UNIVERSITY OF Southampton

University of Southampton Research Repository
ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

http://eprints.soton.ac.uk

University of Southampton
Faculty of Engineering and the Environment

Hierarchical modelling of multiphase flows using fully resolved fixed mesh and PDF approaches

by
Sina Haeri

A thesis submitted for the degree of
Doctor of Philosophy
November 2012

University of Southampton

Abstract

Faculty of Engineering and the Environment

Doctor of Philosophy

Hierarchical modelling of multiphase flows using fully resolved fixed mesh and PDF approaches

by Sina Haeri

Fully–resolved simulations of multiphase flow phenomena and in particular particulate flow simulations are computationally expensive and are only feasible on massively parallel computer clusters. A 3D SIMPLE type pressure correction algorithm is implemented and extensively tested and parallelized to exploit the power of massively parallel computing clusters currently available. Domain decomposition and communication schemes applicable to a general unstructured or structured multi–block CFD codes are discussed and algorithms are proposed, implemented and tested. Several high–performance linear solvers and a multi–grid strategy for the current framework are implemented and the best types of solvers are identified.

A 2D CFD code is developed by the author to test several possible fixed–mesh strategies. Variations of immersed boundary (IB) and fictitious domain (FD) methods are implemented and compared. FD methods are identified to have better properties especially if other transport phenomena are also considered. Therefore an FD method is adapted by the author for the SIMPLE type flow solvers and is extended to heat transfer problems. The method is extensively tested for the simulation of flow around stationary in addition to freely moving particles and forced motion where both natural and forced convection are considered. The method is used to study the flow and heat transfer around a stationary cylinder and a new high resolution correlation is devised for the estimation of the local Nusselt number curves. Free fall problem for a single circular cylinder is considered and the effects of internal heat generation and also long term behavior of single cold particle subject to natural convection are also studied in detail. A particle collision strategy is also adapted and tested for the particle–particle collision problems. The FD algorithm is extended to the 3D framework and the flow around single stationary sphere and also free fall of a single sphere are used to validate the FD algorithm in 3D.

A unique polydispersed fluid-particle turbulent modelling process is reviewed and the closure problem for this framework is studied in detail. Two methods for the closure of the non–integer moments which results from the polydispersity of the particles are proposed namely PDF reconstruction using Laguerre polynomials and a unique direct method named Direct Fractional Method of Moments (DFMM). The latter is derived using the results of the fractional calculus by writing an equation for the fractional derivatives of the moment generating function. The proposed methods are tested on a number of problems consisting of analytical, experimental and DNS simulations to asses their validity and viability which shows that both methods provide accurate results with DFMM having more desirable properties.

Dedication

I dedicate this thesis to my family, especially ...

to my beloved wife, Parvaneh, for her support, patience and understanding

to my parents for their continuous encouragements throughout my entire life.

d

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

A	Stochastic force on a particle, drift coefficient, Particle acceleration
B	Diffusion matrix
C	Specific heat capacity
E	Graph Edge
F	Force
f	Body force in NS equations, friction factor
G	A Graph, Moment generating function
g	Gravity
I	Moment of inertia tensor, Modified Bessel function
J	Jump Process
j	Colburn j–factor
K	Permeability
m	mass
N	Number of Lagrangian points in IB method, Number of material CVs in FD method
P	Pressure
r	Position Vector
S	Rate of strain tensor
s	Arc length
T	Temperature, Torque
U	Velocity Vector
V	Graph Vertex Vector
W	Wiener process
X	Position Vector (Center), Stochastic process
x	A realization of a stochastic process

Greek Symbols

α	Adjustable parameter in VB method
β	Adjustable parameter in VB method
ϵ	Turbulent dissipation, error, coefficient of restitution
ϵ_{ijk}	Permutation symbol
Γ	Boundaries of the computational domain, Gamma function
κ	thermal conductivity, Spring constant
μ	Viscosity
ν	Kinematic Viscosity
Ω	Computational Domain
ω	Angular velocity
ρ	Density
ξ	Collision constant

Superscripts

E	Eulerian
e	Equilibrium
L	Lagrangian
n	n^{th} time point
R	Rigid body
r	Reduced Property

Subscripts

d	Entire computational domain
f	Fluid property
i	i^{th} Vector component, used for tensor notations

j	j^{th} Vector component, used for tensor notations
l	Fluid property
m	m^{th} Particle
n	n^{th} Particle, counter, surface normal vector
p	Particle property
s	Seen velocity
sg	Superficial gas velocity
sl	Superficial liquid velocity

Other Symbols

$\langle\cdot\rangle$	Statistical or ensemble average, Defined as the average over 'large' number of realizations
$\mathcal{A}$	Force generating function
$\mathcal{M}$	Kummar (confluent hypergeometric) function
$\mathcal{N}$	NS operator, Normal distribution
$\mathcal{P}$	Probability Density Function (PDF)
$\mathcal{V}$	Volume
$\mathscr{N}$	Discrete NS operator
$\tilde{\cdot}$	Filtered quantity

Acronyms

ALE	Arbitrary Lagrangian Eulerian
BFGS	Broyden-Fletcher-Goldfard-Shanno
BiCG	Bi-Conjugate Gradient
BiCGSTAB	Bi-Conjugate Gradient Stabilized
CFD	Computational Fluid Dynamics
CG	Conjugate Gradient
CGS	Conjugate Gradient Squared
CK	Chapman-Kolmogorov
CT	Communication Time
CV	Control Volume
DDF	Discrete Direct Forcing
DLM	Distributed Lagrange Multiplier
DNS	Direct Numerical Simulation
DSD	Deforming Spatial Domain
EE	Eulerian - Eulerian
EL	Eulerian - Lagrangian
FBM	Fictitious Boundary Method
FD	Fictitious Domain
FDM	Finite Difference Method
FEM	Finite Element Method
FFT	Fast Fourier Transform
FRS	Fully Resolved Simulation
FVM	Finite Volume Method
GL	Grunwald-Letnikov
GMRES	General Minimalised Residual
IBM	Immersed Boundary Method
IC	Incomplete Cholesky factorization
ILU	Incomplete LU factorization
LES	Large Eddy Simulations
MDF	Mass Density Function
MGF	Moment Generating Function
MPI	Message Passing Interface
MSI	Modified Strongly Implicit procedure
NS	Navier-Stokes
NSB	Navier-Stokes-Brinkman
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PMD	Particle Mass Distribution
PSIC	particle-source-in-cell
QMR	Quasi-Minimalised Residual
RANS	Reynolds Average Navier-Stockes

RL	Rienmann-Liouville
RMS	Root Mean Square
SDE	Stochastic Differential Equation
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SIMPLEC	Semi-Implicit Method for Pressure-Linked Equations - Consistent
SIMPLER	Semi-Implicit Method for Pressure-Linked Equations - Revised
SIP	Strongly Implicit Procedure
SOR	Successive Over Relaxation
SSE	Residual sum of squares
SST	Stabilized Space Time
SST	Total sum of squares
VB	Virtual Boundary

DECLARATION OF AUTHORSHIP

I, Sin Haeri, declare that the thesis entitled **Hierarchical modeling of multiphase flows using fully resolved fixed mesh and PDF approaches** and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as given in the list of publications.

Signed:

Date:

List of Publications

Journal Publications

- Haeri S., Shrimpton J.S., (2012) *Closure of non-integer moments arising in multiphase flow phenomena* , Chemical Engineering Science, 75:424–434.
- Haeri S., Shrimpton J.S., (2012) *On the application of immersed boundary, fictitious domain and body-conformal mesh methods to many particle multiphase flows*, International Journal of Multiphase Flow, 40:38–55.
- Haeri S., Shrimpton J.S., (2011) *A mesoscopic description of polydispersed particle laden turbulent flows*, Progress in Energy and Combustion Science, 37:716–740.

Conferences

- Haeri S., Shrimpton J.S., *CFDComm: An Optimized Library for Scalable Point-to-Point Communication for General CFD Applications*, HPCC 2012, AHPCN Symposium, Liverpool, UK.

Acknowledgements

I would like first to thank my supervisor Dr John Shrimpton for his valuable advise and introducing me to several different topics. In addition I would like to thank Professor Kai Luo and Dr Edward S Richardson for their valuable comments during my studies at the University of Southampton.

1

Introduction

The final goal in many fluid engineering problems is to improve the quality of a product, reduce the amount of unwanted by–products and to obtain higher production rates using less resources and emit less pollution. This is usually achieved by extending our understanding of underlying processes which generally consist of momentum, heat and mass transfer. Traditionally a flow system is studied by creating a similar, scaled down version of that system or device, measuring the required variables and stating them as empirical or semi empirical equations. These are subsequently used for similar systems using dimensional analysis and scale up procedures. However, especially in the last two decades, computer simulation of flow systems using computational fluid dynamics (CFD) techniques has been accepted as a design tool. CFD methods are especially useful when studying critical situations where performing experiments is dangerous or extremely difficult.

Complexity of the underlying physical phenomena and the range of scales that need to be resolved in order to fully determine the real life systems is usually such that a direct numerical simulation becomes prohibitively expensive in terms of computer resources. In such cases models are introduced into the numerical procedure to reduce the degree of freedom of the system or range of scales, which makes the simulation feasible by sacrificing accuracy. Having said that, the properties of numerical solution such as availability of data in any (solved) point in space and time, easy visualization and manipulation of data puts these methods in the centre of attention especially as a research tool where computation cost and resources are not of primary concern.

There are several steps for a successful CFD simulation. First the system and its governing equations should be understood and any simplifying assumptions should be applied to these equations; then the physical domain on which the equations hold should be produced. The second step is the discretization of both physical domain (grid generation) and differential equations with appropriate schemes (FDM , FVM, FEM or spectral) which results is a system of linear equations that should be solved to yield the solution on the discretized domain or grid points. Final step would be to analyse the numerical solution in terms of error and accuracy. In this thesis the particle–flow systems and their physics and solutions methods are studied, especially those that contain no further modelling than those assumed in the derivation of the underlying equations. This should be done on a grid that resolves all scales of the system and is usually referred to as Direct Numerical Simulation or DNS. It is also shown how each of the aforementioned steps should be applied for a successful DNS simulation of particle–flow systems.

1.1 Motivation

Particulate flows are encountered in many engineering and environmental systems. For example, controlling the combustion which results in efficient fuel consumption, is only possible by understanding the atomization, dispersion and evaporation processes of fuel droplets in the internal combustion engine. On the other hand, pollutant transport in the atmosphere can be considered as the other extreme of the spectrum. Particle separation in cyclones and filters, hydraulic conveying, liquid–solid separation, particle dispersion in stirred vessels, spray drying and cooling, mixing of immiscible liquids, liquid–liquid extraction, bubble columns, aeration of sewage water and flotation are only some of the industrial processes involving a continuous (may also be turbulent) and a dispersed phase.

There are many parameters such as coalescence and breakup of bubbles [1], turbulence modulation [2], clustering and preferential accumulation [3, 4] that affect the efficiency of processes involving particulate flows. In some processes we might need to increase a flow parameter while in another a minimization of the same parameter might be essential, for instance in internal combustion engines a large inter–particle separation implies large fuel evaporation rate and better fuel/air mixing and hence results in less

soot production and more efficient combustion[5, 6] while for floating or separation processes a small inter–particle distance is needed to alleviate the clustering and separation of particles. There are many other complicated phenomena that occur in such systems involving inter–phase heat, mass and momentum transfer which are not yet fully understood. Theoretically all these inter–phase phenomena can be added to the simulation but the main difficulty is the presence of a very large range of scales in a turbulent flow, which gets even larger for a multiphase system, that makes the simulation extremely challenging. Thus providing a general numerical framework in a DNS sense to study these systems is crucial for the understanding of the complexities associated with them.

Direct numerical simulation (DNS) of turbulent single phase flows with spectral methods due to their desirable numerical properties, i.e no dissipation and dispersion error from the discretization [7] and available FFT numerical algorithms with $O(nlog^n)$ complexity, were used extensively, see [8] for a review. However in many particulate flow DNS simulations, a DNS code is used for the fluid part and a point–particle assumption, is used to track the particles. In one way coupling the influence of the particles on the flow field and particle–particle interactions are entirely disregarded and the results should be used with utmost care. In two–way coupling there is a feedback force from particle to the flow field which is calculated with the assumption of an undisturbed fluid field. Again a point–particle assumption where spatial extensions of the particle are not considered is used. Therefore the method is unable to capture very important phenomena such as particle rotation or wake effects which can cause totally different flow structures in some conditions. Even in the case of four–way coupling the flow around the particle is not fully–resolved and only the collision effects are taken into account.

There are however a number methods for fully resolving the flow around the particles and considering their finite dimensions which are computationally very expensive but at the same time necessary for understanding the complex phenomena. Moreover for a reliable application of an engineering approach to a multiphase simulation, i.e Eulerian–Eulerian or Eulerian–Lagrangian, modelling of the relevant physical mechanisms affecting the particle motion such as turbulent transport of particles, wall interactions, collisions and agglomeration, is needed. In some cases the physical phenomena are far too complicated to allow for a derivation of the model from basic principles of physics (e.g. particle agglomeration). Therefore, detailed numerical simulation are

required to analyse the phenomena and propose reliable empirical or semi empirical models to be used for engineering simulations.

The field of multiphase flow is extremely wide and can be classified and studied from many perspectives, as a result there are many different simulation techniques available, each applicable to specific subclasses and phenomena. Thus a brief discussion on the different types of the multiphase flows seems necessary at this early stage.

1.2 Multiphase Flows

The term two–phase flow refers to the situation where a mixture is formed either by two immiscible fluids or a fluid and a solid phase. If the mixture consists of a fluid and a solid phase, the solid phase will usually appear as small particles floating inside the fluid phase, in this case the fluid phase is connected continuously and is usually referred to as carrier phase and the solid phase as dispersed. The dispersed phase can be further classified into two different classes: mono– and poly–dispersed depending on the size distribution of particles. Different flow regimes can be categorized by considering a dense particulate phase, for instance a fluidized bed. Figure 1.1 adopted from Grace [9] shows the different regimes for a gas–solid suspension.

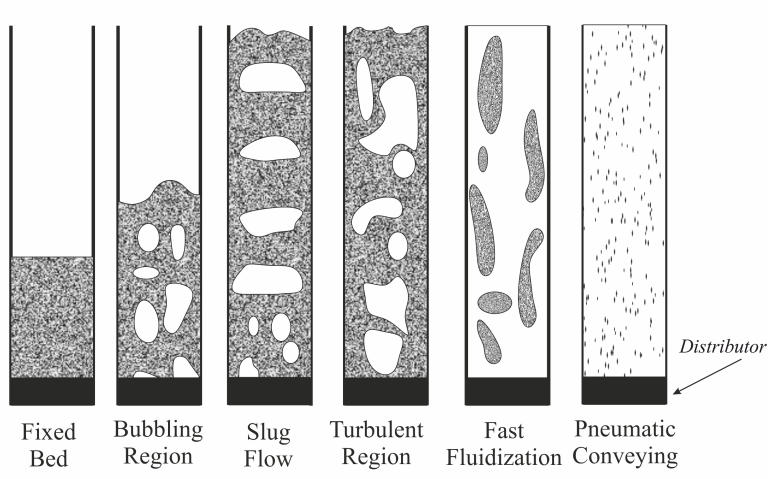


Figure 1.1: Gas-Solid flow regimes - Different regimes observed by increasing gas velocity from left to right [9]

The properties of the gas–solid regimes can be summarized as follows [9–12]:

- Fixed bed regime ($0 < U < U_{mf}$): Particles are quiescent, pressure drop increases by increasing velocity.
- Bubbling Fluidization ($U_{mf} < U < U_{ms}$): Voids form near the distributor and grow by coalescence, large irregular pressure fluctuations, well defined surface.
- Slugging Fluidization ($U_{ms} < U < U_c$): Voids fill the most of the cross section, rise and collapse of the top surface with regular frequency, large and regular pressure fluctuation.
- Turbulent Fluidization ($U_c < U < U_{se}$): Small voids and particle clusters are formed and destroyed, small pressure fluctuations, hard to distinguish the upper surface.
- Fast Fluidization ($U_{se} < U$): No upper surface, cluster of particles moving down specially near walls and dilute mixture removes particles in the middle.

Here U_{mf} is the minimum fluidization velocity which is extensively studied and a number of equations are available [9, 10] for its calculation, U_c is the critical superficial velocity at which the standard deviation of pressure fluctuations passes through a maximum and transition to turbulence fluidization occurs [13, 14] and U_{se} is the velocity that the entrainment process starts where particles can no longer be maintained in the column unless entrained particles are captured and returned to the bed efficiently. However for columns with large diameters, W, or small particle diameters, D_p, the slugging regime may be by–passed altogether, for which equations can be found in [13].

Fluid–solid regimes are of little practical importance in horizontal tubes except in dilute dispersed cases (pneumatic conveying) whereas different patterns of equal practical importance are observed in horizontal, vertical and inclined tubes. Therefore the classification of the flow regimes is far more complicated. However, generally a fluid–fluid mixture falls into a variety of different regimes depending mainly on the relative superficial velocity of each phase [15, 16]. Figure 1.2 shows the different regimes observed in a vertical tubes. Main flow structures in vertical tubes and their characteristics can be summarized as follows [17, 18]:

- Bubbly flow: Discrete small bubbles are contained within the liquid continuous phase.

- Slug Flow: Contains large 'Taylor bubbles' with an ellipsoidal nose with diameters approaching the channel width, the gas is separated from the side walls by slowly descending liquid films.
- Churn or Froth Flow: It is formed by the breakdown of large gas bubbles in the slug flow with oscillatory or time varying characteristic.
- Annular Flow: A liquid film forms at the channel side wall with a continuous gas core with large amplitude coherent waves usually present on the surface of the film.
- Mist Flow: Discrete liquid drops are contained within the continues gas phase which has specific applications in spray processes.

Note that in gas–liquid flows the transitions between different regimes is a function of both liquid and gas superficial velocities (U_{sl} and U_{sg}) [19], thus it can approximately be said that in a constant superficial liquid velocity, U_{sl}, the aforementioned patterns are orderly observed by increasing the gas superficial velocity U_{sg}.

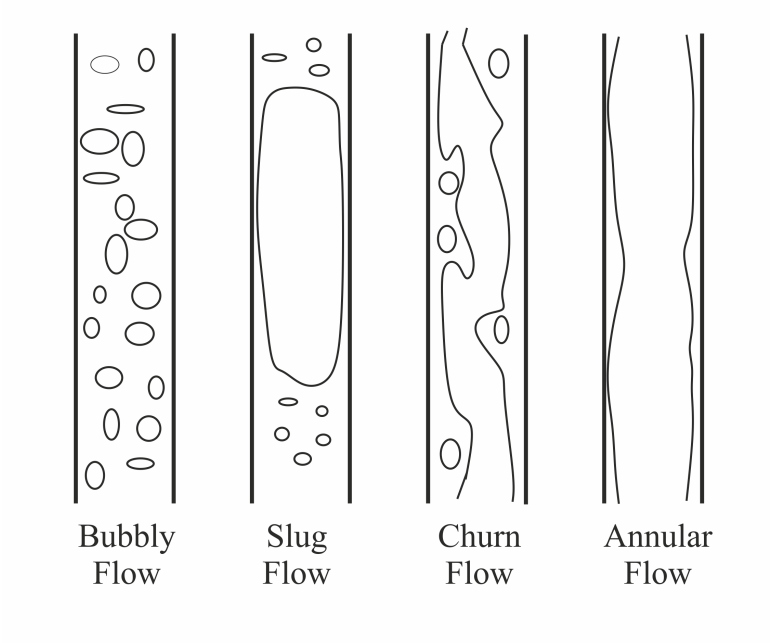


Figure 1.2: Gas-Liquid flow regimes in vertical tubes - Different regimes observed by increasing gas velocity from left to right at constant liquid velocity

In addition to flow patterns mentioned above, there is also the possibility of forming stratified or wavy stratified flow at low gas and liquid velocities [15, 20]. In this regime liquid flows at the bottom and gas flows on top with a quiescent well defined boundary which can become wavy with a slight increase in gas velocity [21], see Figure 1.3. Flow patterns in inclined tubes are also studied extensively for downward, upward and also for co– and counter–current flow conditions [22–25]. An exhaustive review of these regimes is outside the scope of current study and only the cases where one fluid appears as a continuous phase and the other as separate inclusions are considered in this study. This usually happens at small volume fraction of the dispersed phase.

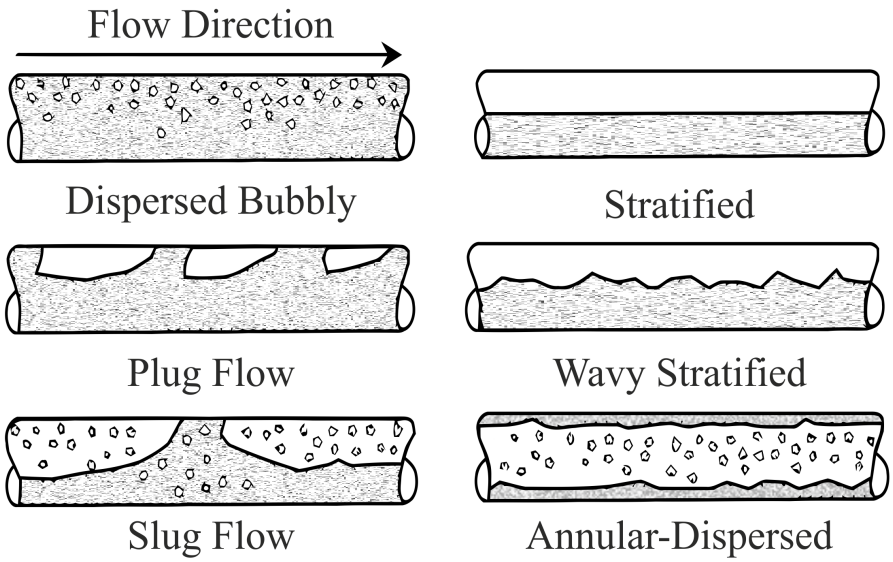


Figure 1.3: Gas-Liquid flow regimes in horizontal tubes - Different regimes observed in horizontal tubes for different gas and liquid velocities

1.3 Multiphase solution strategies

Processes in industry involving fluid flow are usually accompanied by heat and mass transfer and CFD techniques have increasingly been used for modelling and optimization purposes over the past two decades. These simulation techniques can be classified in terms of levels of complexity included, related to the resolution of the interface between the phases and the turbulence modelling, as follows:

- Direct numerical simulations of particulate flows by resolving the flow around the particles and accounting for their finite dimensions have become feasible in the past couple of years due to the drastic increase of computational power. In such an approach the time dependent solution of the three dimensional Navier–Stokes (NS) equations on a grid which resolves the particles is obtained using the appropriate boundary conditions on the surface of the particle. Two main approaches that are used to implement the boundary conditions and resolve the particles are: (*i*) body conformal mesh techniques and (*ii*) fixed grid methods. The advantages and disadvantages of each of these methods will be discussed later in Chapter 3 where a variation of the fixed grid methods is identified to have better properties. The implementation will be discussed in detail in the context of SIMPLE type NS solvers.

- Direct numerical simulations (DNS) for the fluid part, and a point–particle assumption for the particle phase. In this approach finite dimensions of the particle are not considered as explained in Section 1.1 and a Lagrangian approach is used to track the dispersed phase. This implies that a large number of particles may simultaneously be tracked through the computed time dependent flow field by considering the relevant forces where the effects of the particles on the flow field may or may not be considered, (Section 1.1). This approach is usually used for basic turbulent research in order to analyse the particle behavior in turbulent flows.

- Large Eddy Simulations (LES) combined with the Lagrangian tracking of point–particles have also been applied to the study of basic phenomena such as particle dispersion in turbulent flows, inter–particle collisions [26] and particle behavior in channel flows [27].

- For engineering purposes two approaches based on the Reynolds averaged NS equations are commonly applied, namely the two–fluid or Eulerian–Eulerian (EE) approach and the Eulerian–Lagrangian (EL) method. In these methods the inter phase interactions for momentum, heat and mass transfer, have to be extended by appropriate source/sink terms.

In the EE (or two–fluid) approach both phases are considered as interacting continua. Hence, properties such as the mass of particles per unit volume are considered as a continuous property, also particle phase velocity, interfacial transfer of mass, momentum, or energy requires averaging over the computational cells.

The Euler/Lagrange approach is only applicable to dispersed two–phase flows where the discrete nature of the individual particles are considered. In this method again particles are tracked in a Lagrangian manner by using a point–particle assumption and consequently using appropriate models for various relevant forces acting on the particles. However since the number of particles in a real simulation is usually too large to be tracked individually another layer of simplification is usually added by introducing 'computational' particles or 'parcels' which represent a number of real particles with the same properties (i.e. size, velocity and temperature). Then instead of individual particles, computational particles are tracked and properties such as dispersed phase density and velocity are obtained by ensemble averaging.

1.4 Plan of the thesis

In this thesis a hierarchy of models applicable to particulate flows starting from fully–resolved particles going down to less expensive, engineering approaches such as EE approach is considered. To exploit the capabilities of massively parallel systems currently available, sophisticated algorithms are required for effective parallelization of the NS solvers. Parallelization is achieved through a domain decomposition strategy. A library is developed based on a graph decomposition algorithm which is discussed in Chapter 2. Other elements required to achieve a scalable parallelization are the inter node communication schemes and linear solvers. Several algorithms for inter node communications are proposed and tested on up to 512 cores in addition to several optimization strategies to mask the communication latency. This has resulted in a well optimized stand alone library for inter node communication for general CFD codes which is discussed in Chapter 2. Discretization of the underlying PDEs describing the flow, results in a large system of linear equations which needs to be efficiently solved. Therefore several linear solvers and a multi-grid strategy are implemented and the best solver combinations for the current framework are identified in Chapter 2.

There are several different modelling techniques available for the fully–resolved multiphase flow simulations. As discussed in Section 1.3 there are generally two methods available to tackle this problem which is the subject of Chapter 3 where the methods available for a fully–resolved DNS of the flow–particle systems are discussed. In this chapter the methods are divided into body conformal and fixed mesh methods. Starting from the body conformal mesh methods the complexities associated with these approaches for remeshing and mesh movement steps are discussed. Alternatively fixed mesh approaches are discussed due to their flexibility, lower computational overhead, faster available solvers for underlying structured mesh and adaptability to available Eulerian flow solvers. The methods are divided into Immersed Boundary (IB) and Fictitious Domain (FD) methods and variation of each approach is discussed in detail. Almost all fixed mesh methods in the literature are implemented and discussed in the context of operator splitting type solvers and hence algorithms need to be adapted for implicit SIMPLE type solvers. Therefore a 2D solver is developed by the author specifically for adapting and testing different fixed mesh formulations. Several numerical tests are performed on different aspects of both FD and IB methods and generally FD methods are found to have better properties if heat or mass transfer phenomena are considered.

Extension of the FD method based on SIMPLE algorithm and source correction technique is extended to the heat transfer calculations for the first time in Chapter 4. In this chapter the numerical method is explained in detail and the method is tested on several basic test cases. In addition the code is used to study the practical problem of flow around a stationary cylinder. Although several correlations are available in the literature for the calculation of the mean heat transfer coefficients, to the best knowledge of the author, only scatter data are available for the local values of the Nusselt number and no reliable correlation can be found. This gap in the literature has been addressed in this chapter by providing a very high quality fit to the data gathered from 600 simulation for flow around a stationary cylinder at low to moderate Reynolds number and a wide range of Prandtl numbers. This also shows how the DNS data can be used to provide correlations for more engineering oriented simulations.

In Chapter 5, the proposed FD method is applied to several challenging problems. In these simulations forced and free motion of particles are considered. The method is extended for the simulation of both natural and convective heat transfer. Conjugate

heat transfer phenomena is also considered where the temperature inside the particle naturally develops and the temperature gradient inside the particle is captured. Motion of the particles in a narrow slit is considered to study the wall effects. Natural convection and the effects of Grashof number on the free motion of the particles with the same terminal Reynolds number is studied in fluids with constant viscosity. Also motion of a particle with internal heat generation in a cold fluid is studied in detail and its long term behavior is reported. A collision strategy is implemented and used to study the particle collision in two different regimes depending on the collision Stokes number. The method is extended to 3D flows and the flow around stationary sphere at low Reynolds numbers is studied and free fall motion of a single sphere are used to test the 3D implementation.

In this thesis a framework for a hierarchical simulation of particulate systems is also established. Using this strategy data from more accurate simulations as explained in Section 1.3 can be used to generate better models for more engineering inclined simulations. To this end engineering EE models and the derivation process based on the probability density function (PDF) approach for polydispersed turbulent particulate flow system are succinctly reviewed in Chapter 6. The terms that require modelling are identified and different methods available for writing closures for such unclosed terms are introduced.

In Chapter 7 two different methods are considered for writing closures for non-integer moments. The problem is considered from a mathematical point of view and two general procedures are tested and compared. First method is based on the reconstruction of the underlying PDF. This is done by writing the PDF as a series of generalized Laguerre polynomials and using a limited number of available integer moments to identify the series coefficients. The non–integer moments are then explicitly calculated using the reconstructed PDF by numerical integrations. The second method is based on the results of fractional calculus. In these methods fractional derivatives of the moment generating function are used to directly calculate the fractional moments. Both methods are applied to a number of analytical PDFs chosen to cover a broad range of highly non–Gaussian PDFs in addition to experimental crystallization data and point–particle DNS data. Using DNS data, it is again demonstrated why having a hierarchy of modelling tools is necessary for the simulation of complex particulate systems where the data from a more accurate simulation can be used to validate the proposed closures for a model in

the lower levels. Finally Chapter 8 summarizes the thesis and provides some suggestions for future research.

2

CFD code parallelization

2.1 Discretization by finite volume method

There are many text books, e.g [7, 28, 29] on this subject and there is no intention to reiterate the whole process here. However a brief introduction is necessary to introduce the notations that will be used throughout this thesis. In this work general discretization procedure described in [7] is adapted and follows next. Also note that all the numerical methods discussed in this thesis are primarily intended for incompressible flows and extending the discussions on numerical methods to compressible systems should be done with utmost care.

Firstly the domain is discretized into a finite number of control volumes covering the whole domain. All the fluid variables are defined at the centre of this control volumes which is usually referred to as a collocated grid arrangement. Other arrangements are also possible such as staggered arrangement where the pressure is defined on the cell centres and velocities are defined on the cell faces. This method ensures tight coupling between the velocity and pressure but requires defining different meshes for the velocity components and the pressure, which proves to be difficult to implement especially in 3 dimensions.

The integration of partial differential equations governing the fluid flow can be briefly described by means of the integration of a convection-diffusion equation for a general grid whether collocated or staggered. A convection-diffusion equation is composed of four terms: a convection term, a diffusion term, source terms and a temporal term. Figure 2.1 shows a simple orthogonal control volume. The upper case letters

$\{P, W, S, B, E, N, T\}$ represent the value of variable ϕ on the cell centre whilst the lower case letters $\{w, s, b, e, n, t\}$ show the variable at the faces.

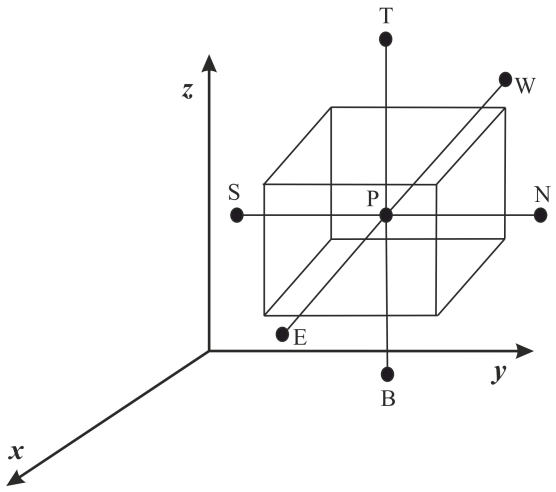


Figure 2.1: Coordinate system - A Cartesian coordinate system with a cell defined on it.

Integrating the general equation

$$\rho\frac{\partial\phi}{\partial t} + \rho\frac{\partial u_i\phi}{\partial x_i} = \Gamma\frac{\partial^2\phi}{\partial x_k\partial x_k} + S_\phi, \tag{2.1}$$

and using the Gauss divergence theorem [30] on the control volume presented in Figure 2.1 to convert the volume integrals to surface fluxes, results in the following algebraic equation

$$\begin{aligned}\frac{\rho\mathcal{V}}{\Delta t}\left(\phi_P - \phi_P^0\right) + F_e\phi_e - F_w\phi_w + F_n\phi_n - F_s\phi_s + F_t\phi_t - F_b\phi_b = \\ D_e\left(\phi_E - \phi_P\right) - D_w\left(\phi_P - \phi_W\right) + \\ D_n\left(\phi_N - \phi_P\right) - D_s\left(\phi_P - \phi_S\right) + \\ D_t\left(\phi_T - \phi_P\right) - D_b\left(\phi_P - \phi_B\right) + S_\phi\mathcal{V}.\end{aligned} \tag{2.2}$$

In Eq.(2.2), $\mathcal{V}$, F_f and D_f are the CV volume, convective and diffusive transport coefficients of the variable ϕ at the face f with the surface area $\mathcal{S}_f$ respectively defined by

$$F_f = (\rho u_n \mathcal{S})_f\,, \quad D_f = \left(\Gamma \frac{\mathcal{S}}{h}\right)_f\,, \tag{2.3}$$

where h is the grid spacing. The face values should be written in terms of the centre values to close the system ensuring the boundedness [29] of the solution. There are many ways of doing this and the details will not be discussed here. However if a simple upwind scheme is used [29] we get a set of algebraic equation with the following general structure

$$a_P \phi_P = \sum a_{nb} \phi_{nb} + b_{P,\phi}, \tag{2.4}$$

where

$$a_E = D_e + \max(0, -F_e), \quad a_W = D_w + \max(F_w, 0) \tag{2.5a}$$

$$a_N = D_n + \max(0, -F_n), \quad a_S = D_s + \max(F_s, 0) \tag{2.5b}$$

$$a_T = D_t + \max(0, -F_t), \quad a_B = D_b + \max(F_b, 0) \tag{2.5c}$$

$$a_P = a_W + a_E + a_N + a_S + a_T + a_B + \rho \frac{\mathcal{V}}{\Delta t} \tag{2.5d}$$

$$b_{P,\phi} = \rho \frac{\mathcal{V}}{\Delta t} \phi_P^0 + \mathcal{S}_\phi \mathcal{V}, \tag{2.5e}$$

and $nb \in \{E, N, T, W, S, B\}$. These algebraic equations can then be solved to yield the value of the variable ϕ on the centre of the control volume ϕ_P. Another method that is especially useful in iterative type methods is a deferred correction scheme. In this method first order upwind is used similar to equations (2.5a)–(2.5c) to write the coefficients and a term corresponding to the difference between the second order central differencing scheme and the upwind scheme is added to the source terms [7]. Upon convergence the method reduces to the second order central difference and a diagonally dominant coefficient matrix is ensured which is important for the convergence of the iterative linear solver. The set of NS equations for an incompressible Newtonian fluid with constant properties, can be written as

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \nu \frac{\partial u_i}{\partial x_j \partial x_j} + g_i, \tag{2.6}$$

$$\frac{\partial u_j}{\partial x_j} = 0. \tag{2.7}$$

For the velocity components the source terms contain the pressure gradients and other body source terms such as gravity and particle forces if two–way coupling is considered. These velocity sources can be written as

$$b_{P,u} = -\frac{P_E - P_P}{\delta h_e}\mathcal{V} + \mathcal{S}_u\mathcal{V} = -c_u\left(P_E - P_P\right) + \mathcal{S}_u\mathcal{V} \tag{2.8a}$$

$$b_{P,v} = -\frac{P_N - P_P}{\delta h_n}\mathcal{V} + \mathcal{S}_v\mathcal{V} = -c_v\left(P_N - P_P\right) + \mathcal{S}_v\mathcal{V} \tag{2.8b}$$

$$b_{P,w} = -\frac{P_T - P_P}{\delta h_t}\mathcal{V} + \mathcal{S}_w\mathcal{V} = -c_w\left(P_T - P_P\right) + \mathcal{S}_w\mathcal{V}. \tag{2.8c}$$

2.1.1 SIMPLE algorithm

For the above system of equations that present the momentum equations in each direction a guessed pressure field P^* is required in order to solve for the velocity components, say u_i^*. This can be written as:

$$a_{P,u}u_P^* = \sum a_{nb,u}u_{nb}^* - c_u\left(P_E^* - P_P^*\right) + \mathcal{S}_u\mathcal{V} \tag{2.9a}$$

$$a_{P,v}v_P^* = \sum a_{nb,v}v_{nb}^* - c_v\left(P_E^* - P_P^*\right) + \mathcal{S}_v\mathcal{V} \tag{2.9b}$$

$$a_{P,w}w_P^* = \sum a_{nb,w}w_{nb}^* - c_w\left(P_E^* - P_P^*\right) + \mathcal{S}_w\mathcal{V}. \tag{2.9c}$$

Since these are calculated using the momentum equation there is no guarantee that they satisfy the continuity equation. In this sense, the continuity equation introduces the following corrections for both the velocities and the pressure:

$$u_i = u_i^* + u_i' \tag{2.10}$$

$$P = P^* + P'. \tag{2.11}$$

Subtracting equations (2.9a)–(2.9c) from Eq. (2.4) (written for each velocity component) using the corresponding source terms (equations (2.8a)–(2.8c)), and introducing equations (2.10) and (2.11), following equations can be written for the velocity corrections:

$$a_{P,u}u_P' = \sum a_{nb,u}u_{nb}' - c_u\left(P_E' - P_P'\right) \tag{2.12a}$$

$$a_{P,v}v_P' = \sum a_{nb,v}v_{nb}' - c_v\left(P_E' - P_P'\right) \tag{2.12b}$$

$$a_{P,w}w_P' = \sum a_{nb,w}w_{nb}' - c_w\left(P_E' - P_P'\right). \tag{2.12c}$$

In order to derive the velocity correction and to limit the computational costs, a common practice is to drop the neighbouring terms. Omission of these neighbour term results in the SIMPLE algorithm and u_i' values can simply be written as

$$u'_P = -d_u \left(P'_E - P'_P\right) \tag{2.13a}$$

$$v'_P = -d_v \left(P'_E - P'_P\right) \tag{2.13b}$$

$$w'_P = -d_w \left(P'_E - P'_P\right), \tag{2.13c}$$

where $d_u = \frac{c_u}{a_{P,u}}$ and similar expressions can be written for d_v and d_w. Other variations of the simple algorithm can be derived by using different approximation to the neighbouring terms. Continuity equation can then be discretized by setting for example $u_e = u_e^* - d_u|_e(P'_E - P'_P)$ to derive an equation for the pressure correction

$$a_{P,P'} P'_P = \sum a_{nb,P'} P'_{nb} + b_{P,P'}, \tag{2.14}$$

where $a_{P,P'} = \sum a_{nb,P'}$ and $a_{nb,P'} = \rho_f d_{f,nb} \mathcal{S}_{nb}$. Also note that the source term is the mass imbalance on each control volume and full derivation can be found in [28]. Finally after solving Eq. (2.14) the velocities are corrected by means of Eq. (2.10) and pressure by Eq.(2.11). Under relaxation is usually needed to avoid divergence because of the approximation used when writing the velocity correction terms (Equations (2.13a)–(2.13c)). Iteration is then required to improve the guessed pressure field up to a predefined tolerance.

2.1.2 Operator splitting and fractional step methods

The SIMPLE type algorithms as explained in Section 2.1.1 are iterative methods. Another very popular solution strategy for the numerical solution of the NS equations are the operator splitting methods. All the simulations in this thesis are built upon the SIMPLE framework however there are several references to the operator splitting type methods and a short discussion of the methods follows.

