MODELING AND SIMULATION OF STRONGLY COUPLED PLASMAS

By

Rahnuma Rifat Chowdhury

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering- Master of Science

2016

ABSTRACT

MODELING AND SIMULATION OF STRONGLY COUPLED PLASMAS

By

Rahnuma Rifat Chowdhury

The objective of this work is to develop new modeling and simulation tools for studying strongly coupled plasmas (SCP). Strongly coupled plasmas are different from traditional plasmas as potential energy is larger than the kinetic energy. The standard plasma model does not account for some major effects in SCP:

1) the change in the permittivity
2) the impact on relaxation of the charged particles undergoing Coulomb collisions in a system with weakly shielded long range interactions
3) the impact of statistical fluctuations in strongly coupled plasmas that leads to non-Markovian effects.

Proper modeling of such systems through consideration of Lévy flight processes gives rise to fractional derivatives in time that result in an incorporation of time history in the model. A Lévy flight is a random walk in which the steps are defined in terms of the step-lengths, which have a certain probability distribution, with the directions of the steps being isotropic and random. Lévy processes in the plasma give rise to fluctuations in medium through which the electromagnetic waves are propagating. Averaging over the Lévy processes will allow us to relate to other important parameters in the plasma.

This thesis is dedicated to my parents, who have never stopped believing in me.

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Professor John Verboncoeur of the Department of Electrical and Computer Engineering at Michigan State University. He consistently steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge Professor Andrew J. Christlieb of the Department of Mathematics at Michigan State University as the co-advisor of this thesis. The door to Prof. Christlieb's office was always open whenever I ran into a trouble spot or had a question about my research or writing.

In addition, I would like to thank Assistant Professor Mohsen Zayernouri of the Department of Mechanical Engineering at Michigan State University as the committee member of this thesis. I am gratefully indebted to him for his very valuable comments on this thesis.

Moreover, I would like to express my gratitude to Michael Murillo, Gautham Dharuman, Guy Parsey and Janez Krek. Without their passionate participation and input, this work could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents and to my fiancé for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

KEY TO SYMBOLS

T_i	Ion temperature
a	Wigner-Seitz radius
Z	Degree of ionization
Γ	Coulomb coupling parameter
r	Radial distance to the particle
κ	Inverse screening parameter
m	Mass of the affected particle
λ_D	Debye length
n_e	Electron temperature
n_i	Ion density
T_e	Electron temperature
e	Electron charge
N	Number of particles
N_{eq}	Number of steps in the equilibration phase
N_{pd}	Number of steps in the production phase
dt	Time step
r_{cut}	Cut-off radius
L	Length of the simulation box
w_{pi}	Ion plasma period
m_i	Ion mass

INTRODUCTION

Lévy diffusion is a diffusion process with a non-linear relationship to time, in contrast to a typical diffusion process, in which the mean squared displacement (MSD), σ_r^2, of a particle is a linear function of time [1]. It can be described by a power law,

$$\sigma_r^2 \sim Dt^{\alpha} \quad (1)$$

Where, D is the diffusion coefficient and t is the elapsed time. In a typical diffusion process, $\alpha = 1$. If $\alpha > 1$, the phenomenon is called super-diffusion. If $\alpha < 1$, the particle undergoes sub-diffusion.

For ultra-cold plasmas, with temperatures from 1K to 1000K, it is possible to capture the two body auto-correlation function for a low temperature ion-electron plasma through the use of a highly resolved Particle-In-Cell (PIC) model. Because one of the dominant interactions in ultra-cold SCP is the Coulomb force, a resolved PIC calculation with an average of one particle per cell can simulate relaxation in ultra-cold plasmas. But with moderate increases in density or length scales, resolved PIC calculations are not practical. This has inspired us to consider the extension of classical plasma models to study ultra-cold SCP, which will then be used as a benchmark tool for the development of models to describe the physics of SCP where it is not possible to provide an ultra-resolved particle based calculation. The key objective is to develop new plasma models that treat long range correlation as a subscale feature [2].

The unique aspect of SCP is that the potential energy exceeds the kinetic energy. Strong coupling is defined in terms of the dimensionless parameter, often referred to as the Coulomb coupling parameter, Γ.

$$\Gamma=(Ze)^2/kT_i a \quad (2)$$

Eq. (2) describes the ratio of the potential energy to kinetic energy [3]. In Eq. (2):

Z = the charge number of the gas species

e = the unit charge

k = Boltzmann's constant

T_i = the temperature of the ion species in Kelvin

$a = [3/(4n)]^{1/3}$ = the Wigner-Seitz radius (mean inter-particle distance)

n = the density of the species

Γ effectively defines correlation. When $\Gamma << 1$, the charged species in the plasma has no long range correlation and binary collisions characterize Coulomb scattering for that species. $\Gamma \sim 1$, the plasma species in question begins to exhibit long range correlation. As Γ increases in these systems, the plasma exhibits a collective behavior, giving the system properties resembling liquids and solids [3, 4, 5].

Plasmas with strong coupling have been created and studied using a range of experimental methods. But there are issues with studying SCP in the context of each of these experimental setups. In many of these systems, the ability to accurately determine the initial condition as well as take accurate non-invasive measurements is difficult. Furthermore, experimental methods do not permit simultaneous measurement of detailed phase space quantities, which can be critical to understanding wave-plasma interaction, collision dynamics, and even the details of anisotropic transport.

In particle and atomic physics, a Yukawa potential (also called a screened Coulomb potential) is a potential of the form

$$U_{Yukawa}(r) = -\Gamma^2 [\exp(-\kappa mr)/r] \quad (3)$$

where Γ is the Coulomb coupling parameter which defines how strongly coupled the

system is.

As the system temperature decreases, the system becomes more and more strongly coupled. m is the mass of the affected particle, r is the radial distance to the particle, and κ is another scaling constant known as the inverse screening parameter. κ depends on Debye length, λ_D as follows:

$$\kappa = aw/\lambda_D \quad (4)$$

Debye length, λ_D is the measure of a charge carrier's net electrostatic effect in solution, and how far those electrostatic effects persist. A Debye sphere is a volume whose radius is the Debye length. With each Debye length, charges are increasingly electrically screened. With every Debye-length, the electric potential due to a given charge will decrease by 1/e. Debye length is defined as follows:

$$\lambda_D = (T_e / 4\pi n_e e^2)^{1/2} \quad (5)$$

where, $n_e = Zn_i$

n_e = electron temperature

n_i = ion density

T_e = electron temperature

Hence, from ion density and ion temperature we can derive Γ, and from electron temperature and ion density, we can derive κ.

