ASE Manual

Amsterdam Modeling Suite 2025.1

www.scm.com

Mar 24, 2025

CONTENTS

CHAPTER

ONE

GENERAL, QUICKSTART, USAGE

The **ASE engine** (this manual) is the interface between any ASE calculator and the AMS Driver (see *Quickstart guide* (page 2) or *Examples* (page 9)).

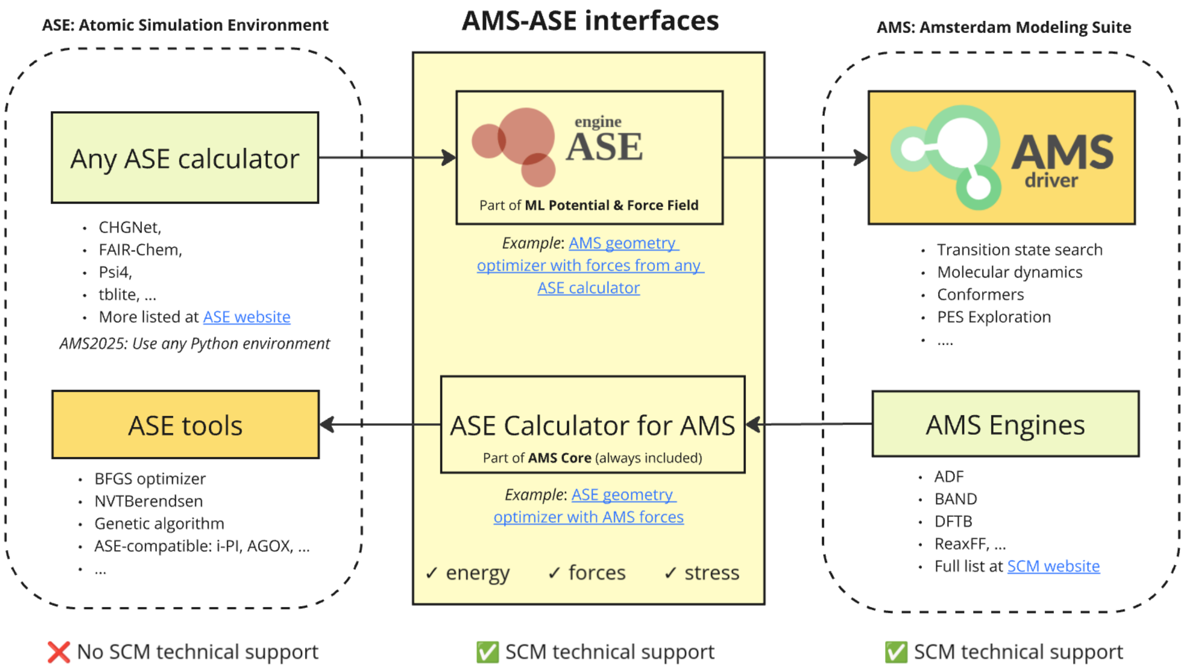
The ASE calculator may come from

- the ASE package (https://wiki.fysik.dtu.dk/ase/ase/calculators/calculators.html#supported-calculators) (included with AMS),
- other external sources (see *Examples* (page 9)), or
- you may implement your own.

New in AMS2025: The ASE calculator can be installed into any *Python environment* (page 33) on the system.

The ASE Calculator for AMS (AMSCalculator) is described in a different manual.

More information about ASE can be found in ref.[1] or on the ASE website (https://wiki.fysik.dtu.dk/ase/).



[1] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. Bergmann Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K. W. Jacobsen. J. Phys.: Condens. Matter Vol. 29 (2007) 273002 https://iopscience.iop.org/article/10.1088/1361-648X/aa680e

1.1 Quickstart guide

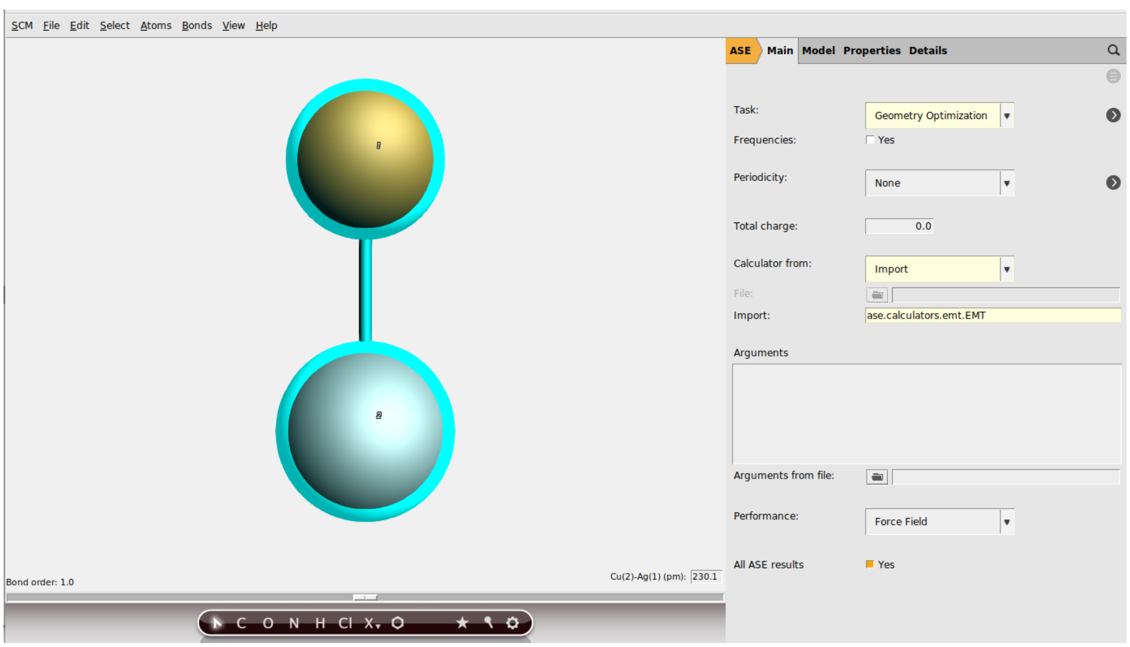
See the below quickstart guide, or have a look at the *example calculators* (page 9).

1.1.1 Quickstart with GUI

- Start AMSinput and switch to the ASE engine in the yellow dropdown.
- In the `Calculator from` drop-down, select `Import`
- In the `Import` field, type `ase.calculators.emt.EMT` (more information: ASE implementation of EMT (https://databases.fysik.dtu.dk/ase/ase/calculators/emt.html))
- Copy and paste the following atom coordinates into AMSinput

```
Ag 0. 0. 0.
Cu 0. 0. 2.0
```

- Run the calculation
- Open the trajectory in AMSmovie, or update the geometry in AMSinput



1.1.2 Quickstart with Python

See the PLAMS example: Engine ASE: AMS geometry optimizer with forces from any ASE Calculator

1.1.3 Quickstart with command-line

Run the below example:

```
$AMSBIN/ams <<eor
Task SinglePoint
Properties
    Gradients Yes
End

System
    Atoms
        Ag 0. 0. 0.
        Cu 0. 0. 2.0
    End
End

Engine ASE
   Type Import
   Import ase.calculators.emt.EMT
EndEngine
eor
```

1.2 Engine input

1.2.1 Calculator from Import

Example without arguments

```
Engine ASE
   Type Import
   Import ase.calculators.emt.EMT
EndEngine
```

Example with arguments

```
Engine ASE
    Type Import
    Import ase.calculators.harmonic.SpringCalculator
    Arguments
      ideal_positions=[[0.2,0.0,0.0],[0.8,0.0,0.0]]
      k=2.0
    End
EndEngine
```

1.2.2 Calculator from Python file

A Calculator is selected by providing a Python file through `File` and should contain a function or class named *get_calculator* which returns an initialized ASE Calculator.

