Brownian Dynamics Simulation for Shear Thickening Solution

Gun Woo Park

November 27, 2015

Contents

Part I

Simulation Details: Brownian Dynamics

Chapter 1

Basic Approaches for Brownian Motion

Neglecting intertial effects for Brownian particles, the evolution equation is working on the momentum space:

$$\frac{\partial \mathbf{r}}{\partial t} = \frac{1}{\zeta}\mathbf{F}^{(s)}, \tag{1.1}$$

where $\mathbf{F}^{(s)}$ is random force contribution from solution. With simple Euler integrator, the evolution equation only account for configurational space:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \delta\mathbf{r}, \tag{1.2}$$

where stochastic step, $\delta\mathbf{r}$ is defined by

$$\delta\mathbf{r} = \frac{1}{\zeta}\int_t^{t+\delta t} \mathbf{F}^{(s)}(t')dt'. \tag{1.3}$$

1.1 Wiener Process for Random Force Contribution

Define Wiener process as

$$\mathbf{W}(t) = \mathscr{W}\left[\mathbf{F}^{(s)}(t')\right] \equiv \int_0^t \mathbf{F}^{(s)}(t')dt', \tag{1.4}$$

the stochastic step, $\delta\mathbf{r}$, is represented by the increment for Wiener process:

$$\zeta\delta\mathbf{r} = \mathbf{W}(t+\delta t) - \mathbf{W}(t) \equiv \Delta\mathbf{W}(\delta t). \tag{1.5}$$

Using simple Euler integrator, the increment for Wiener process has property (Greiner et al. 1988):

$$\langle(\Delta\mathbf{W}(\delta t))^2\rangle = 2\zeta k_B T\delta t \quad \text{with } \Delta\mathbf{W}(\delta t) \sim \mathscr{N}(0, 2\zeta k_B T\delta t), \tag{1.6}$$

where $\mathscr{N}(\mu, \sigma^2)$ denote normal distribution with mean μ and variance σ^2.

1.2 Non-dimensionalization

Since the given micelle size is fixed as R_0, it is appropriate to set characteristic length, l_c, as R_0. Then, we can define characteristic time as

$$t_c = \frac{\zeta R_0^2}{k_B T}. \tag{1.7}$$

Note that

$$[t_c] = \frac{[\zeta][R_0^2]}{[k_B T]} = \frac{M \cdot T^{-1} L^2}{M \cdot L^2 \cdot T^{-2}} = T. \tag{1.8}$$

From dimensionality for increment of Wiener process, we can define dimensionless Gaussian noise $\tilde{\mathbf{R}} \sim \mathscr{N}(0, 1)$ as

$$\Delta \mathbf{W} = \sqrt{2\zeta k_B T \delta t}\tilde{\mathbf{R}}. \tag{1.9}$$

Note that the previous condition is equivalent to the $\tilde{\mathbf{R}}$ is following uniform distribution with the interval $[-\sqrt{3}, \sqrt{3}]$ because the variance of it becomes $(\sqrt{3} + \sqrt{3})^2/12 = 1$.

Finally, we have non-dimensional Euler integrator from Eq. :

$$\tilde{\mathbf{r}}(\tilde{t} + \delta\tilde{t}) = \tilde{\mathbf{r}}(\tilde{t}) + \sum_{i,j} \tilde{\mathbf{F}}^{(r)}(\tilde{\mathbf{r}}_i, \tilde{\mathbf{r}}_j)\delta\tilde{t} + \tilde{\mathbf{F}}^{(s)}\sqrt{\delta\tilde{t}}, \tag{1.10}$$

with

$$\tilde{\mathbf{F}}^{(r)}(\tilde{\mathbf{r}}_i, \tilde{\mathbf{r}}_j) = -C\left(1 - \tilde{\mathbf{r}}_{ij}^2\right)\frac{\tilde{\mathbf{r}}_{ij}}{\tilde{r}_{ij}} \tag{1.11}$$

$$\tilde{\mathbf{F}}^{(s)} = \sqrt{2}\tilde{\mathbf{R}} \tag{1.12}$$

Note that if we want the uniform distribution from -1 to 1 for random noise, we can replace the random potential as

$$\tilde{\mathbf{F}}^{(s)} = \sqrt{2 \times 12}\tilde{\mathbf{R}}'. \tag{1.13}$$

1.3 Simulation Results

1.3.1 Notes

Before going further, it would be emphasis that the simple Euler integrator for Brownian dynamics is purely configurational one. That is there is no involvement of velocity of each beads, and the velocity profile is purely given by flow field. In addition, the Wiener process is proportional to the square-root of increment of time, i.e., the differentiation of position with respect to time becomes diverse:

$$\lim_{\delta t \to 0} \frac{\delta \mathbf{r}}{\delta t} \sim \lim_{\delta t \to 0} \frac{1}{\sqrt{\delta t}} \to \infty. \tag{1.14}$$

Therefore, even if we define velocity as numerical differentiation of position with respect to time, it is diverse due to increment of time approaches zero. That is this is artificial velocity, and does NOT have physical meaning. For these reason, it is prefer to use configurational distribution functions rather than momentum space.

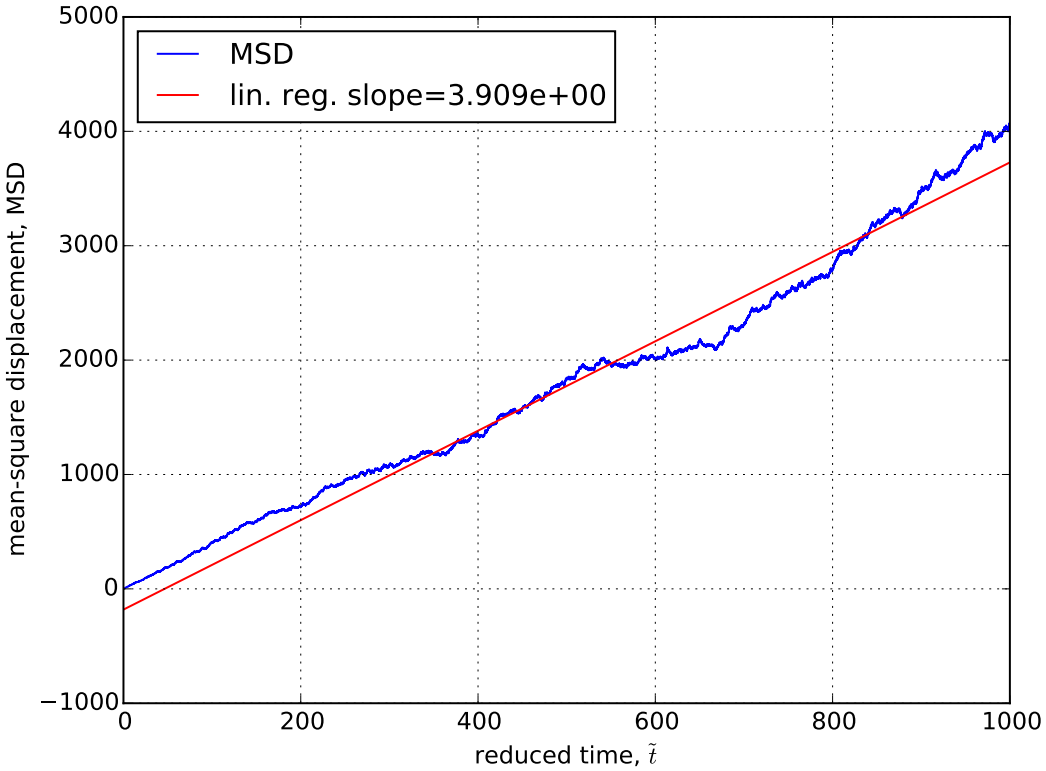


Figure 1.1: MSD for pure Brownian simulation with 1000 reduced time. The blue line represent MSD profile while the red line is linear regression for it. Notice that the slope on here is 3.909 that is similar to the 4 by theoretical interpretation.

1.3.2 Mean-square displacement

Since long-time average for the MSD has linear with respect to time, the slope gave us the diffusion coefficient, $D = k_B T/\zeta$. Recall the reduced time, equation (1.7), we have

$$t_c = \frac{R_0^2}{D}. \tag{1.15}$$

In dimensional form, the MSD has the form of

$$\lim_{t\to\infty} \langle (\mathbf{r}_i(t) - \mathbf{r}_i(0))^2 \rangle_i = 2N_D D t, \tag{1.16}$$

where N_D is spatial dimension. Combined with equation (1.15), the dimensionless form for MSD becomes

$$\lim_{\tilde{t}\to\infty} \langle \left(\tilde{\mathbf{r}}_i(\tilde{t}) - \tilde{\mathbf{r}}_i(0)\right)^2 \rangle_i = 2N_D \tilde{t}. \tag{1.17}$$

Figure 1.1 is plot for MSD with linear regression, but it showed some problematic. Since the slope on the figure 1.1 is 3.909 that is similar to 4, it describe nicely the theoretical interpretation because of $2N_D = 4$. For detail about conversion of trajectory file from periodic boundary condition to real space, and evaluation MSD, see the appendix G.2.

Chapter 2

Brownian Motion with Repulsive Potential

2.1 Basic Fomular

Let assumed there existing repulsive potential between particles on Brownian motion with effective radius. For convenience, let $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$. When distance between two micelles, $|\mathbf{r}_{ij}|$ are lesser than the diameter of single micelle, R_0, the repulsive force is defined by

$$\mathbf{F}^{(r)}(\mathbf{r}_i, \mathbf{r}_j) = \mathbf{F}^{(r)}(\mathbf{r}_{ij}) = -C\frac{k_B T}{R_0}\left(1 - \frac{\mathbf{r}_{ij}^2}{R_0^2}\right)\hat{\mathbf{r}}_{ij} \tag{2.1}$$

where hat denote directional vector, R_0 denote expected size for micelle, and non-dimensional parameter C is given by

$$C = \frac{9}{\pi} n_p^2 N^{0.2} \tag{2.2}$$

with number of polymer in micelle, n_p, and Kuhn number for each polymer, N.

From evolution equation of Brownian motion, equation (1.1), we can add repulsive force to the existing equation as

$$\frac{\partial \mathbf{r}}{\partial t} = \frac{1}{\zeta}\left(\sum \mathbf{F}^{(r)} + \mathbf{F}^{(s)}\right), \tag{2.3}$$

where $\mathbf{F}^{(r)}$ is repulsive force and $\mathbf{F}^{(s)}$ is random force contribution. With simple Euler approach, we can represent the given evolution equation as

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \frac{1}{\zeta}\sum \mathbf{F}^{(r)}(t)\delta t + \delta \mathbf{r}, \tag{2.4}$$

where stochastic step, $\delta \mathbf{r}$ is given by equation (1.3).

2.2 Non-dimensionalization with prefactor C

Appropriate non-dimensionalization should made order of unity for the reduced (basic) variables. That means, the time scale should be involved the stifness for each potential, i.e., repulsive potential on this article. So, re-defined characteristic time becomes

$$t_c = \frac{\zeta R_0^2}{k_B T}\frac{1}{C}, \tag{2.5}$$

while characteristic length is given constant, R_0. Then, we have

$$\tilde{\mathbf{r}}(\tilde{t}+\delta\tilde{t}) = \tilde{\mathbf{r}}(\tilde{t}) + \sum_{i,j} \tilde{\mathbf{F}}^{(r)}(\tilde{\mathbf{r}}_i, \tilde{\mathbf{r}}_j)\delta\tilde{t} + \tilde{\mathbf{F}}^{(s)}\sqrt{\delta\tilde{t}}, \tag{2.6}$$

with

$$\tilde{\mathbf{F}}^{(r)}(\tilde{\mathbf{r}}_i, \tilde{\mathbf{r}}_j) = -\left(1-\tilde{\mathbf{r}}_{ij}^2\right)\frac{\tilde{\mathbf{r}}_{ij}}{\tilde{r}_{ij}} \tag{2.7}$$

$$\tilde{\mathbf{F}}^{(s)} = \sqrt{\frac{2}{3}}\tilde{\mathbf{R}} \tag{2.8}$$

On this regards, the repulsive potential contribution is following

$$U(\mathbf{r}_i, \mathbf{r}_j) = U(\mathbf{r}_{ij}) = \frac{1}{3}\left(1+\tilde{\mathbf{r}}_{ij}\right)^2\left(2+\tilde{\mathbf{r}}_{ij}\right). \tag{2.9}$$

It is noteworthy that the repulsive coeffficient, C, effect to the system. If it is order of unity, both of repulsive and random forces are comparable. If $C \sim 5900$, the contribution for random force is $\sim 1/77$.

From now on, the dimensionless scheme is following the prefactor one.

2.3 Simulation Results

2.3.1 Note

The basic scheme on here is following the pure Brownian motion.

