Brownian Dynamics Simulation of Colloidal Suspension with Hydrodynamic Interactions

By

Rohit Nikam

4th year undergraduate, Department of Chemical Engineering

Submitted as a course project (Rheology of Complex Fluids -CL 651)

Indian Institute of Technology Bombay

Mumbai, India

October 2014

Abstract

Brownian dynamics simulation has been used to study the dynamics of a dilute system containing monodisperse uniformly charged Brownian hard spheres in a quiescent viscous fluid in a Stokes flow regime. The particles influence each other via screened Coulomb interactions and hydrodynamic interactions. Ermak-McCammon algorithm derived from Langevin equation for a system of particles is implemented to simulate their motions. The algorithm is applicable at Brownian time scale, at which the momenta of particles are fully relaxed but positions are not. Hydrodynamic interactions are assumed to be pairwise additive and their effect is assumed to be instantaneous. Effective suspension diffusivity and the contribution to the resultant stress in the system by electrostatic forces is calculated at the end of simulation. Programing is done in MATLAB.

Table of Contents

1 Introduction

Particles having size range of 1 nm to 1μm are called colloidal particles. Systems having colloidal particles of variety of shapes dispersed in a continuum fluid medium are designated as colloidal suspensions. Such systems come under mesoscopic length scales. The system behavior at such length scale is significantly different than those with higher length scales. The origin of this disparity is the enormous interfacial area available in mesoscale systems which leads to very high interfacial energy. Hence the influence of surface forces viz. van der Waal interactions becomes comparable to that of body forces such as gravitational force or bulk electrostatic or magnetic forces.

The unique characteristic of the motion of colloidal particles in a fluid medium is their "Brownian motion". Many times this phenomenon is misinterpreted as the motion caused by 'uncompensated' collisions of colloidal particle with solvent molecules, which result in a kick to the particle and the particle otherwise would have been totally inert dynamically. If it is experimentally observed that the colloidal particle is diffusing, it does not necessarily mean that there are unseen solvent molecules, since the colloidal particle itself has intrinsic translational kinetic energy which has nothing to do with the surrounding environment.

The dynamics of colloidal systems has a series of time scales, mainly due to large difference between colloidal length scales and sizes of solvent molecules. For solvent molecules, colloidal particle is huge and extremely sluggish object but the colloidal particle sees a dense swarm of surrounding molecules colliding with it at enormously high frequency. The time scales involved in such systems are

1. *Molecular collision time*: Fastest (10^{-13} sec)
2. *Momentum relaxation time*: Time required to span the velocity distribution for colloidal particle
3. *Diffusive/Brownian time scale*: Mean squared displacement grows linearly in time
4. *Configurational relaxation time*: Net colloidal displacement becomes comparable to its radius
5. *Angular momentum relaxation time*: Close to linear momentum relaxation time
6. *Rotational relaxation time*: Time to significantly change the orientation. Close to diffusive time scale
7. *Brownian collision time*: Determines coagulation kinetics in colloids

8. *Hydrodynamic decay time*: Time scale of disturbance transfer through the solvent due to motion of colloidal particle

Our system of interest contains 20 colloidal particles having 10 nm radius and uniform surface charge density suspended in a dielectric, viscous and quiescent medium having no externally imposed velocity gradients. Solvent is assumed on a coarse grained level as a continuum with no density gradients.

2 Nature of Inter-particle Interactions

In our system, solvation forces due to solvent are neglected. Particles interact with each other via screened Coulomb interactions and hydrodynamic interactions. Both are long range interactions and vary as r^{-1}.

2.1 Screened Coulomb Interaction

Yukawa potential has been chosen as the model for this interaction. It is given by

$$E = A\frac{e^{-\lambda r}}{r} \tag{1}$$

where r is the distance between the particles, λ is the screening length and A is the scaling constant.

2.2 Hydrodynamic Interaction

A Brownian particle that attained a velocity at a certain time induces a fluid flow in the solvent. This fluid flow propagates through the solvent and encounters other Brownian particles, which are thus affected in their motion, giving rise to an interaction which depends on their velocities and positions.

In the diffusive time scale, the propagation of fluid flow disturbances is so fast that momenta and positions of Brownian particles hardly change during the time the disturbance takes to reach other Brownian particles. So this interaction is assumed instantaneous if we are simulating the system in diffusive time scale.