Example without arguments

The AMS input file contains:

```
Engine ASE
   Type File
   File /path/to/pythonfile.py
EndEngine
```

With *pythonfile.py* containing e.g.

```
from ase.calculators.harmonic import SpringCalculator
def get_calculator():
   return SpringCalculator(ideal_positions=[[0.2,0.0,0.0],[0.8,0.0,0.0]], k=2.0)
```

See also the *Examples* (page 9).

Example with arguments

```
Engine ASE
   Type File
   File /path/to/pythonfile.py
   Arguments
      ideal_positions=[[0.2,0.0,0.0],[0.8,0.0,0.0]]
      k=2.0
   End
EndEngine
```

With *pythonfile.py* containing e.g.

```
from ase.calculators.harmonic import SpringCalculator
def get_calculator(ideal_positions, k):
   return SpringCalculator(ideal_positions=ideal_positions, k=k)
```

See also the *Examples* (page 9).

1.2.3 Specifying arguments for the Calculator

Arguments can be specified either with the `Arguments` argument as above, or with `ArgumentsFromFile`.

`ArgumentsFromFile` is either a yaml file with the .yml extension or a python file with the .py extension.

1.2.4 Obtaining results from the Calculator

When available, the *energy*, *forces* and *stress* are always obtained and are fully integrated in AMS. If a Calculator holds any additional results in its results dictionary, then by default they are stored in the *Other* section of ase.rkf, but without any unit conversions. If additional results are undesired, they can be turned off through `AllASEResults` and specific results can be requested by setting `Results`.

Example. The AMS input file contains:

```
Engine ASE
   Type File
   File custom/calculator.py
   AllASEResults no
   Results specific_result1
   Results specific_result2
EndEngine
```

See also:

Tutorial: 10 Ways to Get the Energy and Other Properties

1.3 Specifying the capabilities of a Calculator

The AMS driver can make several decisions based on what an engine is able to do. Use the *scm.amspipe.AMSExternalCapabilities* class:

```
from scm.amspipe import AMSExternalCapabilities
from ase.calculators.calculator import Calculator

class MyDipoleCalculator(Calculator):
    implemented_properties = ['energy', 'forces', 'dipole']
    ...

def get_calculator():
    calc = MyDipoleCalculator()
    capabilities = AMSExternalCapabilities(charges=True) #indicate the calculator can␣
↪calculate atomic charges.
    capabilities.apply_implemented_properties(calc.implemented_properties) #will find␣
↪that 'dipole' is supported so AMS will e.g. automatically compute intensities when␣
↪doing a normal mode calculation
    calc.ams_capabilities = capabilities
    return calc
```

1.4 Troubleshooting

- If the required Calculator needs advanced setup or input arguments, it is usually more convenient to use the python input style and handle such details in there.
- The *Arguments* block is stripped of any indentation and comments ("!", "#" and "::"). If this is not desirable, use *ArgumentsFromFile* or python input style instead.
- If the Calculator accepts any arguments related to files, make sure to provide **absolute paths** and not relative paths. The Calculator does not run in the same directory as AMS to avoid conflicts.

- If an AMS calculation fails to run with the ASE calculator, make sure you have specified correctly what capabilities the Calculator has.
- If you get an error like `ERROR: Worker failed to start`, you can usually find more information about the error in the **standard output file** if you scroll up a bit. For example, if you use an *external Python environment* (page 33) and get an error like `ModuleNotFoundError: No module named 'scm'`, then make sure to install `scm.amspipe` and `scm.external_engines` into the other Python environment.

1.5 Parametrization with ParAMS

You can use ParAMS to fit the parameters of your ASE calculator.

See the ASE calculator parametrization example.

1.6 Support

SCM does not provide support for any ASE calculators.

1.7 Licensing

In AMS2023, the ASE engine is part of the product "ML Potentials & Classical Force Fields".

1.8 Changelog

1.8.1 AMS2025.1

- ASE calculators no longer need to be installed into amspython environment, but can be used from any *Python environment* (page 33).
- The default `Performance` setting changed from `ForceField` to `DFTB`.
- The AMSinput GUI module can now pick up docstrings, arguments, and default values from calculator.py files.

1.8.2 AMS2024.1

- It is now possible to indicate to AMS *what properties the Calculator can compute and what kind of system is supported* (page 5) (e.g. if the Calculator can support periodic systemsand if it can, whether those systems can additionally be charged.).

1.8.3 AMS2023.1

- The ASE engine is new in AMS2023.

1.9 References

CHAPTER

TWO

EXAMPLES

See the *Quickstart guide* (page 2) for a fast example with the ASE EMT Calculator.

Please note the following important points:

1. **External Software**: The external programs and packages referenced in this documentation are not developed, maintained, or endorsed by SCM.
2. **Installation and Use**: Users are responsible for the installation, configuration, and management of external software and tools. By following these instructions, you will be installing third-party software on your system. SCM has no control over the content, functionality, or updates of such software. Before installation, read and agree to the external software's license terms.
3. **Liability**: SCM is not responsible for any issues arising from the installation, configuration, or use of external software, including but not limited to system instability, data loss, or security vulnerabilities. See also the SCM license terms.
4. **Support**: For questions or issues related to the external software, users should contact the respective developers or refer to their official documentation. See also *Troubleshooting* (page 5).

This page contains tips on how to couple some ASE calculators to AMS. It is provided for information purposes only. **The information may become outdated** and is not guaranteed to work for you on your system.

> **Warning:** When you manually install packages into the AMS Python environment, you may break the SCM-supported ML Potential packages, for example by installing incompatible versions of dependencies. If this happens, it is easiest to remove the AMS Python virtual environment completely and reinstall the ML Potential packages from the package manager.
>
> **We recommend that you install packages into a separate** *Conda environment* (page 33).

