High Ion Concentration Effects on Particle Charging

Andrei Robert Fendley
HIGH ION CONCENTRATION EFFECTS ON PARTICLE CHARGING

by

Andrei Fendley

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Mechanical Engineering

The University of Memphis

December 2021
ACKNOWLEDGEMENTS

I would like to thank Dr. Ranganathan Gopalakrishnan for assisting me toward obtaining a master’s degree in mechanical engineering. He guided me to learn many skills not learned from undergraduate school. I would also like to thank my lab mates Vikram Suresh and Zhibo Liu who used their time for my research. I would thank Dr. Daniel Foti, Dr. Jeffrey Marchetta, and Dr. Gladius Lewis for serving on my advisory panel. Finally, I would like to express gratitude toward family, friends. Also, I would like to thank God for blessing me for giving me this opportunity.
ABSTRACT

Developing a model for the collision rate constant between a dust grain and an ion in a high ion concentration system is essential to estimate/calculate the electric charge of a grain in a plasma. The particle is negatively charged due to higher mobility of electrons in the plasma than with ions that are positively charged. The non-dimensional collision rate constant $H$ of the dust particle and ion are calculated using Langevin Dynamics trajectory simulations. The trajectories are calculated using a first order time stepping scheme to solve the Langevin equation of motion for the ions. To account for ion neutral collisions, $H$ is calculated over a wide range of $Kn_D$, the diffusive Knudsen number, from the free molecular or low-pressure regime ($Kn_D \to \infty$) to the continuum or high-pressure regime ($Kn_D \to 0$). To account for the ion-grain electrostatic interaction parameterized by $\psi_E$, $H$ is calculated over a range of $\psi_E$ that varies between 0 to 300. To account for concentration of the ions within the domain and temperature of the gas, $H$ will be calculated over a range of $0.01 \leq \chi \leq 10$ and $0 < \Gamma_{ig} \leq 30$. This will be the first model of the dust particle ion collision rate constant that accounts for ion-ion interactionism, in addition to the ion-particle and ion-neutral gas interactions, as opposed to existing ones that account for the latter two effects only. $H$ will be compared to such previous models to quantitatively establish the effect of ion-ion interactions and motivate future experimental validation, as currently no experimental data exists to validate the model for $H$. In the future, we hope this model will be validated with experimental results. This thesis will give results for $0.01 \leq \chi \leq 1$ at $\Gamma_{ig}=1.5$. 
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>vi</td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Methods</td>
<td>9</td>
</tr>
<tr>
<td>Results and Discussion</td>
<td>14</td>
</tr>
<tr>
<td>Conclusion</td>
<td>58</td>
</tr>
<tr>
<td>References</td>
<td>59</td>
</tr>
<tr>
<td>Appendix</td>
<td>62</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1 Representation of a dusty plasma. .................................................. 1

Figure 2 Representation of an ion moving toward a particle. ..................... 2

Figure 3 Representation of the Orbital Motion Limited Theory. ............... 4

Figure 4 Representation of screening to simulate high ion concentration. ...... 7

Figure 5 Periodic boundary domain is shown where the ion will leave the upper left edge and reenter through the lower right edge. .................. 8

Figure 6-14 β_{ratio} vs Kn_D Plot for different χ,ψ_E. .......................... 17

Figure 15-44 Model Comparison for χ,ψ_E. ........................................... 28
LIST OF ABBREVIATIONS

R=charging rate

\( \beta_i \) =particle ion flux coefficient or collision kernel

\( n_p, n_i \) =particle and ion concentration

\( \psi_E \) = particle ion interaction parameter

\( z_p, z_i \) =particle and ion charge

e=electron charge

\( \varepsilon_0 \) =permittivity of free space

\( a_p \) =particle radius

\( k_b \) =Boltzmann constant

\( T_g \) =temperature of gas

\( \Gamma_{i-gas} \) =plasma coupling parameter

\( \lambda_D \) =debye length

\( \eta_p, \eta_c \) =enhancement factors

\( m_o \) =mass of ion

\( f_o \) =friction factor
INTRODUCTION

Particles/grains in dusty plasma systems are charged through collisions with electrons, ions, and other particles. Studying the collision rate is important to know the charge on the particle because the charge can be used to determine particle sizes\cite{1}. The study of dusty plasmas can vary from lunar dust\cite{2} to fusion reactor contamination\cite{3}\cite{23}. Dusty plasmas are created through ionization which can free electrons and produce ions to create a charged gas. Ionization processes can range from lasers, radioactivity, electromagnetic fields or anything that can excite the electrons\cite{29}. The physical processes in a dusty plasma include the electron and ion currents toward the particle and neutral collisions from molecules who have not lost the valence electrons. Our study will be solely on particle ion collisions in the presence of background neutrals in a high ion concentrated system $n_i \sim 10^8 - 10^{15} \text{ cm}^{-3}$.

![Figure 1. Representation of a dusty plasma.](image)

Our particle is assumed to have a negative charge from the electron flux to focus only on the particle ion collisions, so we negate any electrons in our model. The electrons are smaller
than the ions, so they move faster and reach the particle first. This causes the particle to have a negative charge at first even with ion collisions. The particle will be negative so that the electrons will be pushed away causing the electron and ion current to be equal \textsuperscript{[30]}. Past models have studied the electron flux onto the particle in a dusty plasma\textsuperscript{[4]}. Due to the dust grain having a negative charge, the coulomb force between the ions and the grain will be attractive, unlike an aerosol where the ions and particle repulse each other. This model does not include any electric field force. The particle charging rate is shown below

\[ R = \beta_i n_p n_i \]

in which \( \beta_i \) is the collision kernel also known as the particle ion flux coefficient toward the particle. \( n_p \) and \( n_i \) are the ion and particle concentrations \textsuperscript{[13]}. This charging rate can be used to calculate the charge of the grain.

\[ \Psi_E = -\frac{Z_p Z_i e^2}{4 \pi \varepsilon_0 a_p k_b T_g} \]

Figure 2. Representation of an ion moving toward a particle. Black path fluctuations represent the diffusive Brownian motion.

The interaction between the ion and particle is denoted as \( \Psi_E \).
Where $k_b T_g$ is the thermal energy of the plasma and $-\frac{z_p z_i e^2}{4\pi \varepsilon_o a_p}$ is the electrostatic energy between the ion and particle. When $\psi_E$ is high, the electrostatic energy is high which means the particle has a more negative charge or the temperature is low.

Another interaction of a dusty plasma is the coupling strength which contrasts the electrostatic potential energy of the ions and gas to the thermal energy of the gas. Strongly coupled plasmas are seen in laboratories while weakly coupled plasmas are seen in space environments. If the coupling strength is large, then the electrostatic potential energy will be larger than thermal kinetic energy. If the plasma is strongly coupled, then a plasma crystal can occur where the grains can create a lattice structure from having high electrostatic energy [5,25]. This phenomenon is also seen with ions. Coupling strength between the ion and gas is denoted by $\Gamma_{ig}$.

$$\Gamma_{i-gas} = \frac{e^2}{4\pi \varepsilon_o n_o^3 k_b T_g}$$

As stated in the introduction, the charge of the grain comes from the collision of the ion and particle. To compute the ion flux, models of the ion movement must be made. Many theoretical models study the movement of the ion, electron, and neutrals in a dusty plasma. The Orbital Motion Limited Theory [24] was used to compute the potential of the charged particle. The theory assumed the electron density was a Boltzmann distribution around the particle. The theory used the conservation of energy and angular momentum in its approach to solve the trajectory and electron and ion current. The particle has an absorption radius where the ion could either fall into it, graze it, or miss it entirely. If the ion falls into the absorption radius, the ion can
orbit till impact with the particle. Whether or not the ion will impact the particle will be determined from ion velocity, ion energy, and the absorption radius\cite{6,7}.

**Figure 3. Representation of the Orbital Motion Limited Theory.** Case 1 is where the ion misses the absorption radius. Case 2 is where the ion grazes the absorption radius. Case 3 is where the ion falls into the absorption radius.

However, the shortcoming of the orbital motion limited theory is the assumption that the ions are collision less. This means that the ion will not collide with neutrals atoms from the background gas. The trajectory of the ion will change due to the collisions with neutrals\cite{8}. A plasma in nature or manually created will have neutrals due to the atoms not losing their electrons from ionization. The next part will compare the different ways to model ion movement with neutral collisions.

Molecular dynamics is one way to simulate the ion particle collisions in the presence of neutrals. However, this process is very computationally demanding because of the simulation and calculation of each individual neutral molecule. Another way the collision rate can be calculated is through Langevin dynamics which has a lesser computational time than molecular dynamics. The neutral atoms from the background gas are not simulated individually in Langevin dynamics because the ions are assumed to have a maxwell Boltzmann velocity. One can model an aerosol particle or charged particle in a plasma using Langevin dynamics which its
basis is Newton’s second law of motion. It can be used to simulate forces which include coulomb forces, gravity, electric fields, and magnetic fields. Real life examples of aerosols/dusty plasmas that can be modeled using Langevin dynamics include particles from a sneeze or soot from a volcano\[9\]. Particle in cell simulations can use Newton’s second law to calculate the collision. Particle in cell (PIC) simulations is similar to computational fluid dynamics in the way it uses grids and boundary conditions. The particle in cell method will place the particles, ions, and electrons onto a grid. Then the computation of the electric potential will occur. The timestep, position, and velocity will be updated. However, PIC models many particles for lower error. This will lead to billions to trillions of timesteps which is very computationally expensive \[10,21\]. The next chapter will contain a more detailed process of how we used Langevin Dynamics in our model.

Back in 2011, Hogan and Gopalakrishnan studied the collision rate of an ion and particle in hard sphere case where the ion drag and Brownian diffusive motion occurs. The model was in dilute concentrated studies where the coulomb force is too small to be accounted for hence the name hard sphere motion. Hogan and Gopalakrishnan’s model of the non-dimensional collision kernel fit very well with six other transition regime’s models \[11\]. Hogan and Gopalakrishnan in 2012 calculated the collision rate of an ion and particle in dilute concentration however with a coulomb force using the Langevin equations. The conclusion was that for \(\Psi_E > 0.5\), the non-dimensional collision rate matched the flux matching theory and the Gopalakrishnan and Hogan 2011 model. However, for \(\Psi_E\) greater than 0.5 the flux matching theory was not suitable, so Gopalakrishnan and Hogan created a model for the non-dimensional collision kernel for a range of \(\Psi_E\) and diffusive Knudsen numbers. Hogan and Gopalakrishnan model did not cover high diffusive Knudsen and high \(\Psi_E\) values.
Chahl and Gopalakrishnan\textsuperscript{[13]} calculated the collision rate of an ion and particle in the presence of a neutral gas for a wide range of diffusive Knudsen numbers and $\Psi_E$ values. Similar to the previous models, Langevin dynamics was used to model the motion of the ion and particle in a cube domain. After creating the model for the non-dimensional collision kernel, Chahl and Gopalakrishnan concluded that the unscreened coulomb potential model was similar to the Gatti and Kortshagen\textsuperscript{[14]} model. Chahl and Gopalakrishnan also wanted to understand how high ion concentration would affect the collision rates. This was done by applying the screened coulomb potential/Debye- Huckel potential which is based from the Debye screening length\textsuperscript{[4]}. The Debye screening length is presented below

$$\lambda_D = \frac{\varepsilon_o k_B T_i^{\frac{1}{2}}}{n_i e^2}$$

in which $\varepsilon_o$ is the permittivity of free space, $k_B$ is the Boltzmann constant, $T_i$ is the temperature of the ions, $n_i$ is the ion concentration, and $e$ is the electron charge. In high ion concentrations, the ions will concentrate around the particle to screen the electrostatic potential or electric field from ions farther away from the particle. The shortcoming of using a screened coulomb potential with the orbital motion limited theory to model a high ion concentration is the neglect of ion-ion interactions. In chapter 3, the results will show the difference between the collision rate with vs without the ion-ion interaction.
Figure 4. Representation of screening to simulate high ion concentration. Ions crowd the particle to screen the coulomb force from ions further from the particle.

Chahl and Gopalakrishnan is a theoretical model, but it is necessary to back theoretical data with experimental results. In 2021 Suresh, Li, Felipe, and Gopalakrishnan used experimental results from the PK-4 experiments on the International Space Station for the particle charge and the results matched well with Chahl and Gopalakrishnan. In the ISS, the neon gas was ionized by lasers in the presence of an electric field without gravity. The particles were inserted into the plasma discharge and the charge was estimated through force balance equations. Suresh and Li compared these particle charge results to Chahl’s model and found consistent matching for 20 – 150 Pa. This comparison showed that the Langevin equations can be accurate, so more expensive approaches such as PIC or MD can be avoided. Now, to this date no PIC, molecular dynamics, and experimental models exist for the non-dimensional collision kernel in high ion concentrations that include ion ion interactions.

Zobnin, Usachev, Petrov, and Fortov calculated the ion current in a weakly ionized plasma. Zobnin calculated the ion current analytically by solving the Boltzmann equation for many collisionalities while including a screened coulomb potential. Gatti and Kortshagen in 2008 created an analytic model to solve for the ion current but used the orbital motion limited
theory with a very small number of neutrals. Hutchinson and Patachini \cite{21} in 2007 used a particle in cell method to calculate the ion flux onto a particle with neutral collisions. These models are for spherical particles. Previous studies have studied charging with different shape particles that can come from coagulation of dust grains. \cite{26-27} These models will be compared against the high ion concentration results in chapter 3.

