MLPotential Manual

Amsterdam Modeling Suite 2025.1

www.scm.com

Mar 25, 2025

CONTENTS

CHAPTER

ONE

GENERAL

The MLPotential engine in the Amsterdam Modeling Suite can calculate the potential energy surface using several different types of machine learning (ML) potentials. To use the ML potentials, you need to first separately *install them* (page 5).

The supported models can be found on the page *Models & Backends* (page 9).

1.1 Quickstart guide

To set up a simple MLPotential job using the graphical user interface, see the

- ANI-1ccx Thermochemistry tutorial
- M3GNet tutorial

1.2 What's new in AMS2025.1?

- Improved support for external ML potentials (not included with AMS but available online) through Engine ASE.
- Deprecated: The SchNetPack (1.0.0) and sGDML (0.4.4) backends. To continue using these models, migrate to Engine ASE.

1.3 What's new in AMS2024.1?

- New *models* (page 9): AIMNet2-B973c and AIMNet2-wB97MD3. These are suitable for molecular systems containing H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I. These are currently the only ML potential models that support charged systems (ions), and that predict atomic charges and dipole moments and that give IR intensities when calculating normal modes.
- Train custom M3GNet models with ParAMS and Simple Active Learning (and use them in the MLPotential engine)
- *Auto detection* (page 15) of GPU.
- When using the ANI (or AIMNet2) models, the `mlpotential.txt` file is no longer produced but the engine uncertainty (standard deviation of committee prediction) is written to the standard output and stored on the binary .rkf results files.

1.4 What's new in AMS2023.1?

- New model: M3GNet-UP-2022 based on M3GNet. This is a universal potential (UP) that can be used for the entire periodic table of elements up to, but excluding, Curium (Cm, 96).
- New backend: M3GNet
- PiNN is no longer a backend in MLPotential, but you can use it through Engine ASE.

1.5 Theory of ML potentials

With machine learning potentials, it is possible to quickly evaluate the energies and forces in a system with close to first-principles accuracy. Machine learning potentials are fitted (trained, parameterized) to reproduce reference data, typically calculated using an ab initio or DFT method. Machine learning potentials are sometimes referred to as machine learning force fields, or as interatomic potentials based on machine learning.

Several types of machine learning potentials exist, for example neural-network-based methods and kernel-based methods.

Several types of **neural network potentials** exist. It is common for such potentials to calculate the total energy as a sum of atomic contributions. In a **high-dimensional neural network potential** (HDNNP), as proposed by Behler and Parrinello[1], each atomic contribution is calculated by means of a feed-forward neural network, that takes in a representation of the chemical environment around the atom as input. This representation, or atomic environment **descriptor** or **fingerprint**, consists of a vector of rotationally, translationally, and permutationally invariant functions known as **atom-centered symmetry functions** (ACSF).

Graph convolutional neural network potentials (GCNNPs), or **message-passing network neural potentials**, similarly construct the total energy by summing up atomic contribution, but the appropriate representations of local atomic chemical environments are learned from the reference data.

Kernel-based methods make predictions based on how similar a system is to the systems in the training set.

There are also other types of machine learning potentials. For more detailed information, see for example references[2] and[3].

1.6 Support

SCM provides technical (non-scientific) support for installation and running simulations via the AMS driver.

See also: *Frequently Asked Questions* (page 23)

1.7 Technical information

Each of the supported backends can be used as ASE (Atomic Simulation Environment) calculators (https://wiki.fysik.dtu.dk/ase/index.html). The MLPotential engine is an interface to those ASE calculators. The communication between the AMS driver and the backends is implemented with a named pipe interface. The MLPotential engine launches a python script, `ase_calculators.py`, which initializes the ASE calculator. The exact command that is executed is written as `WorkerCommand` in the output.

[1] J. Behler, M. Parrinello. Phys. Rev. Lett. 98 (2007) 146401 https://doi.org/10.1103/PhysRevLett.98.146401
[2] J. Behler. J. Chem. Phys. 145 (2016) 170901. https://doi.org/10.1063/1.4966192
[3] T. Mueller, A. Hernandez, C. Wang. J. Chem. Phys. 152 (2020) 050902. https://doi.org/10.1063/1.4966192

1.8 References

CHAPTER

TWO

INSTALLATION & UNINSTALLATION

Tip: If AMS does not support your preferred ML potential, you may be able to install it into the AMS Python environment yourself and use it through engine ASE.

