Documentation for shearableWLC code. Simulations using the dssWLC model.

E. F. Koslover, A. J. Spakowitz

Last updated June 15, 2013

This code can be used to run Brownian Dynamics or Monte Carlo simulations of the dssWLC model.

Contents

1 Compilation Instructions

To compile and run the program, you will need the following:

- a compiler capable of handling Fortran90. The code has been tested with the gfortran compiler.
- BLAS and LAPACK libraries installed in a place where the compiler knows to look for them
- Python (version 2.5 or higher) to run various auxiliary scripts (e.g., for visualization). The scripts have been tested with Python 2.6.4 only. You will also need the NumPy extension package.
- Recommended: PyMOL to visualize pdb files.

The code has been tested on Ubuntu Linux only.

To compile with gfortran, go into the `source` directory. Type `make`. To compile with any other compiler that can handle Fortran90, type

```
make FC=compiler
```

substituting in the command you usually use to call the compiler.

If the compilation works properly, the executable `shearableWLC.exe` will appear in the main directory.

2 Usage Instructions

To run the program in the main directory, type:

```
./shearableWLC.exe suffix
```

Here, `suffix` can be any string up to 100 characters in length. The program reads in all input information from a file named `param.suffix` where, again, `suffix` is the command-line argument. If no argument is supplied, it will look for a file named `param`. If the desired parameter file does not exist, the program will exit with an error.

The parameters in the input file are given in the format "*KEYWORD* value" where the possible keywords and values are described in Section 7. Each keyword goes on a separate line. Any line that starts with "#" is

treated as a comment and ignored. Any blank line is also ignored. The keywords in the parameter file are not case sensitive. For the most part, the order in which the keywords are given does not matter. All parameters have default values, so you need only specify keywords and values when you want to change something from the default.

detail in Section 6. Example parameter files for the different calculations are provided in the `examples` directory.

3 Examples for a Quick Start

Example parameter files are located in the `examples` directory

3.1 Example 1: equilibrium sampling

Sample a number of chain configurations from the equilibrium distribution. Run this example by typing

```
../shearableWLC.exe ex.equildistrib
```

This example uses a chain where the 2 edge segments on either side have a segment length of 0.1 and corresponding parameters, whereas all other segments have a segment length of 0.2. This will sample 10000 chains in total, and output end-to-end vectors in the first three columns of the output file ex.equildistrib.out. The next 3 columns give the first orientation vector. Snapshots of every 1000^{th} chain will be output in the `ex.equildistrib.snap.out` file, which can then be converted to pdb format as follows:

```
../snapshot2pdb.py ex.equildistrib.snap.out
```

The resulting pdb file (`ex.equildistrib.snap.pdb`) can be loaded into PyMOL and colored using the following commands within PyMOL itself.

```
load ex.equildistrib.snap.pdb, multiplex=0
../scripts/viewsnapshots.pml
```

3.2 Example 2: Brownian Dynamics

Run a Brownian Dynamics simulation for 2000 chains in parallel, keeping track of shear stress correlations in the file `ex.browndyn.stress.out`. The simulation starts from an equilibrium distribution and runs for 10000 steps.

3.3 Example 3: Looping first-passage time

Run a Brownian Dynamics simulation to track the first passage time to looping for 1000 chains in parallel. This example uses a chain where the 2 edge segments on either side have a segment length of 0.1 and corresponding parameters, whereas all other segments have a segment length of 0.2. The friction coefficients are supplied per unit length, which means the edge beads have half the friction of the 2-nd beads from the edge. Those in turn have half the friction of the inner beads (which correspond to longer segment lengths). The time to looping for each chain is output in the 3rd column of the file `ex.loop.loop.out` as each chain loops.

3.4 Example 4: Looping first-passage time, with dynamic rediscretization

Run a Brownian Dynamics simulation to track the first passage time to looping for 1000 chains in parallel. Rediscretize the whole chain dynamically by a factor of 4 (to segment lengths of 0.05) whenever the end-to-end distance goes below 0.4. Coarsen the chain whenever the end-to-end distance goes above 0.5. Parameters are interpolated from the file `dssWLCparams.txt`.

