User guide for polymer lattice simulation code

Zhouyi He[1] and Tyler Harmon[1]
[1]*Leibniz-Institut für Polymerforschung, Institut Theorie der Polymer, 01069, Dresden, Germany*
(Dated: December 11, 2024)

Tutorial description

Hello, welcome to the lattice polymer world, hope you enjoy it. After going through this tutorial, you would have a general picture of how this codes works and how to work with it. Specifically, we will model the phase separation of a system made of a star polymer and a linear polymer. You will understand the keywords well enough to run your own simulations without issues.

What is the code doing

This code is Monte Carlo lattice simulation engine for polymers. It models your polymers as a set of beads connected with linkers on a lattice. The beads move around and interact with beads on adjacent lattice sites. For well-behaved at equilibrium phase separating systems you will see a single large cluster (droplet). The code calculates and prints the most interesting (to us) analyses while it is running. As a general rule of thumb, the simulations take something on the order of one day to run.

One of the most important things to understand is how the beads interact. Beads can form 'transient bonds' with neighboring beads. Beads are either bonded to nothing or bonded to one other bead. Beads can never be bound to more than one bead at any given time. This gives a nuanced impact of bond energy. "0" and positive bond energies still correspond to attractions between beads. Only positive infinity energy corresponds to no attraction. This is a subtle point and its origin isnt obvious but comes from a counting entropy. As such there is an exception written into the code where putting zero in the interaction table is interpreted as +infinity. All other numbers close to zero behave moderately differently.

Technical details of the simulations moves

The interactions for the beads are modeled as 'transient bonds' with neighboring beads. Beads are either bonded to one other bead or unbonded. Beads can never be bonded to more than one bead at any given time. The strength of such pairwise bond is characterized by the bond energy.

For each step one of seven moves are attempted. They were designed based on what we thought would get the system to equilibrium as fast as possible without consideration for realistic dynamics.

- **Rotation move**
  Randomly choose one bead. If it has a bond with another bead, break this bond. Randomly form a new bond with an unbonded neighboring bead or remain unbonded.

- **Translation move I**
  Randomly choose one bead. If it has a bond with another bead, break this bond. Translate this bead to another position, within geometrical constraints. Randomly form a new bond with an unbonded neighboring bead or remain unbonded.

- **Translation move II**
  Randomly choose one bead.
  -If it has a bond with another bead, translate this pair to another position, within both of their geometrical constraints.
  -If it is unbonded, translate this bead to another position, within geometrical constraints.

- **Slither move I**
  This is only used for homopolymers. Randomly choose one polymer. Remove one bead from one end of the

polymer and add a bead to the other end. For the new bead, randomly form a new bond with an unbonded neighboring bead or remain unbonded. (Equivalent to the polymer slithering 1 bead.)

- **Slither move II**
  This is only used for linear polymers and polymers with the identical linkers. Randomly choose one polymer. Remove x ($x < 5$) beads from one end and add x beads to the other end. Break all the bonds the polymer has. For each bead in the polymer, randomly form a new bond with an unbonded neighboring bead or remain unbonded. (Equivalent to the polymer slithering x beads.)

- **Cluster move I**
  Randomly choose one cluster, defined as all polymers that are connected through bonds. Translate the cluster to another position anywhere in the lattice.

- **Cluster move II**
  Randomly choose one polymer. Calculate the cluster that it is in. If the cluster has fewer than 6 polymers, translate the cluster to another position anywhere in the lattice.

Note that the simulation is written in a slightly odd way where it does "Inner loop number of moves" then increases the "Outer loop of moves" by one. The "outer loop of moves" is used for how often it prints the different analysis routines. Altogether it does "inner loop" times "outer loop" number of steps.

Setting up a simulation

Setting up a simulation is relatively straight forward but care must be made to ensure you are simulating what you were hoping to simulate. In short you will follow the same procedure each time

- Define the number of types of polymers, the number of each of those types, and the number of beads each of those polymers has.
- Define the structural architecture of those polymers (if they are linear, star, or branched... most of the times you will want linear but the tutorial includes a star polymer so you can see what to do or not to do).
- Define how the beads interact
- Define the initial conditions, either all random or a subset in a sphere or slab
- Define the simulation length and what analyses you want performed

These specific details are all defined in the keyfile or input files linked explicitly in the keyfile. **The formatting of the files is strict and has no margin for error.** The keyfile is always
keyword
value
Here extra spaces are ignored. If multiple values are expected then they are space separated. Following the tutorials keyfile formatting will work. **Note, all energy scales must be integers and are in units of mKT (ie KT/1000).**
In the following sections we are much more explicit about the meaning and formatting of the different files.