Work Plan

We are trying to look at particular Ultra Cold (UC) experiment with our existing set of simulation tools. The main focus is to capture the impacts of raising Γ in traditional SCP. Depending on that, we are going to validate the existing models or move towards developing a Fractional Calculus Model if there is any evidence for Levy flights for moderate $\Gamma \sim 1$. The

hypothesis is that jumps in v due to collisions plus long range interactions might produce Levy flights. So, we have to execute “equilibrium” and “non-equilibrium” simulations for both collisional and non-collisional cases and do least square fitting to the tail of the differential velocity distribution.

Levy Flight is a random walk in which step lengths have a probability distribution that is heavy tailed:

$$f(x) \sim x^{-1-b},\ x \to \infty,\ 0 < b < 2 \qquad (6)$$

If distribution of mean-square velocity change $|\Delta v|^2$ has a fat tail for some effective Coupling, then Levy-flight like behavior is possible. Therefore, Levy flights might be possible in a non-equilibrium plasma system [6].

MOLECULAR DYNAMICS

Popular molecular simulation techniques such as molecular dynamics or Monte Carlo are used to study the physical and chemical processes occurring in systems containing large numbers of particles. These methods require evaluation of either the total potential energy of a system of N particles (V_{Tot}) or the gradients of the potential energy. The total potential energy consists of terms that describe the various interactions among the particles in the system. These terms are usually functions of internal coordinates, such as internuclear distances between two particles, bond angles among three particles, or torsional angles among four particles. For condensed phase modeling, the total potential energy is often described as a sum of two-body interactions over all particle pairs.

The interaction terms are typically simple functions of the internuclear distance r_{ij} between particles i and j.

$$V_{Tot} = \sum_{i=1}^{N-1} \sum_{j>1}^{N} V\,(r_{ij}) \qquad (7)$$

The evaluation of Eq. (7) and the gradients are usually the most computationally demanding steps in a simulation, even if the functional forms for $V(r_{ij})$ are extremely simple. Brute force evaluation of Eq. (7) requires the calculation of at least $N(N-1)/2$ internuclear distances. In a molecular dynamics simulation, each integration step often requires the evaluation of Eq. (7) and its gradients more than once depending on the integration scheme that is chosen [7]. It is clear that methods to reduce the computational burdens associated with numerous evaluations of Eq. (7) are required. The most obvious recent approaches are to modify the codes for scalable platforms. However, modifications of existing algorithms designed to reduce the computational burdens associated with evaluation of Eq. (7) can be made to increase the serial

performance and exploit scalable architectures to achieve enhanced performance. The algorithms that were developed can reduce potentially unnecessary computations of the internuclear distances for particle pairs used in the evaluation of Eq. (7). Common strategies to reduce the computational demands associated with Eq. (7) include the use of simple functions to describe the pair interaction potentials, and the assumption that the interaction between two particles is negligible beyond a certain cutoff distance r_{cut}. The assumption of a cutoff distance in the interaction potential allows for a reduction in computational time, since the interaction between particles separated by distances exceeding r_{cut} are not calculated. The easiest and most direct way to determine the set of internuclear distances that are within r_{cut} is to evaluate all distances between all pairs, and eliminate those that exceed r_{cut}. This step requires a potentially large number of unnecessary calculations, and might be the costliest computational step in such a simulation.

A reduction of unnecessary calculations of internuclear distances can be accomplished through the use of the Verlet neighbor list [8]. This method requires the construction of a list of neighbors for each parrticle. A particle's neighbors are usually defined to be all of the particles that are within a distance slightly greater than the range of the interaction potential. Information about the neighbors is stored in arrays. For the duration of the simulation or until the lists are updated, each particle is assumed to interact only with the particles on its neighbor list. The internuclear distances, interaction potentials, and forces are evaluated for each particle and its neighbors only. The list may be periodically updated to allow for the movement of particles into or out of the interaction range. Brute force construction or update of the list requires the evaluation of all $N(N-1)/2$ internuclear distances. The method has been shown to be efficient when the system contains a relatively small number of particles [8, 9]. However, as the system

becomes larger, the memory requirements for maintaining the neighbor lists become prohibitive.

Alternative methods for the efficient determination of the interacting neighbors for each particle include grid or cell approaches [8, 10-12]. These approaches partition the simulation space into grids or cells, to which the particles are assigned by virtue of their positions relative to the cells. Since each cell has an unchanging set of neighboring cells that contain the volume within the distance r_{cut} of that cell, a particle associated with one of the cells has as its neighbors those particles assigned to the same or neighboring cells. The implementations of these methods usually assign the particles to the cells at each integration step. However, the same considerations used for the frequency of updating the Verlet neighbor-lists are applicable here. There is some overhead associated with these methods, and they are preferable only for systems that contain more than 1000 particles [8]. These methods substantially reduce the number of unnecessary internuclear distance calculations in evaluating Eq. (1), but do not completely eliminate unnecessary computations.

THE PARTICLE IN CELL (PIC) METHOD

The particle-in-cell (PIC) method refers to a technique used to solve a certain class of partial differential equations [13-15]. In this method, individual particles in a Lagrangian frame are tracked in continuous phase space, whereas moments of the distribution such as densities and currents are computed simultaneously on stationary mesh points.

The method typically includes the following procedures:

- Integration of the equations of motion.
- Interpolation of charge and current source terms to the field mesh.
- Computation of the fields on mesh points.
- Interpolation of the fields from the mesh to the particle locations.

Models which include interactions of particles only through the average fields are called PM (particle-mesh). Those which include direct binary interactions are PP (particle-particle). Models with both types of interactions are called PP-PM or P3M.

The PIC Algorithm

i. The plasma is described by a number of computational particles having position x_p, velocity v_p and each representing a fixed number N_p of physical particles.

ii. The equations of motion for unmagnetized particles are advanced by one time step using a leap frog integrator,

$$x_p^{n+1} = x_p^n + \Delta t\, v_p^{n+1/2} \tag{8}$$

$$v_p^{n+3/2} = v_p^{n+1/2} + \Delta t\, \frac{q_s}{m_s} E_p \tag{9}$$

using the particle electric field from the previous time step.

iii. The charge densities are computed in each cell using:

$$\rho_i = \sum_p \frac{q_p}{\Delta x} W(x_i - x_p) \tag{10}$$

iv. The Poisson equation is solved:

$$\epsilon_0 \frac{\Phi_{i+1} - 2\Phi_i + \Phi_{i-1}}{\Delta x^2} = -\rho_i \tag{11}$$

and the electric field E_i in each cell is computed:

$$E_i = -\frac{\Phi_{i+1} - \Phi_{i-1}}{\Delta x} \tag{12}$$

v. From the field known in the cells, the field acting on the particles is computed as:

$$E_p = \sum_i E_i W(x_i - x_p) \tag{13}$$

which is used in the next cycle.

vi. The cycle restarts [16].

