Supercoiled DNA Monte Carlo Simulation : Guidelines for Use

Brad Krajina

December 16, 2019

---

1 Introduction

This simulation package performs Monte Carlo simulations of a polymer modeled as a wormlike chain with twist for simulations of supercoiled rings. The simulation allows for the possibility of simulating polymer chains with both linear and ring topology. To improve sampling, the simulation allows for replica exchange Monte Carlo in which each replica corresponds to a distinct linking number. Replicas are simulated in parallel between exchanges using openMPI.

2 Requirements

This simulation package has been tested on Ubuntu 18.04 and is written primarily in FORTRAN 95 with some FORTRAN 2003 support. The following packages/libraries are required:

- gfortran compiler v. 5 or greater
- openMPI v. 2.01
- BLAS linear algebra library (FORTRAN)
- LAPACK linear algebra library (FORTRAN)

3 Using the Simulation package

3.1 Inputs

All inputs for the simulation are read from the **input** folder. Two files are read from: **input/input** and **input/Lks**. The former is always read from. The latter is read from only when parallel tempering with respect to linking number is performed. The input file should be written as a set of key-value pairs, separated by a space. Comments can are preceded by '#'. The ordering is not important. The following inputs should be provided (note that default values are specified in **src/SIMcode/params.f03** if values are not provided).

- **L** – (double precision) the contour length of the polymer (in units of your choice)
- **LP** – (double precision) the persistence length of the polymer (in units of your choice)
- **LT** – (double precision) the twist persistence length of the polymer (in units of your choice)
- **NB** – (integer) number of "beads" used to represent the polymer
- **RING** – (1 or 0). Is polymer a ring?
- **TWIST** – (1 or 0). Include twist energy?

- **LK** – (integer). Polymer linking number. Relevant only for rings with twist. Note that if parallel tempering is used, this value will be overwritten by the 'LKs' file.
- **INTON** – (1 or 0) Include polymer self-repulsion interactions?
- **VHC** – (double precision) polymer steric repulsion strength (in units of k_BT).
- **LHC** – (double precision) polymer self-repulsion interaction diameter
- **INDMAX** – (integer) number of polymer configurations to save
- **NINIT** – (integer) number of initialization monte carlo simulations to perform
- **PTON** – (T or F). Is parallel tempering on?
- **NSTEP** – (integer) number of Monte Carlo steps to perform between save points (if parallel tempering off), or number of Monte Carlo steps to perform between replica exchange attempts (if parallel tempering is on)
- **NREPLICAEXCHANGE** – (integer) number of replica exchange attempts to perform between saving polymer configuration

  If parallel tempering is included (PTON = T), then an additional file 'LKs' is necessary. This is simply a text file in which each row contains a value of the linking number to be included in the set of simulations (in order of increasing linking number). A simulation will be launched using openMPI in which each process corresponds to a linking number, with periodic attempts to exchange polymer chain configuration between linking numbers.

3.2 Outputs

All saved outputs are stored in the **data** folder. Currently, the polymer bead position matrix $\mathbf{r}x$ is saved at each save point, together with the set of orientation vectors $\mathbf{u}x$, where x is the save point number. In addition, at each save point, the simulation will update a new row in the **EELAS**, **Eint**, **wr**, **rgsq** files, which store elastic energy terms, repulsive interaction energy terms, the polymer write, and the squared radius of gyration, respectively. Each row corresponds to a save point. For 'EELAS', each column corresponds to one of the elastic energy terms. At the end of the simulation, summary statistics for the standard deviation (STDEV), standard error of the mean (STDERR), and the arithmetic mean (AVG) are saved for the write and squared radius of gyration. If parallel tempering is on, outputs will be directed to the appropriate folder within 'data' named 'LK_x' where x is the value specified for the linking number. In addition, at the highest level of the data directory, a file **pSWAPup** is saved, which stores the probability for each replica to exchange with the replica one linking number above it (except the highest linking number replica). If parallel tempering is not on, configurations are saved directly in the highest level of **data**.

3.3 Running a simulation

To run a successful simulation, it is essential that the program is launched using a number of threads for openMPI that is consistent with the parallel tempering settings. The number of threads specified with **mpirun** must be 1 greater than the number of linking numbers used in the parallel tempering scheme. The program will exit with an error if this does not hold. If parallel tempering is not used, the executable should be run directly (i.e. not using **mpirun**).

A script **'compile.sh'** is included to compile the program with mpifort and generate an executable **wlcsim**. If using parallel tempering with $n-1$ replicas, the executable can be run with the command:
mpirun -np n **./wlcsim**
where n is an integer. An example script that compiles the code and runs the simulation with openMPI using a hard-coded number of processes is available in **'runwlcsim.sh'**. Note that in order for this to

compile correctly on your system, you must link to the correct path for the liblapack.so library, which may differ from what is currently provided in the script.

Alternatively, it may be desirable to launch the simulation from a higher-level script that ensures that the data directory is properly set up, the input files are configured, and the program is launched with the proper number of processors. An example python script that performs this is available in **simrun.py**. The script uses a template input file, **input/template** to generate a new input file in which only the specified parameters are changed to new values. Unspecified parameters present in the original template are retained. The script also cleans the data directory (by moving old data to **trash** and ensures that folders are created for the appropriate linking numbers that the simulation is to be run at.

3.4 Visualizing simulation results

In the **visualization** folder, a couple of example scripts are available for generating .pdb files from the output and visualizing them in pymol. The script **visualization/r2pdb.py** contains a function for transforming a text file containing the polymer bead positions into a pdb file. Example scripts for generating a set of pdb files are available in **visualization/pdb_gen.py**. A pymol script for simple visualization is available in **visualization/mksnap.pml**, which can be run from the pymol command line using the command:
@mksnap.pml