Tabulated Monte Carlo for Proteins

Justin Spiriti

December 1, 2016

1 General

The program is designed to perform Monte Carlo simulations of peptides or proteins, using tables to calculate major parts of the energy function. Each protein is divided into fragments that are considered rigid. Interaction energy tables based on the net displacement (expressed in spherical coordinates (r, θ, ϕ) and relative orientation (expressed as Euler angles (ϕ', θ', ψ')) between each pair of fragments are used to calculate most of the van der Waals and electrostatic interactions. Covalent tables may be used to calculate internal energy terms along the peptide backbone. Smoothing techniques may also be applied to the tables, thus creating coarse-grained models that are continuously adjustable.

The dynamical variables in the simulation are the center of mass $\mathbf{r}$ and a quaternion describing the orientation $\mathbf{q}$ of each fragment. Each fragment type has a "reference geometry" and the Cartesian coordinates for the system are generated from this reference geometry and the center and orientation of each fragment.

The program is designed to use the CHARMM 19 force field together with either a constant or distance-dependent dielectric constant.

2 Files and compilation

Several directories contain code and a compile script. Use the `compile` script to compile the code. The script takes parameters that indicate preprocessor directives (the script will convert them to upper case). The name of the executable will be suffixed with the names of any processor directives . The possible processor directives are as follows:

Directive	Meaning
`DEBUG`	Disables optimization and provides copious, detailed output
`EXCHANGE`	Enables temperature or resolution replica exchange (uses MPI)
`TIMERS`	Provides detailed profiling data
`NO_CRC`	Disables CRC checksum checking of tables (can save time in certain situations)
`NO_MMAP_TABLES`	Disables memory mapping of tables (may be needed for portability if compiled in an environment that does not support the `mmap()` system call
`NO_TRIG_TABLES`	Disables fast, tabulated trignometric functions in table lookups, replacing them with trigonometric functions from the C++ standard libraries (reduces performance)

3 Running simulations

The program is invoked as follows:

```
tablemc run <input-file>
```

The format of the input file is as follows:

```
defs-protein4-charmm.txt fragments3-scaled/%s.xyz charmm19-scaled.prm 1 0
#Definitions file, fragment geometries (must include one %s), parameter file.
#Use of interaction tables (1=yes, 0=no), use of covalent tables (1=yes, 0=no)
ACE GLY GLU TRP THR TYR ASP ASP ALA THR LYS THR PHE THR VAL THR GLU CBX
END
#Sequence. May occupy multiple lines. Must terminate with "END".
#If using standard N- and C-terminii, must use special definitions for those residues (indicated by
     "N" or "C" suffix) provided in definitions file.
start gb1-hairpin-min.pdb gb1-hairpin-min2.pdb
#"start" or "restart". With "start": PDB file of starting structure, PDB file to write initial
    fitted structure.
#With "restart": restart file to read, initial structure.
100000000 1000 10000 100000000 0 0 300.00
#number of steps, trajectory save frequency, trajectory print frequency
#table checksum check frequency (keep this infrequent!) ,
#random number seed (0=pick based on time), detailed energy writing (1=yes, 0=no), temperature (K)
#in all cases a frequency of zero means don't do it
0 0.0 12.0 0.0 2.0 1
#periodic boundary conditions (1=yes, 0=no), box size (A), nonbond cutoff (A), list cutoff (A),
#dielectric constant and type (0=constant dielectric, 1=distance-dependent dielectric) -- these
    must match the tables #being used!
#the above are the recommended options taken from the FACTS documentation for CHARMM
backbone 0.50 20
#monte carlo move type, fraction, and size
sidechain 0.25 45
backrub 0.25 20
end #of monte carlo moves
dcd/gb1-hairpin-0.1-10-30-0.2-20-30-0-0-20-1.dcd dcd/gb1-hairpin-0.1-10-30-0.2-20-30-0-0-20-1.dat
    rest/gb1hairpin-0.1-10-30-0.2-20-30-0-0-20-1.txt
#DCD trajectory file (coordinates), data file (centers and quaternions), restart file to write
/home1/dzuckerman/spiriti/tabulation2/protein-set/tables-frank3/tables-0.1-10-15-0.1-10-15-0-0/%s-%
    s.dat
#(this last line needs to be included only if tables are used) tables file name (must include two %
    s's).
```

The trajectory is output in DCD file format. It is necessary to generate a PSF file using CHARMM or the PSF builder in NAMD in order to view them.

A real number between 0 and 1 may also be specified for "use of interaction tables" in the first line of the script. In this case the program will use a potential function of the form $U = (1-\lambda)U_{exact} + \lambda U_{tabulated}$ where λ is the number specified.