Installation and compiling

GFortran is needed as this code is written based on Fortran90. For example, you can follow the instructions here: Installing GFortran.
You compile it using the terminal with the command:
gfortran G_5.74.f90 − O3

Input files

Before you run the simulation, make sure all necessary input files prepared in correct format, which includes:

- **Run Key File (y_runkey.txt)**
  This file contains all initial set-up for the simulation.
- **Polymer Connectivity File (z_Connection.txt)**
  This file describes the structure of the polymers
- **Energy File (z_Energy.txt)**
  This file describes how the beads of the polymers interact with each other.
- **Spring Linker Potential File (z_SpringLinkerEq.txt)**
  This is only needed under some special circumstances. This is turned on in **y_runkey.txt** when **SpringLinkerEq** keyword is *T* (True/On). This creates additional constraints in the polymer conformations.

Run

Run the code in the terminal with following command:
./**a.out y_runkey.txt**
Note: **y_runkey.txt** is specifically calling the runkey-file. The runkey-file contains keywords that points to the connection-file, energy-file and springlinker-file. The folder that these are in are denoted by including directories in the file names.

runkey-file

The Run Key File formatting is always given by a keyword on one line followed by its value below it. Here we list each keyword with a description of what it does and the value from the tutorial below it.

Tmax : outer loop: simulated steps used for counting and printing information
100000
Ti : inner loop: number of steps inside 1 step
10000
Note: altogether, there are 1e9 total Monte Carlo steps
BoxSize : x y z
100 100 100
NkN : number of unique polymer types
2
Nk : number of polymers of each polymer type
750 1500
Note: altogether, there are 2 types of polymers and a total of 2250 polymers to be simulated.
MaxSumNk : only used when Grand canonical ensemble is on. This sets the maximum number of allowed polymers.
1000 2000
Vk : number of beads that each molecule contains
5 5
Vak : number of interacting beads that each molecule contains (valency), only used under a niche analysis (WriteCoreCluster=T).
4 2
UseGrand : whether to use Grand canonical ensemble
F
LSteric : Effective persistence length in general
1

ESelf : Energy penalty for self interaction
0
Note: Polymers that can interact with themselves and others strongly prefer to interact with themselves because of polymer constraints. LSteric and ESelf are used to capture geometric constraints that proteins have but lattice polymers don't. In the code, the total energy are calculated as
ETC=ETC-floor((Energy(i,j)-ESelf)*exp(real(-(abs(i-j)-1)/LSteric)))
ChemPot : only used when Grand canonical ensemble is on
1000
EnergyFile
z_Energy.txt
ConnectionFile
z_Connection.txt
UseSpringLinkerEq : whether to add effective spring linker connection to certain beads pairs
T
E_SpringLinkerVector : print out the spring linker vector information at every 100 steps
100
SpringLinkerEqFile
z_SpringLinkerEq.txt
Note: Most of the times, this SpringLinkerEq potential should be turned off (UseSpringLinkerEq=F). It's left on here just for the completeness of the tutorial.
StartImported : whether to start simulation from an existing initial condition
F
Note: Typically this is used for restarting a simulation, potentially with different energies or other parameters. Using this is quite technical because everything must be written in an exact way. You can reliably do this by renaming renaming all the "End* files" to "Start* files". You can generate the "End* files" by setting WriteLatticeEnd=T.
StartDroplet : whether to start simulation with polymers inside a sphere
T
StartDropletR : radius of such sphere
20
StartDropletNk : number of molecules in sphere. Note the sphere must be big enough to hold all the molecules.
1000 2000
StartSlab : whether to start simulation with polymers inside a slab
F
StartSlabH : height of such slab, along z axis
30
StartSlabNk : number of molecules in slab
2000
WriteLatticeEnd : whether to write the last frame of lattice
T
iMoveRot : relative probability for this move to happen
1000
iMoveTransI
1000
iMoveTransII
1000
iMoveSlitherI
1000
iMoveSlitherII
1000
iMoveClusterI
1
iMoveClusterII
100
iMoveGrand
0
E_PrintStep : the interval of outer steps at which the current simulation step number is printed in the terminal