In these methods the coupled NS equations are split into a set of simpler equations such as the advection–diffusion and the Poisson equations followed by a series of explicit or implicit updates [31]. The method starts by providing a solution to the momentum equation, which is derived either explicitly or implicitly, by entirely disregarding the incompressibility condition followed by a projection onto a divergence–free field. A very

simple explicit scheme can be constructed by first generating an initial guess for the velocity field [31]:

$$u_i^* = u_i^n - \Delta t u_j^n \frac{\partial u_i^n}{\partial x_j} - \frac{\Delta t}{\rho}\frac{\partial P^n}{\partial x_i} + \Delta t \nu \frac{\partial^2 u_i^n}{\partial x_j \partial x_j}. \tag{2.15}$$

Note also that the diffusion terms or both advection and diffusion terms may be treated implicitly to improve the stability of the scheme [31]. For u_i^{n+1} to fulfil the continuity equation, Eq. (2.15) should be evaluated using the pressure at time $n+1$:

$$u_i^{n+1} = u_i^n - \Delta t u_j^n \frac{\partial u_i^n}{\partial x_j} - \frac{\Delta t}{\rho}\frac{\partial P^{n+1}}{\partial x_i} + \Delta t \nu \frac{\partial^2 u^n}{\partial x_j \partial x_j}. \tag{2.16}$$

Subtracting Eq. (2.16) from Eq. (2.15) and setting $\phi = P^{n+1} - P^n$, results in

$$u_i^{n+1} = u_i^* - \frac{\Delta t}{\rho}\frac{\partial \phi}{\partial x_i}. \tag{2.17}$$

Since u_i^{n+1} should be divergence–free Eq. (2.17) leads to the Poisson equation for the pressure difference:

$$\frac{\partial^2 \phi}{\partial x_i \partial x_i} = \frac{\rho}{\Delta t}\frac{\partial u_i^*}{\partial x_i}. \tag{2.18}$$

After solving the pressure Poisson equation velocity is updated by Eq. (2.17) and pressure by

$$P^{n+1} = P^n + \phi. \tag{2.19}$$

The details of implementing boundary conditions for Eq. (2.18) are outside the scope of this thesis and the reader can consult [32, 33] for more information. Fractional step methods are based on similar principles, however slightly different splitting in conjunction with the Range–Kutta integration are usually employed to enhance the accuracy, see e.g [34–36].

2.2 Domain Decomposition

The most popular approach for the parallelization of the CFD codes is a domain decomposition [37] strategy. The objective is to distribute the computational grid to a number of processors such that the work load on each processor is equivalent and the

amount of required communication between different cores is minimized. The problem is trivial for simple geometries and it can easily be coded in the solver. However for irregular geometries the problem is far more complicated and several parameters should be considered for an efficient and scalable parallelization. A multilevel graph min-cut algorithm based on the algorithm proposed by [38] is coded for this purpose which will be discussed next.

The problems of interest comprise at least a few hundred blocks or even more generally discretized using an unstructured grid, which are solved on arbitrary number of processors. At the first glance a brute–force algorithm may seem feasible i.e to calculate the cost of all combinations and choose the configuration with the smallest cost. This algorithm is simply not feasible even for problems with number of blocks of the order $O(10)$[1] or yet worse for an unstructured grid where each single node should be considered for the distribution. The term "block" (a set of computational nodes) is used in the following discussion since block–structured type grids are of main concern in this study. However similar discussions hold by replacing blocks with nodes in case of unstructured grids.

Obviously more sophisticated algorithms are needed to perform this task even for small problems. The natural data structure for this problem is a graph with vertices presenting the blocks and vertex weights presenting the computation cost of each block, an edge between two vertices presents a block connectivity with a weight presenting the communication cost. The problem is now reduced to a graph partitioning problem. A formal statement of the problem will be provided in the next section but it can simply be said that we want to partition the vertices of a graph into p roughly equal parts, such that the number of edges connecting vertices of different parts is minimum [39] ('*number*' can be changed with '*weight*' for a generalized algorithm).

This problem is NP-complete [38–40] meaning that if a solution to the problem is known it can be tested in a polynomial time however no polynomial time algorithm to solve the problem is known [41]. Despite this fact there are several algorithms to give reasonably good approximate partitioning of a graph. The spectral methods [42] are known to produce very high quality partitions however they are very expensive due to

[1] Assume distributing a 64-block problem on 4 processors. Then we need to calculate $\binom{64}{16} \approx 10^{18}$ possible combination assuming similar blocks, without this assumption the number possibilities are even more.

calculation of Fiedler vector[1]. There are also geometrical methods [43] which tend to be fast but not applicable to graphs with no explicit geometrical information. A multilevel graph partitioning [39] is chosen where a smaller graph is produced from the original graph by collapsing the edges and vertices then this smaller graph is partitioned and original graph is reconstructed from each partition by a series of 'uncoarsening' stages refining the quality of the partition. This method is proven to be very fast, general and produces partitions with qualities as high as those produced by spectral methods [44].

2.3 The k-way graph partitioning

The k-way graph partitioning is defined as follows: given a graph $G = (V, E)$ with $|V| = n$, partition V into k subset $V_1 \cdots V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $|V_i| = \frac{n}{k}$ and $\bigcup_i V_i = V$ and the number of edges in E with incident vertices belonging to different subsets is minimized [40]. This partitioning is presented by a vector P with $|P| = n$ such that $1 \leq P[v] \leq k \ \forall v \in 1 \cdots n$ which simply means vector P stores the partition that the v^{th} vertex belongs to, using an integer in the range one to the number of partitions k. K-way partitioning is usually solved by a recursive algorithm. i.e we first find the 2-way partitioning of the graph and then further divide the graph using 2-way partitioning on each part [39].

The current code (FMETIS) is based on the algorithm devised by Karypis and Kumar [39] and their METIS library. The METIS library is originally written in C programming language and was not appropriate to use in conjunction with the current framework which is written in FORTRAN95. Therefore a FORTRAN95 version of the algorithm is implemented by the author to avoid future compatibility problems.

Considering the graph $G_0 = (V_0, E_0)$ with possible weights on both vertices and edges, the algorithm can be summarized as follows:

- Coarsening phase: The graph G_0 is transformed into a sequence of smaller graphs $G_1, G_2 \cdots, G_m$ such that $|V_0| > |V_1| > \cdots > |V_m|$.
- Partitioning phase: A 2-way partition P_m of the graph $G_m = (V_m, E_m)$ is computed such that each contain half of the vertices of the original graph G_0. For

[1] eigenvector corresponding to the second smallest eigenvalue

a k-way partitioning this subroutine is called recursively log^k times to get the k partitions.

- Uncoarsening Phase: The partition P_m of G_m is projected back to G_0 by going through partitions $P_{m-1}, \cdots, P_1, P_0$ doing some refinement at each step.

2.4 Testing the Code

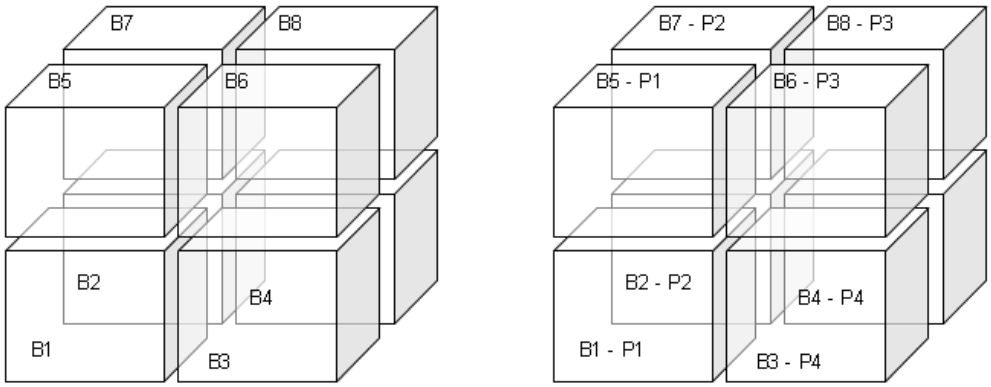
The code has a driver which constructs the input graph by estimating the edge and vertex weights after reading the input geometry file based on the number of nodes on connection faces and the number of nodes on each block. Then this graph is passed to the main subroutine to get the partitioning P which is then used to completely decompose the domain. Some tests are performed on the developed code which follows next.

2.4.1 8-block test: A trivial example

An 8-block geometry is presented in the Figure 2.2a. The corresponding graph and input data is presented in Figure 2.3. All blocks are similar and have 729 internal nodes also all connection faces are identical having 49 nodes. The decomposition of blocks on 4 processors is presented in Figure 2.2b. The first line in the input data is the number of vertices and the number of edges on the graph which is half the number of connection faces. The following lines are the adjacency list of the graph where the first integer is the weight of the vertex in the current line, followed by the index of the vertexes (and their corresponding weights), the current vertex has an edge (connection) to, see Figure 2.3.

2.4.2 Other tests

The program is also tested to partition a graph with $|V| = 15606$ and $|E| = 45878$ into 2 to 20 partitions. The results were checked against the results obtained from the original METIS library and same partitioning were found in all cases. The average running time of the code is 28.9% faster than the original code using same level of compiler optimizations. However there is no clear reason for this improvement since the same algorithm are used in the new library without any improvements. Therefore this is solely



(a) 8-Block geometry **(b)** Optimum Decomposition

Figure 2.2: 8-block decomposition on 4 processors - A simple geometry consisting of 8-blocks and a balanced decomposition with minimum communication cost produced by FMETIS.

related to the choice of programming language and compiler implementations. Table 2.1 shows a summary of this test. Second and third column show the CPU time in seconds required for the partitioning the graph into the required number of parititions and the last two columns are the number of unmatched vertices between the two libraries and the speed up respectively. Where the speedup is calculated by

$$S.U = 100 \cdot \frac{\text{METIS CPU Time} - \text{FMETIS CPU Time}}{\text{FMETIS CPU Time}}. \tag{2.20}$$

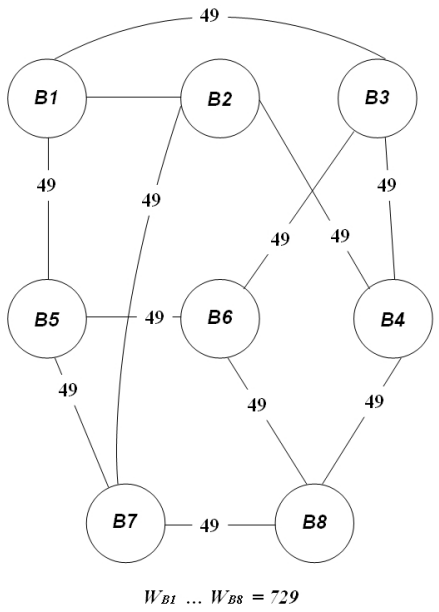
Also note that due to the random nature of the algorithm 1 or 2 vertices might be placed differently from the one in the METIS library. These vertices are those with the same weights associated with them and therefore have no impact on the final solution.

2.5 Multi Block Presentation

After the domain decomposition each processor has a set of blocks to perform the computations but an additional layer of mesh is required where the information of the neighbouring processors is needed. This additional layer is added such that each processor has the required data to do its computation independently reducing the amount

#Partitions	METIS CPU Time (sec)	FMETIS CPU Time (sec)	Unmatched Vertices	S.U (%)
2	0.093	0.063	0	47.61905
3	0.093	0.078	0	19.23077
4	0.093	0.062	0	50.00021
5	0.093	0.063	0	47.61905
6	0.093	0.078	0	19.23077
7	0.093	0.094	0	-1.06383
8	0.093	0.078	0	19.23077
9	0.093	0.094	0	-1.06383
10	0.109	0.079	1	37.97468
11	0.109	0.079	0	37.97468
12	0.109	0.093	1	17.2043
13	0.109	0.079	0	37.97468
14	0.109	0.078	0	39.74359
15	0.109	0.078	0	39.74359
16	0.109	0.094	0	15.95745
17	0.109	0.078	0	39.74359
18	0.109	0.094	0	15.95745
19	0.109	0.094	0	15.95745
20	0.109	0.063	0	73.01587

Table 2.1: FMETIS test results - An average 28.9% speedup in FORTRAN implementation.



8	12	11				
729	5	49	3	49	2	49
729	1	49	7	49	4	49
729	1	49	6	49	4	49
729	3	49	2	49	8	49
729	1	49	7	49	6	49
729	5	49	3	49	8	49
729	5	49	2	49	8	49
729	7	49	6	49	4	49

Figure 2.3: 8-block Graph - The graph of the 8-block geometry and the corresponding input data to the FMETIS program.

of communications. This is necessary for a reasonable parallel performance otherwise a great deal of fine–grained communication is required whenever one processor needs information stored on its neighbours. Figure 2.4 shows the decomposition of the domain into six blocks and addition of the overlapping layers to the each block to hold the information received from the neighbouring blocks.

2.6 Communication schemes

The complexity of the communication pattern directly depends on the domain decomposition and how the computational blocks are associated to the computing cores. It can be said that the minimum requirement for a communication algorithm is to avoid Coffman deadlock or simply the deadlock situation [45]. Deadlock refers to a specific condition when two or more processes are each waiting for the other to release a resource, or more than two processes are waiting for resources in a circular chain. Deadlock is a common problem in multiprocessing where many processes share a specific type of mutually exclusive resource known as a software lock [46].

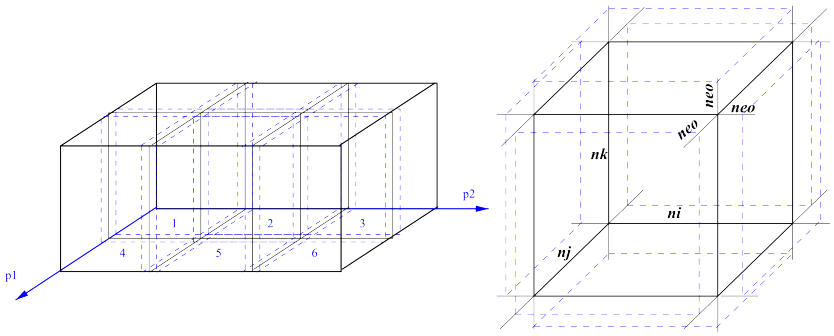


Figure 2.4: Overlapping mesh layers - Left shows the decomposition into 6 blocks and on the right addition of the overlapping mesh layers for a general case.

Deadlocks can happen as a result of wrong logic or buffer requirements. In the case of two processors communicating, the former can happen for example if both issue a blocking type function call at the same time. The latter can happen for example if both start sending data before any of them issues the corresponding receive, see [47, 48] for further details. Some solutions to prevent these problems are

- Using Buffered `Send` and `Recv`: these functions allow for a user specified buffer and thus never run out of buffer. However for large problems this extra memory is the main drawback of this method,
- Using MPI `Sendrecv` function or MPI collectives: using MPI `Sendrecv`, the MPI library takes care of the correct execution of the `Send` and `Recv`, however for large number of blocks and arbitrary connections it gets overly complicated to execute function calls for communication pairs instead of considering single processes one at a time. In case of MPI collectives optimization of the non–contiguous data types which will be discussed in Section 2.6.3 is not possible.
- Matching each `Send` with the corresponding `Recv`.

The last solution which is believed to be a feasible solution for complicated communication schemes is adopted here. Avoiding deadlocks efficiently using this strategy is very simple for regular decompositions, e.g one can use an odd–even communication

pattern for 1D slice decomposition where odd numbered processors start sending a message while even numbered processors receive the message and in the next step reverse operation is performed. Similar pattern can be used for 2D pencil or 3D regular blocks but in two or three different directions. The discussion can be formalized by defining communicator topologies [46, 47]. If the decomposition is regular, which is only guaranteed for structured grids, then the underlying communication pattern conforms to a Cartesian topology. In the general case of an unstructured grid or a block structured grid however, the communication pattern can only be defined by a graph topology and efficiently avoiding deadlocks is far more complicated.

2.6.1 Connectivity Pattern

As discussed earlier in Section 2.6, of interest are the general communication patterns where the processors can logically be grouped into graph topologies. There are some outdated performance analysis and algorithm suggestions for CFD applications which are conducted on machines with low compute power and for small scale problems [49–51]. Djomehri and Biswas [52] also suggested an asynchronous algorithm for their block–structured code. A general communication library independent of the underlying CFD code named CFDComm with several optimization strategy is developed by the author. The library is tested for up to 512 processor cores and shows almost perfect scalability which will be discussed in the following sections.

As discussed in Section 2.2 a graph partitioning algorithm is used to decompose the domain. The output of the graph partitioning algorithms is usually an integer array containing the partition number of each Control–Volume (CV), for unstructured grids, or blocks, for block–structured grids. Then all CVs with same partition number are assigned to a compute node. Any decent mesh generating software also provides a list of neighbours of each CV and having the partition information a global connectivity graph (GCG) can be produced where vertices represent processors and edges represent a connection to another processor. For more reliable comparison instead of partitioning a real mesh and generating the GCG, a GCG is simulated by first producing a connected graph. It is assumed that the maximum number of connections for each processor is $\texttt{MAXCON} = \text{MIN}(6, \texttt{NPROCS})$, where NPROCS is the number of the processors. Then assuming that the graph has half of the maximum number of possible edges, i.e $\texttt{MAXCON} \times \texttt{NPROCS}/4$, a number of additional random edges are added to the GCG. Figure 2.5 shows

the simulated GCG for the 32 processor test case using the aforementioned procedure. Similar graphs are produced for other test cases and same sequence of random numbers are used to ensure cases with the same number of processors have identical GCG for all proposed algorithms which makes the efficiency comparisons more reliable.

2.6.2 Synchronous/Asynchronous communication

Almost all of the communication that is required in general CFD application concerns the exchange of inter–grid boundary data. The message passing can be synchronous or asynchronous, but the choice significantly affects the MPI programming model. As a first proposal a simple synchronous scheme is implemented and tested which shows acceptable performance on small number of processors up to 16 cores and guarantees a deadlock free communication. The synchronous communication is performed with blocking MPI `send–receive` calls, while the asynchronous communication uses non-blocking calls. However the implementation of the asynchronous type communications is far more complicated. For synchronous communication, the total number of `send–receive` calls is `nprocs` $\times$ (`nprocs` $-$ 1), where `nprocs` is the number of processors. A send call is blocked until the receiving processor is ready to accept the message, i.e., until the matching receive call is posted. The overhead of this synchronization causes the communication time to increase by increasing the number of processors and hence very pure scalability.

The synchronous algorithm is tested for 4–256 cores. 200–20000 64bit contiguous real elements are transferred in 100 iterations and the communication time of each iteration is calculated. Only the results for 4–128 cores are plotted in Figure 2.6 for better presentation. No dead-locks were observed but obviously the algorithm is not scalable and communication time increases by increasing the number of processors.

To overcome the scalability issue an asynchronous point-to-point communication algorithm in implemented. This algorithm includes a non-blocking post-receive and a non-blocking send operation. The algorithm is tested for 4-512 processors and no dead-lock is observed for communicating 200-20000 64bit real elements. Figure 2.7 shows the communication time for 128 and 256 processors which shows almost no increase in communication time by doubling the number of processors. Results of the synchronous communication for 256 processors is also provided in this figure which shows that the

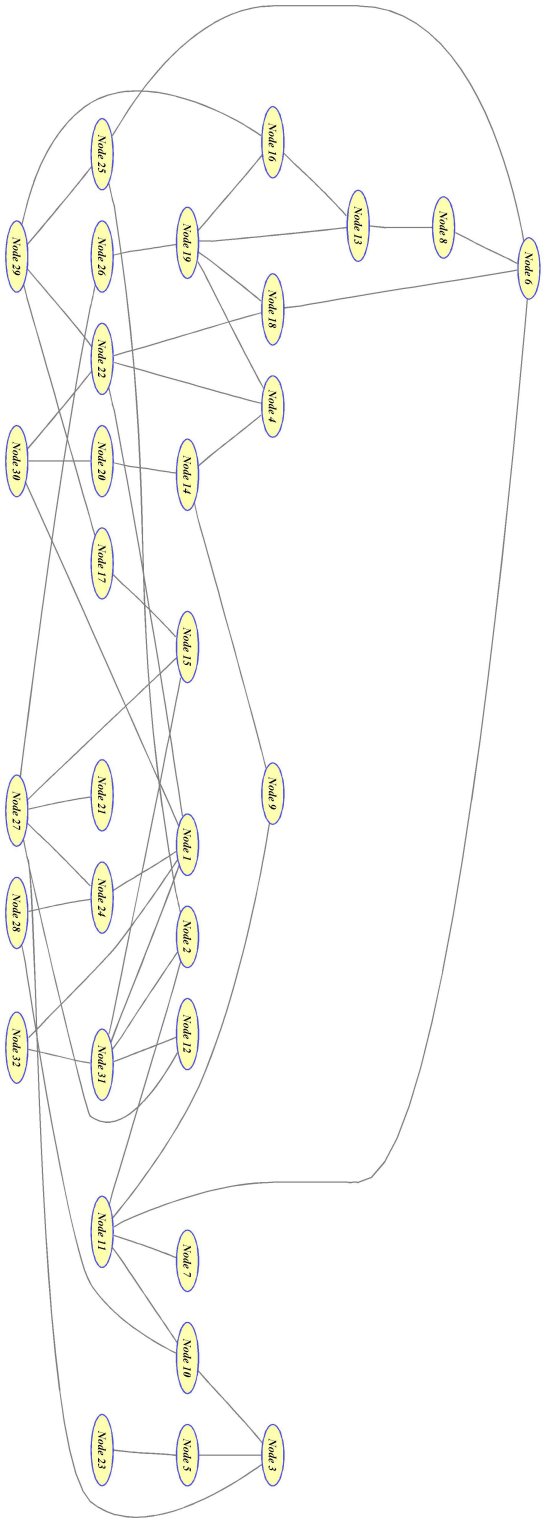


Figure 2.5: 32–processor GCG - simulated using the algorithm explained in Section 2.6.1

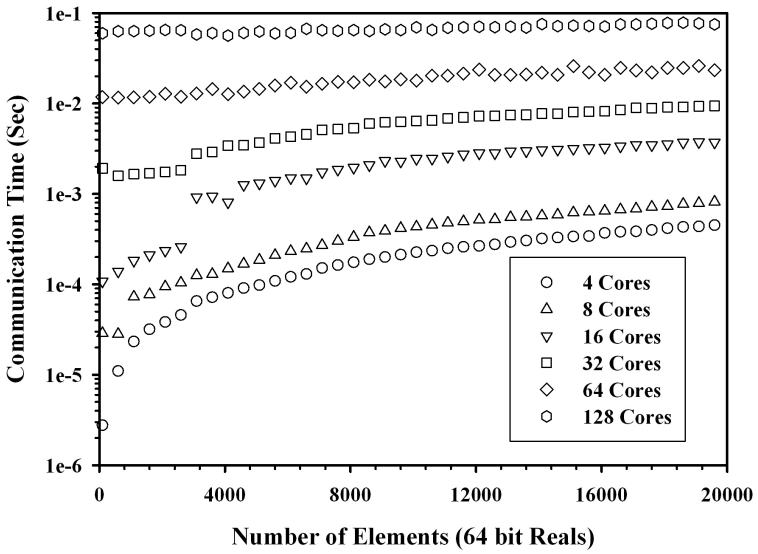


Figure 2.6: Synchronous communication time - time for communicating 200–20000 64bit contiguous real elements on 4–128 cores.

asynchronous algorithm requires almost 3 orders of magnitude less time to communicate the same amount of data for the same GCG.

2.6.3 Non-contiguous data communication

It is not usually guaranteed for the boundary data to be contiguous in the memory and hence for sending and receiving the information packing the data into contiguous arrays is required, see Figure 2.8. In Figure 2.9, the average time of 100 iterations, required for the communication are compared to the cumulative time required for both the communication and the buffer copies. This shows that the time required for the buffer copies can be as large as the time required for communication. Therefore in this library two optimization strategies are provided to reduce the buffer copy time namely using MPI data types and overlapping communication with the buffer copy which will be discussed next.

2.6.3.1 Using MPI Data Types

In this method MPI indexed block types are defined to directly perform the communication on the original data arrays, see Figure 2.10. This is done conveniently by accepting

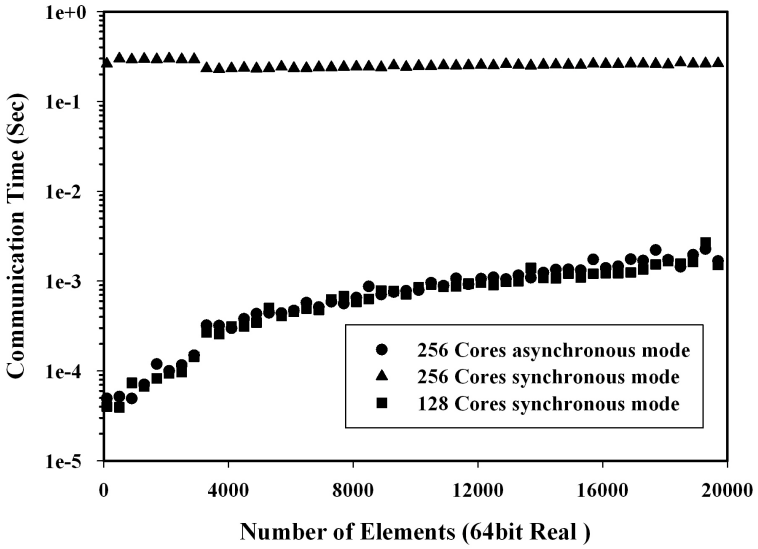


Figure 2.7: Scalability of the asynchronous scheme - no increase in the computation time for communicating a contiguous buffer using the asynchronous algorithm on 128 and 256 processors.

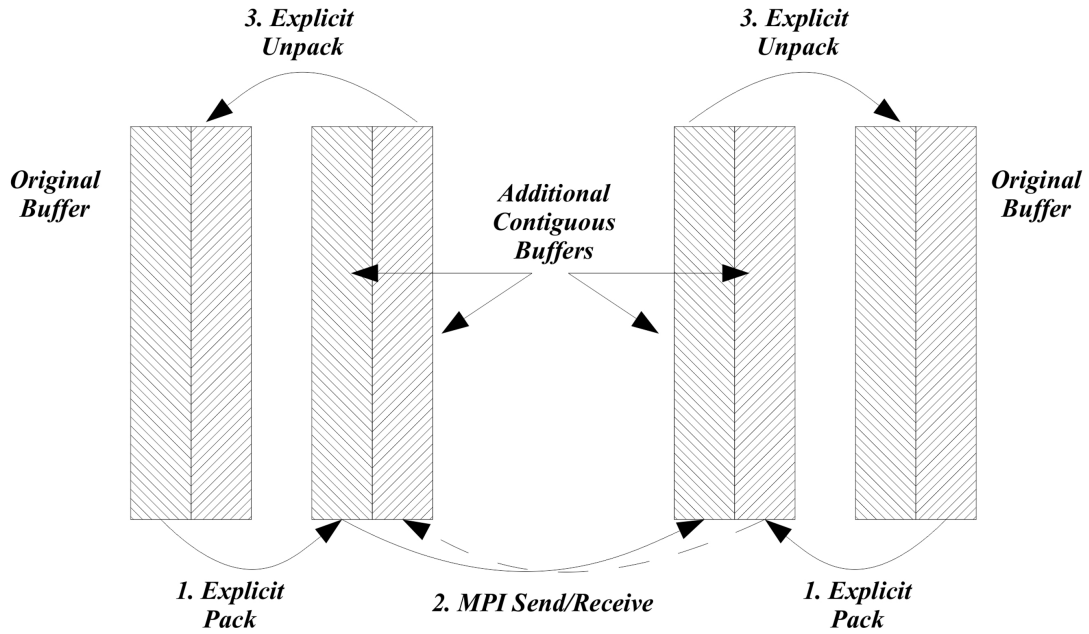


Figure 2.8: Explicit-Packing - First copy the data into auxiliary arrays so that they are contiguous in memory ready for communication.

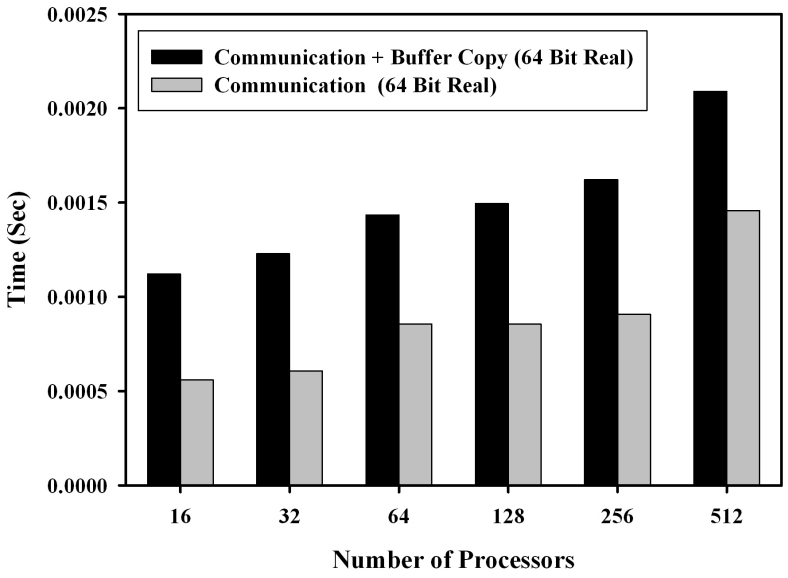


Figure 2.9: Total Communication time - time required for the buffer copy can be as large as the communication time.

a displacement array from the user and the code performs all required function calls without further user input. Figure 2.11 compares the total communication and buffer copy time to the communication time using MPI types. Note that the buffer copies take place in the calling code which is compiled using the highest level of compiler optimization and hence the packing performance is highly optimized. Evidently for single 64bit real elements using MPI types actually deteriorates the performance. However in almost all unstructured CFD applications (and hence the current framework) a collocated grid arrangement is used. In this arrangement all variables are stored on the cell centres and hence the indexing for all the variables is the same. This provides a unique opportunity to first embed these variables in Fortran types, for example three velocity components can be embedded in Fortran vector type which consists of an array with 3 elements. Now instead of an array for each velocity component only an array of vector types is available.

The possibility of communicating on two Fortran types in addition to real values, namely a first order (T1) and a second order (T2) tensor type is added to the current implementation. These types directly correspond to the first and second order tensors

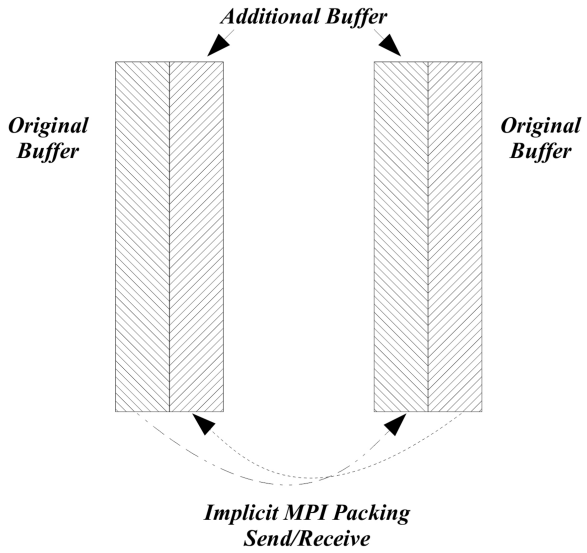


Figure 2.10: Implicit-Packing - direct communication using MPI data types letting MPI library do the packing.

in the fluid mechanics. Figure 2.12 shows the average communication–buffer copy time compared to the average communication time using MPI data types for the T1 type. A huge performance gain is obtained using this strategy and the time for communicating the T1 type (three 64bit real values) is almost the same as the time required for a single real element communication, see Figure 2.11. This is mainly due to the fact that three memory read/write is required per data element if simple buffer copy is used whereas using MPI indexed types the whole message (3 real elements) is read and copied at once.

2.6.3.2 Communication–unpacking overlap

A well known strategy to increase the performance of parallel applications is to overlap the communication with computation. A trivial amount of computation that can be done while communicating is the data unpacking which is exploited in the current library version. An algorithm is devised to perform this task in conjunction with the asynchronous algorithm.

In Figure 2.13 the average time of 100 iteration for a simple buffer copy strategy is compared to the time with overlapping optimization. On average an 18% performance

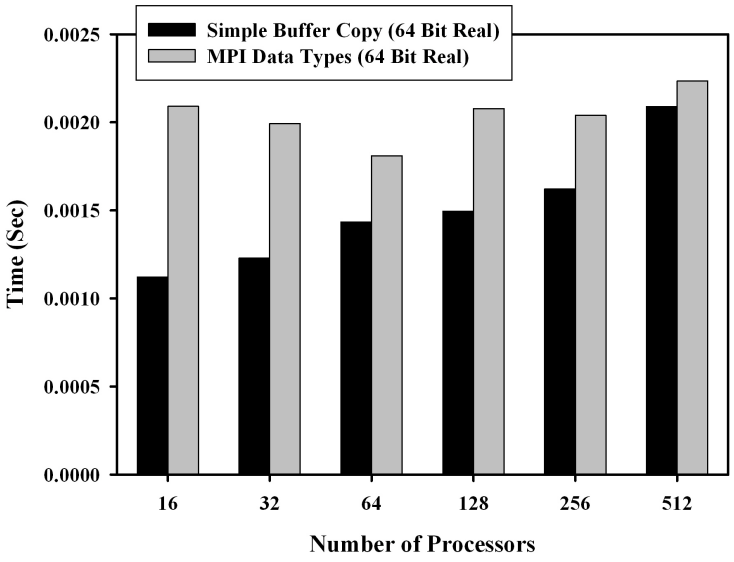


Figure 2.11: Comparison of total Communication time for single real elements - using MPI indexed types deteriorates the performance of the communication due to the buffer optimizations in the calling code.

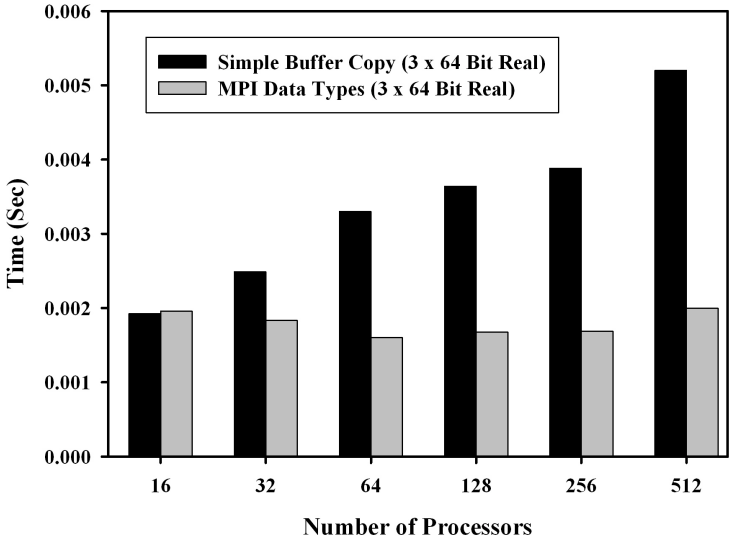


Figure 2.12: Comparison of total Communication time for T1 types - time for communicating the T1 type (three 64bit real values) is almost the same as the time required for a single real element communication.

increase is observed for 16–512 core tests. A larger performance increase is gained by increasing the number of processors. For example the performance gain is about 7% for the 16 processor case and is more than 25% for the 512 case. This optimization strategy seems to be very effective and no further effort is required on the user side.

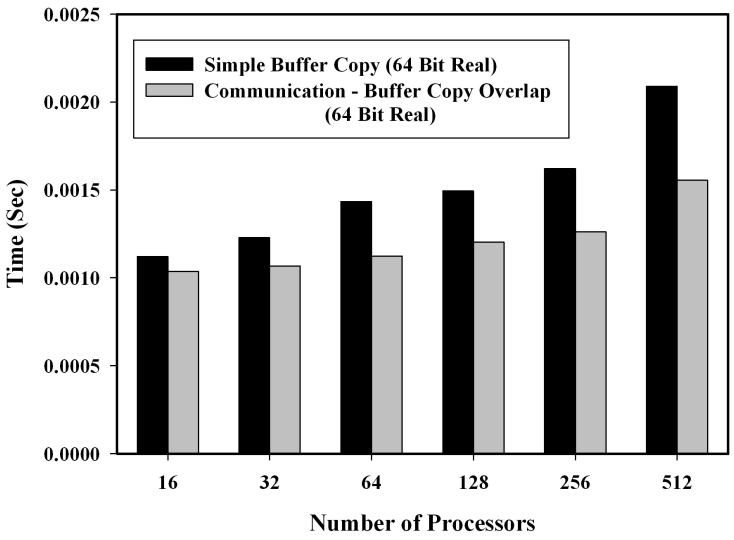


Figure 2.13: Communication–buffer copy overlap - on average 18% higher performance is obtained by overlapping buffer copy (unpacking) with communication.

2.6.4 Addition of MPI topologies

The library also provides the opportunity to conveniently add the MPI topologies to the communicator. On the user side the only required information is the GCG information which should be passed to the library and the library will define the required communicators and perform all the required function calls. Figure 2.14 shows the comparison of the total time for the simple buffer copy with and without adding the MPI topologies. Obviously adding the MPI topology information to the communicator greatly increases the efficiency of the communication. On average 44% increase is observed but the effects decrease by increasing the number of processors. For example for the 16 processors case a 52% increase is observed whereas for the 512 processor case this increase is reduced to 30%.

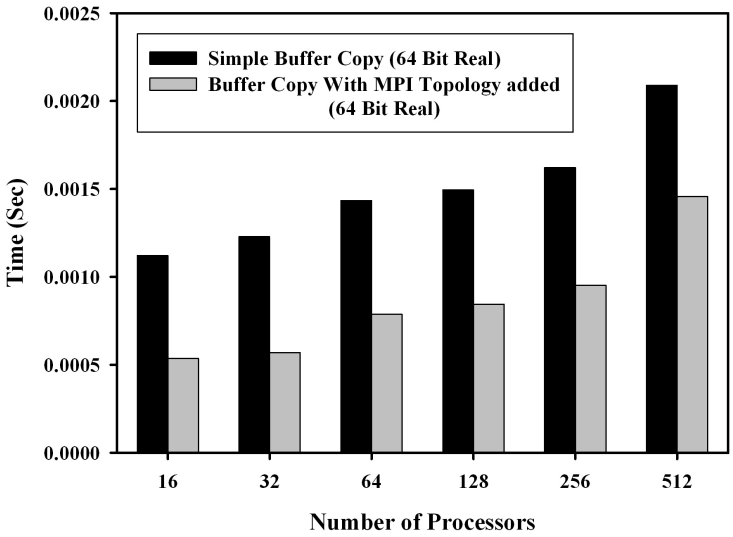


Figure 2.14: Addition of MPI topology - 44% increase in the communication performance is observed by adding MPI topologies to the communicator.

2.7 Linear Solvers

Discretization of the NS and other transport equations describing fluid flow problems using finite volume method generates large sparse system of linear equations. For a successfully CFD simulation the underlying algorithm should repetitively solve several transport equations resulting from the discretization of momentum, energy and other fields of interest, Eq. (2.4), in addition to a pressure (or pressure correction) equation, Eq. (2.14). Solving these linear equations usually is the most expensive part of the solution process. Therefore efficiently solving the linear system of equations is crucial for any large scale simulation. Therefore implementation of more efficient solvers for the current framework was absolutely necessary and hence several linear solvers widely used to solve the linear systems resulting from the discretization of the NS equations are implemented and tested which are discussed in the next section.

2.7.1 An Overview of the linear solvers

Methods used to solve a system of linear equations can be divided into two categories [53, 54]: (i) Direct methods which provide the exact solution in a predetermined number

of steps. (*ii*) Iterative methods which start by an initial guess and try to improve it in an unknown number of steps. Figure 2.15 shows the typical information available from each type of the solvers in the course of the solution. In an iterative process starting from an initial guess convergence to the final solution in at most $O(n)$ iterations, where n is the number of equations, is required otherwise the iterative process is not considered efficient.

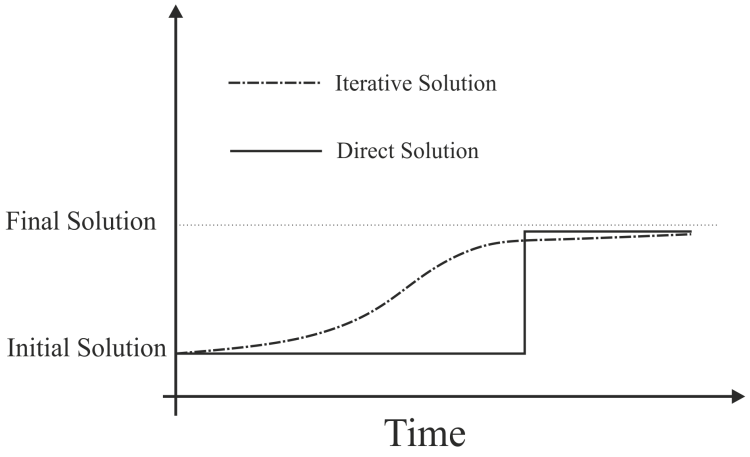


Figure 2.15: Linear Solvers - Available information from the linear solvers in time. A direct solver provides the exact solution after a specific number of steps whereas an iterative solver improves the initial guess.

