MoBioTools Documentation

Juanjo Nogueira and Gustavo Cardenas

February 2, 2022

Contents

Chapter 1

Introduction

MoBioTools is a toolkit that allows for setting up quantum mechanical/-molecular mechanical (QM/MM) calculations to ensembles of geometries generated from molecular dynamics (MD) trajectories, or more generally, to arbitrary ensembles of geometries of a given molecular species. Its usefulness stems from its simplicity and its versatility: on the one hand all that is required is a file containing the ensemble of geometries (in what follows we will refer to it as a trajectory file) and a file that contains all the structural information as well as the nature of the atoms composing the system of interest and (if any) the force field parameters (we will refer to it as a topology file); on the other hand, it is versatile in that it works as an interface for several quantum chemistry packages while maintaining a general syntax.

1.1 QM Capabilities

The toolkit provides interfaces with the Gaussian, [1] Orca, [2] (Open)Molcas [3] and NWChem [4] quantum chemistry packages. It allows for constructing customized quantum mechanical (QM) input files for an arbitrary set of geometries accounting for the syntax and the capabilities of the QM packages to be used. It also enables the possibility of generating input files to perform QM/MM calculations using the electrostatic-embedding paradigm, in which the QM atoms are polarized by the surrounding MM point charges. There is also the possibility of subdividing the system into fragments to account for the basis set superposition error (BSSE) correction, [5, 6] for example in the computation of non-covalent interactions. In this regard, there is the possibility of considering either custom residues or systems in which there is a molecule embedded in a complex system for which some (but not all of the) residues/molecule need to be considered within the QM region, and consistency in the number of such residues/molecules is required along the entire trajectory under study.

Chapter 2

Installation

2.1 Installation Requirements

A python installation with version >= 3.6 is required. Most Linux-based systems have a default python3 installation, otherwise one can use, e.g.: Anaconda, which includes more than 150 data analysis python libraries, or miniconda, which is a pocket version of anaconda that includes exclusively python. In either case, new libraries can be installed using the

```
conda
```

command.

Numpy: It comes with the anaconda distribution. Otherwise it can be installed via:

```
conda install numpy
```

```
pip install numpy
```

Cpptraj [7] and pytraj [8]:

If you have an Amber or an AmberTools installation, they should already be present in your $AMBERHOME amber installation directory. Otherwise, cpptraj needs to be installed first. To install pytraj, please visit this link. To install cpptraj and pytraj from scratch, visit this link.

2.2 Installation

Get the MoBioTools source

```
git clone git@github.com:mobiochem/MoBioTools.git
```

then, go to the MoBioTools directory

```
cd MoBioTools
```

set manually the environment variables in the config.sh script, then source it

```
source ./config.sh
```

and finally, carry out the installation itself

```
python3 setup.py install
```

Chapter 3

General Structure of the Input Files

The driver of the QM(/MM) input file generator is the script `main_qminputs.py`, which should be present in the $PATH environment variable if the installation was performed successfully. The `main_qminputs.py` script receives two arguments (input files): a general input, on which the names of the trajectory and the topology files, the name of the QM program and other general settings are provided, and a template file, on which the QM settings are included, depending on the QM program to be used. A help message reminding the user of the arguments to be used can be printed by executing:

```
main_qminputs.py -h
```

The corresponding output should be the following:

```
usage: MoBioTools QM/MM input generator [-h]
[-i MAINFILE] [-t TPLFILE]

optional arguments:
  -h, --help    show this help message and exit
  -i MAINFILE   Input file
  -t TPLFILE    Template file
```

Caveat: the script relies on the pytraj library; this implies that only trajectory and topology files that are compatible with pytraj can be used.

3.1 The General Input File

The general (main) input file admits five mandatory and two optional keywords:

- `tpl`: QM software (gaussian, molcas, nwchem, orca)

- `top`: topology file
- `traj`: trajectory file
- `qmmask`: QM mask (*i.e.* QM atoms that are part of the QM region) in amber format
- `geoms`: geometries to be considered. 1 argument: indexs of the geometry of interest. 2-3 arguments: range of geometries
- `solvmask`: atoms or residues of the environment to be considered in the QM region
- `closest`: Number of closest residues (of the `solvmask`) to be considered within the QM region.

