MOPAC Manual

Amsterdam Modeling Suite 2019.3

www.scm.com

Nov 08, 2019

CONTENTS

CHAPTER

ONE

INTRODUCTION

MOPAC [*1* (page 9)] is a general-purpose semiempirical quantum chemistry engine for the study of molecular and periodic structures. A good trade-off between speed and accuracy is achieved through a minimal basis and parameterization against experimental data, with parameters for most elements.

As of the 2019.3 release of the Amsterdam Modeling Suite, MOPAC has become an engine in the new AMS driver setup. If you have not done so yet, we highly recommend you to first read the General section of the AMS Manual. In practice the inclusion of MOPAC into AMS means that MOPAC can now be used for many applications that were previously not supported:

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

Please refer to the AMS manual for a complete overview.

1.1 What's new in MOPAC 2019

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

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

- Calculation of *pKa* (page 5)
- *COSMO* (page 4): all solvents available in ADF/Band are now also available in MOPAC.
- *Static polarizability tensor* (page 5)
- *External electric filed* (page 4)
- Localized orbitals (Natural Bond Orbitals)
- *SCF options* (page 6): Camp-King converger, ...

CHAPTER

TWO

INPUT KEYWORDS

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

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

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

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

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

See also:

The *Examples* (page 11) section of this manual contains several example calculations

2.1 Model Hamiltonian

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

`Model`

Type Multiple Choice

Default value PM7

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

GUI name Method

Description Selects the model Hamiltonian to use in the calculation. AM1: Use the AM1 Hamiltonian. MNDO: Use the MNDO Hamiltonian. MNDOD: Use the MNDO-d Hamiltonian. RM1: Use the RM1 Hamiltonian. PM3: Use the MNDO-PM3 Hamiltonian. PM6: Use the PM6 Hamiltonian. PM6-D3: Use the PM6 Hamiltonian with Grimme's D3 corrections for dispersion. PM6-DH+: Use the PM6 Hamiltonian with corrections for dispersion and hydrogen-bonding. PM6-DH2: Use the PM6 Hamiltonian with corrections for dispersion and hydrogen-bonding. PM6-DH2X: Use PM6 with corrections for dispersion and hydrogen and halogen bonding. PM6-D3H4: Use PM6 with Rezac and Hobza's D3H4 correction. PM6-D3H4X: Use PM6 with Brahmkshatriya, et al.'s D3H4X correction. PM7: Use the PM7 Hamiltonian. PM7-TS: Use the PM7-TS Hamiltonian (only for barrier heights)

The default PM7 model [2 (page 9)] is the latest parametrization for MOPAC and should be the most accurate for most calculations.

EField

Type Float List

Unit Hartree/(e Bohr)

Description Apply the specified homogeneous external electric field. You must specify 3 numbers corresponding to the the x,y and z component of the electric field. Note: EFiled should only be used for single point calculations.

Sparkles

Type Bool

Default value False

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

UnpairedElectrons

Type Integer

GUI name Spin polarization

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

2.2 Solvation

Solvation effects can be included via the COSMO model.

```
Solvation
   Enabled [True | False]
   NSPA [...]
   Solvent
      Eps float
      Name [...]
      Rad float
   End
End
```

Solvation

Type Block

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

Enabled

Type Bool

Default value False

GUI name Use COSMO

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

NSPA

Type Multiple Choice

Default value 42

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

GUI name NSPA

Description Maximum number of COSMO surface points per atom.

Solvent

Type Block

Description Solvent details

Eps

Type Float

GUI name Dielectric constant

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

Name

Type Multiple Choice

Default value Water

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

GUI name Solvent

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

Rad

Type Float

Unit Angstrom

GUI name Radius

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

2.3 Properties

```
Properties
   StaticPolarizability [True | False]
   pKa [True | False]
End
```

Properties

Type Block

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

StaticPolarizability

Type Bool

Default value False

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

pKa

Type Bool

Default value False

GUI name pKa

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

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

```
CalcLocalOrbitals [True | False]
```

CalcLocalOrbitals

Type Bool

Default value False

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

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

2.4 Technical settings

```
SCF
   CampKingConverger [True | False]
   ConvergenceThreshold float
   MaxIterations integer
End
```

SCF

Type Block

Description Options for the self-consistent field procedure.

CampKingConverger

Type Bool

Default value False

GUI name Use Camp-King

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

ConvergenceThreshold

Type Float

Default value 0.0001

Unit kcal/mol

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

MaxIterations

Type Integer

Default value 2000

Description Maximum number of SCF iterations.