Because there are no experimental or theoretical models that study the collision rates in high ion concentrations and include the ion-ion interactions, this work will be a tool and motivation to experimentally validate the results to be presented. This work will also be a tool to understand particle charging for contamination process in fusion devices where the ion concentration is high. \cite{23,28}

In methods, the model development of high ion concentration will be discussed and its difference to the Chahl and Gopalakrishnan model. In Chapter 3, the results of the collision kernel in high ion concentration will be presented and compared against screened coulomb potential models as well as the Chahl and Gopalakrishnan model. Chapter 4 will present an overview of the results obtained and conclusions.
METHODS

As stated in the introduction, our model will consist of multiple ions to resemble high ion concentration in a dusty plasma. Our simulation will be similar to Chahl and Gopalakrishnan\textsuperscript{[13]} and Gopalakrishnan and Hogan\textsuperscript{[13]} in which we use Langevin Dynamics\textsuperscript{[9]} to model the motion of the ion toward the particle. In our simulation we will account for the particle ion interaction ($\Psi_E$), coupling strength ($\Gamma_{ig}$), and the concentration ($\chi$).

To begin the simulation, the ions are placed randomly in a cube domain around a particle in the center. The number of ions chosen is 1024. Preliminary tests were done to ensure the results will not change with more ions than 1024. The ions will be reinitialized if they are initially touching or too close to the particle before the simulation occurs. The domain shape was chosen to be able to apply a periodic boundary where the ion will reenter the opposite side of the domain it left. The periodic boundary condition can be easily applied in a cube domain. However, in a spherical case the ion will be reinitialized randomly if it leaves the domain\textsuperscript{[17]}. The periodic domain effect is shown in the next figure.
Figure 5. Periodic boundary domain is shown where the ion will leave the upper left edge and reenter through the lower right edge.

The ions are then initialized with a Maxwell Boltzmann velocity,

\[ \vec{v} = k n_D \eta_f \text{NRAND}(0,1) \]
\[ \chi_i \eta_c \text{NRAND}(0,1) \]

where \( k n_D \) is the diffusive Knudsen number, \( \eta_f, \eta_c \) are the enhancement factors, \( \chi \) is the concentration factor, and \( \text{NRAND} \) is a normally distributed number with mean 0 and standard deviation 1. The ions are put randomly into the domain, but the simulation does not start yet. The ions are simulated for a period of time or a number of iterations with the interactions to obtain more of a natural position of the ions instead of having a manually inputted position.

The concentration of the simulation is denoted through

\[ \chi_i = \frac{a_p}{n_o^3} \]
where \( a_p \) is the particle radius and \( n_o^{-1} \) is the intermolecular distance. With a higher intermolecular distance, the concentration is lower. The lower the concentration, the domain size is bigger and vice versa. The \( \chi_i \) range is from 0.1 to 1 to simulate concentrations from \( 10^8 \) to \( 10^{20} \) cm\(^{-3} \). The definitions of \( \Psi_E \) and \( \Gamma_{ig} \) have been stated in the introduction. \( \Psi_E \) will range from 0 to 300 to model hard sphere motion of the ion and highly charged motion of the ion\(^{[13]} \). \( \Gamma_{ig} \) will be 1.5 to model coupling strength\(^{[4]} \). These inputs will go into our Langevin equations to calculate the motion of the ion.

As stated in the introduction, Langevin equations is used to determine the motion of the ion which comes from Newton’s second law. The simple form of Langevin

\[
m_i \frac{d\vec{v}_i}{dt} = -F_D + \vec{F}_{ext} + F_B
\]

in which \( m_i \) is the mass of the ion, \( \frac{d\vec{v}_i}{dt} \) is the acceleration of the ion, \( F_D \) is the drag force of the ion, \( \vec{F}_{ext} \) is a force applied to the ion (electrostatic, electromagnetic, gravity, electric fields, or fluid flow), and \( F_B \) is the stochastic Brownian random motion\(^{[9]} \). The Langevin equation can also be applied to covid-19 studies to reduce infection risk\(^{[19-20]} \). Ermak and Bukholtz used a first order time stepping scheme to solve the Langevin equation\(^{[9,18]} \). The basic equations for the numerical method are:

\[
\vec{v}_i(t+\Delta t) = \vec{v}_i(t) \exp \left( -\frac{f_o}{m_o} \Delta t \right) + \frac{\vec{F}_i}{f_o} \left( 1 - \exp \left( -\frac{f_o}{m_o} \Delta t \right) \right) + A_i
\]

\[
\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \frac{m_o}{f_o} \left( \vec{v}_1(t+\Delta t) + \vec{v}_1(t) - 2 \frac{\vec{F}_1}{f_o} \right) \frac{1 - \exp \left( -\frac{f_o}{m_o} \Delta t \right)}{1 + \exp \left( -\frac{f_o}{m_o} \Delta t \right)} + \frac{\vec{F}_i}{f_o} \Delta t + B_i
\]

\[
\langle A_i^2 \rangle = \frac{3k_B T}{m_o} \left( 1 - \exp \left( -\frac{2 f_o}{m_o} \Delta t \right) \right)
\]
\[ \langle B_1^2 \rangle = \frac{6m_o k_B T}{\epsilon_o f_o^2} \left( \frac{f_o}{m_o} \Delta t - 2 \left( \frac{1 - \exp \left( - \frac{f_o}{m_o} \Delta t \right)}{1 + \exp \left( - \frac{f_o}{m_o} \Delta t \right)} \right) \right) \]

\[ \vec{F}_i = \frac{e^2}{4\pi\epsilon_o} \left( z_p z_i \frac{\vec{r}_i - \vec{r}_p}{|\vec{r}_i - \vec{r}_p|^3} + \sum_{j=1, j\neq i}^{N} z_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^3} \right) \]

An initial timestep is chosen from equation

\[ dt = \text{d}_{\text{factor}} \cdot \min(dt_1, dt_2) \]

\[ dt_1 = \frac{\left( \frac{R_{\text{min}}^2 \eta_c^2}{\eta_f} \right)^2}{kn_D^2} \]

\[ dt_2 = \frac{1}{|F_{\text{max}}|} \]

where \( R_{\text{min}} \) is the minimum distance, an ion is from the particle and \( F_{\text{max}} \) is the calculated max force on an ion. \( A_1^2 \) and \( B_1^2 \) are related to the Brownian random fluctuation. \( \vec{F}_i \) is the term to account for the total force on an ion (ion-ion and particle-ion interaction).

The distance of each ion is consistently calculated to check to see if a collision happens when the distance is less than or equal to one. The collision kernel (particle ion flux coefficient) is then calculated from the collision rate as shown below:

\[ H(Kn_D, \chi, \Gamma_{ig}, \Psi_{\text{f}}) = \frac{\text{Volume}}{\text{taverage} \cdot N_{\text{ions}}} \left( \frac{\eta_5}{\eta_f} \right) \]
The particle ion flux coefficient from the high ion concentration will be graphed as a ratio of the particle ion flux coefficient from the dilute concentration\cite{12} to understand the difference and make conclusions quicker.

\[ \beta_{\text{Ratio}} = \frac{H_{\text{HIC}}}{H_{\text{Dilute}}} = \frac{H(K_{nD}, \chi, \Gamma_{ig}, \Psi_{E})}{H(K_{nD}, \Psi_{E})} \]

The screening equation considers the concentration and coupling parameters.

\[ S_D = \frac{\lambda_D}{a_p} = \left( \frac{1}{\chi_p} \right) \sqrt{\frac{1}{4\pi\Gamma_{ig}}} \]
RESULTS AND DISCUSSION

For validation, the HIC code was ran at a low χ to simulate a dilute concentration\[12\]. The results are shown in the appendix. Next, the code was ran at a zero ψ\text{E} to simulate hard sphere motion\[13\]. \(H(\text{Kn}_D, \chi, \Gamma_{ig}, \Psi_E)\) will be compared against models that include screening distance which simulates a high ion concentrated system\[14\][21][22]. Though the screening distance models do not include ion ion interactions, the comparisons can determine the difference between including ion ion interactions vs not including ion ion interactions. The results will be shown as the β\text{ratio} vs. \text{Kn}_D for different χ and ψ\text{E} values.

Below are plots of the β\text{ratio} vs. \text{Kn}_D for \(0.1 \leq \chi \leq 1\) and \(1.5 \leq \psi_E \leq 300\) at \(\Gamma_{ig}=1.5\). The general trend for these plots is that for higher ψ\text{E}, the lower the β\text{ratio}. The lower the β\text{ratio}, the more time it takes for the ion to reach the particle. This phenomenon starts to occur at \(\psi_E=7\) and goes till 300. At low \(\psi_E\) and high \(\chi\), the ion will take less time to reach the particle. Another trend seen is at high pressure or low \text{Kn}_D, the ion will take less time to reach the particle than at low pressure. This means at more neutral collisions the ion will reach the particle faster. This is a very interesting concept, and the explanation is that the ions will surround the particle due to the interactions and not be able to reach the particle. The ions further away will be screened so their interaction will be lessened. At low \(\psi_E\) and high \(\chi\), the interactions will be weak causing an ion to move more freely toward the particle. In the dilute model at high \(\chi\), the ion reached the particle more quickly. At high ion concentration, the opposite happens. This will be very important in fusion where the concentration is high due to the different species of ions. However, experiments must be done to explain this phenomenon further. The appendix has a 3x3 figure to show the trends in a concise way.
In figure 6, at $\chi=1$, the $\beta_{\text{ratio}}$ is higher signifying a lower time the ion will reach the particle. The lower the $\chi$ in figure 6, the collision rate will increase. This means at lower particle ion interaction; the particle will obtain a more positive charge at higher concentration. Next, higher particle ion electrostatic interactions will be discussed. In figure 7 at $\chi=1$, the data goes underneath lower $\chi$ below a Knudsen number of 1. This shows at a specific particle ion interaction, the collision rates trends are different at different pressures. At high pressure at high $\chi$ the beta ratio is higher. However, at lower pressure the beta ratio is lower than the low chi. The transition is due to the ions screening the particle and the ions are interacting more with each other. This phenomenon is important in any application where high ion occurs because the charge on the particle is greatly influenced. Again, in figure 8, at high $\chi$, the transition occurs but the transition occurs at even higher pressure. One interesting occurrence is at high Knudsen or low pressure and at $\chi$ from 0.5 to 1, the data lines up. There is a steady state where increasing the concentration will not change the time the ion reaches the particle.

In figure 9 -11, the trend continues as discussed before. The beta ratio goes lower at higher $\chi$ and $\varphi_{E}$. This means the screening and ion ion interactions are larger causing the ion to reach the particle in more time than the 2019 Chahl model.

Figure 12 shows at lower Knudsen number, the different $\chi$ line up with each other at a beta ratio of 1. The particle will have the same collision rate regardless of concentration of the ions at those pressure values. Figures 13 and 14 show this to be the same as well. Increasing the particle ion interaction will not change the collision kernel for high pressure. Because the charging rate has the ion concentration term, the charge will be different regardless of the beta ratio, but the collision rates will be the same. In figure 14, at $\chi=1$, the beta ratio is much lower than the dilute model. As stated before, the screening of the particle, ion- ion interactions, and
lattice formation of the ions cause the ion to reach the particle in more time in a high concentration. For model development more $\Gamma_{ig}$ values will be run to understand lattice formation effects on the collision kernel.
Figure 6. $\beta_{\text{ratio}}$ vs $Kn_D$ Plot for different $\chi$, $\Psi_E=1.5$. 
Figure 7. $\beta_{\text{ratio}}$ vs $Kn_D$ Plot for different $\chi, \psi_E = 7$. 
Figure 8. $\beta_{\text{ratio}}$ vs $\text{Kn}_D$ Plot for different $\chi$, $\Psi_E = 10$. 
Figure 9. $\beta$ ratio vs $Kn_D$ Plot for different $\chi, \Psi_E = 20$. 
Figure 10. $\beta_{\text{ratio}}$ vs $Kn_D$ Plot for different $\chi, \Psi_E = 30$. 
Figure 11. $\beta_{\text{ratio}}$ vs KnD Plot for different $\chi, \Psi_E = 60$. 
Figure 12. $\beta_{\text{ratio}}$ vs $Kn_D$ Plot for different $\chi, \psi_E = 150$. 

\[ \Psi_E = 150 \quad \Gamma_{ig} = 1.5 \]
Figure 13. β_ratio vs KnD Plot for different χ, ψ_E=200.
Figure 14. $\beta$ ratio vs $Kn_D$ Plot for different $\chi, \psi_E=300$. 

$\Psi_E=300 \Gamma_{ig}=1.5$
Next, we have comparisons between the high ion concentration results with previous models which are explained in the introduction of the paper. These models give us the particle ion flux coefficient. The particle ion flux coefficients are presented as a ratio over the dilute particle ion flux coefficient just like the previous results. Hutchinson, Zobnin, and Gatti lines are the particle ion flux coefficient models that include screening over the dilute Chahl model. The comparison with the HIC results and these models show the difference of including the ion ion interactions.

In figure 15, the HIC data lines up well at beta ratio of 1 to signify running high ion concentration cases but at dilute give similar results to the dilute mode. The hard sphere model also lines up well because the $\psi_E$ is small. In figure 16 and 17 no model that includes screening is similar to the HIC results. This means that at low $\psi_E$ screening effects are not relevant. In most cases, the HIC results do not converge to the continuum limit. The continuum limit is the particle ion flux coefficient but with the ion drag only involved. At the continuum limit, the neutrals are infinite so the only force will be coming from the ion drag. At the free molecular limit, the neutral molecules are nonexistent so there are no collisions. The ion’s trajectory only comes from its velocity. When the HIC does not converge to these limits, another limit perhaps that account for multiple ions must be used.