THE PARTICLE MESH (PM) METHOD

Particle Mesh (PM) is a computational method for determining the forces in a system of particles [17]. These particles could be atoms, stars, or fluid components and so the method is applicable to many fields, including molecular dynamics and astrophysics. The basic principle is that a system of particles is converted into a grid (or "mesh") of density values. The potential is then solved for this density grid, and forces are applied to each particle based on what cell it is in, and where in the cell it lies.

Various methods for converting a system of particles into a grid of densities exist. Once the density distribution is found, the potential energy of each point in the mesh can be determined from the differential form of Gauss's law, which after identifying the electric field E as the negative gradient of the electric potential Φ gives rise to a Poisson equation that is easily solved after applying the Fourier transform. Thus it is faster to do a PM calculation than to simply add up all the interactions on a particle due to all other particles for two reasons: firstly, there are usually fewer local grid points than particles, so the number of interactions to calculate is smaller, and secondly the grid technique permits the use of Fourier transform techniques to evaluate the potential, and these can be very fast.

PM is considered an obsolete method as it does not model close interaction between particles well. It has been supplanted by the Particle-Particle Particle-Mesh method, which uses a straight particle-particle sum between nearby particles in addition to the PM calculation.

The PM Algorithm

Huge model systems with 1/r potentials (celestial masses or microscopic charged particles) may be described by introducing "superparticles" consisting of some 10^4-10^8 ions, electrons, or stars.

Let us consider a square cell of MxM subcells of side length $\Delta x = \Delta y = \Delta l$. Each subcell should contain an average of 10-100 particles. The equation of motion for the particle k is:

$$\ddot{r_k} = -\frac{q_k}{m_k}\nabla_k \Phi = \frac{q_k}{m_k} E(r_k) \tag{14}$$

Where Φ_r is the electrostatic or gravitational potential. It is determined by charge (or mass) density ρ (r, t) defined by the positions of all superparticles.

To compute Φ_r at time t_n at the centers of the subcells, the given configuration of superions is first replaced by lattice-like charge distribution $\rho_{i,j}$. The easiest discretization method is the nearest grid point (NGP) rule:

$$\rho_{i,j} = \frac{1}{(\Delta l)^2}\sum_{k=1}^{N} q_k \delta\left(\frac{x_k}{\Delta l} - i\right)\, \delta\left(\frac{y_k}{\Delta l} - j\right) \tag{15}$$

The next step is the calculation of the values of the potential produced by the charge lattice at all cell centers. The field strength at the position r_k of some superparticle k in cell (i, j) is then:

$$E_x = \left[\Phi_{i+1,j} - \Phi_{i-1,j}\right]/2\Delta l \tag{16}$$

$$E_y = \left[\Phi_{i,j+1} - \Phi_{i,j-1}\right]/2\Delta l \tag{17}$$

Next we integrate the equation of motion:

$$r_k^{n+1} = 2r_k^n - r_k^{n-1} + \frac{q_k}{m_k}(\Delta t)^2 E_{i,j}^n \tag{18}$$

$$v_k^n = [\, r_k^{n+1} - r_k^{n-1}\,]/2\Delta t \tag{19}$$

which completes the time step.

Combining the PM method and the molecular dynamics technique, we may take into account the short-range forces up to a certain interparticle distance, while the long-range contributions are included by the particle-mesh procedure. This combination of particle-particle and particle-mesh methods has come to be called the PPPM or P3M technique [17].

THE PARTICLE-PARTICLE-PARTICLE MESH (P3M) METHOD

Particle–Particle-Particle–Mesh (P3M) is a Fourier-based Ewald summation method to calculate potentials in N-body simulations [18-21]. The potential could be the electrostatic potential among N point charges i.e. molecular dynamics, the gravitational potential among N gas particles in e.g. smoothed particle hydrodynamics, or any other useful function. It is based on the particle mesh method, where particles are interpolated onto a grid, and the potential is solved for this grid (e.g. by solving the discrete Poisson equation). This interpolation introduces errors in the force calculation, particularly for particles that are close together. Essentially, the particles are forced to have a lower spatial resolution during the force calculation. The P3M algorithm attempts to remedy this by calculating the potential through a direct sum for particles that are close, and through the particle mesh method for particles that are separated by some distance.

The P3M Algorithm

i. Mapping the particle x_i to the mesh

ii. Calculating

$$\nabla^2 \Phi_{mesh} = -\frac{\rho_{mesh}}{\epsilon_0} \tag{20}$$

iii. Subtracting

$$\frac{e^{r_{i_{mesh}}/\lambda_D}}{r_{i_{mesh}}} \tag{21}$$

from Φ_{mesh}

iv. Computing E_{mesh} and interpolating E_{mesh} to particles

v. Calculating the Ewald sum over particles

$$\dot{x}_i = v_i \tag{22}$$

$$\dot{v}_i = F_i = q_i E_i \tag{23}$$

Ewald Sum

Ewald summation is a method for computing long-range interactions (e.g., Coulombic interactions) in periodic systems. It was first developed as the method for calculating electrostatic energies of ionic crystals, and is now commonly used for calculating long-range interactions in computational chemistry [22]. Ewald summation is a special case of the Poisson summation formula, replacing the summation of interaction energies in real space with an equivalent summation in Fourier space. In this method, the long-range interaction is divided into two parts: a short-range contribution, and a long-range contribution which does not have a singularity. The short-range contribution is calculated in real space, whereas the long-range contribution is calculated using a Fourier transform. The advantage of this method is the rapid convergence of the energy compared with that of a direct summation. This means that the method has high accuracy and reasonable speed when computing long-range interactions, and it is thus the standard method for calculating long-range interactions in periodic systems. The method requires charge neutrality of the molecular system in order to calculate accurately the total Coulombic interaction.

METHOD OF CELL LINKED LISTS

In the conventional method of cell-linked lists, the simulation space is partitioned into cells, the edges of which are no smaller than cutoff distance of the interaction potential. The particles are then assigned to the various cells, by virtue of their position in the simulation space. A linked-list of the particle indices is created during the sorting procedure. Also, at the beginning of a simulation, an array that contains a list of cell neighbors for each cell is created. The list remains fixed unless the simulation space changes during the simulation. This is dynamically adjusted to balance the particle numbers.

A cell i_{cell} has as its neighbors any cell that contains at least one point that is within the distance r_{cut} of any point within i_{cell}. Since the conventional method requires that the edges of each cell be no smaller than r_{cut}, each cell has eight nearest neighbors (we are assuming periodic boundary conditions in all dimensions). These requirements ensure that all particles that are within the interaction range of any particle within i_{cell} are assigned to the eight nearest-neighbor cells of i_{cell} or i_{cell} itself. All particles occupying cells other than these are outside the interaction range of any particle located within i_{cell} [8].

