MCparallelMPI

Quinn MacPherson, Shifan Mao

July 21, 2016

1 How to run

1. Download from get if you haven't already

```
git clone https://github.com/SpakowitzLab/MCparallelMPI.git
```

2. Install MPI, if you haven't already. Download from them openmpi web page and then on linux

```
tar -zxvf openmpi...
tar.gz cd openmpi...
./configure --prefix=/usr/local
sudo make all install
```

and then add the following lines to your etc/profile

```
PATH=$PATH:/usr/local/bin export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

3. Set appropriate input files [see below].

4. Now compile and run the program. I have the following lines in `MCparallelMPI/runwlcsim.sh` which I then run.

```
#/bin/bash
echo "Compile" rm MCparrll_out
cd code
# compile with mpi's fortran compiler
mpifort -c DATAcode/* mersenne_twister.f90
mpifort -c -fbounds-check -Wall -W -fmax-errors=5 -O5 SIMcode/*  MCcode/*
mpifort *.o -o MCparrll_out rm *.o
cd ..
mv code/MCparrll_out .
mkdir -p data
mv data/* trash/
echo "Now run"
# now run the output
# --prefix used to avoid changing path
mpirun -np 8 MCparrll_out
```

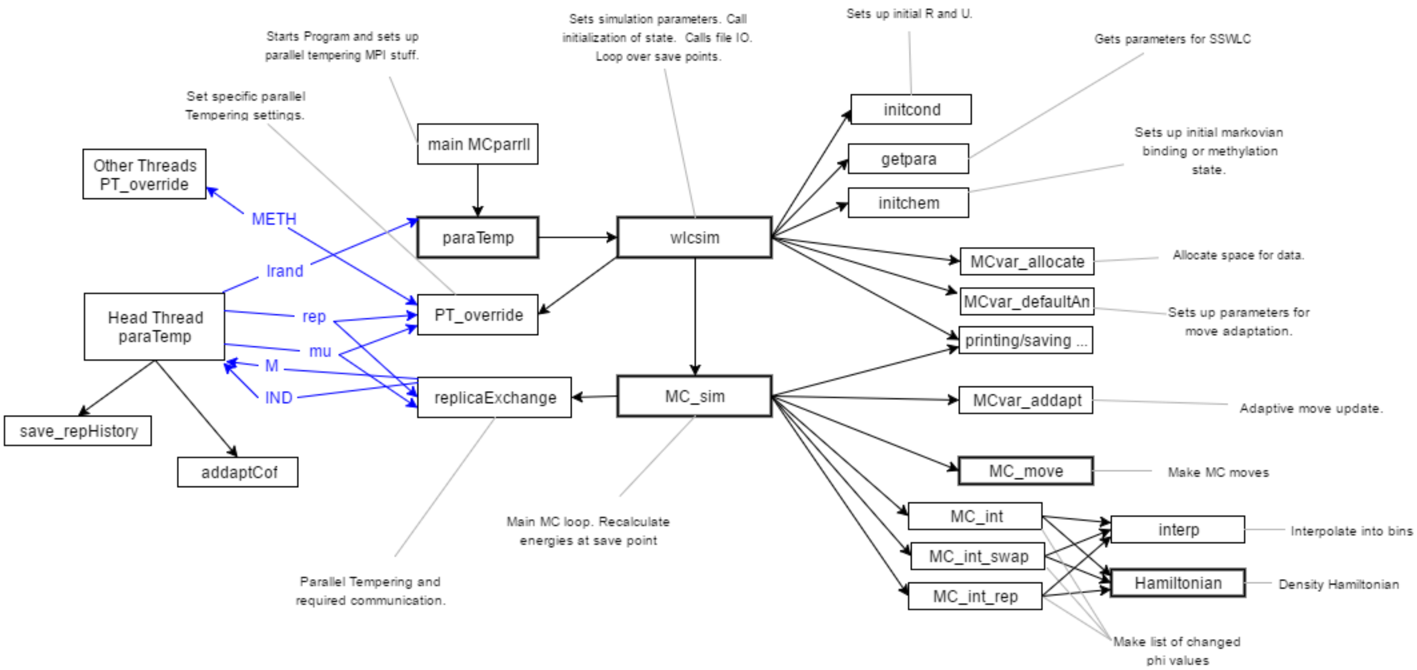
you will want to change the 8 in the last line to the number of threads [processes] you want. The number of replicas is one less than the number of threads unless there is only one thread in which case there is still one replica.