In the Stokes flow regime, the velocity of Brownian particle is linearly related to the surface force per unit area exerted on it by the surrounding fluid. So it is possible to write this relationship as

$$\mathbf{F}_i^h = -\sum_{j=1}^{N} \Gamma_{ij}(r_1, r_2, \ldots r_N) \cdot \mathbf{v}_j \qquad (2)$$

which says that hydrodynamic force on particle i is a linear function of velocities of other Brownian particles. Γ_{ij} is 3×3 microscopic mobility matrix which depends on the particle position and not dependent on velocity. To calculate hydrodynamic interactions, these matrices have to be explicitly calculated. Minus sign is a result of opposite directions of hydrodynamic force and velocity.

Off-diagonal mobility matrices Γ_{ij} with $i \neq j$ describe hydrodynamic interaction between particles i and j. In case of diagonal mobility matrices Γ_{ii}, fluid flow from the movement of i^{th} sphere "reflects" back from other Brownian particles to i^{th} sphere, thus exerting a force on that particle in addition to the friction force of the isolated particle.

Another matrix called microscopic diffusion matrix D_{ij} which is defined as,

$$\boldsymbol{\Gamma}^{-1} = \beta \mathbf{D} = \beta \begin{bmatrix} \boldsymbol{D}_{11} & \cdots & \boldsymbol{D}_{1N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{D}_{N1} & \cdots & \boldsymbol{D}_{NN} \end{bmatrix} \qquad (3)$$

So equation (2) can be re-written as

$$\begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = -\beta \begin{bmatrix} \boldsymbol{D}_{11} & \cdots & \boldsymbol{D}_{1N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{D}_{N1} & \cdots & \boldsymbol{D}_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{F}_1^h \\ \vdots \\ \mathbf{F}_N^h \end{bmatrix} \qquad (4)$$

where D_{ij} is 3×3 microscopic diffusion matrix which connects the force F_i^h exerted by the fluid on the i^{th} Brownian particle, to the velocities v_j of Brownian particles. The reason for writing equation (2) in this way is that velocities are generally unknown and forces are known variables.

Fluid flow is governed by creeping flow equations which are Navier-Stokes equations without nonlinear convective terms.

$$-\boldsymbol{\nabla} p(\boldsymbol{r}) + \eta \boldsymbol{\nabla}^2 \mathbf{u}(\mathbf{r}) = \mathbf{0} \qquad (5)$$

$$\boldsymbol{\nabla} \cdot \mathbf{u}(\mathbf{r}) = \mathbf{0} \qquad (6)$$

For the velocity of Brownian particle, we can write as

$$\mathbf{u}(\mathrm{r}) = \int \mathbf{dr}'\mathbf{T}(\mathbf{r}-\mathbf{r}')\cdot\mathbf{f}^{ext}(\mathbf{r}') \tag{7}$$

where $\mathbf{f}^{ext} = \mathbf{f_0}\boldsymbol{\delta}(\mathbf{r}-\mathbf{r}')$ is a point force.

$\mathbf{T}$ is Green's function. Similar equation can be written for the linear relationship between pressure and force. Substituting these values of velocity and pressure in creeping flow equation gives T as

$$\mathbf{T}(\mathbf{r}) = \frac{1}{8\pi\eta}\frac{1}{r}\left[\mathbf{I}+\frac{\mathbf{rr}}{r^2}\right] \tag{8}$$

This is called Oseen tensor. For a system of N colloidal particles, equation (3) can be written as

$$\mathbf{u}(r) = \sum_{j=1}^{N}\int_{\partial V_j} dS'\mathbf{T}(r-r')\cdot\mathbf{f}_j(r') \tag{9}$$

Using no slip boundary condition at the particle surface,

$$\mathbf{v}_i + \boldsymbol{\Omega}\times(\mathbf{r}-\mathbf{r}_i) = \sum_{j=1}^{N}\int_{\partial V_j} dS'\mathbf{T}(r-r')\cdot\mathbf{f}_j(r') \text{ at } \mathbf{r}\in\partial V_i \tag{10}$$

Simplifying the Oseen tensor and *assuming that the distance between two particles is much larger than the particle diameter*, we get

$$\mathbf{v}_i = \frac{1}{6\pi\eta a}\mathbf{F}_i^h - \sum_{j\neq i}^{N}\mathbf{T}(\mathbf{r}_i-\mathbf{r}_j)\cdot\mathbf{F}_j^h \tag{11}$$

This is called 'Oseen approximation for microscopic diffusion matrices'. Comparing this result with equation (4) gives

$$\mathbf{D}_{ii} = D_0\,\mathbf{I}$$

$$\mathbf{D}_{ij} = k_B T\mathbf{T}(\mathbf{r}_i-\mathbf{r}_j) = \frac{3}{4}D_0\frac{a}{r_{ij}}\left[\mathbf{I}+\overline{\mathbf{r}_{ij}}\,\overline{\mathbf{r}_{ij}}\right]; i\neq j \tag{12}$$

where $D_0 = \frac{k_B T}{6\pi\eta a}$ is the characteristic diffusivity of Brownian particles according to Stokes-Einstein equation and $\bar{\mathbf{r}}_{ij} = \frac{(\mathbf{r}_i-\mathbf{r}_j)}{|(\mathbf{r}_i-\mathbf{r}_j)|}$. 'a' is the particle radius.