In Appendix A detailed discussions and references for different solvers are provided. Based on the discussions in Appendix A, iterative solvers can be classified into four categories:

- Simple iterative methods, such as Jacobi iteration and SOR.
- Incomplete factorization schemes, such as Incomplete Cholesky, ILU, SIP (also known as Stone's method) and MSI.
- Krylov space methods, such as the CG, CGS, GMRES, QMR, BiCG and BiCGSTAB methods.
- Multigrid schemes

Multigrid methods are usually used to improve the performance of the simple iterative and incomplete factorization methods and hence they better classified as acceleration techniques rather than being a specific solution method. Incomplete factorization or simple iterative methods are usually used as preconditioners to improve the convergence rate of the Krylov space solvers. As a result many combination of these methods can be found in the literature and a clear distinction cannot be made between the methods. In the current version of the solvers only the basic implementations for the current CFD code is considered. No attempt is made to optimize the solver using optimization techniques such as block matrix multiplication, vectorization and taking advantage of system pipelines, see [54] for more information.

2.7.2 Residual form of the equations

Assuming $\boldsymbol{\delta} = \mathbf{x}^k - \mathbf{x}^{k-1}$, then the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be written as

$$\begin{aligned} \mathbf{A}\left(\mathbf{x} + \boldsymbol{\delta}\right) &= \mathbf{b}, \\ \mathbf{A}\boldsymbol{\delta} &= \mathbf{b} - \mathbf{A}\mathbf{x}^k = \mathbf{r}^k, \end{aligned} \tag{2.21}$$

where $\mathbf{r}^k$ is the residual of the current iteration. The system can then be transformed to

$$\begin{aligned} \mathbf{A}\boldsymbol{\delta} &= \mathbf{r}^k, \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \mathbf{r}^k. \end{aligned} \tag{2.22}$$

When using finite precision arithmetic the residual form of the system gives a greater accuracy. This is due to the fact that the corrections made to the $\boldsymbol{\delta}$ values are very likely to be of the same order as the $\boldsymbol{\delta}$ values themselves, whereas for the $\mathbf{x}$ terms, especially after a few iterations, correction are, almost surely, several orders of magnitude smaller than the $\mathbf{x}$ values. This results in much smaller rounding errors in the residual form of the equations.

2.7.3 Norms and Stopping Criteria

A vector norm is a function $||\cdot|| : \mathbb{R}^m \rightarrow \mathbb{R}$ that assigns a real valued length to each vector with the following properties

$$\begin{aligned} &\|\mathbf{x}\| \geq 0 \; and \; \|\mathbf{x}\| = 0 \; only \; if \; \mathbf{x} = 0, \\ &\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|, \\ &\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|. \end{aligned} \tag{2.23}$$

A class of important vector norms namely a p–norm is defined by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}. \tag{2.24}$$

The most important p–norms are 1, 2 and ∞ norms. By directly calculating the limits of Eq. (2.24) as $p \rightarrow \infty$ it is easy to show $\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$.

For a $m \times n$ matrix each of the mn entries is an independent coordinate and any mn–norm can be used to measure the size of the matrix. However when dealing with matrices, induced matrix norms defined in terms of the behavior of a matrix as an operator between its domain and range space is more appropriate. Mathematically this can be written

$$\|\mathbf{A}\|_{m,n} = \sup_{\substack{\mathbf{x} \in \mathbb{R}^m \\ \mathbf{x} \neq 0}} \frac{\|\mathbf{A}\mathbf{x}\|_m}{\|\mathbf{x}\|_n} = \sup_{\substack{\mathbf{x} \in \mathbb{R}^m \\ \|\mathbf{x}\|_n = 1}} \|\mathbf{A}\mathbf{x}\|_m. \tag{2.25}$$

Although the best type of matrix norm for numerical problems is the vector induced norms, calculating this norm is not trivial and is numerically difficult except for special cases. For a diagonal matrix it can be shown [54] $\|\mathbf{D}\|_p = \max_{1 \leq i \leq n} |d_{i,i}|$. In the current version only 1–norm and ∞–norm are used which are actually the only vector induced norms that can trivially be computed. It is easy to show [54] that the 1–norm is equal to the maximum column sum and ∞–norm is equal to the maximum row sum, or

$$\|\mathbf{A}\|_1 = \max_{1 \leq j \leq n} \|\mathbf{a}_j\|_1, \tag{2.26}$$

$$\|\mathbf{A}\|_\infty = \max_{1 \leq i \leq m} \|\mathbf{a}_i^T\|_1. \tag{2.27}$$

Another matrix norm commonly used is the Frobenius norm defined by

$$\|\mathbf{A}\|_F = \sqrt{tr(A^T A)}, \tag{2.28}$$

where $tr(\mathbf{B}) = \sum_{i=1}^{n} b_{i,i}$.

Ideally a forward error analysis which involves measuring the distance between the current iterate and the true solution, using some vector norm, should be used to stop the iterative solution of the linear system. However this is not possible since the true solution is not available and hence a backward error analysis should be used. Backward error is defined as the size of the perturbation matrix $\delta\mathbf{A}$ and the right hand side vector $\delta\mathbf{b}$ such that the computed iterate is the solution of the system $(\mathbf{A} + \delta\mathbf{A})\,\mathbf{x}^k = \mathbf{b} + \delta\mathbf{b}$ [54, 55]. A good stopping criterion should then [55]

(a) identify when the error $\mathbf{e}^k = \mathbf{x}^k - \mathbf{x}$ is small enough to stop,

(b) identify the rate of the convergence and stop if it is too slow and

(c) restrict the iteration time.

Last two conditions are conveniently satisfied by implementing a `maxit` variable that restricts the number of iterations. A backward error analysis should also be used to effectively bound the forward error and satisfy the conditions (a). We can bound the error using the equality

$$\mathbf{e}^k = \mathbf{x}^k - \mathbf{x} = \mathbf{A}^{-1}\left(\mathbf{A}\mathbf{x}^k - \mathbf{b}\right) = \mathbf{A}^{-1}\mathbf{r}^k, \tag{2.29}$$

which implies $\|\mathbf{e}^k\| \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}^k\|$. A commonly used criteria can be written as

$$\|\mathbf{r}^k\| \leq \texttt{tol}\|\mathbf{r}^0\|. \tag{2.30}$$

This simple criterion produces a forward error bound of $\|\mathbf{e}^k\| \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}^k\| \leq \texttt{tol}\|\mathbf{r}^0\| \cdot \|\mathbf{A}^{-1}\|$ and has no additional overhead on the solution process but depends strongly on the initial guess. For example if the algorithm starts from $\mathbf{x}^0 = \mathbf{0}$, and $\|\mathbf{b}\| \ll \|\mathbf{A}\| \cdot \|\mathbf{x}\|$, to a very high possibility the criterion will never be satisfied. A better criteria as suggested by Barrett et al. [55] would be

$$\|\mathbf{r}^k\| \leq \texttt{tol}\left(\|\mathbf{A}\| \cdot \|\mathbf{x}^k\| + \|\mathbf{b}\|\right), \tag{2.31}$$

which provides a bound for the forward error given by

$$\|\mathbf{e}^k\| \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}^k\| \leq \texttt{tol} \left(\|\mathbf{A}^{-1}\| \left(\|\mathbf{A}\| \cdot \|\mathbf{x}^k\| + \|\mathbf{b}\| \right) \right), \tag{2.32}$$

see [55]. This is a relatively simple stopping criteria and seems to be adequate in terms of accuracy and computational costs and hence is adapted for the current implementation, see [55] for other possibilities.

2.7.4 Solver tests

To test the solvers a simple 3D diffusion problem is defined on a cube where heat is conducted from a hot face at 300k to five other faces fixed at 200k. The iteration is continued to attain tolerance of 10^{-7} but note that the reported values in all the following figures are residuals calculated using Eq. (2.31). Figure 2.16 shows the final solution on the finest (128^3) grid. Figure 2.17 shows the slow convergence rate of the basic solvers making them only suitable as smoothers or preconditioners for other solvers. The main property of basic and incomplete factorization based solvers is their tendency to eliminate the high frequency errors very fast and smooth the error field. However the rate of convergence deteriorates as soon as the error field is smoothed out. This property is evident from the large slope of the residual curve for the first few iteration of all the basic and incomplete factorization based solvers in Figures 2.17 and 2.18. As expected the convergence rate drops quickly after the first few iterations.

Figures 2.19 and 2.20 show the convergence rate of the two implemented Krylov subspace solvers. In both cases the convergence rate is irregular and depends on the choice of the preconditioner. The BiCGSTAB solver has a smoother convergence and for the same preconditioners needs less iteration to converge. However this does not mean it is faster than the CG solver since it requires two times the number of operations in each iteration, see Table 2.2. Results of the GMRES solver are not presented here for the sake of brevity but generally the rate of convergence falls between that of the CG and the BiCGSTAB. However it requires more memory and timing shows that it is slower than both.

Figure 2.21 shows the convergence rate of the proposed multigrid solver. The solver converges faster that all the others and the convergence is smooth compared to the Krylov space solvers. The number of iterations to achieve the tolerance is equal for

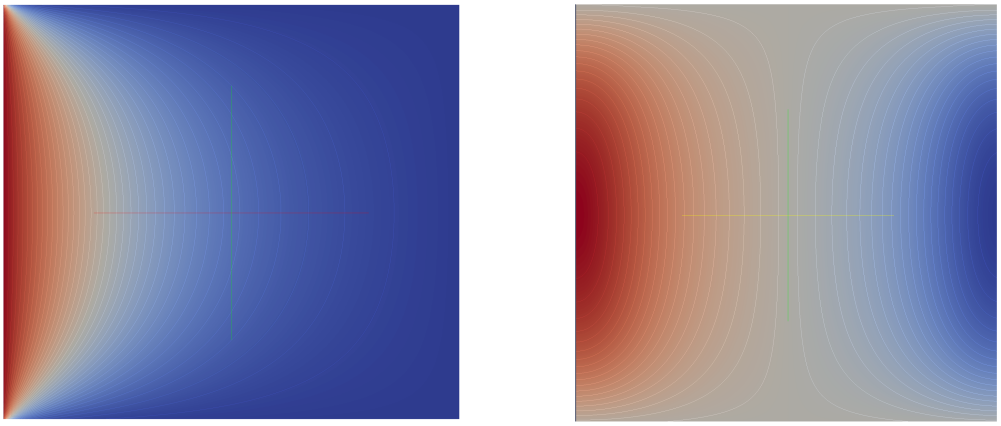
both MSI (Modified SIP) and SIP smoothers with MSI performing marginally better. Nevertheless SIP smoother is suggested due to its lower memory requirements. V and W cycles, see Section A.4.2 are also compared using same ILU smoother which shows that simpler V-cycle actually performs better which is also suggested for use in parallel environments due to lower communication costs.

Figure 2.22 shows the scalability of the solvers which is the main advantage of the multigrid methods compared to any other linear solvers. The test is only performed using multigrid solver and Krylov base solvers since the test requires very large number of iterations using the basic or incomplete factorization base solvers. Two grid levels are used for this test 64^3 and 128^3. The number of iterations is just one of the factors to measure the performance of a solver. One also needs to consider the convergence time since the multigrid solvers usually need more work per iteration due to restriction and prolongation operations. Table 2.2 summarizes the timing test for the three solvers discussed before on the 128^3 nodes. Obviously the number of iterations is the smallest for the multigrid solver but it requires a CPU time of almost three times the CG and two times the BiCGSTAB solvers per iteration. The CG solver although requires more iteration it converges more rapidly since it requires less operations per iteration and BiCGSTAB requires more CPU time than both other solvers. Also note that the time measured for one iteration, includes the time to factorize the preconditioner or smoother which is done only once therefore the $T_{convergance} <$ No. of Iter. $\times\, T_{one-iteration}$.

Obviously the multigrid solver with SIP or even a simpler smoothers such as standard ILU or even Jacobi, is the best choice for large systems due to its scalability. However conjugate gradient base solvers with special preconditioners such SIP or MSI have satisfactory performance on medium size problems. In addition the parallelization of the conjugate gradient solvers is straight forward and does not require a very delicate design. In contrast a multigrid solver, solves the system several times on very smaller scale subsystems and utmost care is required to make sure minimum amount of communication is performed when solving the coarse systems. For example one such strategy could be to solve the coarsest level on all cores using a direct solver. Therefore the parallelization strategies for the multigrid solver is not considered further in this thesis. CG, BiCGSTAB and SIP (only for some of the 2D cases) solvers are used to solve the linear systems in all the simulations in this thesis.

2.8 Conclusion

In this chapter several tools for efficient parallelization of the available CFD framework are implemented and discussed. A domain decomposition library based on the k–way graph partitioning is implemented and tested. A communication library for general CFD application is implemented and several optimization strategies are discussed and implemented. Finally linear solvers which are an integral part in any CFD application are studied in detail and several iterative methods are implemented for the current framework. Using multigrid accelerators is necessary for very large scale simulations. Therefore a multigrid accelerator is implemented but no attempt is made for the parallelization of the multigrid accelerator since it would require very careful implementation specially for coarse grid levels. This is essential for a scalable multigrid solver since the large communication overhead can quickly dominate any other performance gains acquired by using the multigrid accelerator. CG and BiCGSTAB solver with special ILU type preconditioner perform very well for medium scale problems and their parallelization is straightforward and hence are used for all the simulations in this study.



(a) Temperature Contours y-plane **(b)** Gradient Contours x-Plane

Figure 2.16: Final Solution on 128^3 grid - Temperature and the y–component of the temperature gradient contours of the final solution on the y– and x–planes respectively.

Solver	Grid Size	Number of Iteration	Time per iteration	Total Time
BiCGSIP	128^3	25	5.650	98.844
CGSIP	128^3	33	3.801	69.406
SIPMGV	128^3	5	10.079	30.406

Table 2.2: Solver Timing - Performance of three solvers on the finest grid.

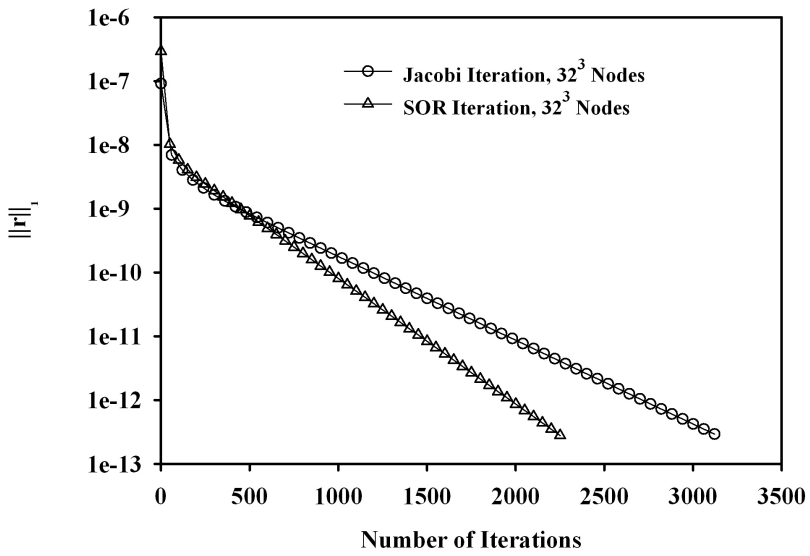


Figure 2.17: Basic solvers - slow convergence rate of basic iterative methods on a 32^3 mesh.

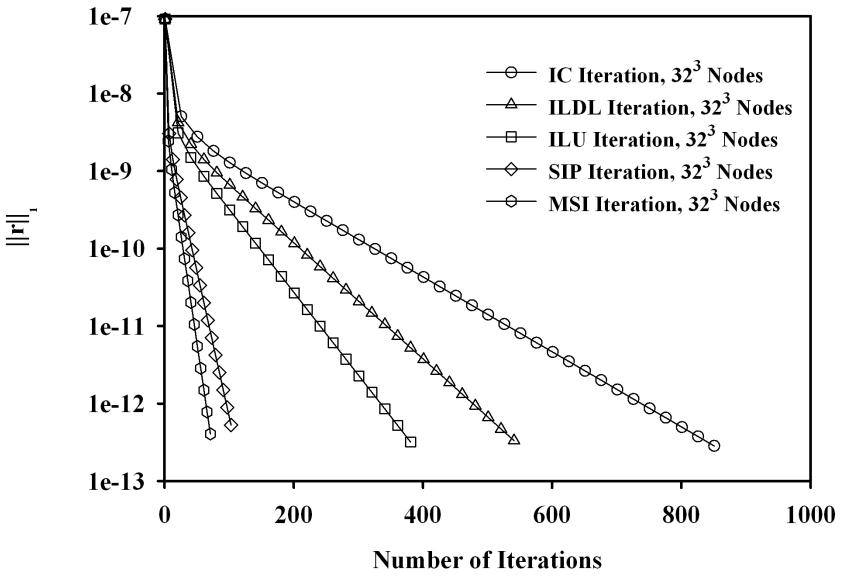


Figure 2.18: Incomplete factorization solvers - Fast elimination of small wave numbers of the error is evident in first few iterations.

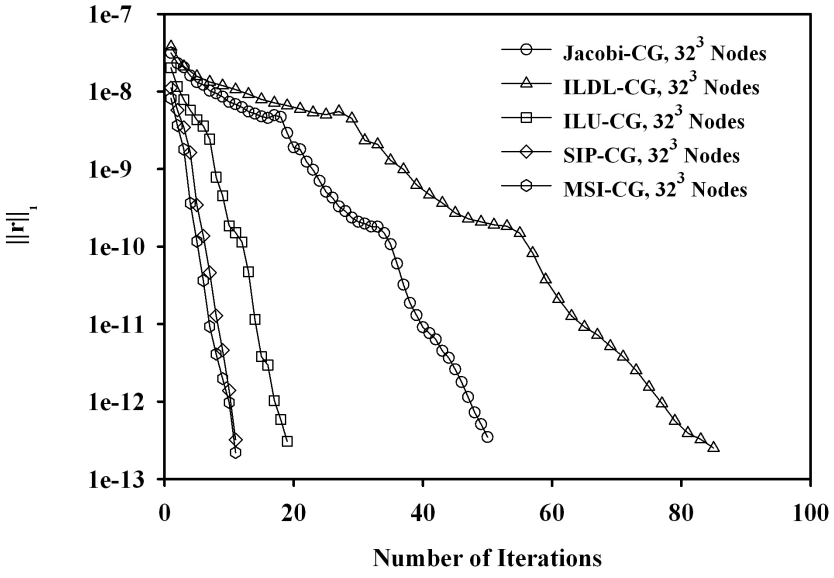


Figure 2.19: Conjugate gradient solver - Fast but irregular convergence rate. Dependence of the convergence rate on the choice of the preconditioner.

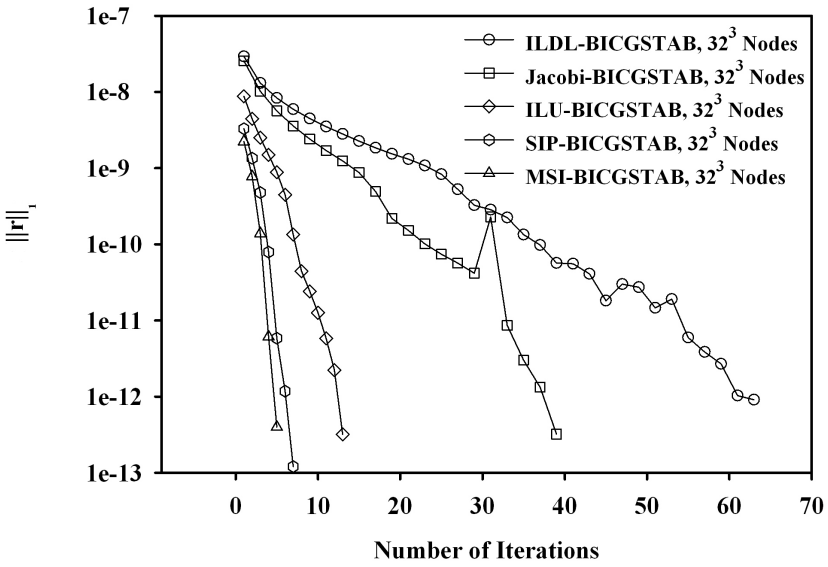


Figure 2.20: BiCGSTAB solver - Marginally faster convergence rate of the BiCGSTAB relative to the CG solver.

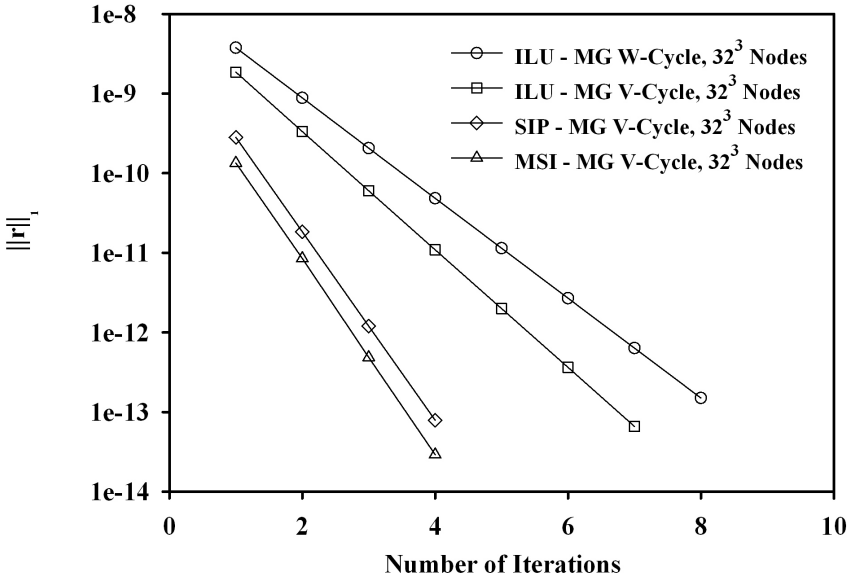


Figure 2.21: Multigrid solver - V-cycle performs slightly better with best convergence rate obtained for SIP or MSI smoothers.

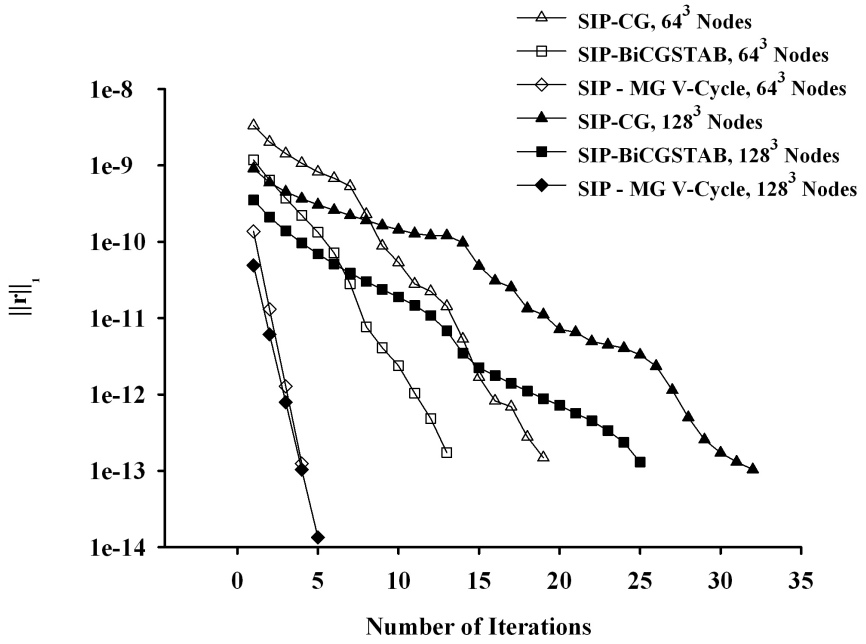


Figure 2.22: Scalability of the solvers - Number of iterations required to achieve a pre-specified tolerance on two different grid levels.

3

DNS methods

In chapter 1 the DNS methods applicable to particulate flows were summarized. Using fluid DNS and a point particle assumption for particle phase although is widely applied to particulate flows, can not capture very important phenomena such as the effects of particle shape, volume and wakes. In addition models used to calculate the forces acting on the particle are usually obtained from analytical solutions for low Reynolds number flows whilst the primary engineering need is for higher Re regimes.

The discussions in this chapter are based on the Navier-Stokes flow solvers. Another class of methods are the Lattice Boltzmann (LB) methods which have also been successfully applied to particulate flows [56, 57]. LB methods have later been combined with immersed boundary methods to improve some of the drawbacks and limitations of the original implementation by Ladd [56] in treating the boundary conditions on the surface of the particles [58, 59]. There is a relatively large body of work using LB methods which are outside the scope of this thesis.

In this chapter the currently available methods for particulate flows, which fully resolve the particle and account for finite particle dimensions, are discussed. To simulate the flow around immersed objects, the no slip condition on the boundary of immersed object should be considered whether the object is a stationary solid or a general deforming and moving object (i.e. $u_{f,i} = u_{b,i}$[1] on ∂V where ∂V is the boundary of immersed object and $u_{b,i}$ is its velocity). The methods can be classified under two general categories based on the treatment of the underlying mesh namely fixed mesh methods and

[1] lower case letters are used for physical quantities on Eulerian grid points and upper case letters for physical quantities evaluated on Lagrangian grid points.

body-conformal mesh methods. First body-conformal mesh methods are discussed and the tools needed for a successful implementation are considered. A general algorithm will be examined and it will be argued why these methods are usually complicated and extremely hard to adapt to the available Navier-Stokes solvers. Presumably an advantage of the body conformal mesh methods is their higher accuracy. However several remeshing and solution projections quickly deteriorate the accuracy of the solution and these methods are not necessarily more accurate than the fixed mesh methods. This will be discussed in more detail in Section 3.1.

Alternatively fixed mesh methods are discussed and categorized under two subcategories namely immersed boundary (IB) methods and fictitious domain/distributed Lagrange multiplier (FD/DLM) methods. Important variations of each method is considered and several algorithms are adapted for the SIMPLE type NS solvers. Numerical examples are performed to examine the effects of key concepts such as discrete delta functions and body forcing. 3–point delta functions are identified to be adequate for practical simulations in terms of accuracy and computational time. Calculation of surface properties such as hydrodynamic forces, which proves to be complicated in fixed mesh methods is studied and a novel approach based on a second order extrapolation of the stress tensor to the surface of the immersed object is proposed and tested. Based on the discussions in the following sections FD methods are identified to have more desirable properties, especially for the simulation of the particulate flows with heat transfer. The extension to heat transfer problems will be discussed in the next chapter.

A 2D NS solver based on the SIMPLE pressure correction algorithm [28] is implemented by the author to test the adapted IB and FD methods. Collocated grid arrangement is used in conjunction with the Rhie-Chow [60] flux correction method to ensure strong pressure–velocity coupling. Convective face fluxes are calculated using central difference approximation with deferred correction [7]. A version of the BiCGSTAB and SIP solvers explained in Section 2.7.1 are modified for the 5–diagonal systems and are used for the solution of the linear systems.

3.1 Body conformal mesh methods

Discretization of the solution domain produces a computational mesh on which the governing equations are subsequently solved. It also determines the positions of points

in space and time where the solution is sought. The procedure can be split into two parts: discretization of time and space. Since time is a parabolic coordinate [7], the solution is obtained by marching in time from the prescribed initial condition. For the time discretization, it is therefore sufficient to prescribe the size of the time-step that will be used during the calculation. Space discretization requires subdivision of the domain into control volumes, these control volumes are divided into two groups, structured and unstructured meshes.

A complex geometry usually requires an unstructured mesh which is complicated to generate and usually requires significant user control regarding the size and quality of the produced mesh. If the domain contains several stationary particles then although complicated, it is possible to use an unstructured grid generation algorithm to mesh around each object and start the solution algorithm. However if the particles are considered as general moving and deforming entities inside the domain then it is needed to remesh the whole domain at each time step taking into account the movement of the particles or their deformation. Figure 3.1 shows the remeshing around three particles in two different time steps. The remeshing procedure proves to be cumbersome and notable works using this method are Deforming-Spatial-Domain/Stabilized Space-Time (DSD/STT), Arbitrary-Lagrangian-Eulerian (ALE) and Fictitious Boundary Method (FBM).

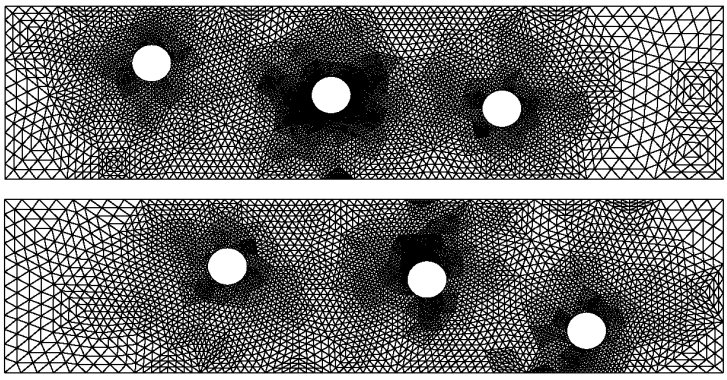


Figure 3.1: Remeshing due to particles motion - Remeshing is required when particles move in the domain top: at t, bottom: at $t + \delta t$.

DSD/STT was first proposed by Tezduyar et al. [61, 62] which included fluid-particle interaction computations in two dimensions and later extended to three dimensions with

parallel implementation by Johnson and Tezduyar [63, 64, 65]. The method is mainly applied to the blood flow simulations and cardiovascular fluid-structure interactions (FSI) [66–68]. Hu et al. [69, 70] and Gan et al. [71] simulated particle motions in both Newtonian and viscoelastic fluids in 2D and 3D using the ALE technique. Choi and Kim [72] used the same technique for the direct numerical simulation of arbitrary shape objects in shear flow but only one object is considered in their simulations. The fictitious boundary method (FBM) was first developed by Wan and Turek [73, 74, 75] and later combined with a new moving-mesh technique [76] incorporating mesh adaptation which can also be classified under the Fictitious Domain (FD) methods due to the use of a fixed background mesh. The simulations are restricted to 2-dimensional cases and as stated in [76] stability and efficiency issues are not their primary concern. Front-tracking methods are used both in the context of fixed mesh methods [77, 78] and body-conformal mesh methods [79–81]. However they are mainly used to study the motion of bubbly flows or flows with free surfaces and hence they will not be discussed further in this study. Main features of a body-conformal mesh method can be summarized as follows:

1. Flow solver: In almost all of the methods mentioned previously the standard finite-element method is adapted to solve the NS equations at each time steps. However if a parallel implementation is required, then the mesh needs to be redistributed on the processors after each remeshing step. Although there are very fast algorithms for domain decomposition that take into account load balancing and minimum communication cost constraints, see Section 2.2, the partitioning can take up to a few minutes for a mesh with $O(10^6)$ nodes which is much larger than the time required for one step advancement in time and hence significantly increases the computational cost.

2. Mesh movement: Tezduyar et al. [61, 62] used a linear elasticity model to calculate the mesh motion. When mesh movement takes place, these equations are solved to determine the relative internal node displacements of the mesh based on the given boundary displacements. Solving this system of equations to determine the displacements makes the method applicable to any mesh type and any type of movement. The added generality comes at the cost of solving this additional system each time the mesh is deformed. This equation can be written as

$$\frac{\partial \tau^*_{i,j}}{\partial x_j} + f^*_i = 0 \quad with, \quad \tau^*_{ij} = \lambda S^*_{ij}\delta_{ij} + 2\mu S^*_{ij}, \tag{3.1}$$

where S^*_{ij} is the rate of strain tensor defined by $\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$. It should be noted that Eq. (3.1) is not the equation of a real material and hence λ and μ should be treated as model parameters which are defined as functions of cell volume so that larger cells deform easier which is required to limit the remeshing frequency [63]. In the ALE approach the material derivative $(\frac{D}{Dt})$ in NS equations is redefined to take into account the mesh motion then a simple ODE of form $\frac{dX_i}{dt} = u_i$ with

$$\frac{\partial}{\partial x_j}\left(k^e \frac{\partial u_j}{\partial x_i}\right) = 0, \tag{3.2}$$

is solved to calculate the new nodal positions [70]. In Eq. (3.2) again k^e is chosen such that the larger cells absorb the largest deformations. Calculating the mesh motion also adds to computation costs due to the fact that the mesh motion ODEs although are very simple but for a high resolution simulation the number of nodes might be of order $O(10^7)$ which translates to solving a huge number of ODEs at each time step.

3. Remeshing: The mesh moving process needs to be interrupted and a new mesh should be generated once the mesh becomes tangled or too distorted and after the meshing process the solution of the previous mesh needs to be projected to new mesh locations. The Delaunay algorithm is usually used for mesh generation, however this does not have good parallel efficiency and should be performed serially [82]. Projecting the solution to new node position is not straightforward and puts a significant overhead on computational costs [65], Johnson and Tezduyar [82] used a sophisticated data structure to reduce this time but still this step can take up to 2 minutes for a medium sized problem with 1.75 million computational nodes distributed on 8 processors.

3.1.1 Application to particulate flows

The methods discussed in this section, although not very well-suited for the fully–resolved simulations (FRS) of particulate flows in terms of efficiency, are used for these type of simulations. Johnson and Tezduyar [64] used STT technique to study the

settling of 101 spheres in a liquid-filled tube. Later they extended the method to periodic boundaries and simulated the free–fall of spheres in a periodic domain with the maximum number of spheres reaching 125 [82]. The ALE technique was used by Hu et al. [70] to simulate the interaction of a pair of particles (spheres) in a Newtonian fluid and also 2D simulation of 90 particles in a pressure-driven flow. Gan et al. [71] used the ALE technique to study the effects of the heat transfer and the Grashof number on the free–fall motion of two 3D spheres. Wan and Turek [75] used the FBM method to simulate the sedimentation of a maximum of 10000 particles in a cavity. However all their simulations is limited to 2D cases and no 3D simulations is attempted.

3.2 Fixed mesh methods

In the previous section general steps for a successful body-conformal mesh method to simulate particulate flows is detailed. Despite using very complicated algorithms and data structures to reduce the costs associated with different steps, it is argued that mesh movement, remeshing, solution projection and for a parallel run, domain decomposition heavily add to the computational costs. For these reasons these methods do not appear to be promising for a high resolution 3D particulate flow simulation with a large number of particles. To the best knowledge of the author the only 3D parallel implementation of such algorithms is that of Johnson and Tezduyar [63, 64, 65] with serial subroutine calls for mesh generation on a dedicated CPU. Such serial function calls in parallel code seriously affects the scalability of the algorithm and efficiency would drop rapidly by increasing the number of CPUs. Even if there were very sophisticated and scalable algorithms for these steps the other drawback would be the fact that the methods are not adaptable to already available flow solvers, which mainly use structured 'stationary' grids. Therefore an entirely new implementation or a major revision of the available solvers is required.

The fixed mesh methods are based on the idea of solving the flow equation on a structured or even unstructured but stationary mesh (Eulerian grid), over the entire domain, and defining the objects inside the flow by adding Lagrangian points on their surfaces which may or may not coincide with the Eulerian grid. Then additional source terms are added to the NS equations to enforce the no-slip boundary condition or to restrict the flow inside the particle domain to rigid motion. The flow around the

object then resembles the flow where the boundary conditions are explicitly inserted. Presumably the main drawbacks of these methods is that firstly the flow is solved on the entire domain and more computational nodes are usually needed. However this does not always imply more computational cost due to the fact that in comparison with structured curvilinear body-conformal grids or unstructured grids, using a Cartesian fixed grid can significantly reduce the per-grid-point operation count due to the absence of additional terms associated with grid transformations or non-orthogonalities. In addition powerful geometric multigrid linear solvers are available that further reduce the computational costs for Cartesian fixed grids. Secondly and more importantly is the fact that the resulting accuracy is not always clear and there are stability issues in some of the variants [83]. However as will be discussed in this chapter there is an assortment of fixed mesh methods and it possible to choose between the accuracy and computational cost for different systems.

Although the fixed mesh methods are used both to simulate flow around rigid objects and general deforming solids with elastic properties to capture the fluid-structure interactions, here the methods applicable to rigid particles are of the main concern, therefore an exhaustive review of fluid–structure interaction is outside the scope of the current chapter.

3.3 IB Methods

Before classifying different IB methods the problem is generalized and an overview of the equations are given to clarify the key points in the solution procedure that result in different IB methods. In the IB methods a body forcing term is added to the NS equations at predefined boundary point to enforce the boundary conditions on these points. Domain boundaries are defined by Γ and Γ_p is used for the immersed boundaries and specifically for particle boundaries. $\Omega_d = \Omega_p + \Omega_f$ represents the entire solution domain and f and p subscripts are used to represent the physical domain occupied by fluid and particle respectively, see Figure 3.2. The same notation is used for the FD methods although a different notation is sometimes used in the literature. The modified momentum equations can be written as

$$\rho_f \frac{Du_{f,i}}{Dt} = \frac{\partial \sigma_{f,ij}}{\partial x_j} + f_i, \tag{3.3}$$

where the stress tensor is defined by

$$\sigma_{f,ij} = -P_f \delta_{ij} + \mu(\frac{\partial u_{f,j}}{\partial x_i} + \frac{\partial u_{f,i}}{\partial x_j}) + \delta_{ij}\lambda_f \frac{\partial u_{f,k}}{\partial x_k}. \tag{3.4}$$

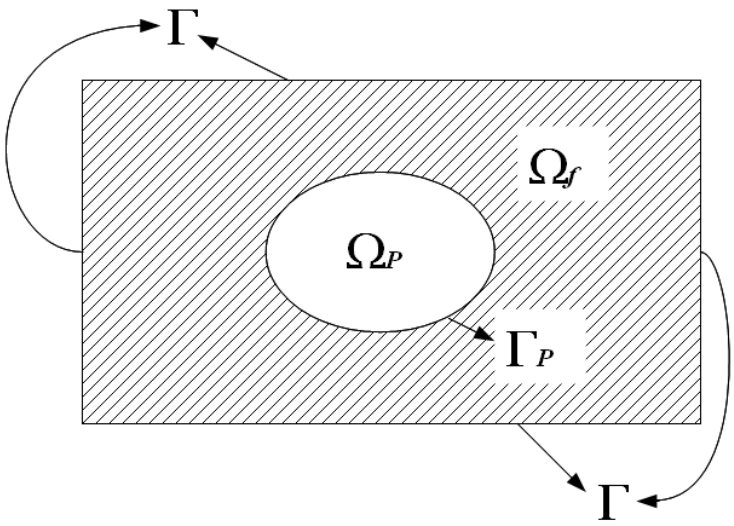


Figure 3.2: Computational domain with an immersed particle - Domain and particle boundaries, Γ and Γ_p, also fluid and particle domains, Ω_p and Ω_f are specified.

The method used for the implementation of the body forcing term f_i, leads to two main variants of the IB methods. In the continuous forcing approach, the forcing term f_i is calculated from a constitutive equation (in case of elastic boundaries) or a model equation and inserted back into Eq. (3.3) resulting in a equation of form $\mathcal{N}(u_i) = f_i$ ($\mathcal{N}$ is the NS operator) which is subsequently discretized on the whole domain Ω_d and solved using a linear solver. In the case of the discrete forcing approach, Eq. (3.3) is first discretized without the forcing term to yield discretized system $\mathscr{N}(u_i) = 0$, ($\mathscr{N}$ is a discrete NS operator) then the discretization near the IB cells (cells containing a surface point) is adjusted to account for the presence of the boundary which yields the modified system $\mathscr{N}'(u_i) = r_i$, where r_i is the extra term due to this modification, which is then solved on the entire domain [83]. Different IB methods and the effects of the

key concepts such as delta functions and body forcing will be examined in detail in the following sections.

