Multiscale modeling of heat transfer

By

Seyed Hossein Hashemi Ghermezi

A dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Tarek I. Zohdi, Chair
Professor Hayden Taylor
Professor Reza Alam
Professor James Demmel

Fall 2016
Multiscale modeling of heat transfer

Copyright 2016
by
Seyed Hossein Hashemi Ghermezi
Abstract

Multiscale modeling of heat transfer

by

Seyed Hossein Hashemi Ghermezi

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Tarek I. Zohdi, Chair

The dynamics of a complex fluid or fluid close to interfaces is affected by its microscopic structure. Because the molecular level dynamics is playing a role, and macroscopic scale is effected by lower levels, in these situations the fluid cannot be completely described by continuum theory and Navier-Stokes equation. However, molecular dynamics also is not a practical approach; the large number of particles in the large scales makes it computational impossible. I solve the problem with a Mesoscopic simulation methods by making use of information from different length scale, macroscopic and microscopic, and solve the problem in middle ground in the mesoscale level. However, by using the coupling of continuum and a mesoscale method, I could solve an industry size problem with accuracy of lower scale problem. For the first time, I present a novel coupling method of dissipative particle dynamics to continuum model to calculate heat transfer. The results of the method shows an excellent agreement with a broad range of experimental results in various cases.
To my Wonderful Family
# Contents

List of Figures ........................................ iv

List of Tables ......................................... vi

Acknowledgments ....................................... vii

1 Introduction ........................................ 1
   1.1 Multiscale modeling and lower scales .......... 1

2 Some basic theories in Molecular Dynamics .......... 3
   2.1 Introduction .................................. 3
   2.2 Classical Mechanics ............................ 3
      2.2.1 Liouville Equation ......................... 4
   2.3 Statistical Mechanics ........................... 6
   2.4 Fluid Dynamics and Continuum Mechanics .... 7
   2.5 Kinetic Theory ................................ 9
      2.5.1 Bogoliubov-Born-Green-Kirkwood-Yvon(BBGKY) hierarchy 10
      2.5.2 Boltzmann Equation ....................... 11
      2.5.3 Molecular Dynamics ....................... 12
      2.5.4 Euler Algorithm ......................... 13
      2.5.5 Verlet Algorithm ......................... 13
      2.5.6 Leap-frog Algorithm ...................... 14
      2.5.7 Velocity-Verlet Algorithm ............... 14
      2.5.8 Liouville Formulation of the velocity-verlet algorithm 15
   2.6 Langevin Dynamics ................................ 16

3 Dissipative particle dynamics ....................... 18
   3.1 Introduction ................................ 18
   3.2 Previous works on DPD ........................ 20
   3.3 Dissipative Particle Dynamics ................ 22
   3.4 Stochastic Differential Equations and Fokker-Planck Equations for DPD 23
   3.5 Energy conserving DPD ........................ 25
   3.6 Deriving the Navier-Stokes equation from DPD 27
4 Simulation process 31
  4.1 Introduction .................................................. 31
  4.2 Particles ......................................................... 31
  4.3 Interactions ..................................................... 33
  4.4 Force Calculation ............................................... 34
    4.4.1 Design of binned code ..................................... 34
  4.5 Integration ..................................................... 38
  4.6 Parallelization ................................................ 39
  4.7 Wall simulation ............................................... 42

5 Hybrid particle continuum model 44
  5.1 Introduction ................................................... 44
  5.2 Overview of hybrid coupling ................................. 45
  5.3 Continuum to particle coupling .............................. 48
    5.3.1 Imposing fluxes ........................................... 48
    5.3.2 Momentum ................................................ 49
    5.3.3 Energy .................................................. 50
  5.4 Particle to continuum coupling ............................. 50

6 Results 51
  6.1 Flow in a channel .............................................. 51
    6.1.1 Parameter study .......................................... 56
    6.1.2 Impinging jet ............................................. 58
  6.2 Flow around a cylinder ....................................... 60
  6.3 Flow on a turbulator .......................................... 65
  6.4 Transient simulation ......................................... 72
  6.5 Conclusion .................................................... 75
## List of Figures

3.1  Length and time scale of different modeling technics[37] ............................................ 19  
4.1  The binned domain. Only particles in the neighbouring bins are considered for force calculation ........................................ 35  
4.2  The speed increase of GPU over serial implementation ............................................. 40  
4.3  The wall is modeled using the same colliding particle with fixed particles on the wall ............................................. 43  
5.1  Spatial decoposition of coupled domain (3D). .............................................................. 46  
5.2  Spatial decomposition in hybrid scheme. ................................................................. 47  
6.1  Schematic of flow in a channel ..................................................................................... 51  
6.2  A sample of particles on the domain colored by temperature (T) ................................ 52  
6.3  Contour plot of flow in a channel for velocity (m/s) .................................................... 53  
6.4  Contour plot of flow in a channel for temperature (T) ................................................ 53  
6.5  Velocity over mean velocity for flow in a channel. The error bars show the maximum and minimum errors for all the runs ................................................................. 54  
6.6  Temperature for flow in a channel. The error bars show the maximum and minimum errors for all the run ................................................................. 55  
6.7  One way coupling of continuum to particle model ..................................................... 56  
6.8  Effect of parameter of conservative force on flow pattern ......................................... 57  
6.9  Effect of paramter of dissipative force on the velcity ................................................. 57  
6.10 Effect of relative particle size on the flow pattern ....................................................... 58  
6.11 CFD model of the impinging jet with sample boxes showing where the particle code is implanted ........................................................................................................ 59  
6.12 The fitted data for $L/D = 2$, dots are result of the DPD and blue line is the experimental data ........................................................................................................ 59  
6.13 The fitted data for $L/D = 6$, dots are result of the DPD and blue is the exp. data ................................................................. 60  
6.14 CFD modeling of flow around a cylinder p(Pa) .......................................................... 60  
6.15 CFD modeling of flow around a cylinder T(K) .......................................................... 61  
6.16 CFD modeling of flow around a cylinder v(m/s) ........................................................ 61  
6.17 Comparison of model and experiment $Re = 34000$ ............................................... 62  
6.18 Comparison of model and experiment $Re = 106000$ ............................................... 62
6.19 The simulated CFD data of Ames experiment (Pressure(Pa)) .................. 63
6.20 The simulated CFD data of Ames experiment (Temperature(K)) ............ 63
6.21 The simulated CFD data of Ames experiment (Velocity(m/s)) .................. 63
6.22 The pressure drop result from CFD comparing to pressure drop of Ames’ experiment 1st row .......................................................... 64
6.23 The heat transfer data of particle model for Ames experiment comparing to Ames’ experiment 1st row .............................................. 64
6.24 The pressure drop result from CFD comparing to pressure drop of Ames’ experiment 2nd row .......................................................... 65
6.25 The heat transfer data of particle model for Ames experiment comparing to Ames’ experiment 2nd row .............................................. 66
6.26 The grid used in CFD model. ............................................................... 66
6.27 The cross section of the grid used in CFD model. ................................... 67
6.28 CFD modeling of flow in a channel with turbulator v(m/s) ...................... 67
6.29 Comparison of the Nu between particle and CFD and experiment in channel with turbulator ......................................................... 68
6.30 A snapshot of the particles over the turbulator ....................................... 69
6.31 $v_x$ for the CFD model in the plane in the middle of channel .................... 69
6.32 $v_x$ for the particle model in the plane in the middle of channel ................ 70
6.33 $v$ for the CFD model in the plane in the middle of channel ...................... 70
6.34 $v$ for the particle model in the plane in the middle of channel ................... 71
6.35 Comparison of the Nu between particle and CFD and experiment in channel with turbulator ......................................................... 71
6.36 Sample picture of the test case (Flow in a channel) with magnified particle model 72
6.37 The speed of one time step in particle model on GPU ............................. 73
6.38 The weak scaling of the CFD problem for one time step ......................... 74
6.39 The strong scaling of the CFD problem for one time step ....................... 74
List of Tables
I would like to express my deepest appreciation to my advisor Professor Tarek Zohdi. He welcomed me into his lab and provided me with the guidance I required.

I would like to thank my committee members, Professor James Demmel, Professor Reza Alam, and Professor Hayden Taylor for their feedbacks and advices during my course work and throughout my progress on this dissertation.

Finally, I gratefully acknowledge Siemens Corporation, the funding source that made this dissertation possible.
Chapter 1

Introduction

1.1 Multiscale modeling and lower scales

The dynamics of fluid close to interfaces is affected by its microscopic structure. Because the molecular level dynamics is playing a role, and macroscopic scale is affected by lower levels, in these situations the fluid cannot be completely described by continuum theory and Navier-Stokes equation. By increasing the computational power, the need for getting higher accuracy models can be satisfied by going to smaller dimension scales. However, molecular dynamics also is not useful in the practical point of view. The point is the number of particle in the large scales are too high that the computation time could take years with current computational power. For the fact that there are multiscale effects in these flow situations none of the molecular dynamics or continuum models work in this situation, so we need a middle ground to solve the dilemma. Moreover, we need a technique to couple both higher and lower scales to interchange the data from higher scale to lower scale and the average lower scale data to the higher scale.

Molecular dynamics (MD) is being used to calculate physical movement of atoms or molecules by calculating the binary collision between the particles and integrating the Newton’s equation of motion for the system of particles. The method was developed in theoretical physics in the late 1950s [2], and early 1960s [32], and is applied today mostly in chemical physics, materials science and the modeling of biomolecules. Molecular systems consist of a large number of particles, and it is impossible to find the properties of such complex systems analytically, so with use of the molecular dynamics this problem can be solved numerically. However, long MD simulations are mathematically ill-conditioned, and have the property of generating cumulative errors in numerical integration. This problem can be minimized by properly selecting algorithms and parameters, but it cannot eliminated entirely. Although computer simulation power has astronomically increased since the beginning of the simulation by computers, and still is increasing, machine performance is still a limiting factor. This primarily restricts the size of the system that can be simulated, for example in the case of molecular dynamics the number of particles that can be handled with the computer, and the number of timesteps that can be calculated during the simulation is part
1.1. MULTISCALE MODELING AND LOWER SCALES

of this restriction. Besides, to capture an important phenomena in the macroscale level, one needs much larger simulation time and extremely large number of particles, but in a conventional MD simulation, a great deal of computing time is used for uninteresting individual particle behavior. As a consequence, there still are problems for which a simulation turns out to be inefficient or even intractable. Fortunately, there exist strategies to overcome these limitations. They aim at reducing the number of degrees of freedom by representing the system through a set of relevant variables that evolve on a more suitable time scale. The general procedure to remove degrees of freedom is called coarse-graining. One approach is known as Brownian dynamics and is based on the Langevin equations. This method is like molecular dynamics but the forces have a stochastic part that models a fluctuating environment. There are also other computationally effective methods like lattice-based algorithms, many of which are based on cellular automata. These methods, such as the lattice gas (LG) or lattice Boltzmann (LB) approach, rely on a spatial discretization of the system. They are quite efficient but also introduce some conceptual difficulties like implementing boundary conditions.

In this thesis, I study Dissipative Particle Dynamics (DPD), a method invented for carrying out particle based simulations of hydrodynamic behavior. DPD combines ideas from Langevin dynamics and lattice gas algorithms. In this study, we tried to use dissipative particle dynamics to address the problem of calculating heat transfer on some applications. In order to do that, the DPD method is used in order to calculate the heat transfer in small scales. Moreover in order to deal with larger scale flow situations a method is developed to couple different scales. Therefore, in combination to the small scale particle model, the continuum model is used to get the higher scale behavior of the flow. In this paper, in chapter 1 we introduce the basic theories of molecular dynamics. Then in chapter 2 we focus on dissipative particle dynamics, and bring some theories in the field. Afterwards, in chapter 3 the simulation technique is discussed. In chapter 4 we discuss the coupling technique between the particle model and the continuum model. In chapter 5 some results of the model is shown, and it is compared to some experimental and theoretical data.
Chapter 2

Some basic theories in Molecular Dynamics

2.1 Introduction

Molecular dynamics (MD) is a N-body computer simulation of physical movements of atoms and molecules. The atoms and molecules are allowed to interact for a period of time, giving a view of the motion of the atoms. In the most common version, the trajectories of atoms and molecules are determined by numerically solving Newton’s equations of motion for a system of interacting particles, where forces between the particles and potential energy are defined by interatomic potentials or molecular mechanics force fields. The method was originally conceived within theoretical physics in the late 1950s [2] but is applied today mostly in chemical physics, materials science and the modelling of biomolecules. In this chapter, we review the general concepts of molecular dynamics from classical mechanics to simulation techniques.

2.2 Classical Mechanics

In the case of the present work the word classical means the motion of the particle obeys the laws of classical mechanics. This means that the velocity of particles are small in comparison to the speed of light, and the vibrational frequency $\nu$ satisfies $h\nu < k_B T$. This condition is an excellent approximation for a large variety of materials[15]. As described by Goldstein [18], the time evolution of a system of particles in Hamiltonian classical mechanics is described by Lagrangian equation of motion.

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = 0 \quad (2.1)$$

where $q_i$ is the generalize location, $\dot{q}_i$ is the velocity and $L$ is the Lagrangian, which is a function of $(q, \dot{q}_i)$. $L$ can be written in terms of kinetic and potential energy, $K$ and $V$
respectively, as

\[ L = K - V. \]  \hfill (2.2)

It worth noting that, in Cartesian coordinates, the location vector is \( \mathbf{r} \), and the kinetic energy becomes

\[ K = \frac{1}{2} \sum_i m_i \mathbf{r}_i^2, \]  \hfill (2.3)

and the potential energy can be used to calculate the force on the particles as

\[ \mathbf{f}_i = \nabla_{\mathbf{r}_i} L = -\nabla_{\mathbf{r}_i} V. \]  \hfill (2.4)

In this case by substituting the above parameters into the Lagrange equation 2.1, it becomes

\[ m_i \ddot{\mathbf{r}}_i - \mathbf{f}_i = 0, \]  \hfill (2.5)

which is the Newton’s equation of motion. In the same way, generalized momentum, \( p_i \), is defined as

\[ p_i = \frac{\partial L}{\partial \dot{q}_i}. \]  \hfill (2.6)

The Hamiltonian also is defined as

\[ H = \sum_i \dot{q}_i p_i - L \]  \hfill (2.7)

The Hamiltonian equation of motion are [18]

\[ \dot{q}_i = \frac{\partial H}{\partial p_i}, \]
\[ \dot{p}_i = -\frac{\partial H}{\partial q_i}, \]  \hfill (2.8)

and subsequently for the cartesian coordinates

\[ \dot{\mathbf{r}}_i = \frac{\mathbf{P}_i}{m_i}, \]
\[ \dot{\mathbf{p}}_i = \mathbf{f}_i \]  \hfill (2.9)

### 2.2.1 Liouville Equation

The phase-space \( \Gamma = \{ \mathbf{r}_i, p_i \} \), in classical mechanics, is the space of all possible states of a physical system. The Liouville equation describes the time evolution of the phase-space distribution function. A trajectory in this phase-space, \( \Gamma(t) \), and the density distribution in phase-space is denoted by \( \rho(\Gamma, t) \). Consider a Hamiltonian dynamical system with canonical coordinates \( q_i \) and conjugate momenta \( p_i \), where \( i = 1, \ldots, n \). Then the phase space distribution \( \rho(p, q) \) determines the probability \( \rho(p, q) d^n q d^n p \) that the system will be found in the
infinitesimal phase space volume $d^nq, d^n p$. The Liouville equation governs the evolution of $\rho(p, q; t)$ in time $t$.

$$\frac{d\rho(\Gamma; t)}{dt} = \frac{\partial \rho(\Gamma; t)}{\partial t} + \sum_{i=1}^{n} \left( \frac{\partial \rho(\Gamma; t)}{\partial q_i} \dot{q}_i + \frac{\partial \rho(\Gamma; t)}{\partial p_i} \dot{p}_i \right) = 0 \quad (2.10)$$

In general for an arbitrary function $A(\Gamma)$ in phase-space, the time evolution can be derived from the equation of motion (2.9).

$$\frac{dA}{dt} = \sum_{i=1}^{n} \left( \frac{\partial A}{\partial q_i} \dot{q}_i + \frac{\partial A}{\partial p_i} \dot{p}_i \right) = i\mathcal{L}A(\Gamma), \quad (2.11)$$

where $\mathcal{L}$ is the Liouville operator

$$i\mathcal{L} = \sum_{i=1}^{n} \left( \frac{\partial}{\partial q_i} \dot{q}_i + \frac{\partial}{\partial p_i} \dot{p}_i \right) \quad (2.12)$$

and in cartesian coordinates

$$\frac{dA}{dt} = \sum_{i=1}^{n} \left( \frac{\partial A}{\partial r_i} \dot{r}_i + \frac{\partial A}{\partial p_i} \dot{p}_i \right) = i\mathcal{L}A(\Gamma), \quad (2.13)$$

and the Liouville operator

$$i\mathcal{L} = \sum_{i=1}^{n} \left( \frac{\partial}{\partial r_i} \dot{r}_i + \frac{\partial}{\partial p_i} \dot{p}_i \right), \quad (2.14)$$

From the equation of motion (2.8), one can derive Liouville’s theorem

$$\nabla_{\Gamma} \dot{\Gamma} = 0, \quad (2.15)$$

which in turn can result in the continuity equation for the phase-space

$$\frac{\partial \rho}{\partial t} + \nabla_{\Gamma} (\rho \dot{\Gamma}) = 0. \quad (2.16)$$

As a consequence, the phase-space density along a trajectory is conserved

$$\frac{d}{dt} \rho(\Gamma; t) = 0 \quad (2.17)$$

and finally results in Liouville’s equation (2.10), which can be written as

$$\frac{\partial}{\partial t} \rho(\Gamma; t) + i\mathcal{L}\rho(\Gamma; t).$$

(2.18)
2.3 Statistical Mechanics

In regular systems there are multiple molecules and atoms. It is obvious that calculating the state of a system by every particle is impossible by current computational units that we have. We also better understand systems by macroscopically measurable quantities like temperature, pressure, and other thermodynamic properties. Statistical mechanics were developed to answer this problem by using statistics on small particles. Statistical mechanics explains the thermodynamic on a molecular level by molecular movements [35]. In this section, the concept of statistical mechanics and its relation to thermodynamical properties are briefly described.