2.1 Overview of external methods/programs

Method/program	AMS Engine	Type	Website
AIMNet2 (page 11)	ML Potential	ML Potential	Github (https://github.com/isayevlab/AIMNet2)
ALIGNN-FF (page 11)	ASE	ML potential	Github (https://github.com/usnistgov/alignn)
ANI (TorchANI) (page 12)	ML potential	ML potential	Github (https://github.com/aiqm/torchani)
CHGNet (page 12)	ASE	ML potential	Github (https://github.com/CederGroupHub/chgnet)
CP2K (page 13)	ASE	DFT	cp2k.org (https://www.cp2k.org/)
Custom (page 14)	ASE	Custom	
DeePMD-kit (page 15)	ASE	ML potential	Github (https://github.com/deepmodeling/deepmd-kit/tree/master)
DFTpy (page 16)	ASE	orbital-free DFT	Gitlab (https://gitlab.com/pavanello-research-group/dftpy)
EMT (page 18)	ASE	Force field	
FAIR-Chem (page 18)	ASE	ML Potential	Github (https://github.com/FAIR-Chem/fairchem)
GPAW (page 20)	ASE	DFT	website (https://wiki.fysik.dtu.dk/gpaw/)
MACE (page 22)	ASE	ML Potential	Github (https://github.com/ACEsuit/mace)
MatGL (M3GNet) (page 23)	ASE	ML potential	Github (https://github.com/materialsvirtuallab/matgl)
MatterSim (page 24)	ASE	ML potential	Github (https://github.com/microsoft/mattersim/tree/main)
M3GNet (page 22)	ML potential	ML potential	Github (https://github.com/materialsvirtuallab/m3gnet)
NequIP (page 25)	ML potential	ML potential	Github (https://github.com/mir-group/nequip)
Open Catalyst Project (page 29)	ASE	ML potential	opencatalystproject.org (https://opencatalystproject.org/)
ORB (page 26)	ASE	ML Potential	Github (https://github.com/orbital-materials/orb-models)
psi4 (page 27)	ASE	DFT/ab initio	Website (https://psicode.org)
PySCF (page 28)	ASE	DFT/ab initio	Website (https://pyscf.org)
Quantum ESPRESSO (page 29)	Quantum ESPRESSO	DFT	quantum-espresso.org (https://www.quantum-espresso.org/)
sGDML (page 29)	ML potential	ML potential	Github (https://github.com/stefanch/sGDML)
SO3LR (page 29)	ASE	ML potential	Github (https://github.com/general-molecular-simulations/so3lr)
tblite (page 30)	ASE	semi-empirical	Github (https://github.com/tblite/tblite)
VASP (page 31)	External	DFT	vasp.at (https://www.vasp.at)
xTB (GFN2-xTB) (page 31)	ASE	semi-empirical	Github (https://github.com/grimme-lab/xtb)

2.2 AIMNet2

See the ML Potential documentation.

2.3 ALIGNN-FF

Tested with: AMS2025.101, Ubuntu Linux 24.04, Jan 13 2024

Install alignn in a *conda environment* (page 33).

```
conda create -n alignn-env
conda install -c conda-forge -n alignn-env \
  dgl=2.1.0 pytorch torchvision torchaudio pytorch-cuda -c pytorch -c nvidia
conda run -n alignn-env \
  python -m pip install \
  "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of Si bulk, then your conda environment has been set up correctly:

```
conda run -n alignn-env \
  python $AMSHOME/scripting/scm/external_engines/backends/_alignn/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf alignn_test.results

AMS_JOBNAME=alignn_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        Si -0.67875 -0.67875 -0.67875
        Si 0.77875 0.67875 0.67875
    End
    Lattice
        0.0 2.715 2.715
        2.715 0.0 2.715
        2.715 2.715 0.0
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_alignn/calculator.py
    Arguments
        path = None
    End
    Python
        Type Conda
        Conda alignn-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

2.4 ANI (TorchANI)

See the ML Potential documentation.

2.5 CHGNet

Tested with: AMS2025.101, Ubuntu Linux 22.04, Nov 11 2024

Install CHGNet into a separate *conda environment* (page 33):

```
conda create -n chgnet-env
conda install -c conda-forge -n chgnet-env "python<3.13" ase
conda run -n chgnet-env \
  python -m pip install \
  chgnet "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n chgnet-env \
   python $AMSHOME/scripting/scm/external_engines/backends/_chgnet/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf chgnet_test.results

AMS_JOBNAME=chgnet_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_chgnet/calculator.py
    Arguments
        use_device = "cpu"
    End
    Python
        Type Conda
        Conda chgnet-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

If you have a CUDA-enabled GPU, you can set `use_device = "cuda"` to run on the GPU instead. Or remove the option completely to use the default device.

See the CHGNet documentation and code for details.

2.6 CP2K

Tested with: AMS2023.101, Ubuntu Linux 22.04, July 5 2023

The below works for **single-node** calculation but fails for multinode (MPI parallelization).

- Install CP2K. In a terminal: `sudo apt install cp2k`
- Run AMS with the CP2K ASE calculator `cp2k_ams.run`:

Listing 2.1: cp2k_ams.run

```
#!/bin/sh

export SCM_DISABLE_MPI=1

# set OMP_NUM_THREADS to the appropriate number below

$AMSBIN/ams -n 1 <<eor

Task GeometryOptimization

System
  Atoms
              O       2.8115000409       2.5498605363       2.0000000000
              H       2.0000000000       2.0000000000       2.0000000000
              H       3.6254485609       2.0005857872       2.0000000000
  End
  Lattice
         5.6254485609     0.0000000000     0.0000000000
         0.0000000000     4.5498605363     0.0000000000
         0.0000000000     0.0000000000     4.0000000000
  End
End

Engine ASE
  Type Import
  Import ase.calculators.cp2k.CP2K
  # see the ASE CP2K documentation for details about the arguments
  Arguments
     command = "OMP_NUM_THREADS=2 cp2k_shell"   # set OMP_NUM_THREADS here
     cutoff = 4000 # eV
     stress_tensor = False   # set stress_tensor here  (defaults to True)
     xc = "PBE"
     inp = """
     &FORCE_EVAL
       &DFT
         &KPOINTS
           SCHEME GAMMA
         &END KPOINTS
         &SCF
           ADDED_MOS 10
           &SMEAR
             METHOD FERMI_DIRAC
             ELECTRONIC_TEMPERATURE [K] 500.0
           &END SMEAR
         &END SCF
       &END DFT
```

(continues on next page)

(continued from previous page)

```
     &END FORCE_EVAL
    """
  End
EndEngine

eor
```

2.7 Custom

Tested with: AMS2025.101, Ubuntu Linux 22.04, Feb 4 2025

This example shows how to set up a custom simple ASE calculator. It always returns zero forces.

Listing 2.2: calculator.py

```
#!/usr/bin/env amspython

from ase.calculators.calculator import Calculator, all_changes
import numpy as np


class ZeroCalculator(Calculator):
    implemented_properties = ["energy", "forces"]

    def calculate(self, atoms=None, properties=["energy"], system_changes=all_
↪changes):
        super().calculate(atoms, properties, system_changes)
        self.results = dict()
        if atoms is not None:
            energy_per_atom = self.parameters.get("energy_per_atom", 0.0)
            self.results = {
                "energy": energy_per_atom * len(atoms),
                "forces": np.zeros((len(atoms), 3)),
            }


def get_calculator(energy_per_atom: float = 0.0):
    """energy_per_atom: float (eV). Always returns zero forces."""
    return ZeroCalculator(energy_per_atom=energy_per_atom)


def example():
    """Simple example for a water molecule.

