ASE Manual

Amsterdam Modeling Suite 2024.1

www.scm.com

Apr 05, 2024

CONTENTS

1 Usage **1**
- 1.1 What's new in AMS2024.1? . . . 2
- 1.2 What's new in AMS2023.1? . . . 3
- 1.3 Quickstart guide . . . 3
  - 1.3.1 Quickstart with GUI . . . 3
  - 1.3.2 Quickstart with Python . . . 4
  - 1.3.3 Quickstart with command-line . . . 4
- 1.4 Engine input . . . 4
  - 1.4.1 Calculator from Import . . . 4
    - Example without arguments . . . 4
    - Example with arguments . . . 5
  - 1.4.2 Calculator from Python file . . . 5
    - Example without arguments . . . 5
    - Example with arguments . . . 5
  - 1.4.3 Specifying arguments for the Calculator . . . 6
  - 1.4.4 Obtaining results from the Calculator . . . 6
- 1.5 Specifying the capabilities of a Calculator . . . 6
- 1.6 Troubleshooting . . . 7
- 1.7 Parametrization with ParAMS . . . 7
- 1.8 Support . . . 7
- 1.9 Licensing . . . 7
- 1.10 References . . . 7

2 AMS driver's tasks and properties **9**
- 2.1 Geometry, System definition . . . 9
- 2.2 Tasks: exploring the PES . . . 9
- 2.3 Properties in the AMS driver . . . 10

3 Examples **11**
- 3.1 Overview of external methods/programs . . . 11
- 3.2 AIMNet2 . . . 12
- 3.3 ALIGNN-FF . . . 12
- 3.4 ANI (TorchANI) . . . 13
- 3.5 CHGNet . . . 13
- 3.6 CP2K . . . 15
- 3.7 DeePMD-kit . . . 16
- 3.8 DFTpy . . . 17
- 3.9 EMT . . . 20
- 3.10 GPAW . . . 20
- 3.11 M3GNet . . . 22

3.12 NequIP . . . 22
3.13 Open Catalyst Project . . . 22
3.14 Quantum ESPRESSO . . . 24
3.15 sGDML . . . 24
3.16 VASP . . . 24
3.17 xTB (GFN2-xTB) (method 1, recommended) . . . 24
3.18 xTB (GFN2-xTB) (method 2) . . . 25

4 ASE Keywords **29**
4.1 Engine ASE . . . 29

5 KF output files **31**
5.1 Accessing KF files . . . 31
5.2 Sections and variables on ase.rkf . . . 32

Index **57**

CHAPTER

ONE

USAGE

The **ASE engine** (this manual) is the interface between any existing ASE calculator (https://wiki.fysik.dtu.dk/ase/ase/calculators/calculators.html#supported-calculators) and the AMS Driver (see *Quickstart guide* (page 3) or *Examples* (page 11)).

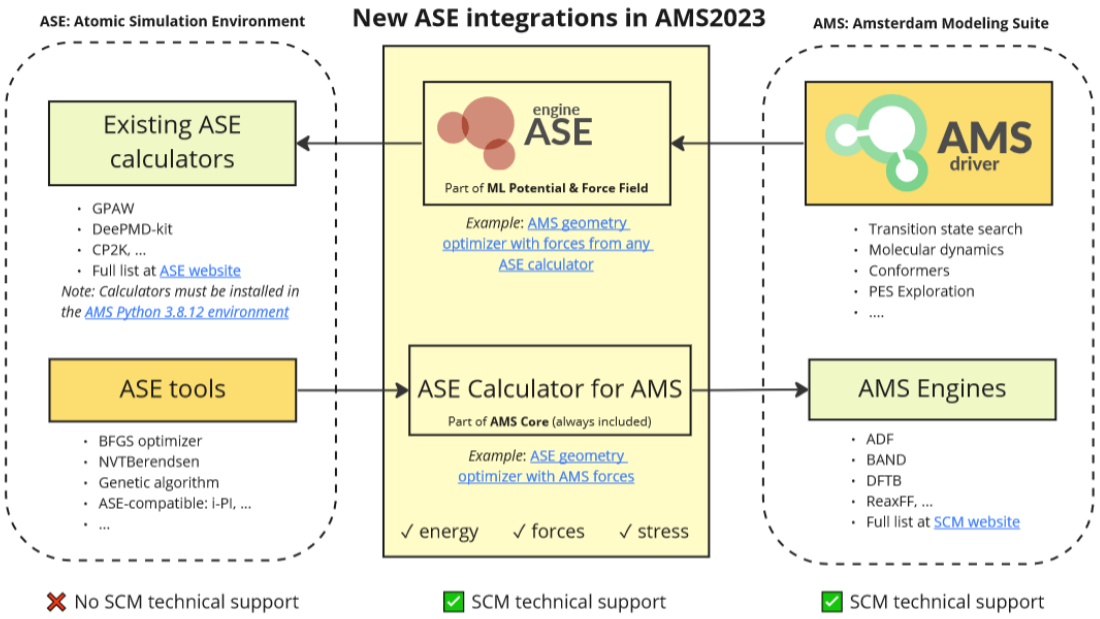
Important: The calculator must be installed in the AMS Python environment.