3.5 Example 5: Monte Carlo

Run a Monte Carlo simulation of 10^8 steps for a dssWLC. Output the end-to-end distance every 1000 steps into columns 5-7 of the file `ex.montecarlo.out`.

4 Auxiliary Scripts

4.1 Visualizing structures

The script `scripts/snapshot2pdb.py` will convert a snapshot file containing many chain configurations into a concatenated pdb file that can be loaded into pymol. Run this script without any arguments to get usage information. To load the resulting snapshots into different states to make a movie do "load snap.pdb, multiplex=0" in pymol. Color an visualize in pymol using `scripts/viewsnapshots.pml`.

5 Parameters for the dssWLC model

The Matlab code for calculating the energetic and dynamic parameters for the dssWLC model is located in `getparams/`. The relevant function for getting the parameters is `dssWLCcalcparams`

There are three approaches to getting the parameters. Firstly, as described in Koslover and Spakowitz, Soft Matter, 2013, one can optimize of the parameter $\alpha = \eta^2\epsilon_b/\epsilon_\perp$ to get the minimal length scale of accuracy. This can be done for a chain with Nseg segments of length $\Delta =$ del by running the following in Matlab:

```
[eb,gam,epar,eperp,eta,alpha,zetau,delt] = dssWLCcalcparams(del,Nseg)
```

These calculations can be very slow (taking on the order of 15 min). The first time this is run will be even slower as it needs to generate a tensor of coupling coefficients for spherical harmonics which will then by saved in a .mat file for future use. The size of the matrix used for calculating the structure factor is set with the optional parameter LMAX. Default value is LMAX=10. Smaller values of Δ require higher values of LMAX.

The output parameters are, in order, $\epsilon_b, \gamma, \epsilon_\parallel, \epsilon_\perp, \eta, \alpha, \zeta_{ub}, \delta t$. They should be input into the parameter files for the Fortran simulation code as follows:

EB	ϵ_b
GAM	γ
EPERP	$\epsilon_\perp + \eta^2\epsilon_b$
EPAR	$\epsilon_\parallel$
EC	$-\eta\epsilon_b$
FRICT	$1/(\text{Nseg}+1) \quad \zeta_{ub}$
DELTSCL	$\delta t/\zeta_{ub}$

(assuming that $\zeta_{ub} < 1/(\text{Nseg}+1)$).

Alternately, one can input the friction coefficients per unit length, which needs to be done in the case where not all segment lengths are equal. If using this format, then even if all segment lengths are equal the edge beads will have have the translational friction of the central beads.

$$\text{FRICT} \quad 1D0 \quad \zeta_{ub}(\text{Nseg}+1)/(\Delta\text{Nseg}) \quad T$$

Generally, if running a chain with different segment lengths, one should use the smallest segment length to determine the appropriate values of δt and ζ_{ub}.

Another approach to selecting the parameters is to use a specific value of α, which can be done as follows in Matlab for a value $\alpha = a$:

```
[eb,gam,epar,eperp,eta,alpha,zetau,delt] = dssWLCcalcparams(del,Nseg,'alpha',a)
```

This will run much faster and is generally more stable.

Finally, the recommended approach is to simply interpolate from the pretabulated values available in `getparams/dssWLCparams.txt`. This file lists on each line the values of $\Delta, \epsilon_b, \gamma, \epsilon_\parallel, \epsilon_\perp, \eta, \zeta_u, \delta t/\zeta_u$, respectively. The interpolation can be done via:

```
[eb,gam,epar,eperp,eta,alpha,zetau,delt] = dssWLCcalcparams(del,Nseg,'intepfile','
```

The tabulated file was generated using optimization over α for $\Delta = 0.01 - 1$, and a constant value of α thereafter for $\Delta = 1 - 4$. The interpolation file was generated using `getparams/tabulateparams.m`

6 Description of Specific Calculations

The *ACTION* keyword specifies what type of calculation will be done. The possible actions are EQUILDISTRIB, BROWNDYN, and MONTECARLO, as described below.

