CONTENTS

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

2 Input keywords .......................................................... 3
  2.1 Model Hamiltonian .................................................... 3
  2.2 Solvation .............................................................. 4
  2.3 Properties ............................................................ 5
  2.4 Technical settings ..................................................... 6
  2.5 Extra keywords ....................................................... 7

3 References .................................................................. 9

4 Examples .................................................................. 11
  4.1 Example: GeoOpt+Frequencies of different O2 spin states 11
  4.2 Example: Polarizability and hyperpolarizabilities ............ 12
  4.3 Example: Phonons ..................................................... 13
  4.4 Example: Geometry optimization of polyethylene ............ 15
  4.5 Example: External electric field .................................... 16
  4.6 Example: Camp-King Converger .................................. 17
  4.7 Example: pKa prediction (PLAMS) ................................ 18

Index ........................................................................ 21
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 $pK_a$ (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, ...
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, MNDO, 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. MNDO: Use the MNDO Hamiltonian. RM1: Use the RM1 Hamiltonian. PM3: Use the 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 \cite{example} 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: EField 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.

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

**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, Methyprrolidinone, 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

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StaticPolarizability</td>
<td>Bool</td>
<td>False</td>
<td>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.</td>
</tr>
<tr>
<td>pKa</td>
<td>Bool</td>
<td>False</td>
<td>If requested, the pKa of hydrogen atoms attached to oxygen atoms is calculated and printed.</td>
</tr>
</tbody>
</table>

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

CalcLocalOrbitals

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalcLocalOrbitals</td>
<td>Bool</td>
<td>False</td>
<td>Compute and print the localized orbitals, also known as Natural Bond Orbitals (NBO). This is equivalent to the LOCAL mopac keyword.</td>
</tr>
</tbody>
</table>

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

2.4 Technical settings

SCF

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CampKingConverger</td>
<td>Bool</td>
<td>False</td>
<td>Options for the self-consistent field procedure.</td>
</tr>
<tr>
<td>ConvergenceThreshold float</td>
<td>float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxIterations integer</td>
<td>integer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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**

*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 nonsensical 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, FREQUENCY, 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.
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)


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

```bash
#!/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
```

4.2 Example: Polarizability and hyperpolarizabilities

Download Polar.run

```bash
#!/bin/sh

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

Download phonons.run

```bash
#!/bin/sh

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

AMS_JOBNAME=PPV $ADFBIN/ams << eor

Task SinglePoint

System
  Atoms
    C 1.432420914962878 -1.133487446664622 -0.6391103371334507
    C 0.075602182675705 -0.94686493711738 -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 -1.855530346352049 -0.415604484802555 0.2316022019049204
    C -2.841044836424757 -0.419157153044205 -0.2271278521017774
    H -0.602199468183589 -1.68133760082688 -0.9836845375123017
    H 2.480073119105696 1.685566870120453 0.9806344160825713
    H 0.050338193748088 2.021718778887199 1.1315059772026770
    H 1.827043768886768 -2.01927515588153 -1.1372628449390670
    H -2.553512025749108 -1.341888903294209 -0.745424111668017
```

4.3. Example: Phonons
# 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 -2.551245696042202 -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
  Lattice
    5.102491392075644 0.0 0.0
    2.551245696042202 4.418887167494105 0.0
End

Properties
  Phonons Yes
End

NumericalPhonons
  SuperCell
    2 0
    0 2
End

ames
4.4 Example: Geometry optimization of polyethylene

Download GO_polyethylene.run

```sh
#!/usr/bin/sh

# Geometry optimization of a slightly distorted polyethylene chain (6 units in the unit cell)

$ADFBIN/ams << eor
Task GeometryOptimization
GeometryOptimization
Convergence
  Gradients 1.0e-4
End
End
System
Atoms
  C -5.689666610289906 -0.00173661090043054 -0.4355683776313619
  C 1.895723638480955 -0.00173661090043054 -0.4355683776313619
  C -3.159403194032952 -0.00173661090043054 -0.4355683776313619
  C 4.49312517927723 -0.0863455367929557 -0.474315563245167
  C -0.644620718677587 0.2951925083203292 -0.391599966867868
  C 6.950850470994863 -0.0173661090043054 -0.4355683776313619
  H -6.951201432748922 0.8860020896101368 1.09838839692907
  H 0.7283521430793004 0.9062923240105974 0.923680662613948
  H -4.047903951160414 0.94267651126296983 0.3853722637672539
  H 3.145873269393606 0.7752976020042145 1.05058933807339
  H -1.90285714187983 1.074510344152748 1.180825321795906
  H 5.57993435062504 1.017854159367372 1.025095354070417
  H -6.95039675307238 -0.8793662426450884 1.10523273612651
  H 0.6317505734636235 -0.8793662426450884 1.10523273612651
  H -4.423376259050285 -0.8793662426450884 1.10523273612651
  H 3.146135026835287 -0.9900707302510107 1.057430676727084
  H -2.067352365692016 -0.7586675287504774 1.334377669481547
  H 5.686877405977529 -0.8793662426450884 1.10523273612651
  H 0.179028346854915 0.878947797439763 -1.127416231785004
  H -3.15905511784623 0.8797167676702464 -1.10339585577878
  H 4.4104556168039341 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
```

4.4. Example: Geometry optimization of polyethylene
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

    Lattice
        15.16538049754172  0.0  0.0
    End

    Engine MOPAC
    EndEngine

eor
```
4.6 Example: Camp-King Converger

Download CampKingConverger.run

```bash
#!/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

eor
```
### 4.7 Example: pKa prediction (PLAMS)

This example should be executed using PLAMS.

**Download pka.py**

```python
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_pKas) is due to different conformers

# Data taken from the online MOPAC manual: http://openmopac.net/manual/ (only a sub
# set)

data_tmp = [
    ['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],
]
```
MOPAC Manual, Amsterdam Modeling Suite 2019.3

# Turn data_tmp into a dictionary:
systems = [dict(zip(['name', 'smiles', 'exp_pKa', 'ref_mopac_pKa'], d)) 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 = rdkit.Chem.MolFromSmiles(system['smiles'], n_conformers=30, force_field='uff')[0]
    mol.SetProp('name', system['name'])
    mol.SetProp('exp_pKa', system['exp_pKa'])
    mol.SetProp('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, max_jobs=multiprocessing.cpu_count())
jr.run(jobs=jobs)

# Collect results:
for i, mol in enumerate(molecules):
    pKaValues = jobs.children[i].results.read_rkf(['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:

"
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 np.ravel(np.array(errors))]))))
print("Done")
INDEX

E
examples, 9