Simulation framework for the Randomly Cross-Linked (RCL) polymer - a user guide

O. Shukron and D. Holcman

Group of applied mathematics and computational biology, Ecole normale Superieure, department of Biology, 46 rue d'Ulm, Paris, France.

Contents

I. Description

This guide contains a description of the codes and workflow for the simulation framework of the Randomly Cross-Linked (RCL) polymer. The simulation framework described here was used to produce results reported in: "Two loci single particle trajectories: a first passage time statistics of local chromatin exploration" by O. Shukron and D. Holcman.

The RCL polymer is a generalized Gaussian chain, composed of *N* monomers connected linearly by harmonic springs and additional *Nc* spring connectors added between randomly chosen monomer pairs, in each realization of the polymer. The framework allows to collect statistics for the first encounter time between two monomers of the chain, before and after the induction of a DSB break between them. The DSB is induced by removing the spring connectors between the chosen monomers.

II. The simulation framework

The polymer is first brought to relaxation, after which a DSB is induced by removing the spring connector between the chosen monomers. All added connectors to these monomers are removed after the induction of DSB. The simulation continues to allow the two detached monomers to drift apart. When the monomers are outside the encounter distance, the first encounter time simulation is executed. Once the two monomers are within the encounter distance, the round stops and a new simulation begins, where *Nc* monomer pairs are chosen anew to be connected. This randomization of the monomer pairs to connect accounts for the heterogeneity in chromatin configuration in a population of cells.

The simulation framework is designed as a test system, in which the number of random connectors added to the RCL polymer determines the number of experiments. This number can be altered (see parameter section VI), and the statistics from each simulation gathered and analyzed.

III. Computational tools

All codes were written in Julia ver. 0.5.1 by Ofir Shukron, group of applied mathematics and computational biology, Ecole normale Superieure, department of Biologie, 46 rue d'Ulm, Paris, France, oshukron@biologie.ens.fr.

For technical details about the Julia language see the article by Bezanson et al. 2016 SIAM "Julia: A Fresh Approach to Numerical Computing"
which can be found in https://julialang.org/publications/julia-fresh-approach-BEKS.pdf
see also the Julia lang. website: https://julialang.org/
The work with Julia codes is performed using a simple text editor and the terminal on both Windows, Linux and Mac.
A Julia IDE can be used to edit and run codes. IDEs such as Juno (http://junolab.org/) on an Atom editor (https://atom.io/) are available for download for free online.

IV. Before you begin

1) Place all codes supplied with this package in the same folder (the working dir).
2) Invoke the Julia REPL (terminal);
3) make sure the path in the Julia terminal is the working directory, by typing *cd("workingDir")*, where *workingDir* is the path to the code folder;
4) make sure the following Julia packages are installed:
   PyPlot
   StatsBase
   Optim

DataFrames
Distances

To install a package such as PyPlot, type in the Julia terminal:
Pkg.add("PyPlot);

5) Make sure all packages are updated by typing in the Julia terminal: *Pkg.update().*

V. Running the codes

The main script is the file *scrRunRCLBreakSimulation.jl*. To run simulations, type in the Julia terminal:

include("scrRunRCLBreakSimulation.jl")

VI. Parameters

Here we summarize the parameters of the simulation framework. The user may change parameters to fit his/her experimental settings. Parameter values should be set in the script file *scrRunRCLBreakSimulation.jl* before invoking the call to the main script *RCLBreakSimulation.jl* rather than changing the values in the parameter file directly, although each method will work. The parameter file: *RCLBreakParams.jl* contains the following parameters, given here with their data types and a short description:

1) ***numSimulations*** (integer): number of simulations to perform for each number of random connectors;
2) ***numRandomConnectors*** (integer vector): number of random connectors (vector of integers), e.g. [10,20,30];
3) ***numSimulationbatches*** (integer): determined automatically according to the number of random connectors tested;
4) ***numRelaxationSteps***(integer): number of steps to polymer relaxation;
5) ***numSteps*** (integer):number of Steps beyond relaxation (not in transient simulation);
6) ***calculateRelaxationTime*** (Boolean): true or false. automatically compute the number of relaxation steps for each simulation;
7) ***numMonomers*** (integer): number of monomers in the RCL polymer chain;
8) ***dimension***(integer): spatial dimension;
9) ***connectorsSTD*** (float64): the standard deviation of springs' length;
10) ***diffusionConst***(float64): diffusion constant
11) ***dtRelaxation*** (float64) time step for polymer relaxation phase
12) ***dt*** (float64): time step for simulation, not including relaxation times;
13) ***springconst***(float64): spring constant (determined automatically);
14) ***encounterDist***(float64) encounter distance between the monomers of chosen for DSB;
15) ***simulationBreak*** (Boolean) true/false whether to induce DSB
16) ***fetSimulation*** (Boolean): perform first encounter time simulation true/false;