3.3.1 Direct Forcing Methods

IBM was originally developed by Peskin [84, 85] to simulate the blood flow inside a beating heart taking into account the interactions between blood flow and muscle contractions. The original version of the method will be used to discuss several features of the method and then discuss the application of the method to the rigid boundary cases. In this method the boundary is defined by a series of Lagrangian points, then the fluid velocities interpolated on the Lagrangian points are used to move the points and a constitutive equation is used to calculate a forcing term which is then spread onto the surrounding cells. The equations to be solved for direct forcing approach are [86–89]

$$F_i(s,t) = \mathcal{A}_f X_i(s,t), \tag{3.5}$$

$$f_i(s,t) = \int_{\Gamma_p} F_i(s,t)\delta\left(\mathbf{x} - \mathbf{X}(s,t)\right) ds, \tag{3.6}$$

$$\frac{\partial X_i(s,t)}{\partial t} = U_i(\mathbf{X}(s,t)) = \int_{\Omega_d} u_i(\mathbf{x},t)\delta(\mathbf{x} - \mathbf{X}(s,t))dA. \tag{3.7}$$

Eq. (3.7) is written for 2D cases, however it can be easily generalized to 3D cases by replacing dA by dV. In Eq. (3.5), $\mathcal{A}_f$ is a force generating operator which is problem dependent, for instance in case of an elastic surface the force is simply the elastic tension force, thus we have [87]

$$\mathcal{A}_f = \gamma \frac{\partial^2}{\partial s^2}, \tag{3.8}$$

where γ is a stretching coefficient. Other forcing types will be discussed in detail for the case of rigid boundaries in the next section. X_i, x_i and s are Lagrangian position vector, Eulerian position vector and arc length respectively, see Figure 3.3.

Eq. (3.6) is the spreading operation and simply means that the summation of all forces along the boundary (integration along the arc length) results in the total Eulerian force on the fluid domain. Eq. (3.7) is the equation of the motion of the boundary and means that the boundary points move with velocity of the fluid at the same location. The discretization of Eq. (3.6) and (3.7) is not straightforward due to the presence of

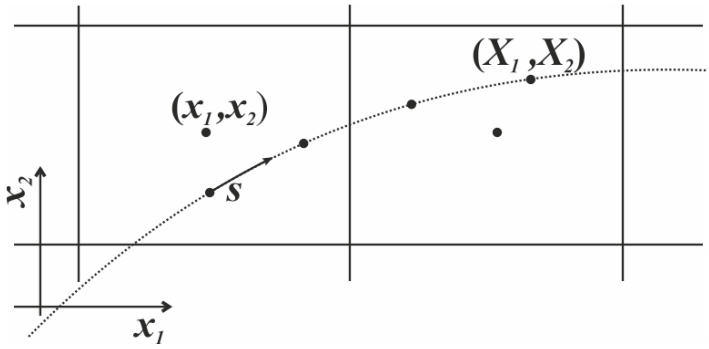


Figure 3.3: Original IB method - coordinate system used for the formulation of the original IB method.

the delta function. To effectively discretize Eq. (3.6) and (3.7) the delta function should be discretized. For 2D or 3D problems, see Figure 3.4, the discrete delta function is defined by

$$\delta_h(\mathbf{x}) = \frac{1}{h^d} \prod_{i=1}^{d} \phi(x_i), \tag{3.9}$$

where $d \in \{2, 3\}$ is the dimensionality of the problem. Function ϕ should have the following properties [90, 91]:

$$\phi(x) \quad \text{is continuous} \quad \forall x \in \mathbb{R} \tag{3.10}$$

$$\phi(x) = 0 \quad \forall |x| \geq 3 \tag{3.11}$$

$$\sum_{j_{even}} \phi(x - j) = \sum_{j_{odd}} \phi(x - j) = \frac{1}{2} \quad \forall x \in \mathbb{R} \tag{3.12}$$

$$\sum_{j} (x - j)^m \phi(x - j) = 0 \quad \forall m = 1, 2, 3 \quad \text{and} \quad \forall x \in \mathbb{R}. \tag{3.13}$$

Eq. (3.11) is required to control the cost of the computation otherwise each Lagrangian point would interact with all Eulerian grid points which is a huge computational overhead. Eq. (3.12) and (3.13) ensure that the transferred quantity between Eulerian and Lagrangian grids is conserved and that the quantity is spread equally on

odd and even grids. Note that the summations are over integer j values but they should hold for any real value $x - j$, i.e $\forall x \in \mathbb{R}$. Several researchers proposed many variations for ϕ, for first and second order accuracies and with different amount of smoothing (2, 3, 4 and 6 cell functions are the most commonly used). Most commonly used functions are

- 2–point hat function [92]

$$\phi(x) = \begin{cases} 1 - |x| & |x| \leq 1 \\ 0 & |x| > 1 \end{cases} \tag{3.14}$$

- compact 3–point function [93]

$$\phi(x) = \begin{cases} \frac{1}{3}\left(1 + \sqrt{1 - 3x^2}\right) & |x| \leq 0.5 \\ \frac{1}{6}(5 - 3|x| - \sqrt{1 - 3(1 - |x|)^2}) & 0.5 < |x| \leq 1.5 \\ 0 & |x| > 1.5 \end{cases} \tag{3.15}$$

- 4–point regularized delta function [90]

$$\phi(x) = \begin{cases} \frac{1}{8}(3 - 2|x| + \sqrt{1 + 4|x| - 4x^2}) & |x| \leq 1 \\ \frac{1}{8}(5 - 2|x| - \sqrt{-7 + 12|x| - 4x^2}) & 1 < |x| \leq 2 \\ 0 & |x| > 2 \end{cases} \tag{3.16}$$

- 6–point delta function [91]

$$\phi(x) = \begin{cases} \frac{61}{112} - \frac{11}{42}|x| - \frac{11}{56}|x|^2 + \frac{1}{12}|x|^3 + \frac{\sqrt{3}}{336}\left(243 + 1548|x| - \right. & \\ \left. 748|x|^2 - 1560|x|^3 + 500|x|^4 + 336|x|^5 - 112|x|^6\right)^{0.5} & |x| \leq 1 \\ \frac{21}{16} + \frac{7}{12}|x| - \frac{7}{8}|x|^2 + \frac{1}{6}|x|^3 - \frac{3}{2}\phi(|x| - 1) & 1 < |x| \leq 2 \\ \frac{9}{8} - \frac{23}{12}|x| + \frac{3}{4}|x|^2 - \frac{1}{12}|x|^3 + \frac{1}{2}\phi(|x| - 2) & 2 < |x| \leq 3 \\ 0 & 3 < |x| \end{cases} \tag{3.17}$$

In Eqs. (3.14)–(3.17), x is a dummy variable. Figure 3.4 shows the values of delta function evaluated on a two-dimensional grid using Eq. (3.16) to evaluate ϕ values and Eq. (3.9) to calculate the 2D delta function. Cell centre values are evaluated assuming that the Lagrangian point is in the middle of 4 'circular' cell points both in x and y direction. Also note that the discrete delta function properties mentioned earlier can easily be verified by this simple example.

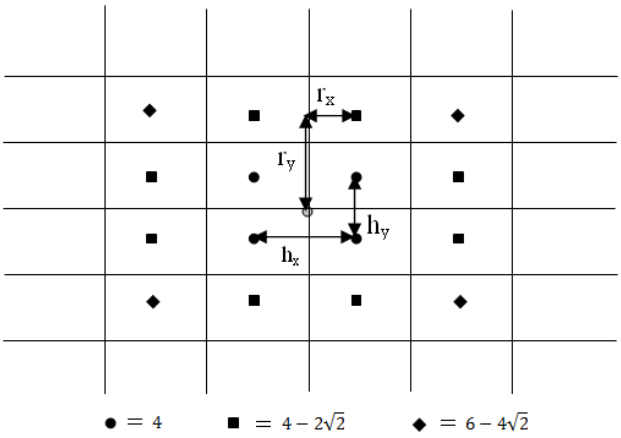


Figure 3.4: A 2D delta function - Eq. (3.9) evaluated on a 2D grid using Eq. (3.16), coefficient $\frac{1}{64h^2}$ dropped for clarity.

Figure 3.5 shows different delta functions defined above with 2, 3, 4 and 6 cell support. Also a cosine function with 4–point support [90] is presented which has similar plot to that of the 4–point regularized delta function but it is suggested to use the regularized function since the evaluation of the cosine function is generally more expensive [90].

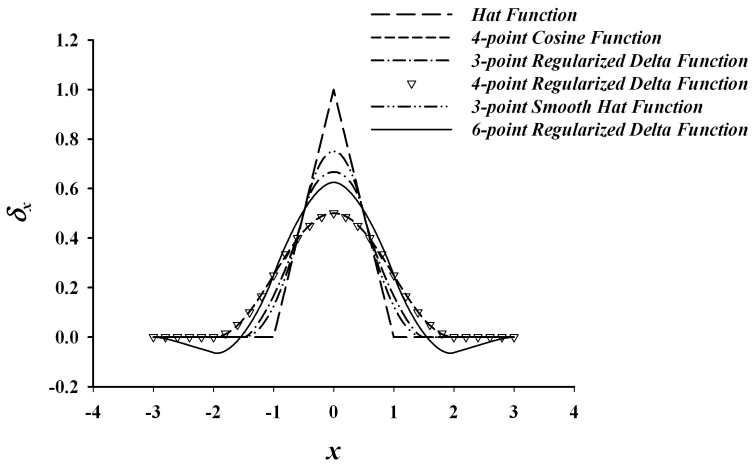


Figure 3.5: Delta functions - Different delta functions with 2, 3, 4 and 6 cell support.

Yang et al. [94] proposes smoothed version of the 2, 3 and 4 point functions by integrating the conventional functions using

$$\phi'(x) = \int_{r-0.5}^{r+0.5} \phi(x')dx', \tag{3.18}$$

which only increases the support by half a cell width. Functions corresponding to standard 2, 3 and 4-point delta functions are derived using Eq. (3.18) and the equations can be found in Yang et al. [94].

Having defined the discrete delta functions, the integral in Eq. (3.6) can be discretized as follows

$$f_i(\mathbf{x}) = \sum_{m=1}^{N} F_i\left(\mathbf{X}_m\right) \delta_h \left(\frac{\mathbf{x} - \mathbf{X}_m}{h}\right) \upsilon_m \quad \forall \mathbf{x} \in g_h, \tag{3.19}$$

where g_h is the support of the delta function, and N is the number of Lagrangian points defining the boundary. υ_m can be interpreted differently depending on how the force $F_i\left(\mathbf{X}_m\right)$ is defined. For example for elastic boundaries F_i is naturally defined as a force per unit surface area and hence $\upsilon_m = \Delta S_m$, where ΔS_m is the surface area of an element defining the surface. In other methods and particularly in the FD methods, F_i is a force per unit volume and υ_m is the volume of a CV defining the object. This might be a source of confusion and will be discussed in more detail when discussing the specific methods. Velocity interpolation to particle locations, Eq. (3.7), can be discretized using any discrete delta function δ_h, viz

$$U_{m,i} = \sum_{\mathbf{x} \in g_h} u_i(\mathbf{x}) \delta_h \left(\frac{\mathbf{x} - \mathbf{X}_m}{h}\right) h^d, \quad 1 < m < N. \tag{3.20}$$

In Section 3.3.1.2 several numerical tests are performed to determine the properties of the delta functions and also the computational costs of evaluating different functions.

3.3.1.1 Interpolation of the total stress tensor

Calculation of the total surface forces is required for the estimation of the drag and lift forces and also if the free motion is considered in IB methods. Total surface force is given by

$$F_{s,i} = \int_{\Gamma_p} \sigma_{ij} n_j ds, \tag{3.21}$$

where σ_{ij} is defined by Eq. (3.4), n_j is the unit outward normal of the surface element. To perform the integration the value of the total stress tensor on the surface is required

which required several interpolations. These interpolations are not straight forward in fixed mesh methods and simple interpolations cause severe underestimation of the stress tensor. Therefore there has been several attempts to use other methods to calculated the total surface force instead of direct integration. Saiki and Biringen [95] calculated the force by calculating the momentum deficit for a 2D stationary case. Later Balaras [96] generalized the equation for 3D force calculations by integrating the NS equations on a bounding box. Lai and Peskin [92] showed that the total force acting on the object is equal to the sum of the body forces on the Eulerian grid which is again only applicable to stationary IB methods and excessive bookkeeping is required if multiple objects are present in the simulation. Shen et al. [97] extended the Lai and Peskin [92] formulation for moving objects. Wan and Turek [76] suggested using the gradient of a step function

$$\Theta_p = \begin{cases} 1 & x \in \Omega_p \\ 0 & x \in \Omega_f, \end{cases} \tag{3.22}$$

to convert the surface integrals to volume integrals which is also used by Blasco et al. [98] for their FD method. Using the step function, for example the surface forces can be calculated by

$$F_{s,i} = \int_{\Omega_p} \sigma_{ij} \frac{\partial \Theta_p}{\partial x_j} dv. \tag{3.23}$$

Eq. (3.23) is easily derived from Eq. (3.21) by noting that the gradient of the step function is zero everywhere except on the surface of the object where it is equal to the unit normal vector to the particle surface i.e $n_j = \frac{\partial \Theta_p}{\partial x_j}$. This equation simplifies the calculations since Eq. (3.23) need only be computed in a narrow band around the particle surface. However the only general treatment applicable to all systems is the explicit integration of Eq. (3.21).

In this section an algorithm for general 3D interpolation of the total stress tensor σ_{ij} to the surface is provided which is required for the direct integration of the surface forces. Let $r_{c,i}$ be the position vector of the centre of the surface element, see Figure 3.6, and r'_i and r''_i be two points on the surface normal such that $||r'_i - r_{c,i}|| = ||r''_i - r'_i||$.

Having defined two auxiliary points r'_i and r''_i on the surface normal, interpolation molecules are created around each auxiliary point consisting of eight surrounding CVs. The interpolation molecules are created such that none of the constituent CVs fall into the particle domain. Following trilinear interpolation

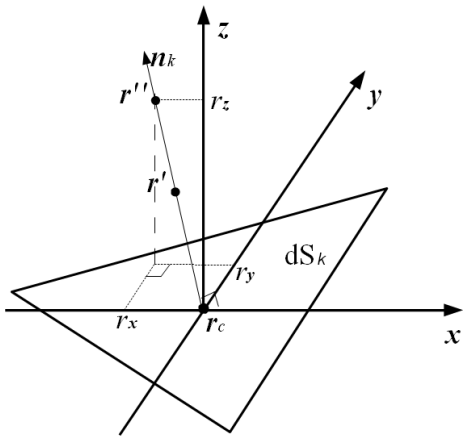


Figure 3.6: Total stress interpolation - schematic presentation of a surface element and its normal in addition to auxiliary points defined on the normal for interpolation. Vectors are presented by bold symbols.

$$\phi(x,y,z) = C_1xyz + C_2xy + C_3xz + C_4yz + C_5x + C_6y + C_7z + C_8, \tag{3.24}$$

can then be used to present the value of a generic variable in the region between the eight nodes surrounding each auxiliary point, see also Press et al. [99]. Then the coefficients $\mathbf{C}^T = [C_1 \cdots C_8]$ can be found by solving

$$\mathbf{C} = \mathbf{V}^{-1}\mathbf{\Phi}, \tag{3.25}$$

where $\mathbf{\Phi} = [\phi_1 \cdots \phi_8]$ is the value of the variable at the eight surrounding CVs. $\mathbf{V}$ is the Vandermonde matrix given by [99]

$$\mathbf{V} = \begin{bmatrix} xyz|_1 & xy|_1 & xz|_1 & yz|_1 & x|_1 & y|_1 & z|_1 & 1 \\ xyz|_2 & xy|_2 & xz|_2 & yz|_2 & x|_2 & y|_2 & z|_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ xyz|_8 & xy|_8 & xz|_8 & yz|_8 & x|_8 & y|_8 & z|_8 & 1 \end{bmatrix}. \tag{3.26}$$

Having the coefficients $[C_1 \cdots C_8]$ value of pressure and stress tensor at two auxiliary points can be calculated using Eq. (3.24) and the value at the surface element centre can be calculated by simple second order extrapolation

$$\phi(\mathbf{r}_c) = \phi(\mathbf{r}'') + \frac{||r_i'' - r_{c,i}||}{||r_i'' - r_i'||}(\phi(\mathbf{r}') - \phi(\mathbf{r}'')). \tag{3.27}$$

Eq. (3.25) is a small 8 by 8 matrix and is best solved by a direct solver. However for each surface element the system should be solved 10 times to find the coefficients for 9 elements of the stress tensor and the pressure. By noting that only the right hand side is different for these 10 systems a fast **LU** decomposition direct solver capable of handling multiple right hand sides, see Anderson et al. [100], can be a good choice to get the coefficients for each of 10 required variables. Note that in 2D Eq. (3.24) reduces to a bi–linear interpolation with four coefficients. Consequently Eq.(3.26) reduces to a 4 by 4 matrix and Eq. (3.25) should be solved 5 times to find the coefficients for 4 elements of the stress tensor and the pressure for each surface element.

3.3.1.2 Direct forcing - Rigid Boundaries

Eq. (3.8) can theoretically be used for rigid boundaries assuming a very stiff elastic body. However this equation is not well posed in the rigid limit. Lai and Peskin [92] addressed this problem and suggested considering the structure attached to an equilibrium state by a stiff spring with restoring force defined by

$$F_{m,i} = -\kappa \left(X_{m,i} - X_{m,i}^e\right), \tag{3.28}$$

where κ is the spring constant and superscript e stands for the equilibrium position. To apply the boundary condition on the IB surface accurately large values of κ are needed which results in a stiff system of equations. Using lower values results in spurious elastic effect such as large deviation from the equilibrium [83, 92, 101].

Another approach is that developed by Goldstein et al. [102] and later used by Saiki and Biringen [95] to simulate flow around circular cylinder up to $Re = 500$, with Reynolds number calculated based on the cylinder diameter and far field velocity. A good comparison between forcing schemes suggested by Lai and Peskin [92] and that suggested by Goldstein et al. [102] (sometimes referred to as Virtual Boundary (VB) or feedback forcing) can be found in Shin et al. [103]. The ideas of control theory are used and the forcing term is defined such that it controls the velocity difference error. This is similar to a PI (proportional–integral) controller using a term proportional to the current error and a term corresponding to the error history (integral of the error) such

that it controls the flow velocity on the surface to eliminate the velocity error (difference between the real velocity of the surface and flow velocity interpolated on the surface). The forcing term can be written as

$$F_{m,i} = \alpha \int_0^t \left(U_{m,i}(\tau) - U_{m_r,i}(\tau)\right) d\tau + \beta \left(U_{m,i}(t) - U_{m_r,i}(t)\right), \tag{3.29}$$

where α and β are large negative model parameters and should be specified by the user. $U_{m,i}$ and $U_{m_r,i}$ are the interpolated flow velocity at the position of the Lagrangian points defining the boundary, Eq. (3.20), and the real velocity of the Lagrangian points. The real velocity can be derived using the equations of motion of the particle or is predefined (e.g $U_{m_r,i} = 0$ for a stationary object). It should be mentioned here that the dimensionality of Eq. (3.29) suggests that in this method $F_{m,i}$ is a force per unit volume. But since the dimensionality of the surface element is always one less than the dimensionality of the problem, using $\upsilon_m = \Delta S_m$, where ΔS_m is the surface element area (3D simulation) or length (2D simulation), in Eq. (3.19) results in wrong dimensionality. Therefore originally Saiki and Biringen [95] used a volume averaging to transfer the forces but later Shin et al. [103] used the same method proposed by Uhlmann [104] and set $\upsilon_m = \Delta V_\ell$. ΔV_ℓ is the assumed volume (area) of the surface element which is defined such that $\Delta V_\ell \approx h^d$, see [103, 104]. However although this definition is necessary for the method proposed by Uhlmann [104] it is not necessary for the direct forcing method. Since α and β are merely two adjustable and problem dependent parameters they can be assumed to have different dimensionality such that $F_{m,i}$ is a force per unit area and hence one can simply use Eq. (3.19) with $\upsilon_m = \Delta s$. In any case it is important to monitor the value of velocity difference error and tune the values of the adjustable parameters to minimize the error.

This slightly modified VB method is adapted for the current framework as the first proposal due to its simplicity and independence from the underlying solver. To discuss some of the properties of this method the problem of an oscillating cylinder in transverse direction in a uniform flow is considered. The cylinder is oscillating harmonically according to

$$y_c(t) = y_c(t_0) + A_m \sin(2\pi f t), \tag{3.30}$$

where y_c is the position of the centre of the cylinder, A_m is the oscillation amplitude, and f is the oscillation frequency. The Reynolds number for this problem is defined by $Re = \rho_f U_\infty D/\mu_f$, where U_∞ and D are the far field velocity (equal to inlet velocity) and the diameter of the cylinder respectively. For this simulation a domain with dimensions $30D \times 10D$ is considered and the sphere is initially placed at $(x_c/D, y_c/D) = (10, 5)$. The boundary conditions at the top and bottom wall are set to slip wall conditions with zero normal gradient and standard inlet and outlet boundary conditions are used in the flow direction. The inlet velocity $U_{in} = 0.5$ and the fluid density $\rho_f = 1$. Using delta functions to transfer variables between Eulerian and Lagrangian quantities restricts the type of mesh to uniform structured meshes. Therefore it is not possible to cluster the structured grid in the domain where higher resolution is required. A method to overcome this restriction is to use uniform grid in a small box around the immersed object and stretch the mesh beyond that box. Ordinary hyperbolic stretching functions [105, 106] provide one or two parameters to control the amount of stretching in one or both coordinate directions but provide no control on the initial or final grid spacing. To produce a consistent mesh the slope of the stretching function is required to vary such that the first node just before or after the box has a spacing equal to that of the first node in the uniform mesh box. This is achieved using a special type of stretching function suggested by Vinokur [107] which is used for all the simulations in this thesis whenever mesh stretching is used for higher resolution.

Mesh spacing around the object is set to $0.01D$ which is equivalent to 100 nodes inside the rigid body and 1500 Lagrangian forcing points are used to define the surface of the cylinder. Values of the parameters α and β in Eq. (3.29) are set to -100 and -1 respectively. The drag and lift coefficients are defined by

$$C_D = 2F_x/(\rho_f U_\infty^2 D), \tag{3.31}$$

$$C_L = 2F_y/(\rho_f U_\infty^2 D), \tag{3.32}$$

where F_x and F_y are the forces in x and y directions calculates using a 2D version of the method suggested in Section 3.3.1.1. Other parameters for this simulation are $A_m/D = 0.2$ and $f/f_0 = 0.9$ where f_0 is the natural shedding frequency calculated from the Strouhal number which is $St = 0.19$ for $Re = 185$ [108]. Some numerical tests are

first performed to determine the properties of delta functions and their computational costs to identify the optimum choice.

To measure the computational cost of using different delta functions, 500 iterations in a single time step are performed and the interpolation, Eq. (3.20), and force distribution, Eq. (3.19), procedures are timed. This test can be considered as a generic numerical test on the computational costs of evaluating delta functions disregarding the fixed grid method used. Table 3.1 compares the mean time of 500 iteration, $\bar{t}$, and the percent of the total iteration time, $\frac{\bar{t}}{\bar{t}_{iter}} \times 100$. Evidently the computational cost of 6–point delta function can be as large as 1.5% of the whole iteration whereas the cost of evaluating 2– and 3–point functions is negligible. If a parallel implementation is sought using a delta function with more than 3–cell support can also significantly increase the communication costs which makes them inappropriate for parallel implementations.

Equation	$\bar{t}$ (sec)	$\bar{t}/\bar{t}_{iter} \times 100$
Eq. (3.14)	2.79E-4	0.087
Eq. (3.15)	8.65E-4	0.267
Eq. (3.16)	1.47E-3	0.463
Eq. (3.17)	4.33E-3	1.340

Table 3.1: Comparison of computational costs of different delta functions.

To identify the accuracy and smoothing properties of the different delta functions the simulation is performed using 2, 3 and 6–point delta functions. Figure 3.7 shows the changes in the drag coefficient for three different delta functions for a full oscillation period plotted against the position of the centre of the cylinder. The 2–point delta function produces a large oscillation in the calculation of the hydrodynamic forces and also over–estimates the drag coefficient which can be related to the generation of unphysical flows due to the use of 2–point functions. Evidently using a 6–point function does not have any significant effect on the oscillations in the calculation of the hydrodynamic forces. These numerical tests show that a 3–point delta function provides a good trade–off between the accuracy and the computational costs. Delta functions with larger support can provide better smoothing and accuracy for specific controlled numerical tests, however for practical purposes a 3–point functions appears to be sufficient.

Evidently the proposed method is capable of capturing complicated hydrodynamic phenomena if optimized values for parameters α and β are used. The parameters are

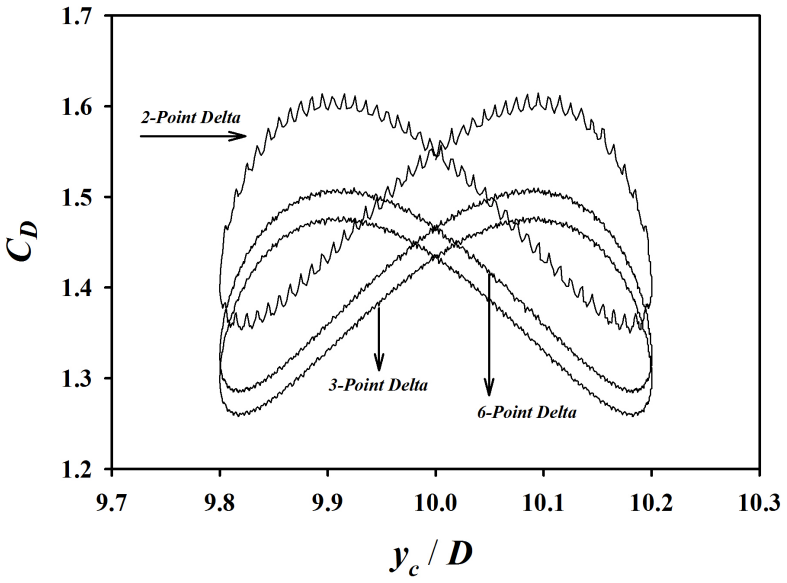


Figure 3.7: Effects of delta function on hydrodynamic forces - 2–Point delta function generates oscillations in calculating the drag force. 3–point function eliminates the oscillations produced by 2–point function. 6-point delta function produces essentially the same results in terms of the oscillations.

not known a priori and are problem dependent. It is possible to set α and β to very large numbers but it is known that the method is stable only for $\Delta t < (-\beta - \sqrt{\beta^2 - 2\alpha k})/\alpha$ where k is a problem dependent constant in order of unity [102]. Therefore trial and error may be required to find good values for the parameters. This is very time consuming and is a significant waste of resources for large scale problems. This is the main drawback of the method. In this method it is necessary to monitor the error in some norm to make sure the boundary conditions are satisfied to a high precision. Figure 3.8 shows the infinity norm of the error defined by

$$\|\epsilon\|_\infty = \max\left(|U_{1,i} - U_{1_r,i}|, \cdots, |U_{N,i} - U_{N_r,i}|\right), \tag{3.33}$$

during the simulation with different delta functions. Evidently smoother delta functions also cause easier control of the velocity on the Lagrangian points using the same parameters. Again note that both δ_3 and δ_6 provide very accurate results with the maximum error never exceeding 0.6%. However δ_2 errors are still acceptable and never exceed 3%.

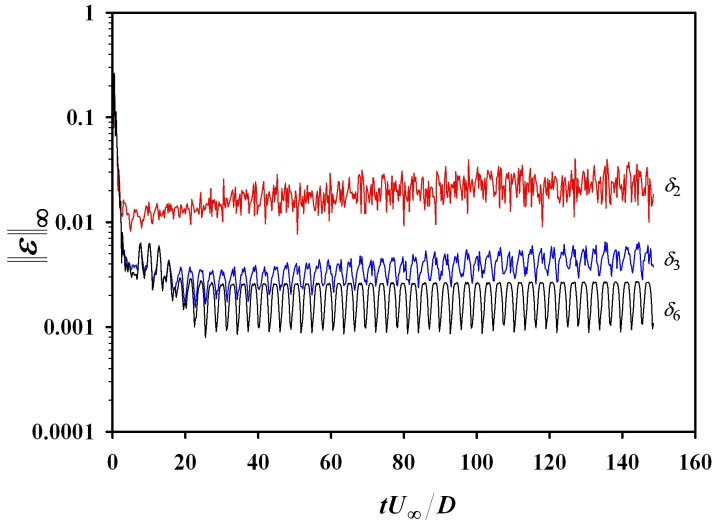


Figure 3.8: Time history of the controller error - smoother delta functions facilitate the control of the velocity error in VB method using the same parameters ($\alpha = -100$, $\beta = -1$).

Figure 3.9 shows the streamlines around the transversely oscillating cylinder at its peak amplitude. Increasing the oscillation frequency from $f/f_0 = 0.9$ to $f/f_0 =$

1.1 a drastic change in the topology of the streamlines is observed. The streamlines connect at higher f/f_0 to form a saddle point which is due to the vortex switching phenomena [108]. Figure 3.10 shows the the switching from one side of the cylinder to the other side. At higher oscillating frequencies the core of the top vortex moves towards the cylinder and its length decreases such that the bottom vortex becomes the dominant one. Figure 3.11 shows the pressure contours around the oscillating circular cylinder. Due to the smoothing of the forces provided by the 3–point delta function no severe point–to–point oscillation is observed in this figure. Figure 3.12 shows the time history of the drag and lift coefficients. Once the vortex shedding is established at $f/f_0 = 0.9$ a uniform sinusoidal graph is observed but at $f/f_0 = 1.1$ a higher harmonic can also be identified. Tables 3.2 compares the mean drag coefficient and the *RMS* lift coefficient for both cases to those calculated by [108]. Where $C_{L,RMS}$ is calculated by

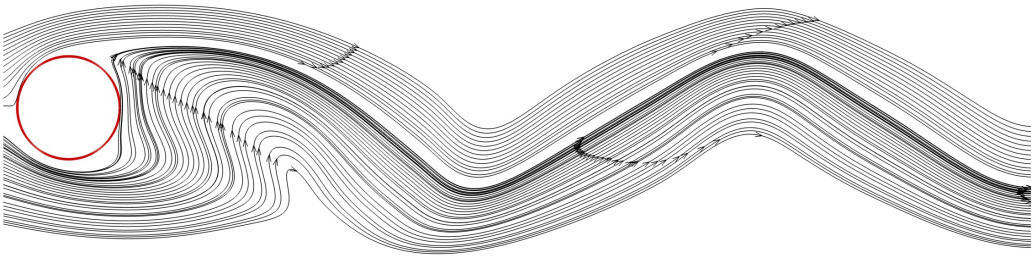
$$C_{L,RMS} = \sqrt{1/n \sum_{i=1}^{n} (C_{L,i})^2}. \tag{3.34}$$

Study	f/f_0			
	0.9		1.1	
	C_D	$C_{L,RMS}$	C_D	$C_{L,RMS}$
Current	1.40	0.22	1.47	0.93
Guilmineau and Queutey [108]	1.33	0.19	1.36	0.87

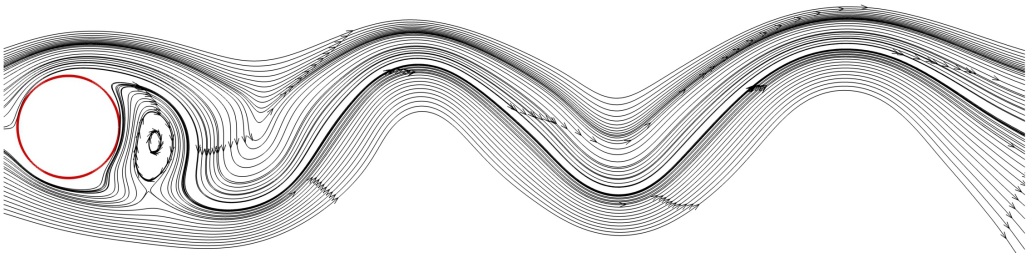
Table 3.2: Comparison of the $\bar{C}_D$ and $C_{L,RMS}$ for a transversely oscillating circular cylinder.

Another method that can be classified under direct forcing approach is that of Angot et al. [109] and Khadra et al. [110]. In this method the entire domain is assumed to be porous, then a volume drag, called the Darcy drag, is added to the NS equations, a term which represents the action of the fictitious porous medium over the flow. They applied this method to rigid particles by assuming that the rigid particles are porous media with zero permeability. Then one single global set of Navier-Stokes-Brinkman (NSB) equations is solved over entire domain Ω_d, NSB equation can be written as

$$\frac{Du_{f,i}}{Dt} = -\frac{1}{\rho_f}\frac{\partial P_f}{\partial x_i} + \nu \frac{\partial u_{f,i}}{\partial x_j \partial x_j} + \frac{\nu}{K} u_{f,i}, \tag{3.35}$$



(a) $f/f_0 = 0.9$

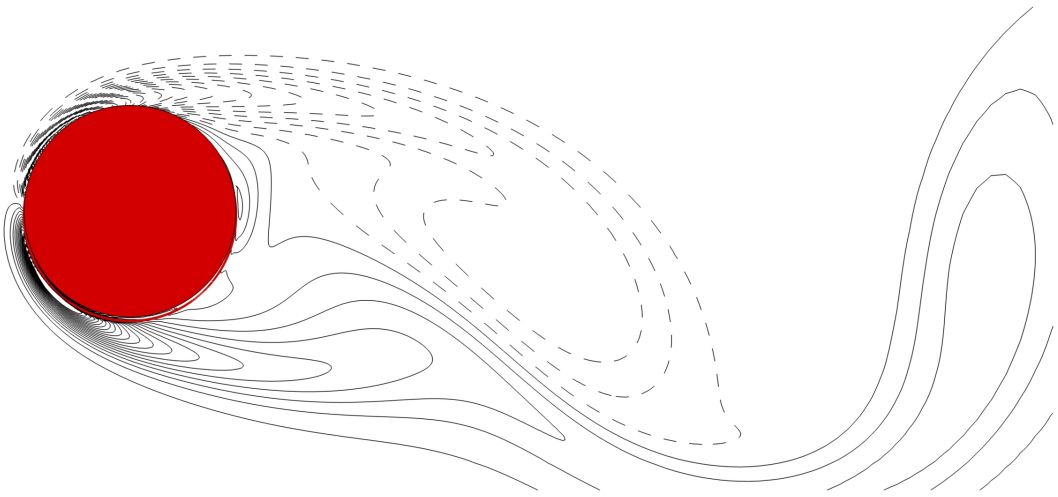


(b) $f/f_0 = 1.1$

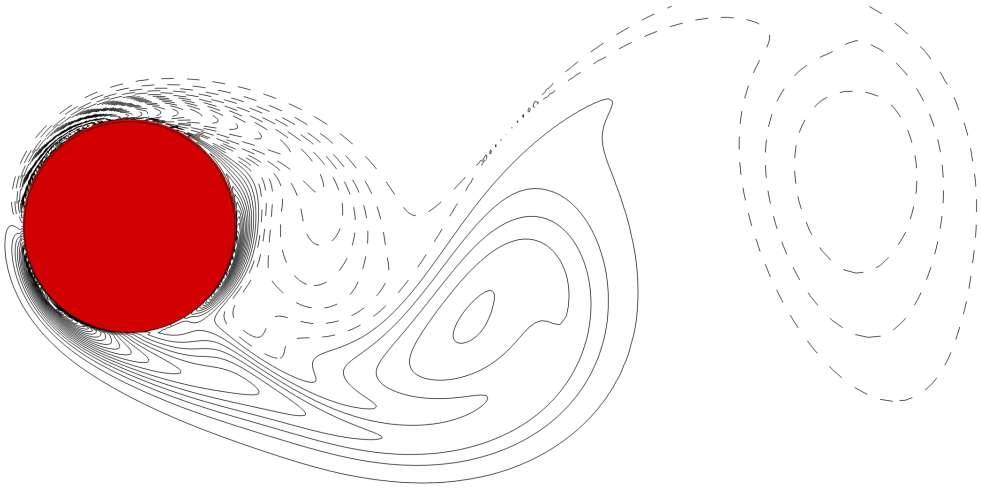
Figure 3.9: Streamlines of flow around a transversely oscillating cylinder at $Re = 185$ - Changes in the topology of the streamlines by moving from $f/f_0 = 0.9$ to $f/f_0 = 1.1$ due to the vortex switching.

where K is the permeability and is set to 0^+ in Ω_p and to $+\infty$ in Ω_f. This formulation again is in fact another simplified version of Eq. (3.29) with $\alpha = 0$ and $\beta = \frac{\nu}{K}$ and consequently susceptible to same stability problem. A subtle difference between porous medium method of Khadra et al. [110] and the other two forcing schemes is that the entire particle domain has a physical interpretation in the porous medium method whereas in the other two methods only the surface is defined by the Lagrangian points and thus the flow occurring inside the object should be neglected.

Application of these methods to particulate flows is limited however their implementation is very easy and independent of the underlying flow solver. Fogelson and Peskin [111] used a version of the original Peskin [84] method to simulate stokes flow around 500–1000 2D circular disks. Lima E Silva et al. [112] used a modified version of VB method (without adjustable parameters) to simulate flow around a single circular cylinder and free-fall of a single circular cylinder (2D). Porous medium methods of Angot et al. [109] and Khadra et al. [110] are both used in the context of 2D flows and a number of simple test cases are provided however no attempt to simulate 3D flows or large number of particles is made.

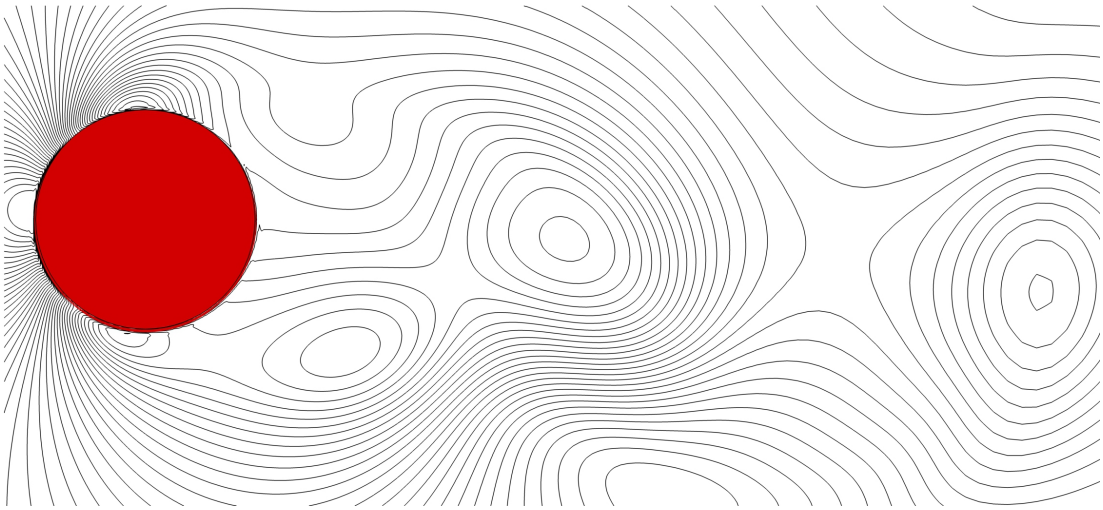


(a) $f/f_0 = 0.9$

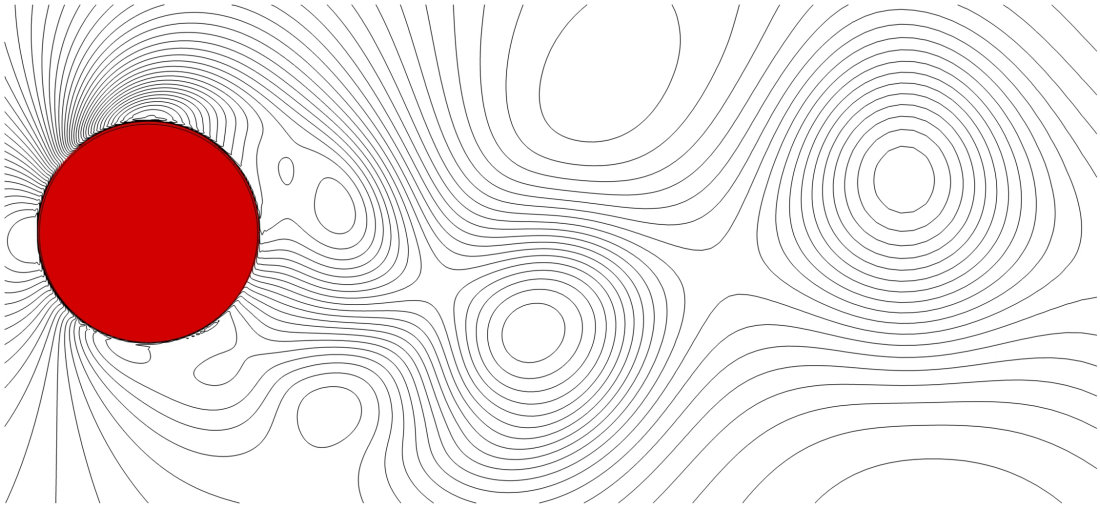


(b) $f/f_0 = 1.1$

Figure 3.10: Z-vorticity contours for flow around a transversely oscillating cylinder at $Re = 185$ - By increasing the oscillation frequency the core of the top vortex moves towards the cylinder such that the bottom vortex becomes the dominant one.