In figure 18 to 20, at high $\chi$ and high Knudsen number, the HIC results are like the screening models. However, the differences come from the ion ion interactions present in the HIC model. In figure 21, at high Knudsen the slope of the HIC and screening models are similar in shape but HIC data is $\sim 40\%$ more. However, at low Knudsen number, the screening models are the same as the HIC data. In this case, the HIC converges to the continuum limit. In figure 22, the shape becomes more of an S – curve. Zobnin has gaps in the data because of the log log
scale. Again, the curves are similar, but the differences with the HIC are noticeable. This effect is seen with figure 24-35. The % difference between the Gatti model and HIC results range from ~0 to 60% difference. The inconsistency between the screening models and HIC results show that screening models are not valid for modeling high ion concentration with ion ion interactions.

Starting at figure 36, $\psi_E$ is large, the HIC results line up even better with the Gatti and Hutchinson model than other $\psi_E$. Even though the line shapes have the same trend, the data is still very different. At larger $\psi_E$, no other model can agree well for all Knudsen numbers. As stated above, the conclusion is that the no model is sufficient to model HIC due to the importance of ion ion interactions. This also proves the importance of the ion ion interactions because ion ion interactions take place in real life dusty plasmas. This is a big step in understanding particle charge in high ion concentration.
Figure 15. Model Comparison \( \chi=0.1, \psi_E=1.5 \).
Figure 16. Model Comparison $\chi=0.5, \psi_E=1.5$. 
\[ \Psi_e = 1.5 \, \chi = 1 \, \Gamma_{ig} = 1.5 \, S_D = 0.23 \]

Figure 17. Model Comparison \( \chi = 1, \Psi_e = 1.5 \).
Figure 18. Model Comparison $\chi=0.1, \psi_E=3$. 
Figure 19. Model Comparison $\chi=0.5, \psi_E=3$. 
Figure 20. Model Comparison $\chi=1, \Psi_E=3$. 
Figure 21. Model Comparison $\chi=0.1, \psi_E=7$. 
Figure 22. Model Comparison $\chi=0.5, \psi_E=7$. 
Figure 23. Model Comparison $\chi=1, \psi_E=7$. 

$\Psi_E = 7 \quad \chi = 1 \quad \Gamma_{ig} = 1.5 \quad S_D = 0.23$
$\Psi_e = 10 \; \chi = 0.1 \; \Gamma_{ig} = 1.5 \; S_D = 2.3$

Figure 24. Model Comparison $\chi = 0.1, \Psi_e = 10$. 
\( \Psi_E = 10, \chi = 0.5, \Gamma_0 = 1.5, S_p = 0.46 \)

![Graph showing model comparison with \( \chi = 0.5, \Psi_E = 10 \).](image)

**Figure 25.** Model Comparison \( \chi = 0.5, \Psi_E = 10 \).
Figure 26. Model Comparison $\chi=1, \psi_E=10$. 
Figure 27. Model Comparison $\chi=0.1, \Psi_E=20$. 
Figure 28. Model Comparison $\chi=0.5, \psi_E=20$. 

\[ \psi_E=20 \quad \chi=0.5 \quad \Gamma_{ig}=1.5 \quad S_D=0.46 \]
Figure 29. Model Comparison $\chi=1, \Psi_E=20$. 

<table>
<thead>
<tr>
<th></th>
<th>HIC</th>
<th>Hutchinson</th>
<th>Zobnin</th>
<th>Gatti</th>
<th>Hard sphere</th>
<th>Continuum</th>
<th>Free Molecule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of $H_{Dilute}$</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
</tbody>
</table>
\[ \Psi_E = 30 \quad \chi = 0.1 \quad \Gamma_{ig} = 1.5 \quad S_D = 2.3 \]

Figure 30. Model Comparison $\chi=0.1,\Psi_E=30$. 
Figure 31. Model Comparison $\chi=0.5, \Psi_E=30$. 

\[ \Psi_E=30 \quad \chi=0.5 \quad \Gamma_{ig}=1.5 \quad S_D=0.46 \]
Figure 32. Model Comparison $\chi=1, \psi_E=30$. 

$\Psi_E=30 \ \chi=1 \ \Gamma_{ig}=1.5 \ \Sigma_D=0.23$
Figure 33. Model Comparison $\chi=0.1, \psi_E=60$. 

$\Psi_E=60 \chi=0.1 \Gamma_{ig}=1.5 S_D=2.3$
Figure 34. Model Comparison $\chi=0.5, \Psi_E=60$. 

![Graph showing model comparison with various ratio curves and labels for different models such as HIC, Hutchinson, Zobnin, Gatti, Hard sphere, Continuum, and Free Molecular. The graph plots $\beta$ Ratio against $\text{Kn}_D$ on a log-log scale.](image-url)
Figure 35. Model Comparison $\chi=1, \psi_E=60$. 
Figure 36. Model Comparison $\chi=0.1, \Psi_E=150$. 

\begin{equation*}
\Psi_E=150 \chi=0.1 \Gamma_{ig}=1.5 S_D=2.3
\end{equation*}
Figure 37. Model Comparison $\chi=0.5, \Psi_E = 150$. 

\[ \Psi_E = 150, \chi = 0.5, \Gamma_{ig} = 1.5, S_D = 0.46 \]
Figure 38. Model Comparison $\chi=1, \psi_E=150$. 

Ratio of $H_{\text{Dilute}}$
Figure 39. Model Comparison $\chi=0.1, \psi_E=200$. 
Figure 40. Model Comparison $\chi=0.5, \Psi_E=200$. 
Figure 41. Model Comparison $\chi=1, \psi_E=200$. 

$\psi_E = 200 \; \chi = 1 \; \Gamma_{ig} = 1.5 \; S_D = 0.23$
Figure 42. Model Comparison $\chi=0.1, \Psi_E=300$. 

$\Psi_E=300, \chi=0.1, \Gamma_{ig}=1.5, S_D=2.3$
Figure 43. Model Comparison $\chi=0.5, \Psi_E=300$. 

\[
\Psi_E=300, \chi=0.5, \Gamma_{ig}=1.5, S_D=0.46
\]
Figure 44. Model Comparison $\chi = 1, \Psi_E = 300$. 

$\Psi_E = 300 \; \chi = 1 \; \Gamma_i = 1.5 \; S_D = 0.23$
CONCLUSION

In conclusion, the high ion concentration results for $\Gamma_{\text{ig}}=1.5$ was compared against previous models that included screening. The conclusion is that no dilute model follows the HIC results perfectly due to the ion ion interactions which makes ion ion interactions very important. The other conclusion is the higher concentration at low $\psi_E$ makes the particle less negative due to ions hitting the particle faster. At higher $\psi_E$ and at higher concentration, the particle is more negative due to the ions hitting the particle less quickly.

Shortcomings of this work include that the particle charge usually fluctuates and can change the particle ion flux coefficient. This model kept the charge constant. In a dusty plasma experiment, the grains will be numerous causing the particle particle interactions which can change the collision kernels of the ion. This model did not include those interactions. Another shortcoming was that experimental work does not exist to validate high ion concentration collision kernels. However, this work can be a motivation for that.

The next step in this research is to run the higher $\Gamma_{\text{ig}}$ cases to create a Langevin dynamic based model for the changing parameters. Knowing the charge on a particle in a dusty plasma is difficult and time consuming, so there needs to be a high ion concentration model with variables that can be translated to physical dimensions. Researchers can then be encouraged to validate the model through experiments. This thesis has shown the importance of ion ion interactions in the plasma science field.
REFERENCES


APPENDIX

1. $\beta_{\text{ratio}}$ vs Kn P Plot 3x3

| $\frac{\phi\eta}{\nu} | \frac{\gamma}{\nu}$ | $\frac{g}{\nu}$ |
|-----------------|-------------|
| 1.5             | 1.5         |
| 1.5             | 2.0         |
| 1.5             | 3.0         |
| 1.5             | 5.0         |
| 1.5             | 50.0        |
| 1.5             | 150.0       |
| 1.5             | 300.0       |
| 1.5             | 1,000.0     |

$\lambda$

- 1
- 0.8
- 0.5
- 0.2
- 0.1

Legend:
- □
- ▲
- ▼
- ◇
2. Verification HIC Code with Hard sphere model

\[
\psi E = 0; \chi_i = 0.01
\]
3. MATLAB Code for creating scripts and files for HPC

```matlab
%% FILE WRITER HIC CODE
clear;
% open original file
% fid=fopen('Dusty_Plasma_3D.f');
fid=fopen('HIC_PARA_SEP7.f');
st=166; %start

%% NOTE : PSIE to be changed in the Fortran code
%% Make sure to enter 'cd pathname' in the sh generator section of this code
%% Number of ions simulated should be divisible by Ncpus

% variable1
VO1 = [1.5]; % gamma
s1 = size(VO1,2);
% VO1 = [0.1]; % gamma1
% VO1 = [2]; % gamma2
% VO1 = [25]; % gamma3
% VO1 = [100]; % gamma4
V1=num_conv(VO1);%format convert

% variable2
VO2 = [0.1 0.2]; % khi
s2 = size(VO2,2);
V2=num_conv(VO2);%format convert

% variable3 % KnD DILUTE
VO3 = [0.01 0.02 0.05 0.08 0.1 0.2 0.5 0.8 1 2 5 8 10 20 50 80 100 200 500 800 1000 2000]; % low KnD
s3 = size(VO3,2);
% VO3 = [100 500 1000]; % high KnD
% VO3 = [1000];
% V3=(num_conv(VO2.*VO3));%format convert
V3=num_conv(VO3);%format convert

% copy and paste original lines here
originalstr1='PARAMETER (GM_IG = 5.00D00)';
originalstr2='PARAMETER (KHI = 1.00D00)';
originalstr3='PARAMETER (KND_D = 1.00D-02)';

% change file names
flname='Fortran%d.f';
a=size(V1);
b=size(V2);
c=size(V3);

% put change-to lines here
```

tostr1=cell(a(2),0);
for i1=1:a(2)
    tostr1{i1}=sprintf('PARAMETER (GM_IG = %s'),V1{i1});
end
tostr2=cell(a(2),0);
for i2=1:b(2)
    tostr2{i2}=sprintf('PARAMETER (KHI = %s'),V2{i2});
end
tostr3=cell(a(2),0);
for i3=1:c(2)
    tostr3{i3}=sprintf('PARAMETER (KND_D = %s'),V3{i3});
end

%getting original text
linenumber=1;
str=[];
while ~feof(fid)
    tline = fgetl(fid);
    str{linenumber}=tline;
    linenumber=linenumber+1;
end
fclose('all');

%replace text
k=1;
for i1=1:a(2)
    strn=strrep(str,originalstr1,tostr1{i1});
    for i2=1:b(2)
        if i2==1
            strn=strrep(strn,originalstr2,tostr2{i2});
        else
            strn=strrep(strn,tostr2{i2-1},tostr2{i2});
        end
        for i3=1:c(2)
            if i3==1&&i2==1
                strn=strrep(strn,originalstr3,tostr3{i3});
            elseif i3==1&&i2~==1
                strn=strrep(strn,tostr3{i(2)},tostr3{i3});
            else
                strn=strrep(strn,tostr3{i3-1},tostr3{i3});
            end
            newname=sprintf(flname,fst+k);
            k=k+1;
            fid = fopen(newname,'w+');
            fprintf(fid,'%s
',strn{iprint});
        end
    end
end
fclose all;
end
end
end

%% sh generator
N_cpu = 4;
days = 15;  \% \% Set 'No of Days' Time limit
hrs = 5;
rem_mb=512;  \% \% Memory per node
path = '/home/afendley/HIC/3/';
Ncodes = s1*s2*s3;
k=1;
startfrom = fst;
Head='Fortran';
for i1=1:Ncodes
  \%sh generation
  sh_gen(startfrom+k,N_cpu,path,days,hrs,rem_mb);
k=k+1;
end

%command
batchs='for i in {3001..3002}; do mpiifort -o job${i}.exe HyD${i}.f; sleep 1; done';
batchs=strrep(batchs,'HyD',Head);
batchs=strrep(batchs,'3001',num2str(startfrom+1));
batchs=strrep(batchs,'3002',num2str(startfrom+k-1));
submits='for i in {3001..3002}; do sbatch A${i}.sh; sleep 0.1; done';
submits=strrep(submits,'3001',num2str(startfrom+1));
submits=strrep(submits,'3002',num2str(startfrom+k-1));
disp('module load intel/2019.5');
disp(path);
disp(batchs);
disp(submits);
fprintf('n');

function sh_gen(k,N_cpu,path,days,hrs,rem_mb)
  mystr{1}='#!/bin/bash';
temp=strrep('#SBATCH --ntasks=8',8,num2str(N_cpu));
  mystr{2}=temp;
  mystr{3}='#SBATCH --partition=computeq';
  mystr{4}=['#SBATCH --time=',num2str(days),'-',num2str(hrs),':00','00'];
  mystr{5}=['#SBATCH --mem-per-cpu=',num2str(rem_mb),',mb'];
  mystr{6}=['#SBATCH --job-name=job',num2str(k)];
  mystr{7}='';
  mystr{8}='module load intel/2019.5';
  mystr{9}='';
  mystr{10}='';
myst{11}=path;
myst{12}="";
myst{13}="";
myst{14}=['mpirun -np $SLURM_NTASKS ./job',num2str(k),'.exe'];
myst{15}="";
myst{16}="";
fname=sprintf('A%i.sh',k);
fsh=fopen(fname,'w+');
    for iprt=1:length(myst)
        fprintf(fsh,'%s
',myst{iprt});
    end
fclose(fsh);
end