The Amsterdam Modeling Suite requires the installation of additional Python packages to run the machine learning potential backends.

If you set up an MLPotential job via the **graphical user interface**, you will be asked to install the packages if they have not been installed already when you save your input. You can also use the package manager. A **command-line installation tool** can also be used, for instance to install the torchani backend:

```
"$AMSBIN"/amspackages install torchani
```

You can use the command line installer to install these packages on a remote system, so that you can seamlessly run MLPotential jobs also on remote machines.

The packages are installed into the AMS Python environment, and do not affect any other Python installation on the system. For the installation, an internet connection is required, unless you have configured the AMS package manager for offline use .

To **uninstall** a package, e.g. `torchani`, run:

```
"$AMSBIN"/amspackages remove torchani
```

2.1 Installing GPU enabled backends using AMSpackages

New in version AMS2023.101.

Various versions of the ML potential packages are available through the AMSpackages, with different system dependencies such as GPU drivers. The option can be selected under the “ML options” menu in the graphical package manager (SCM -> Packages). You can choose from the following options,

- CPU, will install CPU-only backends, including PyTorch and Tensorflow-CPU.
- GPU (Cuda 11.6), will install GPU enabled backends, including Tensorflow, and a CUDA 11.6 specific version of pyTorch.
- GPU (Cuda 11.7), will install GPU enabled backends, including Tensorflow, but will include CUDA 11.7 enabled pyTorch instead.

The default is CPU. Note that this is the only option available under MacOS.

Using the package manager on the the command line or in shell scripts you can use the `--alt` flag, together with one of the options. On the command line the options are denoted as *mlcpu*, *mlcu116* and *mlcu117* respectively. To install GPU enabled versions of the ML potential backends on the command line, for instance using the CUDA 11.7 enabled version of PyTorch:

```
$ "$AMSBIN"/amspackages --alt mlcu117 install mlpotentials
Going to install packages:
nvidia-cuda-runtime-cu11 v[11.7.99] - build:0
tensorflow v[2.9.1] - build:0
All ML Potential backends v[2.0.0] - build:0
torch v[1.13.1+cu117] - build:0
nvidia-cudnn-cu11 v[8.5.0.96] - build:0
M3GNet ML Backend v[0.2.4] - build:0
sGDML Calculator patch v[0.4.4] - build:0
TorchANI Calculator patch v[2.2] - build:0
SchNetPack ML Backend v[1.0.0] - build:0
nvidia-cuda-nvrtc-cu11 v[11.7.99] - build:0
nvidia-cublas-cu11 v[11.10.3.66] - build:0
ANI Models for TorchANI backend v[2.2] - build:0
TorchANI NN module patch v[2.2] - build:0
TorchANI ML backend v[2.2] - build:0
sGDML ML backend v[0.4.4] - build:0
```

Alternatively, to install a single backend for instance torchani:

```
"$AMSBIN"/amspackages --alt mlcu117 install torchani
```

To change the default value, you can set an environment variable `SCM_AMSPKGS_ALTERNATIVES`. For advanced configuration options of the package installation, see also the package manager instructions.

2.2 Installing packages using pip

The package manager installs trusted and tested versions of packages from our website, but if you require a different version you can use pip to install packages from https://pypi.org:

```
"$AMSBIN"/amspython -m pip install -U torch
```

Note: Packages installed through pip alone by the user will not show up as installed in the package manager, but they will be detected and used if possible.

If you install a package into your amspython environment, using `amspython -m pip install`, the package manager will not display it in its overview. However, it will allow you to make use of it for running calculations with the ML Potential module. If you want to make sure that the version you installed will be detected, you can use

```
$ "$AMSBIN"/amspackages check --pip torch
05-11 10:47:57 torch is not installed!
05-11 10:47:57 User installed version located through pip: torch==1.8.1
```

Not all versions of the packages on PyPI work with our ML potential backends.

2.3 Installing NequIP using pip

NequIP (https://github.com/mir-group/nequip) is a popular equivariant machine learning potential, and is technically supported by the MLPotential engine. However, it cannot be installed through AMSpackages.

To install NequIP into the AMS Python environment, you may take the below instructions as a starting point. However, there is no guarantee that they will work for you on your system. SCM does not provide support for installing NequIP.