5. To run the job on the tower without having an interactive job going first run inside an interactive job as described above then type [ctrl]-z and then

```
disown -h %1
bg 1
```

in general these lines disown job 1 from your terminal and then run it in background. If do decide to run the program in background I suggest you pipe the output to a file.

2 Code Organization



3 MC Subroutines

3.1 MC_move

The purpose of MC_move is to update the proposed configuration RP, UP, and ABP based on a set of moves the obays the condition of ballance. Values of RP, UP, and ABP are only updated in the range IT1 to IT2 as these are the only values need to calculate the change in energy. In the case of the swap move the range IT3 to IT4 are also updated. The number IT* are run from 1 to the number of beads in the simulation while numbers IB* index the position of a bead within the polymer it exists inside of.

Warning: if you should decide to add another move be sure to update RP, UP, and ABP between IT1 and IT2 even if they are the same as R, U, and AB! Not doing so leads to errors which may be very difficult to find.

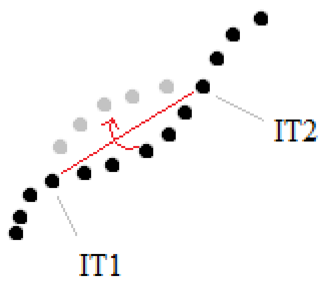
```
SUBROUTINE MC_move(R,U,RP,UP,NT,NB,NP,IP,IB1,IB2,IT1,IT2,MCTYPE &
                ,MCAMP,WINDOW,AB,ABP,BPM,rand_stat,winType &
                ,IT3,IT4)
use mersenne_twister
use setPrecision
IMPLICIT NONE
DOUBLE PRECISION, PARAMETER :: PI=3.141592653589793_dp ! Value of pi
DOUBLE PRECISION, intent(in) :: R(NT,3)   ! Bead positions
DOUBLE PRECISION, intent(in) :: U(NT,3)   ! Tangent vectors
DOUBLE PRECISION, intent(out) :: RP(NT,3)   ! Bead positions
DOUBLE PRECISION, intent(out) :: UP(NT,3)   ! Tangent vectors
INTEGER, intent(in) :: NB     ! Number of beads on a polymer
INTEGER, intent(in) :: NP     ! Number of polymers
INTEGER, intent(in) :: NT     ! Total beads in simulation
INTEGER, intent(in) :: BPM    ! Beads per monomer, aka G
INTEGER, intent(out) :: IP    ! Test polymer
INTEGER IP2   ! Second Test polymer if applicable
INTEGER, intent(out) :: IB1   ! Test bead position 1
INTEGER, intent(out) :: IT1   ! Index of test bead 1
INTEGER, intent(out) :: IB2   ! Test bead position 2
INTEGER, intent(out) :: IT2   ! Index of test bead 2
INTEGER, intent(out) :: IT3   ! Test bead position 3 if applicable
INTEGER, intent(out) :: IT4   ! Test bead position 4 if applicable
type(random_stat), intent(inout) :: rand_stat  ! status of random number generator

!      MC adaptation variables
```

```fortran
INTEGER, PARAMETER :: moveTypes=7 ! Number of different move types
DOUBLE PRECISION, intent(in) :: MCAMP(moveTypes) ! Amplitude of random change
INTEGER, intent(in) :: MCTYPE              ! Type of MC move
INTEGER, intent(in) :: winType
Double precision, intent(in) :: WINDOW(moveTypes) ! Size of window for bead selection
```

Crank-shaft move, MCTYPE=1

The crank-shaft moves a section of interior beads about the axes between them. The only change in elastic energy occures at the ends. The amplitude is given in radians and is chosen from a uniform distribution centered at zero. If applied to only a single bead the orientation vector for the bead is rotated.