2.3.2 Trajectory Analysis

2.3.3 Distance and Energy Distribution

Since the proposed scheme is purely configurational one, the velocity autocorrelation cannot be obtained during simulation. On this regards, the distribution functions for the spatially related one, such as distance between beads, are of importance to study and judge equilibrium conditions. To obtain cumulative distribution function and probability distribution function is described on . Figure 2.1 represent the distance distribution for the system with 100 beads and 80 beads with C is equal to the 100 as testing purpose. The figure showed the slightly different maximum probability distance. One of the main reason for shifted maximum probability distance is that the minimum potential happens even inside effective range because of density of particles. It is clear when we see figure 2.2, which clearly shows the trend. Since the repulsion is not so stiff, the distance distribution showed relatively broad peaks around 1.0. If we consider the

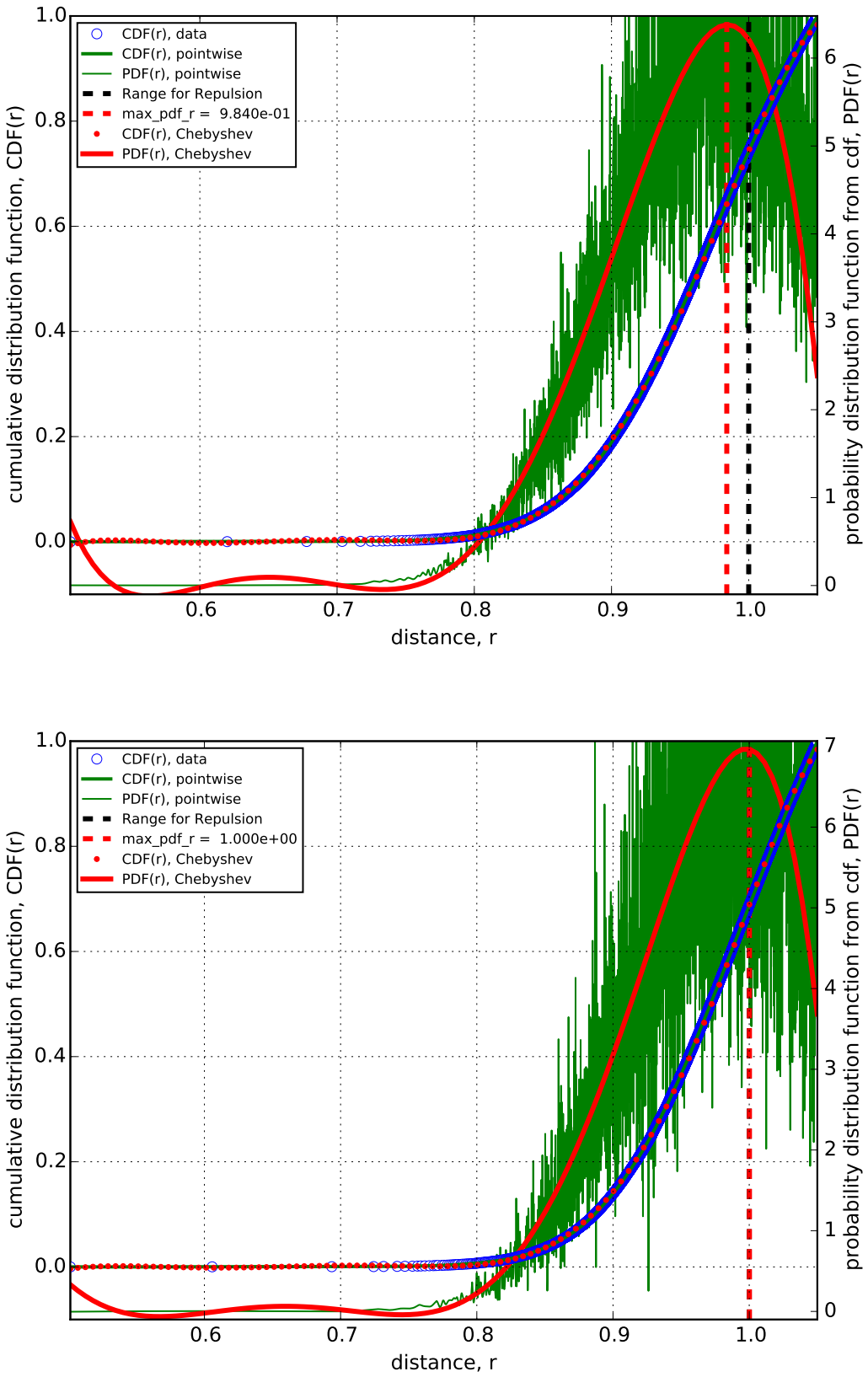


Figure 2.1: Cumulative distribution and probability distribution function for NP = 100 (up) and NP = 80 (down). The red color represent the regression, green color represent the interpolation for piece-wise cubic spline, and blue circle represent cumulative distribution from data.

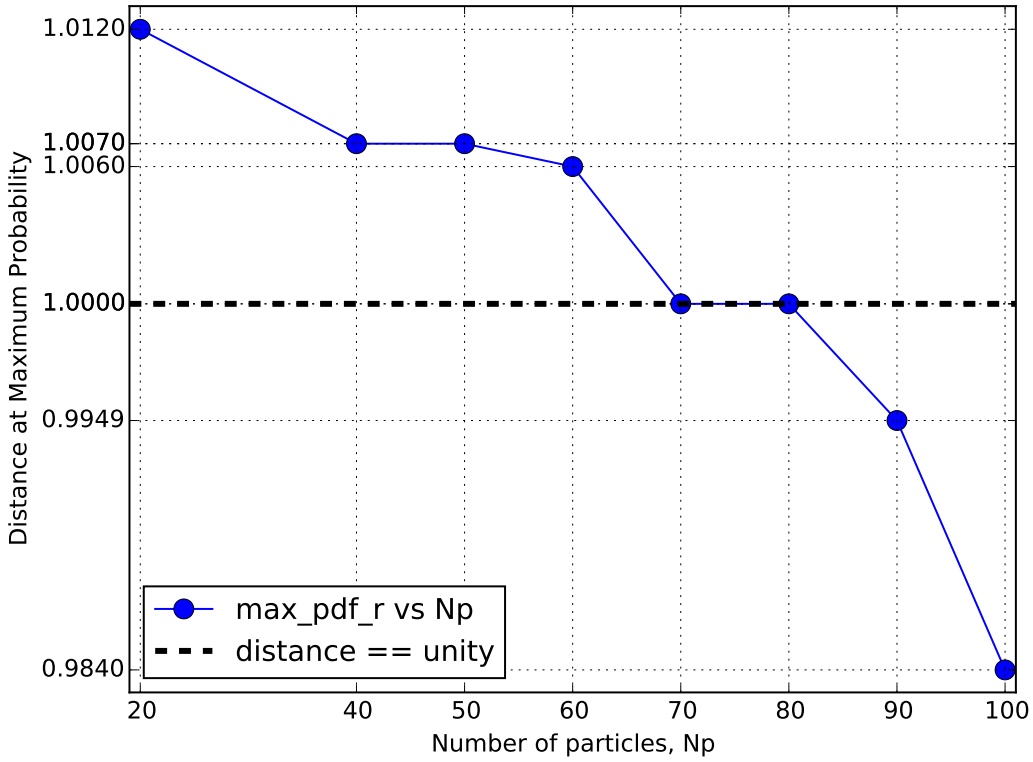


Figure 2.2: Maximum probability distance due to number of particles.

Chapter 3

Brownian Dynamics for Associative System

3.1 Evolution Equations for Associative System

The evolution equation is simply derived by adding association contribution to the equation (2.3). The results is described on here:

$$\frac{\partial \mathbf{r}}{\partial t} = \frac{1}{\zeta}\left(\sum_i \sum_{\substack{j>i,\\ j\in\mathscr{C}_i}} \mathbf{F}^{(c)}(\mathbf{r}_i,\mathbf{r}_j) + \sum_i \mathbf{F}_i^{(r)} + \mathbf{F}^{(s)}\right), \tag{3.1}$$

where $\mathscr{C}_i$ is set of index for the beads associated to the i-th beads and the connected force.

3.1.1 Gaussian Connector

If we assumed that the connected chain is in Gaussian, we have potential

$$U^{(c)}(\mathbf{r}_i,\mathbf{r}_j) = U^{(c)}(\mathbf{r}_{ij}) = \frac{N_D}{2}k_BT\frac{\mathbf{r}_{ij}^2}{R_0^2}, \tag{3.2}$$

it implies

$$\mathbf{F}^{(c)}(\mathbf{r}_i,\mathbf{r}_j) = N_D k_B T\frac{\mathbf{r}_{ij}}{R_0^2}. \tag{3.3}$$

Therefore, the dimensionless potential and force become

$$\tilde{U}^{(c)}(\tilde{\mathbf{r}}_{ij}) = \frac{N_D}{2}\tilde{\mathbf{r}}_{ij}^2, \tag{3.4}$$

$$\tilde{\mathbf{F}}^{(c)}(\tilde{\mathbf{r}}_{ij}) = N_D\tilde{\mathbf{r}}_{ij}. \tag{3.5}$$

3.1.2 Finite Extensible Connector

The original approach for finite extensible chain is come from Kuhn and Grün (1942), and the simpler form is derived from James and Guth (1943). From this results, the potential for finite extensible chain is in complicate form as (Treloar 1975)

$$U = -k_BTn\left\{\log\left[4\pi\sinh\left(\frac{fb}{k_BT}\right)\right] - \log\left(\frac{fb}{k_BT}\right)\right\}, \tag{3.6}$$

where n and b are number and length of Kuhn segment, respectively. From inverse maps for Langevin function, $\mathscr{L}^{-1}$, the final form for force with finite extensibility becomes

$$f = \frac{k_B T}{b} \mathscr{L}^{-1}\left(\frac{r}{nb}\right). \tag{3.7}$$

Since there is no closed form of inverse Langevin function, various approximation is derived. Cohen (1991) derived quite simple but accurate form using Padè approximation:

$$\mathscr{L}^{-1}(\lambda) = \lambda \frac{3-\lambda^2}{1-\lambda^2} + O(\lambda^6). \tag{3.8}$$

Note that the 3 in the formula is not come from spatial dimension, but it is due to the approximation of inverse Langevin function.

Both of potential and force described on above are in complicate to handle. On this regards, alternative finite extensible form suggested by Harold R. Warner (1972) which is frequently called FENE (Finite Extendable Nonlinear Elastic) connector:

$$f = \frac{k_B T}{b} \frac{3\frac{r}{nb}}{1-(r/nb)^2}. \tag{3.9}$$

Here, prefactor 3 is regarded as spatial dimension in comparison with Gaussian spring. From FENE connector, using we can derive potential by integrating over vector, $\mathbf{r}$. Using the maximally extendable length, R_M, and the mean-square equilibrium end-to-end distance, R_0, the potential and force becomes

$$U(\mathbf{r}) = -\frac{N_D}{2} k_B T \left(\frac{R_M}{R_0}\right)^2 \log\left(1 - \frac{\mathbf{r}^2}{R_M^2}\right) \tag{3.10}$$

$$\mathbf{F}(\mathbf{r}) = k_B T \frac{R_M}{R_0^2} \frac{N_D \frac{\mathbf{r}}{R_M}}{1-\frac{\mathbf{r}^2}{R_M^2}}. \tag{3.11}$$

In comparison with Gaussian spring, the force becomes

$$F_{\mathbf{F}}(\mathbf{r}) = F_N \mathbf{F}_G(\mathbf{r}), \tag{3.12}$$

where $\mathbf{F}_G$ is Gaussian spring and the non-Gaussian factor F_N is given by

$$F_N = \frac{1}{1-\frac{\mathbf{r}^2}{R_M^2}}. \tag{3.13}$$

The non-dimensionalized form becomes

$$\tilde{U}(\tilde{\mathbf{r}}) = -\frac{N_D}{2}\left(\frac{R_M}{R_0}\right)^2 \log\left(1 - \frac{\tilde{\mathbf{r}}^2}{(R_M/R_0)^2}\right), \tag{3.14}$$

$$\tilde{\mathbf{F}}(\tilde{\mathbf{r}}) = \left(\frac{1}{1-\frac{\mathbf{r}^2}{(R_M/R_0)^2}}\right) \tilde{\mathbf{F}}_G(\tilde{\mathbf{r}}) \equiv \tilde{F}_N \tilde{\mathbf{F}}_G(\tilde{\mathbf{r}}). \tag{3.15}$$

Notice that the non-Gaussian factor $\tilde{F}_N$ is the same with (3.13):

$$\tilde{F}_N = \frac{1}{1-\frac{\mathbf{r}^2}{(R_M/R_0)^2}} = F_N. \tag{3.16}$$

As reference point of view, Ianniruberto and Marrucci (2015) use $R_M/R_0 = 11$ which this will add for the code as additional parameter.

3.2 Implementation for Association

First of all, the association is pre-determined before solving evolution equation. The expected algorithm is described on below.

1. Visiting randomly selected bead
2. Selecting chains on the bead as transition state based on the Z_k
3. Determine the behaviour of specific chains based on Gaussian distribution
4. Loop until the potential energy is in equilibrium

These steps does not vary the time, instant time, and determine the association information for evolution equation. When initial box is generated, the position is equilibrated without applying association. Then the core of simulation is experience the loop of association step (instant time) up to equilibrium based on number of association and solving evolution equation for next time step. At this moment, each association step does not inherit, which means the information for bridges are re-set when new instant time is given. This is due to testing purpose since the state variables should invariance from history of selection. However, from this option, the computing time for association step occupy more than 99% of all computing expenses, which will be dramatically reduced when switch off the renewal bridge options.

3.2.1 Equilibration for Each Association Step

It is of importance to judge the equilibrium due to MC steps for time optimization. The simple way is using number of associations as state variables, then the equilibrium identify is based on this value. Here, I have used number of association averaged over previous times, then taking numerical differentiation. The block average for numerical differentiation is applied. Let $N_b(m)$ be number of bridge chains at step m. The averaged over previous time in discretized form is given by

$$\bar{N}_b(m_K) = \frac{1}{K}\sum_{k=1}^{K} N_b(m_k). \tag{3.17}$$

Then, the difference of evolving time is given by

$$\bar{N}_b(m_K) - \bar{N}_b(m_{K-1}). \tag{3.18}$$

The last form has benefit to computing expenses. Since it is averaged value, the convergence rate is slower than the real value, which is good point for avoiding the pathology. The results for one instant time is described in figure 3.1. If we apply this scheme for each time, the numerical differenciation is quite flutuation. On this regards, the condition box is given by new variable, N_steps_block. Let P for this value, then the equilibrium scheme becomes

$$\bar{N}_b(m_K) - \bar{N}_b(m_{K-P}) = \frac{1}{K}\sum_{k=1}^{K} N_b(m_k) + \frac{1}{K-P}\sum_{k=1}^{K-P} N_b(m_k). \tag{3.19}$$

Both of equation (3.20) and (3.21) have the form of sum over number of association, that is benefit to measure rather than other method (variance-based one). To be specific, we do not have to sacrifice the computation time because all the MC steps, the sum of associaiton will be computed during MC steps.