% function of convert
function F=num_conv(MT)
    F=cell(length(MT),0);
    MTN=cell(length(MT),0);
    for i=1:length(MT)
        K=sprintf('%.2E',MT(i));
        MTN(i)=K;
        MTN(i)=strrep(MTN(i),'E','D');
        MTN(i)=strrep(MTN(i),'+','');
        F(i)=MTN{i};
    end
end
4. MATLAB Code for extracting files from hpc

```matlab
Ni = 1024;
PsiE = 300;
if PsiE == 0;
    etaf = 1;
etac = 1;
else
    etaf = 1+PsiE;
etac = PsiE/(1-exp(-PsiE));
end
factor = etac/((etaf^2));

fnamepath = "C:\Users\afendley\Downloads\HIC_EXTRACT\300";
[theAnswer] = readdata1(fnamepath);

%% Set pck = 1 if you have only one coll_inf file for each case
pck = 1; %% No split = 1, with split = 2
splitn = 5;

boxfac = theAnswer(:,3);
boxvol = boxfac.^3;
KnD = theAnswer(:,4);
SD = 1e8;
H_dil = zeros(length(KnD),1);
for i = 1:length(KnD)
    H_dil(i) = ComputeHmodel(PsiE,KnD(i),SD);
end
cd(fnamepath);
filename_before = dir('COLL-INF*.DAT');
k = length(filename_before);
names = extractfield(filename_before,'name');
extr = zeros(k,2);
if pck == 1
    mu = zeros(k,1);
    H_inst = [];
    for a = 1:k
        fname = names{a};
        Dats = importdata(fname);
        Collisions(a,:) = length(Dats);
count(a,1) = size(Dats,1);
```
if isempty(Dats)
    extr(a,1) = 0;
    extr(a,2) = 0;
else
    if length(Dats) <= 3
        extr(a,1) = Dats(1); %% Collision rate avg
        extr(a,2) = size(Dats,1);
    else
        extr(a,1) = Dats(length(Dats),1);
        extr(a,2) = length(Dats);
    end
    %             plot(Dats(:,1))
    %             pause(0.5)
    end
    % extr(a,2) = Dats(length(Dats),2); %% CPU time
    % extr(a,3) = Dats(length(Dats),3); %% Instantaneous TC
    % TC = boxvol(a)*(extr(a,1))./Dats(:,1);
    % d_TC = diff(TC); %% time for each collision
    % H_inst = boxvol./d_TC; %%
    % params = gevfit(log(H_inst));
    % % mu(a) = params(3);
    % figure(1)
    % plot(Dats(:,1),Dats(:,3))
    % hold on
end

H_sim = factr.*extr(:,1);
H_ratio = H_sim./H_dil;
myanswer = [theAnswer(:,[1,2,3,4]),H_sim,H_dil,H_ratio,Collisions];
theAnswer = [theAnswer, H_dil,H_sim,H_ratio,extr(:,2)];
else
    mu = zeros(k/splitn,1);
    H_inst = [];
for a = 1:k
    fname = name{a};
    Dats = importdata(fname);
    extr(a,1) = Dats(length(Dats),1); %% Collision count
    extr(a,2) = Dats(length(Dats),2); %% H_HIC
    extr(a,3) = Dats(length(Dats),3); %% Simulation time
    TC = boxvol*(extr(a,1))./Dats(:,1);
    d_TC = diff(TC); %% time for each collision
    if mod(a,splitn)==0
        H_inst = [H_inst ; boxvol./d_TC]; %% append H instantaneous from each file
        params = gevfit(log(H_inst));
        mu(a) = params(3);
        H_inst = [];
    end
end
function [B]=readdata1(fpathname)
cd (fpathname)
filename_before = dir('MB_DE*.DAT');
k = length(filename_before);
B = zeros(k,15);
name = extractfield(filename_before,'name');

for a = 1:k
    fname = name{a};
    A=textread(fname,'%s');
    format long;
    if length(A) < 35
        B(a,1)=sscanf(A{4},'%f');
        B(a,2)=sscanf(A{12},'%f');
        B(a,3)=sscanf(A{19},'%f');
        B(a,4)=sscanf(A{21},'%f');
    else
        B(a,1)=sscanf(A{4},'%f');
        B(a,2)=sscanf(A{12},'%f');
        B(a,3)=sscanf(A{19},'%f');
        B(a,4)=sscanf(A{21},'%f');
    end
end

%%% Compute H_Fit
function H_model = ComputeHmodel(PsiE, KnD, SD)

%%% H - Hard sphere
H_HS = (4*pi*KnD^2 + 25.836*KnD^3 + sqrt(8*pi)*KnD*(11.211*KnD^3))/(1+3.502*KnD + 7.211*KnD^2 + 11.211*KnD^3);

%%%
% Constants from C
\[ c_1 = 32.64; \]
\[ c_2 = 0.1362; \]
\[ c_3 = -33.2; \]

\[
\text{if } SD < 10 \\
\quad \gamma = 0; \\
\text{elseif } SD > 1000 \\
\quad \gamma = 1; \\
\text{else} \\
\quad c_4 = -2.4; \\
\quad c_5 = 0.9898*(\Psi E^{-0.164}); \\
\quad \gamma = 1 + c_4/(SD^{c_5}); \\
\text{end}
\]

% Compute A
\[
\text{if } SD < 10 \\
\quad \alpha = 0; \\
\text{elseif } SD > 1000 \\
\quad \alpha = 1; \\
\text{else} \\
\quad a_1 = -0.003998*\Psi E - 0.3929; \\
\quad a_2 = 0.001431*\Psi E + 0.1791; \\
\quad \alpha = 1 + a_1/(SD^{a_2}); \\
\text{end}
\]

% CONSTANTS FROM B
\[ b_1 = 1.76; \]
\[ b_2 = 4.956; \]
\[ b_3 = 0.6109; \]
\[ b_4 = 2.925; \]

\[
\text{if } SD < 10 \\
\quad \beta = 0; \\
\text{elseif } SD > 1000 \\
\quad \beta = 1; \\
\text{else} \\
\quad b_5 = (3.23e-5) *(\Psi E^2) - 0.008786*\Psi E - 1.557; \\
\quad b_6 = -0.0006008*\Psi E + 0.5654; \\
\quad \beta = 1 + b_5/(SD^{b_6}); \\
\text{end}
\]

% k constants
\[ k_1 = -0.03425; \]
\[ k_2 = 0.4494; \]
\[ k_3 = 0.0864; \]

% Compute A,B,C,k
A = 2.8*alpha;
$$B = (b_1 \exp(-b_2 \Psi E) + b_3 \log(1+b_4 \Psi E)) \beta;$$
$$C = (c_1 \Psi E^c_2 + c_3) \gamma;$$
$$k = k_1 \Psi E^k_2 + k_3;$$
$$\mu = \frac{(C/A)((1+k(\log(KnD)-B)/A)^(-1/k))}{\exp(((1+k(\log(KnD)-B)/A)^(-1/k))}(\exp((-1/k)));$$
$$H_{\text{model}} = \exp(\mu) \cdot H_{\text{HS}};$$
end
5. Fortran Code for computing collision kernel in high ion concentration

!!! MULTIBODY CODE - high ion concentration charging
!!! parallel code - mpi-based
PROGRAM HIC_FIXED_PARTICLE
IMPLICIT NONE
!!! Parallel code
INCLUDE 'mpif.h' !USE MPI

! INPUTS: GAMMA_ION-Gas, KND_CONC, PSIE, KHI, N_IONS (512)
! Inputs are with respect to gas temperature
! Gamma_ion-ion is inferred from simulation

INTEGER I,MJ,J1,ICPU
INTEGER IERR,RANK,NPROCS
INTEGER ITER,ITERMIN_1,ITERMIN_2,ITERMAX
INTEGER TR,TR1,TR2,TR3,TR5,TR6,
CTR8,TR9,TR10,TR11,TR13,TR14,TR15,TR16 ! FOR FILE OPERATIONS
INTEGER COL_CHK ! CHECKFLAG FOR COLLISION
INTEGER JMP,JMPL ! JUMPS, NO. OF COLLISIONS LIMIT
INTEGER N_TOTAL ! TOTAL NO OF PARTICLES
INTEGER TPL,SKP,SKP_2,SKP_3 ! TRAJECTORY PRINT LOOP
INTEGER LPS, TPLIM ! FOR WRITING TO FILE
INTEGER SEED
INTEGER F1SIZE,F2SIZE,F_LIM,FLINE
INTEGER NBINS,GSTEP,GSTEPMAX,N_BINS
INTEGER F_TR ! FILE TRIAL NUMBER
INTEGER I1_TRI,I2_TRI,I1_REC,I2_REC,SCT_SZ ! PARA LOOP INDEX
INTEGER, DIMENSION(8) :: VALUES ! STORES DATE AND TIME INFO
INTEGER FEND_CHECK ! CHECK IF OUTPUT FILES HAVE BEEN FILLED COMPLETELY

DOUBLE PRECISION KHI ! ap/ni(-1/3) RATIO
DOUBLE PRECISION LX, LY, LZ, PI
DOUBLE PRECISION R_MIN,SI
DOUBLE PRECISION GM_IIG ! Gamma_ion-gas, Particle charge
DOUBLE PRECISION F_MAX,RHOD
DOUBLE PRECISION MSDND,MSVND,FAC1,FAC2,FAC3,FAC4 ! FOR LANGEVIN
DOUBLE PRECISION DT,TC,DFACTOR,TC_J ! TC_J = FOR INSTANTANEOUS RATE
DOUBLE PRECISION RATE_NDIM,MEAN_TIME
DOUBLE PRECISION H_MEAN,H_HS,C1,C2,C3,C4
DOUBLE PRECISION TSTART,TFINISH,TDIFF
DOUBLE PRECISION GI_AVG
DOUBLE PRECISION BOX_F,TEMP1 ! BOX FACTOR
DOUBLE PRECISION RANGE1,RANGE2,BWID
DOUBLE PRECISION RANGE_1,RANGE_2,B_WID
DOUBLE PRECISION KND ! KND CONCENTRATED
DOUBLE PRECISION PSIE,KND_D,ETA_F,ETA_C ! DILUTE
DOUBLE PRECISION WD_1,WD_2,AVGSQ,GM_II,KEAVG

INTEGER, ALLOCATABLE :: INDEX_TRI(:),INDEX_REC(:) !LOOP SEGMENTATION INDEX
DOUBLE PRECISION, ALLOCATABLE :: X(:,),Y(:,),Z(:,),TEMP(:,)!position
DOUBLE PRECISION, ALLOCATABLE :: X_SC(:,),Y_SC(:,),Z_SC(:,)!position SCATTERED
DOUBLE PRECISION, ALLOCATABLE :: UC(:,),VC(:,),WC(:,)!velocities
DOUBLE PRECISION, ALLOCATABLE :: UN(:,),VN(:,),WN(:,)

73
DOUBLE PRECISION, ALLOCATABLE :: UC_SC(:,), VC_SC(:,), WC_SC(:)
DOUBLE PRECISION, ALLOCATABLE :: UN_SC(:,), VN_SC(:,), WN_SC(:)
DOUBLE PRECISION, ALLOCATABLE :: T_REC(:, :) ! record collision of each ion
DOUBLE PRECISION, ALLOCATABLE, DIMENSION(:) :: FX, FY, FZ ! forces
DOUBLE PRECISION, ALLOCATABLE :: FX_SC(:,), FY_SC(:,), FZ_SC(:)
DOUBLE PRECISION, ALLOCATABLE :: TEMP_DBIN(:,), TEMP_BIN_SUM(:)
DOUBLE PRECISION, ALLOCATABLE :: TEMPDBIN(:,), TEMPBINSUM(:)
DOUBLE PRECISION, ALLOCATABLE :: RBIN(:,), R_BIN(:)
DOUBLE PRECISION, ALLOCATABLE, DIMENSION(:) :: BINCNT, BCNT_PI

CHARACTER(LEN=300):: FILE01, FILE02, FILE03, FILE05, FILE06 ! output files
CHARACTER(LEN=300):: FILE08, FILE09, FILE10, FILE11, FILE13
CHARACTER(LEN=300):: FILE14, FILE15, FILE16
CHARACTER(LEN=8):: GMIC
CHARACTER(LEN=10):: KNDC
CHARACTER(LEN=6):: NTC
CHARACTER(LEN=4):: FTRC
CHARACTER(LEN=6):: DFACC
CHARACTER(LEN=10):: PSIEC
CHARACTER(LEN=6):: KHIC

!!! INPUT PARAMETERS
PARAMETER (PI = 4.D0*DATAN(1.D0))
PARAMETER (GM_IG = 1.50D0) ! Gamma value
PARAMETER (DFACOR = 1D-2) ! dfac
!PARAMETER (Jmplim = 15000)
PARAMETER (N_TOTAL = 1024) ! NUMBER OF IONS
PARAMETER (KHI = 8.00D-01) ! (ap / ni^(-1/3))
PARAMETER (KND_D = 5.00D-01) ! knd dilute
PARAMETER (PSIE = 3.0D2) ! PSIE DILUTE
PARAMETER (F_TR = 1)

!INITIATE MPI
CALL MPI_INIT(IERR)
CALL MPI_COMM_SIZE(MPI_COMM_WORLD, NPROCS, IERR)
CALL MPI_COMM_RANK(MPI_COMM_WORLD, RANK, IERR)
ICPU=RANK
SCT_SZ=INT(N_TOTAL/NPROCS) ! SCATTER SIZE

Jmplim = 2000
ETA_F = 1D0+PSIE
ETA_C = PSIE/(1D0-EXP(-PSIE))

IF (PSIE .EQ. 0D0) THEN
  ETA_F = 1.0D0
  ETA_C = 1.0D0
ENDIF

KND = KND_D*KHI ! KnD_conc -> sqrt(mkT) / (ni^-1/3 * fi)

Box and screening parameters
BOX_F = (DBLE(N_TOTAL**((1.0D0/3.0D0))))/KHI
RHOD = DBLE(N_TOTAL+1)/(BOX_F**3D0)
LX = BOX_F
LY = BOX_F
LZ = BOX_F