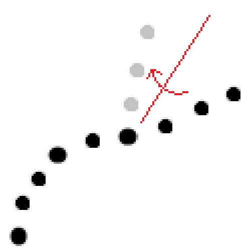


Slide move, MCTYPE=2

This move translates a section of the chain. The amplitude on this move must be small. However, it is necessary as the inter-bead spacing is not fixed. The distance moved in each direction is chosen from a uniform distribution centered at 0.

Pivot move, MCTYPE=3

The pivot move is like Crank shaft except that it allows the end of the chain to move. This is necessary in order to change the end-end distance. The axes is chosen randomly.



Rotate move, MCTYPE=4

The rotate rotates a single bead. This move is necessary to decouple U and R though this could be accomplished via translate and crank shaft. Turning off this move has been found to have little effect on the simulation.

Full chain rotation, MCTYPE=5

Rotates the entire chain about a random axes that runs through the central bead. If you would like the axes to run through the center of mass you would need to add that to the code.

Full chain translation, MCTYPE=6

As the name implies.

AB flip, MCTYPE=7

Exchanges $A \leftrightarrow B$ if the chain identity for all beads in the window. This move is necessary in the chromatin model where A/B represents binding state which can change but is unneccary for the co-polymer simulation.

Chain flip move, MCTYPE=8

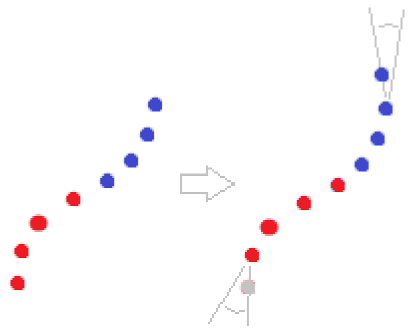
This move is intended to reverse the order of beads on the chain. It is currently out of order because the DSSWLC is not symmetric under this move making it a more complicated a less attractive move.

Swap chain move, MCTYPE=9

This move switchs the location of two polymers bead by bead. This is a useful move for random copolymers. This move suffers a poor acceptance ratio when inter-bead interactions become strong.

Reptation move, MCTYPE=10

The reptation move updates each bead to the location of the next bead along the chain [the direction along the chain is chosen randomly]. The angle that bead added to the end is located at is chosen so as to not change elastic energy



3.2 MCvar_adapt

`MCvar_adapt` adjusts move amplitudes and windows to maintain an acceptance rate of ~ 0.5. When the acceptance rate is too low the move windows and amplitudes are both decreased exponentially. The converse is true for high acceptance rates. Additionally the window is slowly adjusted toward a target windows size to maintain a reasonable window/amplitude ratio. Minimum and maximum windows and amplitudes are set. For many Monte-Carlo move the window size is actually the mean of an exponential distribution from which actual window sizes are chosen.

In order to allow for quick equilibration of the chain configuration it in necessary to choose a window size for the crank-shaft and pivot moves that is not too small. I often the minimum average window set to 6 beads for a chain of 40 beads.

3.3 MC_int

`MC_int` calculates the change in change in interaction energy between chains. It first calculates the change in coarse grained density and then calls Hamiltonian to calculate the change in energy. `MC_int` assumes that only beads between `I1` and `I2` have moved unless initialize is specified, in which case it calculates the total energy from `R` and `U`. The change in energy is stored in the `mc`.

```
SUBROUTINE MC_int(mc,md,I1,I2,initialize)
use simMod use setPrecision IMPLICIT NONE
TYPE(MCvar), intent(inout) :: mc   ! <---- Contains output
TYPE(MCData), intent(inout) :: md
LOGICAL, intent(in) :: initialize    ! if true, calculate absolute energy
INTEGER, intent(in) :: I1            ! Test bead position 1
INTEGER, intent(in) :: I2            ! Test bead position 2
```

3.4 Hamiltonian

The Hamiltonian subroutine calculates the change in interaction energy caused by a change in the density profile.

```
subroutine hamiltonian(mc,md,initialize)
use simMod
use setPrecision
implicit none
TYPE(MCvar), intent(inout) :: mc
TYPE(MCData), intent(in) :: md
logical, intent(in) :: initialize
```

When the initialize is set to True the entire energy will be calculated from scratch based on the PHI values. Note that this energy is put in the `dE_chi`, `dE_Kap`, etc variables rather than the `E_chi`, `E_Kap`, etc. variables.