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

Mozyme

Type Bool

Default value False

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

2.5 Extra keywords

Finally it is possible to pass any other keywords directly to the MOPAC program [*1* (page 9)]. The full list of keywords can be found on the standalone MOPAC manual (http://openmopac.net/manual/index.html).

```
Keywords string
```

Keywords

Type String

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

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

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

CHAPTER

THREE

REFERENCES

The MOPAC engine in the 2019.3 of the Amsterdam Modeling Suite is a modified version of the standalone MOPAC2016 program developed by Dr. Jimmy Stewart.

1. AMS 2019.3 MOPAC: MOPAC Engine based on the MOPAC2016 source code (James J.P. http://OpenMOPAC.net)
2. James J.P. Stewart, *Optimization of parameters for semiempirical methods VI: more modifications to the NDDO approximations and re-optimization of parameters*, J. Mol. Modeling 19, 1-32 (2013) (https://doi.org/10.1007/s00894-012-1667-x)

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

CHAPTER

FOUR

EXAMPLES

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

4.1 Example: GeoOpt+Frequencies of different O2 spin states

Download GOFREQ_unrestricted.run

```
#!/bin/sh

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

AMS_JOBNAME=O2_singlet $ADFBIN/ams << EOF

Task GeometryOptimization

Properties
   NormalModes Yes
End

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

Engine MOPAC
EndEngine
EOF

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


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

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

Task GeometryOptimization
```

```
Properties
   NormalModes Yes
End

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

Engine MOPAC
   UnpairedElectrons 1
EndEngine
EOF

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


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

AMS_JOBNAME=O2_triplet $ADFBIN/ams << EOF

Task GeometryOptimization

Properties
   NormalModes Yes
End

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

Engine MOPAC
   UnpairedElectrons 2
EndEngine
EOF

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

4.2 Example: Polarizability and hyperpolarizabilities

Download Polar.run

```
#! /bin/sh

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

```
$ADFBIN/ams << eor

Task SinglePoint

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

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

eor

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

cat ams.results/mopac.out
```

4.3 Example: Phonons

Download phonons.run

```
#! /bin/sh

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

AMS_JOBNAME=PPV $ADFBIN/ams << eor

Task SinglePoint

System
   Atoms
      C  1.432420914962878  -1.133348744664622  -0.6391103371334507
      C  0.075602182675705  -0.946866493711738  -0.5497084115413023
      C  2.345587368530869  -0.191932196525464  -0.0965381875924778
      C -0.466207830009865   0.191351632533680   0.0976709467922905
      C  1.803663911626683   0.948320770238396   0.5481842048314089
      C  0.446862721780109   1.134635005787038   0.6370714302545314
      C -1.855533046352049   0.415640484802555   0.2316022019049204
      C -2.841044836424757  -0.419157153044205  -0.2271278521017774
      H -0.602199468183589  -1.681633760082688  -0.9836845375123017
      H  2.480073119105696   1.685566870120453   0.9806344160825713
      H  0.050338193748088   2.021718778887199   1.1315059772026770
      H  1.827043768886768  -2.019275515588153  -1.1372628449390670
      H -2.553512025749108  -1.341888903294209  -0.7454241111668017
```

```
      H -2.143094970839948   1.336869222541756   0.7521871009187797
   End
   Lattice
      6.575588248161897 0.0 0.0
   End
End

Properties
   Phonons Yes
End

NumericalPhonons
   SuperCell
      3
   End
End

Engine MOPAC
   SCF
      ConvergenceThreshold 1.0E-5
   End
EndEngine

eor

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

AMS_JOBNAME=BN $ADFBIN/ams << eor

Task SinglePoint

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

Properties
   Phonons Yes
End

NumericalPhonons
   SuperCell
      2 0
      0 2
   End
```

```
End

Engine MOPAC
   SCF
      ConvergenceThreshold 1.0E-5
   End
EndEngine

eor
```