6.1 EQUILDISTRIB action

This generates a bunch of dssWLC chain configurations sampled from an equilibrium distributions. The sampling method is set by the *STARTEQUIL* keyword (2 types of rejection sampling or monte carlo). Number of chains sampled is set by *MCSTEPS* keyword. Dumping of snapshots is set by the *SNAPSHOTS* keyword. End-to-end vectors for all configurations are output into the file set by *OUTFILE*. Will also work with a Gaussian chain (see *GAUSSIANCHAIN* keyword) and a bead-rod chain (if *STRETCHABLE* and *SHEARABLE* are set to false).

6.2 BROWNDYN action

This runs a Brownian Dynamics simulation for a set of chains, keeping track of the shear stress correlation over time. Use *NCHAIN* to set the number of chains being run in parallel, *BDSTEPS* to set the total number of steps and the printing / output frequency. Use *LOOPING* keyword to tabulate first passage times for looping of chain ends. *FRICT* keyword sets friction coefficients. *DELTSCL* keyword sets the timestep as a multiple of the friction coefficients. Can use *STARTEQUIL* keyword to start from an equilibrated set of configurations. Can periodically dump out snapshots of chain configurations.

6.3 MONTECARLO action

Run a Monte Carlo simulation for a dssWLC chain. Only Monte Carlo of a single chain at a time has been recently tested. Set total number of steps and number of initialization steps with the *MCSTEPS* keyword. Set printint / output frequency with *MCPRINTFREQ*. Can output snapshots with *SNAPSHOTS* keyword. Used *ADJUSTRANGE* keyword to set how often step sizes are adjusted. For the most part, this has been supplanted by the equilibrium configuration sampling (EQUILDISTRIB action). However this general procedure can be implemented with additional complications (ie: chain meshes, non-local interactions) which EQUILDISTRIB cannot.

7 Keyword Index

The code will attempt to read parameters out of a file named `param.suffix` where "suffix" is the command line argument. If no command line arguments are supplied, it will look for a file named `param`. If multiple arguments are supplied, it will read multiple parameter files in sequence.

The parameter file should have one keyword per line and must end with a blank line. All blank lines and all lines beginning with # are ignored. For the most part, the order of the lines and the capitalization of the keywords does not matter. All keywords except *ACTION* are optional. The default values for each parameter are listed below. If a keyword is supplied, then values may or may not be needed as well. Again, the required and optional value types are listed below.

Keywords and multiple values are separated by spaces.

When reading the parameter file, lines longer than 500 characters will be truncated. To continue onto the next line, add "+++" at the end of the line to be continued. No individual keyword or value should be longer than 100 characters.

Floating point numbers can be formated as 1.0, $1.1D0$, $10e{-}1$, $-1.0E{+}01$, etc., where the exponential notation specifier must be D or E (case insensitive). Integer numbers can also be specified in exponential notation without decimal points (eg: 1000 or 1E3). Logical values can be specified as T, F, TRUE, FALSE, 1, or 0 (with 1 corresponding to true and 0 to false).

By default, all energy units are in kT.

- *ACTION*
  - value: 1 string of at most 20 characters; no default
  - This keyword sets the overall calculation performed by the program (see Sec.6)
  - Possible values are: MONTECARLO, BROWNDYN, EQUILDISTRIB
- *ADJUSTRANGE*
  - value: 1 required integer (ADJUSTEVERY), 3 optional floats (FACCTARGET, FACCTOL, ADJUSTSCL)
  - When doing a Monte Carlo simulation, how to adjust the step size.
  - The accepted fraction is checked every ADJUSTEVERY steps. If it is outside the range of FACCTARGET ± FACCTOL, then the step sizes are multiplied or divided by ADJUSTSCL
  - defaults are ADJUSTEVERY=1000, FACCTARGET=0.5, FACCTOL=0.1, ADJUSTSCL=2
- *BDSTEPS*
  - value: 1 required integer (BDSTEPS), 1 optional float (BDPRINTEVERY), 1 optional logical (BDPRINTLOG)
  - Sets the total number of Brownian Dynamics steps (BDSTEPS) and how often to print output.