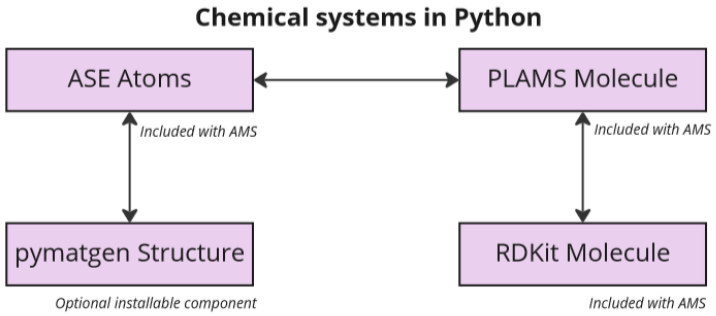
If you launch `$AMSBIN/amspython`, you need to be able to import all the needed packages. Otherwise, make sure to install them.

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 What's new in AMS2024.1?

- It is now possible to indicate to AMS *what properties the Calculator can compute and what kind of system is supported* (page 6) (e.g. if the Calculator can support charged and periodic systems).

1.2 What's new in AMS2023.1?

- The ASE engine is new in AMS2023.

1.3 Quickstart guide

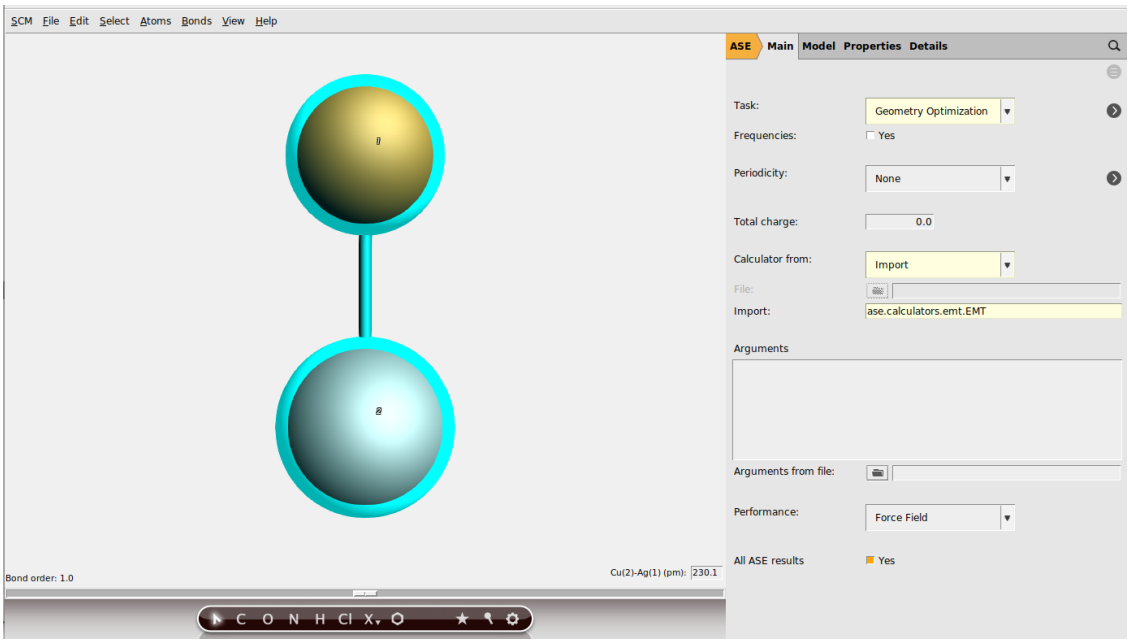
See the below quickstart guide, or have a look at the *example calculators* (page 11).