A sample input file is displayed below. The sections enclosed by squared brackets are optional.

```
&main
tpl         = nwchem
top         = GUA-O2-S5.prmtop
traj        = GUA-O2-S5.nc
qmmask      = :1,2
geoms       = 999 [start, [stop, step]]
[solvmask = :WAT]
[closest  = 5]
&end
```

The above input requests to fetch the geometry 999 (**beware**: indexing starts from 0) from the GUA-O2-S5.nc trajectory file and generate an NWChem input file for that geometry, in which atoms belonging to residues 1 and 2 are considered as QM atoms. If instead of generating a single input file it is desired to generate a range of inputs, it can be done by defining the range in the `geoms` keyword as the initial index, the final index and, optionally, the step. An example is displayed below.

```
geoms       = 100  200  2
```

If the `solvmask` and the `closest` keywords are included, the script generates an input file in which (in the present case) the five closest water molecules are included in the QM region.

3.2 The Template File

Unlike the main input file, the syntax of the template depends on the syntax of the quantum mechanical software for which to generate the input files. This implies that a minimum knowledge of the syntax of the QM program

to be used is required. The template file is subdivided into different sections enclosed by ampersands, similar to the `&main...&end` statement in the main input file. To maintain the versatility throughout the available QM interfaces, the template file consists of common sections and optional sections, and only some of the common sections are mandatory (as they contain the minimum information to properly run a QM calculation). The common section are so in the sense that the same section names are used regardless of the QM program to be used.

3.2.1 Common Sections

The mandatory common sections are the `&header`, `&basis` and the `&chgspin`, and the optional common sections are `&bsse` and `&externchg`.

&header

This section encompasses what intuitively represents the header of an QM input file, and in most cases corresponds to a specific section on the input. For example, in Gaussian it corresponds to the `-Link0-` commands, in orca it corresponds to the keywords introduced by `"!"` and in Molcas, the header corresponds to the `GATEWAY` program. In NWChem this section is usually related with the `start` and `title` commands. It is syntax dependent.

&basis

The choice of the basis function. It should be emphasized that in Gaussian, Orca and Molcas the basis functions are in general introduced in sections other than a specific basis section, so that consistency is required in the case the commands declared in the `&header` section include the definition of a basis set. The usefulness of the `&basis` section arises whenever a custom basis set (*e.g.* a mixed basis set or a non-standard basis) is to be used. Below are two examples of the syntax of the `&basis` section when a non-standard basis is used.

```
# Gaussian template
&basis
C N O 0
6-31G*
****
H 0
3-21G

&end

# NWChem template
```

```
&basis
C library 6-31g*
N library 6-31g*
O library 6-31g*
H library 3-21g
&end
```

N.B.: whenever using Effective Core Potentials, the NWChem template file requires a custom section named `&ecp`, in agreement with the ecp section of an NWChem input file.

&chgspin

This section encompasses the definition of the charge and the spin multiplicity of the system under study, in terms of a single string :

```
&chgspin
charge,spin[,charge_mon1,spin_mon1,charge_mon2,spin_mon2]
&end
```

where the charge and the spin are separated by a comma. The squared brackets indicate optional arguments, and account for situations in which the system is to be considered as two separate fragments (or monomers). this is the case, for example whenever the user needs to perform calculations considering the BSSE correction (*e.g.*: whenever the user employs the `solvmask` and `closest` keywords in the main input file and/or whenever the `&bsse` section is defined in the template file. See below for further information). In the latter case, the sys the first two arguments correspond to the charge and the spin multiplicity of the overall system; charge_mon1,spin_mon1 correspond to the charge and the spin multiplicity of the first monomer and likewise for charge_mon2,spin_mon2, for the second monomer. An example of a charge-separation system (consisting of a donor and an acceptor, each locally charged and a doublet) is shown below.

```
&chgspin
0,1,+1,2,-1,2
&end
```