4.4 Example: Geometry optimization of polyethylene

Download GO_polyethylene.run

```
#! /bin/sh

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

$ADFBIN/ams << eor

Task GeometryOptimization

GeometryOptimization
   Convergence
      Gradients 1.0e-4
   End
End

System
   Atoms
      C -5.686966610289906 -0.00173661090043054 -0.4355683776313619
      C 1.895723638480955 -0.00173661090043054 -0.4355683776313619
      C -3.159403194032952 -0.00173661090043054 -0.4355683776313619
      C 4.491312517927723 -0.0863455367929557 -0.474315563245167
      C -0.6414620718677587 0.2951925083203292 -0.3915990966867868
      C 6.950850470994863 -0.00173661090043054 -0.4355683776313619
      H -6.951201432748922 0.8860020896101368 1.098388839692907
      H 0.7283521430793004 0.9062923240105974 0.9236806626313948
      H -4.047903951160414 0.9426765116296983 0.8853722637672539
      H 3.145873269393606 0.7752976020042145 1.050585933807339
      H -1.902858714187983 1.074510344152748 1.180825231795906
      H 5.579937435062504 1.017854159367372 1.025095354070417
      H -6.950939675307238 -0.8793662426450884 1.105233273612651
      H 0.6317505734636235 -0.8793662426450884 1.105233273612651
      H -4.423376259050285 -0.8793662426450884 1.105233273612651
      H 3.146135026835287 -0.9900707302510107 1.057430367727084
      H -2.067352365692016 -0.7586675287504774 1.334377669481547
      H 5.686877405977529 -0.8793662426450884 1.105233273612651
      H -5.686618534103184 0.8797167676702464 -1.103339585577878
      H 1.790283468854915 0.878947797439763 -1.127416231785004
      H -3.15905511784623 0.8797167676702464 -1.103339585577878
      H 4.410456168039341 0.7690122800643241 -1.151142491463446
      H -0.576790284167599 1.020121306579756 -1.135070326918629
      H 7.127011768776353 0.7534682953709397 -1.016196632797457
      H -5.571852371888105 -0.783856089153288 -1.124998626807626
```

```
      H 1.895778201220154 -0.8852857607466311 -1.100407519984987
      H -3.159348631293752 -0.8852857607466311 -1.100407519984987
      H 4.410162654591847 -0.9959902483525568 -1.148210425870549
      H -0.6344641535070402 -0.6484916142655238 -1.015540900330991
      H 6.950905033734066 -0.8852857607466311 -1.100407519984987
      C -6.950812943854352 0.0006697570117673826 0.4356933698886703
      C 0.7242710564399106 0.03708203634208995 0.4116378321176493
      C -4.428926336438604 -0.04612139106755444 0.3956424425613723
      C 3.38068654505114 -0.01625773059919498 0.3275387816426286
      C -1.921486943590773 0.2660741237064986 0.6146828354694926
      C 5.687004137430418 0.0006697570117673826 0.4356933698886703
   End
   Lattice
      15.16538049754172 0.0 0.0
   End
End

Engine MOPAC
EndEngine

eor
```

4.5 Example: External electric field

Download EField.run

```
#! /bin/sh

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

for EField in 0 0.001 0.01 0.1 ; do

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

Task SinglePoint
System
    Atoms
        C  2.09820318  1.21139817   0.0
        C -0.69940106  1.21139817   0.0
        C  1.39880212  0.0          0.0
        C  1.39880212  2.42279634   0.0
        C  0.0         2.42279634   0.0
        C  0.0         0.0          0.0
        H  3.18949204  1.21139817   0.0
        H  1.94444655  3.36788021   0.0
        H -0.54564443  3.36788021   0.0
        H -1.79068992  1.21139817   0.0
        H -0.54564443 -0.94508387   0.0
        H  1.94444655 -0.94508387   0.0
    End
End

Engine MOPAC
    EField 0.0 0.0 $EField
EndEngine
```

```
eor

done

# If I apply an electric field of 1 [a.u.] on a system with charge 1, I expect the net
# force to be equal to the 1 [a.u.]

AMS_JOBNAME=OH_plus $ADFBIN/ams << eor

Task SinglePoint
System
    Atoms
        O 0.0 0.0 0.0
        H 1.0 0.0 0.0
    End
    Charge 1
End

Properties
    Gradients Yes
End

Engine MOPAC
    EField 0.0 1.0 0.0
EndEngine

eor
```

4.6 Example: Camp-King Converger

Download CampKingConverger.run

```
#! /bin/sh

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

$ADFBIN/ams << eor

Task SinglePoint
System
   Atoms
      Au  0.991939 -1.013256  6.087687
      N   0.671226 -0.526321  4.067029
      Au  1.387933 -1.619200  8.613660
      C  -0.555388 -0.148486  3.616932
      C   1.681804 -0.577717  3.158165
      Au -1.113240 -0.959652  8.002939
      Au  3.551662 -1.763674  7.076475
      C  -0.799200  0.178830  2.295071
      H  -1.346696 -0.123715  4.362730
      C   1.503026 -0.266909  1.821998
      H   2.653410 -0.874661  3.546413
      C   0.236703  0.129690  1.334288
      H  -1.814620  0.448943  2.011007
      H   2.368652 -0.310806  1.163512
```

```
      C   0.011948  0.467735 -0.077072
      C   0.874402  0.017077 -1.100014
      C  -1.079433  1.261872 -0.491478
      C   0.629560  0.357920 -2.422535
      H   1.729669 -0.619555 -0.876597
      C  -1.259607  1.557051 -1.835225
      H  -1.784097  1.673119  0.230431
      N  -0.422938  1.118535 -2.804801
      H   1.284502  0.027292 -3.228959
      H  -2.097962  2.162456 -2.180355
      Au -0.765534  1.615397 -4.922645
      Au -1.186659  2.214533 -7.501957
      Au -3.056147  2.893410 -5.586159
      Au  1.119984  0.909275 -6.730463
      Br -1.580087  2.774299 -9.904465
   End
   Charge -1
End

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

eor
```