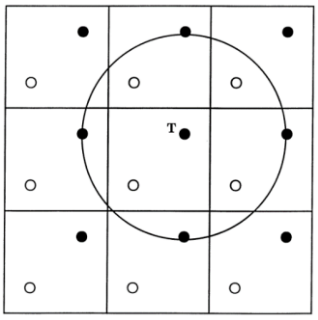

Figure 6.1 The division of a region of the simulation space into cells [8]

In Figure 6.1, both the x and y cell edges (denoted as Δx and Δy hereafter) equal r_{cut}. Evaluation of Eq. (7) occurs through looping over the cells using the linked-list of particles rather than accessing the particle indices sequentially as written in Eq. (7). This method

dramatically reduces the number of unnecessary internuclear distance calculations that would result from a brute force calculation of all $N(N-1)/2$ internuclear distances. However, modifications can be made to further reduce the number of unnecessary distance calculations. In the conventional method, the distances between all particle pairs located within the rectangular area of $9\Delta x\Delta y$ are calculated. Assuming the limiting case $\Delta x = \Delta y = r_{cut}$, the area within which all distances are calculated is $9r_{cut}^2$. The area within the cutoff radius for a single particle is only πr_{cut}^2. Thus, the traditional cell-linked list method calculates distances between all particle pairs within an area that is almost three times larger (or more, since $\Delta i \geq r_{cut}$, i = x or y) than that actually required for a particle [8].

RESULTS

Steps for Executing the Code

- Γ and κ were set to the desired values. This defines the physical parameters.
- Number of particles, N, in the input file yukawa.in was changed.
- Cut-off radius, r_{cut} was set following the mandatory relation to be satisfied: $r_{cut} < L/2$. Example: $r_{cut} = L/5, L/10$ etc. Interactions for $r > r_{cut}$ are ignored.
- According to the number of particles, the cut-off radius should be changed. The relationship between the number of particles, N and the length of the simulation box which is linked to r_{cut} is as follows:

$$n_i = N/L^3 \quad (24)$$

$$n_i a_w^3 = N/ (L')^3 \quad (25)$$

where $L' = L/ a_w$

$$L' = (4\pi N/3)^{1/3} \quad (26)$$

- Yukawa_MD.py file was run to generate the output files i.e. irp_eq.out and irp_pd.out (saves particle positions, velocities and accelerations during the equilibration and production phase respectively over the simulation time), temp_eq.out and temp_pd.out (saves the temperature of the system during the equilibration and production phase respectively), energy_eq.out and energy_pd.out (saves the total energy, kinetic energy and potential energy of the system during the equilibration and production phase respectively vs. time).
- The energy_plot.py was run to generate the plot of the total energy, kinetic energy and potential energy of the system vs. time.
- The temperature_plot.py was run to generate the plot of the temperature of the system

during the equilibration and production phase.

- The velocity_distribution.py was run to generate the distribution function of the particle velocities during the production phase.
- The diff_velocity_distribution.py was run to generate the distribution function of the particle differential velocities and also to do the least square fitting that calculates the co-efficient a and b values.
- All the distances are in the units of aw.
- Times are in units of inverse ion plasma period w_{pi}^{-1} where

$$w_{pi} = (4\pi n_i e^2/m_i)^{1/2} \tag{27}$$

- All the energies and temperatures are in units of e^2/aw.

Simulation Results for Non-Collisional Cases

The system was run for Γ=1 with time step dt=0.1 for 1000 particles.

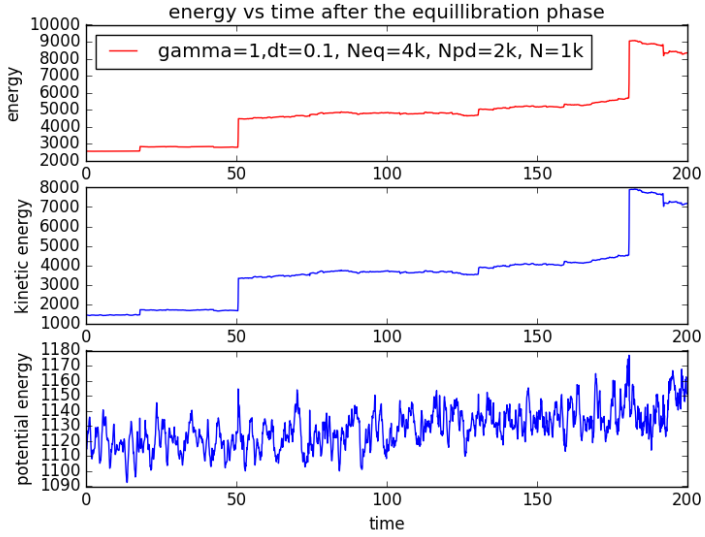

Figure 7.1 Energy after the equilibration phase for Γ=1, dt=0.1, N=1000

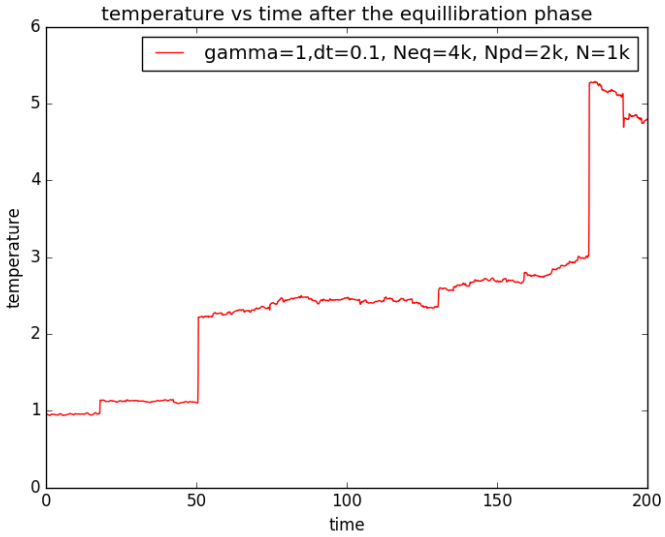

Figure 7.2 Temperature after the equilibration phase for Γ=1, dt=0.1, N=1000

We can see from Figure 7.1 and Figure 7.2 that the energy and the temperature of the system do not stabilize which means that the system has not been equilibrated. Also the kinetic energy exceeds the potential energy, which does not imply SCP.