&bsse

If this section is defined in the template file, the script assumes that the system consists of two different monomers, and it will generate one input file for each of the monomers (each monomer in the geometry of the complex) including the basis functions of the other monomer, plus an input file for the overall complex (for a given geometry X, the input files will be named - without extension - mon1_geomX, mon2_geomX and complex_geomX, respectively). There are two ways of using the `&bsse` tool:

1) By manually defining the atoms that belong to each one of the monomers. For the time being, only the amber atom selection mask (using the @ symbol) is allowed. In the example below, atoms 1 to 16 (included) belong to monomer 1 (mon1) and atoms 17 to 18 (included) belong to monomer 2 (mon2).

```
&bsse
mon1 = @1-16
mon2 = @17-18
&end
```

2) By using the `solvmask` and the `closest` keywords in the main input file. In this case, no arguments are required in the `&bsse` section, and the script implicitly assumes that monomer 1 (mon1) consists of the atoms encompassed in the user-defined `qmmask`, whereas monomer 2 (mon2) consists of the N closest residues/molecules to the `qmmask`, that belong to the solvent mask (`solvmask`). This is particularly useful whenever one needs to compute non-covalent interaction energies between a molecular species and some molecules present in the surrounding environment.

```
# MAIN INPUT FILE

&main
 ...
solvmask = :WAT
closest  = 5
&end


# TEMPLATE FILE
 ...
&bsse
&end
```

N.B.: the `solvmask` and the `closest` keywords of the main input file, and the `&bsse` section of the template can be used independently of one another. For example, if the `solvmask` and the `closest` keywords are used as in the listing above, but no `&bsse` section is defined in the template, the script will generate a single QM input file in which the QM atoms will correspond to the atoms in whatever `qmmask` was defined by the user plus the five closest water molecules to that `qmmask`.

&externchg

This section enables the inclusion of external point charges to the generated input files. By default, all those atoms that do not belong to the user-defined

`qmmask` (plus eventually those atoms present in the mask that results from the `solvmask` + `closest` keyword combination in the main input file) are considered as external point charges. No arguments are required.

```
&externchg
&end
```

Important: when using the `solvmask` and `closest` keywords in the main input, the point charges will be considered as belonging to monomer 2, so that they will only be present as background charges of the input files of the complex and the monomer 2. See [9] for further information in this regard.

3.2.2 Program-Specific Sections

Gaussian

Use the `&route` syntax to define the Route section commands for a gaussian input file. Important: use the appropriate keywords in the route section coherently with the settings requested in the template file, *e.g.*: if the `&externchg` is requested in the template file, include the `Charge` keyword in the route section.

```
&route
#P M062X/6-31G* NoSym Charge
&end
...

&externchg
&end
```

Orca

The sections introduced by a % symbol in a standard Orca input file are included as sections in the template file. Some examples are `&cpcm`, `&tddft` and `&casscf`. An example is displayed below.

```
&tddft
nroots 10
maxdim 5
tda 5
&end
```

Molcas

In Molcas, each program-specific section in the template corresponds to a specific program/module called by the software, the exception being the `&GATEWAY` program, for which the `&header` section is dedicated.

```
&seward
 ...
&end

&scf
 ...
&end

&casscf
 ...
&end
```

NWChem

Some of the custom section for the NWChem template file include `&dft`, `&cpcm` and `&tasks`. Caveat: the `&tasks` section can only include one task (no job concatenation is supported), and in the current version only `set` commands associated with the desired task need to be included in the `&tasks` section as well. An example of a template requesting a constrained DFT calculation is shown below.

```
&dft
xc m06-2x
odft
mult 5
cdft 1 16 spin 2
cdft 1 16 spin 2
&end

&tasks
set dft:cdft_maxiter 1000
task dft
&end
```

Important: The `mult 5` value is superseded by the multiplicity value defined in the `&chgspin` section of the template.