1.3.1 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



1.3.2 Quickstart with Python

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

1.3.3 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
```

1.4 Engine input

1.4.1 Calculator from Import

If a Calculator is installed in the AMS python stack, it can be selected by specifying the module and name of the Calculator in `Import`.

Example without arguments

The EMT calculator is a fast method applicable to many systems, especially metals.

```
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
```

1.4.2 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)
```

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)
```

1.4.3 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.

1.4.4 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_result
EndEngine
```

See also:

Tutorial: 10 Ways to Get the Energy and Other Properties

1.5 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␣
↪handle a charged system
    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
```

1.6 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.

1.7 Parametrization with ParAMS

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

See the ASE calculator parametrization example.

1.8 Support

SCM does not provide support for any ASE calculators.

1.9 Licensing

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

1.10 References

CHAPTER

TWO

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.

2.1 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.

2.2 Tasks: exploring the PES

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

These are the tasks available in the AMS driver:

- 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

2.3 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

THREE

EXAMPLES

See the *Quickstart guide* (page 3) for a fast example with the ASE EMT Calculator.

Important: This page only contains tips on how to couple certain 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. SCM does not provide support for how to install different ASE calculators into AMS, nor how to set up the input for them.

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.

3.1 Overview of external methods/programs

Method/program	AMS Engine	Type	Website
AIMNet2 (page 12)	ML Potential	ML Potential	Github (https://github.com/isayevlab/AIMNet2)
ALIGNN-FF (page 12)	ASE	ML potential	Github (https://github.com/usnistgov/alignn)
ANI (TorchANI) (page 13)	ML potential	ML potential	Github (https://github.com/aiqm/torchani)
CHGNet (page 13)	ASE	ML potential	Github (https://github.com/CederGroupHub/chgnet)
CP2K (page 15)	ASE	DFT	cp2k.org (https://www.cp2k.org/)
DeePMD-kit (page 16)	ASE	ML potential	Github (https://github.com/deepmodeling/deepmd-kit/tree/master)
DFTpy (page 17)	ASE	orbital-free DFT	Gitlab (https://gitlab.com/pavanello-research-group/dftpy)
EMT (page 20)	ASE	Force field	
GPAW (page 20)	ASE	DFT	website (https://wiki.fysik.dtu.dk/gpaw/)
M3GNet (page 22)	ML potential	ML potential	Github (https://github.com/materialsvirtuallab/m3gnet)
NequIP (page 22)	ML potential	ML potential	Github (https://github.com/mir-group/nequip)
Open Catalyst Project (page 22)	ASE	ML potential	opencatalystproject.org (https://opencatalystproject.org/)
Quantum ESPRESSO (page 24)	Quantum ESPRESSO	DFT	quantum-espresso.org (https://www.quantum-espresso.org/)
sGDML (page 24)	ML potential	ML potential	Github (https://github.com/stefanch/sGDML)
VASP (page 24)	External	DFT	vasp.at (https://www.vasp.at)
xTB (GFN2-xTB) (page 24)	ASE	semi-empirical	Github (https://github.com/grimme-lab/xtb)

3.2 AIMNet2

See the ML Potential documentation.

3.3 ALIGNN-FF

Tested with: AMS2023.101, Ubuntu Linux 22.04, July 19 2023

- SCM → Packages, select CUDA or CPU under ML options
- Install **TorchANI** (will install PyTorch)
- Install DGL into the AMS python environment using instructions from https://www.dgl.ai/pages/start.html. For example (depending on CUDA version):