Now, Faxen's theorem on translational velocity of the spherical particle expresses it in terms of velocity of background fluid. Using the iterative *method of reflections*, *Rotne-Prager matrix* is formed, which contains an extra higher order term than the matrix after Oseen approximation (equation 12). It is formed by considering the effect of velocity of particle i on the velocity of particle j and expressing velocity of particle j in terms of the velocity of background fluid using Faxen's theorem. Continuing the method of reflections, the effect on the velocity of particle i by the consequential flow field induced by particle j, *the first order reflected fluid flow field,* is calculated again using Faxen's theorem. This leads to addition of one more higher order term in the expression for the diffusion matrix. This matrix is more accurate than Rotne-Prager matrix.

The resultant diffusion matrices are

$$\mathbf{D}_{ij} = D_0\mathbf{I} + D_0 \sum_{j=1,\, j\neq i}^{N} \left[A_s\left(r_{ij}\right)\overline{\mathbf{r}_{ij}}\,\overline{\mathbf{r}_{ij}} + B_s\left(r_{ij}\right)\left(\mathbf{I} - \overline{\mathbf{r}_{ij}}\,\overline{\mathbf{r}_{ij}}\right)\right] \tag{13}$$

$$\mathbf{D}_{ij} = D_0\left[A_c\left(r_{ij}\right)\overline{\mathbf{r}_{ij}}\,\overline{\mathbf{r}_{ij}} + B_c\left(r_{ij}\right)\left(\mathbf{I} - \overline{\mathbf{r}_{ij}}\,\overline{\mathbf{r}_{ij}}\right)\right] \tag{14}$$

where the mobility functions (s=self ; c = cross) are

$$A_s(r_{ij}) = -\frac{15}{4}\left(\frac{a}{r_{ij}}\right)^4 + \frac{11}{2}\left(\frac{a}{r_{ij}}\right)^6 + O\left(\left(\frac{a}{r_{ij}}\right)^8\right) \tag{15}$$

$$A_c(r_{ij}) = \frac{3}{2}\left(\frac{a}{r_{ij}}\right) - \left(\frac{a}{r_{ij}}\right)^3 + \frac{75}{4}\left(\frac{a}{r_{ij}}\right)^7 + O\left(\left(\frac{a}{r_{ij}}\right)^9\right) \tag{16}$$

$$B_s(r_{ij}) = -\frac{17}{16}\left(\frac{a}{r_{ij}}\right)^6 + O\left(\left(\frac{a}{r_{ij}}\right)^8\right) \tag{17}$$

$$B_c(r_{ij}) = \frac{3}{4}\left(\frac{a}{r_{ij}}\right) + \frac{1}{2}\left(\frac{a}{r_{ij}}\right)^3 + O\left(\left(\frac{a}{r_{ij}}\right)^9\right) \tag{18}$$

3 Simulation Details

3.1 Simulation Model

In order to incorporate hydrodynamic interaction and screened Coulomb interaction between colloidal particles, Ermak-McCammon scheme is used to solve the Smoluchowski equation, describing the motion of each particle. Smoluchowski equation for N particle system is obtained by substituting the Langevin equation for N particles system in the diffusive time scale, in the Liouville equation. Liouville equation is the conservation equation for overall probability and says that its substantial derivative should always be zero. In the diffusive time scale, net force on each colloidal particle becomes zero, so the velocity of the particle as a function of the forces acting on it, is substituted in the Liouville equation.