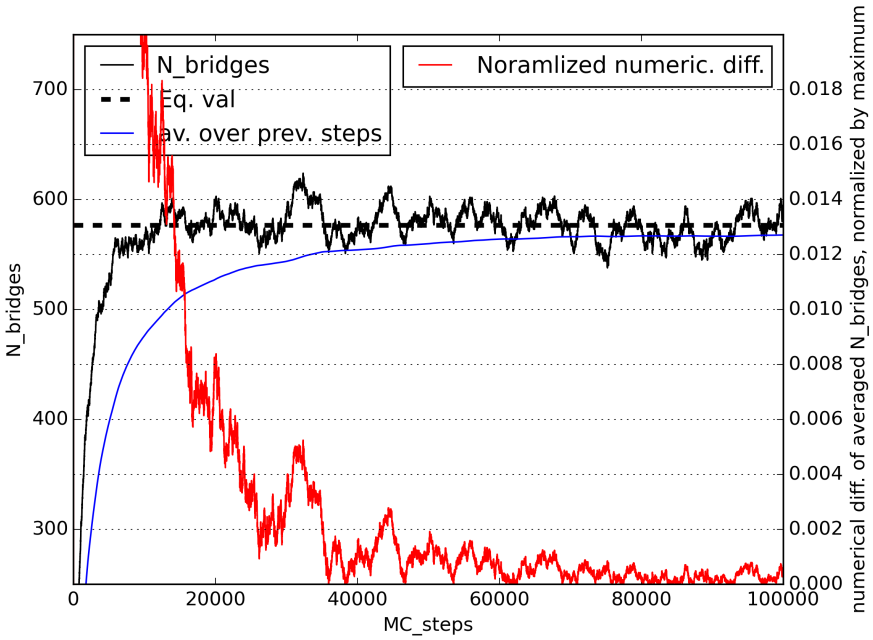


Figure 3.1: Identification of equilibrium for association steps. The black line represent the number of bridges of each MC step, blue line is the number of bridges averaged over previous time, and the red line is normalized difference value without smoothing. To be safe side, slight smoothing process is applied by using several steps.

3.2.2 Probability to Select Chain in Beads

Let there n_p chains for each micelle, i.e., the micelle has $2n_p$ of attached chain end. Each chain end allowing to detach from the original micelle and to attach different micelle. Let U be a potential related to the force exerted on the strand between different micelle, we have partition function for a bead, Z_k, as follow:

$$Z_k = \sum_{i=1}^{2n_p} \exp\left(\beta U(\mathbf{r}_i, \mathbf{r}_{\mathscr{C}_k(i)})\right), \tag{3.20}$$

where $\beta = (k_B T)^{-1}$ and $\mathscr{C}_k(i)$ is index for bead that attached i-th chains on k-th beads. Notice that typical Boltzmann factor for partition function is given by $\exp(-\beta H)$ where Hamiltonian is given by $H = T - V$. Therefore, the sign on here is plus. The given number of case directly suggest that the probability for choosing the chain that will act. For instance, let consider the first beads, Z_1, if the beads does not associate with any other bead, the $\mathscr{C}_1(i)$ becomes 1 for all $i \in [1, n_p]$. From given potential, equation (**??**), all the potential on each chains become zeros that means the each contribution to the partition function is unity. Therefore, Z_1 in this case becomes $2n_p$.

3.2.3 Behaviour of Selected Chain

It is of importance to determine the next step after selection of chain. For given number of beads, N_B, the target to attachment is based on the following distance probability:

$$p_k(\{\mathbf{r}\} \equiv \exp\left(-\beta U(\{\mathbf{r}\} - \mathbf{r}_k)\right), \tag{3.21}$$

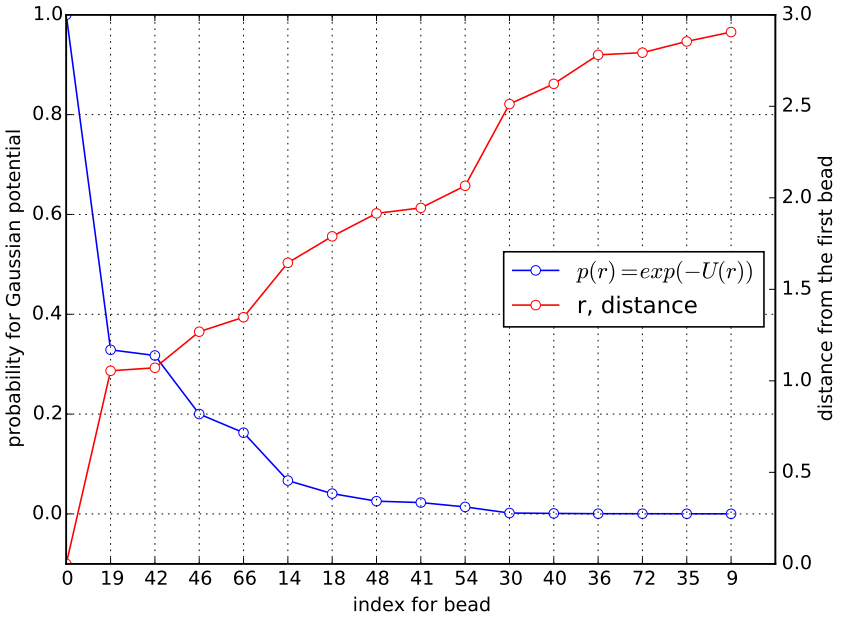


Figure 3.2: Example for the probability of chain extension using Gaussian connector. The data is sampling from 80 beads system and only account for repulsive potential (no association). The variables are dimensionless. The red color represent distance from the first bead (index is 0), and the blue color represent probability using Boltzmann factor.

where U is given potential and $\{\mathbf{r}\} = \{\mathbf{r}_0, \mathbf{r}_1, \cdots, \mathbf{r}_{N_B}\}$. In addition, it should following the ceiling probability rather than floor one. Figure 3.2 is one example to compute probability from distance distribution. When we roll the random generation from 0 to 1, we make decision where to attach and most probably the original bead. When we toggle the random number from 0 to 1, the randomly generated number determine newly selected beads. For instance, if the number is 0.4 in figure 3.2, then the selected chain will attach for the 0-th bead. Notice that the position of beads are fixed during association steps, the maps for adjacent beads is only computed once to reduce computing price.

3.2.4 Allowance for Number of Chain Ends per Bead

If the number of chain ends attached a beads is constant, the movement of association is the only possible way during Monte-Carlo steps. Hence, it would be importance to check the effects of number of allowance for attached chain ends per bead. Figure 3.3 is tested with the 25 chains per bead (i.e., 50 chain ends per bead) and allowing 3, 5, and 7 for the allowance number of fluctuating. It seems to be there is small trend exist due to the allowance number, but in minor. Even if it is quite early to say the total number of association is free from the number of allowance (if it is larger than 3), this will not be major effects for our future works. The results will be re-visited when our algorithms and methods are stable and re-producible.

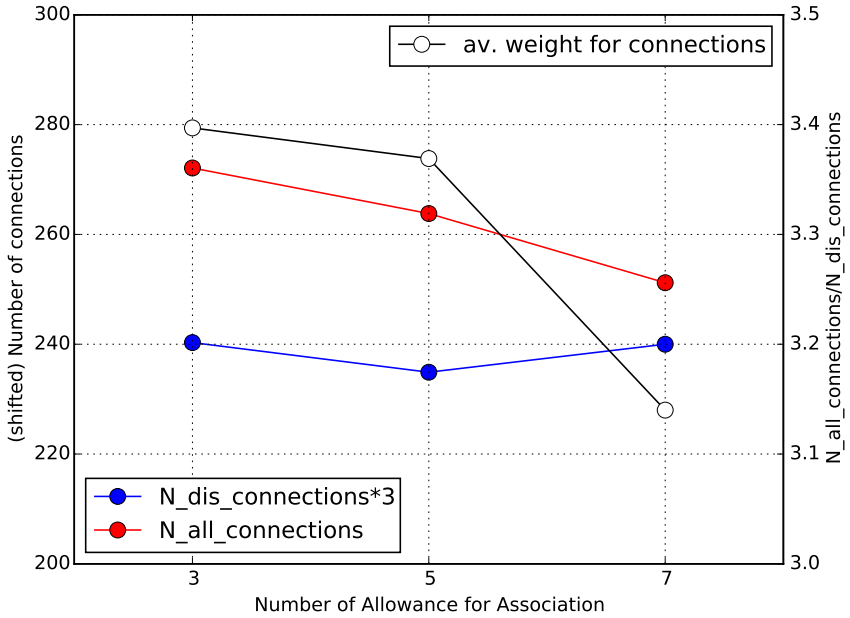


Figure 3.3: Effects of number of allowance for fluctuating attached chain ends per bead. N_dis_connections state number of distinguashable connections, i.e., all different pairs of association, and N_all_connections means number of all bridge chains. The right axis is the ratio between this two counted numbers, which refer intensity per bridge chains.

3.3 Simulation Results

3.3.1 Clustering with Gaussian Connector

The critical concentration for 100 area box (10^2) in dimensionless is based on the figure 2.2 that suggest the 80 beads/100 area can be overlap concentration with applied repulsive potential. To be safe side, the 40 beads is selected for testing purpose. Evolving time make clster in the system, which is shown in figure 3.4. The size effects has been checked that is also reported in figure 3.4. From this aspect, the given parameter sets may show aggregations. Note that, however, the connecting vector shows isotropic; therefore, the clustering in equilibrium simulation does not deviate from isotropic solution concepts.

3.3.2 New Length Scale with FENE Connector

We introduced FENE Connector which is reported in previously. Interestingly, this system does not show macroscopic phase separation but shows local length scale that related with the partially sparse case. Since the radial distribution function, RDF (appendix **??**), is good aspect for length scale related case, figure 3.5 is the results for RDF and also trajectory file itself. It is clearly show there is local length scale related with the sparseness which is around 5 in dimensionless distance. The reported RDF also shows this tendency dimensionless distance from 3 to 6 as under-correlated while distance from 6 to 9 as over-correlated. This heterogeneity significantly shows that implementation of FENE Connector introduce new length scale. It is of importance that the given FENE Connector uses the maximum available distance, R_{max}, as 11, the same with

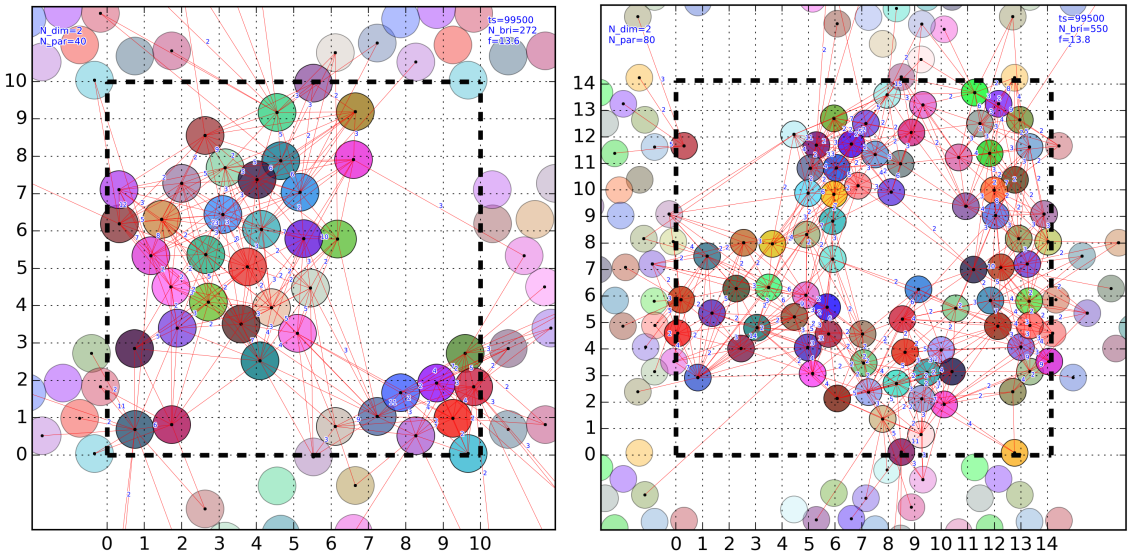


Figure 3.4: Builded cluster during simulation. Both system has the same concentration but different box size.

Ianniruberto and Marrucci (2015) while the Gaussian connector is regarded as infinite length scale for it (commented by Prof. Marrucci).

3.3.3 Isotropic for connecting vector

According to the Ianniruberto and Marrucci (2015), the stress tensor is given by

$$\boldsymbol{\sigma} = 3\nu k_B T \bar{f} \langle \tilde{\mathbf{R}}\tilde{\mathbf{R}} \rangle, \tag{3.22}$$

where ν is excluded volume of a monomer of polymer chain, and $\bar{f}$ is non-Gaussian factor which is unity when we use Gaussian spring for association. On this regards, the associative contribution for the stress tensor proportional to the diadic of connecting vector, $\langle \mathbf{RR} \rangle$. On this regards, the isotropy of connecting vector is of importance for equilibrium simulation, which is reported in figure 3.6. The initial transition parts (up to 200 time steps) is results of clustering. Notice that the given stress tensor is the only associative part. Hence, the check for the istorpic should be checked with the stress contribution from the repulsive parts and Brownian motion as well.

3.3.4 Size Effects

The system size is of important factor for dynamics of the system. By doubling system size, we can checked the given system is in experience for the size effects. The simple result is reported in figure 3.4. **The other aspects will be checked later on**.