Tested with: AMS2024.101, Ubuntu Linux 22.04, February 13 2024

- To install NequIP, first install `TorchANI` through the package manager:

```
amspackages install torchani
```

- Next, install the NequIP package and related packages. Note that these versions are only a recommendation and might not work on every system.

```
amspython -m pip install nequip==0.5.5 --no-dependencies
amspython -m pip install e3nn==0.5.1 --no-dependencies
amspython -m pip install opt-einsum==3.3.0 --no-dependencies
amspython -m pip install opt-einsum-fx==0.1.4 --no-dependencies
amspython -m pip install sympy==1.11.1 --no-dependencies
amspython -m pip install mpmath==1.2.1 --no-dependencies
amspython -m pip install torch-runstats==0.2.0 --no-dependencies
amspython -m pip install scikit-learn==1.2.0 --no-dependencies
amspython -m pip install joblib==1.3.2 --no-dependencies
amspython -m pip install threadpoolctl==3.2.0 --no-dependencies
amspython -m pip install torch-ema==0.3 --no-dependencies
```

- For using the Allegro plugin (https://github.com/mir-group/allegro) for NequIP (see `$AMSHOME/scripting/scm/params/examples/Allegro` for an example) install the Allegro package from source (only python files):

```
cd <some place where it is convenient to install programs>
git clone --depth 1 https://github.com/mir-group/allegro.git
cd allegro
amspython -m pip install . --no-dependencies
```

2.4 Debugging installation and available resources

A tool is provided to investigate the current installation of ML backends and frameworks. This tool will also report the resources that would be found by AMS if a calculation was performed with default settings.

The tool is used as follows:

```
$AMSBIN/amspython $AMSHOME/Utils/check_ml_backends.py
```

The following sections are part of the output:

- **Installed machine learning frameworks**: which frameworks are installed
- **Installed machine learning backends**: which backends are installed
- **Machine learning framework details**: details by the frameworks about CPU and GPU usage

Example output:

```
Installed machine learning frameworks:
PyTorch    : installed!
TensorFlow : installed!

Installed machine learning backends:
AIMNet2    : installed!
ANI2       : installed!
NequIP     : installed!
M3GNet     : installed!

Machine learning framework details (simulating an AMS calculation):

#####################PyTorch setup###################
PyTorch 1.13.1+cpu found the following devices:
Number of threads were not limited, using all available CPU cores.
Using CPU only.
#####################################################

###################TensorFlow setup#################
TensorFlow 2.9.1-cpu found the following devices:
PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')
NumThreads was not specified in the MLPotential engine block, so TensorFlow will use␣
↪all available cores.
#####################################################
```

If there are any issues, before contacting support (support@scm.com), please run the following command:

```
$AMSBIN/amspython $AMSHOME/Utils/check_ml_backends.py --debug
```

and then report the output with your question.

This will give us additional details on why a framework or backend was not considered installed and reports about potential issues in the environment.

CHAPTER

THREE

MODELS & BACKENDS

3.1 Included (pre-parameterized) models

A **model** is the combination of a functional form with a set of parameters. Six pre-parameterized models can be selected: M3GNet-UP-2022 (**U**niversal **P**otential), AIMNet2-B973c, AIMNet2-wB97MD3, ANI-2x, ANI-1ccx, and ANI-1x. The predictions from the AIMNet2-* and ANI-* models are calculated from **committees** (ensembles), meaning that the final prediction is an average over several independently trained neural networks.

Table 3.1: Pre-parameterized models for the MLPotential engine

	M3GNet-UP-2022	AIMNet2-B973c	AIMNet2-wB97MD3	ANI-2x	ANI-1ccx	ANI-1x
Type	Neural network	Neural network	Neural network	Neural network	Neural network	Neural network
Committee size	1	4	4	8	8	8
Atomic environment descriptor	m3gnet	AIMNet2	AIMNet2	ACSF	ACSF	ACSF
Supported elements	95: H, He, Li, .., Am	H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I	H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I	H, C, N, O, F, S, Cl	H, C, N, O	H, C, N, O
Supports charged systems	No	Yes	Yes	No	No	No
Supports 3D periodicity	Yes	No	No	Yes	Yes	Yes
Predicts atomic charges	No	Yes	Yes	No	No	No
Predicts dipole moment	No	Yes	Yes	No	No	No
Predicts energy uncertainty	No	Yes	Yes	Yes	Yes	Yes
Predicts force uncertainty	No	Yes	Yes	Yes	Yes	Yes
Retrainable with ParAMS	Yes	No	No	No	No	No
Training set structures	materials project	molecules	molecules	organic molecules	organic molecules	organic molecules
Reference method	PBE, PBE+U	B97-3c	ωB97M-D3/Def2-TZVPP	ωB97-x/6-31G(d)	DLPNO-CCSD(T)/CB	ωB97-x/6-31G(d)
Backend	m3gnet	AIMNet2	AIMNet2	TorchANI	TorchANI	TorchANI
Reference	[1]	[2]	Page 10, [2]	[3]	[4]	[5]