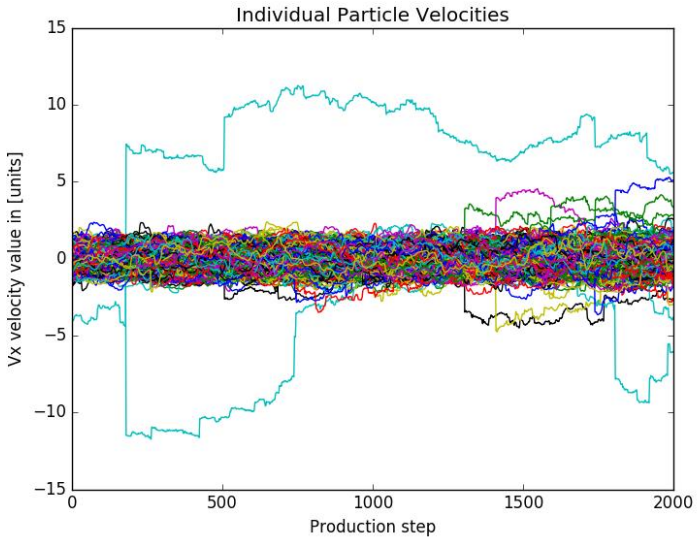

Figure 7.3 Individual particle velocities for Γ=1, dt=0.1, N=1000

Figure 7.3 represents all the particle velocities during the production phase. We can see jumps in particle velocities due to collisions and long range interactions.

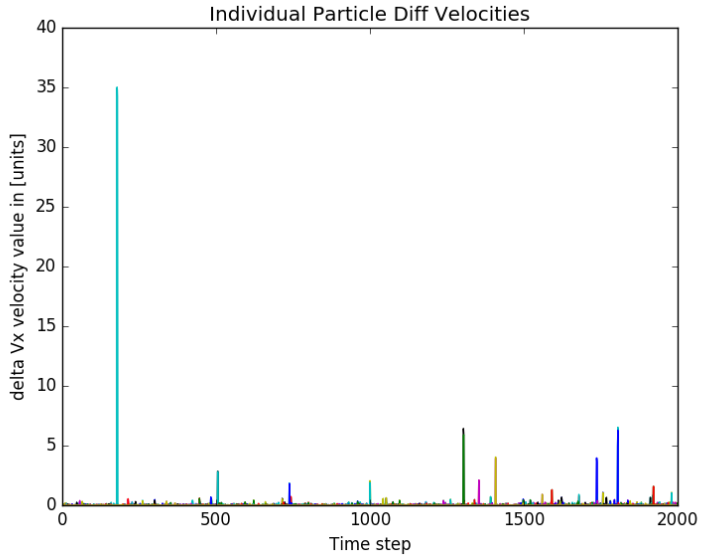

Figure 7.4 Individual particle differential velocities for Γ=1, dt=0.1, N=1000

Figure 7.4 represents all the particle differential velocities during the production phase. We can also see jumps in particle velocities due to collisions and long range interactions.

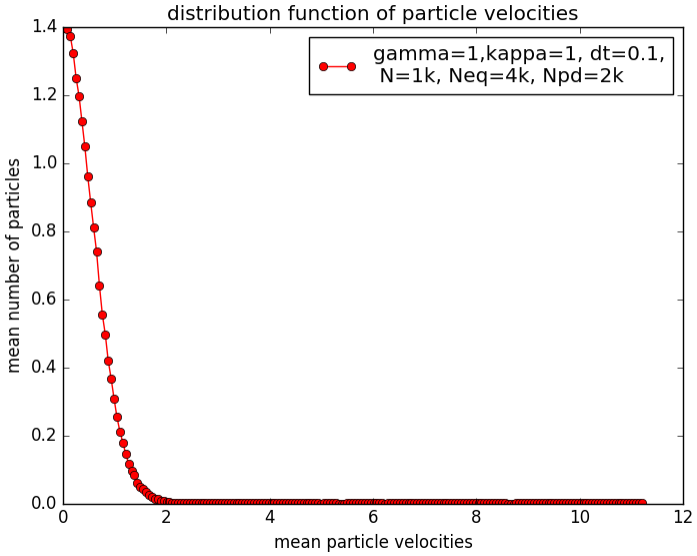

Figure 7.5 Distribution function of individual particle velocities for Γ=1, dt=0.1, N=1000

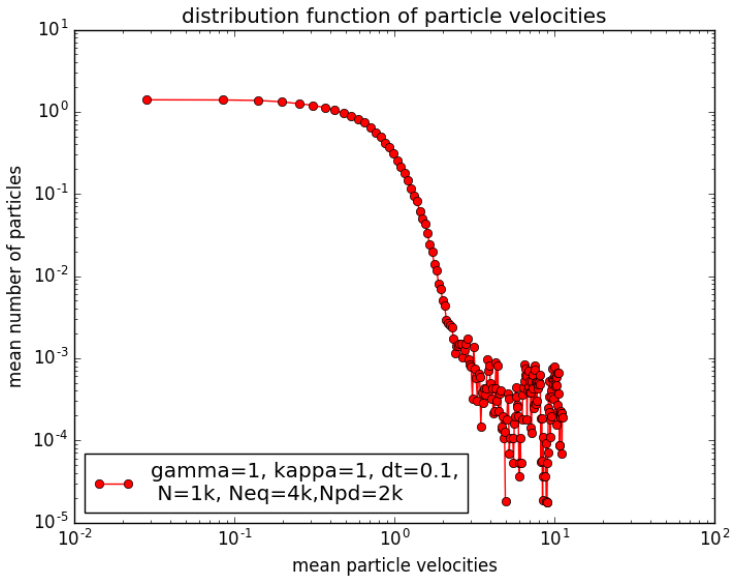

Figure 7.6 Distribution function of individual particle velocities in log-log scale for Γ=1, dt=0.1, N=1000

The distribution function was calculated by taking the average of all the particle velocities over all the time steps in the production phase. Jumps in velocities are more prominent in Figure 7.5 and Figure 7.6 which plots the distribution function of individual particle velocities during the production phase in linear scale and log-log scale respectively.