A system is described by the positions and momenta of all particles in the phase space $\Gamma = (r^N, p^N)$. These microscopic degrees of freedom change based on the equations of classical mechanics. However, as it is stated before, we are not interested in microscopic values, the microscopic state of the system is not important, as long as it delivers the same macroscopic state. These microscopic set which gives a same macroscopic state is called the ensemble. The distribution of the systems in phase space is given by the phase space density $\rho(\Gamma; t)$. "Usually, one makes the ergodic assumption, which states that the system will come arbitrarily close to any point in the accessible phase space, i.e. almost every trajectory will cover almost the entire accessible phase space. If a system is ergodic, time averages can be replaced by ensemble averages" [35] as

$$\lim_{T \to \infty} \int_0^T A(\Gamma) dt = \int_{\Omega} \rho(\Gamma) A(\Gamma) d\Gamma.$$  \hfill (2.19)

in which $A(\Gamma)$ is an arbitrary macroscopic quantity, so the ensemble average of an observable becomes

$$\langle A \rangle = \int_{\Omega} \rho(\Gamma) A(\Gamma) d\Gamma$$  \hfill (2.20)

The partition function, which describes the statistical properties of a system in thermodynamic equilibrium in the canonical ensemble is

$$Z_K = \frac{1}{N! \hbar^{3N}} \int_{\Omega} e^{-B H(\Gamma)} d\Gamma$$  \hfill (2.21)

where $H(\Gamma)$ is the Hamiltonian of the system, $\hbar$ is a constant to make $Z$ dimensionless, and the inverse temperature $\beta$ is $\beta = 1/k_B T$, in which $k_B$ is the Boltzmann constant. With the partition function we can write the average of equation 2.20.

$$Z_K = \frac{1}{Z_K} \int_{\Omega} e^{-B H(\Gamma)} d\Gamma$$  \hfill (2.22)

The thermodynamic potentials for different ensembles are related to each other by Legendre transformations [35]. The canonical ensemble, for example, is described by the Helmholtz free energy $F$ as

$$F(T, V, N) = \frac{1}{\beta} \ln Z_K(T, V, N)$$  \hfill (2.23)
It is a function of the temperature $T$, the volume $V$ and the number of particles $N$. The differential of the free energy is

$$dF = -SdT - PdV + \mu dN$$  \hspace{1cm} (2.24)

From this follows the macroscopic quantities.

$$\mu = - \left( \frac{\partial F}{\partial N} \right)_{T,V}$$

$$P = - \left( \frac{\partial F}{\partial V} \right)_{T,N}$$

$$S = - \left( \frac{\partial F}{\partial T} \right)_{V,N}$$  \hspace{1cm} (2.25)

where $S$ is the entropy, $P$ is the pressure, and $\mu$ is the chemical potential. We can derive the thermodynamic coefficients, for example the coefficient of thermal expansion, specific heat, and the isothermal compressibility respectively as follows

$$\alpha = \frac{1}{V} \left( \frac{\partial V}{\partial T} \right)_{P,N}$$

$$c_v = \frac{T}{N} \left( \frac{\partial S}{\partial T} \right)_{V,N}$$

$$\kappa_T = - \frac{1}{V} \left( \frac{\partial V}{\partial P} \right)_{T,N}$$  \hspace{1cm} (2.26)

The thermodynamic coefficients can be related to fluctuations in the extensive variables. The specific heat is for example given by the fluctuations in the total energy

$$k_B T^2 c_v = \langle E^2 \rangle - \langle E \rangle^2 = \langle (\Delta E)^2 \rangle.$$  \hspace{1cm} (2.27)

The fluctuations of a quantity are also related to the response of the quantity to an external field. Consider a quantity $A$ that is coupled to an external field. The susceptibility $\xi_A$ is then given by

$$\xi_A = - \beta \left( \langle A^2 \rangle - \langle A \rangle^2 \right) = \langle (\Delta A)^2 \rangle$$  \hspace{1cm} (2.28)

Equation 2.28 is called a linear response theorem. Linear response theory will be useful to derive expressions for transport coefficients.

### 2.4 Fluid Dynamics and Continuum Mechanics

The materials that we are dealing with in our daily life are mostly consists of a huge number atoms and molecules. Therefore, it is unlikely that we could describe these system
of materials with the characteristics of each molecule and atom, and deriving its behaviour via Newton’s equation. Besides, we are usually interested in measurable macroscopic values, like temperature and pressure, which appear on larger scales. This situation leads to continuum theory and fluid dynamics. The Fluid dynamics is the theory of motion of liquids and gases. In this section, I briefly describe the continuum mechanical description, and I give an introduction to the kinetic theory of fluids.

As stated in the previous paragraph, since the systems we are dealing with in many situations possess many particles, and molecular scale are extremely below the relevant scale for fluid dynamics systems, a fluid can be viewed as a continuum. In this continuum picture, the fluid consists of small volumes, so-called fluid elements or fluid particles, which still contain many atoms or molecules. These elements are considered to move as point-like entities through the fluid and determine the state of the fluid. This state is described by the density $\rho(r, t)$, the velocity $v(r, t)$, and the pressure $p(r, t)$ in each point $r$ at time $t$. In this manuscript we introduce some basic equations of continuum mechanics. However, for further detail calculation, one can check Landau and Lifshitz [27].

The first basic principle in fluid dynamics is conservation of mass. From the conservation of mass follows the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot j = 0 \tag{2.29}$$

where $j = \rho v$ is the mass flux density. The conservation of momentum yields the motion equation for a fluid element subject to the pressure $p$ from the surrounding fluid elements, described by Euler’s equation.

$$\frac{\partial v_i}{\partial t} + (v \nabla) v = -\frac{1}{\rho} \nabla p \tag{2.30}$$

As it can be seen in the equation 2.30, this equation does not yet contain any viscous effects and no heat transport. It is therefore only valid for ideal fluids, for which heat transport and viscous effects can be neglected. The pressure $p = p(\epsilon, \rho)$ is a function of the internal energy $\epsilon$ and the density $\rho$ and is assumed to satisfy a local equilibrium assumption. The pressure tensor $\sigma$ takes the form

$$\sigma_{ij} = p \delta_{ij}. \tag{2.31}$$

Using the pressure tensor, equation 2.30 can be rewritten as

$$\left( \frac{\partial v_i}{\partial t} + \Sigma_j v_j \frac{\partial v_i}{\partial x_j} \right) = -\frac{1}{\rho} \Sigma_j \frac{\partial \sigma_{ij}}{\partial x_j}. \tag{2.32}$$

To incorporate friction and heat transport, we substitute equation 2.31 with the stress tensor for a viscous fluid. The stress tensor can be written in the general form [27]

$$\sigma_{ij} = p \delta_{ij} + \sigma'_{ij}. \tag{2.33}$$
with
\[
\sigma'_{ij} = \eta \left( \frac{\partial v_i}{\partial t} + \frac{\partial v_i}{\partial t} - \frac{2}{3} \sigma_{ij} \sum_k \frac{\partial v_k}{\partial x_k} \right) + \zeta \delta_{ij} \sum_k \frac{\partial v_k}{\partial x_k} \quad (2.34)
\]

Here, \( \eta \) is the shear viscosity and \( \zeta \) is the bulk viscosity, both of which are functions of the pressure and the temperature and may vary through the liquid. Introducing this pressure tensor into the right-hand side of Euler’s equation we get the general motion equation for a viscous fluid
\[
\rho \left( \frac{\partial v_i}{\partial t} + v_i \frac{\partial v_i}{\partial x_j} \right) = -\frac{\partial p}{\partial x_i} + \frac{\partial \sigma'_{ij}}{\partial x_j} \quad (2.35)
\]

If the variation of the viscosities in the fluid can be neglected, the motion equation can be brought into vectorial form
\[
\rho \left( \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \nabla) \mathbf{v} \right) = -\nabla p + \eta \Delta \mathbf{v} + \left( \zeta + \frac{\eta}{3} \right) \text{grad div} \mathbf{v} \quad (2.36)
\]

In an incompressible fluid, \( \text{div} (\mathbf{v}) = 0 \), and the last term of 2.36 vanishes. We finally arrive at the Navier-Stokes equation
\[
\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \nabla) \mathbf{v} = -\nabla p + \nu \Delta \mathbf{v} \quad (2.37)
\]

where \( \nu \) is the kinematic viscosity
\[
\nu = \frac{\eta}{\rho} \quad (2.38)
\]

The stress tensor for an incompressible fluid has the simple form
\[
\sigma_{ik} = -p \delta_{ik} + \eta \left( \frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} \right) \quad (2.39)
\]

which contains the shear viscosity \( \eta \). The solutions of the Navier-Stokes equation 2.37 have to be determined with respect to initial and boundary conditions. The flow of a viscous fluid can be very complex, ranging from laminar flow to turbulent flow.

2.5 Kinetic Theory

Now that we familiarize ourselves with the concepts of microscopic behavior of molecules and macroscopic description of continuum mechanics, we need to find the relation between two concepts. One of the earliest descriptions of this relation is the kinetic theory. The main idea in kinetic theory is to replace a huge number of objects, whose physical states are completely described by points in a certain phase space, and whose properties are otherwise identical, by a statistical distribution over that phase space. In particular, a large crowd of classical point particles will be described by a statistical distribution on the space of positions and velocities.
The kinetic theory describes a gas as a large number of randomly moving microscopic particles. The rapidly moving particles constantly collide with each other and with the walls of the container. Kinetic theory explains macroscopic properties of gases, such as pressure, temperature, viscosity, thermal conductivity, and volume, by considering their molecular composition and motion. Therefore, kinetic theory provides the link between the microscopic dynamics of Newton’s equations and the description of the macroscopic properties in continuum mechanics [38]. The kinetic theory was first developed by Boltzmann for dilute gases. Boltzmann’s theory describes the statistical properties of the fluid via distribution functions. The number of particles in a volume element $dr$ around $r$ that have a velocity in the volume $dv$ around $v$ is at time $t$ given by

$$f^{(1)}(r, v; t)drdv,$$  \hspace{1cm} (2.40)

were $f^{(1)}(r, v; t)$ is the one particle distribution function. The macroscopic quantities can be derived from this distribution function as average values: The local number density

$$n(r; t) = \int f^{(1)}(r, v; t)dvdv,$$  \hspace{1cm} (2.41)

the local velocity

$$u(r; t) = \frac{1}{n(r; t)} \int f^{(1)}(r, v; t)vdv,$$  \hspace{1cm} (2.42)

or in general

$$A(r; t) = \frac{1}{n(r; t)} \int A(v)f^{(1)}(r, v; t)vdv,$$  \hspace{1cm} (2.43)

The distribution in the whole phase space of the system is given by the $N$-particle distribution function $f^{(N)}(r^N, p^N; t)$, which resembles the phase space density $\rho(\Gamma; t)$. The motion equation for $f^{(N)}$ is the Liouville equation.

### 2.5.1 Bogoliubov-Born-Green-Kirkwood-Yvon (BBGKY) hierarchy

The BBGKY hierarchy is a set of equations describing the dynamics of a system of a large number of interacting particles. The equation for an $n$-particle distribution function (probability density function) in the BBGKY hierarchy includes the $(n + 1)$-particle distribution function thus forming a coupled chain of equations. However, if we are only interested in simple observables, we do not need the full phase-space density. Therefore, we define reduced distribution functions by integrating out a number of degrees of freedom

$$f^{(n)}(r^n, v^n; t) = \frac{N!}{(N-n)!} \int \int f^{(N)}(r^N, v^N; t)dr^{(N-n)}dv^{(N-n)}$$  \hspace{1cm} (2.44)
2.5. KINETIC THEORY

For \( n = 2 \) we get the pair distribution function \( f^{(2)}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{v}_1, \mathbf{v}_2; t) \). The Liouville equation yields a relation for the reduced distribution function

\[
\left( \frac{\partial}{\partial t} + \sum_{i=1}^{n} \left( \mathbf{v}_i \cdot \frac{\partial}{\partial \mathbf{r}_i} + \frac{\mathbf{X}_i}{m} \cdot \frac{\partial}{\partial \mathbf{v}_i} \right) \right) f^{(n)}(\mathbf{r}^n, \mathbf{v}^n; t) = -\sum_{i=1}^{n} \int \int \frac{\mathbf{F}_{i,n}^{n+1}}{m} \cdot \frac{\partial}{\partial \mathbf{v}_i} f^{(n+1)}(\mathbf{r}^{n+1}, \mathbf{v}^{n+1}; t) d\mathbf{r}_{n+1} d\mathbf{v}_{n+1},
\]

(2.45)

where \( \mathbf{X}_i \) is the external force on particle \( i \), and \( \mathbf{F}_{ij} \) is the pair force between particles \( i \) and \( j \). For \( n = 1, 2, \cdots \) this is the BBGKY hierarchy. It relates the one particle distribution function to the pair distribution function, which in turn is related to the three particle distribution function, and so on.