The density part of the polymer melt Hamiltonian is:

$$H_{int} = \int d^3\mathbf{r} \left(\chi \phi_A \phi_B + \kappa \left(\phi_A + \phi_B - 1 \right)^2 - h\phi_A \right)$$

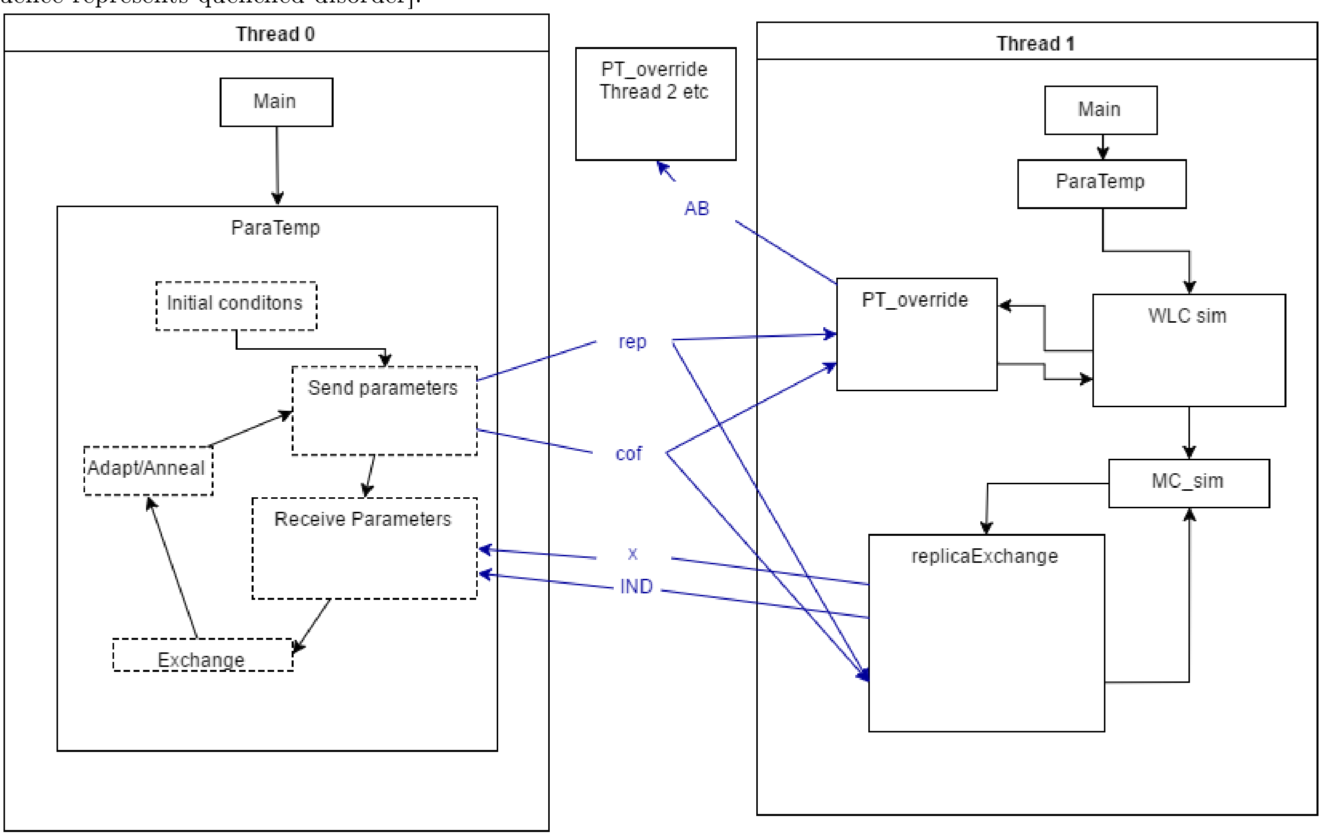
The density part of the hetero-chromatin Hamiltonian is[1]:

$$H_{int} = \int d^3\mathbf{r} \left(\chi \left(\phi_A + \phi_B \right) \left(1 - \phi_A - \phi_B \right) + J\phi_A^2 \right)$$

4 MPI Subroutines

The code is parallel in the sense that it occurs on multiple cpu's at the same time and it is also parallel in the sense that it uses a parallel tempering algorithm. A simulation which uses 48 computational threads will have 47 replicas as one thread is used as a head thread to organize replica exchanges. When two replicas exchange the treads swap coefficients [e.g. $\chi, \mu, T...$] while keeping the `R` and `U` data on the same thread; this dramatically reduces message passing time. The output is reorganized into replicas when it is written to output files.