The `%s` markers in the fragment and table file name specifications are filled in by the program with fragment type names. The file names of the actual files must be all lowercase.

3.1 Replica exchange simulations

The program may also be compiled to do temperature or resolution exchange simulations using the `EXCHANGE` compilation directive (see above). In this case the program takes two arguments: the input file and a format string for the output file with one `%d` that indicates the replica number.

```
tablemc-exchange <input-file> <output-file>-%d.out
```

the input file looks like this:

```
defs-protein4-charmm.txt fragments3-scaled/%s.xyz charmm19-scaled.prm 1 0
ACE GLY GLU TRP THR TYR ASP ASP ALA THR LYS THR PHE THR VAL THR GLU CBX
END
```

```
restart rest/gb1-hairpin-0.1-10-30-0.2-20-30-0-0-20-0.txt gb1-hairpin-min.pdb
100000000 1000 10000 100000000 0 0 300.00
0 0.0 12.0 0.0 2.0 1
backbone 0.50 20
sidechain 0.25 45
backrub 0.25 20
end
dcd/gb1-hairpin-0.1-10-30-0.2-20-30-0-0-20-1.dcd dcd/gb1-hairpin-0.1-10-30-0.2-20-30-0-0-20-1.dat
    rest/gb1-hairpin-0.1-10-30-0.2-20-30-0-0-20-1.txt
/net/roos/home4/zuckerman/spiriti/tabulation2/protein-set/tables10/tables-0.1-10-30-0.2-20-30-0-0/%
    s-%s.dat
dcd/gb1-hairpin-rex-%d-2.dcd dcd/gb1-hairpin-rex-%d-2.dat rest/gb1-hairpin-rex-%d-2.txt
#the above is the same as previous, except the trajectory and restart files must contain one "%d"
    which is replaced by the replica number
10000 rex-trpzip4-log-2
#replica exchange freqency and log file name (produces information on exchange probabilities)
1 exact 0 300
#each line: replica number, table set name (must contain two "%s"), lambda (0 for exact, 1 for full
     use of tables), temperature in K
2 exact 0 325
3 exact 0 352
4 exact 0 382
5 exact 0 414
6 exact 0 448
7 exact 0 486
8 exact 0 526
9 exact 0 570
10 exact 0 618
11 exact 0 669
12 exact 0 725
13 exact 0 786
14 exact 0 852
15 exact 0 923
16 exact 0 1000
```

To do resolution exchange the `exact` strings in the above script would be replaced by the name of a table set.

4 Interaction tables

The interaction tables were originally designed for a fully pairwise energy function (such as a constant or "distance-dependent" dielectric). Current work focuses on redesigning them to accommodate a generalized Born model. Therefore, they are not to be used for implicit solvents at the moment.

4.1 Table structure

The table consists of the interaction energy (van der Waals and electrostatics combined) as a function of the translational displacement between two fragments, expressed in terms of spherical coordinates (r, θ, ϕ) and the relative orientation of the two fragments, given by Euler angles (ϕ', θ', ψ'). The radius r is constructed on an exponential grid, with the resolution becoming coarser as we move away from the origin, defined by the following:

$$\begin{aligned} r_{n+1} &= f r_n \text{ where } f = 1 + \frac{\Delta r}{r_0} \qquad (1) \\ r_n &= f^n r_0 \\ n &= \text{round}\left(\frac{\log r_n/r_0}{\log f}\right) \end{aligned}$$

Note that Δr (which is what is specified) represents the *minimum* resolution (that is, the resolution at the minimum radius r_0).

The grids for the angular variables are constructed as follows:

$$\begin{aligned}
\theta_n &= \left(n + \frac{1}{2}\right)\Delta\theta && (2)\\
\phi_n &= n\Delta\phi && (3)\\
\phi'_n &= n\Delta\phi' && (4)\\
\theta'_n &= (n + \frac{1}{2})\Delta\theta' && (5)\\
\psi'_n &= n\Delta\psi' && (6)\\
& && (7)
\end{aligned}$$

where $\Delta\theta = \Delta\phi$ is the angular resolution of the table and $\Delta\phi' = \Delta\theta' = \Delta\psi'$ is the orientational resolution. I have adopted the practice of describing the resolution of a table compactly by giving the radial, angular, and orientational resolutions. For example, a "0.1-10-15" table has $\Delta r = 0.1$Å, $\Delta\theta = \Delta\phi = 10°$, and $\Delta\phi' = \Delta\theta' = \Delta\psi' = 15°$. Note that such a table does not have 0.1 Å resolution everywhere, but only at the minimum radius, and will have a proportionately coarser resolution at the maximum radius.

