PyMobCal/CMobCal Quick Guide 3.0

November 11, 2019

NOTE: If you plan on modifying the code read the developer's manual as well.

1 Installation

Each version of mobcal contains a README.TXT file with details.

Compilation for C version is through a makefile. GNU Scientific Library (GSL) are used for the random number generator.

For the Python version, you need Python 3.X, numpy, and scipy.

MPI version of the code need a working MPI environment, generally provided by your cluster facilities. In addition, you will most probably need to write a submissione script. Ask your IT specialist for details.

1.1 Notes on a single structure

Mobcal assumes that the input files (see details below) provide multiple structures. However, if you have only one structure available for your molecule of interest, you should increase the deafult value of some parameters found in the code. For best results, replace:

```
#define  IMP    25
#define  INUM   250000
```

with

```
#define  IMP    2500
#define  INUM   5000000
```

If you are using a parallel version of the code (MPI or thread), read at the end of the "Algorithms" section.

2 Algorithms

The code includes the following algorithms from the original MobCal papers[1, 2]:

1. Projection Approximation (PA)
2. Hard Sphere Approximation (EHS)
3. Trajectory method (TM)

In these algorithms a particle of the gas is shot towards the molecule of interest along the x-axis.

The PA method is a simple Monte Carlo sampling. If the initial coordinates y and z are within the volume of any atom of the molecule, then a hit is recorded. In this method an actual trajectory for the particle is *not* computed.

The EHS method is similar to PA, but it considers multiple collisions. Thus, we start computing an actual trajectory for the particle. In this method, every time we hit an atom, the code computes the scattering angle and checks if the new trajectory of the particle can actually hit another atoms. If this is the case, a new collision is recorded and a new trajectory is computed. The algorithm continues until the particle leaves the molecules, or a maximum number of collisions INOR is met. INOR can be set by the user, with default value 30.

The TM method is an actual molecuar dynamics simulation. In this case the dispersion and electrostatic forces are explicitly computed.

For PA and EHS the final value is an average over N_{MC} starting position. For each of these starting positions the molecule is rotated of a random angle about its center of mass. This is a way of simulating N_{MC} particles coming from different directions. For a CPU with N_{CORES} cores:

$$N_{MC} = INUM * N_{CORES} \tag{1}$$

For the MPI version of the code N_{CORES} is the number of processes requested. For the pthread code N_{CORES} is equal the constant NCORES in the source doe. For the non parallel version of the code, $N_{CORES} = 1$. The number of starting positions is recorded in the output file as "Number of Monte Carlo trajectories".

For TM, the code computes N_{TM} complete cycles defined as:

$$N_{TM} = ITN * N_{CORES} \tag{2}$$

For each cycle, the code selects INP velocities. For each velocity, IMP random initial positions are evaluated. For each of these IMP positions, a different random rotation of the molecules is used. The equations of motion are integrated using the velocity verlet algorithm. The total numebr of trajectories is reported in the output file as "total number of points"

3 Input File

The Sample folder contains some examples.

The software expect a file called mobcal.run. The file should contain up to three lines (extra lines will be disregarded). The first line contains the name of the file with the structure(s). The second line the name of the output file and the third (optional) line contains the seed for the random number generator. If this line is missing a default seed 5489 is assumed. For the MPI version of the code, each process uses a different seed = RANK * seed. Note that for the GSL library seed=0 means to use a default seed, usually different from 0. Details about the random number generator are reported in the output file.

Example input file (3 line version):

```
a10A1.mfj
a10A1.out
5498
```

Example input file (2 line version, uses default seed):

```
a10A1.mfj
a10A1.out
```

For the input file the structure (A10A1.mfj in the example above):

1. A label for the system
2. Number of structures in the file
3. Number of atoms in each structure

4. units used. Two option: "ang" for Angstron and "au" for atomic units
5. charges. Three options: "calc" for reading the charge from the file, "equal" for uniform charge distribution and "none" for no charge
6. scalng factor. Usually 1.0
7. Now a list with the properties of each atom. One line per atom. up to 5 elements per line: x,y,z of the atom, integer mass of the atom, charge. Only the charge is optional and is need if "calc" was selected above.
8. If there is more than one structure, add a blank line and then a new list with the properties of each atom.

References

[1] M. F. Mesleh, J. M. Hunter, A. A. Shvartsburg, G. C. Schatz, and M. F. Jarrold, Structural Information from Ion Mobility Measurements: Effects of the Long Range Potential, J. Phys. Chem. 1996, 100, 16082-16086; Erratum, J. Phys. Chem. A 1997, 101, 968.

[2] A. A. Shvartsburg and M. F. Jarrold, An Exact Hard Spheres Scattering Model for the Mobilities of Polyatomic Ions, Chem. Phys. Lett. 1996, 261, 86-91.