100
E_Sanity : the interval of outer steps at which sanity check is prformed
1000
E_Energy : the interval of outer steps at which the energy information is printed
1000
E_BoundSites : the interval of outer steps at which the energy information is printed
1000
E_Lattice : frequency of storing the entire lattice. Note that this is incredible information to be written, so use it only as needed.
1000
E_Network : frequency of storing all the connections for network analysis (no more than 20)
0
E_SelfLoop : frequency of storing self loops (no more than 20)
10
E_ClusterHist : frequency of storing histogram cluster size (no more than 20)
100
E_LargestCluster : frequency of storing largest cluster
100
WriteCoreCluster : whether to write the core cluster, which is defined as the cluster of polymers with relative occupancy more than half.
F
E_XYZDist_All : frequency of storing the xyz dimension histogram of the whole system
10
E_RadDist_Cluster : frequency of storing the radial distribution of the largest cluster (from the center of mass (COM) of the droplet, used for spherical droplet)
0
E_XYZDist_Cluster : frequency of storing the xyz dimension histogram of the largest cluster. Note that at each frame, the COM of the cluster is shifted to the center of the simulation box to align the dense phase.
10
E_RG_All : frequency of storing radius of gyration of whole system
100
E_RG_Cluster : frequency of storing radius of gyration of the largest cluster
100
Seed : intial number for the random number generator. "0" will assign a new initial random number seed, and store it in the "B_Seed.txt" file. Exact same parameter set-up with same seed will give exact same trajectory.
0
END

- **Bead numbering convention**

  The beads are numbered in a sequential way and is one of the most common ways we screw up our own simulations. Polymers are numbered based on the order they are introduced in the key file. The beads are numbered sequentially for the first polymer, then the numbering continues for the next polymer type, and continues until all polymer types have been sequentially included. For example, the tutorial has a polymer of length 5 (star) and a polymer of length 5 (linear). The beads are labeled 1,2,3,4,5,6,7,8,9,10. 1-5 are for the first polymer, 6-10 are for the second polymer. This scheme is used for all the following files.

- **z_Connection.txt**

  The general format is two beads that are covalently connected then how far (in lattice distances) they can be separated on the lattice. The first example polymer is a star polymer with a central bead and four beads connected to it, like an X. The second example polymer is a linear polymer of the form abcde.

  1,2,1 : the connection is between bead 1 and bead 2, with a linker length of 1
  1,3,1
  1,4,1
  1,5,1 : the connection is between bead 1 and bead 5, with a linker length of 1
  6,7,2 : the connection is between bead 6 and bead 7, with a linker length of 1

7,8,2
8,9,2
9,10,2

- **z_Energy.txt**

  The energy table must be a sum(Vk) by sum(Vk) table of integers. Strongly interacting beads have negative numbers, very weakly interacting beads are slightly negative or positive, **and non-interacting beads are exactly zero**. Remember the units are milli-K_BT

  Below is an example of energy table when bead 2, 3, 4 and 5 would interact with bead 6 and bead 10 with energy of $-6.0K_BT$.
  0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 -6000 0 0 0 -6000
  0 0 0 0 0 -6000 0 0 0 -6000
  0 0 0 0 0 -6000 0 0 0 -6000
  0 0 0 0 0 -6000 0 0 0 -6000
  0 -6000 -6000 -6000 -6000 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0
  0 -6000 -6000 -6000 -6000 0 0 0 0 0

- **z_SpringLinkerEq.txt**

  The Sping Linker potential file is formatted as a list of distance potentials between beads. They list the two beads the potential is between, the harmonic ideal distance, and the potential strength, again in units of milli-k_BT.

  Below is an example of SpringLinkerEq input when there is a harmonic potential between bead 6 and bead 10 with energy of $0.999K_BT$. This potential would try to stretch the polymer to a configuration that distance between bead 6 and bead 10 is around equilibrium length 7 in 3D.
  6,10, 7, 999