Generalized Smoluchowski equation for the N particle system is

$$\frac{\partial}{\partial t} P(\mathbf{X},t) = \overline{O}\left(P(\mathbf{X},t)\right) \tag{19}$$

where
$$\overline{O}(...) = \sum_{i,j=1}^{N} \left[\nabla_{r_i} \cdot \mathbf{D}_{ij}(\mathbf{X}) \cdot \left(\beta(...) \nabla_{r_j} \Phi(\mathbf{X}) + \nabla_{r_j}(...) \right) \right] \tag{20}$$

$$\mathbf{X} = \left(\mathbf{r}_1(t), \mathbf{r}_2(t), ... \mathbf{r}_N(t)\right) \quad ; \quad \beta = \frac{1}{k_B T}$$

$\mathbf{D}_{ij}$ is a 3×3 part of 3N×3N sized $\mathbf{D}$ matrix i.e. the overall translational hydrodynamic diffusivity tensor. This can be re-written in terms of particle drift velocity as

$$\frac{\partial P(\mathbf{X},t)}{\partial t} = -\sum_{i=1}^{N} \nabla_i \cdot \left(v_i^D(\mathbf{X}) P(\mathbf{X},t) \right) + \sum_{i,j=1}^{N} \nabla_i \cdot \left[\nabla_j \cdot \left(\mathbf{D}_{ij}(\mathbf{X}) P(\mathbf{X},t) \right) \right] \tag{21}$$

where, drift velocity
$$v_i^D(\mathbf{X}) = \sum_{j=1}^{N} \left[\beta \mathbf{D}_{ij}(\mathbf{X}) \mathbf{F}_j(\mathbf{X}) + \nabla_j \cdot \mathbf{D}_{ij}(\mathbf{X}) \right] \tag{22}$$

$\mathbf{F}_j(\mathbf{X}) = [\mathbf{F}_{j1}, \mathbf{F}_{j2} \ldots \mathbf{F}_{jN}]$ is the 3N×1 sized force vector on particle j due to all particles.

Here, the divergence of $\mathbf{D}$ is the extra term which is repulsive and sends particles into the configurations where they have higher mobility. This term is zero if Rotne-Prager approximation to the diffusivity tensor is used.

Smoluchowski equation is applicable in the case when the momenta of all colloidal particles have been relaxed but their positions have not changed significantly. This time scale is Brownian/Diffusive time scale which is lesser than configurational relaxation time scale. The advantage of this situation is that the implicitness in numerical scheme for calculating particle positions can be killed. This can be done by taking the time step very less than configurational relaxation time scale but large than momentum relaxation time scale and assuming that during this time step, configurations have changed so little that $\mathbf{D}(\mathrm{X}) \approx \mathbf{D}(\mathrm{X}_0)$ and $v_i^D(\mathbf{X}) = v_i^D(\mathbf{X}_0)$.

Taking time step as τ, equation (21) can be simplified as

$$\frac{\partial}{\partial t} P(\mathbf{X}, \tau \mid \mathbf{X}_0) = -\boldsymbol{v}^D(\mathbf{X}_0) \cdot \nabla P(\mathbf{X}, \tau \mid \mathbf{X}_0) + \nabla \cdot \left[\nabla \cdot \mathbf{D}(\mathbf{X}_0) P(\mathbf{X}, \tau \mid \mathbf{X}_0) \right] \tag{23}$$

If $P(\mathbf{X}, 0 \mid \mathbf{X}_0) = \delta(\mathbf{X} - \mathbf{X}_0)$, then the solution to equation (23) is

$$P^*(\mathbf{X}, \tau \mid \mathbf{X}_0) = \frac{(4\pi\tau)^{-3N/2}}{\sqrt{\det(\mathbf{D}(\mathbf{X}_0))}} \exp\left(- \frac{\left(\left[\mathbf{X} - \mathbf{X}_0 - \tau \boldsymbol{v}^D(\mathbf{X}_0) \right]^T \cdot \mathbf{D}^{-1}(\mathbf{X}_0) \cdot \left[\mathbf{X} - \mathbf{X}_0 - \tau \boldsymbol{v}^D(\mathbf{X}_0) \right] \right)}{4\tau} \right) \tag{24}$$

This is a multivariate Gaussian distribution.

The truncation error between above expression for the probability distribution function and the true solution is $\mathrm{O}(\tau)$.

$$P(\mathbf{X}, \tau \mid \mathbf{X}_0) = P^*(\mathbf{X}, \tau \mid \mathbf{X}_0) + O(\tau) \tag{25}$$

From equation (24), we get following relations.

$$\left\langle \left(\mathbf{X} - \mathbf{X}_0 \right) \right\rangle = \boldsymbol{v}^D(\mathbf{X}_0)\tau + O(\tau) \tag{26}$$

$$\left\langle \left(\mathbf{X} - \mathbf{X}_0 \right)^2 \right\rangle = 2\mathbf{D}(\mathbf{X}_0)\tau + O(\tau) \tag{27}$$

3.2 Simulation Algorithm

Following is the Ermak-McCammon algorithm.