2.5.2 Boltzmann Equation

Setting \( n = 1 \) in equation 2.45 we get the famous Boltzmann equation

\[
\frac{\partial f^{(1)}}{\partial t} + \mathbf{v} \cdot \nabla_r f^{(1)} + \frac{\mathbf{F}}{m} \cdot \nabla_v f^{(1)} = \left( \frac{\partial f^{(1)}}{\partial t} \right)_{\text{coll}}
\]

(2.46)

Boltzmann made the two fundamental assumptions to obtain a closed form for this equation. First, only binary collisions between the particles take place. Second, The collisions are uncorrelated. Both of these assumptions are very well justified for dilute gases, and Boltzmann’s description is also very successful for many other cases. Boltzmann's assumptions lead to the molecular chaos assumption [35]

\[
f^{(2)}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{v}_1, \mathbf{v}_2; t) = f^{(1)}(\mathbf{r}_1, \mathbf{v}_1; t)f^{(1)}(\mathbf{r}_2, \mathbf{v}_2; t).
\]

(2.47)

The collision term on the right hand side of equation 2.46, which basically describes scattering collisions, then becomes

\[
\left( \frac{\partial f^{(1)}}{\partial t} \right)_{\text{coll}} = \int d\mathbf{v}_2 \int \sigma(\theta, v_{12}) v_{12} \left( f^{(1)}(\mathbf{r}, \mathbf{v}_1; t)f^{(1)}(\mathbf{r}, \mathbf{v}_2'; t) - f^{(1)}(\mathbf{r}, \mathbf{v}_1; t)f^{(1)}(\mathbf{r}, \mathbf{v}_2; t) \right) d\Omega
\]

(2.48)

Here, \( \sigma(\theta, v_{12}) \) is the scattering cross section and \( v_{12} = ||\mathbf{v}_1 - \mathbf{v}_2|| \). With this expression, the Boltzmann equation yields a closed equation for the one particle distribution function. The Boltzmann equation is a nonlinear equation which is in general complicated to solve. In most situations, one is only interested in the linear response behavior of the system, and the deviations from equilibrium are assumed to be small. The one-particle distribution function can then be written as

\[
f^{(1)}(\mathbf{r}, \mathbf{v}; t) = f_0(v) + \delta f(\mathbf{r}, \mathbf{v}; t) = f_0(v)(1 + \phi(\mathbf{r}, \mathbf{v}; t))
\]

(2.49)

where \( \phi(\mathbf{r}, \mathbf{v}; t) \) describes a small perturbation \( \delta f \) of the equilibrium state

\[
\phi(\mathbf{r}, \mathbf{v}; t) = \frac{\delta f(\mathbf{r}, \mathbf{v}; t)}{f_0(v)}
\]

(2.50)
With these definitions, the Boltzmann equation reduces to the linear Boltzmann equation in the absence of external forces $F = 0$

$$\frac{\partial \delta f}{\partial t} + v \cdot \nabla \delta f = C\delta f$$ (2.51)

where the collision operator $C$ is given by

$$C\delta f(r, v_1; t) = \int dv_2 \int d\sigma(\theta, v_{12}) v_{12} f_0(v_1) f_0(v_2) (\phi(r, v_1'; t) + \phi(r, v_2'; t) - \phi(r, v_1; t) - \phi(r, v_2; t)) d\Omega$$ (2.52)

Terms of order $(\delta f)^2$ have been neglected. In a spatially uniform system, the solution of the linear Boltzmann equation can be written as

$$\delta f(v; t) = \sum_i c_i \phi_i(v) e^{\lambda_i t} = \sum_{\alpha=1}^{5} c_\alpha \phi_\alpha(v)$$ (2.53)

This equation shows that the new equilibrium state is completely determined by the collision invariants of the system.

### 2.5.3 Molecular Dynamics

Molecular Dynamics is a computer simulation technique, in which the physical movement of the microscale particles are calculated over time by numerically integrating Newton’s equations of motion 2.9. Newton’s equations might be seem easy to integrate, but when it comes to many particles moving for many timesteps, there could be large error propagation and stability problem in the simulation. An algorithm for this purpose should meet several requirements [3]

- It should produce results as close as possible to the exact state
- It should satisfy symmetry of the system and the conservation laws
- It should be computationally efficient (fast execution, low memory requirements)
- It should be easy to implement

The integration of the Newton’s equation is done by series expansion. It is obvious that the more terms you keep in the expansion the more accurate the calculation becomes. Nevertheles, using more terms means that we are using more computation in each timestep, which result in slower program runtime. Time reversibility is another important aspect of the Newton’s equation. Therefore, any algorithm for solving the molecular dynamics must be time reversible.

Another extremely important aspect of an algorithm is stability. That is starting with the same initial condition could end in totally different answers. However, since the average behavior is important for us, this is not a huge problem as long as the conservation laws
are fullfilled. Especially, preserving energy is very important in any molecular dynamics algorithms. Since the most time in the molecular dynamics is used to calculate the inter-molecular forces, the integration speed is not of very high importance. However, if we could use a large timestep, the speed of the simulation will increase accordingly. Moreover, there are algorithms that produce the correct equilibrium distribution only in the limit of infinitely small time step. In practice, the time step therefore represents a trade off between speed and stability of the simulation.

Molecular Dynamics algorithms basically fall into two classes: Verlet-like algorithms and Gear predictor-corrector algorithms [3]. While in the predictor-corrector algorithms also higher moments are calculated, Verlet-like algorithms only use the positions $r$, the velocities $v$ and the forces $f$ to determine their respective new values. Taylor expansion can be used to expand the equation of motion as

$$r_i(t + \Delta t) = r_i(t) + \Delta t v_i(t) + \frac{\Delta t^2}{2} f_i(t) + \frac{\Delta t^3}{3} \ddot{r}_i(t) + O(\Delta t^4).$$

$$v_i(t + \Delta t) = v_i(t) + \frac{\Delta t}{m_i} f_i(t) + \frac{\Delta t^2}{2} \ddot{v}_i(t) + \frac{\Delta t^3}{3!} \dddot{v}_i(t) + O(\Delta t^4).$$

### 2.5.4 Euler Algorithm

One of the most simple integration scheme based on equations 2.54 is realized by the Euler algorithm. The positions and velocities are updated according to

$$r_i(t + \Delta t) = r_i(t) + \Delta t v_i(t) + \frac{\Delta t^2}{2} f_i(t) + O(\Delta t^3)$$

$$v_i(t + \Delta t) = v_i(t) + \frac{\Delta t}{m_i} f_i(t) + O(\Delta t^2)$$

Euler algorithm is not time reversible, and not phase-space preserving, so in the long term it is not necessarily stable [15], and in the short-term it has the stability of the order $\Delta t$. The Euler algorithm is therefore not an ideal way to integrate the equation of motion.

### 2.5.5 Verlet Algorithm

The Verlet algorithm solves the second order system 2.9 based on the current positions $r_i(t)$ and forces $f_i(t)$ and the previous positions $r_i(t - \Delta t)$. We can derive it by Taylor expansion for $r_i(t - \Delta t)$ similar to 2.54

$$r_i(t - \Delta t) = r_i(t) - \Delta t v_i(t) + \frac{\Delta t^2}{2m_i} f_i(t) - \frac{\Delta t^3}{3!} \dddot{r}_i(t) + O(\Delta t^4).$$
By adding first part of 2.54 and 2.56, and by subtracting them, one can get respectively

\[
\begin{align*}
\mathbf{r}_i(t + \Delta t) &= 2\mathbf{r}_i(t) - \mathbf{r}(t - \Delta t) + \frac{\Delta t^2}{m_i} \mathbf{f}_i(t) + O(\Delta t^3) \\
\mathbf{v}_i(t) &= \frac{\mathbf{r}_i(t + \Delta t) - \mathbf{r}_i(t - \Delta t)}{2\Delta t} + O(\Delta t^2).
\end{align*}
\] (2.57)

As it can be seen the velocities are actually not needed to compute the update. However, they are eventually needed for calculating observables like the kinetic energy. However, in the Verlet scheme the velocities \( \mathbf{v}(t) \) are only available once \( \mathbf{r}(t + \Delta t) \) has been calculated, i.e. one time step later.

### 2.5.6 Leap-frog Algorithm

Another way of the integration is using leap-frog algorithm[15]. The updating equations are

\[
\begin{align*}
\mathbf{v}_i(t + \frac{\Delta t}{2}) &= \mathbf{v}_i(t - \frac{\Delta t}{2}) + \frac{\Delta t}{m_i} \mathbf{f}_i(t), \\
\mathbf{r}_i(t + \Delta t) &= \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t + \frac{\Delta t}{2}).
\end{align*}
\] (2.58)

Because velocities are evaluated at half time steps, they calculated first, and then the positions. The current velocities updated from

\[
\mathbf{v}_i(t) = \frac{\mathbf{v}_i(t - \frac{\Delta t}{2}) + \mathbf{v}_i(\Delta t + \frac{\Delta t}{2})}{2}
\] (2.59)

The drawback is that the velocities are still not accessible directly from the scheme and it needs separate calculation.

### 2.5.7 Velocity-Verlet Algorithm

To solve the problem of having both velocity and positions and the same time we can use the Velocity-Verlet scheme. The positions and velocities are updated according to

\[
\begin{align*}
\mathbf{r}_i(t + \Delta t) &= \mathbf{r}_i(t) + \Delta(t) \mathbf{v}_i(t) + \frac{\Delta t^2}{m_i} \mathbf{f}_i(t) + O(\Delta t^3) \\
\mathbf{v}_i(t + \Delta t) &= \mathbf{v}_i(t) + \frac{\Delta t}{2m_i} (\mathbf{f}_i(t) + \mathbf{f}_i(t + \Delta t)) + O(\Delta t^3).
\end{align*}
\] (2.60)

The Velocity-Verlet scheme is algebraically equivalent to the original Verlet algorithm. Equations can be derived from 2.60 by elimination of the velocities in the position update. The Velocity-Verlet algorithm is very stable. The reason for the stability of the Velocity-Verlet algorithm is that it is reversible and it preserves the volume in phase-space. In the next section I describe the algorithm in more detail from [15].
2.5.8 Liouville Formulation of the velocity-verlet algorithm

As stated before, important conditions for an accurate and stable Molecular Dynamics algorithm are time reversibility and phase-space conservation. The time evolution of an arbitrary function of positions and momenta can be written as

\[
\frac{d}{dt} A(\Gamma) = i\mathcal{L}A(\Gamma),
\]

where \( \mathcal{L} \) is the Liouville operator defined in 2.10. By integrating it, we can get

\[
A(t) = e^{i\mathcal{L}t} A(0).
\]

We can separate it to a position part and a momentum part as

\[
i\mathcal{L} = i\mathcal{L}_r + i\mathcal{L}_p,
\]

where

\[
i\mathcal{L}_r = \sum_i \dot{r}_i \frac{\partial}{\partial r_i}
\]

and

\[
i\mathcal{L}_p = \sum_i \dot{p}_i \frac{\partial}{\partial p_i}.
\]

To use \( i\mathcal{L}_r \) and \( i\mathcal{L}_p \) in equation 2.62 we write the Trotter expansion of the Liouville operator [15]

\[
e^{i(\mathcal{L}_r + \mathcal{L}_p)\Delta t} = e^{i\mathcal{L}_p \frac{\Delta t}{2}} e^{i\mathcal{L}_r \Delta t} e^{i\mathcal{L}_p \frac{\Delta t}{2}} + O(\Delta t^3)
\]

By expanding, we get

\[
e^{i\mathcal{L}_p \frac{\Delta t}{2}} \mathbf{r}_i = \mathbf{r}_i
\]

\[
e^{i\mathcal{L}_r \Delta t} \mathbf{r}_i = \mathbf{r}_i + \Delta t \dot{\mathbf{r}}_i
\]

and

\[
e^{i\mathcal{L}_p \frac{\Delta t}{2}} \mathbf{p}_i = \mathbf{p}_i
\]

\[
e^{i\mathcal{L}_r \Delta t} \mathbf{p}_i = \mathbf{p}_i + \Delta t \dot{\mathbf{p}}_i
\]

so together

\[
e^{i\mathcal{L}_p \frac{\Delta t}{2}} e^{i\mathcal{L}_r \Delta t} e^{i\mathcal{L}_p \frac{\Delta t}{2}} \mathbf{r}_i = \mathbf{r}_i(t) + \Delta t \mathbf{\dot{r}}(\frac{\Delta t}{2})
\]

\[
= \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t) + \frac{\Delta t^2}{2m_i} \mathbf{f}_i(t)
\]

\[
e^{i\mathcal{L}_p \frac{\Delta t}{2}} e^{i\mathcal{L}_r \Delta t} e^{i\mathcal{L}_p \frac{\Delta t}{2}} \mathbf{p}_i = \mathbf{p}_i(t) + \left( \frac{\Delta t}{2} \right) (\mathbf{\dot{p}}_i(t) + \mathbf{p}_i(t + \Delta t))
\]

\[
= \mathbf{p}_i(t) + \left( \frac{\Delta t}{2} \right) (\mathbf{\dot{f}}_i(t) + \mathbf{\dot{f}}_i(t + \Delta t))
\]
which result in the Velocity-Verlet algorithm. The operators $L\tau$ and $Lp$ are hermitian, so $e^{i(L\tau+Lp)\Delta t} \approx e^{iLp\frac{\Delta t}{2}} e^{iL\tau \Delta t} e^{iLp\frac{\Delta t}{2}}$ is a unitary operator, which implies that the volume in phase-space is preserved. Time-reversibility is also satisfied because the equations are symmetric with respect to future and past coordinates. The Trotter expansion 2.66 is correct up to terms of order $\Delta t^3$. This is the reason for the very good long-term stability of the Velocity-Verlet algorithm. [15]

2.6 Langevin Dynamics

In the molecular dynamics all the atoms have to be considered in the calculation in each timestep, and each timestep also need to be considered in a manner that are atom collision are covered, so the timestep need to be very small. However, when we want to deal with larger timescales this approach is not appropriate anymore, because it take a long time to get to larger timescale or in many cases with the computational power currently in have it is impossible.

One solution to deal with different timescales is approximate techniques that reduce the degrees of freedom. One approach to do this are Langevin dynamics, where the fast degrees of freedom are modeled using the stochastic terms in the equation of motion. [10] The Langevin equation is

$$\dot{p}_i = F_C^i - \gamma \frac{p_i}{m_i} + \sigma_i.$$  \hfill (2.70)

$F_C^i$ is the conservative force, $\gamma$ is the friction coefficient and $\sigma_i$ is a stochastic force with mean of zero, and it is not correlated for different particles, and different times.

The different terms of the Langevin equation have physical interpretations. The term $\gamma \frac{p_i}{m_i}$ represents a dissipative force which is caused by the friction, and the stochastic force $\sigma_i$ is a kind of driving force which is caused by "kicks" from the surrounding particles [16].

To use the Langevin equation in a Molecular Dynamics simulation, one can easily just adjust the force by the dissipative and stochastic terms.

$$f_i(t) = F_C^i(t) + \gamma v_i(t) + \frac{1}{\sqrt{\Delta t}} \sigma_i(t)$$ \hfill (2.71)

where $z_i(t)$ is a vector of random numbers drawn at time step $t$. This force can then be used in a Molecular Dynamics integration scheme. The factor $\frac{1}{\sqrt{\Delta t}}$ in the equation is due to the discretization by interpreting the random force as a Wiener process [16]. The time integral of the random force and its discretized form is as the following [19],

$$\langle \left( \int_0^t \sigma(t') dt' \right)^2 \rangle = \langle \left( \sum_{k=1}^N \sigma_k \right)^2 \left( \frac{t}{N} \right)^2 \rangle = N\sigma^2 \frac{t^2}{N^2} = \sigma^2 t \Delta t.$$  \hfill (2.72)

"The integral represents the mean diffusion in the physical process and should hence be independent of the step size $\Delta t$. Consequently we have to substitute the amplitude of the
random force in the discretization by $^n[35]$

$$\hat{\sigma} \rightarrow \frac{\hat{\sigma}}{\sqrt{\Delta t}} \quad (2.73)$$

The Langevin equation is obviously not energy conserving. The energy is not conserve and it fluctuates because of the friction and stochastic forces. As a result, the ensemble describing it is the canonical or NVT-ensemble, and the equilibrium distribution should be the Boltzmann distribution. In order to achieve this, the dissipative and random forces have to be related. To derive the correct relation, we consider the phase-space distribution $\rho(\Gamma; t)$. It satisfies the equation

$$\frac{\partial \rho}{\partial t} = -\sum \left( \frac{\partial \rho}{\partial r_i} \dot{r}_i + \frac{\partial \rho}{\partial p_i} \dot{p}_i \right)$$

$$= -\sum \left( \frac{\partial \rho}{\partial r_i} \hat{\dot{r}}_i + \frac{\partial \rho}{\partial p_i} \hat{\dot{p}}_i \right) + \sum_i \frac{\partial}{\partial p_i} \left( \gamma \frac{p_i}{m_i} \rho + \frac{\hat{\sigma}^2}{2} \frac{\partial \rho}{\partial p_i} \right)$$

$$= -\sum \left( \frac{\partial \rho}{\partial r_i} \dot{p}_i + \frac{\partial \rho}{\partial p_i} \dot{r}_i \right) + \sum_i \frac{\partial}{\partial p_i} \left( \gamma \frac{\partial H}{\partial p_i} \rho + \frac{\hat{\sigma}^2}{2} \frac{\partial \rho}{\partial p_i} \right)$$

$$= \left( \mathcal{L} + \mathcal{L}_{LD} \right) \rho \quad (2.74)$$

where $\mathcal{L}$ is the classical Liouville operator and

$$\mathcal{L}_{LD} = \sum_i \frac{\partial}{\partial p_i} \left( \gamma \frac{\partial H}{\partial p_i} \rho + \frac{\hat{\sigma}^2}{2} \frac{\partial \rho}{\partial p_i} \right) \quad (2.75)$$

The equation is the Fokker-Planck equation for the density of Langevin dynamics. By applying the Boltzmann distribution to it we get

$$\left( \mathcal{L} + \mathcal{L}_{LD} \right) e^{-\beta H} = \mathcal{L}_{LD} e^{-\beta H} = \sum_i \frac{\partial}{\partial p_i} \left( \gamma \frac{\partial H}{\partial p_i} \rho + \frac{\hat{\sigma}^2}{2} \frac{\partial \rho}{\partial p_i} \right) \quad (2.76)$$

and it will result in the following condition that relate the dissipative and random forces together, and to the temperature of the system $^{[10]}$

$$\hat{\sigma}^2 = 2m\gamma k_B T \quad (2.77)$$

Obviously, Langevin dynamics doesn’t satisfy the conservation of momentum. The conservation of momentum is the fundamental assumption in fluid mechanics, so this makes Langevin dynamics not a good approach for calculation in fluid dynamics. That is why the dissipative particle dynamics is introduced, and in the next chapter we go over general introduction of the method.
Chapter 3
Dissipative particle dynamics

3.1 Introduction

The dynamics of a complex fluid or fluid close to interfaces is affected by its microscopic structure. Because the molecular level dynamics is playing a role, and macroscopic scale is effected by lower levels, in these situations the fluid cannot be completely described by continuum theory and Navier-Stokes equation. However, molecular dynamics also is not useful in the practical point of view. The point is the number of particle in the large scales are too high that the computation time could take years with current computational power, which is not useful. For the fact that there are multiscale effects in these flow situations none of the molecular dynamics or continuum models work in this situation, so we need a middle ground to solve the dilemma.