  - If BDPRINTLOG is true then print at logarithmically spaced step numbers, where BDPRINTEVERY sets the multiplicative factor for the spacing. Otherwise, print every BDPRINTEVERY steps.
  - dafaults: BDSTEPS=1000, BDPRINTEVERY=1, BDPRINTLOG=false

- *BRCRELAX*
  - value: 1 float
  - parameter for extra force to keep segment lengths fixed when running Brownian Dynamics with a bead-rod model
  - Bead-rod BD are not debugged! do not use.

- *CONNECT*
  - value: 4 integers
  - Connection point in a mesh of many chains
  - Many-chain mesh calculations are not tested! do not use.

- *CONNECTMOD*
  - value: 2 floats
  - something about connecting chains in a mesh...
  - Many-chain mesh calculations are not tested! do not use.

- *CONNECTTYPE*
  - value: 2 logicals
  - whether to connect together bead positions and/or orientations
  - Many-chain mesh calculations are not tested! do not use.

- *CONSTMOD*
  - value: 1 float
  - not currently used

- *COUPLED*
  - value: 1 logical

  - not currently used

- *DELTSCL*
  - value: 1 float
  - scaling constant used to set the timestep in Brownian Dynamics simulations
  - Ultimately, if the chain is designated as *SHEARABLE* then the timestep is $\delta t = \text{DELTSCL} * \min(\zeta_{ub}, \zeta_{rb})$, where the friction coefficients ζ_{ub}, ζ_{rb} are set using keyword *FRICT*. If the chain is not shearable, then $\delta t = \text{DELTSCL} * \zeta_{rb}$

- *DIAMONDLATTICE*
  - value: 3 integers; 1 optional float
  - connect up a mesh of chains in a diamond lattice
  - Many-chain mesh calculations are not tested! do not use.

- *DOLOCALMOVES*
  - value: no values
  - When running a Monte Carlo calculation for a single chain, move individual beads rather than the default crank-shaft type moves

- *DYNAMICREDISC*
  - values: 2 floats (REDISCCUTFINE, REDISCCUTCOARSE), 1 integer (REDISCFACT), 1 optional integer (REDISCEDGE), 1 optional float (REDISCEQTIME)
  - When running a first-passage looping time calculation, rediscretize the chain dynamically depending on how close the two ends are
  - Switch to a finer discretization when the end-to-end distance is smaller than REDISCCUTFINE. Switch back to a coarse discretization when the end-to-end distance is larger than REDISCCUTCOARSE.
  - Rediscretization is by an integer factor of REDISCFACT. Each segment turns into REDISCFACT shorter segments

  - REDISCEDGE sets how many of the edge segments are rediscretized (make it bigger than half chain length to rediscretize the whole chain). By default, REDISCEDGE=1, so only the first and last segment are rediscretized.
  - REDISCEQTIME gives the total time for running a mini brownian dynamics simulation to equilibrate newly inserted beads. Default value is $\zeta_r \Delta^4/(3\pi/2)^4$, the timescale for bending equilibration of a single coarse segment.
  -

- *EC*
  - value: 1 float; default: 0
  - The bend-shear coupling energetic parameter for the dssWLC
  - EC $= -\eta\epsilon_b$ using the notation in the Soft Matter paper
- *EDGESEGS*
  - value: 1 integer (N); 6 floats
  - set separate parameters for the first and last N segments
  - parameters, in order, are the ones usually set with the *LS, LP, GAM, EPAR, EPERP, EC* keywords
- *EPAR*
  - value: 1 float; default: 1D3
  - Stretch modulus for the dssWLC energetics
  - EPAR $= \epsilon_{\parallel}$ using the notation in the Soft Matter paper
- *EPERP*
  - value: 1 float; default: 1D3
  - Modified shear modulus for the dssWLC energetics
  - EPERP $= \hat{\epsilon}_{\perp} = \epsilon_{\perp} + \eta^2\epsilon_b$ using the notation in the Soft Matter paper
- *EPAR*