Run following steps for all particles and at each time step

$$r_i(t_0+\tau) = r_i(t_0) + v_i^D(\mathbf{X}_0)\tau + (\sqrt{2\tau})\mathbf{d}(\mathbf{X}_0)\cdot\mathbf{n} \tag{28}$$

$$v_i^D(\mathbf{X}_0) = \sum_{j=1}^{N}\left[\beta\mathbf{D}_{ij}(\mathbf{X}_0)\mathbf{F}_j(\mathbf{X}_0) + \nabla_j\cdot\mathbf{D}_{ij}(\mathbf{X}_0)\right] \tag{29}$$

Here, **d(X)** is the square root matrix of the positive definite matrix **D**. This can be done by Cholesky decomposition of **D**.

$$\mathbf{D} = \mathbf{d}\cdot\mathbf{d}^T$$

n is 3N×1 sized vector containing elements as random numbers having mean zero and variance 1. (Gaussian distribution)

3.3 Simulation Parameters

Number of nanoparticles = 20

Time step = 1 ps

Maximum time = 20 ns

Radius of nanoparticles = 10 nm

Length of simulation box = 1000 nm

Coulomb screening length = 1 nm

Temperature = 300 K

Viscosity of surrounding medium = 10^{-4} Pa.s

4 Results

Following is the result showing the average mean square displacement of particles with respect to time. The time step used is 1 picosecond and the simulation is carried out till 20 nanoseconds.

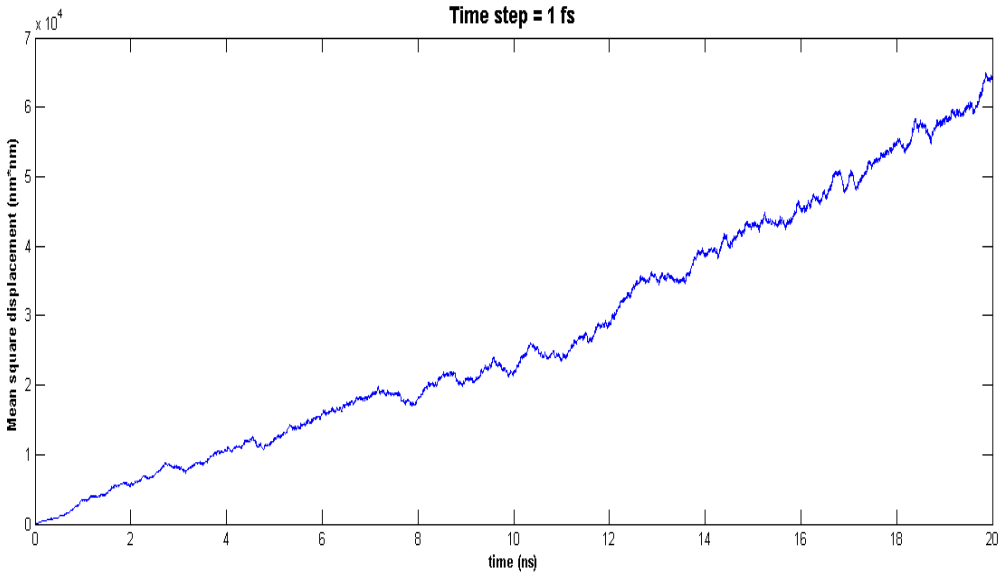

Figure 1: Average mean square displacement vs. Time

It can be seen by observing the plot that the system has passed the ballistic regime where mean displacement is proportional to time. The plot can be approximately interpreted as linear, which confirms the diffusive regime.

Suspension diffusivity is calculated from the above plot and is found to be **3.1972×10^{-6} m^2/s at 300 K.**

The calculated value of suspension diffusion coefficient is significantly larger than its average value. This might be primarily due to strong repulsive electrostatic forces and instantaneous hydrodynamic interactions.

Stress due to direct interparticle forces in the system can be calculated to first order as

$$\mathbf{T} = \frac{1}{V}\sum_{j=1}^{N} \boldsymbol{f_j r_j} \tag{30}$$

where f_j is the total direct force on j^{th} particle. V is the volume of simulation domain.

Using this formulation, the stress contribution due to screened Coulombic forces in the system is calculated to be $\mathbf{3.8859\times10^{-9}\ kg/ms^2}$.This value corresponds to the system specifications where each colloidal particle is singly positively charged and the Debye screening length is 1 nm. Since the system is very dilute, and electrostatic interactions are screened, the magnitudes of the net electrostatic force on each colloidal particle is very small, which might be the reason for very low value of the stress.