At the start of the simulation each worker thread makes a single call to the subroutine `PT_override` which sets the initial condition. Thereafter, replica exchanges are made by periodically calling `replicaExchange`. `PT_overide` and `replicaExchange` mostly send and receive data from the head node except that `PT_override` on thread 1 sends the initial chemical configurational data to the other worker threads at the beginning of execution. In general the reads do [and should] start with different R and U configurations but need to have the same chemical configuration [we are working in the cononical ensemble, the random chemical sequence represents quenched disorder].



Main

Main sets up the mpi infrastructure and calls `ParaTemp`. If only one thread is used it will call `singleCall` and only one thread will be used. All threads execute main.

ParaTemp

Para temp splits up the threads into head and workers. The head node executes the parallel tempering algorithm for the head node, for example, accepting or rejecting replica exchanges. The worker nodes call `wlcsim`.

[1] Actually the hetero-chromatin Hamiltonian also has a kappa term but this can be thought of as a hard constraint if κ is large.

The code is designed to facilitate replica coupling in χ, μ, h and the other coefficients in the Hamiltonian. Though it is possible to replica couple over a grid of values in parameter space the code is currently set up replica couple over a path defined by `chi_path(s)` and `h_path(s)` where `s` is a the path-length parameter.

PT_override

This subroutine is called by a all the worker threads. It overwrites the values that the threads read from the input file with values assigned by the head node.

replicaExchange

This routine is called every `NPT` steps by the worker nodes. It sends the head node the variables, $x_{\chi/\mu/h}$, congregate to each energy coefficient. It then receives the respective energy coefficients corresponding the replica that the thread will be running until the next call to `replicaExchange`.

5 Input

5.1 Hard coded / argument settings

Most importantly the number of replicas is set in the MPI run command.

The replica path is defined in the functions `h_path`, `chi_path`, `mu_path`, `kap_path`, and `HP1_Bind_path` which are defined in the `MCparrallel_mpi.f90` file. Specific annealing patterns can be set by changing the file `schedule.f95`.

5.2 Initialization

The following files are read from the `input` directory.

h_A The file `input/h_A` contains applied field strength by bin indexed by

```
INDBIN=IX+(IY-1)*mc%NBINX(1)+(IZ-1)*mc%NBINX(1)*mc%NBINX(2)
```

This file will only be used if `FRMFIELD` is set to true.

ab This file contains the chemical composition by bead. Each bead should be on a separate line. This is the same format as the 4th column of the r*v* output. This file will only be used if the `FRMCHEM` option is set to true.

u0 This file contains the unit vectors associated with each bead. One bead per line with columns x y z. This file is only used if `FRMFILE` is set to true.

r0 This file contains the position of each bead. One bead per line with columns x y z. This file is only used if `FRMFILE` is set to true.

meth This file contains the methylation state by bead. Each bead should be on a separate line. This is the same format as the 5th column of the r*v* output. This file will only be used if the `FRMMETH` option is set to true.

dssWLCparams These are the parameter for the discrete shearable stretchable worm-like chain. They are generated by a Matlab script.

params See next section.

5.3 input/params

The simulation parameters are input via a keyword format. Keywords can be listed in any order, however if contradictory inputs are given typically the later will be used. If not listed default values will be used.