Chapter 4

The `pyoverlaps.py` Program

This program generates a Molcas input file for a selected geometry (a simple script with a for loop can be used to consider a range of geometries), performs a CASSCF calculation and compares the obtained active space orbitals with those of a user-provided reference active space (in a molden format). The program then performs iterative CASSCF computations until the active space of the geometry under study is in agreement with the reference. The "correction" of the active space is carried on by using the algorithm described in, [10] in conjunction with the `alter` keyword of the `&RASSCF` module in Molcas. The `pyoverlaps.py` program is a stand-alone program, but it is currently being integrated within the `main_qminputs.py` programs. The arguments the program takes can be visualized by executing

```
main_qminputs.py -h
```

The output displayed is the following

```
usage: PyOverlaps MO recovery pof the active space [-h]
[-p TOP] [-a ALIGN] [-c CRD] [-r REFERENCE]
[-tpl TEMPLATE] [-rng RNG [RNG ...]] [-cl CALC]
[-rst RESTART] [-u UNITS] [-qm QMMASK] [--ig IGEOM]
[--n NGEOM]

optional arguments:
  -h, --help            show this help message and exit
  -p TOP                Topology file
  -a ALIGN              Align geometries?
  -c CRD                Trajectory file
  -r REFERENCE          Reference molden file
  -tpl TEMPLATE         Molcas template file
  -rng RNG [RNG ...]    Range of MOs in the active space
                        (e.g: 30 43)
  -cl CALC              Perform CASSCF intermediate
```

```
                    calculations? (yes/no)
-rst RESTART        Restart from iteration N (N>=1)
-u UNITS            Units (bohr, angstrom)
-qm QMMASK          QM mask
--ig IGEOM          Specific frame for which to generate
                    an input file.
                    Default = 0
--n NGEOM           Number of geometries to generate.
                    If chosen, do not set
                    --ig. Default = 1
```

Consider the example files present in `examples/overlaps`. To perform the molecular orbital (MO) analysis on geometry 50 of the trajectory pAZO-10.nc, execute `pyoverlaps.py` as follows (The active space under consideration comprises MOs 51 to 60):

```
pyoverlaps.py -p pAZO-10.prmtop -c pAZO.nc -a true
-r ref-vacuum.molden -tpl tpl.inp -rng 51 60 -cl yes
-qm :LIG --ig 50
```

Bibliography

[1] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox. Gaussian˜16 Revision C.01, 2016. Gaussian Inc. Wallingford CT.

[2] Frank Neese, Frank Wennmohs, Ute Becker, and Christoph Riplinger. The orca quantum chemistry program package. *The Journal of Chemical Physics*, 152(22):224108, 2020.

[3] Ignacio Fdez. Galván, Morgane Vacher, Ali Alavi, Celestino Angeli, Francesco Aquilante, Jochen Autschbach, Jie J. Bao, Sergey I. Bokarev, Nikolay A. Bogdanov, Rebecca K. Carlson, Liviu F. Chibotaru, Joel Creutzberg, Nike Dattani, Mickaël G. Delcey, Sijia S. Dong, Andreas Dreuw, Leon Freitag, Luis Manuel Frutos, Laura Gagliardi, Frédéric Gendron, Angelo Giussani, Leticia González, Gilbert Grell, Meiyuan Guo, Chad E. Hoyer, Marcus Johansson, Sebastian Keller, Stefan Knecht, Goran Kovačević, Erik Källman, Giovanni Li Manni, Marcus Lundberg, Yingjin Ma, Sebastian Mai, João Pedro Malhado, Per Åke Malmqvist, Philipp Marquetand, Stefanie A. Mewes, Jesper Norell, Massimo Olivucci, Markus Oppel, Quan Manh Phung, Kristine Pierloot, Felix Plasser, Markus Reiher, Andrew M. Sand, Igor Schapiro, Prachi Sharma, Christopher J. Stein, Lasse Kragh Sørensen, Donald G. Truhlar, Mihkel Ugandi, Liviu Ungur, Alessio Valentini, Steven Vancoillie, Valera Veryazov, Oskar Weser, Tomasz A. Wesołowski, Per-Olof

Widmark, Sebastian Wouters, Alexander Zech, J. Patrick Zobel, and Roland Lindh. Openmolcas: From source code to insight. *Journal of Chemical Theory and Computation*, 15(11):5925–5964, 2019. PMID: 31509407.