  - value: 1 float; default: 1D3
  - Stretch modulus for the dssWLC energetics
  - EPAR = $\epsilon_{\|}$ using the notation in the Soft Matter paper

- *FINITEXT*
  - value: 1 optional float; default: 1D-3
  - For Monte Carlo simulations, prevent individual segments from stretching beyond the contour length.
  - optional float is a scaling factor F where the stretch energy is a fraction (1-F) of the usual gaussian and a fraction F of a logarithmic term that prevents the overextension of the segment (as in the FENE model)
  - finite extension not tested for a while. Use at own risk.

- *FIXBEAD*
  - value: 1 integer; 1 optional integer; 2 optional logical
  - first integer: which bead to hold fix
  - second integer: on which chain? (for multi-chain mesh runs); default=1
  - logicals: fix position and/or orientation

- *FIXBEAD1*
  - value: no value
  - hold the first bead fixed, when running Brownian Dynamics
  - only set up to work **without** Runge-Kutta

- *FIXBEADMID*
  - value: no value
  - hold the middle bead fixed, when running Brownian Dynamics
  - only set up to work **without** Runge-Kutta

- *FIXBOUNDARY*

  - value: 2 or 4 integers
  - hold the boundary of a multi-chain mesh fixed
  - Many-chain mesh calculations are not tested! do not use.

- *FIXBOUNDARY*
  - no value
  - something to do with multichain mesh simulations...
  - Many-chain mesh calculations are not tested! do not use.

- *FORCE*
  - values: 2 integers; 3 optional floats
  - Add a force on a bead in a particular chain
  - first integer: which bead; second integer: which chain; floats: force vector
  - Used in Monte Carlo calculations only

- *FRICT*
  - values: 2 floats; one optional logical (FRICTPERLEN); default FRICTPERLEN is false
  - if FRICTPERLEN is true, then the position and u-vector friction per length of chain (ζ_r, ζ_b). For a chain with identical segment sizes, the 2 edge beads have half the friction coefficients of the other beads
  - if FRICTPERLEN is false, friction coefficients ζ_{rb}, ζ_{ub} for the bead positions and orientation vectors explicitly
  - Used for Brownian Dynamics calculations only

- *GAM*
  - values: 1 float; default=1
  - Fractional preferred segment extension γ for the dssWLC energetics.

- *GAUSSIANCHAIN*

  - no values
  - For Brownian Dynamics calculations, treat the chain as a plain bead-spring chain with modulus EPAR/2/LS for each spring.
  - For EQUILDISTRIB calculations, treat it as a bead-spring chain with modulus EPAR/2/LS along $\vec{u}$ and modulus EPERP/2/LS perpendicular to $\vec{u}$.

- *INITRANGE*
  - values: 4 floats; defaults: 1D0 1D0 1D0 1D0
  - For Monte Carlo simulations, initial step size ranges
  - angle range for type 1 moves; shift range for type 1 moves; angle range for type 2 moves; shift range for type 2 moves

- *INTERPPARAMS*
  - values: 1 logical; 1 string
  - if logical is true then at the start of the calculation, extract the energetic and dynamic parameters of the chain by interpolating from a data file.
  - string supplies the file containing data for the parameters (see 5)

- *LOGRTERM*
  - no values
  - For an non-shearable chain, this includes the additional logarithmic terms to make it behave as a chain with $\epsilon_\perp \to \infty$ rather than just a plain stretchable chain
  - See appendix in Soft Matter paper for details
  - implemented in Brownian Dynamics and EQUILDISTRIB calculations only

- *LOOPING*
  - optional float LOOPRAD; optional string LOOPFILE
  - track the first-passage looping time in a Brownian Dynamics simulation