Note: Use the horizontal scrollbar in the above table to see all supported models.

`Model`

Type
Multiple Choice

Default value
ANI-2x

Options
[Custom, AIMNet2-B973c, AIMNet2-wB97MD3, ANI-1ccx, ANI-1x, ANI-2x, M3GNet-UP-

[1] C. Chen, S. P. Ong. Nature Computational Science 2, 718–728 (2022). arXiv.2202.02450 (https://doi.org/10.48550/arXiv.2202.02450).
[2] D. M. Anstine, R. Zubatyuk, O. Isayev. https://doi.org/10.26434/chemrxiv-2023-296ch
[3] C. Devereux et al., J. Chem. Theory Comput. 16 (2020) 4192-4202. https://doi.org/10.1021/acs.jctc.0c00121
[4] J. S. Smith et al., Nat. Commun. 10 (2019) 2903. https://doi.org/10.1038/s41467-019-10827-4
[5] J. S. Smith et al., J. Chem. Phys. 148 (2018) 241733. https://doi.org/10.1063/1.5023802

2022]

Description

Select a particular parameterization.

ANI-1x and ANI-2x: based on DFT (wB97X)

ANI-1cxx: based on DLPNO-CCSD(T)/CBS

M3GNet-UP-2022: based on DFT (PBE and PBE+U) data.

AIMNet2: based on ωB97m-D3 or B97-3c data.

ANI-1x and ANI-1ccx have been parameterized to give good geometries, vibrational frequencies, and reaction energies for gasphase organic molecules containing H, C, O, and N. ANI-2x can also handle the atoms F, S, and Cl.

M3GNet-UP-2022 is a universal potential (UP) for the entire periodic table and has been primarily trained to crystal data (energies, forces, stresses) from the Materials Project.

AIMNet2 has been parametrized to give good geometries and reaction energies for gasphase molecules and ions containing H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I.

Set to Custom to specify the backend and parameter files yourself.

3.2 Custom models (custom parameters)

Tip: You can use Engine ASE to use any ASE calculator as the engine.

Tip: You can use ParAMS to train your own ML potential parameters.

Set `Model` to **Custom** and specify which backend to use with the `Backend` option. In a typical case, you would have used that backend to train your own machine learning potential.

The backend reads the parameters, and any other necessary information (for example neural network architecture), from either a file or a directory. Specify the `ParameterFile` or `ParameterDir` option accordingly, with a path to the file or directory. Read the backend's documentation to find out which option is appropriate.

Some backends may require that an energy unit (`MLEnergyUnit`) and/or distance unit (`MLDistanceUnit`) be specified. These units correspond to the units used during the training of the machine learning potential.

Example:

```
Engine MLPotential
    Backend SchNetPack
    Model Custom
    ParameterFile ethanol.schnet-model
    MLEnergyUnit kcal/mol
    MLDistanceUnit angstrom
EndEngine
```

`Backend`

Type

Multiple Choice

Options
[M3GNet, NequIP, SchNetPack, sGDML, TorchANI]

Description
The machine learning potential backend.

`MLDistanceUnit`

Type
Multiple Choice

Default value
Auto

Options
[Auto, angstrom, bohr]

GUI name
Internal distance unit

Description
Unit of distances expected by the ML backend (not the ASE calculator). The ASE calculator may require this information.

`MLEnergyUnit`

Type
Multiple Choice

Default value
Auto

Options
[Auto, Hartree, eV, kcal/mol, kJ/mol]

GUI name
Internal energy unit

Description
Unit of energy output by the ML backend (not the unit output by the ASE calculator). The ASE calculator may require this information.

`ParameterDir`

Type
String

Default value
GUI name
Parameter directory

Description
Path to a set of parameters for the backend, if it expects to read from a directory.