[4] E. Aprà, E. J. Bylaska, W. A. de Jong, N. Govind, K. Kowalski, T. P. Straatsma, M. Valiev, H. J. J. van Dam, Y. Alexeev, J. Anchell, V. Anisimov, F. W. Aquino, R. Atta-Fynn, J. Autschbach, N. P. Bauman, J. C. Becca, D. E. Bernholdt, K. Bhaskaran-Nair, S. Bogatko, P. Borowski, J. Boschen, J. Brabec, A. Bruner, E. Cauët, Y. Chen, G. N. Chuev, C. J. Cramer, J. Daily, M. J. O. Deegan, T. H. Dunning, M. Dupuis, K. G. Dyall, G. I. Fann, S. A. Fischer, A. Fonari, H. Früchtl, L. Gagliardi, J. Garza, N. Gawande, S. Ghosh, K. Glaesemann, A. W. Götz, J. Hammond, V. Helms, E. D. Hermes, K. Hirao, S. Hirata, M. Jacquelin, L. Jensen, B. G. Johnson, H. Jónsson, R. A. Kendall, M. Klemm, R. Kobayashi, V. Konkov, S. Krishnamoorthy, M. Krishnan, Z. Lin, R. D. Lins, R. J. Littlefield, A. J. Logsdail, K. Lopata, W. Ma, A. V. Marenich, J. Martin del Campo, D. Mejia-Rodriguez, J. E. Moore, J. M. Mullin, T. Nakajima, D. R. Nascimento, J. A. Nichols, P. J. Nichols, J. Nieplocha, A. Otero-de-la Roza, B. Palmer, A. Panyala, T. Pirojsirikul, B. Peng, R. Peverati, J. Pittner, L. Pollack, R. M. Richard, P. Sadayappan, G. C. Schatz, W. A. Shelton, D. W. Silverstein, D. M. A. Smith, T. A. Soares, D. Song, M. Swart, H. L. Taylor, G. S. Thomas, V. Tipparaju, D. G. Truhlar, K. Tsemekhman, T. Van Voorhis, Á. Vázquez-Mayagoitia, P. Verma, O. Villa, A. Vishnu, K. D. Vogiatzis, D. Wang, J. H. Weare, M. J. Williamson, T. L. Windus, K. Woliński, A. T. Wong, Q. Wu, C. Yang, Q. Yu, M. Zacharias, Z. Zhang, Y. Zhao, and R. J. Harrison. Nwchem: Past, present, and future. *The Journal of Chemical Physics*, 152(18):184102, 2020.

[5] S.F. Boys and F. Bernardi. The calculation of small molecular interactions by the differences of separate total energies. some procedures with reduced errors. *Mol. Phys.*, 19(4):553–566, 1970.

[6] Sílvia Simon, Miquel Duran, and J. J. Dannenberg. How does basis set superposition error change the potential surfaces for hydrogen-bonded dimers? *J. Chem. Phys.*, 105(24):11024–11031, 1996.

[7] Daniel R. Roe and Thomas E. Cheatham. Ptraj and cpptraj: Software for processing and analysis of molecular dynamics trajectory data. *Journal of Chemical Theory and Computation*, 9(7):3084–3095, 2013. PMID: 26583988.

[8] Hai Nguyen, Daniel R. Roe, Jason Swails, and David A. Case. Pytraj: Interactive data analysis for molecular dynamics simulations. 2016.

[9] Gustavo Cárdenas, Álvaro Pérez-Barcia, Marcos Mandado, and Juan J. Nogueira. Characterization of cisplatin/membrane interactions by qm/mm energy decomposition analysis. *Phys. Chem. Chem. Phys.*, 23:20533–20540, 2021.

[10] Gustavo Cárdenas and Juan J. Nogueira. An algorithm to correct for the casscf active space in multiscale qm/mm calculations based on geometry ensembles. *International Journal of Quantum Chemistry*, 121(6):e26533, 2021.