!!! ALLOCATE
ALLOCATE(X(1:N_TOTAL),Y(1:N_TOTAL),Z(1:N_TOTAL))
ALLOCATE(UC(1:N_TOTAL),VC(1:N_TOTAL),WC(1:N_TOTAL))
ALLOCATE(UN(1:N_TOTAL),VN(1:N_TOTAL),WN(1:N_TOTAL))
ALLOCATE(FX(1:N_TOTAL))
ALLOCATE(FY(1:N_TOTAL))
ALLOCATE(FZ(1:N_TOTAL))
ALLOCATE(T_REC(1:N_TOTAL,1:2))
ALLOCATE(INDEX_TRI(1:NPROCS+1))
ALLOCATE(INDEX_REC(1:NPROCS+1))
ALLOCATE(TEMP(1:N_TOTAL))
ALLOCATE(X_SC(1:SCT_SZ))
ALLOCATE(Y_SC(1:SCT_SZ))
ALLOCATE(Z_SC(1:SCT_SZ))
ALLOCATE(UC_SC(1:SCT_SZ))
ALLOCATE(VC_SC(1:SCT_SZ))
ALLOCATE(WC_SC(1:SCT_SZ))
ALLOCATE(UN_SC(1:SCT_SZ))
ALLOCATE(VN_SC(1:SCT_SZ))
ALLOCATE(WN_SC(1:SCT_SZ))
ALLOCATE(FX_SC(1:SCT_SZ))
ALLOCATE(FY_SC(1:SCT_SZ))
ALLOCATE(FZ_SC(1:SCT_SZ))

!!! FILE SIZE AND STEPS
F_LIM = 30000000 ! LIMIT FILE TO CERTAIN KBYTES SIZE
TPL = 1 ! FOR FILE OPERATIONS
FEND_CHECK = 0
IF (KND_D .LT. 1D1) THEN
  ITERMIN_1 = 1000
  SKP = 50 ! PRINT TRAJ AFTER EVERY "SKP" LOOPS
  SKP_2 = 500 ! G(R) CALCS
  SKP_3 = 500 ! skip for GAMMA calc.
ELSEIF (KND_D .LT. 1D2) THEN
  ITERMIN_1 = 50000
  SKP = 1000 ! PRINT TRAJ AFTER EVERY "SKP" LOOPS
  SKP_2 = 2000 ! G(R) CALCS
  SKP_3 = 2000 ! GAMMA calc.
ELSE
  ITERMIN_1 = 100000
  SKP = 2000 ! PRINT TRAJ AFTER EVERY "SKP" LOOPS
  SKP_2 = 4000 ! G(R) CALCS
  SKP_3 = 4000 ! GAMMA calc.
ENDIF
GSTEPMAX = 2000 ! FOR G(R) NO,OF.SETS OF RUNS
ITER = 1
TC = 0.0D0 ! NON-D SIMULATION TIME
TC_J = 0.0D0
WD_1 = 2D2 ! Number of bins for ion-ion G(R)
WD_2 = 2D2
ELSE
    WD_2 = 4D2
ENDIF

!!! FILE INITIALIZATIONS
IF (ICPU .EQ. 0) THEN
    WRITE (GMIC, '(I8.8)') INT(ABS(GM_IG)*1D5)
    WRITE (KNDC, '(I10.10)') INT(KND_D*1D5)
    WRITE (NTC, '(I6.6)') INT(N_TOTAL)
    WRITE (FTRC, '(I4.4)') INT(F_TR)
    WRITE (DFACC, '(I6.6)') INT(DFACTOR*1D3)
    WRITE (PSIEC, '(I10.10)') INT(ABS(PSIE)*1D4)
    WRITE (KHIC, '(I6.6)') INT(ABS(KHI)*1D3)
    FILE01='MB_DETAILS_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE02='TRAJ1_GMIG'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE05='H_DETAILS_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE06='VEL1_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE08='COLL_INF_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE09='RAD_DIST_BIN1_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE10='RAD_DIST_BIN2_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE13='HIC_GMI_II_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE14='TRAJ2_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
    FILE15='VEL2_GMI'//GMIC//'_'//KNDC//'_'//KHI//'_'//KHIC
        C//'_NI'//'NC//'_DFAC'//'_DFACC'//'_PSIE'//'FTR'//'FTRC'//'_DAT'
TR1  = 1
TR2  = 2
TR3  = 3
TR5  = 12
TR6  = 6
TR8  = 8
TR9  = 9
TR10 = 10
TR11 = 11
TR13 = 13
TR14 = 14
TR15 = 15
TR16 = 16
OPEN(TR1,FILE=FILE01,STATUS='UNKNOWN')
OPEN(TR2,FILE=FILE02,RECL=200,STATUS='UNKNOWN')
OPEN(TR6,FILE=FILE06,RECL=200,STATUS='UNKNOWN')
OPEN(TR8,FILE=FILE08,RECL=200,STATUS='UNKNOWN')
OPEN(TR9,FILE=FILE09,RECL=200,STATUS='UNKNOWN')
OPEN(TR10, FILE=FILE10, RECL=200, STATUS='UNKNOWN')
OPEN(TR5, FILE=FILE05, RECL=200, STATUS='UNKNOWN')
OPEN(TR13, FILE=FILE13, STATUS='UNKNOWN')
OPEN(TR14, FILE=FILE14, RECL=200, STATUS='UNKNOWN')
OPEN(TR15, FILE=FILE15, RECL=200, STATUS='UNKNOWN')
ENDIF

!!! SETTING UP G(R): hic
RANGE1 = 0D0 !
RANGE2 = LX/2D0 ! assuming lx=ly=lz
BWID = RANGE2/WD_1
NBINS = NINT((RANGE2 - RANGE1)/BWID)
RANGE2 = RANGE1 + BWID*NBINS ! MAKES RANGE2 MULTIPLE OF BINWIDTH
ALLOCATE(BINCNT(NBINS))
ALLOCATE(TMPDBIN(NBINS))
ALLOCATE(TMPBINSUM(NBINS))
ALLOCATE(RBIN(NBINS+1))

DO I = 1, NBINS
   BINCNT(I) = 0 ! SET ALL BIN COUNTS TO ZERO INITIALLY
ENDDO

!!! SETTING UP G(R): particle ion
RANGE_1 = 0D0
RANGE_2 = LX/2D0
B_WID = RANGE_2/WD_2
N_BINS = NINT((RANGE_2-RANGE_1)/B_WID)
RANGE_2 = RANGE_1 + B_WID*N_BINS
ALLOCATE(BCNT_PI(N_BINS))
ALLOCATE(TMP_DBIN(N_BINS))
ALLOCATE(TMP_BIN_SUM(N_BINS))
ALLOCATE(R_BIN(N_BINS+1))

DO I = 1, N_BINS
   BCNT_PI(I) = 0 ! SET ALL BIN COUNTS TO ZERO INITIALLY
ENDDO

!!! FILE OUTPUT INPUTS
IF (ICPU .EQ. 0) THEN
   ! CALL F_OUTPUT_1(PSIE, GM_IG,, KND, N_TOTAL, KHI,
   ! BOX_F, DFACTOR, TR1)
   WRITE(TR1,*) '!INPUT PARAMETERS'
   WRITE(TR1,*) 'PSIE: ' , PSIE
   WRITE(TR1,*) 'KND: ' , KND
   WRITE(TR1,*) 'NUMBER OF IONS: ' , N_TOTAL
   WRITE(TR1,*) '(ap/ni^-1/3)): ' , KHI
   WRITE(TR1,*) 'DFACTOR: ' , DFACTOR
   WRITE(TR1,*) 'BOX FACTOR: ' , BOX_F
   WRITE(TR1,*) 'GAMMA_ION: ', GM_IG
   WRITE(TR1,*) 'KND_DILUTE: ', KND_D
   WRITE(TR1,*) 'ETAF: ', ETA_F
   WRITE(TR1,*) 'ETAC: ', ETA_C
   !!! WRITE TO SLURM FILE
   WRITE(*,*) '!INPUTS'
   WRITE(TR1,*) '! INPUTS'
ENDIF
WRITE(*,*) 'PSIE: ', PSIE
WRITE(*,*) 'KND: ', KND
WRITE(*,*) 'NUMBER OF IONS: ', N_TOTAL
WRITE(*,*) '(ap/ni^(-1/3)): ', KHI
WRITE(*,*) 'DFACTOR: ', DFACTOR
WRITE(*,*) 'BOX FACTOR: ', BOX_F
WRITE(*,*) 'GAMMA_ION:', GM_IG
WRITE(*,*) 'KND_DILUTE:', KND_D
WRITE(*,*) 'ETAF:', ETA_F
WRITE(*,*) 'ETAC:', ETA_C
ENDIF

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!!! SEE D AND INITIALIZATION OF POSITIONS AND VELOCITIES
IF (ICPU .EQ. 0) THEN
  CALL DATE_AND_TIME(VALUES=VALUES)
  SEED = 62345678*VALUES(8) + 1234*VALUES(3) + 123*VALUES(5)
  C + FLOOR(12345*N_TOTAL*KND_D*GM_IG*KHI*PSIE) + 12*VALUES(1)
  CALL INIT_POS_VEL(N_TOTAL,LX,LY,LZ,KND_D,UC,VC,WC,X,Y,Z,
       ETA_F,ETA_C,SEED)
  CALL CHECK_COINC(X,Y,Z,N_TOTAL,LX,LY,LZ,SEED)
  UN=0D0
  VN=0D0
  WN=0D0
  CALL CPU_TIME(TSTART)
ENDIF

CALL MPI_BCAST(SEED,1,MPI_INT,0,MPI_COMM_WORLD,IERR)
SEED=SEED+12345*RANK
!BROADCAST VEL INFO
CALL MPI_BCAST(UC,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALLMPI_BCAST(VC,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(WC,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!TRIANGULAR PARA INDEX CALCULATION
INDEX_TRI(NPROCS+1)=0
DO I=1,NPROCS
  TEMP1=N_TOTAL**2/NPROCS+INDEX_TRI(NPROCS-(I-1)+1)**2
  TEMP1=TEMP1**5D-1
  TEMP1=CEILING(TEMP1)
  INDEX_TRI(NPROCS-(I-1))=INT(TEMP1)
ENDDO
INDEX_TRI=N_TOTAL-INDEX_TRI
INDEX_TRI(1)=1
I1_TRI=INDEX_TRI(ICPU+1)
I2_TRI=INDEX_TRI(ICPU+2)-1

!RECTANGULAR PARA INDEX CALCULATION
INDEX_REC(1)=1
TEMP1=N_TOTAL/NPROCS
TEMP1=CEILING(TEMP1)
DO I=2,NPROCS
  INDEX_REC(I)=INDEX_REC(I-1)+INT(TEMP1)
  SEED=SEED+12345*RANK
!BROADCAST VEL INFO
CALL MPI_BCAST(UC,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(VC,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(WC,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
}
ENDDO
INDEX_REC(NPROCS+1)=N_TOTAL+1
I1_REC=INDEX_REC(ICPU+1)
I2_REC=INDEX_REC(ICPU+2)-1

JMP = 1
MEAN_TIME = 0D0
GSTEP = 0
T_REC = 0.0D0
GI_AVG= 0.0D0
FLINE = 0

!!!!!!!!  !!!!!!!!!  !!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!! MAIN LOOP !!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
DO WHILE((FEND_CHECK .EQ. 0) .OR. (JMP .LE. JMPLIM))

!BROAD CAST POSITION INFO
CALL MPI_BCAST(X,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(Y,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(Z,N_TOTAL,MPI_DOUBLE,0,MPI_COMM_WORLD,IERR)

!CALCULATE FORCE AND MAX F AND R
CALL RMIN_F(X,Y,Z,GM_IG,KND_D,PSIE,LX,LY,LZ,R_MIN,KHI
C,N_TOTAL,ETA_F,ETA_C,FX,FY,FZ,I1_TRI,I2_TRI,I1_REC,I2_REC)

! GATHER RESULTS
!!! Send from Fx to Temp with 'sum' as the reduce operator
CALL MPI_REDUCE(FX,TEMP,N_TOTAL,MPI_DOUBLE,MPI_SUM,0,
CMPI_COMM_WORLD,IERR)
FX=TEMP  !REPLACE FX
CALL MPI_REDUCE(FY,TEMP,N_TOTAL,MPI_DOUBLE,MPI_SUM,0,
CMPI_COMM_WORLD,IERR)
FY=TEMP  !REPLACE FY
CALL MPI_REDUCE(FZ,TEMP,N_TOTAL,MPI_DOUBLE,MPI_SUM,0,
CMPI_COMM_WORLD,IERR)
FZ=TEMP  !REPLACE FZ
CALL MPI_REDUCE(R_MIN,TEMP1,1,MPI_DOUBLE,MPI_MIN,0,
CMPI_COMM_WORLD,IERR)
R_MIN=TEMP1

!GET F MAX
TEMP=FX**2+FY**2+FZ**2
TEMP=TEMP**5D-1
F_MAX=MAXVAL(TEMP)
MSDND = 0D0
MSVND = 0D0
!!! COMPUTE DT, MSDND, MSVND
IF (ICPU .EQ. 0) THEN

    CALL MSDV(DFACTOR, DT, R_MIN, KND_D, MSDND, MSVND,
              CFAC1, FAC2, FAC3, FAC4, ETA_F, ETA_C, F_MAX)
ENDIF

! BROADCAST DT INFO
CALL MPI_BCAST(DT, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)

!!! Scattering for Langevin N * 3 equations run
! SCATTER R
CALL MPI_SCATTER(X, SCT_SZ, MPI_DOUBLE, X_SC, SCT_SZ,
                  MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)
CALL MPI_SCATTER(Y, SCT_SZ, MPI_DOUBLE, Y_SC, SCT_SZ,
                  MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)
CALL MPI_SCATTER(Z, SCT_SZ, MPI_DOUBLE, Z_SC, SCT_SZ,
                  MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)