`ParameterFile`

Type
String

Default value
Description
Path to a set of parameters for the backend, if it expects to read from a file.

3.3 Backends

Table 3.2: Backends supported by the MLPotential engine.

	M3GNet	SchNetPack	sGDML	TorchANI
Reference	[1]	[8]	[9]	[10]
Methods	m3gnet	HDNNPs, GCN-NPs, …	GDML, sGDML	[ensembles of] HDNNPs
Pre-built models	M3GNet-UP-2022	none	none	ANI-1x, ANI-2x, ANI-1ccx
Parameters from	ParameterDir	ParameterFile	ParameterFile	ParameterFile
Kernel-based	No	No	Yes	No
ML framework	TensorFlow 2.9.1	PyTorch	none, PyTorch	PyTorch

Note: Technically, there is also an AIMNet2 backend but it can only be activated through the pre-parametrized models AIMNet2-B973c and AIMNet2-wB97MD3.

Note: Starting with AMS2023, PiNN[7] is only supported as a custom Calculator through Engine ASE[6].

Note: For **sGDML**, the order of the atoms in the input file **must** match the order of atoms which was used during the fitting of the model.

Note: If you use a custom parameter file with **TorchANI**, the model specified via `ParameterFile filename.pt` is loaded with `torch.load('filename.pt')['model']`, such that a forward call should be accessible via `torch.load('filename.pt')['model']((species, coordinates))`. The energy shifter is not read from custom parameter files, so the absolute predicted energies will be shifted with respect to the reference data, but this does not affect relative energies (e.g., reaction energies).

3.4 References

[8] K. T. Schütt et al., J. Chem. Theory Comput. 15 (2019) 448-455. https://doi.org/10.1021/acs.jctc.8b00908
[9] S. Chmiela et al. Comp. Phys. Commun. 240 (2019) 38-45. https://doi.org/10.1016/j.cpc.2019.02.007
[10] X. Gao et al. J. Chem. Inf. Model (2020). https://doi.org/10.1021/acs.jcim.0c00451
[7] Y. Shao et al., J. Chem. Inf. Model. 60 (2020) 1184-1193. https://doi.org/10.1021/acs.jcim.9b00994
[6] https://wiki.fysik.dtu.dk/ase/index.html

CHAPTER

FOUR

CPU & GPU (CUDA), PARALLELIZATION

If the GPU-enabled versions of the backends have been *installed* (page 5), then by default the calculation will try to autodetect if there is a GPU available to run on, and if so use that. But you can also enforce to run on a specific `Device`.

To limit the number of CPU threads, the `NumThreads` keyword can be used if the backend uses PyTorch as its machine learning framework. Alternatively, you can set the environment variable OMP_NUM_THREADS.

To use a CUDA-enabled GPU, ensure that a CUDA-enabled version of TensorFlow or PyTorch has been installed (see *Installation & Uninstallation* (page 5)).

If the software has problems detecting the GPU, see *Debugging installation and available resources* (page 7).

`Device`

Type
Multiple Choice

Default value

Options
[, cpu, cuda:0, cuda:1, mps]

Description
Device on which to run the calculation (e.g. cpu, cuda:0).

If empty, the device can be controlled using environment variables for TensorFlow or PyTorch.

`NumThreads`

Type
String

Default value

GUI name
Number of threads

Description
Number of threads.

If not empty, OMP_NUM_THREADS will be set to this number; for PyTorch-engines, torch.set_num_threads() will be called.

Note: Because the calculation runs in a separate process, the number of threads is controlled by the input keyword NumThreads and *not* by the environment variable NSCM. **We recommend setting NSCM=1** when using the MLPotential engine.

Only single-node calculations are currently supported.

AMS will report the compute resources it found for machine learning potentials in standard out after applying any restrictions from environment variables as well as `NumThreads` and `Device`.

GPU with PyTorch backend:

```
####################PyTorch setup####################
Number of threads were not limited, using all available CPU cores.
Using GPU: "cuda:0" as found by auto setup.

#####################################################
```

GPU with TensorFlow backend:

```
##################TensorFlow setup##################
TensorFlow found the following devices:
PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')
PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')
NumThreads was not specified in the MLPotential engine block, so TensorFlow will use␣
↪all available cores.
#####################################################
```

CPU only with PyTorch backend:

```
####################PyTorch setup####################
Number of threads were not limited, using all available CPU cores.
Using CPU only.
#####################################################
```

CPU only with TensorFlow backend:

```
##################TensorFlow setup##################
TensorFlow found the following devices:
PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')
NumThreads was not specified in the MLPotential engine block, so TensorFlow will use␣
↪all available cores.
#####################################################
```

CHAPTER

FIVE

EXAMPLES

Before running these examples, you need to *install* (page 5) the appropriate backend.