  - LOOPRAD is the radius such that chain is looped if ends approach within this distance
  - LOOPFILE is the filename in which to save the looping time of each chain
  - output columns in LOOPFILE: chain, step when first looped, loop time, end-to-end vector when first looped

- *LP*
  - 1 float; default 1
  - bending modulus (ϵ_b) for the dssWLC model

- *LS*
  - 1 float; default 1
  - Segment length (Δ) for the dssWLC model

- *MCPRINTFREQ*
  - 1 integer; 1 optional integer
  - first number is how often to print output to screen during MC simulation (in terms of number of steps)
  - second number is how often to output to the file set by *OUTFILE*
  - When doing EQUILDISTRIB action, second number is how often to print to screen (in terms of number of chains)
  - By default, second number is same as the first

- *MCSTEPS*
  - 1 integer; 2 optional integers; defaults: 1000,100,100
  - First number is total number of MC steps to run if doing the MONTECARLO action
  - Second number is how often to update statistics, such as average end-to-end distance
  - Third number is the number of initialization steps in the Monte Carlo before you start calculating statistics or outputting to file

  - When running the EQUILDISTRIB action, the total number of sampled chains is given by the first parameter here

- *NCHAIN*
  - one integer
  - For Brownian Dynamics or Monte Carlo simulations, the number of chains to run in parallel
  - Warning: multi-chain Monte Carlo has not been tested in a long while. Use at own risk

- *NOBROWN*
  - no values
  - Do not include the random Brownian forces in the Brownian Dynamics simulations

- *NPT*
  - 1 integer; 1 optional integer (MAXNPT)
  - Number of beads in each chain
  - If second integer provided, can also set the maximum allowed number of beads for the case of dynamic rediscretization. By default MAXNPT is the same as NPT
  - MAXNPT sets the sizes of all the different arrays in the chain object, so things will break massively if more beads than this ever show up

- *OBSTACLE*
  - 3 floats
  - radius, steric modulus, friction coefficient
  - sets up an obstacle to interact with the chain
  - Never fully implemented. Do not use!

- *OUTFILE*
  - string; default *.out

  - General output file; * is replaced with suffix for the job
  - for EQUILDISTRIB action: end-to-end vector and first orientation vector for each chain
  - for BROWNDYN action: step, chain, energy, end-to-end vector, center of mass, first $\vec{u}$ vector; output frequency set by *BDPRINTFREQ*
  - for MONTECARLO action: step, type 1 move acceptance frequency, type 2 move acceptance frequency, average R^2, end-to-end vector, correlation between first and last $\vec{u}$, radius of gyration, correlation between end-to-end vector and first $\vec{u}$, some other stuff; output frequency set by *MCPRINTFREQ*

- *OUTPUTBEADWEIGHT*
  - 2 optional integers; defaults 500 50
  - For MC simulations, when dumping snapshots, output an extra two columns which contain, for each mobile bead, the partition function integrated over $\vec{u}$ and integrated over $\vec{r}$ respectively
  - integers are number of integration points in each dimension; 2-dimensional integration over $\vec{u}$, 3-dimensional over $\vec{r}$

- *PARAMFROMSNAPSHOT*
  - 1 optional logical; if not supplied then value set to true; if keyword is missing then default is false
  - if true, then energetic parameters LS, LP, GAM, EPAR, EPERP, EC will be extracted from the input snapshot file (set with *RESTART*) rather than using the ones in the parameter file

- *REDISCRETIZE*
  - 2 floats
  - dynamic rediscretization based on segment lengths
  - Not fully implemented. Do not use

- *REDISCRETIZE*
  - 2 floats

  - dynamic rediscretization based on segment lengths
  - Not fully implemented. Do not use

- *RESTART*
  - 1 optional string, 1 optional integer; defaults: start.out, 0
  - restart calculation from a previously output chain snapshot
  - first parameter is the snapshot file
  - second parameters allows for skipping first few configurations in thefile
  - Will attempt to reach NCHAIN chains from the snapshot file. If there are not enough will start cycling through the configs in the file
  - For Monte Carlo simulations, the snapshot file contains a number on the first line that indicates what step to start from