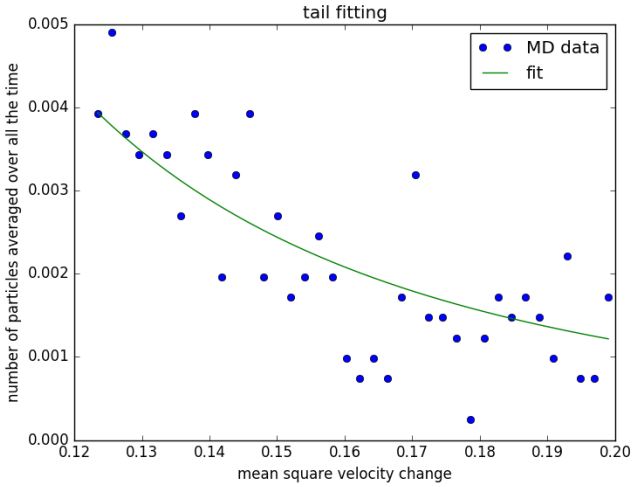

Figure 7.7 Data fitting for mean square differential velocities for Γ=1, dt=0.1, N=1000

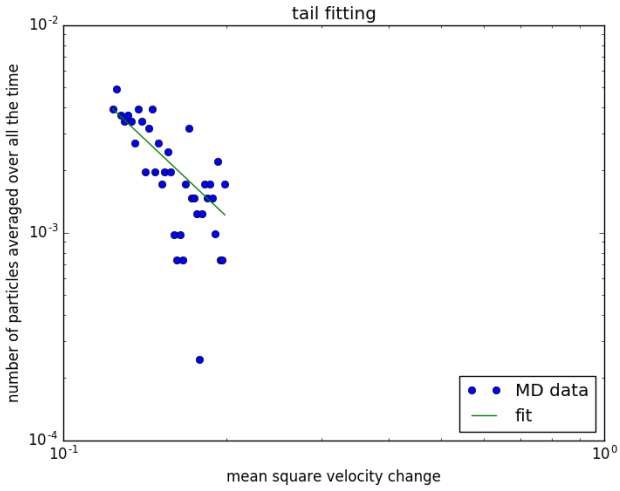

Figure 7.8 Data fitting for mean square differential velocities in log-log scale for Γ=1, dt=0.1, N=1000

The mean square velocity change was calculated by taking the average of the differential velocities of all the particles over all the time steps during the production phase. Figure 7.7 represents only the tail of the differential velocity distribution and Figure 7.8 is a log-log representation of Figure 7.7. We can see from these figures that the distribution has a fat tail. After least square fitting to the differential velocity distribution, the value of the co-efficient b comes out as 1.46 which implies Lévy type behavior.

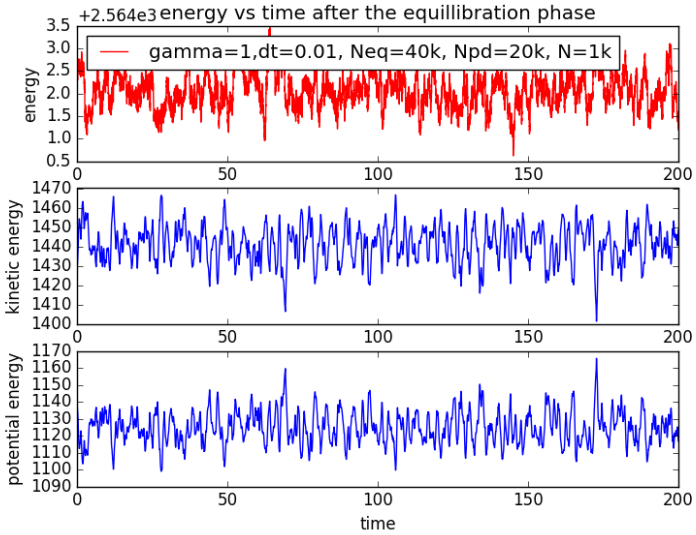

Figure 7.9 Energy after the equilibration phase for Γ=1, dt=0.01, N=1000

Next, the system was run for 1000 particles with Γ=1 but with reduced time step, dt=0.01. From Figure 7.9 and Figure 7.10, we can see that the energy and the temperature has been stabilized and the system has been equilibrated properly at much lower energy levels.

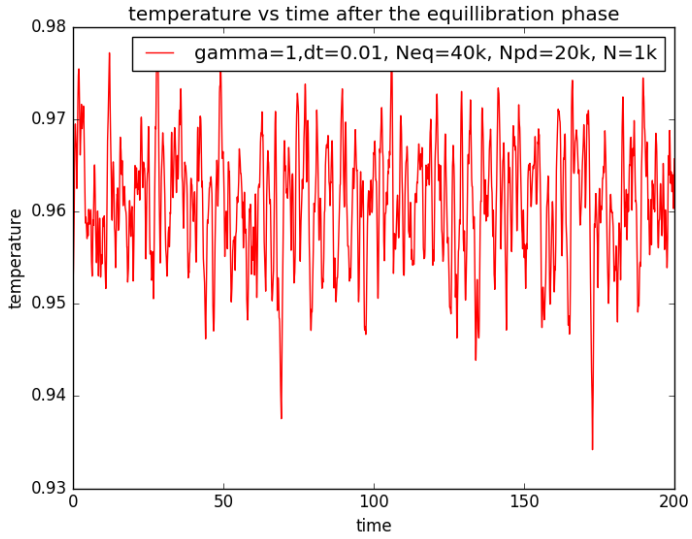

Figure 7.10 Temperature after the equilibration phase for Γ=1, dt=0.01, N=1000

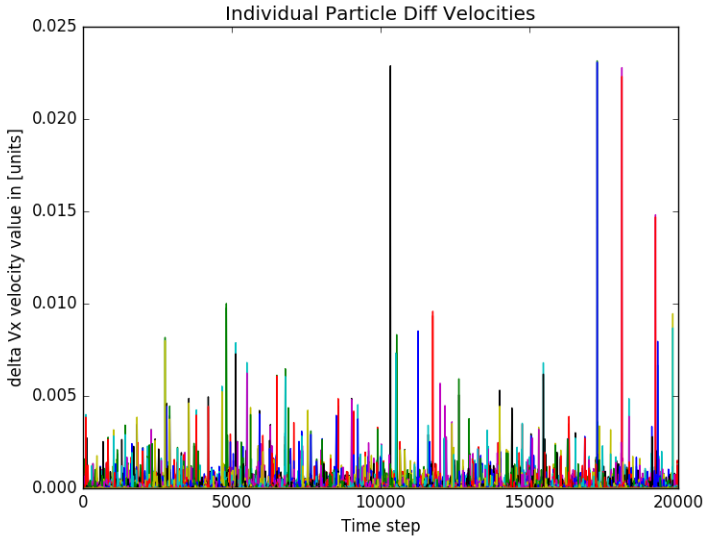

Figure 7.11 Individual particle differential velocities for Γ=1, dt=0.01, N=1000

Figure 7.11 represents all the differential particle velocities over all the time during the

production phase. No large jumps in Figure 7.11 means the system has been stabilized properly. The value of the co-efficient b after least square fitting comes out as 5.49 which implies no Lévy type behavior.

Simulation Results for Collisional Cases

The system was run for 1000 particles, Γ=1 and time step, dt=0.1 with collisions added into the system. At each time step, we took 5% of the particles and rotated the direction of the velocity by a random amount while keeping the magnitude the same.