5.1 AIMNet2

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor

Task SinglePoint
Properties
    Gradients Yes
    DipoleMoment Yes
    Charges Yes
End
System
    Atoms
        C -3.1168071304396 -3.303815558265976 0.0
        C -1.981697818585937 -4.344278296437102 -0.02375164458614451
        H -2.786224865835057 -2.422406202569603 0.5293702995256127
        H -3.978089852534168 -3.72201295092763 0.4997029251388921
        H -3.382157359184612 -3.037352354271593 -1.012416227162533
        H -1.120415096491369 -3.926080903775447 -0.5234545697250367
        H -2.312280083190481 -5.225687652133474 -0.5531219441117571
        H -1.716347589840925 -4.610741500431485 0.9886645825763885
    End
    Charge 0
End

Engine MLPotential
    Model AIMNet2-B973c
EndEngine
eor
```

5.2 M3GNet-UP-2022

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" <<EOF
Task SinglePoint

Properties
    Gradients True
    StressTensor Yes
End

System
    Atoms
        C 1.1705 0.2287 -0.3792
        C 0.0000 -0.6414 0.0001
        C -1.1706 0.2288 0.3792
        O -2.2331 0.0875 -0.1770
        O 2.2330 0.0876 0.1771
        H 1.0604 0.9753 -1.1517
        H -0.2743 -1.2708 -0.8464
        H 0.2743 -1.2705 0.8469
        H -1.0592 0.9787 1.1484
    End
    Lattice
       7.0000 0.0000 0.0000
       0.0000 7.0000 0.0000
       0.0000 0.0000 7.0000
    End
End

Engine MLPotential
    Model M3GNET-UP-2022
EndEngine

EOF
```

5.3 NequIP

```
#!/bin/sh

# Important:
# To run this example you must first install NequIP into the AMS Python
# environment.

# See
# MLPotential documentation ->
#   Installation & Uninstallation ->
#     Installing NequIP using pip

export NSCM=1
```

(continues on next page)

(continued from previous page)

```
$AMSBIN/ams --delete-old-results <<EOF
Task SinglePoint

Properties
    Gradients Yes
End

System
    Atoms
        O -1.239192986364078 0.7447630849670073 0.2053776714668624
        C -0.4264399234121155 -0.3765076809707508 -0.005559613642175442
        C 1.015520464999812 -0.00839542074865508 0.1578564463228227
        O 1.445056954461275 0.3261487267553756 1.250054826420532
        H -1.187895927300546 1.292145824840463 -0.6216000200358711
        H -0.6908065725149979 -1.158618889736239 0.7362545961529634
        H -0.5963066387066123 -0.7970408421468641 -1.022241368854639
        H 1.680064628837262 -0.02249480296033578 -0.7001425378304952
    End
End

Engine MLPotential
    Backend NequIP
    Model Custom
    ParameterFile $AMSHOME/examples/MLPotential/NequIP-Custom/model.pth
    MLEnergyUnit eV
    MLDistanceUnit angstrom
EndEngine
EOF
```

CHAPTER

SIX

AMS DRIVER'S TASKS AND PROPERTIES

MLPotential is an engine used by the AMS driver. While the specific options for the MLPotential 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.

6.1 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.

Note: The MLPotential Engine currently only supports 0D (molecules) and 3D (bulk) systems.

6.2 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

6.3 Properties in the AMS driver

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

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

CHAPTER

SEVEN

FREQUENTLY ASKED QUESTIONS

7.1 General

7.1.1 Can I train my own ML Potentials?

Yes, see documentation pages for ParAMS and Simple Active Learning.

7.1.2 Can I use implicit solvation (e.g., COSMO) with ML potentials?

No.

7.2 Errors and warning messages

7.2.1 Isolated atoms found in the structure

This message can appear with M3GNet when there are isolated atoms (atoms with no neighbors) in the structure. If this is what you expect then you can ignore the message.