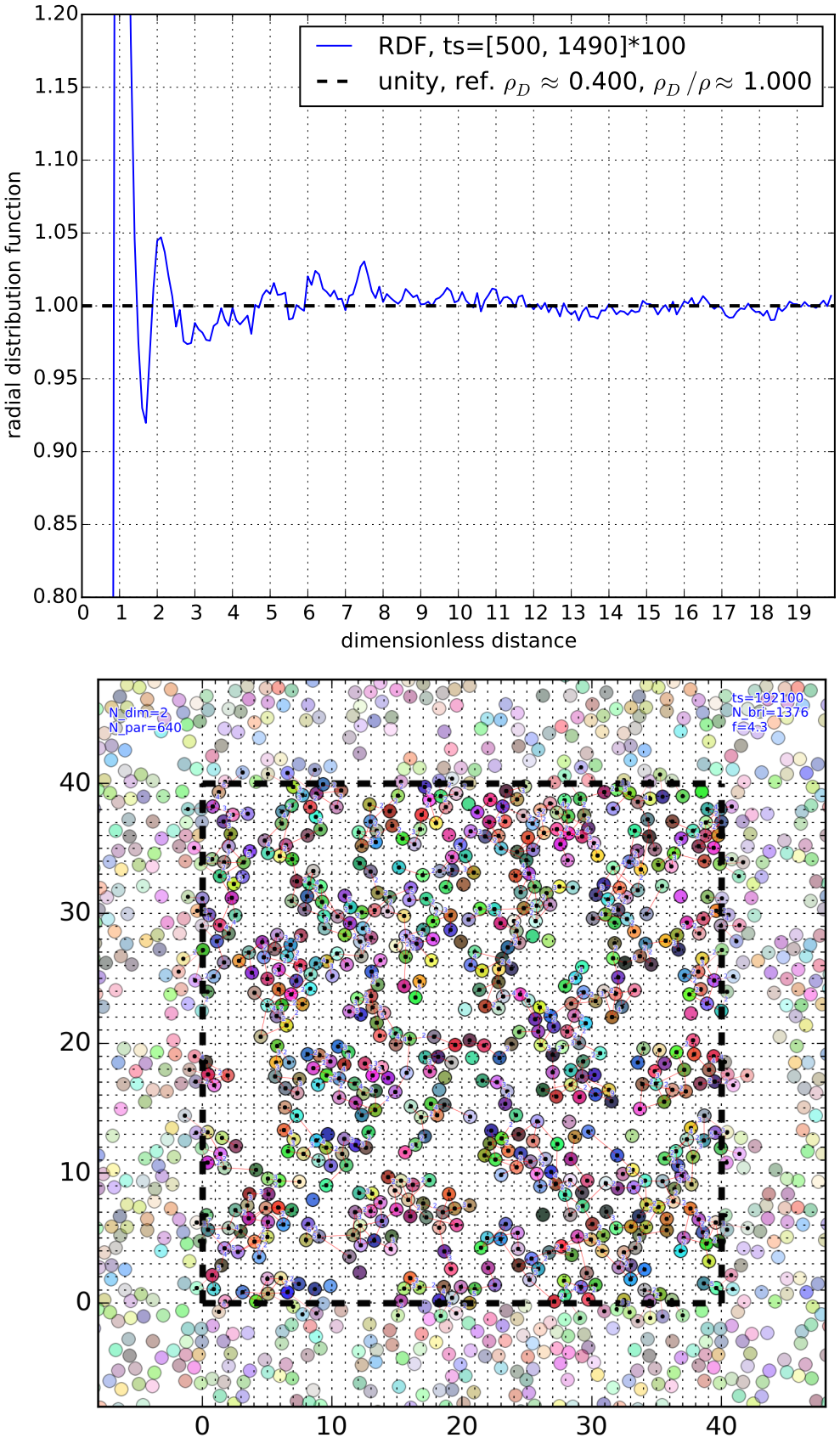


Figure 3.5: Example of trajectory for NP640/1600AREA.

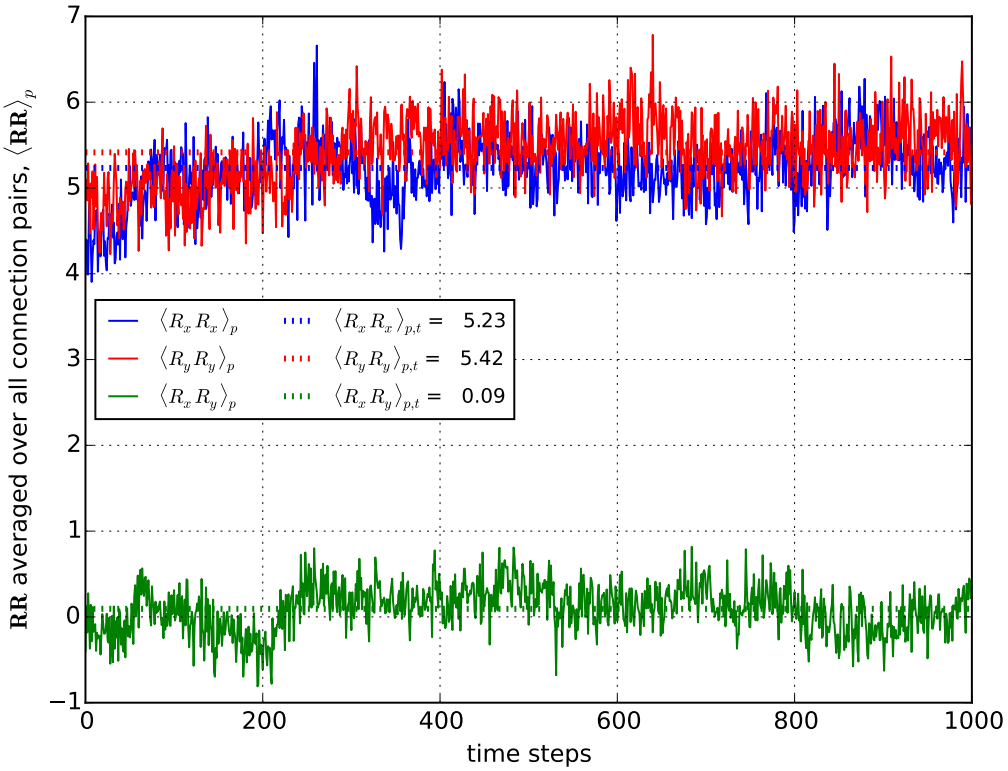


Figure 3.6: Measured $\langle \mathbf{RR} \rangle_p$ for 80 beads per 200 area system. The diagonal components are almost identical while the off-diagonal components are negligible, which is the sign for isotropy.

Appendices

Appendix A

Finite Extensibility

Describe.

Appendix B

Time Correlated Data

B.1 Stationary State Variable

Let $\mathscr{A}$ be a state variable and the system is in equilibrium. That means the $\mathscr{A}$ is fluctuation around a certain average value $A = \langle \mathscr{A} \rangle_t$ when the system is assumed Ergodic. If the sampling time is sufficiently large, the number for time sampling does not affect to the average value, A. In the case of variance, however, it affected by sampling number since we cannot avoid time correlation between two time steps. It is noteworthy that the average value itself said the first moment of $\mathscr{A}$ while the variance is the second moment. On this regards, we can say the time correlation does affect to the moment of $\mathscr{A}$ when the order of moment is higher than 1.

The easiest way to avoid this problem is to use the sufficiently large time step between data, that means the time steps for statistical processing is higher than the correlation time for the data. There are various method to measure real values, but the box average method is the popular and practically useful (Allen and Tildesley 1989).

B.2 Regression

Appendix C

Regression Scheme

C.1 Regression by Chebyshev Polynomial

Let assumed that the variable ξ is in $[-1, 1]$. The generating function for the Chebyshev polynomial is

$$\sum_{n=0}^{\infty} T_n(\xi) t^n = \frac{1 - t\xi}{1 - 2t\xi + t^2}, \tag{C.1}$$

where T_n is n-th order Chebyshev polynomial of the first kind. From it, we can define recursive form:

$$T_0(\xi) = 1 \tag{C.2}$$

$$T_1(\xi) = \xi \tag{C.3}$$

$$T_{n+1}(\xi) = 2xT_n(\xi) - T_{n-1}(\xi). \tag{C.4}$$

For given analytic function $y = f(\xi)$ where $\xi \in [-1, 1]$, we can expand

$$y = \lim_{N \to \infty} \sum_{n=0}^{N} a_n T_n(\xi). \tag{C.5}$$

Note that typical regression is used for appropriate finite number of N rather than infinite.

To compute the coefficient for the equation (C.5), we can define following sum of square error (SSE):

$$\chi = \sum_{\alpha=1}^{M} \left[y_\alpha - \sum_{n=0}^{N} a_n T_n(\xi_\alpha) \right]^2, \tag{C.6}$$

where y_α denote α-th components for given data and ξ_α is its corresponding ξ value. This procedure is basically come from the least square method, and directly gave us the normal equation as

$$\sum_{k=0}^{N} \left[\sum_{\alpha=1}^{M} T_n(\xi_\alpha) T_k(\xi_\alpha) \right] a_k = \sum_{\alpha=1}^{M} y_\alpha T_n(\xi_\alpha) \quad \text{for } n \in [0, N]. \tag{C.7}$$

From this equation, we can define the coefficients, $\mathbf{a} = \{a_1, \cdots, a_M\}$.

It is noteworthy that even if Chebyshev polynomial is well-known and good basis for regression, analytical handling for the polynomial need some time. In addition, it is equivalent to the Taylor expression with order of N, that means we can find the coefficient relation between Chebyshev- and Taylor-style regressions. For further detail for the usefulness and properties of Chebyshev polynomial, see the reference Arfken and Weber (2008).

C.2 Typical Polynomial Expression

The description on here is following Cho (2013) that is practically useful expanation for the regression using Chebyshev polynomial. Consider the basic Taylor expansion for $y = f(x)$ with finite order N:

$$y = \sum_{n=0}^{N} c_n x^n = \sum_{n=0}^{N} b_n \xi^n, \tag{C.8}$$

where ξ is scaled x given by

$$\xi = \frac{2(x - x_c)}{x_{max} - x_{min}} \tag{C.9}$$

with $x_c = \frac{1}{2}(x_{max} + x_{min})$. That is related with the valid region for ξ that defined on the previous section. On this regards, we should find the relation between $\mathbf{a}$ and $\mathbf{b}$. Let $T_k^{(n)}$ be the n-th order Chebyshev polynomial of the first kind with k-th power of ξ, from recursive relation for the Chebyshev polynomial, equation (C.4), we have

$$T_0^{(n+1)} = -T_0^{(n)} \tag{C.10}$$
$$T_{n+1}^{(n+2)} = 2T_n^{(n+1)} \tag{C.11}$$
$$T_{n+2}^{(n+2)} = 2T_{n+1}^{(n+1)} \tag{C.12}$$
$$T_k^{(n+2)} = 2T_{k-1}^{(n+1)} - T_k^{(n)} \quad \text{for } 1 \leq k \leq n, \tag{C.13}$$

with $T_0^{(0)} = 1$, $T_0^{(1)} = 0$, and $T_1^{(0)} = 1$. Then we can express $\mathbf{b}$ by α as

$$b_n = \sum_{k=n}^{N} T_n^{(k)} a_k, \tag{C.14}$$

implies

$$c_n = \sum_{k=n}^{N} \left(\frac{2}{\Delta x}\right)^k \begin{pmatrix} k \\ n \end{pmatrix} (-x_c)^{k-n} b_k, \tag{C.15}$$

where $\Delta x = x_{max} - x_{min}$. Finally, we found the coefficients for typical polynomial form

$$y = \sum_{n=0}^{N} c_n x^n, \tag{C.16}$$

based on Chebyshev polynomial expressions. Further details about this connections, see the reference Cho (2013).

C.3 Overhead for Recursion

It is noteworthy that the given Chebyshev polynomial is generated using recursive relation described on equation (C.4). The recursion is easily implemented into the code by recursive call on function as following python code.

```
1     def Chebyshev_1st(n, x):
2     if n==0:
3     return 1.0
4     elif n==1:
5     return x
6     return 2.0*x*Chebyshev_1st(n-1, x) - Chebyshev_1st(n-2, x)
7
```

```
8     def coe_Chebyshev_1st(k, n): # for coefficient generation
9     if k<0 or k > n or (k==0 and n==1):
10    return 0
11    elif (k==1 and n==1) or (k==0 and n==0):
12    return 1
13    else:
14    return 2*coe_Chebyshev_1st(k-1, n-1) - coe_Chebyshev_1st(k, n-2)
```

However, it has potential overhead for computing purposes since recursive call for function take time for computing. At the moment, the overhead is ignored because it is only used for post-processing.

Appendix D

Cumulative Distribution and Probability Distribution Function

Distance distribution function is of importance to identify the system properties because the given integrator cannot be specify the velocity of each particles.

The practical cumulative distribution is simply calculated by sorting method. Let Y be data column that related with random generation. To make sort Y column then making index for Y related with its population rate to increase. This is exactly the same with the definition for the cumulative distribution that we are using for. For given cumulative distribution, $\mathscr{F}(r)$, we can define probability distribution function, $P(r)$, as

$$P(r) = \frac{d}{dr}\mathscr{F}(r). \tag{D.1}$$

It is of importance that the CDF on this scheme has irregular X-axis, that is the $dx_i = x_i - x_{i-1}$ has different values while $dy_i = y_i - y_{i-1}$ has constant values, which make difficulty to use regression along the CDF. Hence, the two-scheme is involved for both of interpolation and derivation for short interval and compared with the results. Typically, the piece-wise cubic spline is involved all area, even if it is highly fluctuating, in order to use reference for trend. Then, the fitted curve will be compared with this reference.

Appendix E

Software architecture

E.1 GNU's scientific library as Front-End for Mathematical Calculations

The core for mathematical calculation is based on GSL with MKL interface supported by Intel composer. On this Brownian simulation, one of the most important feature is generating random noise. This is achieved using GSL's Random Number Generation packages that support efficient and sufficient white noise for our purpose. In addition, mathematical multiplication and various kinds of processing will be used based on matrix structure of GSL and also its linear algebra packages at the moment.

If we need to achieve better performance, the matrix algebra will be replaced by cBLAS that is C-ported BLAS package, and linear algebra package will be replaced by LAPACK.

E.2 Math Kernel Library (MKL) as interface between Front- and Back-End

The MKL in Intel composer has functionality with its inter-facial function with various packages such as GSL, cBLAS, and LAPACK. Combining with the Intel compiler, the performance is reaching the one of the best in the numerical packages. The benefits of MKL is simple: support various package as their own style.

E.3 Personally developed MATRIX class as Back-End

Not only the opensource packages, the personally developed MATRIX class is used because the core interface for GSL is not so convenience for scientific purpose. In addition, GSL is designed for C rather than C++, that means it is lack of advanced functionality such as operator overloading. Therefore, the core of GSL has high compatibility with various environment but low user-friendly.

The basic idea for developing MATRIX class is to overcome the shortage of connecting between mathematical formalism and code expressions. For instance, for given matrix $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, when we calculate $\mathbf{A} \cdot \mathbf{B} + \mathbf{C}$ to input $\mathbf{C}$, using cBLAS, we need to use following sourcecode.

```
gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1.0, &A.matrix, &B.matrix, 1.0, &C.
    matrix);
```

When we are using operator overloading with specialized object, than we can simply compute

```
C = A*B + C;
```

which is very simple and intuitive to use. Hence, I have developed several way to this expressions using operator overloading. Up to now, there is still performance issue since the equality operator (=) is internally generating new MATRIX object and returning it, that make overheads compared with the function to use call-by-reference style asGSL or cBLAS. To reduce this overhead is high rated for further refinement of sourcecode which will not be solved soon. For this reason, following operator overload is defined (only headerfile is included):

```
1   double& MATRIX::operator()(MKL_LONG i, MKL_LONG j);
2   double& MATRIX::operator()(MKL_LONG i);
3   MATRIX& MATRIX::operator=(const MATRIX &Mat);
4   MATRIX& MATRIX::operator+=(const MATRIX &Mat);
5
6   // MATRIX Addition : C = A+B
7   MATRIX operator+(const MATRIX &A, const MATRIX &B);
8   MATRIX operator-(const MATRIX &A, const MATRIX &B);
9   // Scalar Multiplification : C = a*A
10  MATRIX operator*(const double a, const MATRIX &A);
11  // MATRIX Multiplification : C = A*B
12  MATRIX operator*(const MATRIX &A, const MATRIX &B);
13
14  // Unary operator
15  MATRIX operator-(const MATRIX &A);
```

However, it should be notice that the operator overloading has potentially overhead for computing. At the moment for the Brownian particles, it is not that severe.