! SCATTER V
CALL MPI_SCATTER(UC, SCT_SZ, MPI_DOUBLE, UC_SC, SCT_SZ,
                 MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)
CALL MPI_SCATTER(VC, SCT_SZ, MPI_DOUBLE, VC_SC, SCT_SZ,
                 MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)
CALL MPI_SCATTER(WC, SCT_SZ, MPI_DOUBLE, WC_SC, SCT_SZ,
                 MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)

! SCATTER F
CALL MPI_SCATTER(FX, SCT_SZ, MPI_DOUBLE, FX_SC, SCT_SZ,
                 MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)
CALL MPI_SCATTER(FY, SCT_SZ, MPI_DOUBLE, FY_SC, SCT_SZ,
                 MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)
CALL MPI_SCATTER(FZ, SCT_SZ, MPI_DOUBLE, FZ_SC, SCT_SZ,
                 MPI_DOUBLE, 0, MPI_COMM_WORLD, IERR)

!!! LANGEVIN EQUATION - N PARTICLES
CALL LANG_NPART_2(X_SC, Y_SC, Z_SC, UC_SC, VC_SC, WC_SC, UN_SC, VN_SC,
                WX, LY, LX, SCT_SZ, FX_SC, FY_SC, FZ_SC,
                SEED)

!!! WRITE TRAJ TO FILE AFTER SIMULATION RAN FOR A CERTAIN TIME
IF (ITER .GT. ITERMIN_1) THEN
    IF (ICPU .EQ. 0) THEN
        IF (MOD(ITER, SKP) .EQ. 0) THEN
            INQUIRE(FILE=FILE06, SIZE=F1SIZE)
            INQUIRE(FILE=FILE15, SIZE=F2SIZE)
            IF ((F1SIZE) .LE. F_LIM) THEN
                DO J1 = 1, N_TOTAL
                    WRITE(TR2, '(ES15.6E2, A, ES15.6E2, A, ES15.6E2)')
                    CX(J1), CHAR(9), Y(J1), CHAR(9), Z(J1)
                ENDDO
                DO LPS = 1, N_TOTAL
        ENDIF
   ENDIF
ENDIF
WRITE(TR6,'(ES15.6E2,A,ES15.6E2,A,ES15.6E2)')
C UN(LPS),CHAR(9),VN(LPS),CHAR(9),WN(LPS)
ENDDO

ELSEIF ((F2SIZE) .LE. F_LIM) THEN

  DO J1=1,N_TOTAL
  WRITE(TR14,'(ES15.6E2,A,ES15.6E2,A,ES15.6E2)')
  CX(J1),CHAR(9),Y(J1),CHAR(9),Z(J1)
  ENDDO

  DO LPS = 1,N_TOTAL
  WRITE(TR15,'(ES15.6E2,A,ES15.6E2,A,ES15.6E2)')
  C UN(LPS),CHAR(9),VN(LPS),CHAR(9),WN(LPS)
  ENDDO

ENDIF
ENDIF
ENDIF
ENDIF ! end of trajectory printing

! START SIMULATION CLOCK AFTER FR2_TLIM FRACTION OF TLIM
IF (ITER .GT. ITERMIN_1) THEN
  TC = TC + DT
  TC_J = TC_J + DT ! TIME FOR EACH COLLISION
ENDIF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! G(r)
IF (ITER .GT. ITERMIN_1) THEN

  !! GAMMA AND KINETIC ENERGY
  IF(MOD(ITER,SKP_3) .EQ. 0) THEN

    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    AVGSQ=0D0

    CALL GET_GAMMA(UN_SC,VN_SC,WN_SC,KND_D,ETA_F,ETA_C,C,SCT_SZ,N_TOTAL,GM_IG,A
    VGSQ)
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    CALL MPI_REDUCE(AVGSQ,TEMP1,1,MPI_DOUBLE,MPI_SUM,C,0,MPI_COMM_WORLD,IERR)

    AVGSQ=TEMP1

    TEMP1=(ETA_C/(KND_D*ETA_F))**2D0
    GM_II = GM_IG/((5.0D-1)*AVGSQ*TEMP1
    GI_AVG = GI_AVG + GM_II
    KEAVG = 0.5*AVGSQ*TEMP1
    FLINE = FLINE + 1

    IF (RANK .EQ. 0) THEN
!!! added inquire statement here
INQUIRE(FILE=FILE13, SIZE=F1SIZE)
IF (F1SIZE .LE. F_LIM) THEN
    WRITE(TR13,*) GM_II, KEAVG
ENDIF

IF (FLINE .EQ. 500000) THEN
    GI_AVG = GI_AVG/DBLE(FLINE)
    WRITE(TR1,*) 'AVERAGE G_II: ', GI_AVG
ENDIF

ENDIF

ENDIF

ENDIF

!  PAIR CORRELATION FUNCTIONS - g-2(R)
IF(MOD(ITER,SKP_2) .EQ. 0) THEN
    GSTEP = GSTEP + 1
    IF (GSTEP .LE. GSTEPMAX) THEN

        ! ION-PARTICLE

        CALL GR_PI(X_SC,Y_SC,Z_SC,TEMP_DBIN,N_BINS,RANGE_1,
                   CRANGE_2,B_WID,GSTEP,SCT_SZ,LX,LY,LZ)

        CALL MPI_REDUCE(TEMP_DBIN,TEMP_BIN_SUM,N_BINS,
                         CMPI_DOUBLE,MPI_SUM,
                         0,MPI_COMM_WORLD,IERR)
        BCNT_PI=BCNT_PI+TEMP_BIN_SUM

        IF (GSTEP .EQ. GSTEPMAX) THEN

            !!! Normalization

            R_BIN(1)=RANGE_1
            DO I = 2,N_BINS+1

                R_BIN(I) = RANGE_1 + (I-1)*B_WID
                TEMP1 = (4D0*PI/3D0)*((R_BIN(I))**3D0
                                       -(R_BIN(I-1))**3D0)
                TEMP1 = DBLE((BCNT_PI(I-1)))/(TEMP1*GSTEP*RHOD)

            END DO

            IF (RANK .EQ. 0) THEN
                WRITE(TR9,*) (0.5*R_BIN(I)+0.5*R_BIN(I-1)),TEMP1
            ENDIF
        ENDDO
    ENDF

ENDIF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
CALL PAIRPOT_GR(X,Y,Z,LX,LY,LZ,N_TOTAL,TEMPDBIN, CNBINS,RANGE1,RANGE2,BWID,I1_TRI,I2_TRI)

!!! n_bins changed to nbins here
CALL MPI_REDUCE(TEMPDBIN,TEMPBINSUM,NBINS,
    CMPI_DOUBLE,MPI_SUM,0,MPI_COMM_WORLD,IERR)

BINCNT=BINCNT+TEMPBINSUM

IF (GSTEP .EQ. GSTEPMAX) THEN
!!! Normalization
    CALL NORM_GR(BINCNT,NBINS,LX,LY,LZ,RANGE1,RANGE2,
        CBWID,N_TOTAL,GSTEP,TEMPDBIN)

RBIN(1)=RANGE1
DO I=2,NBINS+1
    RBIN(I) = RANGE1 + (I-1)*BWID
    IF (RANK .EQ. 0) THEN
        WRITE(TR10,*) (0.5*RBIN(I)+0.5*RBIN(I-1)),
            CTEMPDBIN(I-1)
    ENDIF
ENDDO

FEND_CHECK = 1 ! SET TO 1 AFTER WRITING G2R TO FILE
ENDDO

ENDIF

ITER = ITER + 1

!!! CHECK COLLISION
! COLL CHECK PARA
J1=0
DO I=1,SCT_SZ
    IF ((X_SC(I)**2+Y_SC(I)**2+Z_SC(I)**2) .LE. 1D0) THEN
        J1=J1+1 ! STORE COLL INFO
CALL DATE_AND_TIME(VALUES=VALUES)

SEED = 654321*J1*VALUES(8) + 12345*J1*VALUES(3) + 10*RANK

CALL RE_INIT(X_SC(I),Y_SC(I),Z_SC(I),UN_SC(I),VN_SC(I),
CWN_SC(I),KND_D,LX,LY,LZ,ETA_F,ETA_C,SEED)

ENDIF
ENDDO

! GATHER NEW POSITION!
CALL MPI_GATHER(X_SC,SCT_SZ,MPI_DOUBLE,X,SCT_SZ,
CMPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALL MPI_GATHER(Y_SC,SCT_SZ,MPI_DOUBLE,Y,SCT_SZ,
CMPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
CALL MPI_GATHER(Z_SC,SCT_SZ,MPI_DOUBLE,Z,SCT_SZ,
CMPI_DOUBLE,0,MPI_COMM_WORLD,IERR)
!
!

IF (ITER .GT. ITERMIN_1) THEN
I=0
CALL MPI_REDUCE(J1,I,1,MPI_INT,MPI_SUM,0,
CMPI_COMM_WORLD,IERR)
IF ((RANK .EQ. 0) .AND. I .GT. 0) THEN
IF (JMP .LE. JMPLIM) THEN
MEAN_TIME = TC/DBLE(JMP)
H_MEAN=(1D0/(MEAN_TIME*DBLE(N_TOTAL)))*(BOX_F**3D0)
WRITE(TR8,*) H_MEAN,JMP
CALL CPU_TIME(TFINISH)
TDIFF = ABS(TFINISH - TSTART)
WRITE(TR5,*) TC,TDIFF
ENDIF
JMP=JMP+I
ENDIF
CALL MPI_BCAST(JMP,1,MPI_INT,0,MPI_COMM_WORLD,IERR)
ENDIF

ENDIF
UC=UN
VC=VN
WC=WN

ENDDO !! END MAIN LOOP

! CALL COLL_KERNEL(BOX_F,TC,COLL_COUNT,TR5,TR1)
!!! WRITE COLLISION KERNEL TO FILE
IF (ICPU .EQ. 0) THEN
  WRITE(TR1,*) 'TOTAL NUMBER OF ITERATIONS:',ITER
  WRITE(TR1,*) 'TC: ', TC
  WRITE(TR1,*) 'JUMPS: ', JMP
  WRITE(TR1,*) 'NON-DIM CR: ', (LX*LY*LZ)*(DBLE(JMP))/TC
ENDIF

!!! DEALLOCATE AND CLOSE
DEALLOCATE(X)
DEALLOCATE(Y)
DEALLOCATE(Z)
DEALLOCATE(UC)
DEALLOCATE(VC)
DEALLOCATE(WC)
DEALLOCATE(UN)
DEALLOCATE(VN)
DEALLOCATE(WN)
DEALLOCATE(FX)
DEALLOCATE(FY)
DEALLOCATE(FZ)
DEALLOCATE(BINCNT)
DEALLOCATE(BCNT_PI)
DEALLOCATE(T_REC)
DEALLOCATE(TEMP_DBIN)
DEALLOCATE(TEMPDBIN)
DEALLOCATE(TEMP_BIN_SUM)
DEALLOCATE(TEMPBINSUM)

!!! CLOSE FILES
IF (ICPU .EQ. 0) THEN
  CLOSE(TR1)
  CLOSE(TR2)
  CLOSE(TR5)
  CLOSE(TR6)
  CLOSE(TR8)
  CLOSE(TR9)
  CLOSE(TR10)
  CLOSE(TR13)
  CLOSE(TR14)
  CLOSE(TR15)
ENDIF
CALL MPI_FINALIZE(IERR)

END PROGRAM HIC_FIXED_PARTICLE

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! FILE OUTPUT 1

SUBROUTINE F_OUTPUT_1(PSIE,GM_IG,KND,N_TOTAL,KHI,
BOX_F,DFACTOR,TR)
IMPLICIT NONE
DOUBLE PRECISION, INTENT(IN) :: PSIE,BOX_F
DOUBLE PRECISION, INTENT(IN) :: KND,KHI,DFACTOR,GM_IG
INTEGER, INTENT(IN)::TR,N_TOTAL

WRITE(TR,*) '*INPUT PARAMETERS*
WRITE(TR,*)
WRITE(TR,*) 'PSIE: ', PSIE
WRITE(TR,*) 'KND: ' ,KND
WRITE(TR,*) 'NUMBER OF IONS: ' ,N_TOTAL
WRITE(TR,*) '(ap/ni^-1/3)): ',KHI
WRITE(TR,*) 'DFACTOR: ', DFACTO
WRITE(TR,*) 'BOX FACTOR: ' ,BOX_F
WRITE(TR,*) 'GAMMA_ION:', GM_IG
RETURN
END SUBROUTINE F_OUTPUT_1

!!! INITIALIZATION OF POSITION AND VELOCITY FOR ALL PARTICLES

SUBROUTINE INIT_POS_VEL(N_TOTAL,LX,LY,LZ,KND_D,UC,VC,WC,
CX,Y,Z,ETA_F,ETA_C,SEED)
IMPLICIT NONE
INTEGER, INTENT(IN) :: N_TOTAL
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: X,Y,Z
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: UC,VC,WC
DOUBLE PRECISION, INTENT(IN):: LX,LY,LZ
DOUBLE PRECISION, INTENT(IN): KND_D,ETA_F,ETA_C
DOUBLE PRECISION NORMRAND
INTEGER I,SEED

DO I = 1,N_TOTAL
X(I) = (-LX/2D0) + LX*RAN(SEED)
Y(I) = (-LY/2D0) + LY*RAN(SEED)
Z(I) = (-LZ/2D0) + LZ*RAN(SEED)
UC(I) = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
VC(I) = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
WC(I) = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
ENDDO