Mesoscopic simulation methods try to solve the problem by making use of information from different length scales, macroscopic and microscopic, and solve the problem in middle ground in the mesoscale level. The formal foundation for mesoscopic simulation techniques is the theory of coarse-graining. However, many mesoscopic simulation methods are not strictly derived in a mathematical sense, but they are constructed in a way that satisfies certain conservation laws or symmetries that are considered to be essential for the observable phenomena. In the figure 3.1 the representation of different models and the scales are presented.
One of the first mesoscopic simulation methods was the lattice gas automata (LGA) introduced by Frisch et al. in 1986 [31]. In this method, the particles of fluid move along the edges of lattice. The system is described by some collision rules for the particles. Therefore the computational costs per particle are reduced considerably compared to molecular dynamics. The lattice gas approach yields correct hydrodynamics, and the time step in lattice gas simulations can be much larger than in molecular dynamics simulations. The other alternative approach is lattice Boltzmann approach, the Boltzmann equation is solved by this difference that lattice nodes are described by distribution functions. One important aspect of LB and LGA approaches are that they depend on the chosen lattice, so the Galilean invariance, which states that the laws of motion are the same in all inertial frames, is not satisfied in some lattice selections. The other important problem with these methods is that assigning boundary conditions is not straightforward [22].

Another technique to deal with different scales in a system is the Langevin method. In this approach, the fast degrees of freedom are eliminated completely and replaced by friction and random forces. This methods is useful for simulating fluid systems where the structure of the fluid is uninteresting [35]. However, the Langevin method is not useful for fluids because it does not produce correct hydrodynamics. The reason for this is that the random and friction forces do not conserve momentum, which is a basic requirement for hydrodynamic behavior.

In this chapter we introduce of dissipative particle dynamics (DPD). Dissipative particle dynamics is a particle based mesoscopic simulation method for isothermal fluids, the energy conservation also added later to the model. It does not suffer from the drawbacks of previous methods, i.e. it satisfies Galilean invariance and isotropy, and it conserves mass and momentum. Dissipative particle dynamics has a number of advantages. First, the lack of a lattice yields Galilean invariance and isotropy, and in practice it is much less cumbersome to include boundaries than in the lattice approaches. Second, since DPD conserves mass and momentum by construction, it is expected to produce correct hydrodynamic behavior. Third, being a particle based method, DPD is quite versatile with respect to addition of new
model feature \[35\]. Complex fluid structures can be modeled straightforwardly by introducing different particle types with corresponding interactions. Further extension are possible by introducing internal variables for the dissipative particles, e.g. internal energies (which is the main purpose of this study) or charges. Therefore, DPD seems a very promising technique for the simulation of complex fluids and soft matter systems.

In this chapter I give a brief history of DPD then I go into introducing the method. From the stochastic differential equations describing the evolution of the DPD system, the equivalent Fokker-Planck equation can be deduced. Then we extend the method to obtain an energy conserving system. Finally, I present a theory for the transport properties of the DPD model, namely the theory of Marsh, Backx and Ernst DPD.

### 3.2 Previous works on DPD

Dissipative particle dynamics was introduced by Hoogerbrugge and Koelman in 1992 \[22\] as a new method for simulating hydrodynamic phenomena. It combines the best of both molecular dynamics and lattice gas approaches. It obeys Galilean invariance and isotropy and conserves mass and momentum by construction. In the original contribution, it was verified both analytically and by simulations that correct hydrodynamic behavior is produced. The coarse-grained variables satisfy the Navier-Stokes equation to lowest order in the field gradients.

The dissipative particle dynamics was used for simulations of hard-sphere suspensions under steady shear by Koelman and Hoogerbrugge \[24\] and polymer solutions in Kong et al., 1994 \[25\], and in Schlijper et al. \[36\]. The dissipative and stochastic forces in DPD are Langevin-like and do not conserve the energy of the system. There is no transport of energy in the original model. Español and Warren in 1995 \[12\], derived the correct fluctuation dissipation relation for the friction and noise terms in dissipative particle dynamics. Español also derived the DPD Fokker-Planck equation, which is analogous to MD Liouville equation, and showed theoretical connection between dissipative particle dynamics and continuum fluid mechanics and is the essential formalism for the derivation of the kinetic and hydrodynamic equations\[12\]. By deriving the Fokker-Planck equation of the DPD equations of motion, they obtained the condition under which the steady-state solution is the Gibbs canonical ensemble with a well defined temperature. The equilibrium solution is not obtained by solving the Fokker-Planck equation directly, but it is a natural outcome of proving the H-theorem and demonstrating that the Gibbs distribution is the stationary solution of the Fokker-Planck equation. They showed that DPD particles tend to migrate towards the equilibrium state, which yields the Gibbs distribution as the equilibrium solution. They also observed in simulations that discretization errors due to finite time steps lead to deviations from the prescribed equilibrium temperature. Further investigations have clarified several aspects of DPD and led to rigorous theoretical justifications of the method. Español in 1995 \[12\] has derived the hydrodynamic equations for the mass and momentum density fields by
applying the projection operator technique. Groot and Warren in 1997[19] review dissipative particle dynamics and discuss how the friction coefficient, noise amplitude and time step may be chosen.

An analytic expression for the dependence of the equilibrium temperature on finite time steps was obtained by Marsh and Yeomans in 1997[30]. This was achieved by following the evolution of the one-particle distribution function under the finite time step algorithm. A stability bound for given model parameters was found. The Fokker-Planck-Boltzmann equation for dissipative particle dynamics was derived by Marsh et al. in 1997 [28] and was shown to obey an H-theorem that guarantees existence and approach of a unique equilibrium. From the Fokker-Planck-Boltzmann equation the macroscopic evolution equations were obtained using the Chapman-Enskog method [28]. Thereby, explicit expressions for the thermodynamic and transport properties were obtained.

Several extensions have been proposed for dissipative particle dynamics. As dissipative particle dynamics is an isothermal model in its original case, no energy transport and heat flux can be simulated. To overcome this limitation an energy conserving extension of DPD was developed independently by Espanol in 1997 [11]. The dissipative particles get an additional variable for the internal energy, and the corresponding evolution equation is derived consistently. The internal energy is coupled to the dissipative forces between the particles, and a fluctuation dissipation relation ensures the correct equilibrium distribution.

DPD used in many applications. Gibson et al. in 1998 [17] used DPD to simulate adsorption of colloidal particles onto a polymer-coated surface and their results achieved good agreement with the theoretical predictions, in that as the size of the polymers relative to the colloidal particle increases or similarly as density of the polymers increases, the adsorption of particles onto the surface would be less likely.

Kim et al. in 2004 [23] studied flows around spheres and cylinders at finite Reynolds numbers with a finite fluid inertia in DPD simulations. Flow around immobile objects and translation and rotation of mobile objects were investigated and it was shown that under computationally feasible conditions, DPD simulations are quantitatively accurate up to Reynolds numbers of $50 \approx 100$, and the inaccuracies at higher Reynolds numbers can be attributed to compressibility effects. Chen et al. in 2006 [6] estimated the drag force and torque of uniform and shear flow around a stationary sphere by employing DPD simulation, and they observed the importance of the dissipative coefficient on the resulting values of the drag force and torque.

In Revenga et al. in 1998 [34], the boundary is modelled as layer of fixed DPD particles, and taking the continuum limit of this layer, the dissipative and stochastic forces on the DPD fluid particles are determined analytically. In Revenga et al. in 1999 [33], the effects of Maxwellian, and bounce back reflections on sticking boundary conditions and temperature distributions, were examined. A similar method for cylindrical and spherical geometries, was used by Colmenares et al. in 2006 [7] to obtain explicit expressions for the effective random and dissipative forces for a point DPD particle.

In regard to heat transfer Abu-Nada in 2011 [1] modeled natural convection in a closed box, and heat conduction in a slab and find good agreement with analytical solutions. Yamada et al. in 2011 [39] investigate effectively on time dependent force convection for flow in a
3.3 Dissipative Particle Dynamics

The system of dissipative particle dynamics consists of a set of particles with continuous positions \( r_i \) and velocities \( v_i \), whose time evolution is described by Newton’s equations of motion

\[
\dot{r}_i = v_i \quad (3.1)
\]

\[
\dot{v}_i = f_i / m_i \quad (3.2)
\]

where \( i = 1, \ldots, N \) are the index of particles and \( m \) is the mass. The difference between dissipative particle dynamics to conventional molecular dynamics is in the definition of forces. The force acting on a particle has three parts

\[
f_i = \sum_{j \neq i} \left( F_{ij}^C + F_{ij}^D + F_{ij}^R \right) \quad (3.3)
\]

The three parts are a conservative force \( F_{ij}^C \), a dissipative force \( F_{ij}^D \) and a random force \( F_{ij}^R \). They are defined as

\[
F_{ij}^C = -\nabla_{r_{ij}} V_{ij}(r_{ij}),
\]

\[
F_{ij}^D = -\gamma \omega_D(r_{ij})(\hat{r}_{ij} \cdot \mathbf{v}_{ij}) \hat{r}_{ij},
\]

\[
F_{ij}^R = \sigma \omega_R(r_{ij}) \zeta_{ij} \hat{r}_{ij},
\]

where \( r_{ij} = r_i - r_j, r_{ij} = |r_{ij}|, \hat{r}_{ij} = e_{ij} = r_{ij}/r_{ij} \) and \( \mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j \). \( \gamma \) and \( \sigma \) are the friction and noise strength, respectively. \( \omega_D \) and \( \omega_R \) are weight functions that describe the range of the dissipative and random forces and vanish for \( r > r_c \). The term \( \zeta_{ij} \) is Gaussian white noise which is symmetric \( \zeta_{ij} = \zeta_{ji} \) and satisfies the properties

\[
\langle \zeta_{ij}(t) \rangle = 0
\]

\[
\langle \zeta_{ij}(t) \zeta_{i'j'}(t') \rangle = (\delta_{ii'}\delta_{jj'} + \delta_{ij'}\delta_{i'j})\delta(t-t'),
\]

which simply means the random force are not correlated between different particles and in different times. The Newton’s third law \( F_{ij} = -F_{ji} \) is satisfied, since there is symmetry in the indices \( ij \) and total momentum is conserved, which is different than the Langevin dynamics. In conventional DPD, the conservative forces are usually chosen as the following

\[
F_{ij}^C = \begin{cases} a_{ij}(1 - \frac{r_{ij}}{r_c}) \hat{r}_{ij} & r_{ij} < r_c \\ 0 & r_{ij} \geq r_c \end{cases}
\]

where the cutoff radius \( r_c \) is the same as for the weight functions \( \omega_D \) and \( \omega_R \). This soft conservative interaction between the particles is what make the use of a large time step
possible for integrating the equations of motion.

The dissipative force $F_{ij}^D$ is a friction force, which is related to the relative velocities of particles. The term $(\hat{r}_{ij} \cdot \mathbf{v}_{ij})$ is the projection of the relative velocity on the distance unit vector. When this is positive the particles are departing, the dissipative force drags them together in the direction of $\hat{r}_{ij}$, and when $(\hat{r}_{ij} \cdot \mathbf{v}_{ij})$ is negative, the particles are approaching together and the dissipative force stops them from approaching to each other. Obviously, the dissipative force dissipates the momentum of the particles, and they would eventually slow down until they stop their relative motion. This is where the random force $F_{ij}^R$ comes in and fix the loss of kinetic energy due to the dissipative force. It provides random momentum in the radial direction $\hat{r}_{ij}$ that keep the particles in motion, which can be assumed as thermal motion of the particles. The noise is symmetric in $ij$, so it satisfies the Newton’s third law. The usage of random force besides dissipative forces make sure that the momentum is conserved and the collision of particles is an isothermal process. The weighting functions $\omega_D$ and $\omega_R$ are not specifically declared by the original model, but from the description we just provided it can be inferred that only one of them can be chosen arbitrarily, and the other one is then fixed by the balance condition [12]

$$\omega_D(r) = [\omega_R(r)]^2 = \omega(r). \quad (3.7)$$

This ensures that a well-defined equilibrium state exists. This state is described by Boltzmann statistics where the temperature is defined in terms of the fluctuation-dissipation relation

$$m\sigma^2 = 2\gamma T, \quad (3.8)$$

that is, the friction and noise amplitudes are also not independent. A typical choice of the weight function is

$$\omega(r) = \begin{cases} a_{ij}(1 - \frac{r}{r_c}) & r_{ij} < r_c \\ 0 & r_{ij} \geq r_c \end{cases} \quad (3.9)$$

It worth noting that the dissipative particles doesn’t represent actual particles, but it represent a fluid package that contains many molecules. This package moves through the surrounding fluid and collisions are defined through the interaction forces. The eliminated degrees of freedom result in Langevin-like friction and stochastic forces, which are constructed to obey Galilean invariance, isotropy and momentum conservation. [35]

### 3.4 Stochastic Differential Equations and Fokker-Planck Equations for DPD

The equations of motion for dissipative particle dynamics can be written as stochastic differential equations. By inserting the DPD forces into the equations of motion we get [12].
\[ \frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \]
\[ \frac{d\mathbf{v}_i}{dt} = \frac{1}{m} \sum_{j \neq i} \left( F_{ij}^C dt - \gamma \omega_D(r_{ij})(\mathbf{r}_{ij} \cdot \mathbf{v}_{ij}) \mathbf{r}_{ij} dt + \sigma \omega_R(r_{ij}) \mathbf{r}_{ij} dW_{ij} \right) \]

where \( dW_{ij} = dW_{ji} \) are independent Wiener increments. We use the Itô-interpretation for which we have the rule
\[ dW_{ij} dW_{i'j'} = (\delta_{ii'} \delta_{jj'} + \delta_{ij} \delta_{i'j'}) dt. \]

It is useful to note that \( dW_{ij} \) is an infinitesimal of order \( 1/2 \), which implies that the random force in a discrete algorithm with finite time step becomes
\[ F_R(r_{ij}) = \sigma \omega (r_{ij}) \Delta t^{-1/2} \zeta_{ij} \mathbf{r}_{ij}. \]

