1 Monte Carlo simulation with the cgDNA model

Before starting the following exercise you should have been done Ex. 5 Session 5. First set $D = \{AT, GC, AA, AC, AG, GG\}$ a set of 6 independent dimers, using the dimer steps in D we can define all 6 distinct poly–dinucleotide sequences (denoted by $\text{poly}(\alpha\beta)_N$) that are the sequences obtained by N repetitions of the dimer $\alpha\beta \in D$.

1.1 cgDNA reconstruction of the 6 distinct poly–dinucleotide

Using the cgDNA MATLAB package reconstruct the ground state of the 6 $\text{poly}(\alpha\beta)_{150}$, $\alpha\beta \in D$ using cgDNAparamset 1 and 2. Then:

1. Visualize the entries of the ground states coordinates and compare them between parameter set 1 and 2. Note that the ground state intra vary very considerably between the six sequences.

2. For the cgDNA parameter set 2, visualize the 3D reconstruction of all the ground states and compare (in the eye–ball metric) the radius and the pitch of the resulting helical structures. (Here you can actually just plot the xyz coordinates of all the base-pair positions)

1.2 Monte Carlo simulation of the cgDNA model with MATLAB

Monte Carlo simulation for a multivariate normal distribution can be easily done in MATLAB when the mean and the covariance are given. The only command needed is `mvnrnd` (use the help command to find out how to use it). For brevity, use only cgDNAparamset 2.

1. Sample 250 configurations for each of the six $\text{poly}(\alpha\beta)_{150}$, and plot the xyz position of all the base–pairs of each sampled configuration plus the one for the ground state on the same figure.

2. For each poly-dinucleotide, plot the position of the last base–pair of each sampled configuration. What you can say about the shape of the point clouds?

We want now to sample a larger number ($\sim 10^6$) of configuration for each $\text{poly}(\alpha\beta)_{150}$, but the function `mvnrnd` is time expensive when the number of samples is high (try yourself to redo part 1) above for 1000, 2000 or 10000 configurations with MATLAB). For that reason we need a more efficient code to do Monte Carlo simulations.

1.3 The cgDNAmc code

The cgDNAmc code (written by J. Glowacki, Computation and Visualization in Multiscale Modelling of DNA Mechanics, PhD dissertation # 7062, EPFL) will allow you to sample 10^6 configurations of a $\text{poly}(\alpha\beta)_{150}$ in a few minutes on a regular laptop. This includes reconstructions of the (r_i, R_i) base–pair configurations from the sampled cgDNA coordinates $\mathbf{x}$.

How to run cgDNAmc

To run a simulation with cgDNAmc you need to run two scripts:

Step 1) build_seq_data : Compute the lower triangular factor of Cholesky factorisation for the cgDNA stiffness matrix. To do once for each sequence.

Input 1) : -p "path to a cgDNAparamsetX.txt"

Input 2) : -s "path to a text file with a DNA sequence"

Input 3) : -o "path to a file where the data should be stored"

Example : ./build_seq_data -p cgDNAparamset2.txt -s test_seq -o test_data

Step 2) run_cgDNAmc :

Input 1) : -e "expectation you want to compute" (see part (1.4) for details)

Input 2) : -l "name you want to use for the generated input files"

Input 3) : -i "path to the data generated with build_seq_data"

Input 4) : -t "argument describing the way the stiffness in provided"(use always **c** for Cholesky factorisation)

Input 5) : -a "requested number of accepted configurations" (exclusive with -g, do not use it in this Exercise session).

Input 6) : -g "requested number of generated configuration" (exclusive with -a)

Input 7) : -d "number of bases to drop from both ends"

Input 8) : -j "this flag indicate whether the Jacobian should be used" (for this Exercise session use always **n**)

Example : ./run_cgDNAmc -e t0 -l test_run -i test_data -t c -g 1000000 -d 0 -j n

Run a few tests on a poly-dinucleotide in order to assure that the code is working properly.

Modified run_cgDNAmc

Modify the script run_cgDNAmc.cpp in order to save a file with the last base pair position for each sampled configuration. Then, load this file in MATLAB and visualize the obtained cloud of points. What can you say?

1.4 Compute the persistence length using the cgDNAmc code

In this section we will compute in two different ways the persistence length of all the six poly-dinucleotide.

- Use the command **-e t0** to compute the tangent-tangent correlation (ttc) with run_cgDNAmc.
- Use the command **-e r** to compute the Flory persistence vector with run_cgDNAmc.

Remark: The tangent-tangent correlation is the (3,3) entry of the quantity $\langle R_1^T R_i \rangle$, while the Flory persistence vector is the quantity $\langle R_1^T (r_i - r_1) \rangle$.

Convergence study of the Monte Carlo simulation

Use cgDNAparamset 2 for the following simulations. The estimation of a mean value of an observable strongly depends on the number of samples used in the Monte Carlo simulation. In this part we will empirically study this convergence for the tangent–tangent correlation (ttc) for poly$(AT)_{150}$.

1. Solo version: Compute ttc for the following number of samples $N_s = 250, 10^3, 10^4, 10^5, 10^6$, and for each N_s repeat the experiment 10 times. For each N_s save all the computed ttc , average them and compute the variance at some fixed base–pairs. Visualize them by plotting the log of ttc against the number of base-pairs.

2. Cooperative version: Work in a group to do the same process explained above where each person do a certain number of repetition, try to have between 30 and 50 repetition for each N_s.

For which number of samples can the values of the ttc be considered as converged? Denote this number by N_s^c.

The tangent–tangent correlation

Use cgDNAparamset 2 and N_s^c number of samples for the following simulations of all the six poly$(\alpha\beta)_{150}$.

1. In a single figure plot the ln of the ttc values versus base–pair index for each distinct poly-dinucleotide. To what are related the wiggles?

2. The persistence length is computed as the negative reciprocal of the slope of the least squares fit to the plot of ln of the ttc. In MATLAB you can use the "backslash" to solve least square problems. Compute the persistence length for each poly-dinucleotide.

The Flory vector

Use cgDNAparamset 2 and N_s^c number of samples for the following simulations of all the six poly$(\alpha\beta)_{150}$.

1. Plot in a single figure (use `plot3` and plot only each base–pair position) the ground state and the Flory persistence vector, moreover on the Flory persistence vector plot a cross each 25 base–pairs. What can you say about the position of the crosses? Is the Flory persistence vector converged? (the convergence here is on the number of base–pairs and not on the number of samples)

2. Do more tests with poly$(\alpha\beta)_N$, $N = 200, 250, 300, \ldots$. For which N is the Flory persistence vector converged ?

3. Using the N founded in the previous part, redo the Monte Carlo simulation using the cgDNA parameter set 1. Compare the result with the Flory persistence vector computed with the cgDNA parameter set 2.

4. (Optional) Do the study of the convergence of the Monte Carlo simulation of the Flory persistence vector for poly$(AT)_{500}$. Is N_s^c the same as for ttc?