E.4 Parsing Test Conditions

For general purpose program, parsing test condition cannot be avoidable. Here, The COND class is newly defined using C++ string class, that contains basic information of given condition file. I have attached one example for the test condition file as follow.

```
1   Method=NAPLE
2   Integrator=Euler
3   N_dimension=2
4   box_dimension=10.0
5   dt=0.001
6   Nt=1000000
7   Np=100
8   N_skip=100
9   N_energy_frequency=1000
10  repulsion_coefficient=100.0
11  effective_distance=1.0
12  output_path=data_test_longer
13  filename_trajectory=100.traj
```

```
    filename_energy=100.ener
    filename_energy_info=100.info
```

The parsing code is given by

```
COND::COND(char* fn)
  {
    GIVEN_FILE.open(fn);
    long cnt = 0;
    string line, cond, val;
    while(getline(GIVEN_FILE, line))
    cnt ++;
    GIVEN_FILE.clear(); // since the previous get-line reach the EOF, this is bad-
        status. So, it need clear to use file object.
    GIVEN_FILE.seekg(0);
    N_arg = cnt;

    arg = (string**) new string* [N_arg];
    for(long i=0; i<N_arg; i++)
        arg[i] = (string*) new string [2];

    cnt = 0;
    while(getline(GIVEN_FILE, line))
      {
        stringstream iss(line);
        getline(iss, cond, '='); arg[cnt][0] = cond;
        getline(iss, val, '\n'); arg[cnt][1] = val;
        cnt ++;
      }
    GIVEN_FILE.close();
    ERR = "ERR";
  }

  string& COND::operator()(string option_type)
{
  for(long i=0; i<N_arg; i++)
      if(arg[i][0] == option_type)
          return arg[i][1];
  cout << "Bad condtion " << option_type << endl;
  return ERR;
}
```

Note that the operator overloading is useful to handling the conditions. For instance, if we need the value for "condition_1", we simply use COND("condition_1") that return the value as string class. If we need C-style string, just use c_str() method. In any case, checking conditional phrase is easily doable by following way.

```
    if (given_condition("Integrator") == "Euler")
    {
      N_basic = 2;
    }
```

E.5 Periodic Boundary Condition

E.5.1 Minimum Image Convention

At the moment, only the rectangular PBC is under the consideration. When we computing some potential or distance, we have to account all image of particles with different cells. Without loss of generality with spatial dimension, the minimum is determined by following codes.

```
1   double UTIL_ARR::get_minimum_image_k_from_x(double x, double k, double dimension
        )
2   {
3     double kd[3] = {k-dimension - x, k - x, k+dimension - x};
4     double re= kd[get_index_minimum_abs(kd, 3)] + x;
5     return re;
6   }
7
8   MKL_LONG GEOMETRY::get_minimum_distance_pos_vector(TRAJECTORY& TRAJ, MKL_LONG
        index_t, MKL_LONG given_index, MKL_LONG target_index, MATRIX& given_vec)
9   {
10    for(MKL_LONG k=0; k<TRAJ.dimension; k++)
11    {
12      given_vec(k) = UTIL_ARR::get_minimum_image_k_from_x(TRAJ(index_t,
            given_index, k), TRAJ(index_t, target_index, k), TRAJ.box_dimension[k])
            ;
13    }
14    return 0;
15  }
```

E.5.2 Applying Periodic Boundary Condition for Trajectory

The application of PBC on the trajectory is easily doable using the absolute value of coordinate. The sourcode is using 0 as origin, and all the box dimension is set as positive value from the origin. Therefore, it is needed to transit to make center as origin, taking modulo operator, then transit to original coordinate. The approach is described on the following codes.

```
1   MKL_LONG GEOMETRY::minimum_image_convention(TRAJECTORY& TRAJ, MKL_LONG target_t)
2   {
3     for (MKL_LONG i=0; i<TRAJ.Np; i++)
4     {
5       for (MKL_LONG k=0; k<TRAJ.dimension; k++)
6       {
7         double diff = TRAJ(target_t, i, k) - 0.5*TRAJ.box_dimension[k];
8         double sign = diff/fabs(diff);
9         if (fabs(diff) > 0.5*TRAJ.box_dimension[k])
10        {
11          TRAJ(target_t, i, k) -= sign*TRAJ.box_dimension[k];
12        }
13      }
14    }
15    return 0;
16  }
```

Appendix F

Parallel Computing

Up to now, the only shared memory parallization is supported via OpenMP package. In order to support GRID computing with multiple nodes, OpenMPI should be involved my sourcecode that is on queuing for further purpose.

It is noteworthy that OpenMP on the C++ has some importance issue to make private variable. Since I am using class instant for our convenience that described in above, the private class instance for OpenMP always call "default constructor". Because default constructor of MATRIX library is not compatible for further computing, it must be used in firstprivate rather than private. The option "firstprivate" is taking copy-constructor rather than default constructor. For detail, see part of Euler Integrator in lib_evolution.cpp file.

```
1     #pragma omp parallel for default(none) shared(TRAJ, index_t_now, index_t_next)
          firstprivate(force_spring, force_repulsion, force_random)
2     for (i=0; i<TRAJ.Np; i++)
3     {
4       FORCE::cal_connector_force(TRAJ, force_spring, index_t_now, i);
5       FORCE::cal_repulsion_force(TRAJ, force_repulsion, index_t_now, i);
6       FORCE::cal_random_force(TRAJ, force_random, index_t_now);
7       for (MKL_LONG k=0; k<TRAJ.dimension; k++)
8       {
9         TRAJ(index_t_next, i, k) = TRAJ(index_t_now, i, k) + TRAJ.dt*(force_spring(k)
              + force_repulsion(k)) + sqrt(TRAJ.dt)*force_random(k);
10      }
11    }
```

At the moment, the copy constructor for the MATRIX class has potential overhead because each constructor allocate memory for the new MATRIX then it will be deleted when the job is finished. The thready safety for MATRIX class is NOT fully tested at the moment, but the functionality of constructor and destructor, and its private properties are checked.

Appendix G

Post-Processing

C++ has high performance and various numerical packages has compatible with it. However, it has lack of package to generating graphs, which is crucial point for the purpose of post-processing. On this regards, Python with numpy, scipy, and matplotlib has very flexible and powerful way to measure various quantities and plot it. However, it should be consider that Python is interactive language and even if Python support compiler option, the performance is very slow compared with C++. In addition, there is performance issue with python. Hence, the multiprocess package is used. Since the purpose to use python is easy and simple, the use of multiprocess is not satisfy the purpose. So, except the plotting, if we need to compute with high performance, then it would be better to use C++ package itself. As described on the previous section, the packages are well-integrated and enough supportablility for other post-porcessing except plotting.

G.1 Plotting and Making Move for Trajectory File

In this case, get appropriate marker size for beads is of importance in order to recognize the effective range for the system. Typically, transform package in matplotlib is used for this aspect.

```
marker_unit = (ax.transData.transform((1, 0)) - ax.transData.transform((0, 0)))[0]
```

Since the given trajectory file is quite big, loading all the fild into the memory has potential problems. It is of importance to get line-by-line rather than loadtxt functionality of numpy package because loadtxt load all file into memory at once. Note that parsing each file line had overhead compared with the loadtxt functionality. Hence, for smaller files, it prefer to use the loadtxt. From line parsing, we can allowed to use parallel computing. For the author's point of view, using GNU's terminal tool parallel is good for various aspect. In this case, however, the data file can be big enough to overflow memory and it is the reason to choose parsing data, directly, rather than using loadtxt. On this regards, multiprocess package in Python is selected. For this integrate, the following main function should be developed. Note that number of processes is set by given value, N_proc, that can be 'None' variable. Then, the multiprocess package automatically allocate one thread for each logical processor. The "partial" package is loaded because the allocation of multiprocess does not taking various arguments, so the package allow to transfer more argument that might be needed for the plotting function.

```
1    from multiprocessing import Pool
```

```
    from functools import partial

    def plot_t(given_traj, t):
    # statement (detail for plot_t is omitted)

    if __name__ == '__main__':
    pool = Pool(processes=N_proc)
    with open(fn, 'r') as f:
    N_cols = 2*N_dimension*Np + 1
    tmp_arr = zeros([N_proc, N_cols])
    cnt_line = 0
    c_t = arange(N_proc)
    for line in f:
    tmp_str = line.split('\t')
    for i in range(N_cols):
    tmp_arr[cnt_line%N_proc, i] = float(tmp_str[i])
      cnt_line += 1
      if (cnt_line <> 0 and cnt_line%N_proc == 0):
      pool.map(partial(plot_t, tmp_arr), c_t)
      c_t += N_proc
```

Finally, we have all the figure with proper image file format. In this processing, author is prefer to use png format rather than pdf since it need for ffmepg incoding. The basic statement for ffmpeg is used as follow. But, codec, output file, and various properties can be set with the ffmpeg package. For instance, h268 codec is useful for Mac compatible movie.

```
    ffmpeg -i t%08d.png -vcodec copy out.mov
```

G.2 Trajectory Conversion

Since the simulation is using periodic boundary condition (PBC), all the trajectory is already cut inside of certain box. For some reasons of post-processing, MSD and so on, we need to recover from PBC image to real beads trajectory. This conversion is easily doable with appropriate condition for the identity for trajectory. Here, I have used half of box dimension is changed from the previous output step, the trajectory is identified as jump and post-processing code will recover it. Note that the trajectory output frequency is lower than the all time steps, if this conversion is needed, we have to reduce the overall time steps. The core parts of the code is listing as below.

```
    def sign(x):
    if x < 0.:
    return -1.
    return 1.

    def inv_PBC(x_now, x_next, LB):
    dX = x_next - x_now
    if abs(dX) > 0.5*LB:
    return inv_PBC(x_now, x_next - sign(dX)*LB, LB)
    return x_next

    for i in range(NP):
    for k in range(ND):
```

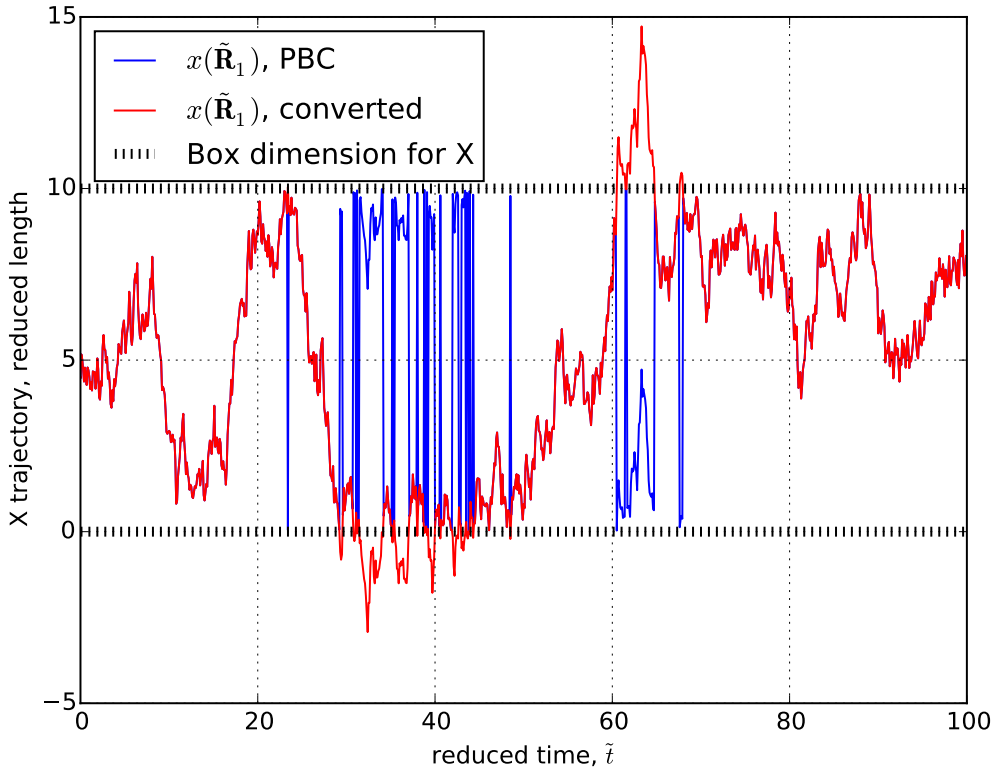


Figure G.1: Test results for trajectory conversion. Blue color represent the trajectory using periodic boundary condition (PBC) and the red color represent the converted data. The test is done using pure Brownian motion with 100 reduced time step, and trajectory is involved only for x-coordinate of the first beads among 100 beads on the system.

```
14      index_Rik = 2*ND*i + 1 + k
15      for t in range(1, Nt):
16      dat[t, index_Rik] = inv_PBC(dat[t-1, index_Rik], dat[t, index_Rik], LB)
```

Note that the inverse mapping function, inv_PBC, is using recursive call, that is because the main process override the opened dat array for effienciency issue, that sometimes amplified the existing gap during correction. Therefore, it is of importance to check the range is valid for inverse mapping using recursive call. From this conversion, we can easily expect the trajectory from figure G.1. Note that the judgment is based on the half of box dimension.

G.3 Pair Correlation Function

G.3.1 Theoretical Background

The pair correlation distribution, $\rho(\mathbf{r}_1, \mathbf{r}_2$, is one of the important distribution function to show the structure information. There are various relation exist with spatial or reciprocal information.