From the stochastic differential equations we can derive the corresponding Fokker-Planck equation using standard procedures. By substituting the expressions of the stochastic differential equation in this expansion, one obtains the following Fokker-Planck equation
\[ \partial_t \rho(\Gamma; t) = \mathcal{L} \rho(\Gamma; t) = \left( \mathcal{L}_C + \mathcal{L}_D + \mathcal{L}_R \right) \rho(\Gamma; t), \]
where the Fokker-Planck operator \( \mathcal{L} = \mathcal{L}_C + \mathcal{L}_D + \mathcal{L}_R \) is split into three parts corresponding to the conservative, dissipative and random forces
\[ \mathcal{L}_C = -\sum_i \mathbf{v}_i \frac{\partial}{\partial \mathbf{r}_i} - \sum_{i,j \neq i} \frac{F_{ij}^C}{m} \frac{\partial}{\partial \mathbf{v}_i} \]
\[ \mathcal{L}_D = \frac{\gamma}{m} \sum_{i,j \neq i} \omega_D(r_{ij}) \left( \mathbf{r}_{ij} \cdot \frac{\partial}{\partial \mathbf{v}_{ij}} \right) (\mathbf{r}_{ij} \cdot \mathbf{v}_{ij}) \]
\[ \mathcal{L}_R = \frac{\sigma^2}{2m^2} \sum_{i,j \neq i} \omega_R^2(r_{ij}) \left( \mathbf{r}_{ij} \cdot \frac{\partial}{\partial \mathbf{v}_{ij}} \right) (\mathbf{r}_{ij} \cdot \left( \frac{\partial}{\partial \mathbf{v}_i} - \frac{\partial}{\partial \mathbf{v}_j} \right)) \]

The conservative part \( \mathcal{L}_C \) is just the classical Liouville operator corresponding to the Hamiltonian
\[ H(\Gamma) = \sum_i \frac{mv_i^2}{2} + \frac{1}{2} \sum_{i,j \neq i} V_{ij} \]

, and by summing up \( \mathcal{L}_D \) and \( \mathcal{L}_R \), one gets
\[ \mathcal{I}_D = \mathcal{L}_D + \mathcal{L}_R = \frac{\gamma}{2m} \sum_{i,j \neq i} \omega(r_{ij}) \mathbf{r}_{ij} \mathbf{r}_{ij} \left( \left( \frac{\partial}{\partial \mathbf{v}_i} - \frac{\partial}{\partial \mathbf{v}_j} \right) v_{ij} + \frac{k_B T}{m} \left( \frac{\partial}{\partial \mathbf{v}_i} - \frac{\partial}{\partial \mathbf{v}_j} \right) \left( \frac{\partial}{\partial \mathbf{v}_i} - \frac{\partial}{\partial \mathbf{v}_j} \right) \right) \]
Assuming that the steady-state solution of the Fokker-Planck equation is the Gibbs canonical ensemble distribution

\[ \rho_0(\Gamma) = \frac{1}{Z_{\text{can}}} \exp \left( -\frac{1}{k_B T_0} \left( \sum_i \frac{mv_i^2}{2} + \frac{1}{2} \sum_{i,j \neq i} V_{ij} \right) \right) \]  

with equilibrium temperature \( T_0 \), we can read off the detailed balance condition \( \omega_D = \omega_R^2 = \omega \) and the fluctuation dissipation relation \( \sigma^2 = 2 \gamma k_B T_0 \).

### 3.5 Energy conserving DPD

The DPD model that we describe is for a system with constant temperature, which means that total energy of the system is not conserved. Therefore, the system is not suitable for temperature gradients, and it is impossible to model energy transport and heat flow. In this section, I present the DPD model with energy conservation. This DPD with energy conserving has been developed by Español[11]. Español add temperature to the particle variables to represent the internal energy of the dissipative particles. This variable is then coupled to the dissipative forces to model the viscous heating process, and the difference in temperature describes the difference in internal energy of different particles, so by setting the temperature, we can model the heat conduction and other energy phenomena. In the following, I describe in detail the model of Español 1997 [11]. In this model a particle \( i \) with temperature \( T_i \) exchanges heat with neighboring particles \( j \) at temperature \( T_j \) using the following

\[ \dot{\epsilon}_{hc}^i = \sum_j \kappa_{ij} \omega(r_{ij}) \left( \frac{1}{T_i} - \frac{1}{T_j} \right) + \sum_j \tilde{q}_{ij}, \]  

where \( \kappa_{ij} = \kappa(\epsilon_i, \epsilon_j) \) is the thermal conductivity, \( \omega(r) \) describes the interaction range, and a random part \( \tilde{q}_{ij} \) is included for thermal fluctuations. The thermal conductivity is assumed to be symmetric \( \kappa_{ij} = \kappa_{ji} \), and the random heat flux is antisymmetric \( \tilde{q}_{ij} = -\tilde{q}_{ji} \) in order to conserve the total internal energy of the system. As before the stochastic differential equation corresponding to the Langevin-like equation is

\[ d\epsilon_{hc}^i = \sum_j \kappa_{ij} \left( \frac{1}{T_i} - \frac{1}{T_j} \right) \omega(r_{ij}) + \sum_j \alpha_{ij} \tilde{\omega}(r_{ij}) dW_{ij}^\epsilon \]  

where \( \tilde{\omega}_{ij} \) is the range of the random interaction and \( \alpha_{ij} \) is the noise amplitude of the Wiener increments \( dW_{ij}^\epsilon \). The Wiener process has to be antisymmetric \( dW_{ij}^\epsilon = -dW_{ij}^\epsilon \) and we adopt the Itô interpretation with the calculus rule

\[ dW_{ij}^\epsilon dW_{jj'}^\epsilon = (\delta_{ij}\delta_{jj'} - \delta_{ij'}\delta_{jj}) dt \]
3.5. ENERGY CONSERVING DPD

By the standard procedure we can obtain the equivalent Fokker-Planck equation for the probability distribution of the internal energy as,

$$\partial_t \rho(\epsilon) = -\sum_{ij} \omega(r_{ij}) \frac{\partial}{\partial \epsilon_i} \kappa_{ij} \left( \frac{1}{T_i} - \frac{1}{T_j} \right) \rho(\epsilon) + \frac{1}{2} \sum_{ij} \bar{\omega}^2(r_{ij}) \left( \frac{\partial^2}{\partial \epsilon_i \epsilon_i} - \frac{\partial^2}{\partial \epsilon_i \epsilon_j} \right) \alpha_{ij}^2 \rho(\epsilon).$$  \hspace{1cm} (3.23)

Just like the previous section, the Fokker-Planck equation makes the balance conditions

$$\alpha_{ij}^2 = 2 \kappa_{ij}$$ \hspace{1cm} (3.24)

$$\omega(r) = \bar{\omega}(r)$$ \hspace{1cm} (3.25)

With the additional assumption

$$\left( \frac{\partial}{\partial \epsilon_i} - \frac{\partial}{\partial \epsilon_j} \right) \kappa(\epsilon_i, \epsilon_j) = 0$$ \hspace{1cm} (3.26)

We have then the Fokker-Planck operator for heat conduction

$$\mathcal{L}_{hc} = \sum_{ij} \omega(r_{ij}) \frac{\partial}{\partial \epsilon_i} \left( \frac{1}{T_i} - \frac{1}{T_j} + \left( \frac{\partial}{\partial \epsilon_i} - \frac{\partial}{\partial \epsilon_j} \right) \right) \kappa_{ij}$$ \hspace{1cm} (3.27)

and the equilibrium solution is of the form

$$\rho_0(\epsilon) = \frac{1}{Z} P \left( \sum_k \epsilon_k \right) \exp \left( \sum_i s(\epsilon_i) \right)$$ \hspace{1cm} (3.28)

$P(\sum_k \epsilon_k)$ is a function of the total internal energy and $s(\epsilon_i)$ is the entropy. The equilibrium solution $\rho_0(\epsilon)$ has the form of the Einstein formula for energy fluctuations. In equilibrium, the expectation values of the temperatures are equal $\langle 1/T_i \rangle = \langle 1/T_j \rangle$.

Another part of the energy equation is the viscous heating. The momentum equation for particles with different pairs of $\gamma$ and $\sigma$ with the index $ij$ is as the following

$$d\mathbf{v}_i = \frac{1}{m} \sum_{j \neq i} \left( \mathbf{F}_{ij}^C dt - \gamma_{ij} \omega_D(r_{ij})(\mathbf{\hat{r}}_{ij} \cdot \mathbf{v}_{ij}) \mathbf{\hat{r}}_{ij} dt + \sigma_{ij} \omega_R(r_{ij})(\mathbf{v}_{ij} \cdot \mathbf{\hat{r}}_{ij}) dW_{ij} \right)$$ \hspace{1cm} (3.29)

These terms in this equation change mechanical energy of the system

$$dE = -\frac{m}{2} \sum_{i,j \neq i} \left( \gamma_{ij} \omega_D(r_{ij})(\mathbf{v}_{ij} \cdot \mathbf{\hat{r}}_{ij})^2 dt - \sigma_{ij}^2 \omega_R^2(r_{ij}) dt - \sigma_{ij} \omega_R(r_{ij})(\mathbf{v}_{ij} \cdot \mathbf{\hat{r}}_{ij}) dW_{ij} \right) \hspace{1cm} (3.30)$$

The change of the mechanical energy must be the same as the change in the internal energy for the model to satisfy the conservation of the total energy, i.e. $dE = \sum_i d\epsilon_i^{vh}$. Therefore, we get the stochastic differential equation

$$d\epsilon_i^{vh} = -\frac{m}{2} \sum_{i,j \neq i} \left( \gamma_{ij} \omega_D(r_{ij})(\mathbf{v}_{ij} \cdot \mathbf{\hat{r}}_{ij})^2 dt - \sigma_{ij}^2 \omega_R^2(r_{ij}) dt - \sigma_{ij} \omega_R(r_{ij})(\mathbf{v}_{ij} \cdot \mathbf{\hat{r}}_{ij}) dW_{ij} \right)$$ \hspace{1cm} (3.31)
3.6 Deriving the Navier-Stokes Equation from DPD

From this we get the Fokker-Planck operator for viscous heating

\[ L_{vh} = \frac{1}{2} \sum_{i,j \neq i} \omega_D(r_{ij})L_{ij} \left( \gamma_{ij}(v_{ij} \cdot \dot{r}_{ij}) + L_{ij} \frac{\sigma_{ij}^2}{2} \right) \]  \hspace{1cm} (3.32)

where we have used the detailed balance condition \( w_D(r) = \omega^2_R(r) \) and the operator \( L_{ij} \) is given by

\[ L_{ij} = \dot{r}_{ij} \cdot \left( \frac{\partial}{\partial v_i} - \frac{\partial}{\partial v_j} - \frac{m}{2} v_{ij} \left( \frac{\partial}{\partial \epsilon_i} + \frac{\partial}{\partial \epsilon_j} \right) \right) \]  \hspace{1cm} (3.33)

The full Fokker-Planck equation for the system can be written in the form

\[ \partial_t \rho = (L_C + L_{vh} + L_{hc})\rho, \]  \hspace{1cm} (3.34)

where \( L_C \) is the classical Liouville operator. If we assume that the equilibrium distribution has the form

\[ \rho_0(r, v, \epsilon) = \frac{1}{Z} P(E(r, v, \epsilon), P) \exp(\sum_i s(\epsilon_i)) \]  \hspace{1cm} (3.35)

and assume that the noise amplitude is constant for all particle pairs \( \sigma_{ij} = \sigma \), the detailed balance condition for \( \gamma_{ij} \) is

\[ \gamma_{ij} = \frac{1}{4} \sigma^2 \left( \frac{1}{T_i} - \frac{1}{T_j} \right). \]  \hspace{1cm} (3.36)

The full set of motion equations for the DPD model with energy conservation is

\[ \dot{r}_i = v_i \]

\[ \dot{v}_i = \sum_{j \neq i} \left( F_{ij}^C - \gamma_{ij} \omega_D(r_{ij})(\dot{r}_{ij} \cdot v_{ij})\dot{r}_{ij} + \sigma \omega_R(r_{ij}) \psi_{ij} \hat{r} \right) \]

\[ \dot{\epsilon}_i = \frac{1}{2} \sum_{j \neq i} \left( \gamma_{ij} \omega_D(r_{ij})(\dot{r}_{ij} \cdot v_{ij})^2 - \sigma^2 \omega_R^2(r_{ij}) - \sigma \omega_R(r_{ij})(\dot{r}_{ij} \cdot v_{ij})\psi_{ij} \right) \]

\[ + \kappa_{ij} \omega(r_{ij}) \left( \frac{1}{T_i} - \frac{1}{T_j} \right) + \alpha_{ij} \omega(r_{ij}) \psi_{ij}^t \]  \hspace{1cm} (3.37)

In a discrete updating scheme, the heat conduction part of the algorithm will conserve energy up to machine precision, while the viscous heating part only conserves energy in the limit of infinitesimal time step [11].

3.6 Deriving the Navier-Stokes equation from DPD

Another interesting aspect of DPD is to see if the formulation can result in Navier Stokes equation. Marsh et al.[29] used a kinetic theory approach to calculate the transport coefficients of the isothermal DPD system and by using Fokker-Planck equation, they found
3.6. DERIVING THE NAVIER-STOKES EQUATION FROM DPD

the macroscopic transport equations. Then the Fokker-Planck-Boltzmann equation is used
to investigate the dynamical behavior of the DPD system. They solve the Fokker-Planck-
Boltzmann equation by Chapman-Enskog method, which yields the Navier-Stokes equation
and the transport coefficients, which we present in this section.
The hydrodynamic variables mass density, momentum density and energy density are as the
following.

\[ \rho(r, t) = m \int f^{(1)}(r, v, t) dv \]

\[ n u(r, t) = \int f^{(1)}(r, v, t) v dv \]  

\[ \epsilon(r, t) = \int \frac{m}{2} v^2 f^{(1)}(r, v, t) dv + \frac{1}{2} \int dv dv' \int V(R) f^2(r, v, r', v', t) dR \] (3.38)

A general macroscopic quantity \( A \) defined by

\[ \langle A \rangle = \int A(\Gamma) \rho(\Gamma, t) d\Gamma \]

The evolution of \( A \) based on Focker Planck equation is

\[ \partial_t \langle A \rangle = \left( \sum_i v_i \frac{\partial}{\partial r_i} + \sum_{i,j \neq i} \frac{F_{ij}^C}{m} \frac{\partial}{\partial v_i} \right) A \]

\[ - \frac{\gamma}{m} \left( \sum_{i,j \neq i} \omega(\hat{r}_{ij} \cdot v_{ij}) \left( \hat{r}_{ij} \cdot \frac{\partial}{\partial v_i} \right) \right) \]

\[ + \frac{\gamma k_B T_0}{m^2} \left( \sum_{i,j \neq i} \omega(r_{ij}) \left( \hat{r}_{ij} \cdot \frac{\partial}{\partial v_i} \right) \left( \hat{r}_{ij} \cdot \left( \frac{\partial}{\partial v_i} - \frac{\partial}{\partial v_j} \right) \right) \right) A \] (3.39)

Applying to the macroscopic variables we have

\[ \partial_t \rho = -\nabla \rho u \]

\[ \partial_t (\rho u) = -\nabla (\rho uu + \Pi) \]

\[ \partial_t \epsilon = -\nabla q + \Gamma \] (3.40)

where \( \Pi \) is the local pressure tensor

\[ \Pi = \Pi_K + \Pi_C + \Pi_D \]

\[ \Pi_K = \int mVV f^{(1)}(r, v, t) dv \]

\[ \Pi_C = \frac{1}{2} \int dv dv' \int RF(R) \tilde{f}^{(2)}(r, v, v', t) dR \]

\[ \Pi_D = -\frac{1}{2} \gamma \int dv dv' \int \omega(R) \hat{R} \tilde{f}^{(2)}(r, v, v', t) dR \] (3.41)

in which The indexes are kinetic, conservative and dissipative, \( V = v - u(r, t) \), \( q \) is the heat
flux and \( \Gamma(r, t) \) is a local source term and

\[ \tilde{f}^{(2)}(r, v, v', t) = \int_0^1 f^{(2)}(r - \lambda R, v, r - (1 - \lambda) R, v', t) d\lambda \] (3.42)
is the average of two particle distribution function.

\[
\Gamma(r, t) = \frac{\gamma}{m} \left\langle \sum_{i,j \neq i} \omega(r_{ij})(k_B T_0 - \frac{m}{2}(\hat{r}_{ij} \cdot \mathbf{v}_{ij})^2)\delta(\mathbf{r} - \mathbf{r}_i) \right\rangle
\] (3.43)