5 Conclusion and Discussion

The effective motion of colloidal particles in a quiescent solvent and in electrostatic and hydrodynamic force fields is simulated. The simulation time scale is in the diffusive range. Rotne –Prager correction to the Oseen approximation of the microscopic diffusive tensor brings the system closer to the physically realizable case, since it considers the effect of first order reflected fluid flow field. More such corrections can be added in the tensor using the method of reflections, to get better physical picture.

Overall diffusivity matrix D can be split as far field part and near field part. The matrix which we have derived and used in simulation is the far field part. Near field part also consists of lubrication effects. Since we are having very dilute system of particles, ($\phi=8.3776\times10^{-5}$) and the particles repel each other, the average distance between the particles is significantly larger than the particle diameter. So we have neglected near field effects.

6 Future Work

Current system has very low volume fraction of colloidal particles. One of the future plans will be to simulate dense systems. To achieve that, hydrodynamic interactions will be modeled as many-body interactions and since they are long range forces, they will be calculated in Fourier space. Screened electrostatic interactions will be calculated in the same fashion using Ewald summation. Lubrication effects will be included.

Another important modification to be done in current work is to introduce a velocity gradient in the background fluid. This will help to evaluate dependence of suspension viscosity on the shear rate. It will

be possible to calculate the contribution to the total stress in the suspension by Brownian and non-Brownian forces on particles.

7 References

1. Ermak, Donald L., and J. A. McCammon. "Brownian dynamics with hydrodynamic interactions." The Journal of chemical physics 69.4 (1978): 1352-1360.
2. Dhont, Jan KG. An introduction to dynamics of colloids. Elsevier, 1996.
3. Rotne, Jens, and Stephen Prager. "Variational treatment of hydrodynamic interaction in polymers." The Journal of Chemical Physics 50.11 (1969): 4831-4837.
4. Hoda, Nazish, and Satish Kumar. "Brownian dynamics simulations of polyelectrolyte adsorption in shear flow with hydrodynamic interaction." The Journal of chemical physics 127.23 (2007): 234902.
5. AP_Philips Notes on Brownian Motion, Van 't Hoff Laboratory, Utrecht University
6. Batchelor, G. K. "Brownian diffusion of particles with hydrodynamic interaction."Journal of Fluid Mechanics 74.01 (1976): 1-29.
7. Veer, J. M. On Brownian dynamics simulations of concentrated dispersions. [Sl]: Van der Veer, 1992.
8. Batchelor, G. K. "The effect of Brownian motion on the bulk stress in a suspension of spherical particles." Journal of Fluid Mechanics 83.01 (1977): 97-117.

8 Appendix (Codes)

8.1 Creating Initial Configuration

```
% Code creates positions for non-overlapping particles for a given box size,
number of particles and uniform radius of particles

function r = init_config( L, N, radius )

min_dist_sqr = ( 2*radius + 20 )^2 ;

h = zeros( N,3 );

for j = 1 : N
    x = 0.5*L*( 2 * rand -1 ) ;
    y = 0.5*L*( 2 * rand -1 ) ;
    z = 0.5*L*( 2 * rand -1 ) ;
    h(j,:) = [x y z] ;
end

for i = 1 : N-1
    for k = i+1 : N
        rxij   =  h(i,1) - h(k,1) ;
        ryij   =  h(i,2) - h(k,2) ;
        rzij   =  h(i,3) - h(k,3) ;

        rxij   =  rxij   -  round( rxij / L ) * L ;
        ryij   =  ryij   -  round( ryij / L ) * L ;
        rzij   =  rzij   -  round( rzij / L ) * L ;

        rij2   =  rxij^2  +  ryij^2  +  rzij^2 ;

        if rij2 <= min_dist_sqr ;

            h(k,:) = [0 0 0] ;

        end
    end
end

k = 1 ;
for i = 1 : N

    if h(i,2) ~= 0
        r(k,:) = h(i,:) ;
        k = k + 1 ;
    end
end

r
end
```