4.7 Example: pKa prediction (PLAMS)

This example should be executed using PLAMS.

Download pKa.py

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

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

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

# Data taken from the online MOPAC manual: http://openmopac.net/manual/ (only a sub␣
↪set)
data_tmp = [
   # Molecule name                 smiles                          exp_pKa   ref_
↪mopac_pKa (from mopac's website)
   ['1-Naphthoic_acid',           'C1=CC=C2C(=C1)C=CC=C2C(=O)O',      3.69,     4.35],
   ['2,2,2-Trichloroethanol',     'C(C(Cl)(Cl)Cl)O',                 12.02,    12.22],
   ['2,2,2-Trifluoroethanol',     'C(C(F)(F)F)O',                    12.40,    12.27],
   ['2,2-Dimethylpropionic_acid', 'CC(C)(C)C(=O)O',                   5.03,     5.23],
```

```
   ['2,3,4,6-Tetrachlorophenol',  'C1=C(C(=C(C(=C1Cl)Cl)Cl)O)Cl',      7.10,      6.08],
   ['Acetic_acid',                'CC(=O)O',                           4.76,      5.00],
   ['Acrylic_acid',               'C=CC(=O)O',                         4.25,      4.65],
   ['Benzoid_acid',               'C1=CC=C(C=C1)C(=O)O',               4.20,      4.30],
   ['Citric_acid',                'C(C(=O)O)C(CC(=O)O)(C(=O)O)O',      3.13,      2.56],
   ['Ethanol',                    'CCO',                              16.00,     16.37],
   ['Formic_acid',                'C(=O)O',                            3.77,      3.77],
   ['Glycine',                    'C(C(=O)O)N',                        2.35,      2.53],
   ['Isoleucine',                 'CCC(C)C(C(=O)O)N',                  2.32,      2.48],
   ['Methanol',                   'CO',                               15.54,     15.23],
   ['o-Nitrophenol',              'C1=CC=C(C(=C1)[N+](=O)[O-])O',      7.17,      7.52],
   ['Pentachlorophenol',          'C1(=C(C(=C(C(=C1Cl)Cl)Cl)Cl)Cl)O', 4.90,      5.55],
   ['Phenol',                     'C1=CC=C(C=C1)O',                   10.00,      9.71],
   ['Pyruvic_acid',               'CC(=O)C(=O)O',                      2.50,      2.85],
   ['T-Butanol',                  'CC(C)(C)O',                        17.00,     16.25],
   ['Terephthalic_acid',          'C1=CC(=CC=C1C(=O)O)C(=O)O',         3.51,      3.59],
   ['Valine',                     'CC(C)C(C(=O)O)N',                   2.29,      2.61],
   ['Water',                      'O',                                15.74,     15.75]]

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

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

   mol.properties.name = system['name']
   mol.properties.exp_pKa = system['exp_pKa']
   mol.properties.ref_mopac_pKa = system['ref_mopac_pKa']

   molecules.append(mol)

# MOPAC input:
s = Settings()
s.runscript.nproc = 1 # serial calculation
s.input.ams.Task = 'GeometryOptimization'
s.input.mopac.model = 'PM6'
s.input.mopac.properties.pKa = 'Yes'

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

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

# Print results in a table:
print("Results:\n")
```

```
print("| {:28} | {:8} | {:8} | {:8} | {:8} |".format("Molecule", "exp pKa", "calc pKa
↪", "ref", 'calc-exp'))
for mol in molecules:
   print("| {:28} | {:>8.2f} | {:>8.4f} | {:>8.2f} | {:>8.2f} |".format(mol.
↪properties.name, mol.properties.exp_pKa, mol.properties.calc_pKa, mol.properties.
↪ref_mopac_pKa, mol.properties.calc_pKa-mol.properties.exp_pKa))
print("")

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

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

INDEX

E