In order to determine the local equilibrium solution \(f_0\), we delocalize the right-hand side of the Fokker-Planck-Boltzmann equation by substituting \((\mathbf{r} - \mathbf{R}) \rightarrow \mathbf{r}\). We get the delocalize operator

\[
I_0(f) = \frac{\gamma}{m} \int d\mathbf{v}' \int \omega(R)f(\mathbf{r}, \mathbf{v}', t) \cdot \left( \frac{\partial}{\partial \mathbf{v}} (\mathbf{v} - \mathbf{v}') + \frac{k_B T_0}{m} \frac{\partial^2}{\partial \mathbf{v} \partial \mathbf{v}} \right)f(\mathbf{r}, \mathbf{v}, t) d\mathbf{R}
\] (3.44)

for which \(f_0^{(1)}\) solves the equation

\[
I_0(f_0) = 0
\] (3.45)

The equilibrium distribution can be written as

\[
f_0(\mathbf{v}) = n \left( \frac{m}{2\pi k_B T} \right)^{\frac{3}{2}} \exp \left( -\frac{m(\mathbf{v} - \mathbf{u})^2}{2k_B T} \right)
\] (3.46)

We consider small deviation from global equilibrium \(\delta f = f^{(1)} - n_0 \phi_0\), where \(\phi_0\) is the Maxwellian velocity distribution, and linearization of the Fokker-Planck-Boltzmann equation around \(n_0 \phi_0\) will become

\[
\partial_t \delta f + \mathbf{v} \cdot \nabla \delta f = \omega_0 \frac{\partial}{\partial \mathbf{v}} (\mathbf{v} + \frac{k_B T_0}{m} \frac{\partial}{\partial \mathbf{v}} \delta f)
\]

\[
+ \frac{n_0^2 \gamma}{k_B T_0} \int \omega(R) \mathbf{R} \cdot \mathbf{u}(\mathbf{r} - \mathbf{R}) \mathbf{v} \phi_0(\mathbf{v}) d\mathbf{R}
\] (3.47)

In which

\[
\omega_0 = \frac{1}{t_0} = \frac{n_0 \gamma}{3m} \int \omega(R) d\mathbf{R} = \frac{n_0 \gamma}{3m} \omega
\] (3.48)

In the subsequent hydrodynamic stage, the solution of the Fokker-Planck-Boltzmann equation is obtained by an expansion in \(\mu \sim l_0 \nabla\)

\[
\partial_t f_0 + \mu \mathbf{v} \cdot \nabla f_0 = I(f_0) + \mu \left( \frac{dI}{df} \right) f_1 + \cdots
\] (3.49)

where \(I(f)\) is the collision operator on the right hand side of the Fokker-Planck-Boltzmann equation. We want to calculate both sides of this equation to \(O(\mu)\) terms. Therefore, we delocalize the collision operator as described above and get

\[
\left( \frac{dI}{df} \right)_{f_0} f_1 = \omega_0 \frac{\partial}{\partial \mathbf{v}} (\mathbf{v} + \frac{k_B T}{m} \frac{\partial}{\partial \mathbf{v}}) f_1 = \omega_0 L_1 f_1
\] (3.50)
For the left hand side, we rewrite the macroscopic conservation equations for local equilibrium, i.e. the pressure tensor is replaced by $\Pi_0 = nk_B T_0 I$

$$\partial_t n = -\nabla(nu)$$
$$\partial_t u = -u \cdot \nabla u - \frac{k_B T}{\rho} \nabla n$$

(3.51)

For the equilibrium distribution $f_0$ we can then write

$$(\frac{\partial}{\partial t} + v \nabla)f_0 = \left( \frac{\partial u^\alpha}{\partial t} \frac{\partial}{\partial u^\alpha} + \frac{\partial n}{\partial t} \frac{\partial}{\partial n} + \frac{\partial u^\alpha}{\partial r^\beta} v^\beta \frac{\partial}{\partial u^\alpha} + v^\alpha \frac{\partial u}{\partial r^\alpha} \frac{\partial}{\partial n} \right)f_0$$
$$= f_0(J : D + \mathcal{J} \nabla \cdot u)$$

(3.52)

in which

$$J_{\alpha\beta}(V) = \frac{m}{k_B T_0} \left( V_{\alpha\beta} V - \frac{1}{3} \delta_{\alpha\beta} V^2 \right)$$
$$D_{\alpha\beta}(V) = \frac{1}{2} \left( \frac{\partial u^\beta}{\partial r^\alpha} + \frac{\partial u^\alpha}{\partial r^\beta} - \frac{2}{3} \delta_{\alpha\beta} \nabla u \right)$$
$$\mathcal{J}(V) = \frac{m V^2}{3 k_B T_0} - 1$$

(3.53)

Collecting equations we get the final equation for $f_1$

$$f_0[J : D + \mathcal{J} \nabla \cdot u] = \omega_0 L_1 f_1,$$

(3.54)

which has the special solution [29]

$$f_1 = -\frac{1}{2\omega_0} f_0[J : D + \mathcal{J} \nabla \cdot u]$$

(3.55)

The zeroth order $f_0$ and the first order correction $f_1$ complete the solution up to order $\mu$. The $O(\mu)$ results are sufficient to obtain the transport coefficients in Navier-Stokes hydrodynamics. For this purpose we write the pressure tensor in the usual form

$$\Pi = p_0 I - 2\eta D - \zeta \nabla \cdot u I$$

(3.56)

With the momentum flow equation this result in the Navier-Stokes equation for DPD

$$\partial_t (\rho u) + \nabla(\rho uu) = -k_B T_0 \nabla n + \nabla((2\eta D + \zeta \nabla \cdot u) I)$$

(3.57)

More detail can be found in Marsh et al.[29]
Chapter 4

Simulation process

4.1 Introduction

In this chapter we focus on assembling the particle-particle interaction models into a framework for numerical computation of the dynamics of a collection of particles. We describe how the particle model works, and how we implement different functions of the code, and the code used for simulation is also presented.

4.2 Particles

The basic entities in a soft matter system are particles. In order to model a particle in a computer program, we need to specify the relevant properties that characterize a particle. These can be extracted from the laws of physics governing the behavior of the particle. In the easiest case, this will be a simple particle governed by Newton’s equation of motion, which relates the position of the particle to its momentum, and the momentum to the force exerted on the particle. Thus, a particle data structure must incorporate at least the position, the momentum, the force, and the temperature of the particle. The implementation of the particle in the code is as a struct

```c
typedef struct
{
    double x;
    double y;
    double z;
    double vx;
    double vy;
    double vz;
    double ax;
    double ay;
    double az;
};
```
4.2. PARTICLES

```c
double T;
double q;
int bin_x;
int bin_y;
int bin_z;
int id;
int boundary;
} particle_t;
```

and of course the memory is needed to be allocated for \( n \) number of particles.

```c
particle_t *particles = (particle_t*) malloc( n * sizeof(particle_t) );
```

In the begining of the simulation the particles are initialized using the field data, and the velocities are applied as a random number based on a standard distribution with mean velocity of the input velocity.

```c
void init_particles( int n, particle_t *p )
{
    for( int i=0; i<bnd_parts;i++ ){
        p[i].x =
        p[i].y =
        .
        .
    }
    for( int i = bnd_parts; i < n; i++ )
    {
        p[i].x =
        p[i].y =
        .
        .
    }
}
```
4.3 Interactions

The forces between the particles are determined by the interaction potential. In computer simulations, the potential is split up into pair potentials. Interactions can occur between arbitrary particles in the system based on their distances. The time progress of DPD particles is governed by conservation of momentum and energy and is described by the following set of equations.

\[
\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i
\]  

(4.1)

\[
\frac{d\mathbf{v}_i}{dt} = \sum_j (f^{\text{Con}}_{ij} + f^{\text{Dis}}_{ij} + f^{\text{Ran}}_{ij})
\]

(4.2)

\[
\frac{d\mathbf{e}_i}{dt} = C_v \frac{dT_i}{dt} = \sum_j (q^{\text{Cond}}_{ij} + q^{\text{Vis}}_{ij} + q^{\text{Ran}}_{ij})
\]

(4.3)

The conservative force \(f^{\text{Con}}_{ij}\), dissipative force \(f^{\text{Dis}}_{ij}\) and random force \(f^{\text{Ran}}_{ij}\) are expressed as

\[
f^{\text{Con}}_{ij} = \sum_{j \neq i} a_{ij} w(r_{ij}) \mathbf{e}_{ij}
\]

(4.4)

\[
f^{\text{Dis}}_{ij} = \sum_{j \neq i} -\gamma_{ij} w^2(r_{ij})(\mathbf{e}_{ij} \cdot \mathbf{v}_{ij}) \mathbf{e}_{ij}
\]

(4.5)

\[
f^{\text{Ran}}_{ij} = \sum_{j \neq i} -\sigma_{ij} w(r_{ij}) \zeta_{ij} \Delta t^{-1/2} \mathbf{e}_{ij}
\]

(4.6)

Where \(r_{ij} = r_i - r_j\) and \(v_{ij} = v_i - v_j\), and \(\mathbf{e}_{ij}\) is the unit vector pointing in the direction from \(j\) to \(i\). The parameter \(a_{ij}\) is a repulsion parameter between the particles. Also, the \(\gamma_{ij}\) and \(\sigma_{ij}\) are the strength of dissipative and random forces, respectively. \(\omega\) is a weight function which describes the range of the force, and in the previous chapter we saw that \(\sigma^2 = 2\gamma k_B T\).

The conduction \(q^{\text{Con}}_{ij}\), viscous heating \(q^{\text{Dis}}_{ij}\) and thermal fluctuation \(q^{\text{Ran}}_{ij}\) heat transfer are expressed as

\[
q^{\text{Con}}_{ij} = \sum_{j \neq i} \kappa_{ij} w^2(r_{ij}) \left( \frac{1}{T_i} - \frac{1}{T_j} \right)
\]

(4.7)

\[
q^{\text{Dis}}_{ij} = \sum_{j \neq i} \frac{1}{2} \left( w^2(r_{ij}) (\gamma_{ij}(\mathbf{e}_{ij} \cdot \mathbf{v}_{ij})^2 - \sigma_{ij}^2) - \sigma_{ij} w(r_{ij})(\mathbf{e}_{ij} \cdot \mathbf{v}_{ij}) \zeta_{ij} \right)
\]

(4.8)

\[
q^{\text{Ran}}_{ij} = \sum_{j \neq i} \alpha_{ij} w(r_{ij}) \zeta_{ij}^l
\]

(4.9)

Also, the \(\kappa_{ij}\) and \(\alpha_{ij}\) determine the strength of the collisional and random heat flux. The weight function \(w\) decreases monotonically with particle-particle separation distance. It
becomes zero beyond the cutoff length and in the present work the following weight function is used.

\[ w(r_{ij}) = \begin{cases} 
1 - \frac{r_{ij}}{r_c} & r_{ij} < r_c \\
0 & r_{ij} \geq r_c 
\end{cases} \] (4.10)

The random number \( \zeta_{ij} \) that appears is a random number that has a zero mean and unit variance and has the property \( \zeta_{ij} = \zeta_{ji} \) to ensure the conservation of the total momentum of the system. We also saw that

\[ \alpha_{ij} = \sqrt{2\kappa_{ij}} \] (4.11)

where \( k_B \) is the Boltzmann constant.

Therefore in order to solve any DPD model we need to find three constants namely, \( a_{ij}, \gamma_{ij}, \) and \( \kappa_{ij} \), which are constant for repulsive, dissipative, and conduction respectively. In order to find these constants, we need to run a test and find constants with which the results of the DPD model fits the analytical or experimental data.

### 4.4 Force Calculation

A naive code calculates forces between any two particles during each step of the simulation. This result in \( O(n^2) \) calculations. However, the particles only interact with their neighbouring particles which is in the order of \( O(n) \). The main idea to make the code faster is to assign particles into small bins, and only calculate pairwise particle forces between particles in the neighbouring bins. Obviously, the size of the bin could be as small as the cutoff distance for each particle to only interact with the neighbouring bins. This idea can change the order of calculation in the order number of particles in nine in 2D and 27 in 3D neighbor bins which is \( O(n) \).

#### 4.4.1 Design of binned code

To keep the code simple and fast we assign bins as an array of integers, that holds particle IDs, so for a particle to be in a bin we only need to add its particle ID to the array of its corresponding bin. The following figure shows neighbouring bins.
The procedure of the simulation is as the following:

1. Define particle and bin variables

2. Iterate over time steps
   
   (a) Clear the bins and assign particles to bins, which is O(n) computation.
   
   (b) Computing the force only for each bin only by comparing the bin to its eight neighbours, as described above this is O(n) computation.
   
   (c) Move the particles, which is just change the location of the particles based on the calculated force, which is again O(n)

The following is the sample code of initializing bins

```c
__device__ void assign_bins( particle_t* particles, int startID, int num_P, int* bin_PID, int* bin_ID, int num_bins_side, int t_num_p) {
    int new_x, new_y, new_z;
    int binIndex;
    int binCount = 0;
    int pID;
    int pOffset;
    particle_t* particle;
    int numBins = num_bins_side * num_bins_side * num_bins_side;

    for(int p = startID; p < startID + num_P; p++) {
```

Figure 4.1: The binned domain. Only particles in the neighbouring bins are considered for force calculation
4.4. FORCE CALCULATION

// get the id of the particle in the list and push it to the correct bin, worry about erasing them later
if (p < 0 || p >= t_num_p) {
    printf("Bad access at index %d\n", p);
}  
particle = &particles[p];

new_x = floor((∗particle).x / BIN_SIZE);
new_y = floor((∗particle).y / BIN_SIZE);
new_z = floor((∗particle).y / BIN_SIZE);

ID_BIN(new_x, new_y, new_z, num_bins_side, &binIndex);
bin_particle_id(binIndex, &pOffset);

if (binIndex < numBins) {
    binCount = atomicAdd(bin_ID + binIndex, 1);
    if (binCount < 0 || binCount >= numBins*MAX_PARTICLES_PER_BIN) {
        printf("BIN COUNT %d\n", binCount);
    }
    if (binCount < MAX_PARTICLES_PER_BIN) {
        pID = pOffset + binCount;
        bin_PID[pID] = p;
        particle->bin_x = new_x;
        particle->bin_y = new_y;
        particle->bin_z = new_z;
    }
    else
        atomicAdd(bin_ID + binIndex, -1);
}

}  
The following code is a sample code of calculating force on neighbouring bin

__global__ void forces_gpu(particle_t * particles, int num_P, int* bin_PID, int*
bin_ID, int num_bins_side) {
}
// get this thread local and global linear id
int global_tid_x = threadIdx.x + blockIdx.x * blockDim.x;
int global_tid_y = threadIdx.y + blockIdx.y * blockDim.y;
int global_tid_z = threadIdx.z + blockIdx.z * blockDim.z;
if (global_tid_x >= num_bins_side || global_tid_y >= num_bins_side ||
global_tid_z >= num_bins_side)
    return;

apply_force_bin(global_tid_x, global_tid_y, global_tid_z, num_bins_side, bin_PID,
                bin_ID, particles);

__device__ void apply_force_bin(int x, int y, int z, int num_bins_side, int* bin_PID,
                                int* bin_ID, particle_t* particles) {
    int binId;
    ID_BIN(x, y, z, num_bins_side, &binId);

    int pOffset;
    int pID;
    int neighborId;
    int neighborOffset;
    int neighborPID;
    int neighborID;
    int i, j, kk;

    bin_particle_id(binId, &pOffset);

    int num = bin_ID[binId];
    // apply force for each bin
    for (int k = 0; k < num; k++) {
        pID = bin_PID[pOffset + k];
        particles[pID].ax = 0;
        particles[pID].ay = 0;
        particles[pID].az = 0;
    }
4.5 Integration

Velocity-Verlet scheme is used for integration. Which is an algorithm that yields the positions, velocities and forces at the same time is given by the scheme. The positions and velocities are updated according to To solve the problem of having both velocity and positions and the same time we can use the Velocity-Verlet scheme. The positions and velocities are updated according to

\[
\begin{align*}
\mathbf{r}_i(t + \Delta t) &= \mathbf{r}_i(t) + \Delta(t)\mathbf{v}_i(t) + \frac{\Delta t^2}{m_i}f_i(t) + O(\Delta t^3) \\
\mathbf{v}_i(t + \Delta t) &= \mathbf{v}_i(t) + \frac{\Delta t}{2m_i}(f_i(t) + f_i(t + \Delta t)) + O(\Delta t^3)
\end{align*}
\] (4.12)
4.6 PARALLELIZATION

The Velocity-Verlet algorithm as mentioned before is very stable. The reason for the stability of the Velocity-Verlet algorithm is that it is reversible and it preserves the volume in phase-space. [15].