    Note: this is just an example of the ASE calculator. This section
    has no effect when running through AMS.
    """
    from ase.build import molecule

    atoms = molecule("H2O")
    atoms.calc = get_calculator(energy_per_atom=1.0)

    energy = atoms.get_potential_energy()
    forces = atoms.get_forces()
```

(continues on next page)

(continued from previous page)

```
    print(f"Energy of water molecule: {energy:.3f} eV")
    print(f"Forces (eV/ang): {forces}")


if __name__ == "__main__":
    example()
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf zero_test.results

AMS_JOBNAME=zero_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_zero/calculator.py
    Arguments
        energy_per_atom = 1.0
    End
    Python
        Type amspython
    End
EndEngine
eor
```

2.8 DeePMD-kit

Tested with: AMS2023.104, Ubuntu Linux 22.04, 14 Nov 2023

- Install `deepmd-kit` into the AMS python environment. This will place the `db` binary in a location like `/home/user/.scm/python/AMS2023.1.venv/bin`
- Either train your own model or download one from the AIS Square (https://www.aissquare.com/)
- If you download a model you may need to convert it using `db convert-from`, see the DeePMD-kit documentation (https://docs.deepmodeling.com/projects/deepmd/en/master/troubleshooting/model-compatability.html)

Then specify `Type Import` and specify the path to the model (`.pb`) file in the Arguments:

Listing 2.3: deepmd-kit_ams.run

```
#!/bin/sh
NSCM=1 $AMSBIN/ams <<EOF
system
  Atoms
```

(continues on next page)

(continued from previous page)

```
              O      -0.0008161597       0.3663784285      -0.0000000000
              H      -0.8123162006      -0.1834821079      -0.0000000000
              H       0.8131323603      -0.1828963206       0.0000000000
  End
End

Task GeometryOptimization

Engine ASE
  Arguments

    model = '/absolute/path/to/graph.pb'

  End
  Import deepmd.calculator.DP
  Type Import
EndEngine
EOF
```

2.9 DFTpy

Tested with: AMS2023.101, Ubuntu Linux 22.04, August 3 2023

```
$AMSBIN/amspython -m pip install dftpy
$AMSBIN/amspython -m pip install pylibxc2
git clone https://gitlab.com/pavanello-research-group/local-pseudopotentials
```

Set the environment variable to the path to the pseudopotentials, for example

```
export DFTPY_DATA_PATH=`readlink -f local-pseudopotentials/BLPS/LDA/reci`
```

An ASE Calculator with some settings for Al is defined in `dftpy_calculator.py`:

```
#!/usr/bin/env amspython
import os
from dftpy.config import DefaultOption, OptionFormat
from dftpy.api.api4ase import DFTpyCalculator
from ase.calculators.calculator import Calculator


class MyCalculator(Calculator):
    implemented_properties = ["energy", "forces", "stress"]

    def __init__(self, config, **kwargs):
        Calculator.__init__(self, **kwargs)
        self.dftpy_calculator = DFTpyCalculator(config=config)

    def calculate(self, atoms, properties=None, system_changes=None):
        super().calculate(atoms, properties, system_changes)
        self.results = dict()
        self.results["energy"] = self.dftpy_calculator.get_potential_energy(atoms)
        self.results["forces"] = self.dftpy_calculator.get_forces(atoms)
        self.results["stress"] = self.dftpy_calculator.get_stress(atoms)

```

(continues on next page)

(continued from previous page)

```
def get_calculator():
    config = DefaultOption()
    config["PATH"]["pppath"] = os.environ.get(
        "DFTPY_DATA_PATH",
        "/home/hellstrom/software/local-pseudopotentials/BLPS/LDA/reci",
    )
    config["PP"]["Al"] = "al.lda.recpot"
    config["OPT"]["method"] = "TN"
    config["KEDF"]["kedf"] = "WT"
    config["JOB"]["calctype"] = "Energy Force"
    config = OptionFormat(config)
    calc = MyCalculator(config=config)

    return calc
```

This file is referenced inside the Engine ASE block in the input to AMS:

```
#!/bin/sh
export SCM_DISABLE_MPI=1

$AMSBIN/ams <<EOF
Engine ASE
  File /home/hellstrom/dftpy_calculator.py # change this!
  Type File
EndEngine

MolecularDynamics
  InitialVelocities
    Temperature 1000
    Type Random
  End
  NSteps 20
  Thermostat
    Tau 100.0
    Temperature 1000
    Type NHC
  End
  Barostat
    Type MTK
    Pressure 1.0
    Tau 10000
  End
  TimeStep 1.0
  Trajectory
    SamplingFreq 1
  End
End

Task MolecularDynamics

system
  Atoms
             Al       0.0000000000       0.0000000000       0.0000000000
             Al       0.0000000000       2.1200000000       2.1200000000
             Al       2.1200000000       0.0000000000       2.1200000000
             Al       2.1200000000       2.1200000000       0.0000000000
```

(continues on next page)

(continued from previous page)

```
             Al       0.0000000000       0.0000000000       4.2400000000
             Al       0.0000000000       2.1200000000       6.3600000000
             Al       2.1200000000       0.0000000000       6.3600000000
             Al       2.1200000000       2.1200000000       4.2400000000
             Al       0.0000000000       4.2400000000       0.0000000000
             Al       0.0000000000       6.3600000000       2.1200000000
             Al       2.1200000000       4.2400000000       2.1200000000
             Al       2.1200000000       6.3600000000       0.0000000000
             Al       0.0000000000       4.2400000000       4.2400000000
             Al       0.0000000000       6.3600000000       6.3600000000
             Al       2.1200000000       4.2400000000       6.3600000000
             Al       2.1200000000       6.3600000000       4.2400000000
             Al       4.2400000000       0.0000000000       0.0000000000
             Al       4.2400000000       2.1200000000       2.1200000000
             Al       6.3600000000       0.0000000000       2.1200000000
             Al       6.3600000000       2.1200000000       0.0000000000
             Al       4.2400000000       0.0000000000       4.2400000000
             Al       4.2400000000       2.1200000000       6.3600000000
             Al       6.3600000000       0.0000000000       6.3600000000
             Al       6.3600000000       2.1200000000       4.2400000000
             Al       4.2400000000       4.2400000000       0.0000000000
             Al       4.2400000000       6.3600000000       2.1200000000
             Al       6.3600000000       4.2400000000       2.1200000000
             Al       6.3600000000       6.3600000000       0.0000000000
             Al       4.2400000000       4.2400000000       4.2400000000
             Al       4.2400000000       6.3600000000       6.3600000000
             Al       6.3600000000       4.2400000000       6.3600000000
             Al       6.3600000000       6.3600000000       4.2400000000
  End
  Lattice
         8.4800000000     0.0000000000     0.0000000000
         0.0000000000     8.4800000000     0.0000000000
         0.0000000000     0.0000000000     8.4800000000
  End
End

EOF
```

2.10 EMT

See the *Quickstart guide* (page 2).

2.11 FAIR-Chem

Tested with: AMS2025.101, Linux Mint 22, Nov 12 2024

- FAIR-Chem Github (https://github.com/FAIR-Chem/fairchem/tree/main)
- FAIR-Chem Installation instructions (https://fair-chem.github.io/core/install.html)
- FAIR-Chem Using pretrained models with ASE (https://fair-chem.github.io/core/quickstart.html)

Install FAIR-Chem in a *conda environment* (page 33).