```
amspython -m pip install  dgl -f https://data.dgl.ai/wheels/cu117/repo.html
amspython -m pip install  dglgo -f https://data.dgl.ai/wheels-test/repo.html
```

- If there's an error about the pydantic version, then do `amspython -m pip uninstall pydantic` followed by `amspython -m pip install pydantic==1.8.0`
- Install ALIGNN: `amspython -m pip install alignn`
- Place the following file in an easily accessible place, for example `/home/user/alignn_ams.py`:

```
from alignn.ff.ff import AlignnAtomwiseCalculator, default_path
import numpy as np


class MyCalculator(AlignnAtomwiseCalculator):
    def calculate(self, atoms, properties, system_changes):
        if sum(atoms.get_pbc()) == 0:
            delta = np.max(atoms.get_positions()) - np.min(atoms.get_positions())
            atoms.set_cell([delta + 10, delta + 10, delta + 10])

        return super().calculate(atoms, properties, system_changes)


def get_calculator(path: str = None):
    if path is None:
        path = default_path()

    return MyCalculator(path=path)
```

Specify Type File and File /path/to/alignn_ams.py in the Engine ASE block:

Listing 3.1: alignn_ams.run

```
#!/bin/sh

$AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 0. 0. 0.
```

(continues on next page)

(continued from previous page)

```
        H 1. 0. 0.
        H -0.7 0.7 0.
    End
End

Engine ASE
    Type File
    File /path/to/alignn_ams.py
EndEngine
eor
```

3.4 ANI (TorchANI)

See the ML Potential documentation.

3.5 CHGNet

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

- SCM → Packages, select CUDA or CPU under ML options
- Install **TorchANI** (will install PyTorch) and **pymatgen** (required by CHGNet)
- In a terminal: `$AMSBIN/amspython -m pip install chgnet==0.1.4`
- Test that the following does not report any error:

```
$AMSBIN/amspython -c "from chgnet.model.dynamics import CHGNetCalculator"
```

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

Listing 3.2: chgnet_ams.py

```
from __future__ import annotations
from chgnet.model.dynamics import CHGNetCalculator
import numpy as np
from ase import Atoms
from pymatgen.io.ase import AseAtomsAdaptor
from ase.calculators.calculator import Calculator, all_changes, all_properties


class AMSCHGNetCalculator(CHGNetCalculator):
    """
    Override the CHGNetCalculator calculate() method.

    Modifications made:

    * add a lattice for non-periodic systems with a gap of at least 10 angstrom

    * only calculate the stress tensor if requested

    The original code can be found at https://github.com/CederGroupHub/chgnet
```

(continues on next page)

(continued from previous page)

```
    """

    def calculate(
        self,
        atoms: Atoms | None = None,
        properties: list | None = None,
        system_changes: list | None = None,
    ) -> None:
        """Calculate various properties of the atoms using CHGNet.

        Args:
            atoms (Atoms | None): The atoms object to calculate properties for.
            properties (list | None): The properties to calculate.
                Default is all properties.
            system_changes (list | None): The changes made to the system.
                Default is all changes.
        """

        if sum(atoms.get_pbc()) == 0:
            delta = np.max(atoms.get_positions()) - np.min(atoms.get_positions())
            atoms.set_cell([delta + 10, delta + 10, delta + 10])

        properties = properties or all_properties
        system_changes = system_changes or all_changes
        Calculator.calculate(
            self,
            atoms=atoms,
            properties=properties,
            system_changes=system_changes,
        )

        # Run CHGNet
        structure = AseAtomsAdaptor.get_structure(atoms)
        graph = self.model.graph_converter(structure)
        task = "e"
        if "forces" in properties:
            task += "f"
        if "stress" in properties:
            task += "s"
        if "magmoms" in properties:
            task += "m"
        model_prediction = self.model.predict_graph(graph.to(self.device), task=task)