8.2 Simulation Parameters file

```
% Parameter file

function bd_parameters()

global N dt tmax a L T beta eta D0 lambda A

N = 20 ;           % Number of nanoparticles %
dt = 1e-2 ;        % Time step of system evolution (ns)
tmax = 20000 ;     % Number of Time marches ( time = t*time step) (ns)
a = 10 ;           % Nanoparticle diameter (nm)
L = 1000 ;         % Cell dimension (nm)
lambda = 0 ;       % Debye length ( thickness of double layer )
T = 300 ;          % Temperature ( K )
beta  = 1.0 / ( 8.314472 * 0.001 * T ) ;  % beta = 1 / kB T
eta = 1e-4 ;  % 602.3*(1e-4) ;            % Viscosity of surrounding medium
D0 = 1 / beta / 6 / pi / eta / a ;        % Characteristic diffusion
coefficient ( Stokes Einstein equation )
A = 1e5 ; % 2.3e-19;%        % Constant in the Yukawa potential : E = A*exp(-
lambda*r) / r
end
```

8.3 Main Execution File

```
% Main execution file

clear all ; close all ; clc

global dt tmax beta N L a Nn

bd_parameters ;

g = init_config( L, N, a ) ;

Nn = size(g,1) ;    % Number of particles in a system

% r = zeros( 3*N ,tmax ) ;
r = zeros( 3*Nn ,1 ) ;
n = zeros( 3*Nn ,1 ) ; % vector of random numbers having mean 0 and variance
1

for i = 1:Nn
    r(3*i-2:3*i) = g(i,:)' ;
end

r0 = r ; % Initial positions
root2t = sqrt( 2*dt ) ;
f = zeros( 3*Nn ,1 ) ;  % force on each particle

rsqr = zeros(1,tmax) ;
```

```
stress_elec = 0 ;
% Starting the run %

time = dt*(1:tmax) ;

for count = 1:tmax

    rx = g(:,1) ;
    ry = g(:,2) ;
    rz = g(:,3) ;

    D = RPY_diffusivity_tensor( rx, ry, rz ) ;

    rootD = chol( D ) ;  % Cholesky decomposition of RPY tensor to get its
square root matrix

    for i = 1:Nn

        f = -Force( i, rx, ry, rz ) ; % 3N*1 electrostatic force on the
nanoparticle due to screened Coulomb interaction

        drift_velocity = beta*D*f' ;

        for j = 1:3*Nn
            n(j) = randn ;
        end

        r   =   r   +   drift_velocity * dt   +   root2t * rootD * n ;
    end
    count

    for i = 1:Nn
        g(i,:) = r( 3*i-2 : 3*i )' ;
    end

    % Calculating the mean square displacement
    DR = (r-r0).^2 ;
    m  = sum(DR) / Nn ;
    rsqr(count) = m / 6 ;

           showbeads(g,25) ;
           drawnow
end

% Post Processing

% Calculating the stress contribution due to electrostatic forces
for p = 1:Nn
    fp = Force_total( p, g(:,1), g(:,2), g(:,3) ) ;
    stress_elec = stress_elec + fp*g(p,:)' ;
end

stress_elec = abs( stress_elec ) / (L^3)
Df = rsqr(tmax) / (dt*tmax) * (10^-9)
```

```
plot(time,rsqr)
```

8.4 Calculating Overall Diffusivity Tensor

```
% Construction of 3N by 3N far field Rotne-Prager-Yamakawa diffusivity tensor
% Near field diffusivity tensor neglected (not pairwise additive)

function D = RPY_diffusivity_tensor( rx, ry, rz )

global Nn L

for i = 2 : 3 : 3*Nn-1       % row index
    for j = 2 : 3 : 3*Nn-1   % column index

        if i==j

            p = ( j + 1 ) / 3 ;

            D( i-1 : i+1 , j-1 : j+1 ) = self_diffusivity( p, rx, ry, rz ) ;
        else

            b = ( i + 1 ) / 3 ;
            c = ( j + 1 ) / 3 ;

            rxij = rx( b ) - rx( c ) ;
            ryij = ry( b ) - ry( c ) ;
            rzij = rz( b ) - rz( c ) ;

            rxij = rxij - round( rxij / L ) * L ;
            ryij = ryij - round( ryij / L ) * L ;
            rzij = rzij - round( rzij / L ) * L ;

            rvij = [ rxij ryij rzij ] ;

            D( i-1 : i+1 , j-1 : j+1 ) = cross_diffusivity( rvij ) ;

        end
    end
end
end
```