RETURN
END SUBROUTINE INIT_POS_VEL

!!! CHECK THAT THE IONS DO NOT OCCUPY THE GRAIN VOLUME

SUBROUTINE CHECK_COI
NC(X,Y,Z,N_TOTAL,LX,LY,LZ,SEED)
IMPLICIT NONE
INTEGER, INTENT(IN) :: N_TOTAL
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: X,Y,Z
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: UC,VC,WC
DOUBLE PRECISION, INTENT(IN):: LX,LY,LZ
DOUBLE PRECISION, INTENT(IN): KND_D,ETA_F,ETA_C
DOUBLE PRECISION NORMRAND
INTEGER I,SEED

DO I = 1,N_TOTAL
X(I) = (-LX/2D0) + LX*RAN(SEED)
Y(I) = (-LY/2D0) + LY*RAN(SEED)
Z(I) = (-LZ/2D0) + LZ*RAN(SEED)
UC(I) = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
VC(I) = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
WC(I) = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
ENDDO

RETURN
END SUBROUTINE CHECK_COI

86
INTEGER, INTENT(IN) :: N_TOTAL  
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: X,Y,Z  
INTEGER COINC_FLAG  
INTEGER J1, SEED  
DOUBLE PRECISION RD, SPD  
DOUBLE PRECISION LX, LY, LZ  
  
SPD = 1.0D0 ! COLLISION CONDITION  
  
DO J1 = 1,N_TOTAL  
   RD = SQRT((X(J1))^2D0+(Y(J1))^2D0+(Z(J1))^2D0)  
   IF (RD .LE. SPD) THEN  
       CALL ONBOX(LX,LY,LZ,SEED,X(J1),Y(J1),Z(J1))  
   ENDIF  
ENDDO  
  
RETURN  
END SUBROUTINE CHECK_COINC  

!!! MIN R AND MAX F COMPUTE, FILLS FX,FY,FZ 2D-ARRAY  
SUBROUTINE RMIN_F(X,Y,Z,GM_IG,KND_D,PSIE,LX,LY,LZ,R_MIN,KHI,C,N_TOTAL,ETA_F,ETA_C,FX,FY,FZ,I1_TRI,I2_TRI,I1_REC,I2_REC)  
IMPLICIT NONE  
INTEGER, INTENT(IN):: N_TOTAL  
INTEGER J1,J2  
INTEGER I1_TRI,I2_TRI,I1_REC,I2_REC  
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: X,Y,Z  
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: FX,FY,FZ  
DOUBLE PRECISION KND_D,GM_IG,PSIE,KHI,ETA_F,ETA_C  
DOUBLE PRECISION F, R_MIN, R  
DOUBLE PRECISION LX,LY,LZ,DX,DY,DZ  
DOUBLE PRECISION FXS,FYS,FZS  
DOUBLE PRECISION PRE_FAC1,PRE_FAC2,PRE_FAC3,PRE_FAC4  
  
PRE_FAC1 = (ETA_C/ETA_F)**2D0  
PRE_FAC2 = (ETA_C/ETA_F)**2D0  
PRE_FAC3 = (KND_D**2D0)*GM_IG/(PRE_FAC1 * KHI)  
PRE_FAC4 = -PSIE*(KND_D**2D0)/PRE_FAC2  
  
!RESET FORCE  
FX=0D0  
FY=0D0  
FZ=0D0  
R_MIN=1D99  
  
DO J1 = I1_TRI,I2_TRI  
   DO J2 = (J1+1),N_TOTAL  
      DX = X(J1) - X(J2)  
      DY = Y(J1) - Y(J2)  
      DZ = Z(J1) - Z(J2)  
      CALL ADJUST_PB(DX,DY,DZ,LX,LY,LZ)  
   ENDDO  
ENDDO  

87
R = SQRT((DX)**2D0+(DY)**2D0+(DZ)**2D0)

F = PRE_FAC3/(R**2D0)

R_MIN = MIN(R_MIN,R)

FX(J1) = FX(J1)+F*(DX/R)
FY(J1) = FY(J1)+F*(DY/R)
FZ(J1) = FZ(J1)+F*(DZ/R)

FX(J2) = FX(J2)-F*(DX/R)
FY(J2) = FY(J2)-F*(DY/R)
FZ(J2) = FZ(J2)-F*(DZ/R)

ENDDO !END J2 LOOP
ENDDO ! END J1 LOOP

DO J1=I1_REC,I2_REC
   ! ION-PARTICLE FORCES CALCULATION
   R = SQRT(X(J1)**2D0 + Y(J1)**2D0 + Z(J1)**2D0)
   F = PRE_FAC4/(R**2D0)

   FX(J1) = FX(J1)+F*X(J1)/R
   FY(J1) = FY(J1)+F*Y(J1)/R
   FZ(J1) = FZ(J1)+F*Z(J1)/R

   R_MIN = MIN(R_MIN,R)

ENDDO

RETURN
END SUBROUTINE RMIN_F

!!! MSDV SUBROUTINE
!!! (COMPUTES MSD AND MSV AND REFINES DT)
SUBROUTINE MSDV(DFACTOR,DT,R_MIN,KND_D,MSDND,MSVND,
                  CFAC1,FAC2,FAC3,FAC4,ETA_F,ETA_C,F_MAX)
IMPLICIT DOUBLE PRECISION(A-Z)

   DT1 = ((R_MIN*ETA_C/ETA_F)**2D0)/(KND_D**2D0)
   DT2 = 1D0/ABS(F_MAX)
   DT=DFACTOR*MIN(DT1,DT2)
   FAC1 = 1D0-EXP(-1D0*DT)
   FAC2 = 1D0+EXP(-1D0*DT)
   FAC3 = 1D0-EXP(-2D0*DT)
   FAC4 = FAC1/FAC2

   MSDND2 = ((KND_D*ETA_F/(ETA_C))**2D0)*(2D0*DT-4D0*FAC1/FAC2)

   DO WHILE (MSDND2 .LE. 0D0)
      DT=DT/2D0

   ENDDO ! END MSDV

RETURN
END SUBROUTINE MSDV
FAC1 = 1D0 - EXP(-1D0*DT)
FAC2 = 1D0 + EXP(-1D0*DT)
FAC3 = 1D0 - EXP(-2D0*DT)
FAC4 = FAC1/FAC2

MSDND2 = ((KND_D*ETA_F/ETA_C)**2D0)*(2D0*DT-4D0*FAC1/FAC2)
ENDDO

MSVND = (KND_D*ETA_F/(ETA_C))*((1D0-EXP(-2D0*DT))**5D-1)
MSDND = MSDND2**(5D-1)

RETURN
END SUBROUTINE MSDV

SUBROUTINE RE_INIT(TJ1,TJ2,TJ3,VL1,VL2,VL3,KND_D, CLX,LY,LZ,ETA_F,ETA_C,SEED)
IMPLICIT NONE
DOUBLE PRECISION TJ1,TJ2,TJ3,VL1,VL2,VL3,KND_D,
CLX,LY,LZ,ETA_F,ETA_C
DOUBLE PRECISION RDIST
DO WHILE(RDIST .LE. 1.0D0)
TJ1 = (-LX/2D0) + LX*RAN(SEED)
TJ2 = (-LY/2D0) + LY*RAN(SEED)
TJ3 = (-LZ/2D0) + LZ*RAN(SEED)
VL1 = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
VL2 = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
VL3 = (KND_D*ETA_F/ETA_C)*NORMRAND(SEED)
RDIST = SQRT(TJ1**2D0+TJ2**2D0+TJ3**2D0)
ENDDO
RETURN
END SUBROUTINE RE_INIT

!!! Collision information
SUBROUTINE F_OUTPUT_8(TR,COLL_COUNT,LX,LY,LZ,T_CPU,TC,TC_J)
IMPLICIT NONE
INTEGER TR,COLL_COUNT
DOUBLE PRECISION TC,TC_J
DOUBLE PRECISION H_TEMP1,LX,LY,LZ
DOUBLE PRECISION T_CPU
H_TEMP1 = (LX*LY*LZ)*(DBLE(COLL_COUNT))/TC !AVERAGE RATE
! WRITE INSTANTANEOUS RATE, AVERAGE RATE AND CPU TIME
WRITE(TR,*H_TEMP1,T_CPU,TC_J
TC_J=0.0D0
RETURN

89
END SUBROUTINE F_OUTPUT_8

! COLLISION KERNEL CALCS.
SUBROUTINE COLL_KERNEL(BOX_F,TC,COLL_COUNT,TR,TR_1)

IMPLICIT NONE
DOUBLE PRECISION MEAN_TIME,TC,BOX_F
INTEGER TR,COLL_COUNT,TR_1
DOUBLE PRECISION H_MEAN

MEAN_TIME = TC/DBLE(COLL_COUNT)
H_MEAN=(1D0/MEAN_TIME)*(BOX_F**3D0)

WRITE(TR,*) H_MEAN,TC,COLL_COUNT
WRITE(TR_1,*) 'H_MEAN(ND_RATE): ', H_MEAN

RETURN

END SUBROUTINE COLL_KERNEL

!!! CALCULATE GAMMA_ION
SUBROUTINE GET_GAMMA(UN_SC,VN_SC,WN_SC,KND_D,ETA_F,ETA_C,SCT_SZ,N_TOTAL,GM_IG,AVGSQ)

IMPLICIT NONE
INTEGER SCT_SZ
INTEGER N_TOTAL,J1
DOUBLE PRECISION KND_D,ETA_F,ETA_C,GM_IG
DOUBLE PRECISION, DIMENSION(SCT_SZ)::UN_SC,VN_SC,WN_SC
DOUBLE PRECISION SUMSQ,AVGSQ

!PREFAC = (KHI*ETA_C/(KND*ETA_F))**2D0
SUMSQ = 0D0
!WRITE(*,*) SCT_SZ
!AVGSQ = 0.0D0
!GM_II=0.0D0
!KEAVG=0.0D0
!!!!!!!!!!!!!!!!!
DO J1 = 1,SCT_SZ
    SUMSQ = SUMSQ+UN_SC(J1)**2D0+VN_SC(J1)**2D0+WN_SC(J1)**2D0 !SUM OF THE ION VELOCITY SQUARE
ENDDO
!WRITE(*,*) 'HERE!!!'
AVGSQ = SUMSQ/DBLE(N_TOTAL)
!GM_II = GM_IG/((5.0D-1)*AVGSQ*PREFAC)
!GI_AVG = GI_AVG + GM_II
!KEAVG = 0.5*AVGSQ*PREFAC
!WRITE(TR,*) GM_II, KEAVG

RETURN
SUBROUTINE F_OUTPUT_14(X,Y,Z,UN,VN,WN,N_TOTAL,KND,KHI,CGM_IG,LX,LY,LZ,Avg_P,INT_E,ETA_F,ETA_C,TR_14,TR_15,TR_16)
IMPLICIT NONE
INTEGER J1,J2,N_TOTAL,TR_16,TR_15,TR_14
DOUBLE PRECISION,DIMENSION(N_TOTAL)::X,Y,Z
DOUBLE PRECISION,DIMENSION(N_TOTAL)::UN,VN,WN
DOUBLE PRECISION,INTENT(IN)::KND,KHI,GM_IG,ETA_F,ETA_C
DOUBLE PRECISION,DIMENSION(1:6)::PRES ! XX,YY,ZZ,XY,YZ,ZX
DOUBLE PRECISION,PREFAC_1,PREFAC_2
DOUBLE PRECISION DX,DY,DZ,LX,LY,LZ,RI
DOUBLE PRECISION U_INT,VSQ
DOUBLE PRECISION AVG_P,INT_E

PREFAC_1 = (KHI*ETA_C/(KND*ETA_F))**2.0D0
PREFAC_2 = GM_IG*ETA_F/(KHI*ETA_C)

PRES = 0.0D0
U_INT = 0.0D0

DO J1 = 1,N_TOTAL-1
     PRES(1) = PRES(1) + PREFAC_1*UN(J1)*UN(J1)
     PRES(2) = PRES(2) + PREFAC_1*VN(J1)*VN(J1)
     PRES(3) = PRES(3) + PREFAC_1*WN(J1)*WN(J1)
     PRES(4) = PRES(4) + PREFAC_1*UN(J1)*VN(J1)
     PRES(5) = PRES(5) + PREFAC_1*VN(J1)*WN(J1)
     PRES(6) = PRES(6) + PREFAC_1*WN(J1)*UN(J1)
     VSQ = UN(J1)**2D0 + VN(J1)**2D0 + WN(J1)**2D0
     U_INT = U_INT + (5.0D-1)*PREFAC_1*VSQ
     DO J2 = J1+1,N_TOTAL
           DX = X(J1) - X(J2)
           DY = Y(J1) - Y(J2)
           DZ = Z(J1) - Z(J2)
           CALL ADJUST_PB(DX,DY,DZ,LX,LY,LZ)
           RI = SQRT(DX**2D0 + DY**2D0 + DZ**2D0)
           PRES(1) = PRES(1) + PREFAC_2*(DX*DX/(RI**3D0))
           PRES(2) = PRES(2) + PREFAC_2*(DY*DY/(RI**3D0))
           PRES(3) = PRES(3) + PREFAC_2*(DZ*DZ/(RI**3D0))
           PRES(4) = PRES(4) + PREFAC_2*(DX*DY/(RI**3D0))
           PRES(5) = PRES(5) + PREFAC_2*(DY*DZ/(RI**3D0))
           PRES(6) = PRES(6) + PREFAC_2*(DZ*DX/(RI**3D0))
           U_INT = U_INT + PREFAC_2/RI
     ENDDO
  ENDDO
PRES(1) = PRES(1) + PREFAC_1*(UN(N_TOTAL))**2.0D0
PRES(2) = PRES(2) + PREFAC_1*(VN(N_TOTAL))**2.0D0