        # Convert Result
        factor = 1 if not self.model.is_intensive else structure.composition.num_atoms

        self.results = dict()
        self.results["energy"] = model_prediction["e"] * factor
        self.results["free_energy"] = model_prediction["e"] * factor
        if "f" in model_prediction:
            self.results["forces"] = model_prediction["f"]
        if "s" in model_prediction:
            self.results["stress"] = model_prediction["s"] * self.stress_weight
        if "m" in model_prediction:
            self.results["magmoms"] = model_prediction["m"]
```

(continues on next page)

(continued from previous page)

```
def get_calculator(use_device: str = "cpu", **kwargs):
    return AMSCHGNetCalculator(use_device=use_device, **kwargs)
```

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

Listing 3.3: chgnet_ams.run

```
#!/bin/sh

$AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 0. 0. 0.
        H 1. 0. 0.
        H -0.7 0.7 0.
    End
End

Engine ASE
    File /path/to/chgnet_ams.py
    Arguments
        use_device = "cpu"
    End
EndEngine
eor
```

If you have a CUDA-enabled GPU, you can set `use_device = "cuda"` to run on the GPU instead.

3.6 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 3.4: 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
```

(continues on next page)

(continued from previous page)

```
             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
     &END FORCE_EVAL
    """
  End
EndEngine

eor
```

3.7 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 3.5: deepmd-kit_ams.run

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

3.8 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):
```

(continues on next page)

(continued from previous page)

```
        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)


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
```

(continues on next page)

(continued from previous page)

```
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
             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
```

3.9 EMT

See the *Quickstart guide* (page 3).

3.10 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 3.6: 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,
    ):
        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()
```

(continues on next page)

(continued from previous page)

```
        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 3.7: 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
  End
End

task GeometryOptimization

GeometryOptimization
  Method Quasi-Newton
  Convergence
    Gradients 0.00019446905
```

(continues on next page)

(continued from previous page)

```
  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
```

3.11 M3GNet

See the ML Potential documentation.

3.12 NequIP

See the ML Potential documentation.

3.13 Open Catalyst Project

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

The Open Catalyst Project (OCP) (https://opencatalystproject.org/) has several pre-trained machine learning potentials for catalytic systems. They write *"The aim is to use AI to model and discover new catalysts for use in renewable energy storage to help in addressing climate change"*.

- To install OCP, first install all ML Potential packages through AMSPackages.
- Next create a directory for the OCP source, for example `/home/user/programs/ocp`.
- Obtain the OCP source code and go into the directory:

```
git clone https://github.com/Open-Catalyst-Project/ocp.git /home/user/programs/ocp
cd /home/user/programs/ocp
```

- Install ocp and the requirements. This requires suitable compilers to be installed, see the respective documentation of each package:

```
amspython -m pip install -e .
amspython -m pip install torch-geometric
export CPPFLAGS=-I$AMSBIN/python3.8/include/python3.8/
amspython -m pip install torch-scatter #takes a while
amspython -m pip install torch-sparse #also takes a while
```

(continues on next page)

(continued from previous page)

```
amspython -m pip install mbdb
amspython -m pip install orjson
amspython -m pip install wandb
amspython -m pip install lmdb
amspython -m pip install numba
amspython -m pip install e3nn
```

- Obtain a checkpoint file from the OCP project (https://github.com/Open-Catalyst-Project/ocp/blob/main/MODELS.md), which contains a model architecture and pre-trained parameters. For example:

```
wget https://dl.fbaipublicfiles.com/opencatalystproject/models/2022_09/oc22/s2ef/
↪gndt_oc22_all_s2ef.pt /home/user/Projects/ocp/gndt_oc22_all_s2ef.pt
```

- Run AMS with the OCP ASE Calculator and specify the **absolute** path to the checkpoint file `ocp_ams.run`:

Listing 3.8: ocp_ams.run

```
#!/bin/sh

$AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 0. 0. 0.
        H 1. 0. 0.
        H -0.7 0.7 0.
    End
End

Engine ASE
    Type Import
    Import ocpmodels.common.relaxation.ase_utils.OCPCalculator
    !absolute path
    Arguments
        checkpoint_path = "/home/user/Projects/ocp/gndt_oc22_all_s2ef.pt"
        cpu = True
    End
EndEngine
eor
```

Important: OCP models typically only predict energies and forces but no stress tensor. When the stress tensor is required, e.g. for lattice optimisations, the stress tensor is computed using finite differences which substantially slows down calculations.

