How to use the MRMC code

Justin Spiriti

January 16, 2019

1 General

The code performs Monte Carlo simulations

2 Form of the potential

$$U = U_{\mathrm{CG}} + U_{\mathrm{AA}} + U_{\mathrm{CG/AA}} + U_{\mathrm{GB}} \tag{1}$$

$$\begin{aligned}
U_{\mathrm{CG}} &= U_{\mathrm{bb}} + U_{\mathrm{G\bar{o}}} && (2)\\
U_{\mathrm{G\bar{o}}} &= \sum_{i,j} \begin{cases} \frac{\varepsilon}{1-m/n}\left[\left(\frac{r_{ij}^0}{r_{ij}}\right)^m - 6\left(\frac{r_{ij}^0}{r_{ij}}\right)^n\right] & \text{(native contacts)} \\ \frac{\varepsilon}{1-m/n}\left(\frac{r^{\mathrm{HC}}}{r_{ij}}\right)^m & \text{(nonnative contacts)} \end{cases} && (3)\\
& && (4)
\end{aligned}$$

3 Input files

Here is an example input file for a Go model simulation of a protein.:

```
READ FFIELD amber99sb.prm
READ DEFS defs-amber99sb.txt
INSERT SEQUENCE
        ASNN LEU TYR ILE GLN TRP LEU LYS ASP GLY GLY PRO SER
        SER GLY ARG PRO PRO PRO SERC
END
AAREGION none
READ PDB 1l2y.pdb
EPS 1.0
CUTOFF 99.0
GO_HARDCORE 1.7
GO_NATIVE 8.0
GO_EXPONENTS 12 10
GO_WELLDEPTH 1.3
MOVES
        backbone 0.5 30.0
        sidechain 0.0 0.0
        backrub 0.5 30.0
end
TEMP 300.0
WRITE PSF trpcage.psf
SAVEFREQ 1000
```

```
PRINTFREQ 100000
TRAJ dcd/trpcage-go-only-1.3.dcd
RUN 1000000000
```

Here is another example for a docking simulation

```
READ FFIELD params/amber99sb-est.prm
READ DEFS params/defs-amber99sb-est.txt
INSERT SEQUENCE
LEUN ALA LEU SER LEU THR ALA ASP GLN MET VAL SER ALA LEU LEU ASP ALA GLU PRO PRO
ILE LEU TYR SER GLU TYR ASP PRO THR ARG PRO PHE SER GLU ALA SER MET MET GLY LEU
LEU THR ASN LEU ALA ASP ARG GLU LEU VAL HIE MET ILE ASN TRP ALA LYS ARG VAL PRO
GLY PHE VAL ASP LEU THR LEU HIE ASP GLN VAL HIP LEU LEU GLU CYS ALA TRP LEU GLU
ILE LEU MET ILE GLY LEU VAL TRP ARG SER MET GLU HIE PRO GLY LYS LEU LEU PHE ALA
PRO ASN LEU LEU LEU ASP ARG ASN GLN GLY LYS CYS VAL GLU GLY MET VAL GLU ILE PHE
ASP MET LEU LEU ALA THR SER SER ARG PHE ARG MET MET ASN LEU GLN GLY GLU GLU PHE
VAL CYS LEU LYS SER ILE ILE LEU LEU ASN SER GLY VAL TYR THR PHE LEU SER SER THR
LEU LYS SER LEU GLU GLU LYS ASP HIP ILE HIE ARG VAL LEU ASP LYS ILE THR ASP THR
LEU ILE HIE LEU MET ALA LYS ALA GLY LEU THR LEU GLN GLN GLN HIE GLN ARG LEU ALA
GLN LEU LEU LEU ILE LEU SER HIE ILE ARG HIE MET SER ASN LYS GLY MET GLU HIE LEU
TYR SER MET LYS CYS LYS ASN VAL VAL PRO LEU TYR ASP LEU LEU LEU GLU MET LEU ASP
ALA HIP ARGC

END
INSERT LIGAND est
#helix 12, plus all AA's with at least one heavy atom within 3 A of the ligand in any reference --
AAREGION 38,41-42,45-46,48,79,82-83,86,89,99,114,116,119-120,123,216,219-220,228-243
READ PDB data/er-active-hie219-est.pdb
EPS 1.0
CUTOFF 10.0 10.0
GO_HARDCORE 1.7
GO_NATIVE 8.0
GO_EXPONENTS 12 10
GO_WELLDEPTH 3.0
MOVES
        backbone 0.1 2.0
        sidechain 0.2 180.0
        backrub 0.1 2.0
        ligand-bond 0.2 180.0
        ligand-trans 0.2 0.1 1.0 2.0
        ligand-rot 0.2 1.0 1.0 180.0
        heavy-atom-trans 0.0 0.1
        heavy-atom-rot  0.0 30.0
end
TEMP 300.0
WRITE PSF psf/er-active-hie219-est-1qku-1-cutoff3allref-movemix10.psf
SAVEFREQ 100
PRINTFREQ 100
TRAJ DCD dcd/er-active-hie219-est-1qku-1-cutoff3allref-movemix10.dcd
SETLIGANDCOM 105.04141998291016 14.675309181213379 23.48505973815918
WRITE PDB data/er-active-hie219-est-1qku-1-cutoff3allref-movemix10-start.pdb
DOCKPREP 4.0 180.0  1000
WRITE PDB data/er-active-hie219-est-1qku-1-cutoff3allref-movemix10-init.pdb
MCLOG mcinfo/mc-log-active-hie219-est-1qku-1-cutoff3allref-movemix10
RUN 100000
```

```
WRITE MCBYATOM mcinfo/mc-byatom-active-hie219-est-1qku-1-cutoff3allref-movemix10
ENERGY LIGAND
```

4 Commands

All distances, lengths, and radii are in Å. The maximum sizes of the moves are in degrees (except for ligand translational moves, which are in Å).