4.6 Parallelization

The GPU code is largely similar to the OpenMP, since both follow the shared memory model. The only major difference between the CUDA and OpenMP is solving the race condition between different threads. In OpenMP this is done by locks, but in CUDA there are atomic operations that deals with this problem. Specifically, we represent the bins in which we find neighboring particles as an array of indices of the particles belonging to that bin, and keep record of the current tail of the array. Whenever a thread is pushing a particle to a bin, it first attempts to increase the current tail of the array by 1, using atomicAdd. AtomicAdd performs an atomic addition of data to the contents of memory and returns the original contents of memory from before the addition occurred. The contents of the memory being updated by the atomic operation are guaranteed not to be modified by any other assignment or atomic memory function in any shader invocation between the time the original value is read and the time the new value is written. This operation is the same as lock in OpenMP, and guarantees that the pushing operation is atomic. Using GPU makes it easy to scale the number of cores to few hundred, while still maintaining the advantage of the shared-memory model. For example, we don’t need to do message passing, and race conditions can be easily handled with atomic operations. As a result, the speedup could be significant - the GPU implementation can simulate 1 million particles for 1000 timesteps in around 10 seconds, which is difficult for all other methods. There seems to be limited ways to synchronize threads within the kernel function. Admittedly we can synchronize threads within each block, but it is impossible to synchronize across blocks. This forces us to write multiple kernel functions and call them separately, synchronizing in the main function, which may affect performance since each call introduces a significant overhead. Also, the call to GPU kernel functions returns immediately in the main function, forcing the use of cudaSynchronize Threads() function to be called every time a kernel function is called. We also recognized the effect of number of threads per block on the running speed, so we tried different numbers of threads, and we observed that each number will result in improvement in different particle number range. It seemed that choosing the number of threads per block to 64 will result in improvement for a number of particles around 1 million. The results of comparison between a serial and GPU binned code is presented in the following figure. As the number of particles increases the overhead associated with communication with the GPU becomes a small fraction of the total time and thus we are able to achieve a sublinear efficiency. Thus to be more descriptive, executing our GPU code will result in lot of time spent on communication with the GPU which will affect the slope as the number of particles increases. The following speed data is for calculating \( n \) particles in \( 1e3 \) timesteps, which shows the speed up over serial implementation.
The memory assigned in the CUDA as the following

```c
cudaMalloc((void **)&d_particles, n * sizeof(particle_t));
```

and the data is copied from cpu to GPU with the following sample code

```c
cudaMemcpy(d_particles, particles, n * sizeof(particle_t), cudaMemcpyHostToDevice);
```

The simulation procedure is defined as the following sample code

```c
for( int step = 0; step < NSTEPS; step++)
{
    cudaMemset(bin_ID, 0, NUM_BINS_PER_SIDE * NUM_BINS_PER_SIDE * sizeof(int));

    update_bins <<< linearNumBlocks, NUM_THREADS >>> (d_particles, n, 
        bin_PID, bin_ID, NUM_BINS_PER_SIDE);
    cudaMemcpy(bin_ID, d_bin_ID, sizeof(int), cudaMemcpyDeviceToHost);
    cudaThreadSynchronize();
    // compute forces
    forces_gpu <<< gridSize, blockSize >>> (d_particles, n, bin_PID, bin_ID, 
        NUM_BINS_PER_SIDE);
}
```

Figure 4.2: The speed increase of GPU over serial implementation
cudaThreadSynchronize();

// move particles

move_gpu <<< linearNumBlocks, NUM_THREADS >>> (d_particles, n, size);
cudaThreadSynchronize();

// save if necessary

if ( fsave && (step%SAVEFREQ) == 0 ){
    cudaMemcpy(particles, d_particles, n * sizeof(particle_t), cudaMemcpyDeviceToHost);
    save( fsave, n, particles );
}

The following is the flow-chart of the program:
4.7 Wall simulation

A wall can be modeled using fixed particles. With this simulation technique, one can use the same procedure of particle modeling without any change for boundary condition, with the advantage of assigning wall characteristics into the wall particles, and even have the advantage of modeling surface roughness.
Figure 4.3: The wall is modeled using the same colliding particle with fixed particles on the wall.
Chapter 5

Hybrid particle continuum model

5.1 Introduction

A complex fluids situation near interfaces is a combination of microscopic dynamics within a small localized region of the system close to the interface and the fluid dynamics in the bulk fluid region. This process arises in a wide variety of applications ranging from nanotechnology (nanofluidics) and other industrial processes, such as wetting, droplet formation, critical fluids near heated surfaces or crystal growth from a fluid phase, to biological systems, for example, membranes or biomolecules near interfaces. These systems depend on the intimate connection of many different scales: from the lower scale to the higher scale. Detailed simulations of such systems via standard classical molecular dynamics (MD) or discrete element method (DEM) are prohibitive, while continuum fluid dynamics (CFD) cannot describe the important details within the interfacial region. In view of this fact, the field of computer simulation is now faced with the need for new techniques that bridge a wider range of time- and length-scales with the minimum loss of information. A hybrid particle-continuum approach provides a resolution to this dilemma. A hybrid algorithm can deal with details within the relevant localized domain and couples this region to the continuum domain.

The general procedure is to connect the particle domain (P) and the continuum domain (C) within an overlapping region comprised of two buffers: CP and PC (see figure 5.1), which represent continuum to particle and particle to continuum domains. Within the PC buffer the particle dynamics are coarse grained to extract the boundary conditions for the C region. The most complicated part of any hybrid scheme is the CP coupling, where the microscopic dynamics need to be reconstructed to adhere to the prescriptions given by the continuum variables.

A question of central interest is to decide what kind of information needs to be transferred at CP and PC. There are essentially two possibilities, to transfer either generalized fluxes of conserved quantities or the local values of the averaged variables. Both kinds of approaches can be found in the published literature. In the context of energy transfer under unsteady flows it is not sufficient to impose the local C quantities at the boundary of P; instead, it
is necessary to couple through fluxes. Another possible benefit of flux coupling was pointed out by Flekkoy et al. [13] who stated that this procedure transcends the problem of working with fluids whose constitutive relations may be only partially or incompletely known. Hajiconstantinou and Patera [20] introduced a reservoir region to impose boundary condition on the particle region, and in the reservoir particles are given a velocity from a Maxwellian distribution with mean and velocity consistent with the velocity and temperature of the continuum flow. In order to obtain the boundary condition for the continuum region, they used a polynomial to smoothen the field variables derived from particle in $PC$ region, and in order to match the boundary condition for both particle and continuum region they used an iterative scheme for steady incompressible flows.

In this section we provide a coupling method for DPD based on the work done in [13] and [9] for molecular dynamics.

## 5.2 Overview of hybrid coupling

The domain decomposition of the hybrid schemes are depicted in figure 5.1, and in 1D in 5.2.
5.2. OVERVIEW OF HYBRID COUPLING

A typical spatial domain decomposition structure for our hybrid scheme is depicted in figure 5.1. The particle region \((P)\) contains \(N(t)\) particles at time \(t\) and it is described using standard DPD. Each particle \(i\), has a mass \(m_i\), a velocity \(v_i\) and inter-particle forces, where \(\{r\}\) denotes the ensemble of particle positions. The force acting on each particle is given by 
\[
    f_i^T = f_i^{ext} + f_i
\]
where \(f_i\) is due to inter-particle interactions and \(f_i^{ext}\) is the external force release by continuum.
The particle region is shown by \( P \) and the continuum region is shown by \( C \). The domain where particle data sets the continuum data is named \( PC \) and the domain where continuum data transfers to particles is called \( CP \).

Particle region is composed of ensemble of particles through an interactive forces, and it is evolving in time using Newtonian dynamics. On the other hand, within the continuum region the variables are the macroscopic local densities associated with the conserved quantities, density \( (\rho) \), momentum \( (j) \), and the energy \( (e) \). The conservation laws can be shown as

\[
\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho u) \tag{5.1}
\]

\[
\frac{\partial j}{\partial t} = -\nabla \cdot (ju + T) \tag{5.2}
\]

\[
\frac{\partial \rho e}{\partial t} = -\nabla \cdot (\rho eu + Tu + q) \tag{5.3}
\]

in which the energy includes the translational, thermal, and the potential energy. The momentum contains the contribution from convection \( ju \), and the tensor \( T = pI + D \).

The viscous stress tensor, satisfies the Newtonian constitutive relation as

\[
D = -\eta \left( \nabla u + \nabla u^T - \frac{2}{3} \nabla \cdot u \right) - \xi \nabla \cdot u. \tag{5.4}
\]

The energy includes convective \( (\rho eu) \), dissipative \( (Tu) \) and conductive \( (q) \), which can be expressed in terms of temperature gradient through Fourier’s Law. It can also be noted that the continuum variables are slower ones \( (\Delta t_c \gg \Delta t_p) \)
5.3 Continuum to particle coupling

Within the CP domain, the fluxes of conserved quantities evaluated from C are imposed on the particle dynamics. Each CP buffer has a volume $\Delta V_{\text{CP}}$ around the interface position $x = x_{\text{CP}}$. We use the idea of Delgado-Buscalioni & Coveney [8] implementation in the general case of mass, momentum and energy transfer in unsteady flows with some changes to fit our problem. An improved version of the heat transfer scheme has been proposed by Flekkoy et al.[14]. In what follows we briefly sketch the CP protocol in order to clarify what are the coupling variables required in the exchange.

The momentum flux across $x = x_{\text{CP}}$ is given by $(p_{\text{CP}} I + T_{\text{CP}}) \cdot n_{\text{CP}}$, where the surface vector $n_{\text{CP}} = -n$ points towards $P$. This flux is introduced within the particle dynamics by adding an external force $f_{\text{ext}} = A(P_{\text{CP}} + T_{\text{CP}}) \cdot n_{\text{CP}}/N_{\text{CP}}$ to $N_{\text{CP}}(t)$ particles within $\Delta V_{\text{CP}}$, so if the particle is outside the area the external force is zero. In order to maintain a desired density $\rho_c$ within the particle system, the hydrostatic pressure needs to be given by the correct equation of state, $P_{\text{CP}} = P(\rho_c, T_c)$. In summary, the momentum flux requires specification of the pressure tensor $P$ from C.

5.3.1 Imposing fluxes

Following Flekkoy et al. [13] in the CP region we shall communicate fluxes of conserved quantities. These fluxes correspond to mass, momentum, and energy transfers through the outer interface of the CP cell the left surface in Fig. 5.1. Flekkoy et al. [13] obtained these fluxes from the values of the continuum variables at the center of the control cell, $x = x_0$, instead of at the exact position of the CP interface, $x = x_W$. Thus, it is essential to take into account this apparently unimportant technicality when dealing with unsteady scenarios. Let us consider a general conservation equation, with

$$\frac{\partial \phi}{\partial t} + \nabla \cdot J_\phi = 0 \quad (5.5)$$

where $J_\phi$ is the flux of $\phi$. Integrating over cell C

$$\frac{\partial}{\partial t} \int_V \phi dV + \int_S J_\phi \cdot n ds = 0 \quad (5.6)$$

we only analyze the 1D situation

$$\frac{\partial}{\partial t} \int_V \phi dV - AJ_{\phi E} \cdot n_w = -AJ_{\phi W} \cdot n_w \quad (5.7)$$

where $A$ is the cross section area, and the subscripts E (east) and W (west) denote that the variables are measured at $x = x_E$ and $x = x_W$, respectively. The surface vectors $n_W$ and $n_E$ are at west and east of the CP domain in figure 5.1. The right-hand side of equation 5.7 is
the flux current of $\phi$ through the interface $W$ of the control cell, which is precisely the force we want to introduce on the particles at the $CP$ buffer. We note that only under steady flows does $J_{\phi W} = J_{\phi 0}$. Hence only in this case does the evaluation of the fluxes at $x_W$ using the continuum variables at $x_0$ lead to the same converged steady state as if the variables at $x_W$ were used, although the transients may of course differ. It is possible to provide an estimate of the global error arising from evaluating the flux at a position $x_0$ shifted $\delta X$ with respect $x_W$, over a certain time interval $t$. As a crude estimate, the accumulated error of the cell-averaged momentum $j \cdot n = \rho u \cdot n$ is of order $\rho e \Delta u \approx \Delta J \Delta t / \Delta X$. Simulations carried out with the momentum flux evaluated at $x_0$ yield relative errors of the averaged velocity at CP of the same order of magnitude as this estimate so that in order to evaluate the fluxes pertaining to the CP exchange it would first be necessary to make use of simple interpolation techniques.

The fluxes arising from the continuum on the RHS of Eqs. 5.7 are imposed on the particle ensemble at the CP cells through expressions involving atomistic variables

\begin{equation}
ms = -A \rho u \cdot n \tag{5.8}
\end{equation}

\begin{equation}
ms < v' > + \sum_{i}^{N_{CP}} F_{i}^{ext} = -A(\rho uu + T) \cdot n \tag{5.9}
\end{equation}

\begin{equation}
ms < e' > + \sum_{i}^{N_{CP}} F_{i}^{ext} \cdot v_i > - < J_{Q}^{ext} > \cdot n = -A(\rho u e + T \cdot u + q) \cdot n \tag{5.10}
\end{equation}

in which, $s(t)$ is the number of particles inserted or removed, $v'$ is the velocity of the particles inserted or removed, $F_{i}^{ext}$ is the external force applies on each particle, $< e' >$ is the energy of particle inserted or removed, and $< J_{Q}^{ext} >$ refers to an externally imposed heat flux.

### 5.3.2 Momentum

The condition $< v' > = u$ ensures the balance of momentum convection. If the mass flux points towards the P region, this condition is fulfilled by choosing the velocity of the inserted particles somehow that the above condition is fulfilled, in the DPD we set velocity of particles equal to velocity of the continuum. Concerning particle removal ($s < 0$), we note that if the average velocity at the $CP$ cell is equal to the continuum velocity $< v > = u$, then the average velocity of the subset of extracted particles. would be precisely $< v' > = < \tilde{v} > = u$. Hence the condition of velocity continuity at $CP$ is needed to ensure the correct balance of momentum convection. Therefore to satisfy the momentum exchange we need to set the external force to satisfy the above equation. Here we distribute the external force uniformly as

\begin{equation}
F_{i}^{ext} = -\frac{ATn}{N_{CP}}. \tag{5.11}
\end{equation}
5.3.3 Energy

The balance of advected energy requires that $<\epsilon'_{\ast}> = \epsilon$, which must be satisfied by picking the temperature of the particles. The dissipation energy can be satisfied by

$$<\sum_i^{N_{CP}} F_{i}^{ext} \cdot v_i> = -ATu \cdot n$$

(5.12)

since we are picking the same external force for all particles then,

$$F_{i}^{ext} <\sum_i^{N_{CP}} v_i> = N_{CP} F_{i}^{ext}. <\bar{v}> = -ATu \cdot n,$$

(5.13)

so the above equation is already satisfied by picking $F_{i}^{ext} = -\frac{ATu}{N_{CP}}$, and $<\bar{v}> = u$.

5.4 Particle to continuum coupling

At the $PC$ interface, the flux measured in the particle system must be imposed onto continuum as a flux boundary condition. The particle flux transferred to $C$ has to be averaged over the local time and length scales of the continuum level. The mean value of any particle quantity $\phi_i$ within a volume $\Delta V$ of a cell centered at position $R$ is $\phi_R(T) = \sum_{i=1}^{N_R} \phi_i(t)/N_R$, where the sum is made over the $N_R$ particles within $\Delta V$. The coarse-grained quantity is defined by time-averaging over $\Delta t_{av}$

$$\bar{\phi}_R(t) = \int_{0}^{\Delta t_{av}} \phi_R(t - \xi) d\xi/\delta t_{av}$$

(5.14)

In a general case of mass, momentum and heat exchange within a single component fluid, the boundary condition for the $C$ domain requires the following averaged fluxes across the $PC$ interface: momentum flux: $\bar{j}_{PC} \cdot n$ and heat flux $\bar{q}_{PC} \cdot n$.