3.14 Quantum ESPRESSO

See the Quantum ESPRESSO documentation.

3.15 sGDML

See the ML Potential documentation.

3.16 VASP

See the VASP via AMS documentation.

3.17 xTB (GFN2-xTB) (method 1, recommended)

Tested with: AMS2023.101, Ubuntu Linux 22.04, July 12 2023

```
$AMSBIN/amspython -m pip install xtb
```

Test that the following does not report an error:

```
"$AMSBIN/amspython" -c "from xtb.ase.calculator import XTB"
```

Specify the corresponding `Import` block in Engine ASE:

Listing 3.9: xtb_ams.run

```
#!/bin/sh

# uncomment the below line if you run into Segmentation fault
#ulimit -s unlimited

$AMSBIN/ams -n 1 <<eor
Task GeometryOptimization

System
    Atoms
        O 0. 0. 0.
        H 1. 0. 0.
        H -0.7 0.7 0.
    End
End

Engine ASE
    Type Import
    Import xtb.ase.calculator.XTB
    Arguments
        method = "GFN2-xTB"
    End
EndEngine
eor
```

If you receive a `Segmentation fault` error message, try to run `ulimit -s unlimited`. See also the xtb documentation.

3.18 xTB (GFN2-xTB) (method 2)

Tested with: AMS2023.101, Ubuntu Linux 22.04, July 12 2023

Follow the instructions to install xtb-python into a separate anaconda environment. See the xtb-python documentation for details.

Because xTB runs in a separate python environment, the communication with AMS happens through files. Running xTB in this way introduces significant overhead, as the program is restarted for every evaluation. This also means that the SCF runs from scratch every time. The overhead becomes less noticeable for larger molecules.

If you receive a `Segmentation fault` error message, try to run `ulimit -s unlimited`. See also the xtb documentation.

- Download `xtb_calculator.py` and place it in an easily accessible place, for example `/home/user/xtb_calculator.py`. **IMPORTANT**: Modify the path to the conda environment where you have installed xtb-python!

Listing 3.10: xtb_calculator.py

```
import os
from ase.calculators.calculator import Calculator, all_changes
import subprocess
import pickle

"""
Example for using xtb (GFN2-xTB) from a different python environment together with␣
↪AMS

IMPORTANT: Set the xtb_python_environment variable to the conda python executable␣
↪where xtb is installed!

Ideally xtb-python would be installed into the AMS python environment, then one
could just access the XTB calculator directly without having to go the
roundabout way via files. But it's easier to install xtb into a separate conda␣
↪environment.

To install xtb and xtb-python into a conda environment, do:
conda config --add channels conda-forge
conda install xtb
conda install xtb-python
conda install ase

"""

xtb_python_environment = "/home/hellstrom/anaconda3/bin/python3"


class MyCalculator(Calculator):
    """this class is used inside the AMS python environment"""

    implemented_properties = ["energy", "forces", "charges"]

    def __init__(self, python: str, file: str):
```

(continues on next page)