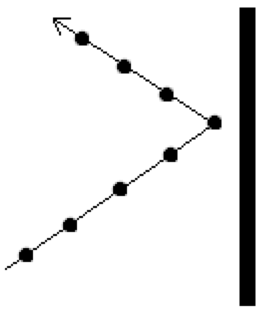
```
# example inputs
PTON T
L0 0.406666
DEL 1.0000
LBOX 20
```

5.3.1 File IO options

- FRMCHEM $\in \{T, F\}$. Read initial chemical sequence form file `input/ab`
- FRMMETH $\in \{T, F\}$. Read initial methylation sequence form file `input/meth`
- FRMFIELD. $\in \{T, F\}$. Read field h in from `input/h_A`.
- FRMFILE. $\in \{T, F\}$
- SAVE_U $\in \{T, F\}$. Save u vectors at every save point. You will need this if you intend to restart a simulation./
- SAVE_PHI $\in \{T, F\}$. Save density values at every save point.

5.3.2 Simulation options

- SETTYPE $\in \{1, 2, 3, 4\}$. This selects how the R and U values are initialized.
  - 1: Beads in straight line in y direction with random starting point for each polymer. Beads set at equilibrium distance
  - 2: Beads on each polymer start in a random position and placed along a randomly oriented line at equilibrium spacing. There is boundary in the z direction. When beads encounter the boundary the line is turned into a new random direction that keeps that puts the next bead within the confinement [see figure].
  - 3: Same as 2, except in a spherical boundary centered at `LBOX`/2



- CONFINE TYPE $\in \{0, 1, 2, 3, 4\}$. This selects the boundary shape.
  - 0: No confinement. Periodic boundary conditions in a cube.
  - 1: Between two plates in z direction located at 0 and `LBOX`
  - 2: Cube of size `LBox`.
  - 3: Circle of radius `LBox`, centered at `LBox`/2
  - 4: Periodic, unequal dimensions.
- RECENTER_ON $\in \{T, F\}$. Should the location of the chains be reentered should they drift out of the periodic boundary.
- SIMTYEP $\in \{0, 1\}$. Set the Hamiltonian and several associated choices.
  - 0: Polymer melt Hamiltonian used by Shifan.
  - 1: Hetero-chromatin in solvent simulation used by Quinn.

5.3.3 Geometry parameters

- L0 $\in \mathbb{R}_{>0}$. Equilibrium bead spacing.
- DEL $\in \mathbb{R}_{>0}$. Discretization of space. A.k.a. bin width. You should probably make this 1.0.
- LBOX $\in \mathbb{R}_{>0}$. Side length of cubic box.
- LBOXX $\in \mathbb{R}_{>0}$. Side length of box in x direction.
- LBOXY $\in \mathbb{R}_{>0}$. Side length of box in y direction.
- LBOXZ $\in \mathbb{R}_{>0}$. Side length of box in z direction.
- NP $\in \mathbb{Z}_{>0}$. Number of polymers.
- G $\in \mathbb{Z}_{>0}$. Number of beads in a monomer.

- N $\in \mathbb{Z}_{>0}$. Number of monomers in a polymer.
- FPOLY $\in \mathbb{R} : 0 < FPOLY \leq 1.0$. Fraction of total volume filled by polymer.
- V $\in \mathbb{R}$. Volume of bead.
- FA $\in \{\mathbb{R} : 0 < F_A \leq 1.0\}$. Fraction of beads of type A.
- LAM $\in \{\mathbb{R} : -1 < \lambda \leq 1\}$. Eigenvalues of markov chemical correlation matrix.
- LAM_METH $\in \{\mathbb{R} : -1 < \lambda_M \leq 1\}$. Eigenvalues of markov chemical correlation matrix for methalation.
- K_FIELD. $\in \mathbb{R}$. Wave-mode of applied field if not entered from file

5.3.4 Timing options

- NNOINT $\in \mathbb{Z}_{>0}$. Number of save points before turning on density interaction.
- N_KAP_ON $\in \mathbb{Z}_{>0}$. Number of save points before turning compression energy on.
- N_CHI_ON $\in \mathbb{Z}_{>0}$. Number of save points before turning χ repulsion energy on.
- STRENGTH_SCHEDULE. $\in \{T, F\}$. Use scheduled ramp in interaction strength(s) defined in `schedule.f95`
- INDMAX $\in \mathbb{Z}_{>0}$. Number of save points in simulation.