```
# see https://fair-chem.github.io/core/install.html#conda
wget https://raw.githubusercontent.com/FAIR-Chem/fairchem/main/packages/env.gpu.yml
conda env create -f env.gpu.yml

conda run -n fair-chem \
  python -m pip install \
  fairchem-core "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. This will download a model file into $HOME/fairchem_pretrained_models. If the below prints an energy at the end, then your conda environment has been set up correctly:

```
conda run -n fair-chem \
   python $AMSHOME/scripting/scm/external_engines/backends/_fairchem/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf fairchem_test.results

AMS_JOBNAME=fairchem_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        Cu       1.27632774       1.27632774       9.80500000
        Cu       3.82898322       1.27632774       9.80500000
        Cu       6.38163870       1.27632774       9.80500000
        Cu       1.27632774       3.82898322       9.80500000
        Cu       3.82898322       3.82898322       9.80500000
        Cu       6.38163870       3.82898322       9.80500000
        Cu       1.27632774       6.38163870       9.80500000
        Cu       3.82898322       6.38163870       9.80500000
        Cu       6.38163870       6.38163870       9.80500000
        Cu       0.00000000       0.00000000      11.61000000
        Cu       2.55265548       0.00000000      11.61000000
        Cu       5.10531096       0.00000000      11.61000000
        Cu       0.00000000       2.55265548      11.61000000
        Cu       2.55265548       2.55265548      11.61000000
        Cu       5.10531096       2.55265548      11.61000000
        Cu       0.00000000       5.10531096      11.61000000
        Cu       2.55265548       5.10531096      11.61000000
        Cu       5.10531096       5.10531096      11.61000000
        O        1.27632774       0.00000000      14.61000000
        C        1.27632774       0.00000000      13.45966000
    End
    Lattice
        7.66 0.0 0.0
        0.0 7.66 0.0
        0.0 0.0 20.0
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_fairchem/calculator.py
    Arguments
        model_name = "EquiformerV2-31M-S2EF-OC20-All+MD"
        # checkpoint_path = "/path/to/eqV2_31M_omat_mp_salex.pt"  # for checkpoint␣
```

(continues on next page)

(continued from previous page)

```
↪paths
        cpu = True
    End
    Performance DFTB
    Python
        Type Conda
        Conda fair-chem   # this environment must have been set up correctly
    End
EndEngine
eor
```

See the fairchem documentation and code for details.

2.12 GPAW

Tested with: AMS2023.102, Ubuntu Linux 22.04, August 1 2023

GPAW (https://wiki.fysik.dtu.dk/gpaw/) is a planewave density functional theory code.

- Install GPAW (https://wiki.fysik.dtu.dk/gpaw/install.html) into the AMS python environment from PyPI after installing all the requirements (https://wiki.fysik.dtu.dk/gpaw/install.html#requirements):

```
export C_INCLUDE_PATH=$AMSBIN/python3.8/include/python3.8/
amspython -m pip install gpaw
```

- Download and install the PAW datasets (https://wiki.fysik.dtu.dk/gpaw/install.html#install-paw-datasets):

```
# VENV_BIN is something like /home/user/.scm/python/AMS2023.1.venv/bin
VENV_BIN=$(dirname $(amspython -c "import sys; print(sys.executable)"))
# set TARGET_DIR appropriately
TARGET_DIR=/home/user/gpaw
# Download and install the PAW dataset
amspython $VENV_BIN/gpaw install-data $TARGET_DIR
```

Follow the instructions from the `install-data` command.

- Download `GPAW_calculator.py` and place it in an easily accessible place, for example `/home/user/GPAW_calculator.py`.

Listing 2.4: GPAW_calculator.py

```
import numpy
import ase
import gpaw


class ASEGPAWCalculator(ase.calculators.calculator.Calculator):
    def __init__(
        self,
        pbc=[True, True, True],
        cancel_total_force=False,
        charge=0,
        name="atoms",
        **gpaw_kwargs,
    ):
```

(continues on next page)

(continued from previous page)

```
        self.name = name
        self.counter = 1
        self.pbc = pbc
        self.cancel = cancel_total_force
        self._kwargs = gpaw_kwargs
        ase.calculators.calculator.Calculator.__init__(self)
        self._setup_gpaw(charge)

    def calculate(self, atoms=None, properties=None, system_changes=None):
        atoms.center()
        atoms.set_pbc(self.pbc)
        super().calculate(atoms, properties, system_changes)

        self.gpaw.calculate(atoms, properties, system_changes)
        self.results = self.gpaw.results
        # remove total force on the molecule
        if self.cancel:
            molecule_force = self.results["forces"].sum(axis=0)
            per_atom_force = molecule_force / self.results["forces"].shape[0]
            self.results["forces"] -= per_atom_force

    def _setup_gpaw(self, charge):
        self.charge = charge
        txt = self.name
        if self.counter > 1:
            txt = txt + f"_{self.counter}"
        txt = txt + ".txt"
        self.gpaw = gpaw.GPAW(txt=txt, **self._kwargs)
        self.counter += 1

    @property
    def implemented_properties(self):
        return self.gpaw.implemented_properties


# AMS looks for "get_calculator"
get_calculator = ASEGPAWCalculator
```

- Run AMS with the ASE engine and specify `File /path/to/GPAW_calculator.py` (the path must be **absolute**, not relative):

Listing 2.5: GPAW_ams.run

```
AMS_JOBNAME=gpaw $AMSBIN/ams -n 1 <<EOF
properties
  gradients yes
End

system
  Atoms
    H       4.630000     5.000000     5.000000
    H       5.370000     5.000000     5.000000
  End
  Lattice
    10.0 0.0 0.0
    0.0 10.0 0.0
    0.0 0.0 10.0
```

(continues on next page)

(continued from previous page)

```
  End
End

task GeometryOptimization

GeometryOptimization
  Method Quasi-Newton
  Convergence
    Gradients 0.00019446905
  End
End

Engine ASE
  File /path/to/GPAW_calculator.py
EndEngine

EOF
```

GPAW always requires a lattice defined in the AMS system since they are part of the basis set definition for planewaves. For non-periodic systems you can turn off periodic boundary conditions in GPAW by specifying the following block in the ASE Engine:

```
Arguments
    pbc = [False, False, False]
End
```

2.13 M3GNet

See the ML Potential documentation. This uses the Tensorflow-based implementation of M3GNet.