For given canonical ensemble (N, V, T), let Z be configurational integral, partition function:

$$Z = \int d\mathbf{\Gamma}_1^N \exp\left(-\beta U(\mathbf{\Gamma}_1^N)\right), \tag{G.1}$$

where β be Boltzmann factor, $\mathbf{\Gamma}_1^N = \{\mathbf{r}_1, \cdots, \mathbf{r}_N\}$ and $d\mathbf{\Gamma}_1^N = d\mathbf{r}_1 \cdots d\mathbf{r}_N$. Then, the probability of an elementary configuration is expressed by

$$P(\mathbf{\Gamma}_1^N)d\mathbf{\Gamma}_1^N = \frac{\exp\left(-\beta U(\mathbf{\Gamma}_1^N)\right)}{Z} d\mathbf{\Gamma}_1^N, \tag{G.2}$$

where $d\mathbf{\Gamma}_1^N = d\mathbf{r}_1 \cdots d\mathbf{r}_N$. Let $P(\mathbf{\Gamma}_1^N)$ be configurational distribution over set of N-number of state variables, $\mathbf{\Gamma}_1^N = \{\mathbf{r}_1, \cdots, \mathbf{r}_N\}$. Since the configurational distribution $P(\mathbf{\Gamma}_1^N)$, is function of potential energy, $U(\mathbf{\Gamma}_1^N)$, it cannot account only for single particle. Hence, we can eliminating dependency of other particles using integrate over configurational space:

$$P(\mathbf{\Gamma}_1^n) = \int d\mathbf{\Gamma}_{n+1}^N P(\mathbf{\Gamma}_1^N), \tag{G.3}$$

which is joint PDF for finding particle $\{1, 2, \cdots, n\}$ at $\{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_n\}$, respectively. It is called *specific* PDF because of the index of particles are fixed. Let $\mathbf{\Omega}_1^n = \{\mathbf{r}_1, \cdots, \mathbf{r}_n\}$ be the set of particles which is arbitrarily chosen. If the particles are identical (indistinguishable system), we can express *generic* PDF when we allow to choose particles, arbitrarily:

$$\rho(\mathbf{\Omega}_1^n) = \frac{N!}{(N-n)!} P(\mathbf{\Gamma}_1^n) \tag{G.4}$$

For simplification, consider the given material is isotropic. Then the first order generic PDF becomes its density: $\rho(\mathbf{r}_1) = \rho = N/V$. When the particles independent to each other, the joint probability becomes

$$P_{id}(\mathbf{\Gamma}_1^N) = \prod_{k=1}^{N} P(\mathbf{r}_k) = \rho^N. \tag{G.5}$$

Therefore, the generic PDF for independent particle becomes

$$\rho_{id}(\mathbf{\Omega}_1^n) = \frac{N!}{(N-n)!} P(\mathbf{\Gamma}_1^n) = \frac{N!}{(N-n)!} \rho^n. \tag{G.6}$$

The correlation function g is given by the fraction of generic PDF of the system to the independent case:

$$g(\mathbf{\Omega}_1^n) \equiv \frac{\rho(\mathbf{\Omega}_1^n)}{\rho_{id}(\mathbf{\Omega}_1^n)} = \frac{P(\mathbf{\Gamma}_1^n)}{P_{id}(\mathbf{\Gamma}_1^n)} = \frac{1}{\rho^n} P(\mathbf{\Gamma}_1^n). \tag{G.7}$$

For convenience, $h(\mathbf{\Omega}) = g(\mathbf{\Omega}) - 1$ is frequently used as correlation function. Inversely, we have *specific* PDF:

$$P(\mathbf{\Gamma}_1^n) = \rho^n g(\mathbf{\Omega}_1^n). \tag{G.8}$$

When order is 2, the correlation functions is called pair correlation function:

$$g(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{\rho^2} P(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{\rho^2} \int d\mathbf{\Gamma}_3^N P(\mathbf{r}_1, \mathbf{r}_2) \tag{G.9}$$

$$= \frac{1}{\rho^2} \int d\mathbf{\Gamma}_3^N \frac{\exp(-\beta U(\mathbf{\Gamma}_1^N))}{Z}. \tag{G.10}$$

For spherically symmetric system, the probability between $\mathbf{r}_i$ and $\mathbf{r}_j$ becomes

$$P(\mathbf{r}_i, \mathbf{r}_j) = P(\mathbf{r}_i - \mathbf{r}_j). \tag{G.11}$$

For further details, see Chandler (1987).

G.3.2 Measuring from Simulation Data

Let $\langle n_i(r, \Delta r; t)\rangle_t$ be the average number in the shell at distance between r and $r + \Delta r$ at time t:

$$\langle n_i(r, \Delta r; t)\rangle_t = \frac{1}{T}\sum_{j=1}^{T} n_i(r; t_j). \tag{G.12}$$

Note that $\langle n_i(r)\rangle$ is independent on i, then we can define ensemble average of number of particles as

$$\langle n(r, \Delta r; t)\rangle \equiv \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{j=1}^{T} n_i(r, \Delta r; t_j). \tag{G.13}$$

Let assumed the particles are uncorrelated, the same assumption of independent probability between particles in equation (G.5), then we averaged number in the shell at distance r:

$$\langle n(r, \Delta r)\rangle_{unc} = \rho V(r, \Delta r)(N-1)/N, \tag{G.14}$$

where $V(r, \Delta r)$ is Volume of shell between r and $r + \Delta r$, and ρ is density. The alternative form of radial distribution function is the ratio between averaged number of particles on shell with distance r and the uncorrelated number:

$$g(r) = \frac{\langle n(r, \Delta r; t)\rangle}{\langle n(r, \Delta r)\rangle_{unc}} \tag{G.15}$$

$$= \frac{1}{\rho V(r, \Delta r)(N-1)T}\sum_{i=1}^{N}\sum_{j=1}^{T} n_i(r, \Delta r; t_j). \tag{G.16}$$

If the given material is liquid-like and is not highly ordered case, the averaged number at long distance becomes un-correlated averaged number:

$$\lim_{r\to\infty} g(r) = \frac{\lim_{r\to\infty}\langle n(r, \Delta r; t)\rangle}{\langle n(r, \Delta r)\rangle_{unc}} = \frac{\langle n(r, \Delta r)\rangle_{unc}}{\langle n(r, \Delta r)\rangle_{unc}} = 1. \tag{G.17}$$

Because of system size, the density ρ should be replaced by local density, i.e., the counted number for all the particles divided by total area that is considered:

$$\rho_D = \frac{1}{T(N-1)V_D}\sum_{k=1}^{N-1}\langle n(r, \Delta r; t)\rangle \equiv \frac{n_D}{T(N-1)V_D}, \tag{G.18}$$

where subscript D denote domain of computation. For simplification, let $\rho(r)$ be the averaged density between r and $r + \Delta r$:

$$\rho(r) = \frac{\langle n(r, \Delta r; t)\rangle}{V(r, \Delta r)}. \tag{G.19}$$

Note that $\rho(r)$ is not affected by Δr since the counted number for histogram directly proportional to the volume: $\langle n(r, \Delta r; t)\rangle \propto V(r, \Delta r)$. The equation (G.16) becomes

$$g(r) = \frac{\rho(r, \Delta r)}{\rho_D} = \frac{V_D}{n_D}\frac{\langle n(r, \Delta r; t)\rangle}{V(r, \Delta r)}, \tag{G.20}$$

which is the same formular compared with GROMACS.

For practical purpose, the pairs are only accounted for once rather than twice. On this regards, the $N - 1$ factors replaced by $(N - 1)/2$ which reduced the time consumption dramatically.

```
def get_ddf(traj, ts, Np, N_dimension, box_dimension, cut_ratio):
    ddf = []
    for t in ts:
        for i in range(Np-1):
            for j in range(i+1, Np):
                d = lin.norm(get_rel_vec(traj, t, i, j, N_dimension, box_dimension))
                if d<cut_ratio*box_dimension:
                    ddf.append(d)
    return ddf

def get_rdf_ref(traj, ts, dr, Np, N_dimension, box_dimension, cut_ratio):
    Nr = int(cut_ratio*box_dimension/dr)
    rdf = zeros([Nr, 3])
    rdf[:,0] = arange(0, cut_ratio*box_dimension, dr)
    ddf = get_ddf(traj, ts, Np, N_dimension, box_dimension, cut_ratio)
    N_tot = size(ddf)
    Nt = size(ts)
    for r in ddf:
        rdf[int(r/dr), 1] += 1
    if (N_dimension == 3):
        Vr = (4./3.)*pi*((rdf[:,0]+dr)**3.0 - rdf[:,0]**3.0)
        Vrmax = (4./3.)*pi*(cut_ratio*box_dimension)**3.0
        rho_local = N_tot/(Nt*0.5*(Np-1)*Vrmax)
    elif (N_dimension == 2):
        Vr = pi*((rdf[:,0]+dr)**2.0 - rdf[:,0]**2.0)
        Vrmax = pi*(cut_ratio*box_dimension)**2.0
        rho_local = N_tot/(Nt*0.5*(Np-1)*Vrmax)
    print 'rho_local = ', rho_local
    rdf[:,2] = rdf[:,1]/(Vr*0.5*(Np-1)*Nt*rho_local)
    rdf[0, 2] = 0.
    return rdf
```

The cut-ratio typically set by half of box dimension because of computing efficiency. When this is half box dimension, we can apply minimum image convention easily. If it is larger than half of box dimension, we have to checked all the possible map for counting. If it is smaller than half of box dimension, we loose some information which is meaningful for us. The results is described on figure G.2 which is used the code reported on here.

G.4 Structure Factor

G.4.1 Basics

The basic definition for the structure factor is given by

$$S(\mathbf{k}) = N^{-1} \left\langle \sum_{i,j=1}^{N} \exp\left[i\mathbf{k}\cdot(\mathbf{r}_i - \mathbf{r}_j)\right] \right\rangle \quad \text{for } \mathbf{r}_i, \mathbf{r}_j \in \mathbf{\Gamma}_1^N, \tag{G.21}$$

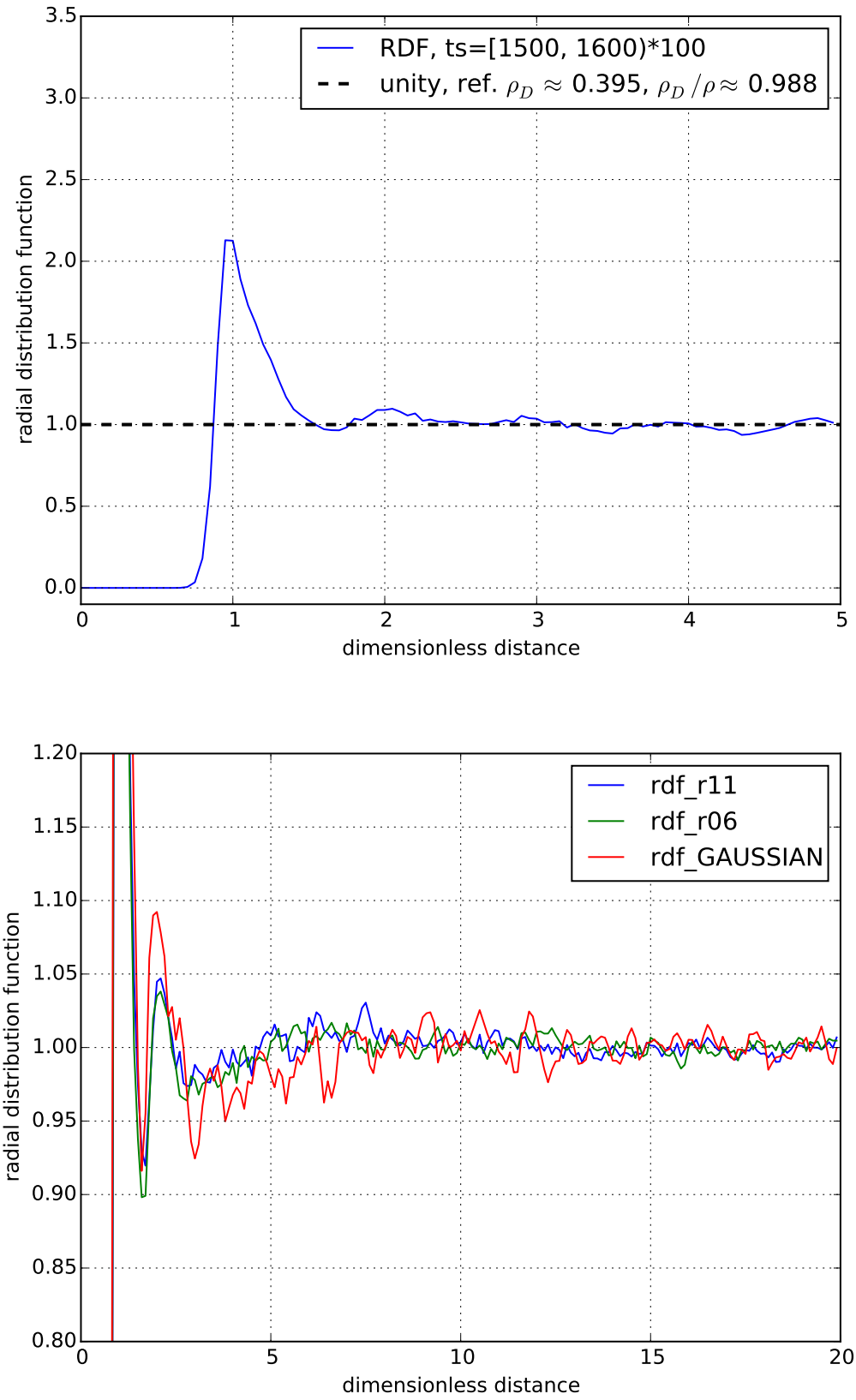


Figure G.2: Radial distribution function for NP640/1600AREA example. Time average is applied over 100 times steps. Upper case is simple time average for FENE with R_m/R_0 ratio 11. The lower case is the comparison between FENE ratio 11 and 6 with Gaussian (red color) case.

where $\mathbf{k}$ is wave vector. The given formula can be analyzed by two parts:

$$S(\mathbf{k}) = N^{-1}\left\langle \sum_{i,j}\delta_{ij}\right\rangle + N^{-1}\left\langle \sum_{i,j\neq i}\exp(i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j))\right\rangle \tag{G.22}$$

$$= 1 + N^{-1}\sum_{i,j\neq i}\frac{1}{Z}\int d\mathbf{\Gamma}_1^N \exp(-\beta U(\mathbf{\Gamma}_1^N))\exp(i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j)) \tag{G.23}$$

$$= 1 + N^{-1}\sum_{i,j\neq i}\int d\mathbf{r}_i d\mathbf{r}_j \left[\exp(i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j))\int d\left(\mathbf{\Gamma}_1^N\backslash\{\mathbf{r}_i,\mathbf{r}_j\}\right)\frac{\exp(-\beta U(\mathbf{\Gamma}_1^N))}{Z}\right] \tag{G.24}$$

$$= 1 + N^{-1}\sum_{i,j\neq i}\int d\mathbf{r}_i d\mathbf{r}_j P(\mathbf{r}_i,\mathbf{r}_j)\exp(i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j)). \tag{G.25}$$