- NSTEP $\in \mathbb{Z}_{>0}$. Number of steps per save point.
- NPT $\in \mathbb{Z}_{>0}$. Number of save points between parallel tempering attempts.
- N_REP_ADAPT. $\in \mathbb{Z}_{>0}$ number of attempted replica exchanges between adaption/annealing of coefficient spacing.
- IND_START_REP_ADAPT. $\in \mathbb{R}_{>0}$. Save point at which to start adapting replica exchange. Most likely you should set this two the time when you turn on the associated term in the Hamiltonian, e.x. `N_CHI_ON`.
- IND_END_REP_ADAPT. $\in \mathbb{R}_{>0}$. Save point at which to stop adapting replica exchange. If you make this too small the system will not have time to adapt/anneal replica coefficients. Only use data from save points after this as they are equilibrated.
- REP_ANNEAL_SPEED. $\in \mathbb{R}_{>0}$. Max speed at which to change replica coupling path length s [or χ if $\chi = s$]. For example, if $s = \chi$ and you want χ to be able to adapt over a range of say 2.5 between IND_START_REP_ADAPT and IND_END_REP_ADAPT with replica adapts occurring every NPT*N_REP_ADAPT moves you should set

$$\text{REP\_ANNEAL\_SPEED} > \frac{2.5 \cdot \text{NPT} \cdot \text{N\_REP\_ADAPT}}{\text{ISTEP}\,(\text{IND\_START\_REP\_ADAPT} - \text{IND\_END\_REP\_ADAPT})}$$

5.3.5 Energy parameters

- EPS $\in \mathbb{R}_{>0}$. Elasticity of chain. $\epsilon = l_0/2l_p$
- CHI $\in \mathbb{R}$. Flory Huggins χ parameter.
- H_A $\in \mathbb{R}$. Strength of applied external field, h.
- KAP $\in \mathbb{R}_{>0}$. Compressability energy coefficient, κ. Related to bulk modulus.
- EU $\in \mathbb{R}$. Energy of binding for unmethalated monomer. Positive for more favorable binding.
- EM $\in \mathbb{R}$. Energy of binding for methalated monomer. Positive for more favorable binding.
- MU $\in \mathbb{R}$. HP1 chemical potential, μ.
- HP1_couple $\in \mathbb{R}$. Coupling energy. Negative for favorable coupling.

5.3.6 Move options

- CRANK_SHAFT_ON $\in \{0, 1\}$. Is crank shaft move on.
- SLIDE_ON $\in \{0, 1\}$. Is slide move on.
- PIVOT_ON $\in \{0, 1\}$. Is pivot move on.
- ROTATE_ON $\in \{0, 1\}$. Is rotate move on.
- FULL_CHAIN_ROTATION_ON $\in \{0, 1\}$. Is full chain rotation move on. Shut off if you have very long polymers in confinement.
- FULL_CHAIN_SLIDE_ON $\in \{0, 1\}$. Is full chain slide move on. Shut off if you have very long polymers in confinement.
- BIND_MOVE_ON $\in \{0, 1\}$. Is bind/unbind move on. This is for hetero-chromatin.
- CHAIN_FLIP_MOVE_ON $\in \{0, 1\}$. Is chain flip move on. This move is currently out of order.
- REPTATION_MOVE_ON $\in \{0, 1\}$. Is reptation move on.
- WINTYPE. Set this to 1.
- *_TARGET. $\in \mathbb{R}_{>0}$. Target window size of move *.
- MIN_*_WIN. $\in \mathbb{R}_{>0}$. Minimum average window size of move *.
- REDUCE_MOVE. $\in \mathbb{Z}_{\geq 1}$. How often do move types 5 and 6 should their acceptance rates fall below `MIN_ACCEPT`.
- MIN_ACCEPT. $\in \{\mathbb{R} : 0 \leq ... \leq 1\}$. Minimum acceptance rate below which moves 5 and 6 are reduced.

5.3.7 Replica settings

- PTON $\in \{T, F\}$. Do parallel tempering.
- LOWER_REP_EXE. $\in \{\mathbb{R} : 0 \leq ... \leq 1\}$. Minimum acceptable replica exchange rate.