2.14 MACE

Tested with: AMS2025.101, Linux Mint 22, Nov 12 2024

Install mace in a *conda environment* (page 33).

```
conda create -n mace-env
conda install -c conda-forge -n mace-env "python<3.13" ase
conda run -n mace-env \
  python -m pip install \
  mace-torch "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n mace-env \
   python $AMSHOME/scripting/scm/external_engines/backends/_mace/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf mace_test.results

AMS_JOBNAME=mace_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_mace/calculator.py
    Arguments
        model_paths = None
        device = "cpu"
    End
    Python
        Type Conda
        Conda mace-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

If you have a CUDA-enabled GPU, you can set `device = "cuda"` to run on the GPU instead. Or remove the option completely to use the default device.

See the mace documentation and code for details.

If you have a CUDA-enabled GPU, you can set `device = "cuda"` to run on the GPU instead. Or remove the option completely to use the default device.

CuEquivariance with MACE is also supported. Simply install CuEquivariance (https://mace-docs.readthedocs.io/en/latest/guide/cuda_acceleration.html) in the conda environment:

```
conda run -n mace-env \
  python -m pip install \
  cuequivariance-torch cuequivariance-ops-torch-cu12
```

Or use `cuequivariance-ops-torch-cu11` for CUDA 11. Next add `enable_cueq = True` to the `Arguments` block. We observed a speedup of 10-20% and a reduction in memory usage of 50% for a 25x25x25 angstrom water box.

See the mace documentation and code for details.

2.15 MatGL

Note: To use the AMS-bundled version of M3GNet, see *M3GNet* (page 22).

MatGL reimplements M3GNet in PyTorch. To use this implementation (and separate parametrization), you need to run at least Python 3.9.

This can be done by using a *separate Python environment for the ASE calculator* (page 33).

Tested with: AMS2025.101, Ubuntu Linux 22.04, November 14 2024

```
conda create -n matgl-env
conda install -c conda-forge -n matgl-env "python<3.13" ase matgl
conda run -n matgl-env \
  python -m pip install \
  "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n matgl-env \
   python $AMSHOME/scripting/scm/external_engines/backends/_matgl/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf matgl_test.results

AMS_JOBNAME=matgl_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_matgl/calculator.py
    Arguments
        model = "M3GNet-MP-2021.2.8-PES"
    End
    Python
        Type Conda
        Conda matgl-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

See the matgl documentation and code for details.

2.16 MatterSim

Tested with: AMS2025.101, Ubuntu Linux 24.04, Dec 4 2024 and Apple M2, Jan 29 2025

Install mattersim in a *conda environment* (page 33).

```
conda create -n mattersim-env python=3.9
conda run -n mattersim-env \
  python -m pip install \
  mattersim "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of Si bulk, then your conda environment has been set up correctly:

```
conda run -n mattersim-env \
  python $AMSHOME/scripting/scm/external_engines/backends/_mattersim/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf mattersim_test.results

AMS_JOBNAME=mattersim_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        Si -0.67875 -0.67875 -0.67875
        Si 0.77875 0.67875 0.67875
    End
    Lattice
        0.0 2.715 2.715
        2.715 0.0 2.715
        2.715 2.715 0.0
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_mattersim/calculator.py
    Arguments
        device = "cpu"
        load_path = "MatterSim-v1.0.0-5M.pth"
    End
    Python
        Type Conda
        Conda mattersim-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

If you have a CUDA-enabled GPU, you can set `device = "cuda"` to run on the GPU instead. Or remove the option completely to use the default device.

See the Mattersim documentation and code for details.

2.17 NequIP

See the ML Potential documentation.

2.18 ORB

A universal potential for materials.

Tested with: AMS2024.104, Ubuntu Linux 22.04, Jan 7 2025 and and Apple M2, Jan 29 2025

Install orb-models into a separate *conda environment* (page 33):

```
conda create -n orb-env
conda install -c conda-forge -n orb-env "python<3.13" ase
conda run -n orb-env \
  python -m pip install \
  orb-models "$AMSHOME"/scripting/wheels/*.whl
conda run -n orb-env \
  python -m pip install "pynanoflann@git+https://github.com/dwastberg/pynanoflann
↪#egg=af434039ae14bedcbb838a7808924d6689274168"
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n orb-env \
   python $AMSHOME/scripting/scm/external_engines/backends/_orb/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf orb_test.results

AMS_JOBNAME=orb_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_orb/calculator.py
    Arguments
        device = "cpu"
    End
    Python
        Type Conda
        Conda orb-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

If you have a CUDA-enabled GPU, you can set `device = "cuda"` to run on the GPU instead. Or remove the option completely to use the default device.

See the orb documentation and code for details.

2.19 psi4

Tested with AMS2025.101, Ubuntu Linux 22.04, September 25 2024

Install psi4 into a separate *conda environment* (page 33):

```
conda create -n psi4-env
conda install -c conda-forge -n psi4-env "python<3.13" ase psi4 simple-dftd3 dftd3-
↪python dftd4 dftd4-python gcp-correction
conda run -n psi4-env \
  python -m pip install \
  "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n psi4-env \
   python $AMSHOME/scripting/scm/external_engines/backends/_psi4/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf psi4_test.results

AMS_JOBNAME=psi4_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End

Engine ASE
  Type File
  File $AMSHOME/scripting/scm/external_engines/backends/_psi4/calculator.py
  Arguments
    method = 'b3lyp'
    basis = '6-31G*'
    num_threads = 'max'  # or set to a number
  End
  Performance DFT
  Python
      Type Conda
      Conda psi4-env   # this environment must have been set up correctly
  End
EndEngine
eor
```

See the psi4 documentation and code for details.

2.20 PySCF

Tested with: AMS2025.101, Ubuntu Linux 22.04, November 14 2024

Install PySCF into a separate *conda environment* (page 33):

```
conda create -n pyscf-env
conda install -c conda-forge -n pyscf-env "python<3.13"
conda run -n pyscf-env \
  python -m pip install --prefer-binary \
  pyscf ase "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n pyscf-env \
   python $AMSHOME/scripting/scm/external_engines/backends/_pyscf/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf pyscf_test.results

AMS_JOBNAME=pyscf_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End

Engine ASE
  Type File
  File $AMSHOME/scripting/scm/external_engines/backends/_pyscf/calculator.py
  Arguments
    xc = 'pbe'
    basis = '631g*'
  End
  Performance DFT
  Python
    Type Conda
    Conda pyscf-env   # this environment must have been set up correctly
  End
EndEngine
eor
```

See the pyscf documentation and code for details.

2.21 Open Catalyst Project

See *FAIR-Chem* (page 18).

2.22 Quantum ESPRESSO

See the Quantum ESPRESSO documentation.

2.23 sGDML

See the ML Potential documentation.

2.24 SO3LR

Tested with: AMS2025.101, Ubuntu Linux 24.04, Dec 4 2024

Install so3lr in a *conda environment* (page 33).