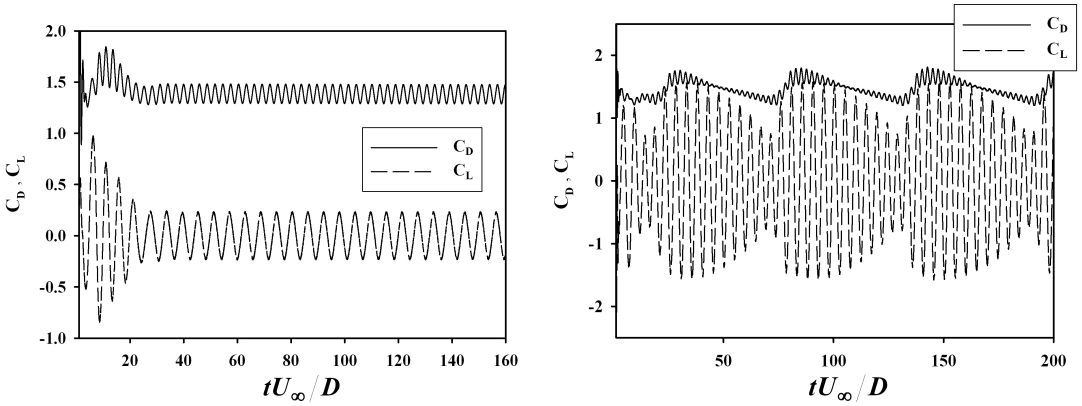


(a) $f/f_0 = 0.9$



(b) $f/f_0 = 1.1$

Figure 3.11: Pressure contours for flow around a transversely oscillating cylinder at $Re = 185$ - No severe pressure oscillation is observed due to the delta function smoothing.



(a) $f/f_0 = 0.9$ **(b)** $f/f_0 = 1.1$

Figure 3.12: Time history of the drag and lift coefficients for a transversely oscillating cylinder at $Re = 185$ - At $f/f_0 = 0.9$ a uniform sinusoidal graph is observed whereas at $f/f_0 = 1.1$ a higher harmonic is also available.

3.3.2 Discrete forcing approach

There are three methods that can be classified under this category namely discrete direct forcing (DDF) [113–116], ghost cell method [117–122] and cut–cell method [123–127]. However since the implementation of cut–cell methods is overly complicated for application to moving geometries [122], only DDF and the ghost cell methods are discussed. Other notable implementations of the cut–cell method are Almgren et al. [128], Johansen and Colella [129], Popinet [130] and Kirkpatrick et al. [123] where the method is applied to 3D simulations around stationary objects. It should also be noted that Kim et al. [131] and Kim and Choi [132] developed a DDF method similar to Mohd-Yosuf [113] but they also added source–sink terms to the pressure equation to get better stability for higher Reynolds numbers.

3.3.2.1 Discrete direct forcing

To simplify the discussion at this stage the fact that the forcing points generally do not coincide with grid points is neglected. The interpolation and spreading processes suggested by Uhlmann [104] will then be discussed which is designed for moving particles using the fractional step NS solver of Le and Moin [35], Section 2.1.2. An algorithm will also be suggested for SIMPLE NS solvers.

An alternate forcing can be found by discretizing Eq. (2.6), the discretized form for an incompressible flow can be written as

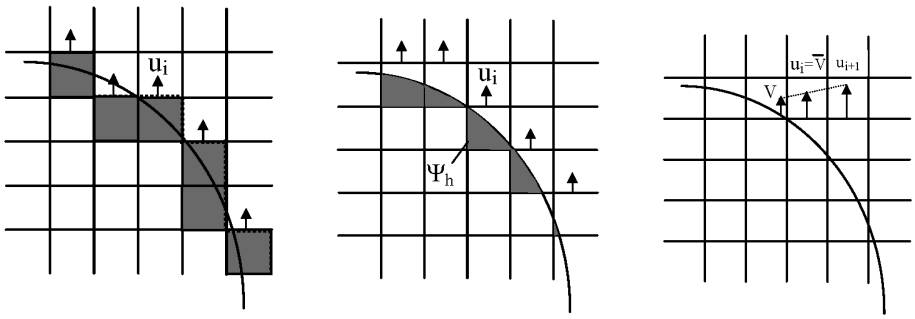
$$\frac{u_{f,i}^{n+1} - u_{f,i}^{n}}{\Delta t} = \text{rhs}_{f,i}^{n+\frac{1}{2}} + f_i^{n+\frac{1}{2}}, \tag{3.36}$$

where $\text{rhs}_{f,i}$ (lower case is used to indicate the value is calculated at grid points) includes all contributions from pressure, viscous and convection terms and $n + \frac{1}{2}$ is the variable evaluated using an intermediate prediction for the velocity field. Now the value of $f_i^{n+\frac{1}{2}}$ that results in $u_{f,i}^{n+1} = U_{m,i}^{n+1}$ is simply

$$f_i^{n+\frac{1}{2}} = -\text{rhs}_{f,i}^{n+\frac{1}{2}} + \frac{U_{m,i}^{n+1} - u_{f,i}^{n}}{\Delta t}. \tag{3.37}$$

This forcing is direct in a sense that the value is calculated directly from the available solution, therefore the boundary condition holds regardless of the flow frequency. In addition there are no flow dependent parameters resulting in a general algorithm with minimum user input. The only issue is that the forcing points (Lagrangian points defining the boundary) and grid point do not coincide. Fadlun et al. [114] suggested volume averaging, linear interpolation of velocities and also moving the forcing term to the nearest cell as potential methods to get the force value at the grid points. The easiest method is to assume that the boundary has a stair step geometry. The IB node is then moved to the nearest grid cell as if they were coincident, see Figure 3.13a. In the volume averaging method, Figure 3.13b, one should calculate the forces based on the grid velocities but impose them on grid points with a volume weighting (using the ratio of the particle volume to cell volume). In the velocity extrapolation method, Figure 3.13c, the boundary velocity $U_{m,i}$ is extrapolated to the nearest grid point using a linear approximation, see Balaras [96] for a second order method, then this point with the extrapolated velocity $U'_{m,i}$ is used for the force calculation step and the forcing term is imposed on the same cell. Note that there is no force spreading or smoothing in any of the methods suggested by Fadlun et al. [114]. Uhlmann [104] found that these methods result in strong oscillations of the hydrodynamical forces due to insufficient smoothing in the case of arbitrarily moving objects. He suggests writing the force on Lagrangian points as

$$F_{m,i}^{n+\frac{1}{2}} = \frac{U_{m_r,i}^{n} - U_{m,i}^{*}}{\Delta t}, \tag{3.38}$$



(a) Stair Step **(b)** Volume Average **(c)** Linear Interpolation

Figure 3.13: Three direct forcing methods - Methods suggested by Fadlun et al. [114] to calculate the force at Eulerian grid points.

where $U^*_{m,i}$ is defined by

$$U^*_{m,i} = U^n_{m,i} + \mathrm{RHS}^{n+\frac{1}{2}}_{m,i}\Delta t. \tag{3.39}$$

Eq. (3.39) is the Lagrangian counterpart of the Eulerian velocity calculated without imposing the forcing term, i.e

$$u^*_{f,i} = u^n + \mathrm{rhs}^{n+\frac{1}{2}}_{f,i}\Delta t. \tag{3.40}$$

Transfer between Eulerian and Lagrangian velocities and forcing terms are implemented using conventional delta function discussed in Section 3.3.1. It is worth mentioning that Uhlmann [104] used the fractional step method for his simulations. The predicted velocity in this method can simply be chosen as the velocity at the first sub time step whereas choosing a prediction for velocity such that the computation time is not increased dramatically is not straightforward for pressure correction based solvers. There are much fewer studies using pressure correction methods, Shen and Chan [133] and Shen et al. [97] claim to use SIMPLEC [7, 29] method for pressure-velocity coupling. However the suggested solution procedure is similar to the operator splitting method discussed in Section 2.1.2. This fact has no effect on their suggested method since the force is calculated explicitly at beginning of each time step using the velocities from the previous step. In addition only a linear interpolation similar to the method suggested by Fadlun et al. [114] is used for the incongruent points. First approximations is very rough and is at best only first order accurate. Extremely small time steps are

required to accurately enforce the boundary condition on the immersed object. The second approximation as discussed earlier causes large oscillations in the calculation of hydrodynamic forces and other unphysical flows. Therefore the following algorithm is proposed for a single iteration of the SIMPLE algorithm.

- Solve the discretized momentum equations (2.9a)–(2.9c),
- Solve the pressure correction equation (2.14),
- Correct pressure (Eq. (2.11)) and velocity components (Eq. (2.10)),
- Similar to Eq. (3.38) calculate a forcing term $F^{IB}_{m,i}$, using the corrected velocities at the current iteration u^*_i projected to the Lagrangian points using Eq. (3.20).
- Calculate a correction to the Eulerian body force $f'_{IB,i}$ by projecting $F^{IB}_{m,i}$ onto the Eulerian grid using Eq. 3.19.
- Correct the body force using

$$f^*_{IB,i} \leftarrow f^*_{IB,i} + \alpha_{f_{IB}} f'_{IB,i}, \tag{3.41}$$

  which will be added to the momentum equations (2.9a)–(2.9c) as a source term for the next iteration. In Eq. (3.41), $\alpha_{f_{IB}}$ is an under–relaxation factor which is best set to the under–relaxation factor of the momentum equation.

Using this approach, the boundary conditions are satisfied up to the pre–specified tolerance upon the convergence. Also note that using smooth transition between Lagrangion and Eulerian variables eliminates any unphysical oscillations. FD methods will be discussed in Section 3.4 and a new algorithm for the SIMPLE solver will be provided. Then the current discrete direct forcing approach will be rigorously compared against the proposed FD method to identify the properties of each. The method with more desirable properties will then be considered for the extension to heat transfer phenomena.

3.3.2.2 Ghost cell method

In this method the boundary condition on the IB is imposed by using 'Ghost Cells'. Ghost cells are defined as computational cells inside the solid domain Ω_p, with at least one neighbour inside the fluid domain Ω_f. The method was first implemented by Majumdar et al. [117] and later used by Ghias et al. [121, 134] and Tseng and Ferziger [120]. Recently Mittal et al. [122] developed a rigorous version of the method specifically applicable to 3D flows using collocated grids arrangement and a fractional step solver for the NS equations. The method is general and implemented for fast solution to 3D flow simulations in complex geometries. A discussion on the main properties of ghost cell methods based on the formulation by Mittal et al. [122] follows next. Differences between various methods in this subclass will be clarified.

First step in successful application of the ghost cell method is to flag the computational cells as either ghost, fluid or solid cells, see Figure 3.14. A general algorithm for finding cells inside the solid object is to find the closest surface element to the cell and forming the dot product of the vector $\mathbf{r}$ and the element normal vector, i.e $r_i n_i$. A negative value implies an internal node whereas a positive value indicates an external cell [118, 135]. The procedure may take a long time but for stationary objects this is a one off calculation and for the moving objects the previous flagging can be used to calculates the new domain flagging assuming the object moves only a small distance during each time step thus taking only a small fraction of the CPU time during each time step.

The general idea is now to modify the discrete equations for the ghost cells such that the boundary condition on the IB surface is exactly enforced. The consensus is to use the normal intercept of the boundary to find a point on the boundary and use fluid cells around it and the point on the boundary to construct an interpolation scheme. A simple bilinear interpolation can be written as [83]

$$\phi = C_1 x_1 x_2 + C_2 x_1 + C_3 x_2 + C_4. \tag{3.42}$$

In Eq. (3.42), ϕ is a general property and the four constants $C_1 \cdots C_4$ should be specified using the normal point on the boundary and the three fluid points around it (for a 2D case). Majumdar et al. [117] suggests using and interpolation scheme that is linear in tangential direction and quadratic in normal direction:

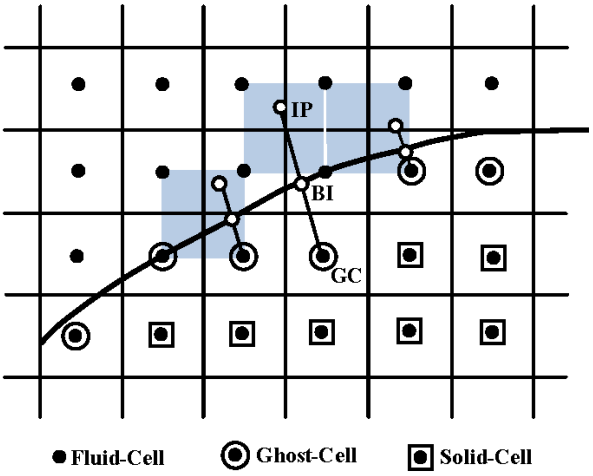


Figure 3.14: Ghost Cell method - Ghost, fluid and solid cells presented. Boundary intersect (BI) point and imaginary point (IP) are also presented for a sample ghost cell [122].

$$\phi = C_1 n^2 + C_2 nt + C_3 n + C_4 t + C_5, \tag{3.43}$$

where n and t are normal and tangential local coordinates to the IB. The coefficients are specified by the three fluid points surrounding the IB, the IB point and another fluid point which is dependent on the surface normal. These methods are not applicable to a general surface due to the fact that it can not be guaranteed that the IB point be surrounded by three fluid point (some of the surrounding point might be ghost cell themselves), thus Mittal et al. [122] suggest the following method

- First step is finding a normal to the boundary from each ghost cell. The algorithm is not trivial as there might be several normals from a ghost cell to the IB surface or even no normal, in case of several normals they suggest using the shortest one and in the case of no normal they suggest using the line connecting the ghost cell to the nearest point on the boundary. However Mark and van Wachem [135] simply used the nearest point on the facet as the surface normal.

- The line is then stretched to the fluid domain such that the intersection of the normal and the IB surface (usually called boundary intersect or BI) is in the mid way between IP (imaginary point inside the fluid domain) and the GC.

- An interpolation function similar to Eq. (3.24) is used assuming that the IP point is completely surrounded by fluid cells. Then by inserting the values of ϕ and x_i, the values of C_n can be found by forming the Vandermonde matrix, Eq. (3.26) and inverting it. Calculating the C_n values we have

$$\phi_{IP} = \sum_{n=1}^{8} \beta_n \phi_n, \tag{3.44}$$

  where β_n are dependent on C_n and the coordinates of IP. Another possible trilinear interpolation procedure is developed by Mark and van Wachem [135] which is equivalent to discrete delta interpolation using a hat function as the kernel. They used this interpolation to write the equations implicitly using the SIMPLEC [7, 29] algorithm for pressure–velocity coupling.

- Next the IP values can be written along the normal line using $\phi_{BI} = \frac{1}{2}(\phi_{GC} + \phi_{IP})$ which after rearranging and using Eq. (3.44) results in the following equation for the ghost cells

$$\phi_{GC} + \sum_{n=1}^{8} \beta_n \phi_n = 2\phi_{BI}. \tag{3.45}$$

  It should be noted that for the pressure a Neumann boundary condition is required, thus the form of Eq. (3.45) is different for the pressure [122].

- Finally Eq. (3.45) is solved for ghost cells alongside the usual discretized NS equations for the fluid cell and the trivial equation $\phi = 0$ for the internal nodes. This completes the treatment of the IB at each time step.

There is an issue with the moving boundaries which is usually referred to as the 'fresh cell' problem. This happens when the boundary moves and new cell with zero ϕ enter the computational domain. Mittal et al. [122] adopted the method of Ye at al. [136] which is similar to the interpolation they carried out for IP using a trilinear interpolant. Mark and van Wachem [135] developed a ghost cell method treating the forcing term implicitly in each time step but the interpolation molecule introduces new diagonals in the coefficient matrix which requires modification to linear solvers and is not obvious how they treat these extra diagonals. They also accounted for the fact that the reverse

velocity field inside the object may cause mass flux over the immersed boundary which they resolved by excluding this fictitious field from the continuity equation.

Trilinear interpolations, Eq. (3.24), are computationally expensive (require solving a system of linear equations for each Lagrangian point) and are needed two times per time step per Lagrangian grid point. Calculation of surfaces forces is also required for free moving objects at each time step further increasing the running time of the algorithm. Another interpolation is also required for the fresh cell problem. In addition, adapting the Neumann condition for the pressure correction equation is not trivial if pressure correction type NS solvers are to be used. Therefore although the method has successfully been applied to many problems (usually used for single large objects or to modify the domain geometry to create more complex domains), it is not very well suited for particulate flow simulations with large number of particles in its current form.

3.4 Explicit DLM/FD and non-DLM/FD methods

FD methods consist of a large class of solution techniques for partial differential equation where a geometrically complex and possibly time dependent domain is embedded in a larger but simpler domain (Fictitious Domain) with the boundary conditions of the original boundary enforce in the new domain. With this definition all the IB methods can also be classified under the fictitious domain methods and be called IB/FD methods but the methods are classified with their common names to avoid ambiguities. One should bear in mind however, that both classes are just different versions of a general FD ideas with different treatments of the original boundaries.

Takiguchi et al. [137] and later Kajishima and Takiguchi [138] used a very simple FD scheme with a finite-difference discretization. They simply set the velocity in the particle domain to the rigid velocity of the particle and used volume averaging near the boundaries. They provide discussions on the grid resolution or number of grid points along the diameter by choosing three different resolutions $D/h = 5$, 8 and 11 [137]. Kajishima and Takiguchi [138] investigated the influence of terminal Reynolds number on the sedimentation of 1024 spheres for terminal Reynolds numbers ranging from 50 to 400. Later the method is used to investigate the two-way interaction between particles and fluid turbulence and the particle clustering effects in homogeneous turbulence [139].

They added previously neglected particle rotation and observed different clustering patterns.

Glowinski et al. [140, 141, 142, 143] were first to introduce the ideas of distributed Lagrange multipliers into the NS equation to enforce the boundary conditions on the surface of the rigid objects as a constraint under a finite-element (FE) framework. This was done by enforcing the rigidity constraint

$$\frac{1}{2}\left(\frac{\partial u_i^R}{\partial x_j}+\frac{\partial u_j^R}{\partial x_i}\right)=0,\ \forall x\in\Omega_p, \tag{3.46}$$

on the particle domain using distributed Lagrange multipliers. This constraint is the direct consequence of the fact that the particle undergoes a rigid-body motion and the velocity of a point in the particle domain is given by

$$u_i^R=U_{p,i}+\epsilon_{ijk}\omega_{p,j}r_k. \tag{3.47}$$

In Eq. (3.47), r_k is the position vector from the centre of the particle to the location of the point under consideration and ω_p is the angular velocity of the particle. The derivations of the variational formulae are lengthy and outside the scope of this thesis and the reader can consult [140–143]. This method although is successfully applied to particulate flows with 1008 2D circular cylinders and 128 3D spheres [144] but proves to be inefficient due to the explicit calculation of the additional DLM.

The method was improved by avoiding the explicit calculation of the Lagrange multiplier [145, 146] and introducing body forces to enforce the rigidity constraint which they named non-DLM/FD method. The same equations are derived by Yu and Shao [147] and Diaz-Goano et al. [148] in a slightly different manner but the formulations are essentially based on the same ideas and final results are the same except for a few details in writing the final form of the body force. Later Sharma and Patankar [149], Veeramani et al. [150] and Apte et al. [151] used the same procedure. Sharma and Patankar [149] used the Finite Volume (FV) and SIMPLER [29] algorithm for pressure–velocity coupling whereas Apte et al. [151] used a standard fractional step FV method and Veeramani et al. [150] implemented the method under an FE framework.

Assuming the domain consists of two fluids with densities ρ_f and ρ_p the momentum and continuity equations on the whole domain Ω_d can be written as

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu_f \left(\frac{\partial u_i}{\partial x_j}\right)\right] + f_{B,i} + f_i, \tag{3.48}$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0. \tag{3.49}$$

In Eq. (3.48), $f_{B,i}$ is the buoyancy forces which is only relevant when freely moving objects or heat effects on the density field are considered and will be discussed in Section 4.2. Here the subscript f in NS equations is dropped since the whole domain is considered as a fluid with variable properties. ρ is the mixture density and can be calculated by

$$\rho = \Theta_p \rho_p + (1 - \Theta_p)\rho_f. \tag{3.50}$$

The step function Θ_p moves with the particle and hence $\frac{D\Theta}{Dt} = 0$, where $\frac{D}{Dt}$ is the material derivative, and it is easy to show that the continuity equation reduces to $\frac{\partial u_i}{\partial x_i} = 0$ for an incompressible flow. However for a moving particle it is still necessary to solve the full continuity equation to account for the density changes in the CVs containing both fluid and particle phases. The body force f_i that enforces the rigidity constraint inside the particle domain can be written as:

$$\rho_p \frac{u_i^R - u_i^l}{\Delta t} = f_i, \tag{3.51}$$

where u_i^R is the rigid body velocity and u_i is the velocity at the current iteration calculated from Eq. (3.48). Eq. (3.51) implies that the motion $u_i + f_i \Delta t/\rho_p$ is a rigid-body motion and should satisfy the rigidity constraint (Eq. (3.46)) in the particle domain. However this equation gives no information about u_i^R and it is actually the velocity field of interest. In this chapter only predefined motion is considered to compare the general behavior of this method to the proposed DDF method. In case of stationary objects or forced motion u_i^R is simply equal to either zero or the predefined velocity. Extension to heat transfer will be discussed in Chapter 4 and free motion in Chapter 5 where the full solution process is also provided.

3.4.1 Numerical Implementation

In this section the details of the basic numerical scheme is discussed where heat transfer and free motion are neglected. First the geometric presentation of the FD phase and then the calculation of rigidity constraint for the FD phase will be considered. A force

correction scheme is proposed which conforms to the SIMPLE algorithm to calculate the rigidity constraint.

3.4.1.1 Geometric presentation

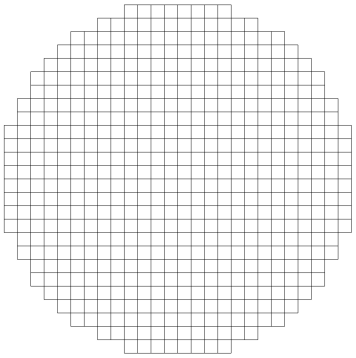
In the FD method the solid body domain is considered as a real fluid domain constrained to a rigid motion and therefore a discretization of the body is needed. Two different methods can be employed, first is to use the background mesh and a geometric method to calculate the volume fractions on the grid points that contain a part of the bluff body [149]. The other method is to define a separate Lagrangian mesh, see Figure 3.15, to represent the body and use discrete delta functions to transfer the variables between the two meshes [147, 151]. There are several advantages in using an explicit grid to present the object; firstly the properties will properly be spread onto the background mesh with sufficient smoothing using discrete delta functions, whereas in the geometric method no force smoothing is applied which incurs unphysical oscillations similar to those explained in Section 3.3.2.1. Secondly the geometrical calculation of volume fractions is time consuming. For example in the method suggested by Sharma and Patankar [149] each control volume containing a part of the object boundary is divided into at least $O(10^2)$ smaller control volumes and each smaller control volume is checked to find out whether is falls inside or outside the object. Therefore in this study delta functions are used to calculate the volume fractions.

Having a discretization of the body discrete delta functions are used to transfer the variables between the two meshes which guarantee the conservation of the projected variable, Eq. (3.13), such that the total volume of the immersed object is equal on both the Eulerian and material mesh. Having defined the delta functions, Section 3.3.1, the volume fractions can be calculated by

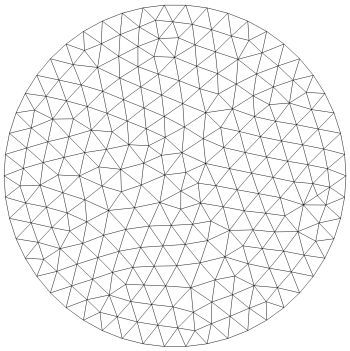
$$\Theta_p(\mathbf{x}) = \sum_{m=1}^{N} \delta_h \left(\frac{\mathbf{x} - \mathbf{X}_m}{h} \right) \upsilon_m \quad \forall \mathbf{x} \in g_h, \tag{3.52}$$

where N, υ_m are the number of material control volumes and the volume of the m-th material CV respectively. Using this volume fraction a mixture density can be calculated using Eq. (3.50).

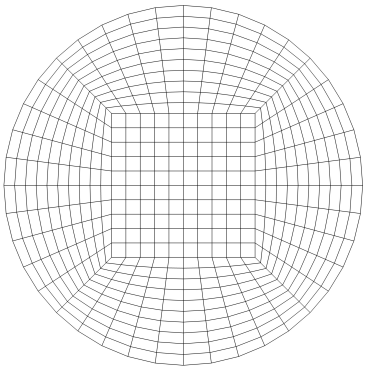
The generated mesh for the body should be fine enough such that the sum of the volume of the discretized grid be a precise approximation to the real volume of the object



(a) Stair-Step Grid



(b) Unstructured Triangular Grid



(c) Structured Quadrilateral Grid

Figure 3.15: Different types of grid used for the discretization of the bluff body - The depicted grids are much coarser than the actual grids used in the simulations for better presentation.

i.e $\sum_{m=1}^{N} \upsilon_m - V_p \leq \epsilon$. Another important property of this grid is to have a uniform element size. Uniform element size is crucial to uniformly transfer the volume fractions and other Lagrangian quantities to the background Eulerian mesh. For example for transferring the volume fractions any Eulerian grid point inside the particle domain should ideally have a volume fraction of one. Therefore the following quality measure is introduced

$$q = \sqrt{\frac{1}{M}\sum_{j=1}^{M}(1 - \Theta_p(\mathbf{x}_j))^2}, \tag{3.53}$$

where M is the number of Eulerian grid points falling inside the particle domain. It is also desirable to be able to generate this grid automatically without any user inputs.

Figure 3.15 shows three possibilities to generate such a grid. In Table 3.3 the accuracy of different types of tested grids is compared. The triangular and quadrilateral grids are generated such that the number of elements are the same as those in stair step grid with $h/h_m = 1$, where h_m is the material grid spacing. Evidently a quadrilateral grid, Figure 3.15c, does not produce a high quality q–measure and therefore may produce unphysical forces especially in freely moving tests due to the consequent non-uniformity in the density field. In addition generation of this grid is not automatic and requires user input and hence this method will not be considered further. Stair–step grids, Figure 3.15a, produce very high q–measure but errors in calculating the total volume fractions can be up to five times higher than those calculated by a proper grid with the same element size. A spacing ratio of at least three is required to get total volume estimate comparable to those produced by a proper grid. A triangular grid, Figure 3.15b, produces very accurate estimates of the total volume and good q–measure similar to those achievable by stair step type grids with third the number of grid points. This reduction in the number of grid point can significantly increase the computational efficiency when the dimensions of the bluff body is large compared to the computational domain or large number of small particles are being simulated. In addition triangular grids can automatically be generated without any user inputs.

3.4.1.2 Calculation of the rigidity constraint

The discretized momentum equations (2.9) for the guessed field in Ω_p can compactly be written as

Grid Type	V_p $(\times 10^3)$	$\sum_{m=1}^{N} v_m$ $(\times 10^3)$	$\%Error$	q
Triangular	7.85398	7.85267	0.017	0.97
Quadrilateral	7.85398	7.85174	0.029	0.88
Stair Step $h/h_m = 1$	7.85398	7.84722	0.086	0.99
Stair Step $h/h_m = 3$	7.85398	7.85606	0.027	0.99

Table 3.3: Comparison of the accuracy of different tested mesh types.

$$a_{P,u_i} u^*_{P,i} = \sum a_{nb,u_i} u^*_{nb,i} - c_{u_i} \delta P^* + S_{u_i} \mathcal{V} + f^*_{FD,i}, \tag{3.54}$$

where $f^*_{FD,i}$ is the guessed Eulerian force field calculated using the velocity field from the last iteration or the values from the previous time step. Note that this equation is the discretized form of the general momentum equation (Eq. (3.48)) rather than Eq. (2.6) and in this sense is different from Eq. (2.9). Similar to the discussions in Section 2.1.1, after solving Eq. (3.54) the velocity field satisfies the momentum equation but not necessarily the rigidity constraint. It can be assumed that $f^*_{FD,i} + f'_{FD,i}$ is the required force to enforce the rigidity constraint in the particle domain such that

$$a_{P,u_i} u^R_{P,i} = \sum a_{nb,u_i} u^R_{nb,i} - c_{u_i} \delta P^* + S_{u_i} \mathcal{V} + f^*_{FD,i} + f'_{FD,i}. \tag{3.55}$$

Subtracting Eq. (3.55) from Eq. (3.54) and similar to the simplifying assumption made in the SIMPLE algorithm, neglecting the term $\sum a_{nb,u_i}(u^*_{nb,i} - u^R_{nb,i})$ we have

$$f'_{FD,i} = a_{P,u_i}(u^*_{P,i} - u^R_{P,i}). \tag{3.56}$$

Therefore the correction field is proportional to the velocity difference. In an unsteady simulation $\frac{\rho_p}{\Delta t}$ can be chosen as the proportionality constant which is the unsteady contribution to the coefficient a_{P,u_i} in the particle domain. One may use a_{P,u_i} as the proportionality constant, however this causes very large corrections and reduces the stability. With the proposed proportionality constant an under–relaxation factor equal to the one used for momentum equation results in a smooth convergence without increasing the number of iterations per time step. Ardekani et al. [152] used a similar approach, however they used an ad-hoc parameter and a problem dependent velocity scale as the proportionality constant. With this suggestion no extra parameters are required and the number of iterations per time step does not increase. Ardekani et al.

[152] also used the background mesh to present the object instead of an explicit Lagrangian mesh to present the body. Advantages of the current approach was discussed in Section 3.4.1.1. The required force can then be written by

$$f'_{FD,i}(\mathbf{x}) = \rho_p \frac{u_i^*(\mathbf{x}) - u_i^R(\mathbf{x})}{\Delta t} \quad \forall \mathbf{x} \in \Omega_p. \tag{3.57}$$

Basic FD solution process can be summarized as follows

- Solve the discretized momentum equations (3.54),
- Solve the pressure correction equation (2.14), again note that this is the discretized form of Eq. (3.49),
- Correct pressure (Eq. (2.11)) and velocity components (Eq. (2.10)),
- Project the corrected velocities at the current iteration u_i^* to the Lagrangian points using Eq. (3.20), to calculate Lagrangian velocities $U_{m,i}^*$.
- Calculate the body forcing term

$$F_{m,i}^{FD} = \rho_p \frac{U_{m,i}^R - U_{m,i}^*}{\Delta t}, \tag{3.58}$$

  where for stationary particles, $U_{m,i}^R$ is simply zero. In case of forced motion it can be calculated using the discretized form of Eq. (3.47)

$$U_{m,i}^R = U_{p,i} + \epsilon_{ijk}\omega_{p,j}(X_{m,k} - X_{p,k}). \tag{3.59}$$

- Calculate a correction to the Eulerian body force $f'_{FD,i}$ by projecting $F_{m,i}^{FD}$ onto the Eulerian grid using Eq. (3.19).
- Correct the body force using

$$f_{FD,i}^* \leftarrow f_{FD,i}^* + \alpha_{f_{FD}} f'_{FD,i}, \tag{3.60}$$

  which will be added to the momentum equation (3.54) as a source term for the next iteration. In Eq. (3.41), $\alpha_{f_{FD}}$ is an under–relaxation factor set equal to that of the momentum equation.

With this approach the velocity in the particle domain will exactly be the rigid velocity up to the predetermined tolerance and consequently the rigidity constraint is precisely satisfied.

3.5 Results and discussions

In this section the DDF method presented in Section 3.3.2.1 and the FD method discussed in Section 3.4 are tested and rigorously compared. Flow around a 2D circular cylinder is considered as the test case and different features of both methods are compared at different Reynolds numbers and flow configurations. Triangular and stair–step meshes are considered and their advantages and disadvantages are pointed out.

3.5.1 Order of the proposed DDF IB and FD methods

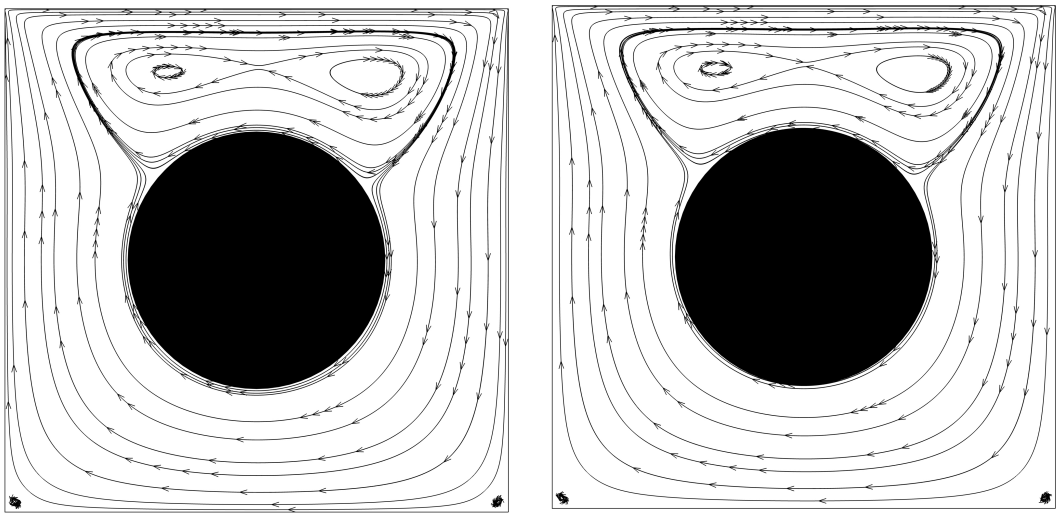
To test the spatial accuracy of the method the lid-driven cavity problem is considered with a stationary rigid cylinder in its centre. This problem is similar to the 2D problem used by Kirkpatrick et al. [123] to test the accuracy of their cut-cell method and later a 3D version of the problem was used by Gilmanov et al. [118] to test their immersed boundary method. The height of the cavity is H and the cylinder diameter is set to $H/2$ where no-slip boundary condition is implemented on all the cavity walls. Under-relaxation factor for the pressure correction is set to 0.2 and for the velocities to 0.8. Since this problem is steady only 10 iterations per time step is performed and the results at the final time step are used as suggested by Ferziger and Peric [7]. For the FD method a triangular mesh with $N \approx 7\times10^4$ is used which substantially reduces the time required for velocity interpolation and force spreading steps compared to a stair–step grid. For the DDF IB method the surface of the cylinder is discretized using $N = 10^3$ nodes. The simulation is performed for $Re = 20$ (based on the cavity width and lid velocity) and three mesh levels of 100^2, 200^2 and 400^2. The simulation is run to reach the steady condition and the steady state streamlines are presented in Figure 3.16a and 3.16b for the finest grid levels for the FD and IB methods respectively. The finest grid is considered to be the exact solution and a Richardson-estimation procedure which is used by many authors to estimate the accuracy of numerical schemes [119, 153] is used to calculate the order of the accuracy of the proposed methods. Figure 3.16c shows the accuracy plots of the proposed methods with the first order reference line. The average

accuracy of the both methods in 1–Norm is slightly better (1.1) than the strictly first order method proposed in [149] which can be related to the correction of the rigidity constraint.

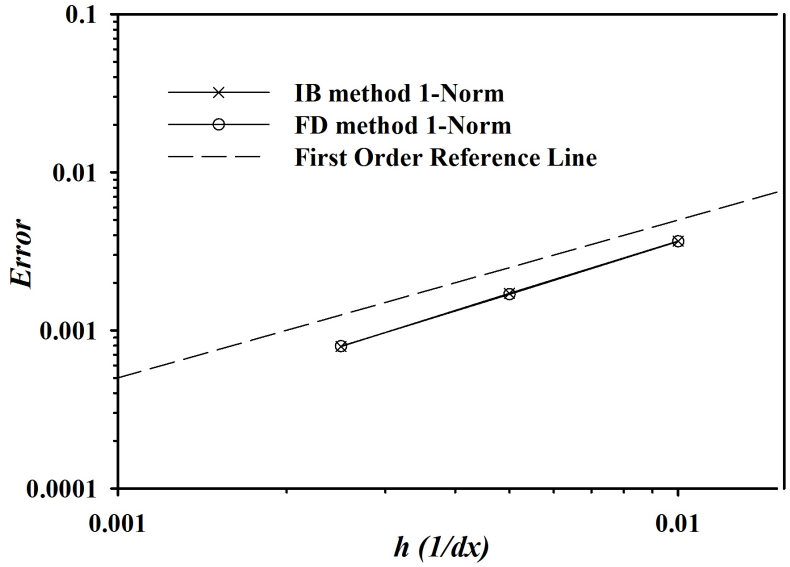
3.5.2 Flow around a circular cylinder

Flow around an stationary cylinder has become an standard problem to asses the fidelity of the NS solvers. The Reynolds number is defined for this problem by $Re = \rho_f U_\infty D/\mu_f$. It is well known that the flow is steady and symmetrical about the wake centreline up to $Re = 47$ and become unstable for higher Reynolds numbers with periodic vortex shedding. The flow remains 2-dimensional up $Re = 180$ after which the flow is intrinsically 3-dimensional [154, 155]. However higher Reynolds numbers are still used to validate the numerical simulations. In this section a $20D \times 25D$ domain is used and the cylinder is placed at $(x_c/D, y_c/D) = (10, 10)$. The domain size is shown in [156] to be large enough to suppress the effects of lateral boundaries on the flow characteristics around the cylinder. The boundary conditions at the top and bottom walls are set to standard slip conditions with zero normal gradient and standard inlet and outlet boundary conditions are used in the flow direction. For the DDF method the surface of the cylinder is defined with 100 Lagrangian points. Both triangular and stair–step meshes are used for the FD method, Figure 3.15. Assuming h_m to be the Lagrangian grid spacing a material to Eulerian grid ratios of three is used ($h_m/h = 3$). This is equivalent to placing nine material CVs into each corresponding Eulerian CV. Also a triangular mesh with $\overline{A}_m/h^2 \approx 1$ is used to show the independence of the results from the material grid type. Defining the object by a proper triangular grid is proposed for the first time in this work, mainly for the computational efficiency. The amount of interpolation/spreading can be reduced by a factor of three since a h_m/h of at least three is required if a stair–step grid is used. Both the inlet velocity U_{in} and the fluid density ρ_f are set to one. Note that the object is stationary and hence the body density ρ_p is irrelevant. Mesh spacing around the circular cylinder is set to $0.01D$ in a box with sides $2D$ and is stretched afterwards as discussed in Section 3.3.1.2, $\Delta t = 0.02$ and velocity and pressure under-relaxation factors are set to 0.8 and 0.2 respectively for all the simulations.