- UPPER_REP_EXE. $\in \{\mathbb{R} : 0 \leq ... \leq 1\}$. Maximum acceptable replica exchange rate. Best to keep below 0.23.
- LOWER_COF_RAIL. $\in \mathbb{R}_{>0}$ minimum spacing on replica path coefficient s. [If $s = \chi$ then it is the minimum χ spacing].
- UPPER_COF_RAIL. $\in \mathbb{R}_{>0}$ maximum spacing on replica path coefficient s. [If $s = \chi$ then it is the maximum χ spacing].
- REPLICA_BOUNDS. $\in \{T, F\}$. Restricts the replica coupling path length parameter to the range $0 \leq s \leq 1$
- INITIAL_MAX_S. $\in \{T, F\}$. Maximum value of s at beginning of simulation. Other replica values will be space below it. If you make this too small it will take adaptation a long time to exponentially increase s.

6 Output Format

The output files are save in the data/ directory. They are moved to trash if the program is re-run. Unless stated otherwise replicas are

r*v* The files such as, `r15v6`, contain the cartesian coordinates of beads at save point 15 and replica 6. Notice that this replica 6 not thread 6. The column stand for x, y, and z while the rows stand for bead index with runs from 1 to `NT`, the total number of beads in the system.

If there is a 4th column in r than it is the AB chemical identity of the bead. If there are 5 columns [for the hetero-chromatin problem] the 5th is the methylation state and the 4th is the binding state.

The units of distance are the same units that `DEL` and `L0` are given in.

u*v* These are the unit vectors associated with each bead [sort of like a course grained tangent vector]. Same format as r*v*.

chi Each row is a later MC time. A line is printed at every `mc%NRepAdapt * mc%NPT` Monte-Carlo steps. The first column is save-point index, `IND`. The remain columns contain the χ values for the replicas in order of replica. You should only trust data after the χ values have stopped changing. The columns are given in terms of increasing s, the replica path length variable.

h_A Same format as chi except it is the field strength instead of chi.

mu Same as chi except for binding chemical potential for h.

nodeNumber Each row is a later MC time. A line is printed at every `mc%NRepAdapt * mc%NPT` Monte-Carlo steps. The fist column of node number is the save-point index, `IND`. The columns correspond to the replicas. The numbers are the threads that are running that particular replica.

out1v* The out1 files contain the energy values and coefficients as labeled in the first line. Rows correspond to save points. The number following the v in the file name corresponds to replica.

out3v* This file contains the adaptation data for MC moves by move type. The number following the v in the file name corresponds to replica. `IND` stands for index. id stands for thread index. `WIN` stands for window. `AMP` stands for amplitude of the move in distance. SUC stands for success rate of the Monte Carlo move. The numbers after `WIN`, `SUC`, etc. refer to `movetype`.

error Contains errors and warnings.
If you see the error:

```
Warning. Integrated binding enrgy: 0.0000000000000000 while absolute binding energy:
6071.6799999999957
```

This means that the some of changes in energy didn't add up to the total energy when it was recalculated from scratch. If the first number is zero you can probably ignore it as this will occur at the beginning of the program or when you turn a term in the Hamiltonian. If the first number isn't zero you should probably investigate the reason why.

paramsv* This file prints some of the parameters for the corresponding replica. This hasn't been kept up to date.

repHistory This is a human readable file used to diagnose how well replica coupling is working. `up` and `down` refer to the success rates or replica exchange up and down from that replica.

7 Tests

Coming soon...

8 Possible improvements

- The current pivot move rotates a bead about a random axes. If the axes is chosen as the u vector of the corner bead then the change in elastic energy could be eliminated. This would be in addition to the existing pivot move, not a replacement. Consider making this improvement if you are haveing difficulty sampling rigid polymers because of a low acceptance rate on the pivot moves.
- Transpose $R(NT, 3) \to R(3, NT)$. And same with `RP`, `U`, and `UP`. This will have better cashing properties. This may become profitable if the number of beads get very large.
- Make a hash table for `INDPHI`. This would speed up what is often the slowest part of the code. This will become necessary if the number of bins changed in a move get's large.