17) ***fetForMonomers*** (integer vector) vector of two monomer indices for which MFET is computed;
18) ***induceUniformDSB*** (Boolean) true/false, induce uniformly distributed DSB between adjacent monomer pairs;
19) ***numberOfDSB*** (integer) number of DSB to induce between adjacent monomer pairs. Only works when *induceUniformDSB=true;*
20) ***monomersToDetach*** (integer vector): [monomer1 monomer2; monomer3 monomer4] a matrix of monomer indices pairs of between which a DSD is induces. Only works when *induceUniformDSB=false;*
21) ***fetSimulationStartFromMaxDist***(Boolean) start FET simulation from maximal distance between broken ends in a set of relaxation time stage;
22) ***fetSimulationStartFromMeanPos***(Boolean): deprecated;
23) ***maxFETSteps*** (integer): maximal steps for FET simulation;
24) ***breakAllConnectorsFromDetachedMonomers*** (Boolean): true/false, whether to break all connectors to and from the broken ends (not including the linear backbone);
25) ***mechanicalSpringPointForce*** (Boolean): deprecated. Harmonic spring pushing force generated around each monomer;
26) ***fitHistToData*** (Boolean): true/false, fit histograms of FET (single exponential model);
27) ***numHistBins*** (integer): number of bins in the FET histogram
28) ***numFitTrials***(integer): number of times to try the fit to FET histograms (the best fit, in terms of R-squared, is chosen);
29) ***histPlotFlag*** (Boolean): true/false, display FET histograms after fitting;
30) ***saveResults*** (Boolean): save result structure in *resultBaseFolder;*
31) ***resultBaseFolder*** (string): indicate the path to result folder "result path";
32) ***resultFolderName*** (string): the name of the current result folder saved in "resultBaseFolder/resultFolder";
33) **showSimulation**(Boolean): true/false, show live simulations (after relaxation steps);
34) ***showChainSnapshot*** (Boolean): show two snapshots of the chain, before DSB, and after the chosen monomers have encountered for the first time in each simulation;
35) ***plotEncounterTimeVsConnectivity*** (Boolean): true/false, show the mean first encounter time vs. the number of random connectors added;
36) ***plotMaxMonomerDistanceVsConnectivity*** (Boolean): true/false, plot the maximal monomer distance vs. number of random connectors.

VII. Code structure

The main script is the file *scrRunRCLBreakSimulation.jl, which calls RunRCLBreakSimulation.jl* with input parameters and produces a result structure as output (see section VIII).

RCLBreakSimulation.jl calls *RCLModule.jl,* which includes all of the RCL polymer's sub functions. *RCLModule.jl* calls *ForceManager.jl.*

VIII. The Result folder

At the end of all simulation rounds, the system saves result structure in the results folder (only when the parameter *saveResults=true*). The result are saves in a .csv file format, which can be viewed by either Excel or other simple text editors (recommended, Visual Studio Code with Excel Viewer add-on).

The script *scrRunRCLBreakSimulation.jl* creates a single folder for the simulation of the unbroken loci simulation, called unbroken, and matching folder for the DSB simulations (called DSB).
Inside these base folders the *resultFolderName* as set in parameters will be generated for each experiment.

IX. File list in Result folder

This section contains a summary of the files generated at the end of simulations in the Result folder. The parameters names appearing in this section match the ones in section VI (parameter table).

1. ***averageMaxDistance.csv***: For each number of random connectors indicating the average maximal distance between the two monomers set for separation over *numSteps* steps;
2. ***FirstEncounterTime.csv***: For each number of random connectors indicating the first encounter time between *monomersToDetach* for each simulation;
3. ***FitParams.csv***: Two fitted parameters lambda and a for the FET histogram using the model *a*Exp(-b*t)*, with *t* the encounter time, and a and b constants;
4. ***Histograms.csv****:* The FET histogram for each number of random connectors, reported in bins units (set in params. *numHistBins*);
5. ***maxMonomerDistance.csv***: For each number of random connectors, for each simulation, the maximal distance between the monomers set for separation (the parameter *monomersToDetach*);
6. ***meanKC.csv***: For each number of random connectors, for each monomer, the estimated apparent spring constant;
7. ***meanLC.csv***: For each number of random connectors and for each monomers, the variance of the monomer position in a set of *numSteps*;
8. ***MFET.csv***: For each number of random connectors, the mean first encounter time between *monomersToDetach*;
9. ***MSRG.csv***: The mean square radius of gyration for the whole polymer;
10. ***numConnectorsRemoved.csv***: Number of random connectors removed after the induction of DSB between *monomersToDetach*.

11. ***parameters.txt***: A list of parameters and their values used in the simulation;
12. ***radiusOfGyration.csv***: The radius of gyration for the RCL polymer in each simulation.

Subfolders

At the end of simulations the result folder will contain an additional sub-folder called *lastChainconfiguration.* This subfolder contains the coordinates of each monomers at the last step of the last simulation of each experiment (each number of added random connectors). The coordinates are given in a csv format with x y z columns.
An additional file called *connectedMonomers_.csv,* will be generated and contains the indices of connected monomers in the last simulation.
An additional subfolder, *PDB,* will contain a .pdb file for each one of the polymer configuration. PDB files can be opened with softweare such as UCSF chimera (https://www.cgl.ucsf.edu/chimera/), Jmol (http://jmol.sourceforge.net/), MATLAB Molecular Viewer app (integrated Jmol), Blender with import pdb extension and many others.