Figure 3.17 and 3.18 show the symmetric wake at the rear of the cylinder. Proposed FD and DDF method generate similar results and the FD method is independent of



(a) Fd Method **(b)** IB Method



(c) Accuracy of the FD method

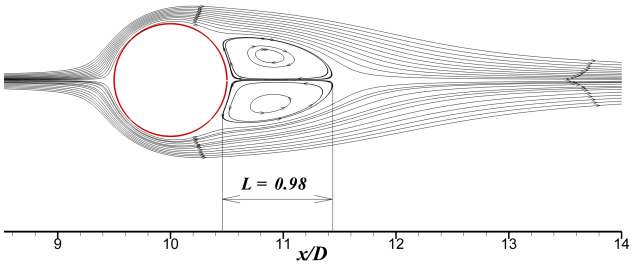
Figure 3.16: Streamlines around a rigid cylinder inside a lid-driven cavity - Steady solution at $t = 100$ for $Re = 20$ is presented and accuracy of the proposed methods is measured using u-velocity and a Richardson estimation method.

the Lagrangian grid used to present the object. Note that both grids shown for the FD method are much coarser than the actual grid used in the simulation for better presentation. The triangular grid used in this simulation consists of $N = 7268$ elements which is almost equal to the number of Eulerian grid points affected by the object, whereas the number of stair–step grid is around $N \approx 20000$. Table 3.4 compares the drag coefficient and the length of the wake bubble L_W with the previous studies which shows that both methods can easily captures the flow characteristics in the steady region. It also further validated the method suggested for the calculation of the surfaces forces discussed in Section 3.3.1.1.

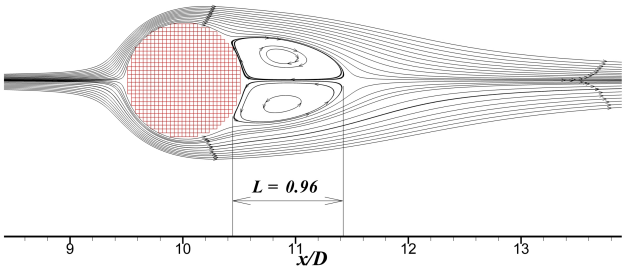
Study	Re			
	20		40	
	C_D	L_W	C_D	L_W
DDF IBM	1.95	0.98	1.47	2.34
FD Stair–step	2.00	0.96	1.50	2.34
FD triangular	1.99	0.96	1.49	2.32
Dennis [157]	2.05	0.94	1.52	2.35
Fornberg [158]	2.00	0.91	1.50	2.24
Ye et al. [136]	2.03	0.92	1.52	2.27

Table 3.4: Bubble length and C_D for flow around a cylinder at $Re = 20,\ 40$

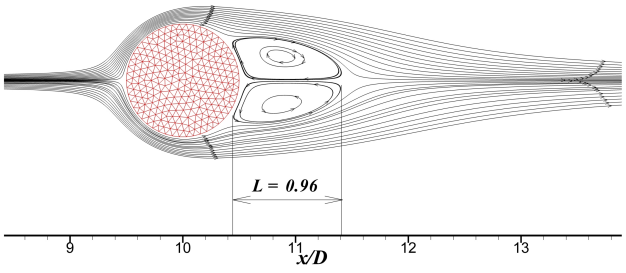
To further compare the general features of the DDF and the FD methods same problem is studied at $Re = 100$. Figure 3.19a shows the vector plot of the velocity field around the cylinder at $Re = 100$. In this figure the vector plot of the whole domain is presented and the object is presented with the actual grid points used in the code. Evidently a flow field develops inside the particle domain which is a direct consequence of momentum balance and the body force inserted at the boundary of the cylinder. This behavior is similar to the VB method discussed in Section 3.3.1.2. Figure 3.19b shows the pressure contour for the same problem at $tU_\infty/D = 200$ clearly no pressure field will develop inside the particle in this method since the no–slip condition is satisfied when solving the momentum equation and the form of the pressure correction equation is approximately (since a sharp definition of the boundary is not available) similar to having the boundary explicitly defined using a body conformal mesh. Figure 3.21a



(a) DDF IBM $Re = 20$

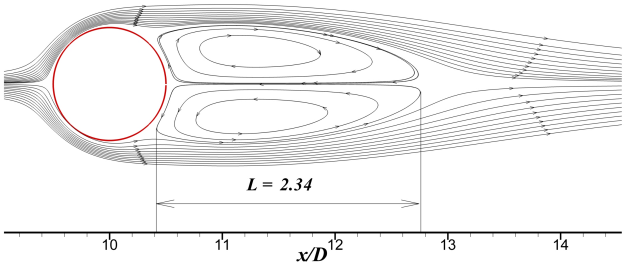


(b) Stair–step FD method $Re = 20$

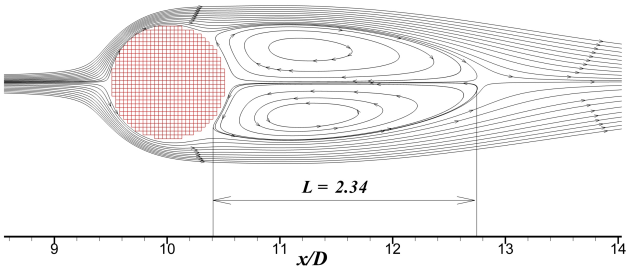


(c) Triangular FD method $Re = 20$

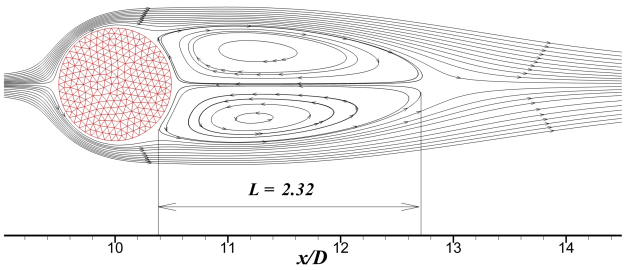
Figure 3.17: Flow around a circular cylinder at $Re = 20$ - Both methods accurately capture the basic flow features around the circular cylinder correctly. Calculations with FD method are independent of the underlying material grid type. Both stair–step and triangular meshes are much coarser than the actual mesh used in the simulations.



(a) DDF IBM $Re = 40$



(b) Stair–step FD method $Re = 40$



(c) Triangular FD method $Re = 40$

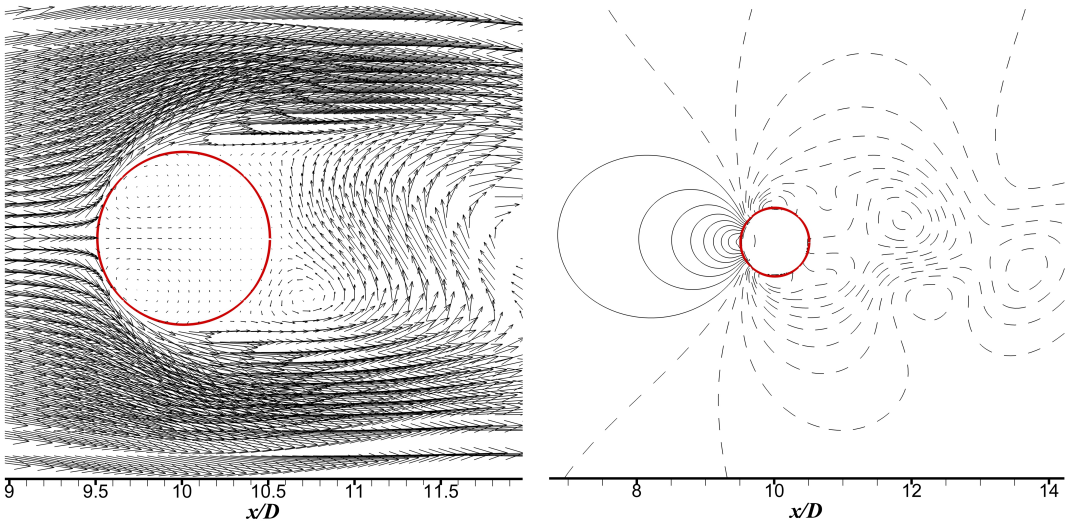
Figure 3.18: Flow around a circular cylinder at $Re = 40$ - See the caption for Fig. 3.17.

shows the evolution of the drag and lift coefficients in time which shows the vortex shedding reaches a steady state at approximately $tU_\infty/D = 80$.

Figure 3.20a shows the vector plots of the velocity field around the cylinder and the particle mesh is produced using a stair–step grid where the ratio of the Lagrangian grid spacing to that of the Eulerian grid is $h_m/h = 3$. Evidently the velocity field inside the domain is equal to the physical rigid-body velocity of the particle as expected which is zero in this case. Viscous stresses are automatically eliminated inside the particle domain due to the implementation of the rigidity constraint (Eq. (3.46)) and hence the viscosity inside the particle domain is irrelevant. Figure 3.20b shows the pressure contours at $tU_\infty/D = 200$. Clearly a pressure field will develop inside the object which is due to the fact that after solving the momentum equation the rigidity constraint inside the particle domain is satisfied and this ensures that the continuity equation (Eq.(3.49)) is also satisfied. However to facilitate the solution process the continuity equation is solved on the whole domain and therefore a pressure field naturally develops inside the particle domain as a result of the solution process. However since the body force which enforces the rigidity constraint is corrected at each time step final body force also counterparts this extra stress inside the object and hence no extra driving force will be exerted on the particle. To better compare the behavior of this method to that of DDF IBM the evolution of the drag and lift coefficients are compared against those calculated by DDF method in Figure 3.21.

Previous numerical comparisons between two proposed method show that both methods are capable of generating basic flow features with high accuracy. It is possible to insert forcing points inside the object in the DDF method to enforce the object velocity in Ω_p. However in the DDF IB method the underlying equations are Eq. (2.6) and Eq. (2.7) whereas in the FD method Eq. (3.48) and Eq. (3.49) are solved. Therefore in the DDF IB method even if the actual velocity is enforced inside the object, integration of the forces on the surface is still required to move the object. Whereas in the FD method this motion is a direct consequence of the Eq. (3.48) and hence the particle motion can be calculated with simple volume integrations which will be discussed in Chapter 5. This is a significant advantage of the FD method over the DDF IB for freely moving objects. Another advantage of this FD method is that since the inside of the solid object is actually considered as a real physical domain with specific physical properties, other transport phenomena can efficiently be implemented. In this method there

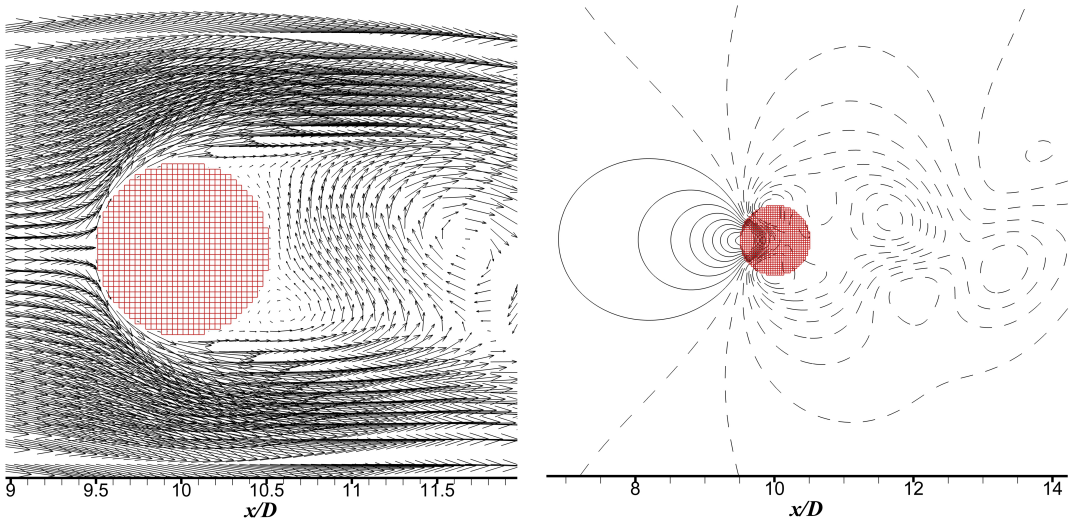
is no need to solve any equations except the required transport equation on the whole domain and conjugate phenomena are directly captured. Application of the method to the heat transfer phenomena will be discussed in Chapters 4 and 5. These advantages make the FD method a better option for the simulation of systems with many objects where other transport phenomena should also be considered.



(a) Vector Plots **(b)** Pressure contours

Figure 3.19: Simulation of flow around a circular cylinder at $Re = 100$ using DDF IBM method - Vector plots are presented and a flow field is developed inside the object using a direct forcing IBM but no pressure field is developed for DDF IBM. 20 contour levels are presented at $tU_\infty/D = 200$.

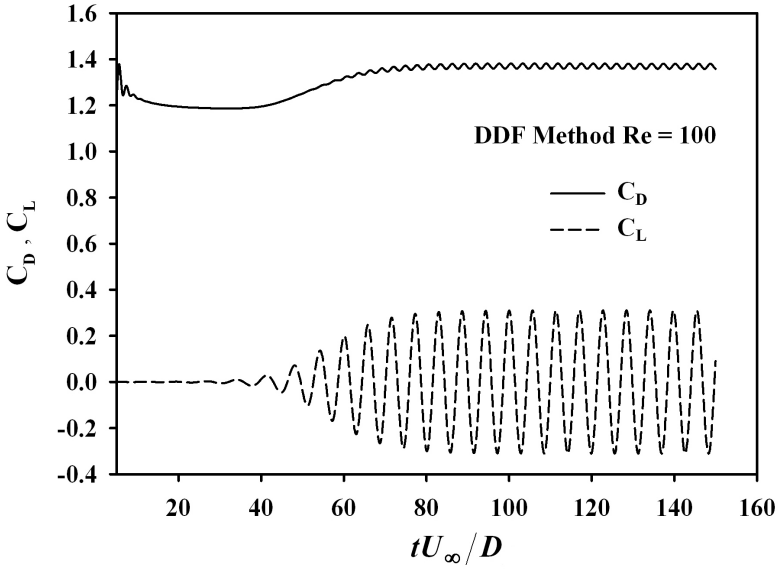
The proposed FD method is further tested by increasing the Reynolds number to $Re = 1000$ using a stair–step grid with $h_m/h = 3$. Figures 3.22a– 3.22c show the vortex structures for $Re = 100$, 300 and 1000 respectively. Evolution of the drag and lift coefficients are plotted in Figure 3.23 which shows after a non-dimensional time $tU_\infty/D = 50$ all test cases reach the stationary state. Vortex shedding frequency f is computed by applying a FFT to the lift coefficient time series and consequently vortex shedding Strouhal number is calculated by $St = fD/U_\infty$ and results are compared with previous numerical simulations [122, 136, 159, 160] in Table 3.5. Average drag coefficients are also computed using the accumulated data after a stationary state is obtained for three Reynolds numbers and the results are compared to previous studies



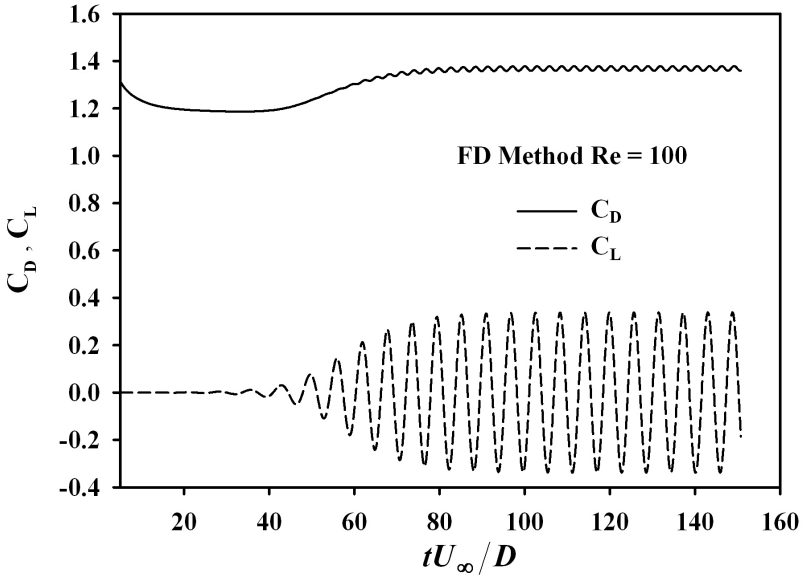
(a) Vector Plots

(b) Pressure contours

Figure 3.20: Simulation of flow around a circular cylinder at $Re = 100$ using FD method - Vector plots and pressure contours around a circular cylinder at $Re = 100$ are presented (20 contour levels at $tU_\infty/D = 200$). A pressure field is developed inside the object using the FD method but the velocity field is the the rigid velocity of the body which is zero in this case.



(a) DDF IBM



(b) FD method

Figure 3.21: Time history of drag and lift coefficients at $Re = 100$ - Evolution of drag and lift coefficients at $Re = 100$ around a stationary circular cylinder. Oscillation in the curves shows the start of the vortex shedding and Strouhal number are calculated using the frequency of the lift coefficient curve.

in Table 3.6. At $Re = 100$ computations are also performed using a triangular grid which shows good agreement with values obtained with the stair-step material grid and independence of the results from the type of the material grid.

Study	Material Grid	St		
		$Re = 100$	$Re = 300$	$Re = 1000$
Current Scheme	Stair-Step	0.166	0.21	0.238
Current Scheme	Triangular	0.163	-	-
Williamson [159]		0.165	0.205	0.238
Zang et al. [160]		0.167	-	-
Mittal et al. [122]		0.165	0.21	0.231
Ye et al. [136]		-	0.21	-

Table 3.5: Strouhal number for flow over a circular cylinder - Comparison of Strouhal number for flow over a circular cylinder for different Reynolds numbers.

Study	Material Grid	$Re = 100$	$Re = 300$	$Re = 1000$
Current Scheme	Stair-Step $h_m/h = 3$	1.31	1.30	1.48
	Triangular $A_m/h^2 \approx 1$	1.30	-	-
Tritton [161]	-	1.30	-	-
Marella et al. [162]	-	1.36	1.28	-
Mittal et al. [122]	-	1.35	1.36	1.45
Apte et al. [151]	-	1.36	1.41	1.51

Table 3.6: Mean drag coefficient for flow over a circular cylinder - Comparison of mean drag coefficient with previous experimental and numerical studies for different Reynolds numbers.

Figure 3.24 compares the $\overline{u}$ and $\overline{v}$ statistics near the circular cylinder to those calculated by Apte et al. [151]. The averaging is run for a total time of $240D/U_\infty$. Initial data for 90 time units are discarded and the statistics are collected over the remaining 150 time units. Very good agreement is observed for the first moments of the velocity components. In Figure 3.25 second central moments of the fluctuating velocity components, $\overline{u'}^2$ and $\overline{v'}^2$, are calculated and compared to those reported in Apte et al. [151] which again shows very good agreement. Finally velocity correlations $\overline{u'v'}$ are calculated near the cylinder and are presented in Figure 3.26. At the first collection point $x/D = 1.2$, the curve slightly deviates from those reported by Apte et al. [151]. The

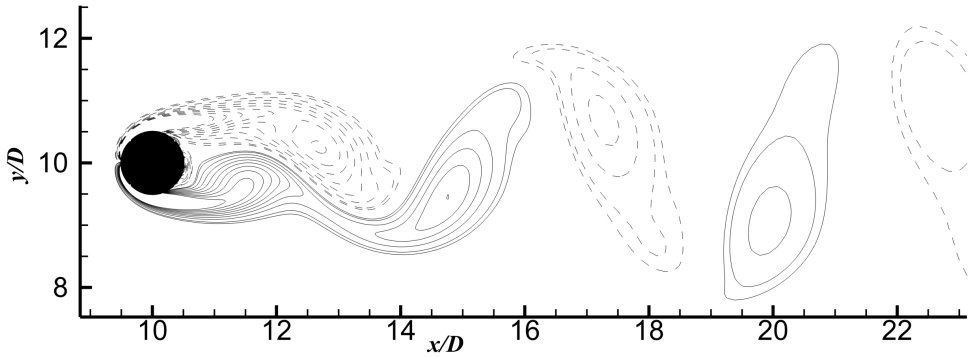
source of this discrepancy can be due to the fact that the value of the correlation very close to the cylinder is sensitive to the location at which it is calculated. Therefore different averaging method used to calculated the velocity at the collection points may produce different results. However it is not reported how these values are calculated in [151]. Note that in this study collection points do not coincide with the grid points and bilinear averaging is used to calculate the velocity component at the collection point. In Figure 3.26 also results of the spectral simulation of Mittal and Balachandar [154] are presented where again it is not reported whether the points are coincident or an averaging is performed. Their results also deviates from those reported by Apte et al. [151] and also from the current results but it seems the current results are closer to those reported by Mittal and Balachandar [154].

3.5.3 In–line oscillation of a circular cylinder

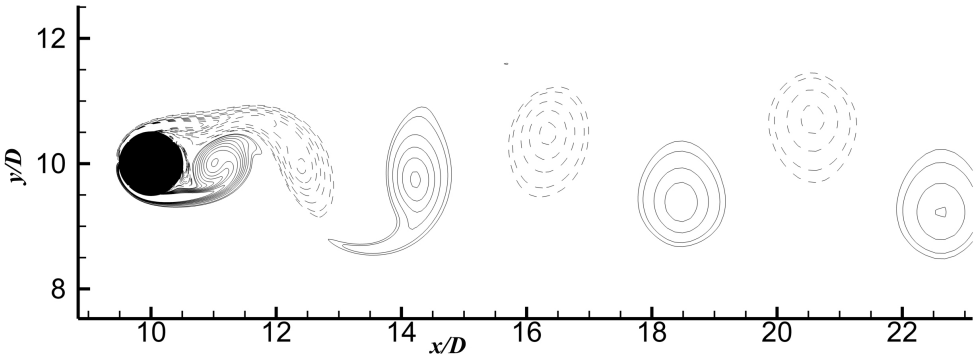
To validate the basic particle motion the flow induced by in-line oscillation of a circular cylinder is considered. The problem is investigated in detail both numerically [132, 151] and experimentally [163]. Such flow is characterized by the Keulegan-Carpenter number $KC = U_m/(fD)$ and the Reynolds number $Re = \rho_f U_m D/\mu_f$ where U_m is the maximum velocity of the cylinder and f is the frequency of the oscillations. The forced motion of the cylinder is governed by

$$x_p(t) = A_p \sin(\omega t), \tag{3.61}$$

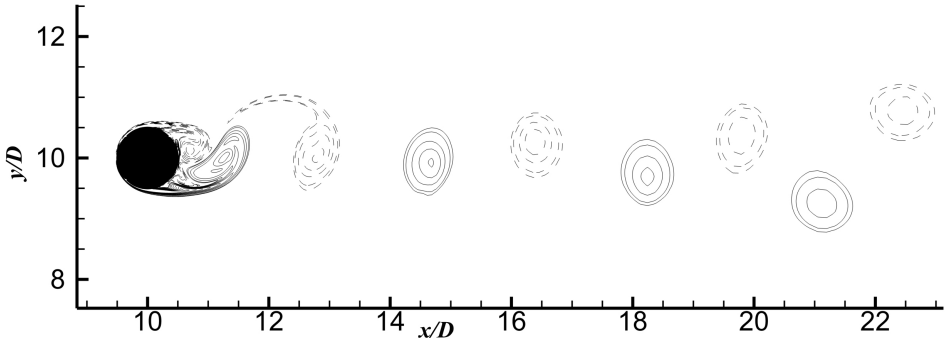
where x_p is the position vector of the centre of the particle, A_p is the amplitude of the oscillations and $\omega = 2\pi f$. Dutch et al. [163] used $KC = 5$ and $Re = 100$ for one of their experiments which is later used to validate other numerical schemes [132, 151]. Accordingly $D = 0.01$, $\mu_f = 10^{-6}$, $\rho_f = 1$, $f = \frac{1}{5}$ and $U_m = 0.01$ are used. Also note that by differentiating Eq. (3.61) maximum velocity can be calculated and Keulegan-Carpenter number becomes $KC = \frac{2\pi A_p}{D}$ which fully determines the system. A fine uniform mesh with $h = 0.025D$ is generated around the object in a square region of $10D \times 10D$ which covers the amplitude of the oscillation and is stretched after that. The circular cylinder is presented by a triangular mesh and the number of material CVs are chosen such that $\overline{A}_m/h^2 \approx 1$, where $\overline{A}$ is the average area of the material CVs and h is the grid spacing in the vicinity of the cylinder. This again substantially reduces



(a) Z-Vorticity contours for $Re = 100$

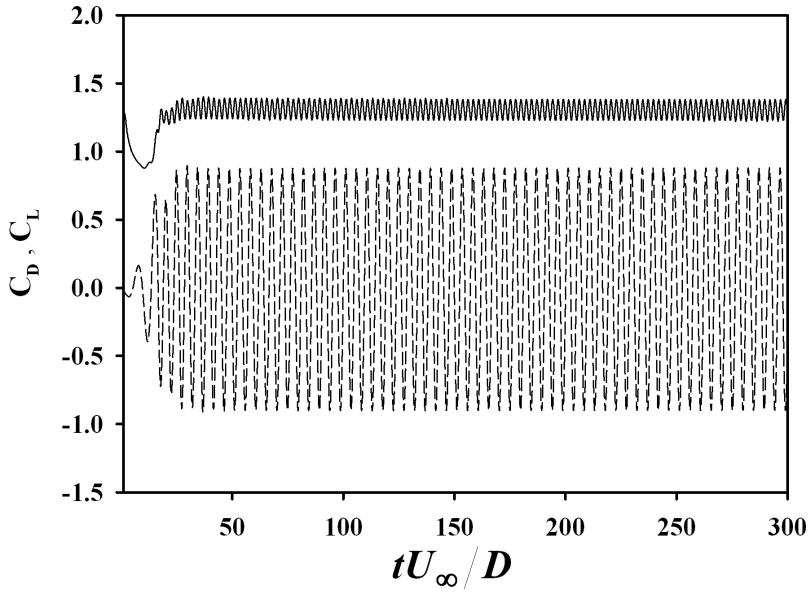


(b) Z-Vorticity contours for $Re = 300$

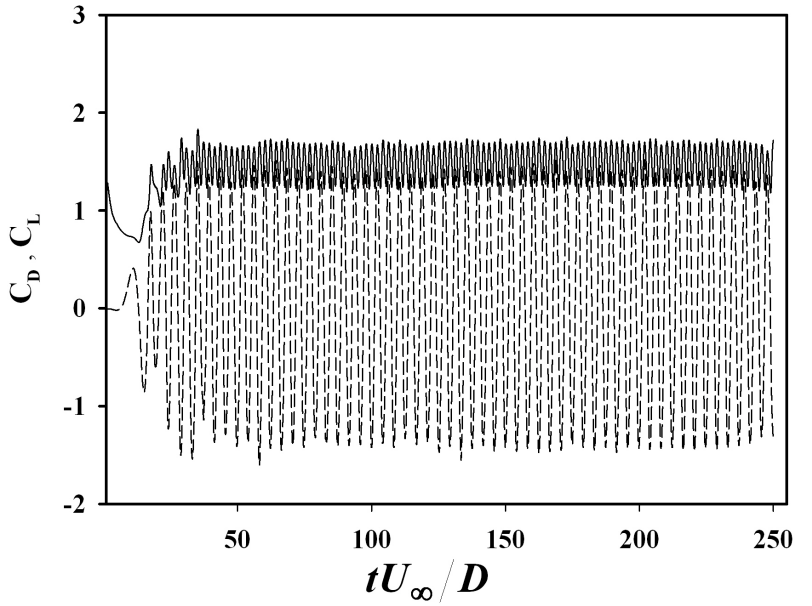


(c) Z-Vorticity contours for $Re = 1000$

Figure 3.22: Simulation of flow around circular cylinder at three different Reynolds numbers. - Periodic vortex shedding for $Re = 100$, 300 and 1000 is correctly simulated. Dashed lines represent negative values and 20 levels of contours are presented between -30 and 30 for all cases.

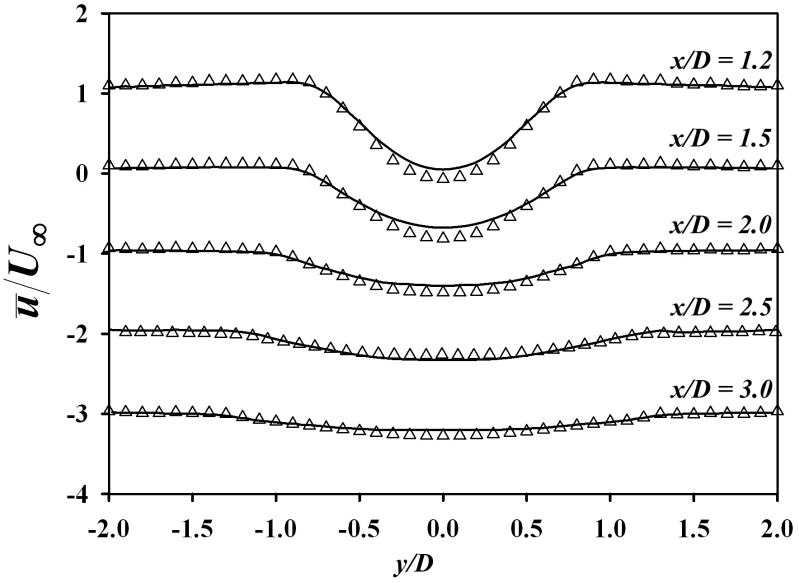


(a) $Re = 300$

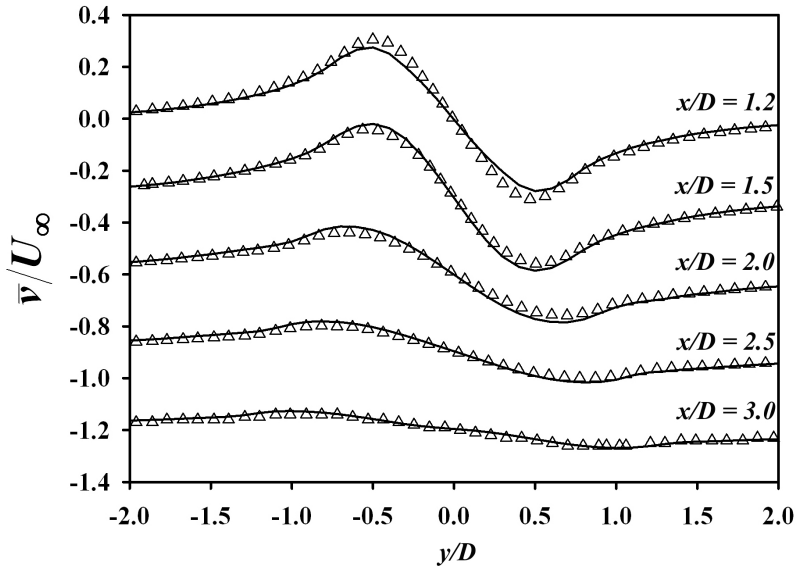


(b) $Re = 1000$

Figure 3.23: Time history of drag and lift coefficients at $Re = 300,\ 1000$ - Strouhal number are calculated using the frequency of the lift coefficient curve.

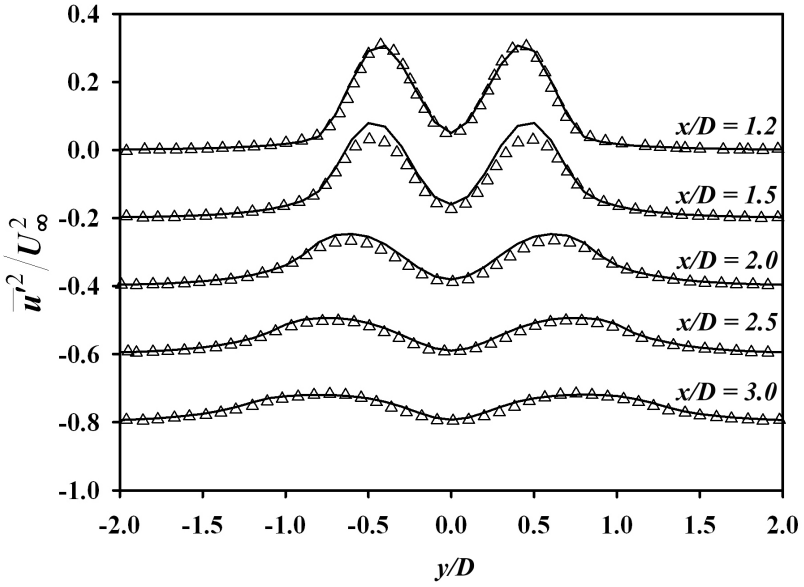


(a) $\overline{u}/U_\infty$

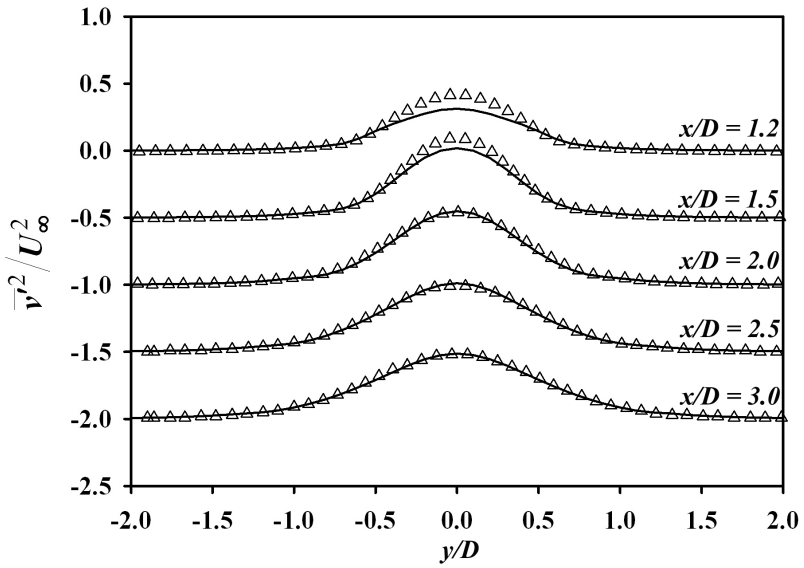


(b) $\overline{v}/U_\infty$

Figure 3.24: First velocity moment near the cylinder at $Re = 300$ - Wake statistic are calculated on 5 different locations near the cylinder for $y/D \in [-2, 2]$. Note that the values are shifted for each curve in $-x$ direction by constant values to compare the results to the previous studies. △, Apte et al. [151], —, Current Simulation.



(a) $\overline{u'}^2/U_\infty^2$



(b) $\overline{v'}^2/U_\infty^2$

Figure 3.25: Second central moments of the velocity components near the cylinder at $Re = 300$ - see the caption of the Figure 3.24.

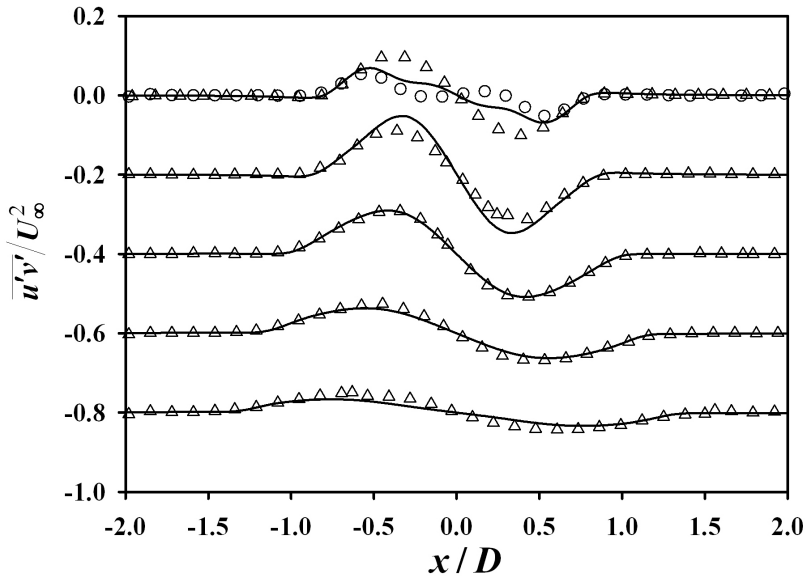


Figure 3.26: Velocity correlation $\overline{u'v'}^2/U_\infty^2$, at $Re = 300$ - △, Apte et al. [151]. ∘, Mittal and Balachandar [154]. —, Current Study.

the number required material CVs since in this example the grid spacing is very fine in the vicinity of the cylinder and a large h_m/h significantly increases the number of material CVs and consequently the computation time. This very fine grid is required to capture a very thin layer of vortices observed on the surface of the cylinder, see Figure 3.27 for $\omega t = 90°$ and $\omega t = 288°$. Figure 3.27 shows the vorticity contours at two different phase angles which is in agreement with 2D simulations of Kim and Choi [132] and 3D simulation of Apte et al. [151]. Figure 3.28 compares the fluid velocity at a fixed x-position in different phase angles with the results of Kim and Choi [132]. This example show that both methods can generate accurate results for active objects.

3.6 Conclusion

In this chapter several numerical strategies for the solution of particulate flows are extensively studied. Properties of discrete delta functions which are required for a class of IBM and FD methods are numerically examined and a 3–point version was found to be adequate (in terms of accuracy) and efficient for the application to the particulate systems. Two different IBM and a FD method are adapted for a SIMPLE

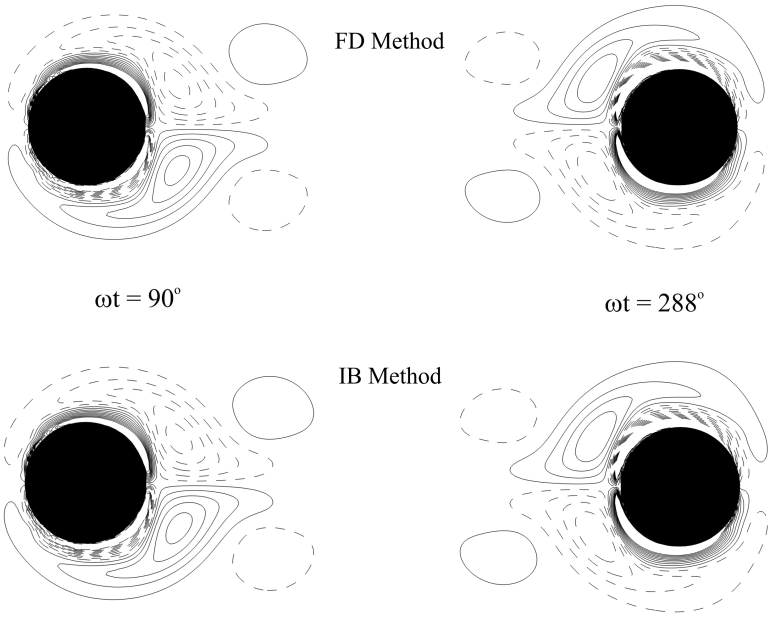


Figure 3.27: Evolution of z-vorticity for IB and FD methods - Normalized vorticity contours ($\omega_z D/U_m$) plotted at four different phase angles. Vorticity contours are plotted in 0.85 increments between -8.5 and 8.5.

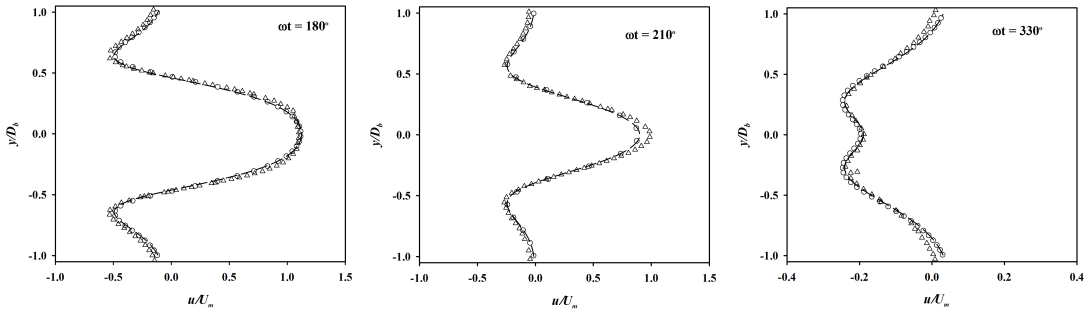


Figure 3.28: Normalized velocity plots for the proposed IB and FD methods - Normalized velocity (u/U_m) in x-direction at three different phases of the oscillatory motion plotted at $x = -0.6D$. △ Kim and Choi [132] simulation, --- Normalized velocity (current FD method), ∘ Normalized velocity (current IB method).

type pressure correction type solver and are rigorously validated. It is found that the methods can efficiently capture the flow characteristics around immersed objects. It was also discussed why the suggested FD method is better suited for the application to other transport phenomena and hence will be considered for extension to heat transfer and moving object problems in the next two chapters.