Let assumed that the system is spherically symmetry, then the probabilty $P(\mathbf{r}_i,\mathbf{r}_j)$ becomes the probability for its relative vector, $P(\mathbf{r}_i-\mathbf{r}_j)$, which implies

$$S(\mathbf{k}) = 1 + N^{-1}\sum_{i,j\neq i}\int d\mathbf{r}_i d\mathbf{r}_j P(\mathbf{r}_i-\mathbf{r}_j)\exp(i\mathbf{k}\cdot(\mathbf{r}_i-\mathbf{r}_j)) \tag{G.26}$$

$$= 1 + N^{-1}\sum_{i,j\neq i}\int d\mathbf{r}_i d\mathbf{r}_j \mathscr{F}\left[P(\mathbf{r})\delta(\mathbf{r}-(\mathbf{r}_i-\mathbf{r}_j))\right] \tag{G.27}$$

$$= 1 + N^{-1}\mathscr{F}\left[P(\mathbf{r})\sum_{i,j\neq i}\int d\mathbf{r}_i d\mathbf{r}_j \delta(\mathbf{r}-(\mathbf{r}_i-\mathbf{r}_j))\right], \tag{G.28}$$

where $\mathscr{F}$ denote spatial Fourier transform:

$$\mathscr{F}[f(\mathbf{r})] = \int d\mathbf{r} f(\mathbf{r})\exp(i\mathbf{k}\cdot\mathbf{r}). \tag{G.29}$$

The integration part becomes

$$\int d\mathbf{r}_i d\mathbf{r}_j \delta(\mathbf{r}-(\mathbf{r}_i-\mathbf{r}_j)) = \int d\mathbf{r}_i d\mathbf{r}_{ij}\delta(\mathbf{r}-\mathbf{r}_{ij}) \tag{G.30}$$

$$= \int d\mathbf{r}_i 1 \tag{G.31}$$

Derivation is on-going

G.4.2 Related with Radial Distribution Function, 3D

$$S(\mathbf{q}) = 1 + \rho\mathscr{F}\left[g(\mathbf{r})-1\right], \tag{G.32}$$

where $g(\mathbf{r})$ is given radial distribution function with respect to distance vector, $\mathbf{r}$, and $\mathscr{F}$ is Fourier transform from spatial vector, $\mathbf{r}$, to the its reciprocal (wave) vector, $\mathbf{q}$. Therefore, the structure factor becomes

$$S(\mathbf{q}) = 1 + \rho\int d\mathbf{r}\, e^{i\mathbf{q}\cdot\mathbf{r}}\left[g(\mathbf{r})-1\right]. \tag{G.33}$$

For isotropic system (orientational symmetric), the integration part becomes further simple. For simplification, let assumed that the direction of axis x is the same with the given wave vector, **q**:

$$\begin{aligned}
S(q) &= 1 + \rho \int r^2 \sin\theta \, dr d\theta d\phi \, \exp(iqr\cos\theta)(g(r)-1) \\
&= 1 + 2\pi\rho \int dr\, r^2 (g(r)-1) \int_0^\pi d\theta \, \sin\theta \exp\left(iqr\cos\theta\right) \\
&= 1 + 2\pi\rho \int dr\, r^2 (g(r)-1) \int_{-1}^1 dt \, \exp(iqrt) \\
&= 1 + 2\pi\rho \int dr\, r^2 (g(r)-1) \frac{\exp(iqr) - \exp(-iqr)}{iqr} \\
&= 1 + 2\pi\rho \int dr\, r^2 (g(r)-1) \frac{2}{qr} \sin(qr), \quad (\because \text{Euler's equation}) \\
&= 1 + 4\pi \frac{\rho}{q} \int dr\, r \sin(qr)(g(r)-1). \qquad \text{(G.34)}
\end{aligned}$$

G.4.3 Isotropic Structure Factor for 2 Dimensional Case

Interestingly, the 2-dimensional case for the isotropic structure factor has more complicated form since the change of variables is not simply work. Similar to the 3-dimensional case, the axis is tuned with the given wave vector, **q**. Then we have

$$S(q) = 1 + \rho \int r \, dr d\theta \, \exp(iqr\cos\theta)(g(r)-1). \qquad \text{(G.35)}$$

Here, the integration of $\exp(iqr\cos\theta)$ over angle θ does not shows the closed form, but we can express it as 0-th order of Bessel function of the first kinds:

$$2\pi J_0(x) = \int_0^{2\pi} d\theta \, \exp(ix\sin\theta) = \int_0^{2\pi} d\theta \, \exp(ix\cos\theta), \qquad \text{(G.36)}$$

where general Bessel function is possibly expressed by trigonometric functions:

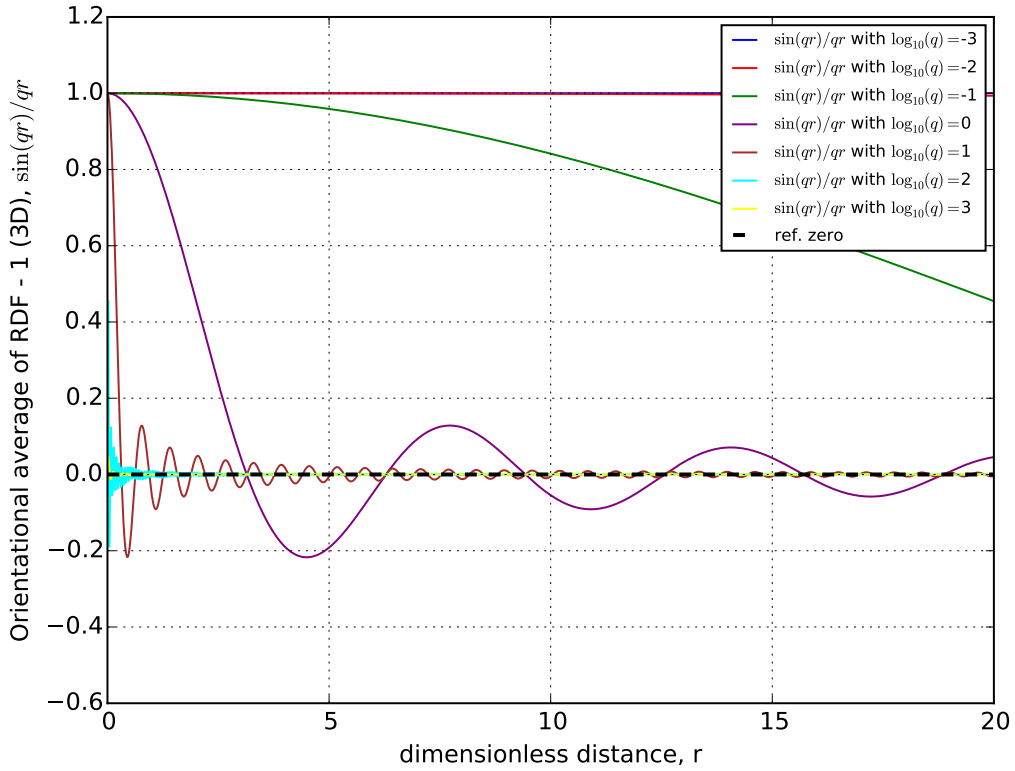
$$\pi J_n(x) = \int_0^\pi d\theta \, \cos(x\sin\theta)\cos(n\theta), \quad \text{n is even} \qquad \text{(G.37)}$$

$$\pi J_n(x) = \int_0^\pi d\theta \, \sin(x\sin\theta)\sin(n\theta), \quad \text{n is odd.} \qquad \text{(G.38)}$$

For further details, see Arfken and Weber (2008). On this regards, the isotropic structure factor for 2-dimensional case is expressed by

$$S(q) = 1 + 2\pi\rho \int dr\, r J_0(qr)(g(r)-1), \qquad \text{(G.39)}$$

which cannot be solved by analytically. The given $J_0(qr)$ is oscillatory decreasing function with respect to qr (reported in figure G.3) but $rJ_0(qr)$ is oscillatory increasing function. Therefore, $g(r)-1$ should be conversed to zero faster than that of $rJ_0(qr)$. At the moment, whether it is trivial or not is ambiguous for me. Figure G.4 is computed from FENE chains with the ratio 11. The Gaussian case, FENE with different ratio, and longer computed case will be reported on here.



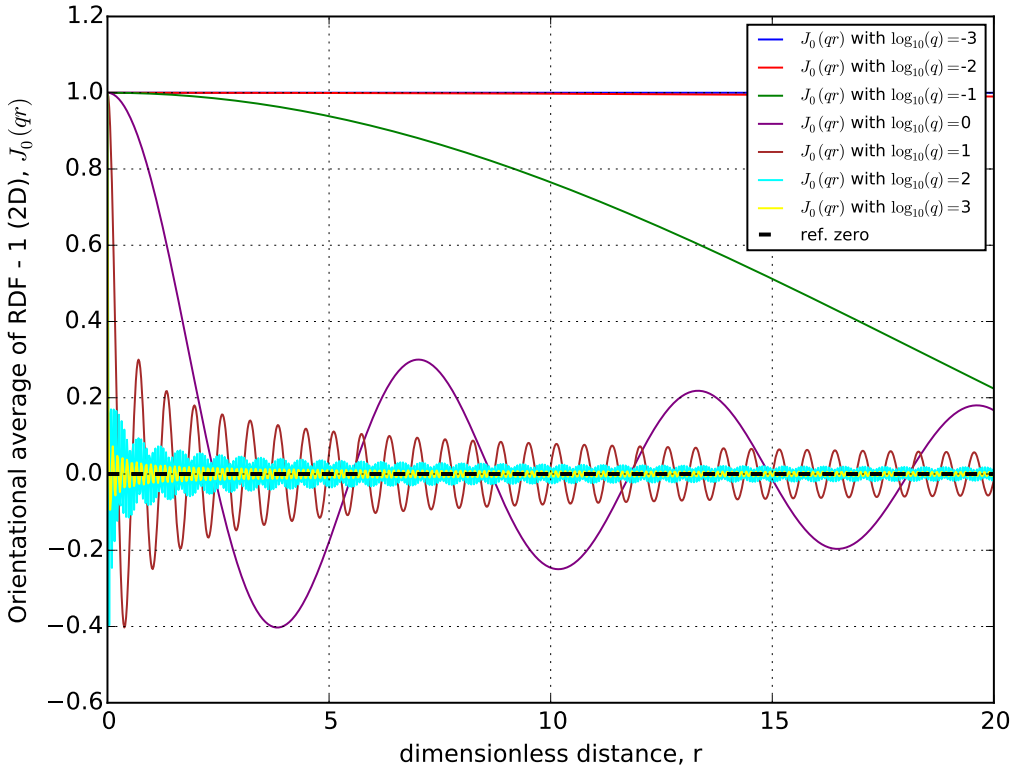


Figure G.3: Orientational average over $g(r) - 1$ in 3-dimensional (up) and 2-dimensional (down) cases. The upper case is related with the Debye equation while the lower case is expressed by the zeroth order of Bessel function with the first kind.

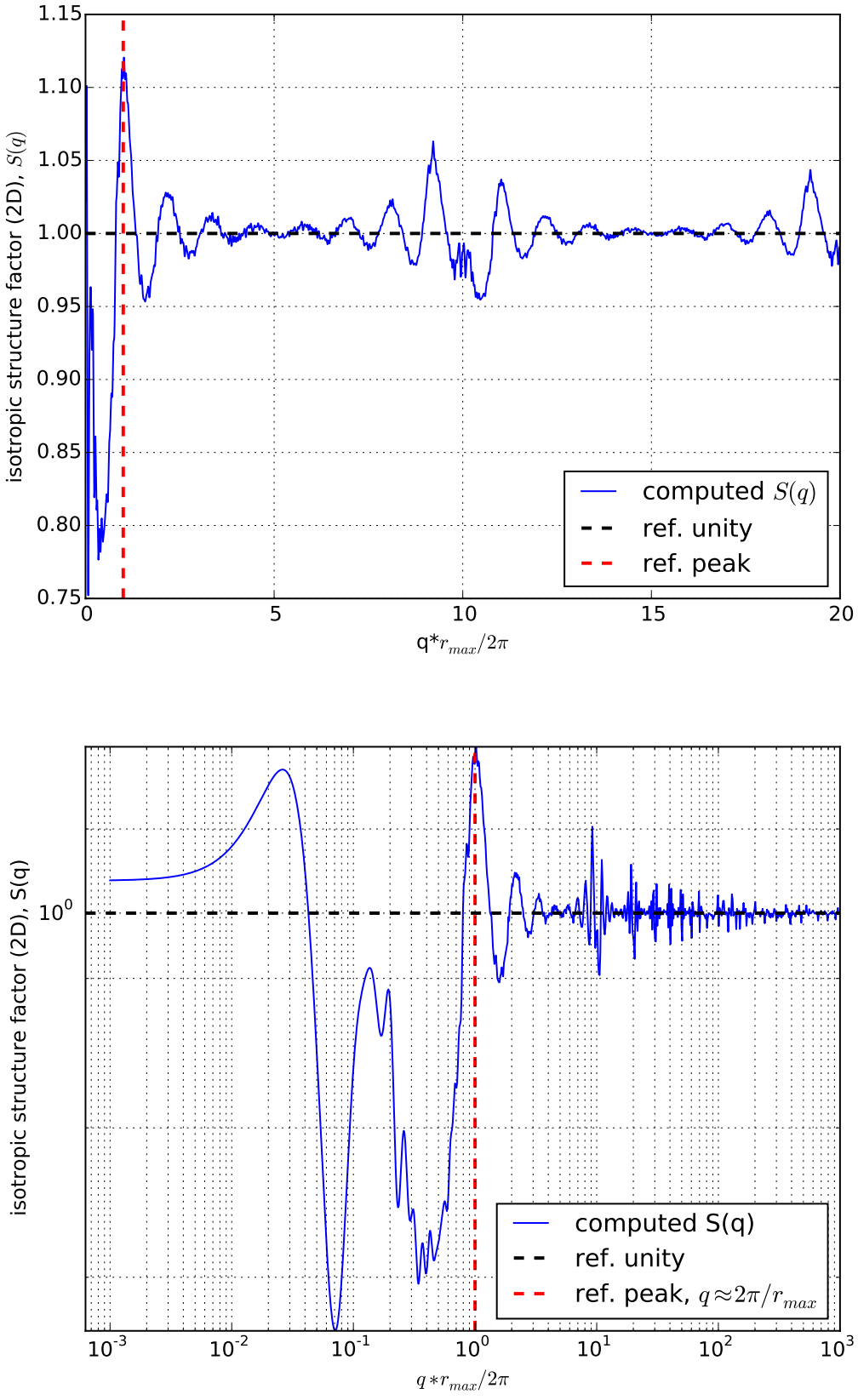


Figure G.4: Results of FENE with ratio $R_M/R_0 = 11$. The given radial distribution is exactly the same with figure G.2 (lower case). The upper case is computed structure factor in linear scale, and the bottom is the double log scales for the structure factor.