- *RNGSEED*
  - 1 integer; default: false
  - seed for random number generator
  - value of 0 will seed with system time in milliseconds
  - value of -1 will use the last 5 characters in the suffix
  - value of -2 will use the last 4 charactes in the suffix and the millisecond time
  - otherwise: the seed is used directly (should be positive)

- *RUNGEKUTTA*
  - 1 integer; default: 4
  - what order of runge-kutta method to use with Brownian Dynamics simulations
  - So far only 1st order (direct Euler's method) and 4th order Runge-Kutta are implemented; only 4-th order has been extensively tested

- *SETSHEAR*

  - 1 float
  - set a shear displasement for a chain mesh
  -
  - Many-chain mesh calculations are not tested! do not use.

- *SHEARABLE*
  - 1 logical; default true
  - set whether chain is shearable

- *SNAPSHOTS*
  - 1 optional integer, 1 optional string, 1 optional logical; defaults: 1, *.snap.out, false
  - Dump snapshots over the course of the calculation (for BROWNDYN, MONTECARLO, or EQUILDISTRIB actions)
  - integer: how often to dump snapshots; string: snapshot file (* is replaced with suffix); logical: append rather than rewriting the snapshot file

- *STARTEQUIL*
  - 1 optional integer (EQUILSAMPLETYPE), 1 optional float (STARTEQUILLP)
  - For Brownian Dynamics simulations only, start from a set of equilibrium chain configurations
  - EQUILSAMPLETYPE must be 1,2, or 3 and indicates how to generate the equilibrium sampling
  - EQUILSAMPLETYPE=1, use rejection sampling with a Lorentz envelope. Becomes very inefficient for short segments, high shear modulus
  - EQUILSAMPLETYPE=2, use rejection sampling with multivariate normal envelope; this is the preferred method for chains with short stiff segments; less efficient for highly flexible segments
  - EQUILSAMPLETYPE=3, use Monte Carlo sampling; really inefficient

  - If optional float is supplied, then sample from a bead-rod distribution with the given bend modulus (even if the actual chain for the BD simulations is a proper shearable chain)

- *SQUARELATTICE*
  - no values
  - set up a chain mesh with a square lattice
  - Many-chain mesh calculations are not tested! do not use.

- *STARTCOLLAPSE*
  - no values
  - start with a collapsed linear chain mesh
  - Many-chain mesh calculations are not tested! do not use.

- *STERICS*
  - 1 float, 1 optional integer, 1 optional float
  - for steric inter-bead interactions
  - First number is the steric radius, second is how many neighboring beads to skip in steric calculations, third is steric modulus
  - Steric calculations are not tested! do not use.

- *STRESSFILE*
  - 1 string; default: *.stress.out
  - For Brownian Dynamics simulations, file in which to output shear stress correlation (C_{shear}) over time
  - How often to output is set by BDPRINTFREQ

- *STRETCHABLE*
  - 1 logical; default true
  - Chain is stretchable
  - Chains that are shearable but not stretchable are not implemented
  - Stretchable but not shearable is implemented

- *TRACKDIST*
  - four integers
  - For a multi-chain Monte Carlo calculation, track the average distance between two specific beads
  - numbers are: bead 1, chain 1, bead 2, chain 2
  - Many-chain mesh calculations are not tested! do not use.
- *USEBDENERGY*
  - no values
  - When running a Monte Carlo simulation, use the energy calculation as it was defined for Brownian Dynamics as opposed to just calculating the local change in energy at each step.
  - Makes for very inefficient simulations!
- *USEPSEUDOFORCE*
  - no values
  - Use pseudo-potential force when running bead-rod Brownian Dynamics simulations (SHEARABLE and STRETCHABLE set to false)
- *VERBOSE*
  - 1 logical; default: false
  - print extra output
  - not really implemented