Supercoiled DNA Monte Carlo Simulation : Guidelines for Use

Brad Krajina

November 23, 2016

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.

2 Requirements

This simulation package has been tested on Ubuntu 14.01 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' and '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. The ordering is not important. The following inputs should be provided (note that default values are specified in 'src/SIMcode/params.f03' is 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 $k_B T$).
- **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 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 'r' is saved at each save point, together with the set of orientation vectors 'u'. 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. 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. If parallel tempering is not used, the executable should be run directly. Otherwise, the number of threads used in openMPI should 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.

A script for compiling the simulation code **'compile.sh'** is included to compile the program with mpifort at the highest level of the simulation directory and generate an executable **wlcsim**. 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'**.

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. The script also cleans the data directory, 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 **'/movie'** folder, a couple of example scripts are available for generating .pdb files from the output and visualizing them in pymol. The script **'/movie/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 **'/movie/pdb_gen.py'**. A pymol script for simple visualization is available in **'mksnap.pml'**, which can be run from the pymol command line using the command '@mksnap.pml'.