4

Application to the heat transfer problems

In chapter 3, different DNS methods for fully–resolved simulation of particulate flows were reviewed and their properties were compared using several numerical examples. A DDF IB method and an FD method were rigorously tested and the FD method was identified to be more capable for simulating other transport phenomena since the physical properties of the solid particles are considered in this method. In this chapter the proposed FD method will be extended to heat transfer problems and in addition to comparing the results against numerical benchmarks, a correlation for the calculation of the local Nusselt number around a circular cylinder will be suggested.

4.1 Introduction

Application of IBM or FD method to heat transfer problems is not as developed as the application to hydrodynamic problems and far less studies are available in the literature. ALE methods are also used to study heat transfer in particulate flows, e.g Gan et al. [71], Feng et al. [164], however as discussed in Section 3.1 these are not efficient methods for particulate flows and are not discussed any further here. Yoon et al. [165] used an immersed boundary method to study the fluid flow and heat transfer around a circular cylinder near moving walls. Kim and Choi [166] used immersed boundary method to study the both mixed and forced convection around a pair of stationary circular cylinders. Pacheco et al. [167] and later Pan [168] and Kim et al. [169] studied the

natural convection heat transfer in a square box (Pan [168] also considered inclined cavities) with a circular cylinder in different vertical positions by extending the IBM to heat transfer problems. Zhang et al. [170] studied the convective heat transfer from the surface of a circular cylinder at different Reynolds numbers using IBM. Wang et al. [171] used a multiple direct forcing IBM with heat transfer to study the heat transfer from a staggered tube bank. A notable application of IBM with heat transfer to particulate flows can be found in Feng and Michaelides [172]. They used the code for simulation of heat transfer from 56 hot settling circular cylinders to the surrounding fluid in a closed box.

An advantage of the current FD method is that since the inside of the solid objects are actually considered as real physical domains with specific physical properties other transport phenomena such as heat transfer can easily be implemented and phenomena inside the particle or solid object such as temperature gradient can easily be captured. Recently Yu et al. [173, 174] used an explicit DLM/FD, Section 3.4, method to simulate the coupled heat transfer inside and outside the particle domain. Wachs [175] also used similar explicit DLM/FD approach with an FE solver to simulate the motion of particles with internal heat generation under the influence of flow induced by natural convection. There are no non-DLM/FD implementations with heat transfer in the literature and the method will be discussed in detail in this chapter.

4.2 Governing equations

To extend the FD method to heat transfer problems the scalar transport equation (2.1), is written for a fluid with variable physical properties, viz

$$\frac{\partial \rho c_p T}{\partial t} + \frac{\partial \rho u_j c_p T}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\kappa \frac{\partial T}{\partial x_j}\right) + q_p + q_{FD}, \tag{4.1}$$

where q_{FD} is the source term to enforce the temperature in Ω_p in case of isothermal simulations and is zero otherwise. q_p is a heat source term in the domain which can also be defined in the particle domain if the particle is not isothermal. c_p and κ are the mixture heat capacity and thermal conductivity which are defined similar to Eq.(3.50) by

$$\varphi = \Theta_p \varphi_p + (1 - \Theta_p)\varphi_f, \tag{4.2}$$

where $\varphi = \{c_p, \kappa\}$ and subscripts p and f are used to refer to the body and fluid properties at some reference temperature respectively.

The buoyancy force $f_{B,i}$, in Eq. (3.48) consists of the buoyant forces on the particle domain caused by the fluid–particle density differences and the buoyant forces on the fluid domain due to the density variations caused by the temperature gradient. If the reference hydrostatic pressure is added to the pressure gradient term, the total body force on the whole domain can be written by

$$f_{B,i} = \left(\Theta_p(\rho_{p_T} - \rho_f) + (1 - \Theta_p)(\rho_{f_T} - \rho_f)\right) g_i, \tag{4.3}$$

where subscript T is used to refer to the current temperature. Generally the temperature effects all the fluid properties however here a Boussinesq approximation is used. This states that all the fluid properties are independent of the temperature except for the density and its variation is only considered in the calculation of the gravitational term. Therefore Eq.(4.3) can be rearranged to get the following equation for the total body force

$$f_{B,i} = -\rho\beta(T - T_{ref})g_i + (\rho - \rho_f)g_i. \tag{4.4}$$

In Eq. (4.4), β is the coefficient of volumetric expansion and is calculated similar to c_p and κ by Eq. (4.2). In this study β is assumed to be constant for the fluid phase and zero for the particle phase.

4.3 Numerical implementation

In case of a body with constant temperature using a rationale similar to one used for momentum equation in Section 3.4.1.2, a source term q_{FD} is added to the energy balance equation to enforce the body temperature in the particle domain. To calculate this source term discretized equation for the temperature in Ω_p can be written viz

$$a_{P,T}T^*_P = \sum a_{nb,T}T^*_{nb} + S_T\mathcal{V} + q^*_{FD}, \tag{4.5}$$

where q^*_{FD} is the guessed Eulerian heat source calculated using the temperature field from the last iteration or the values from the previous time step. It can be assumed

that $q^*_{FD} + q'_{FD}$ is the required source to enforce the required temperature, T^F, in the particle domain such that

$$a_{P,T}T^F_P = \sum a_{nb,T}T^F_{nb} + S_T\mathcal{V} + q^*_{FD} + q'_{FD}. \tag{4.6}$$

Subtracting Eq. (4.6) from Eq. (4.5) and similar to Section 3.4.1.2, neglecting the term $\sum a_{nb,T}(T^*_{nb} - T^F_{nb})$ we have

$$q'_{FD} = a_{P,T}(T^*_P - T^F_P). \tag{4.7}$$

The proportionality constant $a_{P,T}$, for unsteady problems is again suggested to be $\frac{\rho_p c_{pp}}{\Delta t}$ which is similar to the unsteady contribution to the source terms. To implement the method, the current temperature estimates are first projected onto the material CVs by

$$T_m = \sum_{\mathbf{x}\in g_h} T^*(\mathbf{x})\delta_h\left(\frac{\mathbf{x}-\mathbf{X}_m}{h}\right)h^d, \quad 1 < m < N, \tag{4.8}$$

and consequently

$$Q^{FD}_m = \rho_p c_{pp}\frac{T^F_m - T_m}{\Delta t}, \quad m = 1\cdots N. \tag{4.9}$$

Eq. (4.9) is then spread on the Eulerian grid by

$$q'_{FD}(\mathbf{x}) = \sum_{m=1}^{N} Q^{FD}(\mathbf{X}_m)\,\delta_h\left(\frac{\mathbf{x}-\mathbf{X}_m}{h}\right)\upsilon_m \quad \forall \mathbf{x} \in g_h. \tag{4.10}$$

Finally the Eulerian source terms are corrected viz

$$q^*_{FD} \leftarrow q^*_{FD} + \alpha_{T_{FD}} q'_{FD}, \tag{4.11}$$

where $\alpha_{T_{FD}}$ is an under-relaxation factor which is best set equal to the under-relaxation factor used in the energy equation.

4.3.1 Calculation of the local Nusselt number

Local Nusselt numbers around the cylinder can be derived by an energy balance and for the case of isothermal objects can be written viz

$$Nu_s = -\frac{D}{T_p - T_\infty}\left.\frac{dT}{dn}\right|_s, \tag{4.12}$$

where $\frac{dT}{dn}\big|_s$ is the temperature gradient in the normal direction evaluated at a specific location on the surface and T_∞ is the fluid bulk temperature. D and T_p are a length scale and the constant temperature of the object. In case of a 2D cylinder a specific location on the surface can be identified using a single angle θ (where θ is arbitrarily measured from the front stagnation point in this study). Therefore Eq. (4.12) can alternatively be written $Nu_\theta = -D/(T_p - T_\infty)(dT/dn|_\theta)$. To calculate the temperature gradient, temperature is extrapolated on surface using a procedure similar to the one employed for the calculation of the hydrodynamic forces in Section 3.3.1.1. Note that since the heat source is spread on the Eulerian grid points using delta functions, simply setting $T_s = T_p$ results in an under-estimated gradient. Temperature gradient is then calculated by a second order forward approximation

$$\left.\frac{dT}{dn}\right|_{\theta,r=R} = \frac{-3T_\theta + 4T_{r'} - T_{r''}}{||r_i'' - r_{c,i}||}, \tag{4.13}$$

where T_θ, $T_{r'}$ and $T_{r''}$ are the temperature on the surface (extrapolated), first and second interpolation points in the normal direction respectively, see Figure 3.6 for a general 3D presentation.

For unsteady cases time averaged local Nusselt numbers are calculated by collecting data for a problem dependent time interval and using the following equation

$$\overline{Nu}_s^t = \frac{1}{t_2 - t_1}\int_{t1}^{t2} Nu_s\, dt. \tag{4.14}$$

Average Nusselt number for a general case can then be written by

$$\overline{Nu} = \frac{\int_{\Gamma_p} \overline{Nu}_s^t dA}{\int_{\Gamma_p} dA}, \tag{4.15}$$

which reduces to

$$\overline{Nu} = \frac{1}{\pi}\int_0^\pi \overline{Nu}_\theta^t\, d\theta, \tag{4.16}$$

for a single circular cylinder. Not that all the simulations in this section are in the region of either symmetric wake or periodic vortex shedding, therefore averaging on the upper and lower semi-circles are equivalent and saves some computational time.

4.4 Results and discussions

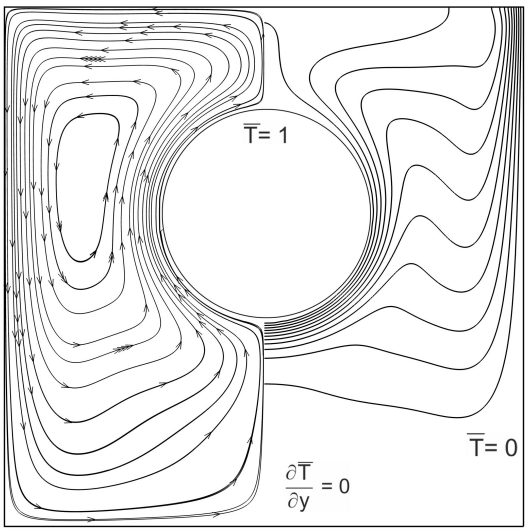
4.4.1 Buoyancy driven flow

Basic heat transfer implementation is tested by considering a buoyancy driven flow using a test case proposed by Demirdzic et al. [176]. The geometry consists of a cavity with sides $H = 1$ with a cylinder ($D = 0.4H$) which is slightly shifted upward ($0.1H$) from the centre of the cavity. Cylinder wall is maintained at a hot temperature where horizontal cavity walls are assumed adiabatic and the vertical walls are kept at a low temperature. The fluid properties are chosen such that Rayleigh number $Ra = Gr.Pr = 10^6$ and $Pr = 10$ and a triangular grid is used to present the body, see Figure 3.15b. Here Prandtl and Grashof numbers are respectively defined by

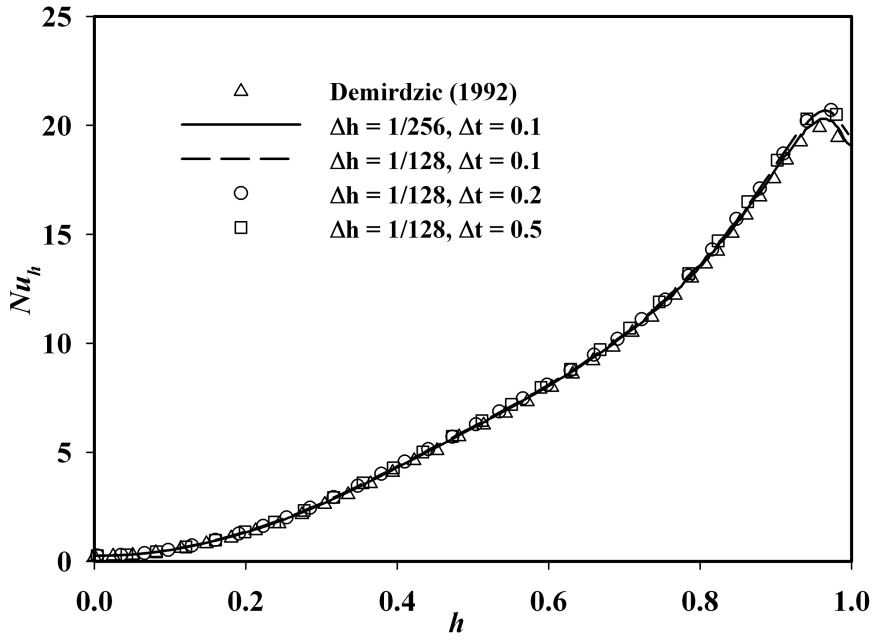
$$Pr = \frac{c_{pf}\mu_f}{k_f} \tag{4.17}$$

$$Gr = \frac{g\beta_f(T_h - T_c)D^3\rho_f^2}{\mu_f^2}. \tag{4.18}$$

Figure 4.1a shows the streamlines and temperature contours of the final steady solution. Local Nusselt numbers (i.e., dimensionless temperature normal derivatives) on the cold wall are calculated and compared to the benchmark results of Demirdzic et al. [176] which are in excellent agreement, Figure 4.1b. Grid and time independence studies are also performed with two different grid levels and three different time levels and the results are plotted in Figure 4.1b. Another important feature of the current source correction procedure is to allow the user to choose very large time step sizes, if the time history of the solution is not the primary concern, whereas Sharma and Patankar [149] proposed estimating the sources at the beginning of the time step which requires much smaller time steps to accurately impose the the rigidity constraint.



(a) Temperature Contours and Streamlines



(b) Local Nusselt Number on the cold wall

Figure 4.1: Simulation of a buoyancy–driven flow in a cavity at $Ra = 10^6$ - 11 contour lines are presented for $\overline{T} = \frac{T-T_c}{T_h-T_c} \in [0, 1]$. Local Nusselt number on the left cold wall is calculated and compared to the benchmark solution of Demirdzic et al. [176] which shows the accuracy of the current method, Grid and time step independence study results are also presented.

4.4.2 Convective heat transfer from a single cylinder - Problem specification

In this section convective heat transfer from a stationary cylinder is considered. Cylinder position and domain size is similar to the problem discussed in the Section 3.5.2. A grid independence study is first performed by considering three different grid levels with 100, 80 and 50 grid points along the diameter of the cylinder. The Prandtl and the Grashof numbers are set to $Pr = 0.7$ and $Gr = 0$ respectively where the Grashof number is defined by Eq. (4.18) except that $(T_p - T_\infty)$ is used as the temperature reference where T_∞ is the far field temperature. Three Reynolds numbers, $Re = 10$, 20 and 40, are considered. Inlet temperature is fixed at $T_{in} = T_\infty = 300$ and body temperature is fixed at $T_p = 301$. Symmetric boundary conditions are applied to the top and bottom walls and standard outflow boundary used at the outlet. Average Nusselt number, defined by Eq. (4.16), is used to assess the convergence of the solution on different grid levels. In Table 4.1, mean Nusselt number calculated for different grid levels is compared to the previous studies. The calculated Nusselt number is generally in very good agreement with previous high resolution numerical studies [177, 178]. Time independence studies are also carried out by considering three time levels of 0.02, 0.05 and 0.1 for $Re = 20$ and the medium size grid and the simulation is run for 100 non dimensional time units which shows independence of the results from the size of the time step.

In addition, to estimate the numerical uncertainties associated with the discretization errors a grid convergence index as suggested by Celik et al. [179] is calculated by first calculating an apparent order p, given by the following set of equations

$$p(q) = \frac{1}{\ln(r_{21})}\left|\ln\left|\frac{\epsilon_{32}}{\epsilon_{21}}\right| + q(p)\right|, \tag{4.19a}$$

$$q(p) = \ln\left(\frac{r_{21}^p - s}{r_{32}^p - s}\right), \tag{4.19b}$$

$$s = \text{sgn}\left(\frac{\epsilon_{32}}{\epsilon_{21}}\right), \tag{4.19c}$$

where $r_{21} = h_{mid}/h_{fine}$, $r_{32} = h_{coarse}/h_{mid}$, $\epsilon_{21} = Nu_{mid} - Nu_{fine}$, $\epsilon_{32} = Nu_{coarse} - Nu_{mid}$ and sgn is the sign function. The system of non–linear equations can be solved using a non–linear solver such as the Newton method [180]. The grid convergence index (GCI) can then be calculated by

$$GCI^{21} = \frac{1.25\,|\epsilon_{21}/Nu_{fine}|}{r_{21}^{p} - 1}. \tag{4.20}$$

The results of these calculations are also presented in Table 4.1 which shows that the uncertainty levels are tight for this grid level which is used for the rest of the simulations.

Study	Mesh Size	$Re = 10$	$Re = 20$	$Re = 40$
Current Study	Fine Mesh	1.81±0.018	2.39±0.012	3.17±0.041
	Medium Mesh	1.80	2.38	3.15
	Coarse Mesh	1.75	2.31	3.07
Soares et al. [177]	-	1.86	2.43	3.20
Bharti et al. [178]	-	1.86	2.47	3.28

Table 4.1: Comparison of mean Nusselt number with previous analytical and numerical studies for different Reynolds numbers and $Pr = 0.7$.

Local Nusselt numbers are calculated on 240 points around the upper half of the cylinder, see Figure 4.2a for the definition of the angle θ and the orientation of the cylinder. All simulations are run for 300 non-dimensional time units (tU_∞/D) which ensures that the flow has reached the steady condition for lower Reynolds numbers and the stationary state for higher Reynolds numbers [154, 181]. Local Nusselt numbers are calculated and collected for the 400 final time steps where the non–dimensional time step size is 0.25.

4.4.3 Convective heat transfer from a single cylinder - A new correlation

There are several correlations for the calculation of average Nusselt number from circular cylinders applicable to different ranges of Re and Pr numbers. Generally these correlations have some form of a power law relation in terms of the Re and Pr:

$$\overline{Nu} = \sum_{\jmath=1}^{M}\sum_{\ell=1}^{L} C_{\jmath,\ell} Re^{m_\jmath} Pr^{n_\ell}, \tag{4.21}$$

where M and L are small integers. Perkins [182] and later Perkins and Leppert [183] suggested a power low type correlation for $\overline{Nu}$ which is applicable for $40 < Re < 10^5$ and $1 < Pr < 300$ using $M = 2$ and $L = 1$. Fand [184] provided a correlation similar to that of Perkins and Leppert [183] with $M = 2$ and $L = 1$ for $0.1 < Re < 10^5$ and

no suggestion on the range of Pr. Zukauskas and Ziugzda [185] provided a very simple correlation with both $M = 1$ and $L = 1$ for a narrower range of Reynolds numbers between $Re = 2 \times 10^2$ and $Re = 10^5$. Another notable correlation in this class is that of Whitaker [186] with $M = 2$ and $L = 1$ applicable for $1 < Re < 10^5$ and $0.67 < Pr < 300$. Churchill and Bernstein [187] suggested using rational functions and provided a correlation for $10^2 < Re < 10^7$ and $Pr\,Re > 0.2$.

Although the aforementioned correlations are widely used in science and engineering for the estimation of the average Nusselt numbers, they provide absolutely no information about the local values of the Nusselt number. The values of parameters $m_\jmath$ and n_ℓ are very strong functions of position around the cylinder due to the formation, growth and the separation of the boundary layer and also formation of the recirculating bubble. At higher Re periodic vortex shedding also affects the value of the local Nusselt number. This information is essential for optimization of high performance heat exchangers were the knowledge of the local values of the Nusselt number can be used to identify the poor transfer regions, which can consecutively help in the determination of the vortex generator position, tube pitches, number of tube rows and other design parameters. To the best of author's knowledge the only study that has addressed this problem is that of Sanitjai and Goldstein [188, 189]. They divide the circumference of the cylinder into three sections namely $0° - 85°$, $85° - 135°$ and $135° - 180°$ and provided separate correlation for each section for $2 \times 10^2 < Re < 10^5$ and $0.7 < Pr < 176$. In Table 4.2 and Table 4.3 similar to Table 4.1 the value of the mean Nusselt numbers are compared to the values calculated from different correlations.

Study	$Re = 10$	$Re = 20$	$Re = 40$
Current Study medium mesh	1.82	2.39	3.18
Fand [184]	1.79	2.44	3.39
Whitaker [186]	1.33	1.93	2.80
Sparrow et al. [190]	1.60	2.21	3.10

Table 4.2: Comparison of mean Nusselt number with previous correlations for different Reynolds numbers and $Pr = 0.7$.

Available correlations are usually suggested for large ranges of Prandtl and Nusselt numbers with very simple equations and it is expected not to provide very accurate

Study	$Re = 80$	$Re = 90$	$Re = 100$
Current Study medium mesh	15.38	16.30	17.16
Fand [184]	12.39	13.14	13.84
Whitaker [186]	14.58	15.53	16.4443
Sparrow et al. [190]	13.64	14.52	15.35

Table 4.3: Comparison of mean Nusselt number with previous correlations for different Reynolds numbers and $Pr = 17$.

results especially near the boundaries of the suggested intervals for Re and Pr. This is evident from the comparison to the correlations provided in [184, 186]. It seems that correlation suggested by Fand [184] captures the lower end more accurately than that suggested by Whitaker [186], despite the fact that both are suggested for almost the same range of Re and Pr. However at slightly higher values of Reynolds number the other correlation performs better. This problem is also mentioned by Sparrow et al. [190] and they provided another correction using the same simple form of the equation to correct the discrepancy between these correlations. However their suggestion still does not provide a very accurate estimate in addition to the initial problem than none of them provide any information on the local values of the Nusselt number. it is obviously that these simple equations can only be used as crude estimates.

The changes in the local Nusselt number, $\overline{Nu}_\theta^t$, curves are shown in Figure 4.2a and 4.2b. Generally these curves can be interpreted as follows. The Nusselt number increases by increasing both the Reynolds number and Prandtl number. The maximum value of the Nusselt number is observed around the front stagnation point where the cold fluid coincides the hot cylinder for the first time and it gradually decreases to a minimum value near the separation point where the thickness of the thermal boundary layer reaches a maximum. Interpretation of the local Nusselt number values is more difficult after this point and will be discussed next.

Figure 4.3 shows the streamlines superimposed on the temperature contours for $Re = 20$ and 250 and different Prandtl numbers. For $Pr = 0.5$ and $Re = 20$, since heat transfer is mainly dominated by a diffusion process and the thermal boundary layer is thicker than the velocity boundary layer, the value of the local Nusselt number remains essentially the same due to the formation of a symmetric separated bubble which is not

strong enough to enhance the convective heat transfer by distorting the thick thermal boundary layer and the transfer mechanism remains dominated by diffusion, see Figure 4.3a and 4.2a. Increasing the Prandtl number and keeping $Re = 20$ reduces the thickness of the thermal boundary layer and therefore the steady wake can enhance the convective heat transfer and after a minimum the value of the local Nusselt number slightly increases, see Figures 4.2b and 4.3b. For larger values of Reynolds number and low Prandtl numbers although the heat transfer process is still diffusion dominated, strong vortex shedding process pushes the diffusive region toward the cylinder wall, Figure 4.3c and hence forced convection takes place at the rear of the cylinder which consequently increases the local Nusselt number, see Figures 4.2a. At large Reynolds and Prandtl numbers strong vortices also increase the local Nusselt number. However a hump is observed in the curve, Figure 4.2b, which can be related to a small secondary recirculating bubble, size of which is comparable to the thickness of the thermal boundary layer at this point. This bubble acts as a barrier in this region, preventing the cold fluid from contacting the cylinder and generates another local minimum in the curve. The generation of this hump depends both on Re and Pr for example at $Re = 250$ by examining all the curves this hump is first observed at $Pr \approx 5$. This is due to the fact that although a similar bubble forms at lower Prandtl numbers the size of the bubble is not comparable to the thickness of the thermal boundary layer and hence has no effect on the heat transfer process.

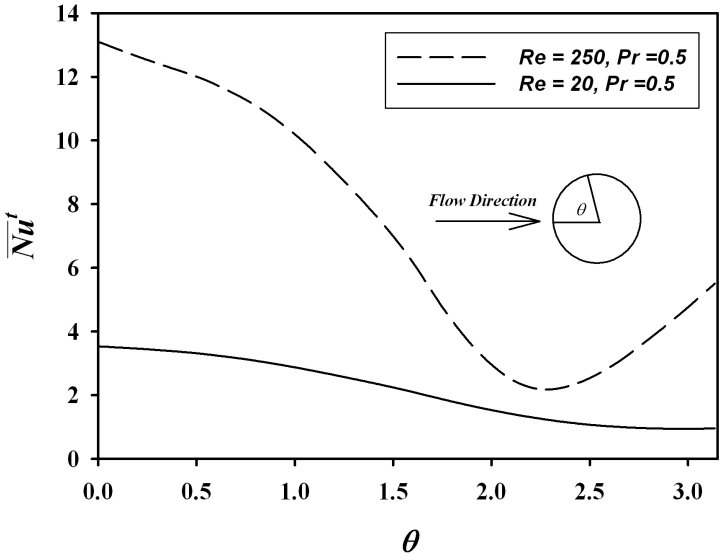
4.4.3.1 Modeling the Local Nusselt number curves

To model the local Nusselt number curves two different models are considered a polynomial and a trigonometric series model. These can be written by

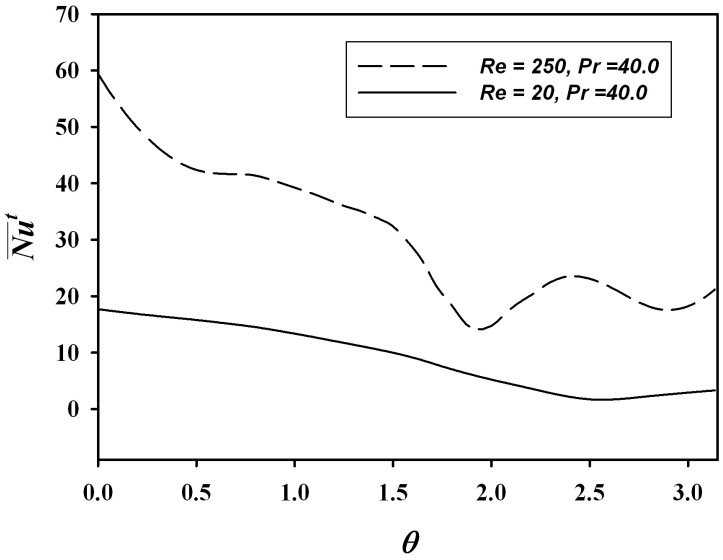
$$\widehat{\overline{Nu}^t}_{poly}(\theta) = a_0 + \sum_{n=1}^{M} a_n \theta^n, \tag{4.22}$$

$$\widehat{\overline{Nu}^t}_{tri}(\theta) = a_0 + \sum_{n=1}^{M} a_n \sin(n\theta) + b_n \cos(n\theta). \tag{4.23}$$

A commonly used measure for the goodness of a fit is the coefficient of determination or R–squared (R^2) measure defined by

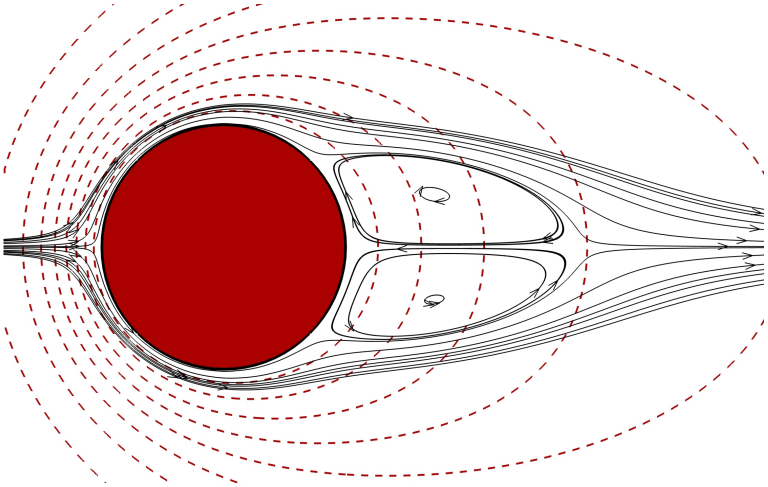


(a) Pr = 0.5

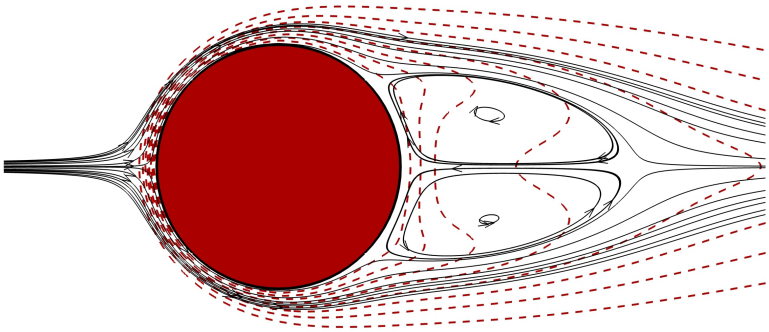


(b) Pr = 40

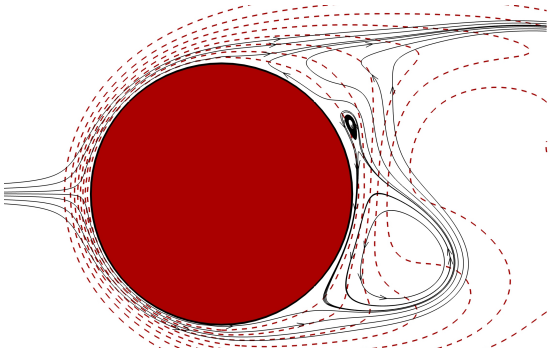
Figure 4.2: Local Nusselt number curves - two different Pr for two different flow regimes are presented: Steady recirculating bubble at $Re = 20$ and periodic vortex shedding at $Re = 250$.



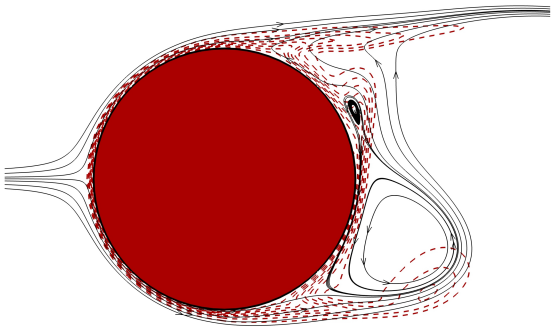
(a) $Re = 20,\ Pr = 0.5$



(b) $Re = 20,\ Pr = 40$



(c) $Re = 250,\ Pr = 0.5$



(d) $Re = 250,\ Pr = 40$

Figure 4.3: Streamlines superimposed on the temperature contours around the cylinder - non–dimensional temperature contours $(T - T_\infty)/(T_p - T_\infty)$ between 0.2 and 1 in 0.1 increments.

$$R^2 = 1 - \frac{SSE}{SST}, \tag{4.24}$$

where SSE and SST are the sum of squares of residuals and sum squares about the mean. Having defined the simulated (exact) local values of the time averaged Nusselt numbers by $\overline{Nu}^t_\theta$ and the modelled time averaged values of the local Nusselt number (Eq. (4.22) and Eq. (4.23)) by $\widehat{\overline{Nu}^t}_\theta$, SSE and SST are defined by

$$SSE = \sum_{n=1}^{N_{Nu}} (\overline{Nu}^t_{\theta_n} - \widehat{\overline{Nu}^t}_{\theta_n})^2, \tag{4.25}$$

$$SST = \sum_{n=1}^{N_{Nu}} (\overline{Nu}^t_{\theta_n} - \overline{Nu})^2. \tag{4.26}$$

Note that in the definition of SST the mean value used should be an ensemble mean, i.e $1/N_{Nu} \sum_{n=1}^{N_{Nu}} \overline{Nu}^t{}_{\theta_n}$, however since the points used for the calculation of the local Nusselt number are equidistant along the upper semicircular arc, both definitions are equivalent. The models used in this study are complicated and have many parameters which increases the risk of over–fitting the data, therefore such simple measures are not reliable to choose between different models. In this study more reliable adjusted R–squared values is used to decide on the final form of the model. The adjusted R–squared is defined by [191, 192], $R^2_{adj} = \gamma R^2$, where γ is the shrinkage factor given by

$$\gamma = \frac{(N_{Nu} - 1)R^2 - \mathscr{M}}{(N_{Nu} - \mathscr{M} - 1)R^2}, \tag{4.27}$$

where $\mathscr{M}$ is the number of fitted coefficients (i.e $2M + 1$ for Eq.(4.23) and $M + 1$ for Eq.(4.22)). Similarly a corrected root mean square error can be defined by

$$RMSE = \sqrt{SSE/(N_{Nu} - \mathscr{M} - 1)}. \tag{4.28}$$

To choose a model that describes the local Nusselt number curves 8 models are proposed namely Eq. (4.22) with $M \in \{6, 7, 8, 9\}$ and Eq. 4.23 with $M \in \{4, 5, 6, 7\}$. Local Nusselt number data from all 600 simulations are fitted to the proposed models and minimum R^2_{adj} and maximum $RMSE$ are calculated for all the fitted curves. Tables 4.4 summarizes the results. Noting that Polynomial8 and Trigonometric4, have the same number of coefficients obviously Eq.(4.23) is a better model for the current data both

in terms of maximum error ($RMSE$) and deviation from the mean (R^2_{adj}). Also checking the R^2_{adj} values for the proposed model in the Trigonometric category shows that addition of more terms actually improves the fitting. However Trigonometric7 is not significantly more accurate than the Trigonometric6 and the simpler Trigonometric6 model is chosen for further tests. Note that yet another level of fitting is required to write the 13 coefficients in this model in terms of the Re and Pr and hence, although Trigonometric7 is a slightly better model the model with smaller number of coefficients is preferred, i.e Trigonometric6.

Model	max $RMSE$	min R^2_{adj}	mean $RMSE$	mean R^2_{adj}
Polynomial6	2.522372	0.955650	0.587655	0.992011
Polynomial7	1.683310	0.980248	0.395748	0.996462
Polynomial8	1.626895	0.981550	0.344830	0.997168
Polynomial9	1.348431	0.987327	0.275158	0.998228
Trigonometric4	1.311299	0.988014	0.271557	0.998251
Trigonometric5	0.752640	0.996023	0.155081	0.999433
Trigonometric6	0.453601	0.998555	0.096935	0.999781
Trigonometric7	0.314185	0.999307	0.068235	0.999892

Table 4.4: Comparison of different model fitted to the local Nusselt number curves.

Worst fit in the Trigonometric6 category in terms of the R^2_{adj} and $RMSE$ occurs at $Re = 240$ and $Pr = 40$. Figure 4.4a shows the simulation data points in addition to the fitted model. Clearly a very high quality fit is provided by this model. However several statistical tests are also performed. Upper and lower prediction bounds for this fit is also presented in this figure which shows with very high probability (99%) any predition falls in the tight interval presented in the figure. Alternatively the value of the local Nusselt number is evaluated on 21 randomly selected points in $[0, \pi]$ which shows the ability of the model to precisely interpolate to new values.

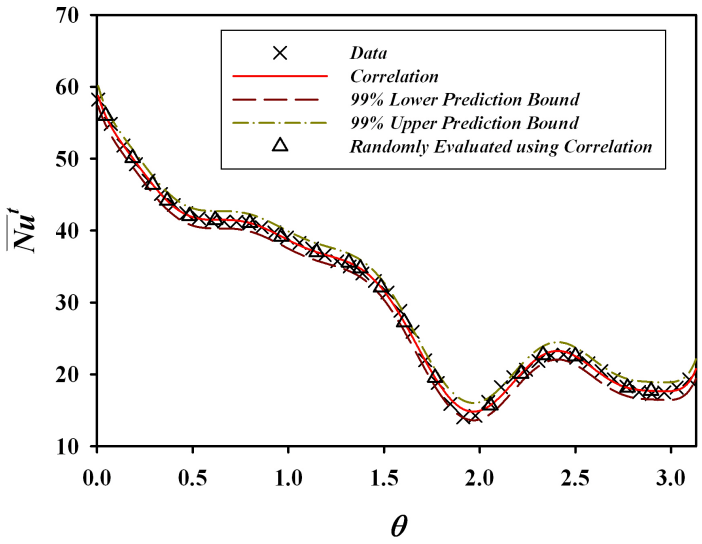
Since an optimization is performed to fit the coefficients it is expected that for a high quality model the residual values be randomly distributed around zero. This ensures that the integration of the curve, for example to calculate $\overline{Nu}$ does not systematically produce under or overestimated values. Figure 4.4b clearly shows the random distribution of the error around zero. However the model produces larger error around $\theta = 2$

due to the complexity of the curve in this region. Noting that this case is the worst fit in all the 600 fits, therefore the analysis clearly show the high accuracy and quality of the model.

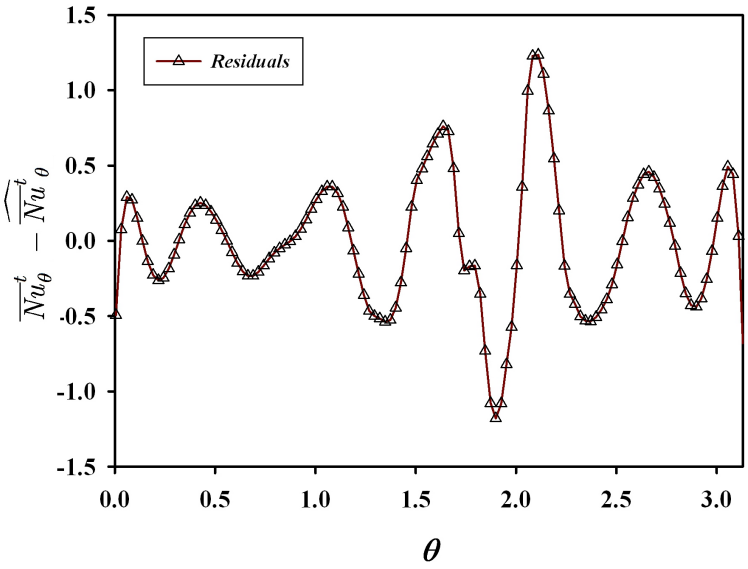
4.4.3.2 Modeling the coefficients

Next step is to model the coefficients of the Trigonometric6 model in terms of the Re and Pr. For this step 2D polynomials of order M in Re and L in Pr are considered. Nine tests are performed with $M = 3$, $L = 3$ to $M = 5$ and $L = 5$. Increasing M significantly improves the goodness of the curve in terms of $RMSE$ and R^2_{adj} but increasing the L beyond 3 has no significant effect on the quality of the fit. Therefore the simpler model with less coefficients is chosen, i.e $M = 5$ and $L = 3$. Accurately fitting a single equation to the whole range of Prandtl number for all the 13 coefficients of the Trigonometric6 model is not possible due to the significant changes in the nature of the curves. A single model either produces large errors in the lower end or higher end of the spectrum. Therefore two separate equations are fitted for $0.1 \leq \mathrm{Pr} \leq 5$ and one for $5.0 < \mathrm{Pr} \leq 40$. Figure 4.5 shows the worst fit for each range of Prandtl numbers. Worst case happens for a_1 for $0.1 \leq \mathrm{Pr} \leq 5$ and for b_1 for $5.0 < \mathrm{Pr} \leq 40$ with values of R^2_{adj} being 0.986789 and 0.918145 respectively. Coefficients for these equations are given in Appendix B. In addition to the aforementioned polynomials a cubic interpolant is also attempted on the whole domain. Figure 4.6 shows the interpolant for the whole data range for coefficients a_1 and b_1 corresponding to the worst fits using the polynomial model. Using this method no sectioning is required and the whole domain can be interpreted using a single interpolant.