Command	Description
`READ FFIELD filename`	Read a force field file (Tinker format). The force field file should match the force field compiler directive used to compile the program (see below).
`READ DEFS filename`	Read a definitions file.
`READ PDB filename`	Read a PDB file. (Use after INSERT SEQUENCE and AAREGION commands described below.)
`WRITE PSF filename`	Write a PSF file that can be used for visualizing the trajectory in VMD (see TRAJ command below). This PSF file should not be used for analysis or simulation in CHARMM or NAMD. Also, bonds will be missing in the coarse-grained region of the protein (except for the backbone), although atoms will still be present. This may result in a strange appearance in VMD.
`WRITE PDB filename`	Write a PDB file containing the current coordinates.
`WRITE MCBYATOM filename`	Write statistics on how many times each atom was moved to the specified file.
`INSERT SEQUENCE`	Add a protein chain to the system with a sequence on subsequent lines. For proteins with standard termini, the N-terminal and C-terminal amino acids must be labeled with an extra "N" or "C" 'in the sequence as shown above. The word "END" terminates the list of residues.
`INSERT LIGAND residue-name`	Add a ligand with the specified residue to the system. The ligand residue must be defined in the definitions file.
`AAREGION subset`	Define the all-atom region. `subset` may be `none`, `all`, or a list of residue numbers and ranges.
`EPS real`	Set the dielectric constant for the electrostatic interaction.
`CUTOFF distance`	Set the cutoff (in Å) for van der Waals and electrostatic interactions.
`BOXSIZE length`	Enable cubic periodic boundary conditions and set the box length. I am not sure about the status of this feature in this code.
`GO_HARDCORE radius`	Set the hard core radius (r^{HC} above) in Å.
`GO_NATIVE distance`	Set the cutoff distance for determining native contacts in the Go model.

`GO_EXPONENTS m n`	Set the exponents m and n in the Go model.
`GO_WELLDEPTH energy`	Set the well depth ε of the Go model.
`MOVES`	Introduces a section describing Monte Carlo move sizes and mixture. Each subsequent line names a move type, then gives the fraction of moves of that type and the maximum size. The list is terminated by END.
`TEMP temperature`	Set the temperature in K.
`SEED seed`	Set the random number seed (an unsigned 64-bit integer). If set to zero or omitted, a random seed will be chosen based on the time.
`TRAJ filename`	Set the filename for writing the trajectory (DCD file format).
`NCMC move-length cycle-length lambda-schedule-file`	Set up NCMC. The move length is the number of MC moves within an NCMC move; the cycle length is the total number of moves, including additional MC moves done in between NCMC moves.
`MCLOG filename`	Set up a log file to which data on the individual Monte Carlo moves (such as which type of move and the probability of acceptance) is written.
`SAVEFREQ int`	Set the frequency for saving frames to the the trajectory.
`PRINTFREQ int`	Set the frequency for printing energies.
`DOCKPREP distance angle bond_angle`	Prepare for docking by centering the ligand on the all-atom region and giving it a random translation, rotation, and bond rotation.
`SETLIGANDCOM x y z`	Translate the ligand so that its center of mass lies at the specified coordinates.
`ENERGY [LIGAND]`	Compute and output the energy and its terms. If the keyword `LIGAND` is used then the internal energy of the ligand and its interaction energy, as well as their components in terms of the force field.
`RUN steps` or `MC steps`	Run Monte Carlo for the designated number of trial moves.

The possible Monte Carlo move types and the syntax of the corresponding lines in the `MOVES` section are as follows:

Move type	Description
`backbone prob size`	Rotations around protein backbone bonds; N-C$^\alpha$ and C$^\alpha$-C bonds are chosen with equal probability. `prob` gives the probability of choosing a backbone move. `size` gives the maximum size, in degrees, of the bond rotation; the angle of rotation is chosen uniformly in the interval [-`size`, `size`].
`sidechain prob size`	Rotations around protein side chain bonds; each rotatable bond in an amino acid side chain may be chosen with equal probability. `prob` gives the probability of choosing a side chain move. `size` gives the maximum size, in degrees, of the bond rotation; the angle of rotation is chosen uniformly in the interval [-`size`, `size`].
`backrub prob size`	Rotations around protein side chain bonds; each rotatable bond in an amino acid side chain may be chosen with equal probability. `prob` gives the probability of choosing a side chain move. `size` gives the maximum size, in degrees, of the bond rotation; the angle of rotation is chosen uniformly in the interval [-`size`, `size`].
`ligand-bond prob size`	Rotations around ligand rotatable bonds bonds; each rotatable bond in an amino acid side chain may be chosen with equal probability. `prob` gives the probability of choosing a side chain move. `size` gives the maximum size, in degrees, of the bond rotation; the angle of rotation is chosen uniformly in the interval [-`size`, `size`].

5 Compiling the code

The code may be compiled using the `compile` script that is provided. To produce a plain, optimized version of the code, simply invoke the `compile` script without arguments. The compilation uses the Intel C++ compiler when compiled on Frank, and the GNU `g++` compiler when on Dvorak.

The script takes arguments that indicate conditional compiler directives to be used. Adding the `debug` argument creates a debuggable version with disabled optimization that produces lots of extra output. Adding the `timers` argument creates a special version that outputs timing data on individual parts of the calculation.

By default the script also applies a conditional compiler directive that selects the force field. The choices are `AMBER` and `CHARMM19`. By default `AMBER` is applied; this can be changed within the script. The `CHARMM19` selection should work, but has not been tested recently.