```
conda create -n so3lr-env
conda install -c conda-forge -n so3lr-env "python<3.13" ase
conda run -n so3lr-env \
  python -m pip install \
  git+https://github.com/general-molecular-simulations/so3lr \
  "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n so3lr-env \
  python $AMSHOME/scripting/scm/external_engines/backends/_so3lr/calculator.py
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf so3lr_test.results

AMS_JOBNAME=so3lr_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
    Lattice
        25.0 0.0 0.0
        0.0 25.0 0.0
        0.0 0.0 25.0
    End
```

(continues on next page)

(continued from previous page)

```
End

Engine ASE
    File $AMSHOME/scripting/scm/external_engines/backends/_so3lr/calculator.py
    Arguments
        calculate_stress = False
        lr_cutoff = 12.0
    End
    Python
        Type Conda
        Conda so3lr-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

If you have a CUDA-enabled GPU, you can set `device = "cuda"` to run on the GPU instead. Or remove the option completely to use the default device.

See the so3lr documentation and code for details.

2.25 tblite

Tested with: AMS2025.101, Linux Mint 22, Nov 12 2024

Install tblite in a *conda environment* (page 33).

```
conda create -n tblite-env
conda install -c conda-forge -n tblite-env "python<3.13" ase tblite tblite-python␣
↪dftd4
conda run -n tblite-env \
  python -m pip install \
  "$AMSHOME"/scripting/wheels/*.whl
```

Test your installation. If the below prints the energy of a water molecule, then your conda environment has been set up correctly:

```
conda run -n tblite-env \
   python "$AMSHOME/scripting/scm/external_engines/backends/_tblite/calculator.py"
```

Run AMS with the ASE engine. Feel free to copy the calculator.py file and modify it to your needs (note: Task and System are provided in the AMS input below).

```
rm -rf tblite_test.results

# tip: copy-paste this input into AMSinput to use the GUI
AMS_JOBNAME=tblite_test $AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 1.5 0. 0.
        C 0.0 0. 0.
        O -1.5 0. 0.
    End
End
```

(continues on next page)

(continued from previous page)

```
Engine ASE
    Type Import
    Import tblite.ase.TBLite
    Arguments
        method = "GFN2-xTB"
    End
    Python
        Type Conda
        Conda tblite-env   # this environment must have been set up correctly
    End
EndEngine
eor
```

See the tblite documentation and code for details.

2.26 VASP

See the VASP via AMS documentation.

2.27 xTB (GFN2-xTB)

See *tblite* (page 30)

CHAPTER

THREE

PYTHON ENVS.: AMSPYTHON, CONDA, ...

New in AMS2025: Use ASE calculators from any Python environment.

Set the input Python Type input option to:

- amspython: Use the Python and ASE package included with AMS. This is the default. See the *quickstart guide* (page 2).
- *Conda* (page 33): If you want to use an ASE calculator that is **not** included in the core ASE package, we recommend that you install it into a separate Python environment using, for example, conda or mamba. See *Get started with conda or mamba* (page 33).
- *Current* (page 36): If you do not have conda or mamba, you can also set the *Python Type to Current* (page 36). This also lets you use either the system Python or any Python virtual environment, by activating it before running AMS.

This means that you can use modern Python tools and packages, without relying on amspython.

Expert option: You can also specify the Python environment with the *SCM_ASE_PYTHON_TYPE environment variable* (page 36).

3.1 Set up a conda environment for use with AMS

3.1.1 Get started with conda or mamba

If you already have `conda` working and installed on your system, you can skip to the next step.

Miniforge (https://github.com/conda-forge/miniforge) holds minimal installers for conda and mamba, and is configured for the community-driven conda-forge (https://conda-forge.org/) channel. Several academic projects that may be interesting to couple to AMS are published on conda-forge.

`conda` and `mamba` are two almost equivalent programs. Typically, `mamba` is a bit faster for installing packages.

> **Warning:** SCM is not affiliated with miniforge or conda-forge. SCM has no control over what packages are installed when you follow these instructions.
>
> Before following these instructions, ensure that your organization allows you to install packages from the internet.

See also:

Blog post: Is conda free? (https://www.anaconda.com/blog/is-conda-free)

Note: All ASE conda *examples* (page 9) use packages from the community-driven conda-forge channel.

- Follow the download and installation instructions for Miniforge (https://conda-forge.org/download/).

- Make sure that `conda` is accessible from a directory in your PATH environment variable. The Miniforge installer may automatically add it to your PATH.
- Once it has been installed, verify that you can open a terminal and successfully run

```
conda --version  # must work to be able to couple to AMS
mamba --version  # optional
```

If conda is not found, you need to add the correct directory to your PATH environment variable (**do not** set PATH from inside the SCM preferences, instead set it through your operating system or shell settings):

- Mac/Linux: Add the correct `/path/to/miniforge3/condabin` directory to your PATH (the `condabin` directory should contain the `conda` script)
- Windows: Add the correct `C:\path\to\miniforge3\Scripts` directory to your PATH (the `Scripts` directory should contain the `conda.exe` executable).

More information about conda on Windows

On Windows, the Miniforge3 installer recommends to not add conda to PATH. However, to use it with AMS:

- conda must exist on PATH, and
- you must be able to run conda in a normal command prompt (*outside* the Miniforge3 prompt)

3.1.2 Create a new conda environment

For creating an environment and installing packages, we will use `conda`, but you can also use `mamba`.

Create a new environment called `my_first_env` containing Python and ASE:

```
conda create -n my_first_env
conda install -n my_first_env "python<3.13" ase  # add more packages as needed

# or use mamba
# mamba create -n my_first_env
# mamba install -n my_first_env "python<3.13" ase  # add more packages as needed
```

Check the installed environments:

```
conda env list
```

Test your ASE and Python installation

```
conda run -n my_first_env python --version
conda run -n my_first_env python -c "import sys; print(sys.executable)"
conda run -n my_first_env python -c "import ase; print(ase.__version__)"
```

3.1.3 Install scm.amspipe and scm.external_engines into the conda environment

Tip: Before installing the SCM packages with `pip`, make sure that you have installed all the conda packages you need using `conda install`.

To couple AMS to your conda environment, you need to install the two packages `scm.amspipe` and `scm.external_engines`. These packages are not available as conda packages but can be installed using `pip`:

```
conda run -n my_first_env python -m pip install "$AMSHOME"/scripting/wheels/*.whl
```

Check your installation:

```
conda run -n my_first_env python -c "import scm.amspipe; import scm.external_engines"
```

If no errors are reported, the SCM packages have been installed correctly.

3.1.4 Use the ASE calculator from the conda environment in AMS

```
"$AMSBIN/ams" -n 1 <<EOF
Task GeometryOptimization

System
   Atoms
      Ag 0. 0. 0.
      Cu 0. 0. 2.0
   End
End

Engine ASE
   Type Import
   Import ase.calculators.emt.EMT
   Python
      Type Conda
      Conda my_first_env
   End
EndEngine
EOF
```

Tip: You can also copy-paste the above input into AMSinput.

For more realistic examples, see *Examples* (page 9)

Tip: The value of the `Conda` input option can also be the absolute path to the conda environment.