We did this rotation of velocity V by,

$$V_x = C\sin(u)\cos(t) \quad (28)$$

$$V_y = C\sin(u)\sin(t) \quad (29)$$

$$V_z = C\cos(u) \quad (30)$$

$$\text{Where, } C = [V_{x\_old}^2 + V_{y\_old}^2 + V_{z\_old}^2]^{1/2} \quad (31)$$

We can see from Figure 7.12 and Figure 7.13 that the energy and the temperature have not been stabilized and the system has not been equilibrated properly.

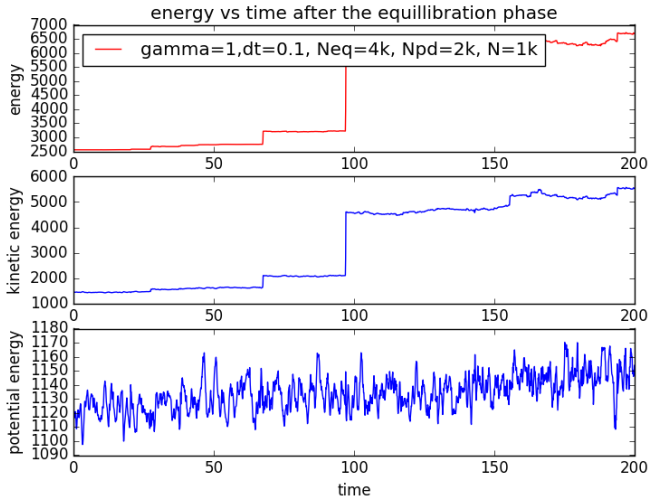

Figure 7.12 Energy after the equilibration phase for Γ=1, dt=0.1, N=1000

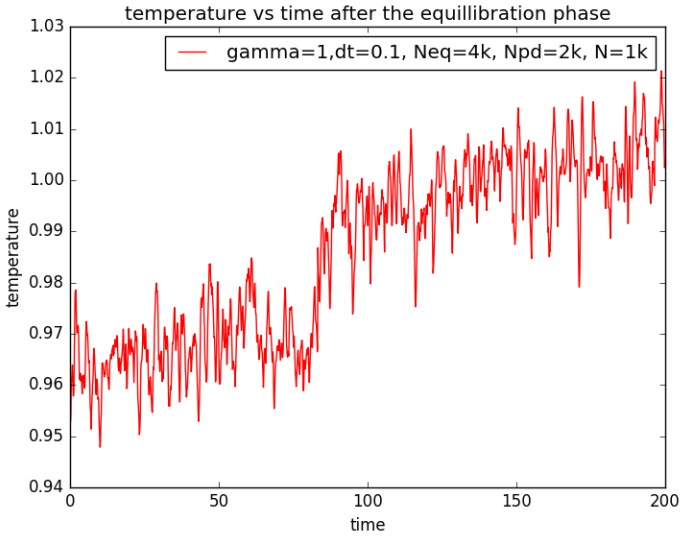

Figure 7.13 Temperature after the equilibration phase for Γ=1, dt=0.1, N=1000

After tail fitting to the differential velocity distribution, the value of the co-efficient b comes out as 0.33 which implies Lévy type behavior.

Next, the system was run for Γ=1 and time step, dt=0.01 with collisions included. We can see from Figure 7.14 and Figure 7.15 that the energy and temperature of the system have been stabilized and the system has been equilibrated properly.

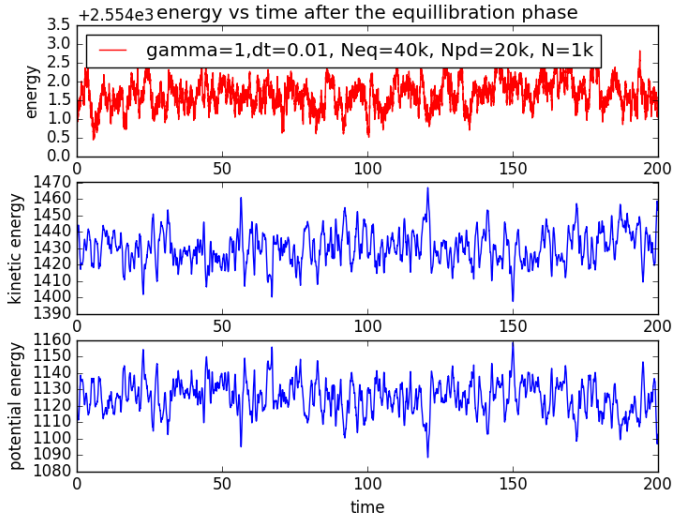

Figure 7.14 Energy after the equilibration phase for Γ=1, dt=0.01, N=1000

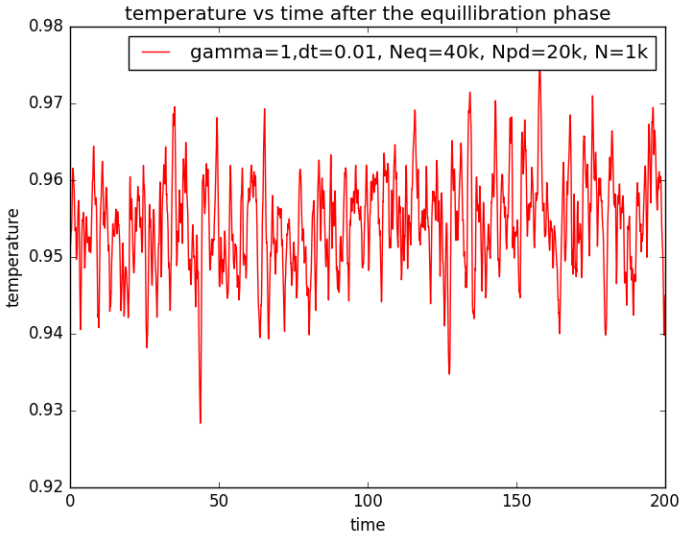

Figure 7.15 Temperature after the equilibration phase for Γ=1, dt=0.01, N=1000

The value of the co-efficient b after least square fitting comes out as 0.65 which implies Lévy type behavior.

Next, the system was run for a very long time with Γ=1 and time step, dt=0.001 with collisions included. We can see from Figure 7.16 and Figure 7.17 that the energy and temperature of the system have been stabilized and the system has been equilibrated properly.