The warning may also appear as "stderr: Isolated atoms found in the structure".

7.2.2 sh: line 1: 1351 Illegal instruction: 4 sh

You may be attempting to run PyTorch on a rather old cpu. Make sure you have installed PyTorch and all other packages through the AMS package manager.

CHAPTER

EIGHT

MLPOTENTIAL KEYWORDS

8.1 Engine MLPotential

Backend

Type
Multiple Choice

Options
[M3GNet, NequIP, SchNetPack, sGDML, TorchANI]

Description
The machine learning potential backend.

Device

Type
Multiple Choice

Default value

Options
[, cpu, cuda:0, cuda:1, mps]

Description
Device on which to run the calculation (e.g. cpu, cuda:0).

If empty, the device can be controlled using environment variables for TensorFlow or PyTorch.

MLDistanceUnit

Type
Multiple Choice

Default value
Auto

Options
[Auto, angstrom, bohr]

GUI name
Internal distance unit

Description
Unit of distances expected by the ML backend (not the ASE calculator). The ASE calculator may require this information.

MLEnergyUnit

Type
Multiple Choice

Default value
Auto

Options
[Auto, Hartree, eV, kcal/mol, kJ/mol]

GUI name
Internal energy unit

Description
Unit of energy output by the ML backend (not the unit output by the ASE calculator). The ASE calculator may require this information.

`Model`

Type
Multiple Choice

Default value
ANI-2x

Options
[Custom, AIMNet2-B973c, AIMNet2-wB97MD3, ANI-1ccx, ANI-1x, ANI-2x, M3GNet-UP-2022]

Description
Select a particular parameterization.

ANI-1x and ANI-2x: based on DFT (wB97X)

ANI-1cxx: based on DLPNO-CCSD(T)/CBS

M3GNet-UP-2022: based on DFT (PBE and PBE+U) data.

AIMNet2: based on ωB97m-D3 or B97-3c data.

ANI-1x and ANI-1ccx have been parameterized to give good geometries, vibrational frequencies, and reaction energies for gasphase organic molecules containing H, C, O, and N. ANI-2x can also handle the atoms F, S, and Cl.

M3GNet-UP-2022 is a universal potential (UP) for the entire periodic table and has been primarily trained to crystal data (energies, forces, stresses) from the Materials Project.

AIMNet2 has been parametrized to give good geometries and reaction energies for gasphase molecules and ions containing H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I.

Set to Custom to specify the backend and parameter files yourself.

`NumThreads`

Type
String

Default value
GUI name
Number of threads

Description
Number of threads.

If not empty, OMP_NUM_THREADS will be set to this number; for PyTorch-engines, torch.set_num_threads() will be called.

ParameterDir

Type
String

Default value
GUI name
Parameter directory

Description
Path to a set of parameters for the backend, if it expects to read from a directory.

ParameterFile

Type
String

Default value
Description
Path to a set of parameters for the backend, if it expects to read from a file.

CHAPTER

NINE

KF OUTPUT FILES

9.1 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.

9.2 Sections and variables on mlpotential.rkf

AMSResults

Section content: Generic results of the MLPotential evaluation.

AMSResults%AtomicDipoleMoments

Type
float_array

Description
Atomic dipole moments computed by the engine.

Unit
e*bohr

Shape
[3, Molecule%nAtoms]

AMSResults%BondInfo

Type
subsection

Description
FIXME: this section should include the file shared/ArchivedBondInfo.json, but there is a problem: the variable 'BondInfo.LatticeDisplacements@dim ('BondInfo.LatticeDisplacements@dim)' is longer than 32 characters (the KF limit) and this messes up things. For now I'll just ignore all the variables in here…

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%EnergyUncertainty

Type
float

Description
Uncertainty in the energy predicted by the engine. Exact meaning depends on the engine used.

Unit
Hartree

AMSResults%Gradients

Type
float_array

Description
The nuclear gradients.

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

AMSResults%GradientsMagnitudeUncertainty

Type
float_array

Description
Uncertainty in the magnitude of the gradients based on the variance formula (error propagation).

Unit
Hartree/Bohr

Shape
[Molecule%nAtoms]

AMSResults%GradientsUncertainty

Type
float_array

Description
Uncertainty in the nuclear gradients predicted by the engine. Exact meaning depends on the engine used.

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%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 MLPotential 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
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

INDEX