Although a table is a six-dimensional array of energy values, it must be stored linearly in memory. Rather than using multidimensional arrays in the code, the index into the table is calculated from the various indices n_r, n_θ,etc. by the foloowing formula:

$$n_{overall} = ((N_{\theta'} n_{\phi'} + n_\theta)N_{\psi'} + n_{\psi'})N_r N_\theta N_\phi + (N_\phi n_\theta + n_\phi)N_r + n_r \qquad (8)$$

Thus the ordering of dimensions from largest memory stride to smallest is $(\phi', \theta', \psi', \theta, \phi, r)$. The index into the table for any combination of indices in each dimension can be calculated using the member function `table::calculate_index` in `tables.h`.

Table files are binary files and consist of a header defined by `struct table_header` in `tables.h`, followed by a list of energy values. This structure contains both important information about the table resolutions (numbers of points in each dimension, resolutions, etc.) and descriptive information about how it was generated. For memory reasons the energy values are stored as single precision numbers; all calculations are performed using double precision arithmetic.

4.2 Smoothing

Smoothing is accomplished by convolving the Boltzmann factor $\exp(-\beta U)$ by a Gaussian-shaped kernel function:

$$V_i \exp(-\beta U'_i) \sum_j w_{ij} = \sum_j w_{ij} V_j \exp(-\beta U_j) \qquad (9)$$

where the sums j are over the neighbors of cell i, V_i and V_j are the volumes of cells i and j.

The smoothing kernel w_{ij} for the angular coordinates is given by

$$w_{ij} = k_{sph}(\gamma_{ij}) = \sum_{l=0}^{l_{max}} \sqrt{\pi(2l+1)} Y_{l0}(\gamma_{ij}) \exp\left(-\frac{l(l+1)\gamma_0^2}{4}\right) = \sum_{l=0}^{l_{max}} \frac{2l+1}{2} P_l(\cos\gamma_{ij}) \exp\left(-\frac{l(l+1)\gamma_0^2}{4}\right) \qquad (10)$$

where $\gamma_{ij} = \cos\theta_i \cos\theta_j + \sin\theta_i \sin\theta_j \cos(\phi_i - \phi_j)$ is the angular spherical distance between cells i and j. $Y_{l0}(\theta)$ is a spherical harmonic (which is independent of the azimuthal angle ϕ) and is equal to $\sqrt{\frac{2l+1}{4\pi}} P_l(\cos\theta)$, where $P_l(x)$ is the l-th order Legendre polynomial.

The smoothing kernel for the Euler angles is given by

$$w_{ij} = k_{orient}(\chi_{ij}) = \sum_{j=0}^{j_{max}} \frac{2j+1}{8\pi^2} \exp\left(-\frac{j(j+1)\chi_0^2}{4}\right) \frac{\sin\left(j+\frac{1}{2}\right)\chi_{ij}}{\sin\frac{1}{2}\chi_{ij}} \qquad (11)$$

where χ_{ij} represents the overall angle of rotation involved in going from orientation i to orientation j, and is given by $\cos \frac{\chi_{ij}}{2} = \mathbf{q}_j \mathbf{q}_i^{-1}$, where $\mathbf{q}_i$ and $\mathbf{q}_j$ are quaternions that represent the respective orientations.

The values of the parameters γ_0 and χ_0 represent the angular scales on which smoothing takes place in the angular coordinates (θ, ϕ) and the orientational coordinates (ϕ', θ', ψ') respectively.

4.3 Generating interaction tables

Interaction tables require fragment geometries and a force field file (suitable files may be found in the `table-generation` folder). The format of the input file is as follows:

```
2.0 12.0 0.1 10 15
#minimum and maximum radius in A, radial, angular, and orientational resolution
2.0 1 300.00 10 10 0 0.0 #dielectric constant, distance dependent dielectric (1=yes, 0=no),
#smoothing temperature (K), angular smoothing scale (degrees),
#orientational smoothing scale (degrees), ignored values
fragments3-scaled/peptide.xyz #first (reference) fragment file
fragments3-scaled/peptide.xyz #second fragment file
charmm19-scaled.prm #force field
```

The table can be generated through the use of the command

```
./tablemc generate control.txt tables-0.1-10-15-0.1-10-15-10-10/peptide-peptide.dat
```

specifying the names of the input file and table file to be generated. Alternatively, the `generate-tables` script may be used to automate the generation of tables for a large number of fragment types.