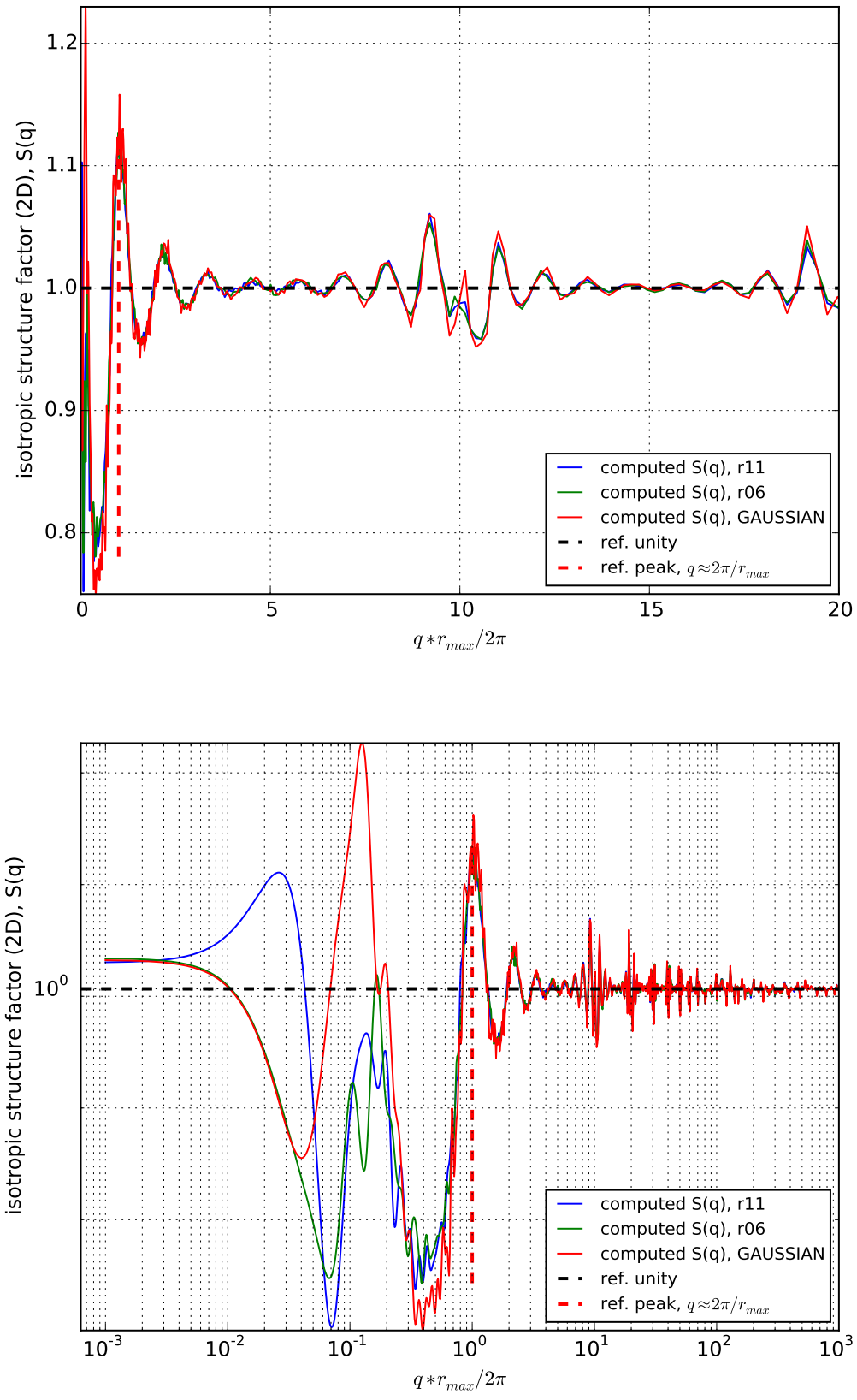


Figure G.5: Results with different FENE ratio and GAUSSIAN. The given radial distribution is exactly the same with figure G.2 (lower case). The upper case is computed structure factor in linear scale, and the bottom is the double log scales for the structure factor.

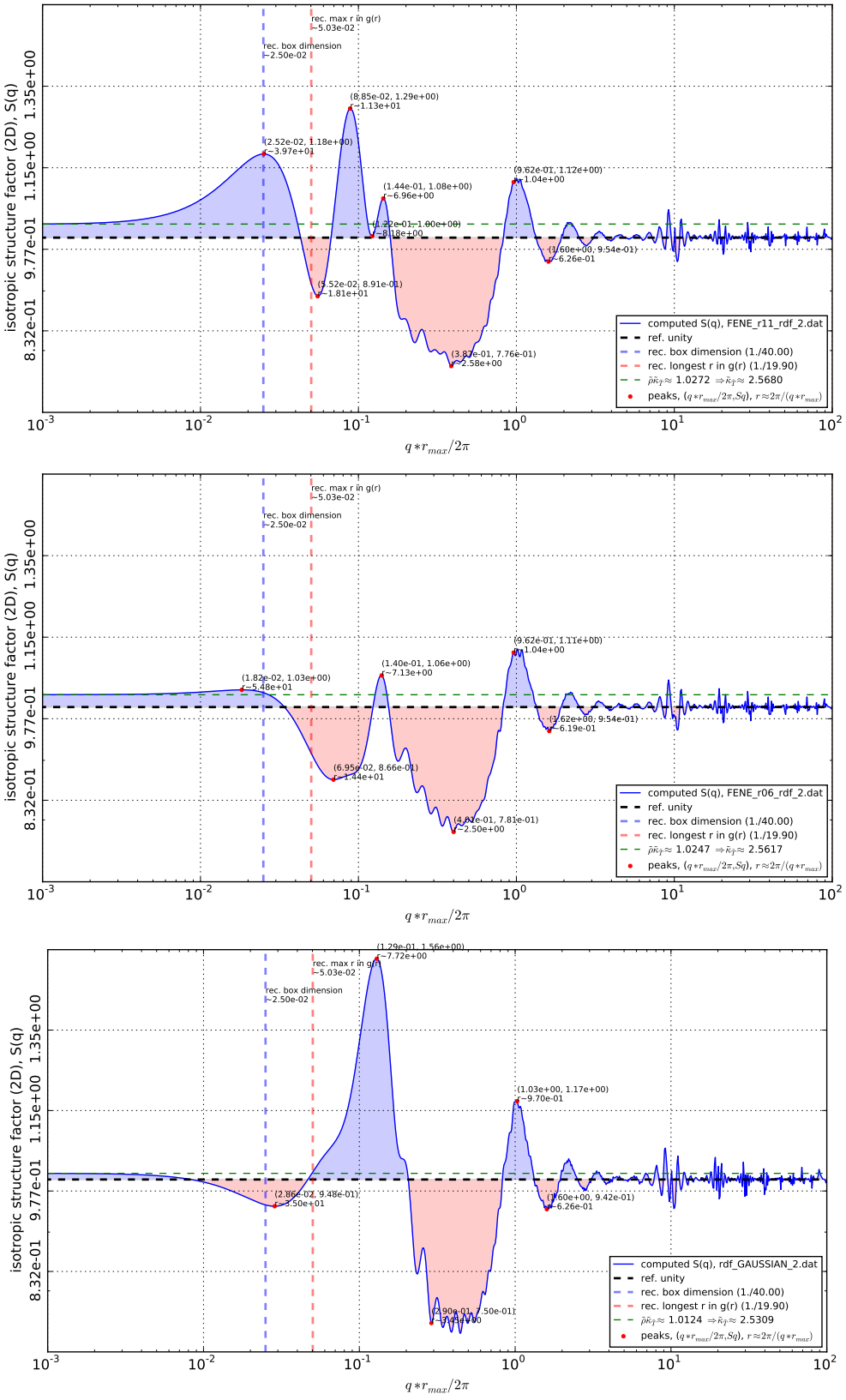


Figure G.6: Computed structure factor for different conditions. From top to bottom, the figure represent FENE with r11, r06, and Gaussian, respectively.

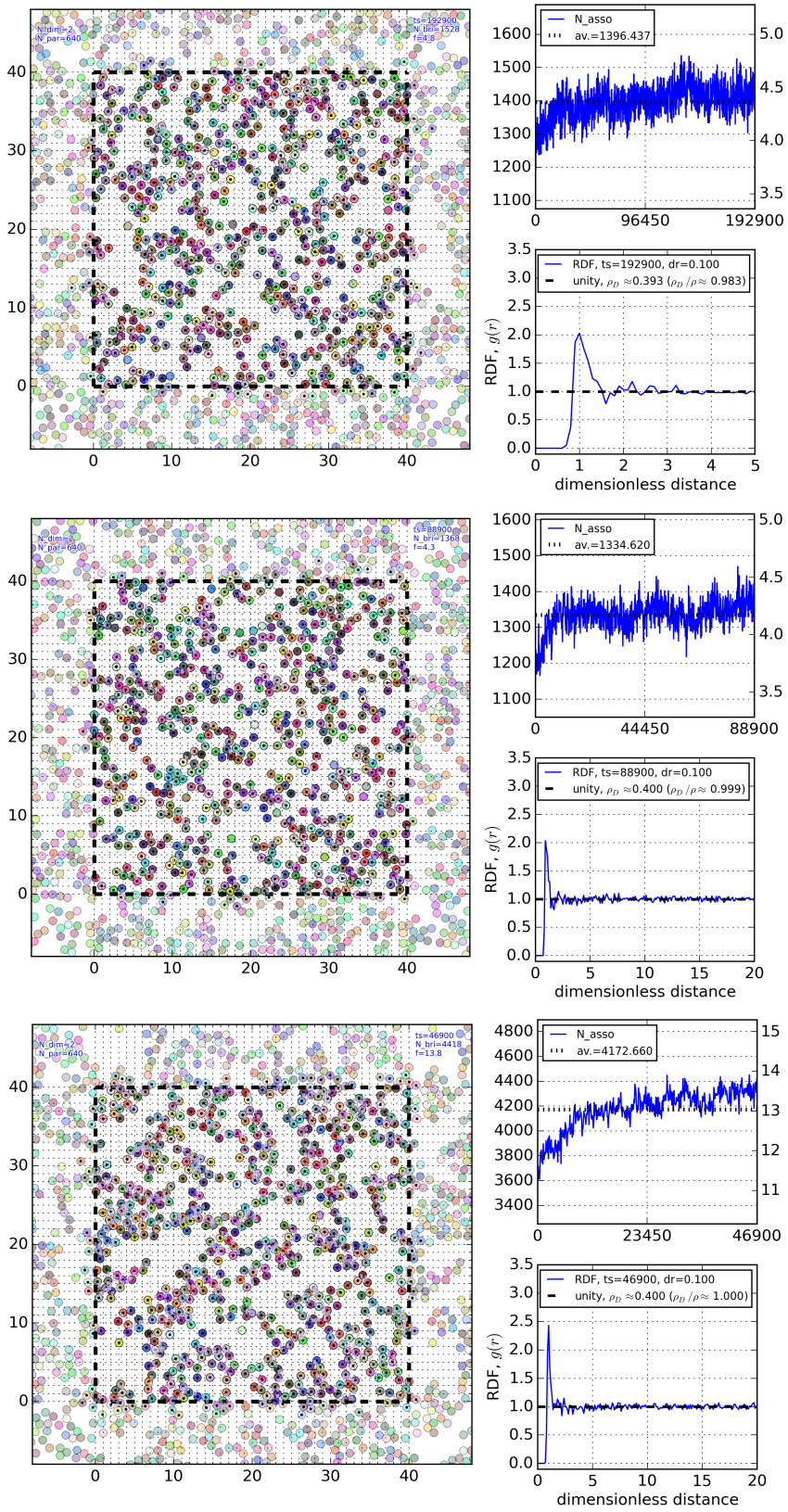


Figure G.7: The examples of trajectory with different conditions. From top to bottom, the figures represent FENE with ratio 11, FENE with ratio 6, and GAUSSIAN connectors, respectively.

Bibliography

Allen, P. and D. Tildesley (1989). *Computer Simulation of Liquids*. Oxford Science Publ. Clarendon Press. ISBN: 9780198556459.

Arfken, G. and H. Weber (2008). *Mathematical methods for physicists*. Elsevier Acad. Press. ISBN: 9780120598762.

Chandler, D. (1987). *Introduction to Modern Statistical Mechanics*. Oxford University Press. ISBN: 9780195042771.

Cho, K. S. (2013). "Power series approximations of dynamic moduli and relaxation spectrum". *Journal of Rheology* 57.2, pp. 679–20.

Cohen, A. (1991). "A Pade approximant to the inverse Langevin function". *Rheologica Acta* 30, pp. 270–273.

Greiner, A., W. Strittmatter, and J. Honerkamp (1988). "Numerical integration of stochastic differential equations". *Journal of Statistical Physics*.

Harold R. Warner, J. (1972). "Kinetic theory and rheology of dilute suspensions of finitely extendible dumbbells". *Industrial and Engineering Chemistry Fundamentals* 11.3, pp. 379–387.

Ianniruberto, G. and G. Marrucci (Aug. 2015). "New Interpretation of Shear Thickening in Telechelic Associating Polymers". *Macromolecules* 48.15, pp. 5439–5449.

James, H. M. and E. Guth (1943). "Theory of the elastic properties of rubber". *Journal of Chemical Physics* 11.10, p. 455.

Kuhn, V. W. and F. Grün (1942). "Beziehungen zwischen elastischen Konstanten und Dehnungsdoppelbrechung hochelastischer Stoffe". *Kolloid-Zeitschrift* 16.3, pp. 248–271.

Treloar, L. (1975). *The Physics of Rubber Elasticity*. Monographs on the physics and chemistry of materials. Oxford University Press, USA. ISBN: 9780191523304.