8.5 Calculating Self-Diffusivity of the Particle

```
% Calculates the self diffusion tensor
function D = self_diffusivity( p, rx, ry, rz )

global a L Nn D0
```

```
rxi = rx(p) ;
ryi = ry(p) ;
rzi = rz(p) ;

D = 0 ;

for j = 1 : p-1

    rxij = rxi - rx( j ) ;
    ryij = ryi - ry( j ) ;
    rzij = rzi - rz( j ) ;

    rxij = rxij - round( rxij / L ) * L ;
    ryij = ryij - round( ryij / L ) * L ;
    rzij = rzij - round( rzij / L ) * L ;

    rvij = [ rxij ryij rzij ] ;

    rij  = norm( rvij ) ;

    aij = a / rij ;

    As = -3.75 *aij^4  +  5.5 *aij^6 ;

    Bs = -1.0625 *aij^6 ;

    rvij = rvij / rij ;

    rr = rvij'*rvij ;

    temp1 =  As*rr + Bs*( eye(3) - rr )  ;

    D = D + temp1 ;

end

for j = p+1 : Nn

    rxij = rxi - rx( j ) ;
    ryij = ryi - ry( j ) ;
    rzij = rzi - rz( j ) ;

    rxij = rxij - round( rxij / L ) * L ;
    ryij = ryij - round( ryij / L ) * L ;
    rzij = rzij - round( rzij / L ) * L ;

    rvij = [ rxij  ryij rzij ] ;

    rij  = norm( rvij ) ;

    aij = a / rij ;

    As = -3.75 *aij^4  +  5.5 *aij^6 ;
```

```
    Bs = -1.0625 *aij^6 ;

    rvij = rvij / rij ;

    rr = rvij'*rvij ;

    temp2 =  As*rr + Bs*( eye(3) - rr )  ;

    D = D + temp2 ;

end

D = D0*( eye(3) + D ) ; % eye(3) is 3*3 identity matrix

end
```

8.6 Calculating Cross-Diffusivity of the Pair of Particles

```
% Calculates the cross diffusion tensor

function D = cross_diffusivity( rvij )

global a D0

rij = norm( rvij ) ;

aij = a / rij ;

Ac = 1.5 *aij - aij^3  +  18.75 *aij^7 ;

Bc = 0.75 *aij + 0.5 *aij^3 ;

rvij = rvij / rij ; % unit vector of interparticle distance

rr = rvij'*rvij ;   % rr tensor (3 by 3)

D = D0* ( Ac*rr + Bc*( eye(3) - rr ) ) ; % eye(3)= 3*3 identity matrix

end
```

8.7 Force on a Particle

```
% Calculates total electrostatic force on the nanoparticle due to screened
Coulomb interaction and records it in 3N*1 vector
% Debye interaction has been modeled by Yukawa potential

function f = Force( p, rx, ry, rz )

global lambda A Nn L

rxi = rx( p ) ;
ryi = ry( p ) ;
rzi = rz( p ) ;

f( 3*p-2 : 3*p ) = [ 0 0 0 ] ;

for j = 1 : p-1

    rxij = rxi - rx( j ) ;
    ryij = ryi - ry( j ) ;
    rzij = rzi - rz( j ) ;

    rxij = rxij - round( rxij / L ) * L ;
    ryij = ryij - round( ryij / L ) * L ;
    rzij = rzij - round( rzij / L ) * L ;

    rvij = [ rxij ryij rzij ] ;

    rij  = norm( rvij ) ;

    unit_vector = rvij' / rij ;

    f( 3*j-2 : 3*j ) =  exp( -lambda * rij ) / rij * ( lambda + 1/rij ) *
unit_vector ;

end


for j = p+1 : Nn

    rxij = rxi - rx( j ) ;
    ryij = ryi - ry( j ) ;
    rzij = rzi - rz( j ) ;

    rxij = rxij - round( rxij / L ) * L ;
    ryij = ryij - round( ryij / L ) * L ;
    rzij = rzij - round( rzij / L ) * L ;

    rvij = [ rxij ryij rzij ] ;

    rij  = norm( rvij ) ;
```

```
    unit_vector = rvij / rij ;

    f( 3*j-2 : 3*j ) = exp( -lambda * rij ) / rij * ( lambda + 1/rij ) *
unit_vector ;

end

f = A * f ;

end
```

8.8 Post processing

```
function n=showbeads(list,beadsize)

% This function draws the particles, the coordinates of which are given in
"list",
% one row of x,y,z coordinates for each particle.
% Use "beadsize" to set the particle size for the plot.

[n] = size(list,1);
[a,b,c] = sphere(12);
a=beadsize/2*a;  b=beadsize/2*b;  c=beadsize/2*c;
for i=1:n
    axis([-500 500 -500 500 -500 500])

surf(a+list(i,1),b+list(i,2),c+list(i,3),'FaceColor','b','EdgeColor','None')
    hold on
end
hold off
camlight left
lighting phong
xlabel('x')
ylabel('y')
zlabel('z')
```