Expressions for these fluxes in terms of particle variables are;

$$\phi_i^{mass} = \frac{1}{V_{PC}} \left( \sum_j^{N} m v_i \right) \cdot n_{PC}$$

(5.15)

$$\phi_i^{momentum} = \frac{1}{V_{PC}} \left( m v_i v_i - \frac{1}{2} \sum_j^{N} r_{ij} f_{ij} \right) \cdot n_{PC}$$

(5.16)

$$\phi_i^{energy} = \frac{1}{V_{PC}} \left( m \epsilon v_i - \frac{1}{2} \sum_j^{N} r_{ij} v_i f_{ij} \right) \cdot n_{PC}$$

(5.17)

On the other hand, as explained by Delgado-Buscalioni et al. (2005) and Delgado-Buscalioni & Coveney (2003a), in order to ensure variable continuity across the $PC$ interface, one need to specify the average particle velocity at the overlapping cell $\bar{v}_P$, the average density $\bar{\rho}_P$ and temperature $\bar{T}_P$. The mass flux across the $PC$ interface can be obtained from $\rho_P \bar{v}_P$. 
Chapter 6

Results

In this chapter we deliver some of the results from the simulation using the dissipative particle dynamics and its coupling to continuum model. At first the flow in a channel is considered and the input parameters are fit to the flow in a channel solution. Afterwards, a parameter study is done on the flow in a channel to figure out the extents of validity of the results, and the flexibility of the fitted parameters. Next, the flow on an impinging jet, flow around cynliners, and flow on turbulators are solved using the method.

6.1 Flow in a channel

The first test case chosen is the force convection of flow in a channel. The model of the channel is represented in figure 6.1.

As it can be seen from the figure 6.1 the force convection in a channel flow is studied. The dimension of the channel is set as a square of 1 mm. 20000 particles are assigned inside the domain with constant radius of \( R/L = 1e^{-4} \), in which \( R \) is the radius (half the critical distance of particles), \( L \) is the length scale (1mm in this case). The external force of 0.01mN is put on the particles, and the particles are left to flow in the channel. The wall is contained
of particles in proximity to each other with the fixed temperature of 310 K, and the initial temperature of the flow is 300 K. The computation was first performed with constant fluid temperature until the fully developed flow field was established (about 200,000 time steps). Each timestep is 1e-4 seconds. First, the dynamic parameters are fitted to establish a fully developed profile for the flow, and then the energy transfer parameter $k_t$ is fitted to deliver the Nu (Nusselt) number for flow in a channel. By setting $\alpha = 0.5184, \gamma = 0.4401, k_t = 24.09$ the following results are established from the simulation.

An snapshot of particles of the simulation in one timestep is presented in figure 6.2, and

![Figure 6.2: A sample of particles on the domain colored by temperature (T)](image)

In order to contour plot the results, the particles values are averaged in 20 by 20 grid, and also the time average is done in the last 1000 timesteps to get the local values. The velocity and temperature contour plots is presented in 6.3, and 6.4.
6.1. FLOW IN A CHANNEL

The flow in a channel test was done multiple times with random initiation of the particles in order to find out the initiation and propagated error in the particle simulation. These results were then compared to the theoretical result from flow in the channel. In figure

Figure 6.3: Contour plot of flow in a channel for velocity (m/s)

Figure 6.4: Contour plot of flow in a channel for temperature (T)
6.1. FLOW IN A CHANNEL

6.5, and 6.6 the error over the multiple runs of the model with randomized initiation are represented as error bars on the curve. The maximum error in the current results is around 15%. The Nu number target for fitting parameter is 7.541. Based on the multiple tests done with the mentioned parameter, the average Nu number is achieved with maximum error of 6 percent. \( \text{Nu} = 7.541 \pm 0.45 \). After fitting the parameters based on the test case of ow in a channel. The coupled continuum-particle model is used to calculate Nusselt number in some specific applications. The next set of tests were done on flow in an impinging jet and flow around a cylinder, and over the turbulator. In these examples we tried to couple the particle model with the CFD steady state calculation as mentioned in the previous section. However, before that we analyze the effect of different parameters on the results to find out the flexibility of the fitted parameters and the sensitivity of the results on input parameters.

![Figure 6.5: Velocity over mean velocity for flow in a channel. The error bars show the maximum and minimum errors for all the runs](image-url)
After fitting the parameters based on the test case of flow in a channel, the coupled continuum-particle model is used to calculate Nusselt number in some specific locations. The next set of tests were done on flow in an impinging jet and flow around a cylinder. In these examples we tried to couple the particle model with the CFD steady state calculation as mentioned in the previous section.

The continuum model is used to calculate the general domain, and in some hot spots particle model is used with the one way coupling. The one way coupling of Continuum to Particle, described in the previous section, is used to assign the fluxes from continuum model to the particle one. In the picture 6.7 the hatched area is the continuum to particle coupling zone and the inside rectangle with particles is the particle simulation zone. The method is explained in chapter 4.

In figure 6.7 in the hatched area the fluxes from continuum model are transferred to the particle model. In the rectangular section in the middle of the figure the particles are moving and having inter-particle force and energy transfer based on the edpd model. The heat transfer is calculated to the particles on the wall to calculate the Nusselt number.
6.1. Parameter study

The effect of input fitting parameters, and particle size on the results are shown in this section. At first while keeping all the parameters constant we keep changing the parameter force constant $\alpha$. The results are presented in figure 6.8. In the figure 6.8 the velocity in the direction of the flow is normalized by the maximum velocity obtained by $\alpha = 0.5184$. By increasing the value of the force constants the interacting force between the particles increases and the momentum of the particles dispered in the cross section. As presented in the figure 6.8 the velocity profile becomes more flattened by increasing the constant, and by reaching $\alpha = 4$ the solution starts to get unstable. As the interaction decreases, the weak interaction result in reducing the dampening effect of interacting particles, so the peak velocity increases.
The friction coefficient is changed while all other characteristics kept constant. Lowering the friction coefficient result in increasing the peak velocity. By increasing the coefficient, the velocity reduced to more dispersed profile.
6.1. FLOW IN A CHANNEL

The next parameter, is the radius of the particles (cutoff radius). The parameters are fitted to the particle radius of radius over length of 0.001 \((r/l = 0.001)\). The flexibility of the parameters are studied by changing the radius. The results shows that, increasing the radius result in more interaction between the particles and more dispersion of velocity over the domain, so the velocity profile becomes flatter. On the other hand, reducing the radius will decrease the particle momentum exchange and reducing the damping of the velocity, which increase the peak velocity. Furthermore, if we increase the \(r/l\) ratio by 50 percent the result shows the interaction becomes higher and the solution start to become unstable.

\[\begin{align*}
\text{Figure 6.10: Effect of relative particle size on the flow pattern}
\end{align*}\]

6.1.2 Impinging jet

Impinging jets is an effective way to transfer energy in many applications. When a jet of flow hit a surface, the flow can efficiently transfer large amounts of thermal energy between the surface and the fluid. Heat transfer applications of impinging jet can be enlisted as cooling of stock material during material forming processes, heat treatment, cooling of electronic components, heating of optical surfaces for defogging, cooling of turbine components, cooling of critical machinery structures, and many other industrial processes. Typical mass transfer applications include drying and removal of small surface particulates [40]. The procedure for calculating the heat transfer is done for impinging jet. The CFD calculation is presented in the next figure for \(Re = 23000\) with the turbulent model \(\nu^2 - f\) for two cases of distance of impinging jet to plate over diameter of the jet equals 2 and 6 \((L/D = 2, L/D = 6)\) in four locations. In figure 6.11 the continuum model solution is provided, and in the areas designated with the black squares the particle model is used to calculate the heat transfer,
and consequently the Nusselt number. Therefore, in this calculation technique, the data needed for particle model is transferred from the continuum one through the buffer zone.

Figure 6.11: CFD model of the impinging jet with sample boxes showing where the particle code is implemented

The result of the problem for two cases are presented in the following graphs

Figure 6.12: The fitted data for $L/D = 2$, dots are result of the DPD and blue line is the experimental data
6.2. FLOW AROUND A CYLINDER

The next application that we try to test the method on is the flow around the cylinder. Two sets of experimental data is considered for this purpose. First, the flow around a single cylinder, and second, the flow around an array of cylindrical fins. In the following picture the CFD model of flow over a single cylinder is presented on the Reynolds number Re=34000.

As it can be seen from the above figures, the result predicts the trend of the graph and it also predicts the results by maximum 10 percent error. These first set of result with coupling techniques shows the method is capable of predicting the real industrial applications with good approximation.

6.2 Flow around a cylinder

As it can be seen from the above figures, the result predicts the trend of the graph and it also predicts the results by maximum 10 percent error. These first set of result with coupling techniques shows the method is capable of predicting the real industrial applications with good approximation.

Figure 6.13: The fitted data for $L/D = 6$, dots are result of the DPD and blue is the exp. data

Figure 6.14: CFD modeling of flow around a cylinder p(Pa)
The test case to compare result is an experimental investigation by Kraabel and Baughn [26] for two different Reynolds number (Re=34000, and Re=106000). The Heat transfer data is presented in figure 6.17, and 6.18.
The other data that is tested for flow over a cylinder is the experimental investigation done by Ames, which is the experimental investigation on fin arrays [5], and [4]. The Cfd calculation is done with the continuum code and again the particle code is implemented in
special hot spots. The CFD data is presented in the following figures for \( Re = 3000 \), and the modeling is done using turbulence model \( v - 2f \)

![Figure 6.19: The simulated CFD data of Ames experiment (Pressure(Pa))](image1)

![Figure 6.20: The simulated CFD data of Ames experiment (Temperature(K))](image2)

![Figure 6.21: The simulated CFD data of Ames experiment (Velocity(m/s))](image3)

The following figure is the pressure drop over fins and CFD result is compared with the Ames experiment. As it can be seen the picture shows good similarities between the results.
6.2. FLOW AROUND A CYLINDER

Figure 6.22: The pressure drop result from CFD comparing to pressure drop of Ames’ experiment 1st row

The below figure shows the heat transfer data calculated for both CFD and DPD simulation.

Figure 6.23: The heat transfer data of particle model for Ames experiment comparing to Ames’ experiment 1st row
6.3. FLOW ON A TURBULATOR

Same procedure is done for the second row of the fins. As it can be seen in the graphs, the pressure drop results from the CFD model shows good agreement with the experimental one. The heat transfer coefficient results are also showing good behavior in the frontal face of the cylinder. However, in the wake area behind the cylinder, the CFD result, and consequently the particle model result are not very close to experimental one. The reason behind this is the particle model is too small to enhance the results from larger scale CFD model. From the result obtain on several models presented here, it can be concluded that this method has very promising results for predicting heat transfer in specific areas. These results also show that one set of fitting parameters can work on some different range of problems, which in turn shows the versatility of the model.

![Graph](image)

Figure 6.24: The pressure drop result from CFD comparing to pressure drop of Ames’ experiment 2nd row

6.3 Flow on a turbulator

The next case that we consider for testing is the flow in a channel with turbulator. In the following picture the grid used to model the CFD simulation is shown. The boundary layer grid is used near the wall to predict the flow using v2-f turbulent model.
Figure 6.25: The heat transfer data of particle model for Ames experiment comparing to Ames' experiment 2nd row

Figure 6.26: The grid used in CFD model.
Figure 6.27: The cross section of the grid used in CFD model.

The result of the CFD simulation is shown in the following figure for the Reynolds number \( Re=20000 \).

Figure 6.28: CFD modeling of flow in a channel with turbulator \( v(m/s) \)

The test case to compare result is an experimental investigation by Han et. al. [21]
for Reynolds number (Re=20000) in one spot between two fins. The Heat transfer data is presented in figure 6.29.

![Figure 6.29: Comparison of the Nu between particle and CFD and experiment in channel with turbulator](image)

The particle model is also used to model larger domain in this example. The model is the area from the middle of two consecutive fins. An snapshot of the 3D particle model is shown in figure 6.30.
6.3. FLOW ON A TURBULATOR

In order to make sense of the data, we present the velocity in the plane in the middle of the channel. The $v_x$ and velocity magnitude of CFD model and particle model is presented in figure fig:turbcfdvx, fig:turbpvx, fig:turbcfdvm, and fig:turbpvm.

Figure 6.30: A snapshot of the particles over the turbulator

Figure 6.31: $v_x$ for the CFD model in the plane in the middle of channel
6.3. FLOW ON A TURBULATOR

Figure 6.32: $v_x$ for the particle model in the plane in the middle of channel

Figure 6.33: $v$ for the CFD model in the plane in the middle of channel
The heat transfer on surface is presented in figure 6.35. The data is average over 1000 timesteps and compared to the CFD result over the same domain. The particle model is predicting the heat transfer with a good precision.
6.4 Transient simulation

The transient simulation is done for flow in a channel to see if it is feasible to calculate transient simulation with this method. The particle code is used coupled with a CFD code to check the feasibility of transient modeling. The continuum code used in this study is OpenFVM which uses Metis to divide the domain and Petsc to solve the problem. The speed of the calculation is tested on Stampede (6th fastest computer in the world) on 128 cores and one GPU. The particle test case is for 10000 time steps. The simulated model and speed data of the problem is presented in the following figures.

![Sample picture of the test case (Flow in a channel) with magnified particle model](image)

Figure 6.36: Sample picture of the test case (Flow in a channel) with magnified particle model
After testing the accuracy of the model, we test the speed of the calculation in Stampede. The TACC Stampede system is a 10 PFLOPS (PF) Dell Linux Cluster based on 6400+ Dell PowerEdge server nodes, each outfitted with 2 Intel Xeon E5 (Sandy Bridge) processors and an Intel Xeon Phi Coprocessor (MIC Architecture). The aggregate peak performance of the Xeon E5 processors is 2+PF, while the Xeon Phi processors deliver an additional aggregate peak performance of 7+PF. The system also includes a set of login nodes, large-memory nodes, graphics nodes (for both remote visualization and computation), and dual-coprocessor nodes. Additional nodes (not directly accessible to users) provide management and file system services.

The speed testing for the MPI part is done using up to 128 computational processors. The weak and strong scaling is presented in the following graphs. Also, the GPU speed is tested in the same time for different particle numbers, and it can be seen in figure 6.37.
6.4. TRANSIENT SIMULATION

Figure 6.38: The weak scaling of the CFD problem for one time step

Figure 6.39: The strong scaling of the CFD problem for one time step

In the above figures it can be seen that using $1e5$ particles for calculating heat transfer in a designated region the time of calculation is not more than that of a regular MPI one
million elements simulation. Therefore, by combining the GPU into the calculation, we are getting more accuracy without increasing the cost.

6.5 Conclusion

The dynamics of a fluid close to interfaces is affected by its microscopic structure. Because the molecular level dynamics is playing a role, and macroscopic scale is effected by lower levels, in these situations the fluid cannot be completely described by continuum theory and Navier-Stokes equation.

In this project, a new method of calculating heat transfer is introduce using a coupling technic between two different scales. For the lower scale simulation the dissipative particle dynamics (DPD) is used, and the data between the continuum model and particle is transfered via transfering the flux. We presented a coupling method of dissipative particle dynamics to continuum model to calculate heat transfer. The particle model parameters is fitted to deliver the required heat transfer and dynamic behavior of flow in the channel. Afterwards, the coupled model is used to solve a few test cases, impinging jet flow, flow around a cylinder, and flow in a channel with turbulator. Furthermore, the transient simulation is done for flow in a channel to see if it is feasible to calculate transient simulation with this method. The particle code is used coupled with a CFD code to check the feasibility of transient modeling. In the impinging jet case the result predicts the trend of the graph. Besides, it calculates the results with a maximum 10 percent error. In the case of flow around the cylinder the model predicts the frontal face of the cylinder very accurately. In the back side of the cylinder the particle model as well as the CFD model predict the trend of the heat transfer well, but the magnitude of heat transfer is calculated with some errors even though it enhances the predictions of the CFD model.

The flow in a channel with turbulator is also tested, and the heat transfer calculated between the two fins are simulated with the error of less than 3 %. The bigger size of particle model is also tested and it predicts the flow behavior of the CFD model well. The transient simulation is done for flow in a channel to see if it is feasible to calculate transient simulation with this method. The particle code is used coupled with a CFD code to check the feasibility of transient modeling. The speed of the particle model is in the same order as CFD model, which shows this model can be used in transient simulations.
Bibliography


