (*Nudged Elastic Band*), 3
`Nuclear Gradients / Forces`, 3

P

`PES`, 3
`PES point character`, 3
`PESScan` (*Potential Energy Surface Scan*), 3
`Phonons`, 3
`Potential Energy Surface`, 3

S

`Single Point`, 3
`Stress tensor`, 3

T

`Task`, 3
`Thermodynamic properties`, 3
`Transition State Search`, 3

V

`VCD` (*Vibrational Circular Dichroism*), 3
`Vibrational Analysis`, 3

X

`xyz`, 3