END SUBROUTINE F_OUTPUT_14
PRES(3) = PRES(3) + PREFAC_1*(WN(N_TOTAL))**2.0D0
PRES(4) = PRES(4) + PREFAC_1*UN(N_TOTAL)*VN(N_TOTAL)
PRES(5) = PRES(5) + PREFAC_1*VN(N_TOTAL)*WN(N_TOTAL)
PRES(6) = PRES(6) + PREFAC_1*WN(N_TOTAL)*UN(N_TOTAL)

VSQ = UN(N_TOTAL)**2D0 + VN(N_TOTAL)**2D0 + WN(N_TOTAL)**2D0
U_INT = U_INT + (5.0D-1)*PREFAC_1*VSQ

AVG_P = AVG_P + (PRES(1) + PRES(2) + PRES(3))/3.0D0  !AVERAGE PRESSURE
INT_E = INT_E + U_INT/DBLE(N_TOTAL) !AVERAGE ENERGY PER ION

WRITE(TR_14,*) PRES(1),PRES(2),PRES(3) !INSTANTANEOUS PRESSURE
WRITE(TR_16,*) PRES(4),PRES(5),PRES(6) !SHEAR
WRITE(TR_15,*) U_INT !INTERNAL ENERGY (TOTAL KE + TOTAL PE)

RETURN
END SUBROUTINE F_OUTPUT_14

!!! LANGEVIN
SUBROUTINE LANG_NPART_2(X,Y,Z,UC,VC,WC,UN,VN,WN,KND_D,ETA_F,
CETA_C,DT,LX,LY,LZ,SCT_SZ,FX,FY,FZ,SEED)

IMPLICIT NONE
INTEGER SCT_SZ
INTEGER J1,SEED
DOUBLE PRECISION, DIMENSION(SCT_SZ) :: X,Y,Z
DOUBLE PRECISION, DIMENSION(SCT_SZ) :: UC,VC,WC
DOUBLE PRECISION, DIMENSION(SCT_SZ) :: UN,VN,WN
DOUBLE PRECISION, DIMENSION(SCT_SZ) :: FX,FY,FZ
DOUBLE PRECISION FAC1,FAC2,FAC3,FAC4
DOUBLE PRECISION LX,LY,LZ
DOUBLE PRECISION MSVND,MSDND
DOUBLE PRECISION KND_D,ETA_F,ETA_C
DOUBLE PRECISION DELX,DELY,DELZ

FAC1 = 1D0-EXP(-1D0*DT)
FAC2 = 1D0+EXP(-1D0*DT)
FAC3 = 1D0-EXP(-2D0*DT)
FAC4 = FAC1/FAC2

MSVND = (KND_D*ETA_F/(ETA_C))*((1D0-EXP(-2D0*DT))**(5D-1))
MSDND2 = ((KND*ETA_F/(KHI*ETA_C))**2D0)*(2D0*DT-4D0*FAC1/FAC2)
MSDND = (((KND_D*ETA_F/(ETA_C))**2D0)
C*(2D0*DT-4D0*FAC1/FAC2))**5D-1

WRITE(*,*) DT,MSVND,MSDND,FAC1,FAC2,FAC3,FAC4
WRITE(*,*) MSVND,MSDND,DT

DO J1 = 1,SCT_SZ
!FXSUM = SUM(FX(J1,:))
!FYSUM = SUM(FY(J1,:))
!FZSUM = SUM(FZ(J1,:))

UN(J1) = (UC(J1))*(1D0-FAC1) +
C
FX(J1)*FAC1 + (MSVND)*NORMRAND(SEED)
DELX = FAC4*(UC(J1)+UN(J1)-2D0*FX(J1))+
C
FX(J1)*DT + (MSDND)*NORMRAND(SEED)
X(J1) = X(J1) + DELX
X(J1) = X(J1) - LX*ANINT(X(J1)/LX)

VN(J1) = VC(J1)*(1D0-FAC1) +
C
FY(J1)*FAC1 + (MSVND)*NORMRAND(SEED)
DELY = FAC4*(VC(J1)+VN(J1)-2D0*FY(J1))+
C
FY(J1)*DT + (MSDND)*NORMRAND(SEED)
Y(J1) = Y(J1) + DELY
Y(J1) = Y(J1) - LY*ANINT(Y(J1)/LY)

WN(J1) = WC(J1)*(1D0-FAC1) +
C
FZ(J1)*FAC1 + (MSVND)*NORMRAND(SEED)
DELZ = FAC4*(WC(J1)+WN(J1)-2D0*FZ(J1))+
C
FZ(J1)*DT + (MSDND)*NORMRAND(SEED)
Z(J1) = Z(J1) + DELZ
Z(J1) = Z(J1) - LZ*ANINT(Z(J1)/LZ)
ENDDO

RETURN
END SUBROUTINE LANG_NPART_2

!!! PAIR POTENTIAL - HISTOGRAM
SUBROUTINE PAIRPOT_GR(X,Y,Z,LX,LY,LZ,N_TOTAL,TEMPDBIN, CNBINS,RANGE1,RANGE2,BWID,I1_TRI,I2_TRI)
IMPLICIT NONE
INTEGER N_TOTAL,TR,NBINS
INTEGER J1,J2,I,I1_TRI,I2_TRI
DOUBLE PRECISION RANGE1,RANGE2,BWID !RANGE AND BIN-WIDTH
DOUBLE PRECISION, DIMENSION(N_TOTAL) :: X,Y,Z
DOUBLE PRECISION, DIMENSION(NBINS) :: TEMPDBIN !BIN FREQNO.
INTEGER BINDEX !NO OF BINS,INDEX FOR EACH BIN,FREQ.FOR EACH BIN

!COMPUTE PAIR-WISE DISTANCES
TEMPDBIN=0D0
DO J1 = I1_TRI,I2_TRI
DO J2 = J1+1,N_TOTAL
DX = X(J1) - X(J2)
DY = Y(J1) - Y(J2)
DZ = Z(J1) - Z(J2)
CALL ADJUST_PB(DX,DY,DZ,LX,LY,LZ)
RDIST=SQRT((DX**2D0)+(DY**2D0)+(DZ**2D0))
IF (RDIST .LE. LX/2D0) THEN
BINDEX = CEILING(ABS(RDIST-RANGE1)/BWID)
ENDDO

RETURN
END SUBROUTINE PAIRPOT_GR
TEMPPBIN(BINDEX) = TEMPPBIN(BINDEX)+2D0 !count for that bin

ENDIF
ENDDO
ENDDO

!!! NORMALIZE BIN COUNTS USING SHELL VOLUME
      IF (GSTEP .EQ. GSTEPMAX) THEN
      CALL NORM_GR(BINCNT,NBINS,LX,LY,LZ,RANGE1,
                   RANGE2,BWID,N_TOTAL,GSTEP,TR)
      ENDIF
RETURN
END SUBROUTINE PAIRPOT_GR

!!! NORMALIZE HISTOGRAM FOR G(R)
SUBROUTINE NORM_GR(BINCNT,NBINS,LX,LY,LZ,RANGE1,RANGE2,
CBWID,N_TOTAL,GSTEP,BIN_NORM)
IMPLICIT NONE

INTEGER NBINS,N_TOTAL,TR,GSTEP,I
DOUBLE PRECISION,DIMENSION(NBINS) :: BINCNT
DOUBLE PRECISION LX,LY,LZ,RANGE1,RANGE2,BWID
DOUBLE PRECISION BOXVOL,P_DENS,BINVOL,PI
DOUBLE PRECISION, DIMENSION(NBINS+1) :: R_BIN !BIN NODE VALUES
DOUBLE PRECISION, DIMENSION(NBINS) :: BIN_NORM !NORMALIZED BIN

BOXVOL = LX*LY*LZ
P_DENS = N_TOTAL/BOXVOL
R_BIN(1) = RANGE1
PI = 4.D0*DATAN(1.D0)

DO I = 2,NBINS+1
      R_BIN(I) = RANGE1 + (I-1)*BWID
      BINVOL = (4D0*PI/3D0)*((R_BIN(I))**3D0 - (R_BIN(I-1))**3D0)
      BIN_NORM(I-1) = ((BINCNT(I-1)))/(P_DENS*BINVOL*N_TOTAL*GSTEP)
      WRITE(TR,*),(0.5*R_BIN(I)+0.5*R_BIN(I-1)),BIN_NORM(I-1)
ENDDO
RETURN
END SUBROUTINE NORM_GR

!!! PARTICLE-ION G(R)
! SUBROUTINE GR_Pi(X,Y,Z,BCNT_Pi,N_BINS,RANGE_1,RANGE_2,B_WID,GSTEP,
!                N_TOTAL,LX,LY,LZ,GSTEPMAX,TR)
SUBROUTINE GR_Pi(X_SC,Y_SC,Z_SC,DBIN,N_BINS,RANGE_1,
                CRANGE_2,B_WID,GSTEP,N_SC,LX,LY,LZ)
IMPLICIT NONE
INTEGER N_SC,J1,J2,N_BINS,BPINDX,GSTEP,B1,TR,GSTEPMAX
DOUBLE PRECISION, DIMENSION(N_BINS)::DBIN
DOUBLE PRECISION,DIMENSION(N_SC)::X_SC,Y_SC,Z_SC
DOUBLE PRECISION R_PI, RANGE_1, RANGE_2, B_WID
DOUBLE PRECISION BX_VOL, BN_VOL, LX, LY, LZ, PI, RHOD
DOUBLE PRECISION, DIMENSION(N_BINS+1) :: RBIN
DOUBLE PRECISION, DIMENSION(N_BINS) :: BNX1

DBIN=0D0
BX_VOL = LX*LY*LZ
RBIN(1) = RANGE_1
PI = 4.D0*DATAN(1.D0)
!
RHOD = (N_TOTAL+1)/BX_VOL
!
DO J1 = 1, N_SC
  R_PI = SQRT(X_SC(J1)**2D0 + Y_SC(J1)**2D0 + Z_SC(J1)**2D0)
  IF (R_PI .LE. LX/2D0) THEN
    BPINDX = CEILING(ABS(R_PI - RANGE_1)/B_WID)
    DBIN(BPINDX) = DBIN(BPINDX) + 1D0
  ENDIF
ENDDO
!
IF (GSTEP .EQ. GSTEPMAX) THEN
!!! Normalization
!
!! DO B1 = 2, N_BINS+1
!!  RBIN(B1) = RANGE_1 + (B1-1)*B_WID
!!  BN_VOL = (4D0*PI/3D0)*((RBIN(B1))**3D0 - (RBIN(B1-1))**3D0)
!!  BNORM(B1-1) = DBLE((BCNT_PI(B1-1)))/(BN_VOL*GSTEP*RHOD)
!!! WRITE STATEMENT
!!  WRITE(TM,*),(0.5*RBIN(B1)+0.5*RBIN(B1-1)),BNORM(B1-1)
!! ENDDO
!! ENDF
!
RETURN
END SUBROUTINE

!!! ADJUST FOR PERIODICITY
SUBROUTINE ADJUST_PB(DX, DY, DZ, LX, LY, LZ)
IMPLICIT NONE
DOUBLE PRECISION DX, DY, DZ
DOUBLE PRECISION LX, LY, LZ
!
IF (DX .GT. LX/2D0) THEN
  DX = DX - LX
ELSEIF (DX .LT. -LX/2D0) THEN
  DX = DX + LX
ENDIF
!
IF (DY .GT. LY/2D0) THEN
  DY = DY - LY
ELSEIF (DY .LT. -LY/2D0) THEN
  DY = DY + LY
ENDIF
!
IF (DZ .GT. LZ/2D0) THEN
  DZ = DZ - LZ
ELSEIF (DZ .LT. -LZ/2D0) THEN
  DZ = DZ + LZ
ENDIF
ENDIF

RETURN
END SUBROUTINE ADJUST_PB

!!! NORMALLY DISTRIBUTED RANDOM NUMBERS
DOUBLE PRECISION FUNCTION NORMRAND(SEED)
IMPLICIT NONE
INTEGER SEED
  INTEGER K

  NORMRAND=0D0
  DO K=1,12
    NORMRAND=NORMRAND+RAN(SEED)
  ENDDO

  NORMRAND = NORMRAND - 6D0
END FUNCTION NORMRAND

SUBROUTINE ONBOX(LX,LY,LZ,SEED,X,Y,Z)
!!ASSIGNING POSITIONS ON THE BOX
IMPLICIT NONE
DOUBLE PRECISION LX,LY,LZ,X,Y,Z
DOUBLE PRECISION BOX(1:3)
INTEGER SEED,DICE,T(1:6)

DICE=FLOOR(6*RAN(SEED))

IF (DICE .EQ. 0) THEN

  T(1)=1
  T(2)=0
  T(3)=1
  T(4)=1
  T(5)=1
  T(6)=1

ELSEIF (DICE .EQ. 1) THEN

  T(1)=0
  T(2)=0
  T(3)=1
  T(4)=1
  T(5)=1
  T(6)=1

ELSEIF (DICE .EQ. 2) THEN

  T(1)=1
  T(2)=1
  T(3)=1
  T(4)=0
  T(5)=1
  T(6)=1
ELSEIF (DICE .EQ. 3) THEN

T(1)=1
T(2)=1
T(3)=0
T(4)=0
T(5)=1
T(6)=1

ELSEIF (DICE .EQ. 4) THEN

T(1)=1
T(2)=1
T(3)=1
T(4)=1
T(5)=1
T(6)=0

ELSE

T(1)=1
T(2)=1
T(3)=1
T(4)=1
T(5)=0
T(6)=0

ENDIF

X=-LX/2*T(1)+(1-T(1))*LX/2+LX*T(2)*RAN(SEED)
Y=-LY/2*T(3)+(1-T(3))*LY/2+LY*T(4)*RAN(SEED)
Z=-LZ/2*T(5)+(1-T(5))*LZ/2+LZ*T(6)*RAN(SEED)

RETURN

END SUBROUTINE ONBOX