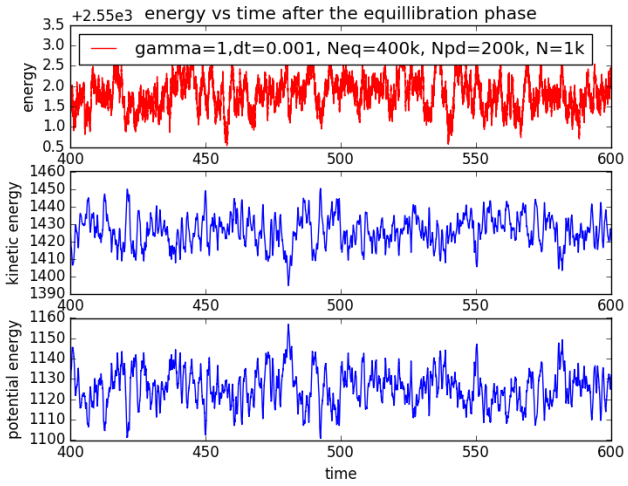

Figure 7.16 Energy after the equilibration phase for Γ=1, dt=0.001, N=1000

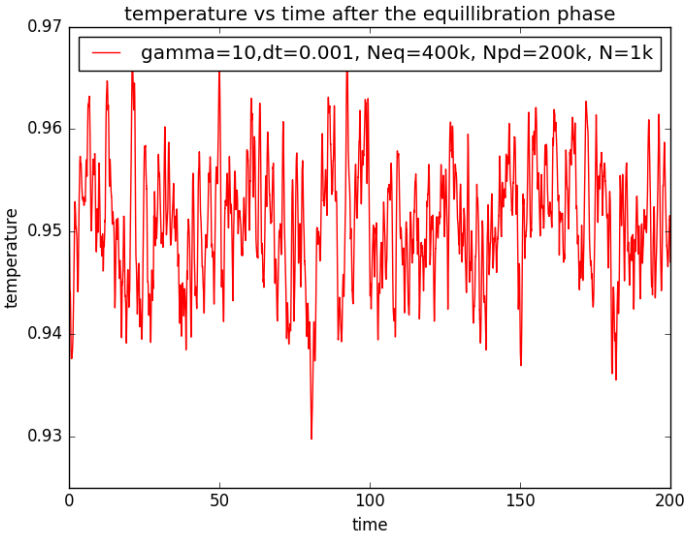

Figure 7.17 Temperature after the equilibration phase for Γ=1, dt=0.001, N=1000

The value of the co-efficient b after least square fitting comes out as 0.97 which implies Lévy type behavior.

All the simulation results have been summarized in table 7.1.

Table 7.1 Simulation Results for Both Non-Collisional (Case 1-4) and Collisional (Case 5-9) Cases

Case	N	Γ	κ	dt	Neq	Npd	r_{cut}	b	Lévy
1	1000	1	1	0.1	4000	2000	4	1.46	Yes
2	1000	1	1	0.01	4000	2000	4	2.69	No
3	1000	1	1	0.01	40000	20000	4	3.49	No
4	1000	10	1	0.1	4000	2000	4	3.96	No
5	1000	1	1	0.1	4000	2000	4	0.33	Yes
6	1000	1	1	0.01	40000	20000	4	0.65	Yes
7	1000	1	1	0.001	400000	200000	4	0.97	Yes
8	1000	10	1	0.1	4000	2000	4	1.39	Yes
9	1000	10	1	0.01	40000	20000	4	1.81	Yes

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Liu, B., Goree, J. and Feng, Y. (2008). Non-Gaussian statistics and superdiffusion in a driven-dissipative dusty plasma.

2. Christlieb, A.J., Cheng, Y., Causely, M. and Verboncoeur, J. Modeling and Simulations of Strongly Coupled Plasmas.

3. Brush, G., Sahlin, H.L. and Teller, E. (1966). Monte carlo study of a one-component plasma. The Journal of Chemical Physics 45: 2102.

4. Ichimaru, S. (1982). Strongly coupled plasmas: high-density classical plasmas and degenerate electron liquids. Reviews of Modern Physics 54 (4): 1017.

5. Morfill, G.E, Thomas, H.M, Konopka, U. and Zuzic, M. (1999) The plasma condensation: Liquid and crystalline plasmas. Physics of Plasmas 6: 1769–1780.

6. Christlieb, A.J., Choi, Y., Dharuman, G., Jain, M., Verboncoeur, J., Murrilo, M. and Roberts, J. (2016). Understanding Ultra Cold Plasmas Through Simulations [Powerpoint Slides].

7. Allen, M.P. and Tildesley, D.J. (1990). Computer Simulation of Liquids. NY: Oxford University Press.

8. Mattson, W. and Rice, B.M. (1999). Near-neighbor calculations using a modified cell-linked list method. Computer Physics Communications 119: 135-148.

9. Morales, J.J., Rull, L.F. and Toxvaerd, S. (1989). Efficiency test of the traditonal MD and the link-cell methods. Computer Physics Communication 56(2): 129-134.

10. Boris, J. (1986). A vectorized "near neighbors" algorithm of order N using a monotonic logical grid. Journal of Computational Physics 66: 1-20.

11. Lambrakos, S.G. and Boris, J.P. (1987). Geometric properties of the monotonic lagrangian grid algorithm for near neighbor calculations. Journal of Computational Physics 73: 183-202.

12. Brugé, F. (1993). Systolic calculation of pair interactions using the cell linked-lists method on multi-processor systems. Journal of Computational Physics 104: 263-266.

13. Birdsall, C. and Langdon, A. (1985). Plasma Physics via Computer Simulation. NY: McGraw-Hill.

14. Hockney, R. W. and Eastwood, J. W. (1988). Computer Simulation Using Particles. PA: Taylor and Francis.

15. Verboncoeur, J. (2005). Particle simulation of plasmas: Review and advances. Plasma Physics Controlled Fusion 47: A231–A260.

16. Lapenta, G. Mathematical Derivation of the PIC method. pp. 21. in Particle In Cell Method, A brief description of the PIC Method. Centrum voor Plasma Astrofysica, Katholieke Universiteit Leuven.

17. Vesely, F.J. "Particle-Mesh Methods (PM and P3M)." University of Vienna. http://homepage.univie.ac.at/franz.vesely/simsp/dx/node48.html.

18. Toukmaji, A.J. and Board Jr., A.J. (1996). Ewald summation techniques in perspective: a survey. Computer Physics Communications 95(2-3): 73-92.

19. Eastwood, J.W., Hockney, R.W. and Lawrence, D.N. (1984). P3M3DP-the three-dimensional periodic particle-particle/particle-mesh program. Computer Physics Communications 35: C618-C619.

20. Deserno, M. and Holm, C. (1998). How to mesh up Ewald sums. II. An accurate error estimate for the particle–particle–particle-mesh algorithm. The Journal of Chemical Physics 109: 7694.

21. Trenti, M. and Hut, P. (2008). "N-body simulations (gravitational)." Scholarpedia 3(5): 3930. http://www.scholarpedia.org/article/N-body_simulations#P3M_and_PM-Tree_codes.