(continued from previous page)

```
        Calculator.__init__(self)
        self.python = python
        self.file = file

    def calculate(self, atoms=None, properties=["energy"], system_changes=all_
↪changes):
        with open("atoms.pkl", "wb") as f:
            pickle.dump(atoms, f)
        subprocess.run([self.python, self.file])
        with open("results.pkl", "rb") as f:
            self.results = pickle.load(f)
        if os.path.exists("atoms.pkl"):
            os.remove("atoms.pkl")
        if os.path.exists("results.pkl"):
            os.remove("results.pkl")


def get_calculator(**kwargs):
    """this function is imported in the amspython environment"""
    return MyCalculator(python=os.path.abspath(xtb_python_environment), file=os.path.
↪abspath(__file__))


def run_xtb_and_pickle():
    """this is run in the python environment where xTB is installed"""
    from xtb.ase.calculator import XTB

    with open("atoms.pkl", "rb") as f:
        atoms = pickle.load(f)
    atoms.calc = XTB(method="GFN2-xTB")
    atoms.get_forces()
    with open("results.pkl", "wb") as f:
        pickle.dump(atoms.calc.results, f)


if __name__ == "__main__":
    """this is run in the python environment where xTB is installed"""
    run_xtb_and_pickle()
```

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

Listing 3.11: run_xtb.py

```
#!/usr/bin/env amspython
import scm.plams as plams
from os.path import abspath

"""
Example showing how to run the external xtb program using GFN2-xTB.

NOTE 1: To instead run GFN1-xTB, you can use the DFTB engine inside AMS.

NOTE 2: You must modify the path below to give the correct path to the xtb_calculator.
↪py file
```

(continues on next page)

(continued from previous page)

```
NOTE 3: To run this script, use the command
$AMSBIN/amspython run_xtb.py
"""


def main():
    plams.init()
    molecule = plams.from_smiles("O")
    s = plams.Settings()
    s.input.ams.Task = "GeometryOptimization"
    s.input.ASE.File = abspath("/home/hellstrom/xtb_calculator.py")
    s.runscript.nproc = 1  # make sure to run AMS in serial!
    job = plams.AMSJob(settings=s, molecule=molecule, name="opt_GFN2-xTB")
    job.run()

    optimized_molecule = job.results.get_main_molecule()

    plams.log(f"Energy: {job.results.get_energy(unit='eV'):.3f} eV")

    # access atomic charges
    # the charges are stored in "Other%charges" on ase.rkf
    plams.log("Atomic charges:")
    try:
        charges = job.results.get_charges()
        for at, charge in zip(optimized_molecule, charges):
            plams.log(f"{at.symbol} {charge:.3f}")
    except KeyError:
        plams.log("Couldn't get charges")


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

The above shows a PLAMS script but you can also use a normal .run file. See the *CHGNet* (page 13) example and modify accordingly.

CHAPTER

FOUR

ASE KEYWORDS

4.1 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 Python style arguments given to the function or class defined in *Calculator*. This is case sensitive.

`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 if it is not installed in the ams python environment. This file should define a callable (e.g. function or class) named *get_calculator* that returns an ASE Calculator and uses the arguments defined in *Arguments* and *ArgumentsFromFile*.

`Import`

Type String

Default value

Description Specify the module and name of a Calculator installed in the AMS python stack, e.g. 'module.submodule.DFTCalculator'. This is case sensitive.

`Performance`

Type Multiple Choice

Default value ForceField

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

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

`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

FIVE

KF OUTPUT FILES

5.1 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.

5.2 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 bewteen 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 Dimenion 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 possilby 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_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

A

AMS driver, 7, 9
Atoms, 9

C

Charge, 9
Coordinates, 9

E

Elastic tensor, 9

G

GCMC (*Grand Canonical Monte Carlo*), 9
Geometry, 9
Geometry Optimization, 9

H

Hessian, 9

I

IRC (*Intrinsic Reaction Coordinate*), 9
Isotopes, 9

L

Lattice Vectors, 9
Linear Transit, 9

M

Molecular Dynamics, 9

N

NEB (*Nudged Elastic Band*), 9
Nuclear gradients (*forces*), 9

P

PES, 9
PES point character, 9
PESScan (*Potential Energy Surface Scan*), 9
Phonons, 9
Point Charges, 9
Potential Energy Surface, 9

S

Single Point, 9
Stress tensor, 9

T

Task, 9
Thermodynamic properties, 9
Transition State Search, 9

V

Vibrational Analysis, 9

X

xyz, 9