3.2 Use the current Python interpreter

If you want to use a custom Python environment but do not have access to conda or mamba (for example, you might use system Python or the Python builtin `venv` for virtual environments), you can set the Python Type to `Current`:

```
Engine ASE
   Python
      Type Current
   End
EndEngine
```

The above means that the python interpreter that is shown by the command `which python` (or `python.exe` on Windows) will be used. If you want this to be different from your system Python, you need to manually activate the corresponding virtual environment.

You need to install `ase`, `scm.amspipe`, and `scm.external_engines` into this Python environment.

For example, before running AMS, you may

```
source /path/to/my/venv/bin/activate
# install packages if not already installed
# python -m pip install ase "$AMSHOME"/scripting/wheels/*.whl

$AMSBIN/ams <<eor
Task SinglePoint

System
    Atoms
        Ag 0. 0. 0.
        Cu 0. 0. 2.0
    End
End

Engine ASE
   Type Import
   Import ase.calculators.emt.EMT
   Python
      Type Current
   End
EndEngine
eor
```

3.3 Specify Python environment with an environment variable

It can sometimes be convenient to set the Conda environment or Python interpreter in an environment variable instead of modifying the input file.

For example, you may want to generate the input file in a script and at that point you may not know what the name or path of the conda environment is.

The below table lists examples of valid engine input and the equivalent value for the `SCM_ASE_PYTHON_TYPE` environment variable.

Environment variable	Engine input

`export SCM_ASE_PYTHON_TYPE=amspython`	`Engine ASE`
	`  Python`
	`    Type amspython`
	`  End`
	`End`

`export SCM_ASE_PYTHON_TYPE=conda:my_`
`↪first_env`
	`Engine ASE`
	`  Python`
	`    Type Conda`
	`    Conda my_first_env`
	`  End`
	`End`

`export SCM_ASE_PYTHON_TYPE=current`	`Engine ASE`
	`  Python`
	`    Type Current`
	`  End`
	`End`

Example: You can set `SCM_ASE_PYTHON_TYPE=conda:my_first_env` as follows:

```
export SCM_ASE_PYTHON_TYPE=conda:my_first_env

"$AMSBIN/ams" -n 1 <<EOF
Task GeometryOptimization

System
   Atoms
      Ag 0. 0. 0.
      Cu 0. 0. 2.0
   End
End

Engine ASE
   Type Import
   Import ase.calculators.emt.EMT
EndEngine
EOF
```

Note that this environment variable will **override** the Python%Type set in the input.

CHAPTER

FOUR

AMS DRIVER'S TASKS AND PROPERTIES

The ASE engine is an engine used by the AMS driver. While the specific options for the ASE engine are described in this manual, the definition of the system, the selection of the task and certain (potential-energy-surface-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.

4.1 Geometry, System definition

The definition of the system, i.e. the atom types and atomic coordinates (and optionally, the lattice vectors and atomic masses for isotopes) are part of the AMS driver input. See the System definition section of the AMS manual.

4.2 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:

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

4.3 Properties in the AMS driver

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

- Elastic tensor
- Hessian
- Nuclear gradients (forces)
- Normal modes
- PES point character
- Phonons
- Stress tensor
- Thermodynamic properties

CHAPTER

FIVE

ASE INPUT OPTIONS

5.1 Engine ASE

AllASEResults

Type
Bool

Default value
Yes

Recurring
False

GUI name
All ASE results

Description
Return all ASE results that are not also part of AMSResults. These values can be found in ase.rkf without any unit conversions.

Arguments

Type
Non-standard block

Description
Arguments to the function or constructor initializing the Calculator. Give each argument on a separate line. This is case sensitive.

Note: Do not perform any Python imports here. Only use Python builtin types. Arguments containing paths must be absolute.

Example:

cutoff = 3.14

title = 'my_string'

my_list_arg = [1, 4, 5]

options_dictionary = {'key1': 11, 'key2': 22}

my_boolean_flag = True

my_path = '/this/must/be/an/absolute/path'

ArgumentsFromFile

Type
String

Default value

Description
Specify the path to a yaml or python file defining the arguments to the function or class defined in *Calculator* or *Callable*.

`File`

Type
String

Default value

Description
Specify the path to a Python file. This file should contain a callable (e.g. function or class) named *get_calculator* that returns an ASE Calculator and uses the arguments defined in *Arguments* or *ArgumentsFromFile*.

You can find examples of suitable calculator.py files (possibly requiring additional installations of packages) in `$AMSHOME/scripting/scm/external_engines/backends`.

`Import`

Type
String

Default value

Description
Specify the module and name of a Calculator installed in the used Python stack. This is case sensitive.

Builtin ASE examples:

ase.calculators.emt.EMT

Other examples requiring special installations:

scm.external_engines.backends._psi4.calculator.get_calculator

scm.external_engines.backends._tblite.calculator.get_calculator

scm.external_engines.backends._mace.calculator.get_calculator

`Performance`

Type
Multiple Choice

Default value
DFTB

Options
[Fast, ForceField, DFTB, DFT, Slow]

Description
Choose which option most accurately corresponds to how long a calculation with the calculator takes.

`Python`

Type
Block

Description
Specify which Python to run.

`Conda`

Type
String

Default value
Description
Name of conda environment. Only used when `Python%Type = Conda`. You may also define the conda environment setting the environment variable SCM_ASE_PYTHON_TYPE=conda:name-of-environment.

If the value is an absolute path it will be executed with `conda run -p`, otherwise it is assumed to be a name of a conda environment that can be executed with `conda run -n`.

The `conda` executable shell script must exist on your `$PATH`. To see available environments, run `conda env list`.

`Type`

Type
Multiple Choice

Default value
amspython

Options
[amspython, Conda, Current]

GUI name
Python environment

Description
Type of Python environment. If `Conda`, set the name or path of the environment under `Python%Conda`. If `Current`, the default `python` (or `python.exe` on Windows) command will be used.

`Results`

Type
String

Recurring
True

Description
The data of this key in the results dictionary of the Calculator is stored in the engine rkf. Multiple results keys can be specified. This is case sensitive.

`Type`

Type
Multiple Choice

Default value
File

Options
[File, Import]

GUI name
Calculator from

Description

Select how to specify which calculator to use.

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 ase.rkf

AMSResults

Section content: Generic results of the ASE 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%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%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%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%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%StressTensor

Type
float_array

Description
The clamped-ion stress tensor in Cartesian notation.

Unit
hartree/bohr^nLatticeVectors

Shape
[:, :]

`AMSResults%UncertaintyScore`

Type
float

Description
?

`AMSResults%YoungModulus`

Type
float

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

Unit
hartree/bohr^3

BZcell(primitive cell)

Section 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]

DOS_Phonons
Section 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]

General

Section content: General information about the ASE 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?

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.

kspace(primitive cell)
Section 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]

Low Frequency Correction

Section 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.

Mobile Block Hessian
Section 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]

Molecule

Section 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
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

MoleculeSuperCell

Section 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
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
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
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

Other

Section content: Contains any information send over by ASE/python which AMS does not know how to handle. This is stored but not documented.

phonon_curves

Section 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
?

Phonons

Section 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]

Thermodynamics
Section 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]

Vibrations

Section 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
10ˉ3

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

INDEX