To rigorously check all modelling processes all the available Re and Pr points are fed to the both polynomial and the interpolant model to calculate the coefficients of the Trigonometric6 model for the whole data set. Then the Trigonometric6 model is integrated using Eq (4.16) to yield a single averaged Nusselt number. This value is then compared to the value directly calculated by the code. Maximum and average observed errors are 9.83% and 1.75% where the maximum error is observed on the lowest end of both Re and Pr lines for the polynomial model. Maximum obsereved error for the interpolant model is 2.16% and the average error is 0.66%. Additionally 12 further simulations at intermidiate points are performed to further check the quality of the modelling for the possible oscillation in the fitted models. Figure 4.7 cmpares the



(a) Fitted Model



(b) Residuals

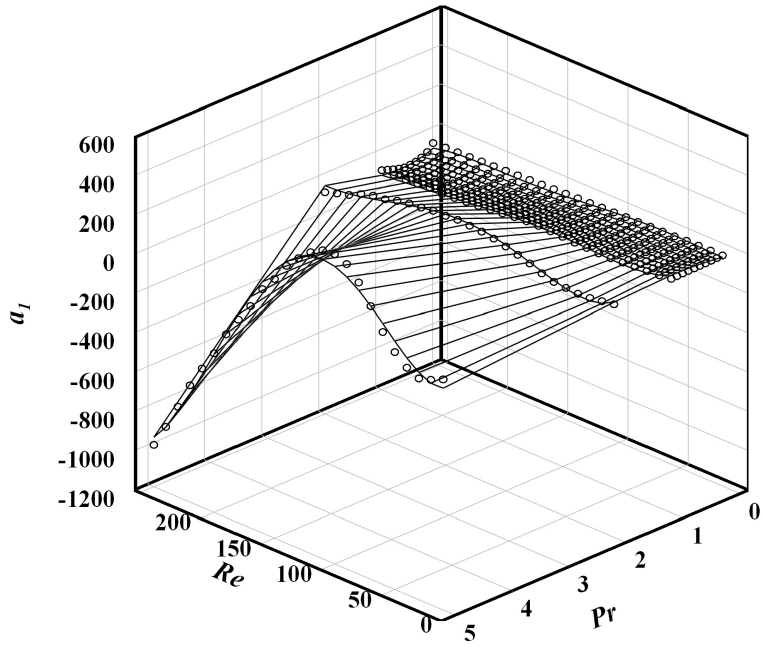
Figure 4.4: Analysis of the worst fit at $Re = 240$ and $Pr = 40$ for the Trigonometric6 model. - *RMSE* and R^2_{adj} are reported in Table 4.4. For better presentation only $N_{Nu}/5$ of the data point are presented in Figure 4.4a, and $N_{Nu}/2$ in Figure 4.4b.

average Nusselt number calculated using the suggested models with values calculated directly in the simulation. Both methods provide very accurate results with the cubic interpolant always providing better estimates. All the errors are less than the maximum errors reported earlier which shows that no large oscillation happens in the fitted models and the fitted model smoothly describes the whole data.

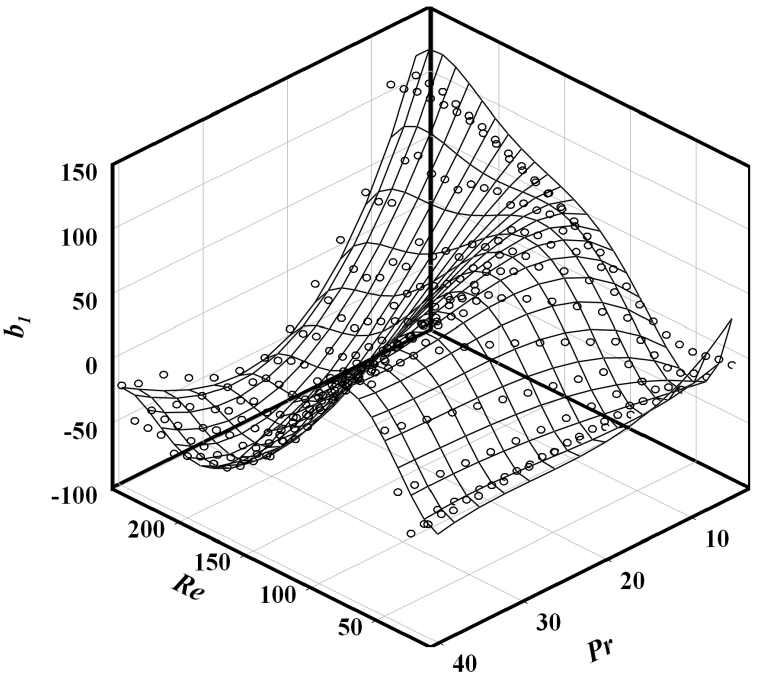
As a final check of the proposed correlations the local Nusselt number defined by Eq. (4.14) for $Re = 20$, 120 and 218 is compared to the previous experimental and numerical calculations in Figure 4.8. Prandtl number is set to $Pr = 0.73$ and 20 for $Re = 20$ and to 0.7 for $Re = 218$ and $Re = 120$ which are chosen to be similar to other studies for comparison [166, 170, 171, 178, 193]. Variation of the Nusselt number around the cylinder is as expected, and the shape of the curve is captured with very high precision, see Figure 4.8.

4.5 Conclusion

In this Chapter extension of the FD domain method to heat transfer problems is discussed in detail and several numerical tests are performed. The hierarchical modelling strategy is demonstrated by providing a very high quality correlation for the estimation of the local Nusselt numbers around isothermal circular cylinders. The local Nusselt number curves are accurately described using a trigonometric series with 13 parameters which is shown to perform better than polynomials with the similar number of coefficients. 13 Coefficients of the Trigonometric6 model are then correlated to the Pr and Re using a very dense data set consisting of 600 simulations. Two models are suggested for writing the coefficients in terms of the Reynolds and Prandtl numbers, namely 2D polynomials and cubic interpolants. Both methods are rigorously tested and is shown that maximum error never exceeds 9.83% using the polynomial model and 2.16% using the cubic interpolants which corresponds to at least a four fold increase in the accuracies. The suggested correlations not only surpass the accuracies provided by previous correlations for calculating the average Nusselt number, but also provide detail information of the local heat tranfer effects which is not provided in any other correlations. This of course comes at a much higher computational cost due to the larger number of arithmetic operations.

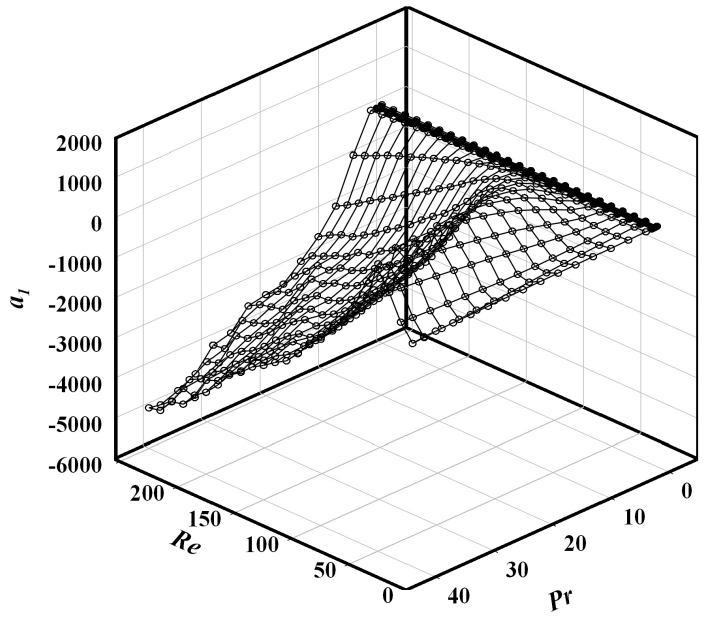


(a) $0.1 \leq Pr \leq 5$

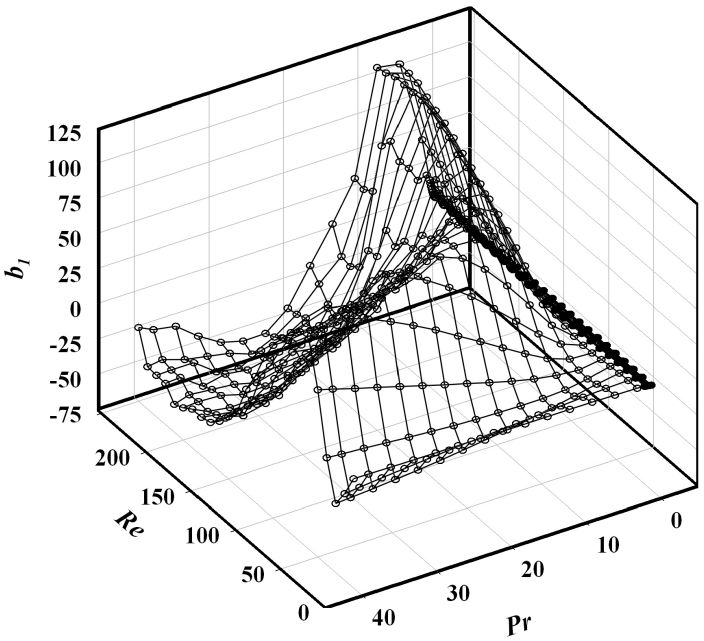


(b) $5 < Pr \leq 40$

Figure 4.5: Analysis of the worst fit to the coefficients of Trigonometric6 model.
- Wireframe present the fit, scatter circles are the original coefficients.



(a) a_1 Coefficient



(b) b_1 Coefficient

Figure 4.6: Analysis of the cubic interpolant. - Scatter circles are the original coefficient values and the wireframe present the interpolant. Note that only two coefficients corresponding to worst polynomial fits are presented.

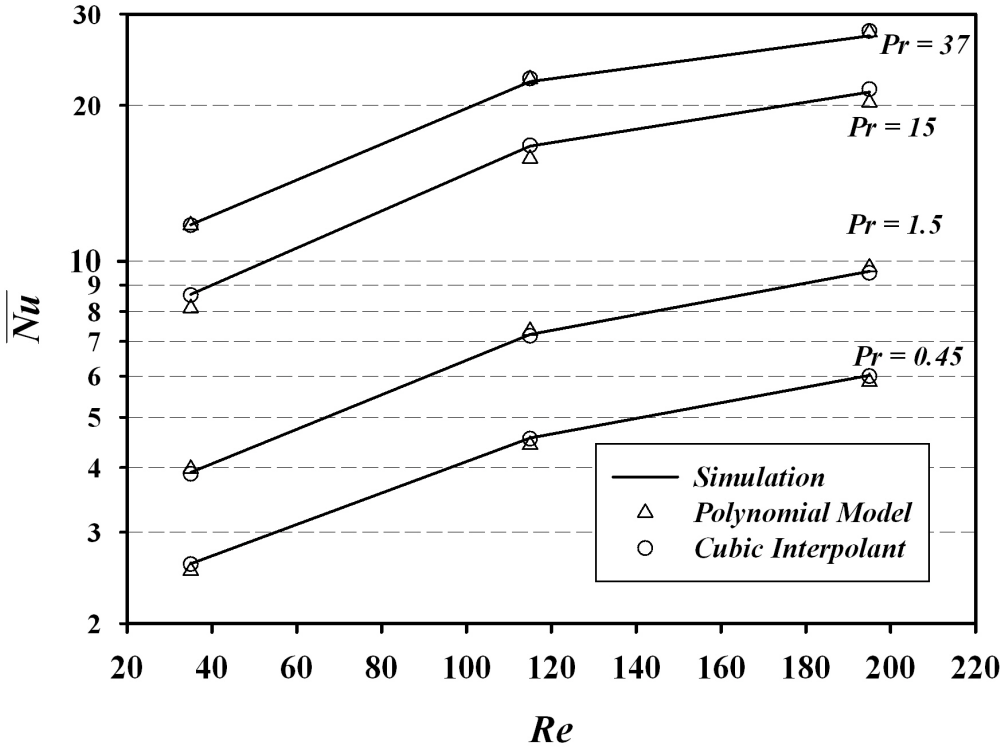
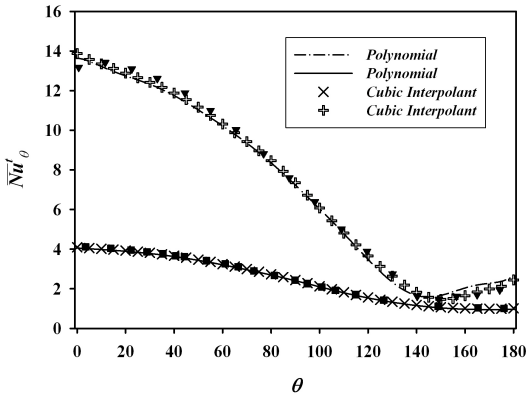
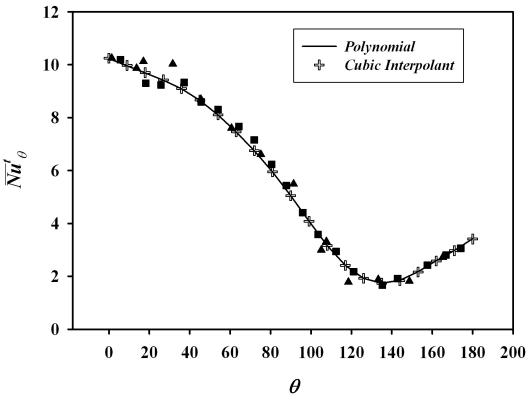


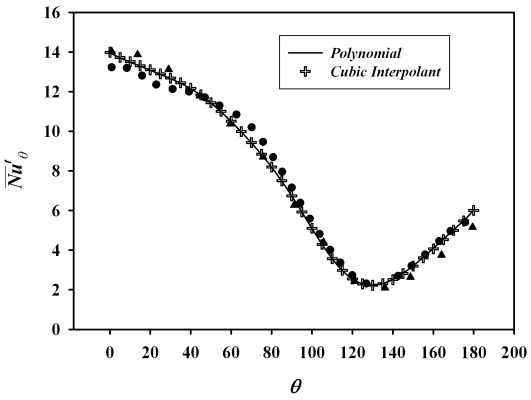
Figure 4.7: Average Nu on the middle points - $\overline{Nu}$ is calculated using polynomial model and a cubic interpolant on 12 middle points.



(a) $Re = 20$.



(b) $Re = 120$, $Pr = 0.7$.



(c) $Re = 218$, $Pr = 0.7$.

Figure 4.8: Local Nusselt number $\overline{Nu}^t_\theta$ for convective heat transfer from a circular cylinder - For $Re = 20$: ■ Wang et al. [171] $Pr = 0.73$, ▼ Bharti et al. [178] $Pr = 20$. For $Re = 120$, $Re = 0.7$: ■ Kim and Choi [166], ▲ Eckert and Soehngen [193]. For $Re = 218$, $Pr = 0.7$: • Zhang et al. [170], ▲ Eckert and Soehngen [193]

5

Application of the method to moving objects and extension to 3D problems

In Chapters 3 and 4 basic formulation of the current FD method and application to heat transfer problems were considered. In this chapter the full algorithm for freely moving particles and extension to 3D simulations and the full calculation process for SIMPLE algorithm will be discussed in detail. A review of the current state of particulate flow simulations using either the IB or FD methods has already been provided in Chapter 3 and hence no further introduction is provided in this chapter. It is also worth mentioning that the particle phase is not parallelized and hence all the simulations in this section including the 3D simulations are performed serially. In Chapter 8.1 some discussions and suggestions for the parallelization of the particle phase are provided.

5.1 Particle Motion

It was discussed in Section 3.4 that the body force f_i which enforces the rigidity constraint inside the particle domain can be written by Eq. (3.51), which is reiterated here for reference:

$$\rho_p \frac{u_i^R - u_i^l}{\Delta t} = f_i.$$

It was then discussed that this equation does not provide any additional information regarding the unknown field u_i^R. However in case of stationary objects discussed in the previous chapters, u_i^R is predetermined and no further equation is required. Normally u_i^R should be calculated by adding the translational and rotational components of the particle velocity given by Eq. (3.47) viz,

$$u_i^R = U_{p,i} + \epsilon_{ijk}\omega_{p,j}r_k,$$

where $U_{p,i}$ and $\omega_{p,i}$ should be calculated using the Newton's equation of motion:

$$M_p\frac{dU_{p,i}}{dt} = F_{s,i} + (\rho_p - \rho_f)\mathcal{V}_p g_i \tag{5.1}$$

$$\frac{dI_{p,ij}\omega_{p,j}}{dt} = T_{s,i}. \tag{5.2}$$

Alternatively in a coordinate frame fixed in the body Eq. (5.2) becomes:

$$I_{p,ij}\frac{d\omega_{p,j}}{dt} + \epsilon_{ijk}\omega_{p,j}I_{p,k\ell}\omega_{p,\ell} = T_{s,i}, \tag{5.3}$$

where M_p is the total mass of the particle, $I_{p,ij}$ is the moment of inertia tensor and $\mathcal{V}_p$ is the total volume of the particle. Since a discretization of the object is available in the FD method, $I_{p,ij}$ can directly be calculated using its general definition [194]:

$$I_{p,ij} = \int_{\Omega_p} \rho_p[r_k r_k \delta_{ij} - r_i r_j]dv, \tag{5.4}$$

where r_i is the position vector from a point in the body domain to the centroid of the body. $F_{s,i}$ and $T_{s,i}$ are the total hydrodynamic force given by Eq. (3.21) and the torque acting on the surface of the particle which can similarly be written by

$$T_{s,i} = \int_{\Gamma_p} \epsilon_{ijk}r_j\sigma_{k\ell}n_\ell ds. \tag{5.5}$$

It was then discussed in Section 3.3.1.1 that the interpolation of the total stress tensor on the surface of the particle is extremely time consuming and hence several remedies are given in the literature. Another notable advantage of the FD method is that since the particle domain is considered as real physical domain with specific physical properties, solution of the NS equation also governs the motion of the particle. This means that since the NS equations are conservation of momentum equations the particle

whether moving rigidly or not, its total momentum is governed by the NS equations. Therefore the total momentum of the particle remains the same when projecting to a rigid motion, this principle was first proposed by Patankar [146] and later also used by Veeramani et al. [150] and Apte et al. [151] in a finite–element and fractional–step frameworks respectively. Using this principle following equations can be written for the translational and angular momentum of the particle

$$M_p U_{p,i} = \int_{\Omega_p} \rho_p u_i dv \tag{5.6}$$

$$I_{p,ij}\omega_{p,j} = \int_{\Omega_p} \epsilon_{ijk} r_j (\rho_p u_k) dv. \tag{5.7}$$

Using Eq. (5.6) and Eq. (5.7) to calculate angular and linear velocities of the particle, u_i^R can easily be calculated from Eq. (3.47) and the rigidity force from Eq. (3.51). Also note that particle motion is naturally governed by Eq. (5.6) and Eq. (5.7) which are merely additions and there is no need to explicitly calculate the surface forces for the motion of the particle which can reduce the computational time significantly. The position of centre point is then updated simply by

$$\frac{dX_{p,i}}{dt} = U_{p,i}. \tag{5.8}$$

Any other point inside the body domain undergoes both translational and rotational motions around the axis of rotation defined by $\omega_{p,i}$. Assuming an internal point at time t_0 is located at x_i^t its location at $t + \Delta t$, $x_i^{t+\Delta t}$, can be calculated using

$$x_i^{t+\Delta t} = X_{p,i}^{t+\Delta t} + R_{ij}\left(x_j^t - X_{p,j}^t\right), \tag{5.9}$$

where R_{ij} is the rotation tensor defined by [194]

$$R_{ij} = \delta_{ij}\cos\alpha + \epsilon_{ikj} e_k \sin\alpha + e_i e_j (1 - \cos\alpha). \tag{5.10}$$

In Eq. (5.10), $e_i = \frac{\omega_i}{\|\omega_j\|_2}$, where $\|\cdot\|_2$ is defined by Eq.(2.24), is the unit vector in the direction of the axis of rotation and $\alpha = \|\omega_j\|_2 \Delta t$. Equations (5.6) and (5.7) together with Eqs.(5.8)–(5.10), conclude the particle motion.

5.1.1 Collision strategy

When considering more than one particle or when a particle comes too close to a wall, a forcing term should be implemented to prevent particles from overlapping other particle domains or penetrating the domain walls. Simulating the particle collision in fully–resolved methods is the subject of profound research. A widely used method to consider particle–particle collision or particle–wall collision is to calculate the distance between each two pair of particles after the particle positions are updated. Then a type of short range repulsive force is calculated for the pth particle by

$$F_{p,i}^{C} = \sum_{\substack{q=1 \\ q\neq p}}^{N} F_{qp,i} + \sum_{w=1}^{N_w} F_{wp,i}, \tag{5.11}$$

where N_w is the number of domain walls. Superscript C is used to indicate the sum of all collision forces on particle p and subscript $\ell p, i$ is used to indicated the collision force in direction i exerted by object ℓ on particle p. Object ℓ can be another particle or a wall which are indicated by q and w respectively. There are a few different forms of potentials suggested for the modelling of the collision forces $F_{\ell p,i}$ on pth particle. However all these models have an adjustable parameter and can be presented by

$$F_{qp,i} = \xi_{qp} \left(X_{q,i} - X_{p,i}\right) \min\left(0, \frac{D_q + D_p}{2} + \Delta - d_{q,p}\right)^2, \tag{5.12}$$

where $d_{q,p}$ is the distance between particle pair (q, p) and Δ is the range of activation of the force [143, 195]. Particle–wall collisions are determined by creating a mirror image p' of the particle p on the other side of the wall, i.e $F_{wp,i} = F_{p'p,i}$. ξ_{qp} is the adjustable parameter and is the same for all pairs in this study. It is well established that the type of collision depends on the particle Stokes number, i.e $St = \frac{1}{9}\left(\frac{\rho_p}{\rho_f}\right) Re$, where Re is based on the particle diameter, collision velocity and the fluid properties [196]. Below a critical value of the Stokes number, $St_c \approx 10$, no rebound occurs since all the particle kinetic energy dissipates in the fluid phase. This corresponds to a collision with zero coefficient of restitution ($\epsilon_{\ell p}$). At larger Stokes number the particle rebound occurs which is equivalent to $\epsilon_{\ell p} > 0$ and the value of the coefficient depends on the Stokes number up to an upper critical Stokes number, where the value of $\epsilon_{\ell p}$ reaches a maximum value of $\epsilon_{\ell p}^{\max}$ equal to the dry collision coefficient of restitution. Value of $\epsilon_{\ell p}^{\max}$ only depends on the material properties [196].

Despite the easy implementation of Eq. (5.12), there are generally a few issues associated with it. Firstly, the adjustable parameter is problem dependent however a series of numerical experiments show that a value of $\xi = 2 \times 10^6$ can be used to simulate a large range of Stokes numbers which is in range of values ($[10^5, 10^7]$) used in other IBM [104] and FD simulations [143, 144]. Secondly this repulsive forcing function is continuous and its value grows as $O(d^2)$, where d is the separation distance defined in Eq. (5.12). Using this equation requires very small time steps such that the force increases continuously in a few steps otherwise the model produces a very sharp impulse which causes unrealistic rebound velocities and ultimately stability issues in the code. Due to the iterative nature of the current implementation, for large Re, required step sizes are prohibitively small. A remedy for this issue will be provided in Section 5.1.2. Finally the range of activation of the repulsive forces should be chosen carefully in the current FD method. The choice of Δ is restricted by the support of delta function used in this study. This restriction is less severe in the IBM and even in the FD methods that do not correct the rigidity constraint. This is due to the fact that if a fractional–step algorithm is used for the FD method the rigidity constraint will be imposed once, either before or after imposing the divergence–free constraint. This results in a field that is either precisely divergence–free or rigid but not both. In the IBM the internal velocity can assume arbitrary values as discussed in Chapter 3, and a slight overlap of the domains can be ignored without causing severe stability issue since the internal velocities can adapt accordingly to satisfy the mass balance. However in the current method both the rigidity constraint and the divergence free criteria are corrected and therefore both criteria are fulfilled to the required precision. Therefore slight overlap of the domains is equivalent to requiring the fulfilment of two different rigidity constraints at the same grid point which causes the failure of the iterative process. Therefore a value of $\Delta = 3h$ is used in this study.

5.1.2 Numerical implementation

Basic FD algorithm described in Section 3.4.1.2 can be modified as follows to include the particle motion. In Eq. (3.58), $U^R_{m,i}$ is calculated using Eq. (3.59) where $U_{p,i}$ and $\omega_{p,i}$ are calculated using the discretized form of Equations (5.6) and (5.7):

$$M_p U_{p,i} = \sum_{m=1}^{N} \rho_p \upsilon_m U^*_{m,i}, \tag{5.13}$$

$$I_{p,ij}\omega_{p,j} = \sum_{m=1}^{N} \rho_p \upsilon_m \epsilon_{ijk}(X_{m,j} - X_{p,j})U^*_{m,k}. \tag{5.14}$$

Consequently steps 6 and 7 of the algorithm described in Section 3.4.1.2 can be performed to update the Eulerian body force $f^*_{FD,i}$ using Eq. 3.58. Upon convergence the moment of inertia tensor is updated using Eq. (5.4) which can be discretized using the following equation,

$$I_{p,ij} = \sum_{m=1}^{N} \rho_p \upsilon_m (r_{m,k} r_{m,k} \delta_{ij} - r_{m,i} r_{m,j}), \quad r_{m,i} = X_{m,i} - X_{p,i}. \tag{5.15}$$

Rotation tensor is updated by Eq. (5.10) and body location by Eq. (5.8) which is discretized using a second order backward estimation assuming constant Δt by

$$X^{n+1}_{p,i} = X^n_{p,i} + \left(\frac{3}{2}U^n_{p,i} - \frac{1}{2}U^{n-1}_{p,i}\right)\Delta t. \tag{5.16}$$

Material positions are updated using discrete form of Eq. (5.9), which can be written as

$$X^{n+1}_{m,i} = X^{n+1}_{p,i} + R_{ij}(X^n_{m,i} - X^n_{p,i}). \tag{5.17}$$

As discussed in Section 5.1.1, to avoid very small time step sizes for a smooth increase in the repulsive force, the following procedure is used. First the time step is divided into a number of smaller sub–steps and the forcing term is calculated by first calculating the movement of each particle using Eq. (5.16) with Δt replaced by Δt_k, where $\Delta t_k = \frac{\Delta t}{N_k}$ and N_k is the number of sub–steps. Then total collision force for each particle is calculated using Eq. (5.11) and Eq. (5.12). The position of the particle is corrected using this force by

$$X^k_{p,i} = X^{k-1}_{p,i} + \frac{1}{4M_p}\left(F^{C,k}_{p,i} + F^{C,k+1}_{p,i}\right)\Delta t_k^2, \tag{5.18}$$

with $X^{n+1}_{p,i} = X^{N_k}_{p,i}$. Then the total amount of particle acceleration due to collision forces added in N_k sub–steps can be calculated by

$$A_{p,i}^{n+1} = \frac{2}{\Delta t^2}\left(X_{p,i}^{n+1} - X_{p,i}^{n} - \left(\frac{3}{2}U_{p,i}^{n} - \frac{1}{2}U_{p,i}^{n-1}\right)\Delta t\right). \tag{5.19}$$

The term $A_{p,i}^{n+1}\Delta t$ is then added to the velocities in the next time step after they are calculated by Eq. (5.13).

5.2 Results and Discussions

5.2.1 Single particle sedimentation

In this section the problem of single particle sedimentation in a slit is considered. A circular particle is released in an infinite channel with $W = 1.5D$ under the influence of gravity with $g = -981\frac{cm^2}{sec}$. A large domain is used to ensure that during the whole simulation the cylinder is always at least $8D$ away from the top and bottom walls which is enough to disregard the boundary effects on the problem [71, 173, 197]. A uniform 100×3000 grid with $h = 0.0075$ is used on the whole domain. No-slip boundary conditions are used for the vertical walls and normal derivative is set to zero on the top and bottom walls with fluid density set to $\rho_f = 1$ and $D = 0.5$. Feng et al. [164] did not provide enough physical parameters, however numerical experiments show that setting $\rho_p/\rho_f = 1.3$ and $\mu_f = 0.1157$ will result in $Re_T = 6.09$ where Re_T is based on the terminal velocity and diameter. Two other Reynolds numbers $Re_T = 2.05$ and 4.02 are also considered by setting $\rho_p/\rho_f = 1.1$ and 1.2 respectively. The cylinder is released near the left wall at $y_0 = 0.6D$.

For an unsteady simulation the time history of the problem is important and no unique suggestion can be given on the number of iterations per time step. However it seems that using optimal combination of under-relaxation factors (i.e. $\alpha_{p'} = 1-\alpha_u$, [7]) and setting the momentum and heat source correction under-relaxation factors (i.e $\alpha_{f_{FD}}$ and $\alpha_{T_{FD}}$) to those used for the momentum and temperature equations, and letting the iterations continue to the required tolerance in each time step provides optimal computational times. About 10-20 iterations are enough for convergence with a Bi-CGSTAB (see Appendix A) solver for both the pressure correction and the transport equations with the aforementioned under-relaxation factors.

Figure 5.1 shows the plot of the non-dimensional y-position of the centre of the cylinder with respect to its x-position. At higher Reynolds numbers a high strain

region between the particle and the wall pushes the cylinder toward the other wall and cylinder gains enough momentum to pass the centre-line. This cause an oscillatory motion around the centre line. This oscillation is damped gradually and the cylinder continues to settle on the centre line. However at $Re = 2.05$ since the particle does not accelerate rapidly, it will not gain enough momentum toward the other wall to pass the centre-line and a smooth transition is observed. In Figure 5.1, Feng et al. [164] results at $Re = 6.28$ are also presented. Small under-estimation of the first peak and slightly faster damping can be related to the marginal difference in the Re_T. Other two simulations are also in line with the results provided by Feng et al. [164] however the calculations are not performed at exactly the same Re_T values.

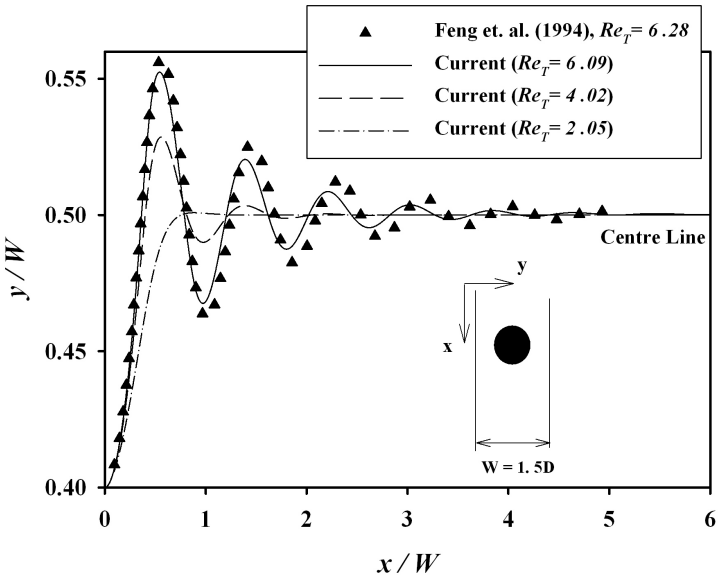


Figure 5.1: Non-dimensional horizontal position of the centre of the cylinder versus its vertical position - cylinder is falling in a slit at different Reynolds numbers. $\rho_p/\rho_f = \{1.1, 1.2, 1.3\}$ corresponding to $Re_T = \{2.05, 4.02, 6.09\}$

5.2.2 Particle–particle collision

In this section sedimentation of two particles in a 2D channel filled with a Newtonian fluid is studied. The problem is equivalent to the one initially studied by Glowinski et al. [143, 144]. A channel with height 6 and width 2 is considered and particles are released at $y = 5$ and $y = 5.5$ with zero velocity and angular momentum. Density

ratio, gravitational acceleration and fluid viscosity are set to $\rho_p/\rho_f = 1.5$, $g = -981$ and $\mu_f = 0.01$ respectively. Particles have a diameter of $D = 0.25$ and the Reynolds number based on the fluid properties and maximum velocity of the trailing particle is 504 and collision Stokes number is 84. Glowinski et al. [143] placed the cylinder exactly at the centre of the channel however with this initial arrangement of the particles no tumbling is observed in the current simulation. It seems since in the current method a uniform structured grid is used the symmetry is preserved and instabilities do not grow large enough in a small contact period. Uhlmann [104] studied the same problem and compared the results to that of Glowinski et al. [144] and reported the same problem. Uhlmann [104] argued that the faster growth of instabilities in the method proposed in [143, 144] is due to the usage of an anisotropic triangular grid. Therefore the particles are initially displaced for $0.004D$ and the centres are not on a vertical line. Figure 5.2a shows the vertical velocity of the particles. Initially both particles start accelerating downwards, at around $t = 0.1$ the wake generated by the leading particle traps the trailing particle and due to the smaller drag the trailing particle accelerates faster and collision occur at around $t = 0.18$. After the collision tumbling occurs which is a manifestation of large instability of the state of the particles. Therefore up to the collision point both vertical velocities are in very good agreement, however after the collision the curves can only be compared qualitatively. Significant deviation of the after collision curves, provided by different numerical methods, is also reported in [152] and is similarly related to the large sensitivity in the positioning of the particles in this stage of motion. However later in this section it is shown that the collision strategy used in this study is indeed grid and time independent. It is also worth mentioning that Uhlmann [104], collision starts slightly earlier which might be due to the particle movement strategy used in this study (Section 5.1.2) which allowed using step size which are 10 times larger than those used by Uhlmann [104]. Figure 5.2b shows the lateral velocity of the particle which is again qualitatively similar to the results reported by earlier studies.

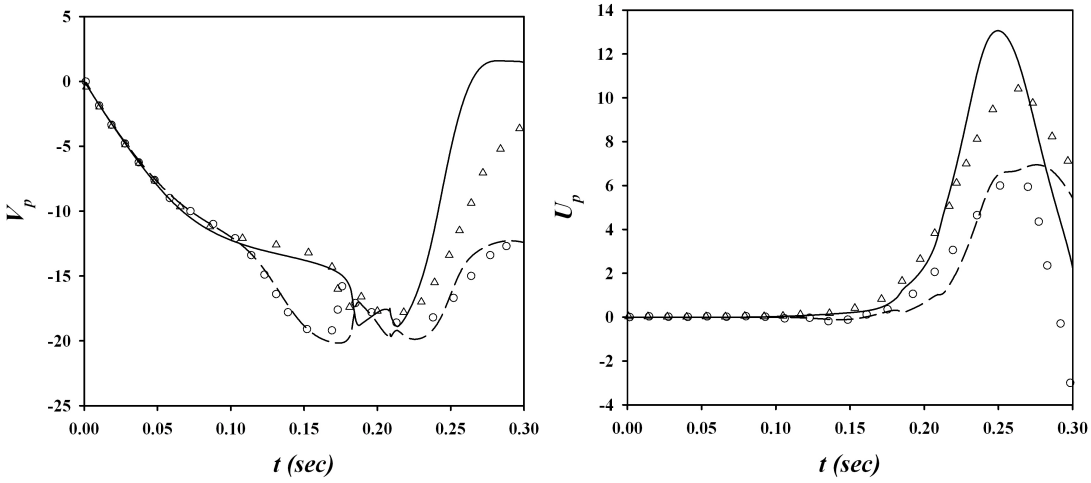
Figure 5.2c and Figure 5.2d show the vertical and lateral position of the leading and trailing particles. After the collision at around $t = 0.26$ the trailing particle actually overtakes the leading particle which is evident in Figure 5.2c where the two graphs intersect. Similar behavior is evident in the results reported by Uhlmann [104] but at a slightly later time around $t = 0.28$. The angular velocity of the two particles is

plotted in Figure 5.3. The angular velocity of the leading particle after the collision is predicted to be larger than those predicted by Uhlmann [104]. This large counter–clockwise angular velocity produces an upward flow which causes the vertical velocity of the leading particle to reduce faster compared to the results reported by Uhlmann [104]. Figure 5.4 shows the vector plots of the three stages of the motion of the two–particle system. $t = 0.2$ corresponds to the second time the particles collide during the course of motion and finally separate at $t = 0.225$ (see also Figure 5.2a).

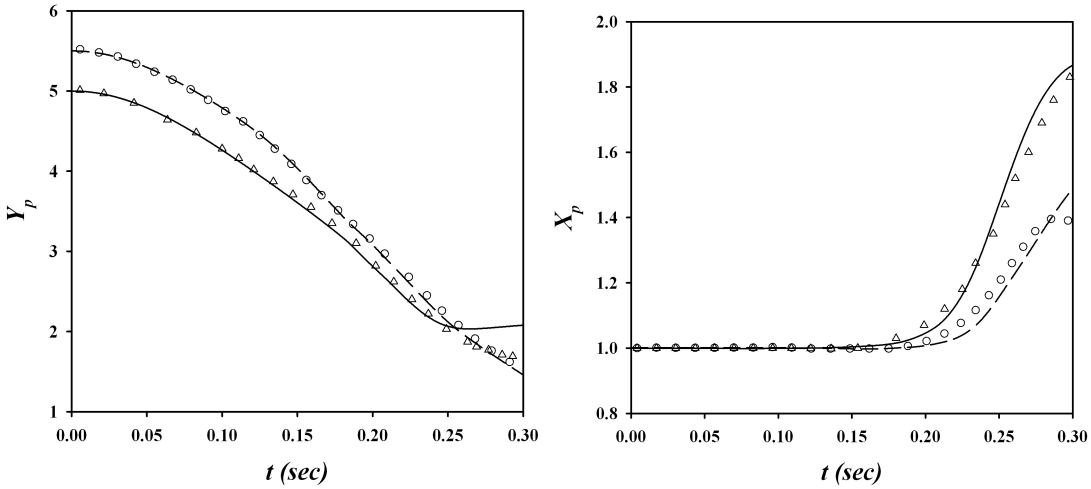
Sedimentation of two particles at lower Stokes number is considered next. The domain size is the same as the previous case however the particle diameters are reduced to $D = 0.2$ and are released at $y = 0.5$ and $y = 5.3$. Density ratio, gravitational acceleration and fluid viscosity are set to $\rho_p/\rho_f = 1.1$, $g = -981$ and $\mu_f = 0.08$ respectively. The Reynolds number based on the fluid properties and maximum velocity of the trailing particle is 9.68 and the corresponding Stokes number is 1.18. Figure 5.5 shows the vector plot of the different stages of the motion. Obviously the particles move much slower than the previous case and remain in contact for a much longer period before the complete separation occur. Figure 5.6 summarizes the motion of the two particles during the course of motion. First particle–particle collision happens at $t \approx 0.4$ and here due to the small Stokes number no rebound should occur. Figure 5.6a shows that as expected vertical velocities of the two particles reach the same values during the collision at $t \approx 0.4$. Overtaking the trailing particle happens similar to the previous larger Stokes number case which happens at $t = 0.81$ and can be identified from the intersection of the two graphs in Figure 5.6d. The base simulation is performed on a grid with uniform spacing, $h = 1/256$, and a time step size of $\Delta t = 0.002$. To examine the time and grid independence of the collision strategy another smaller grid with $h = 1/192$ and smaller time step of 0.001 are used. In Figure 5.6 results of all these simulations are also summarized. This shows that particle motion even after the collision is indeed time and grid independent and only depends on the order of the accuracy of the method used.

5.2.3 Sedimentation of a single isothermal cold particle in a vertical channel

In this section the problem of a single isothermal cold particle falling in a hot channel is considered. Gan et al. [71] studied the problem using an ALE technique where a



(a) Vertical velocity **(b)** Lateral velocity



(c) Vertical position **(d)** Lateral position

Figure 5.2: Velocity and position of two colliding particles for $St_p = 84$ - Velocity and position of the two particles compared to the results reported in [104]. — leading particle (current study), --- Trailing particle (current results), △ leading particle [104], ○ Trailing particle [104]

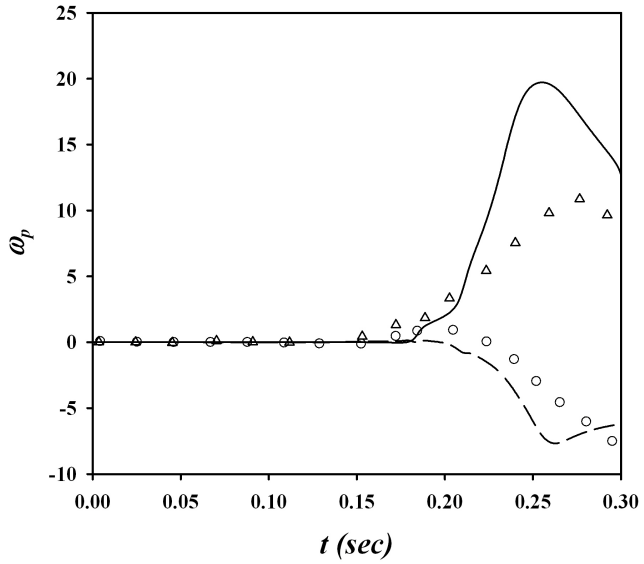
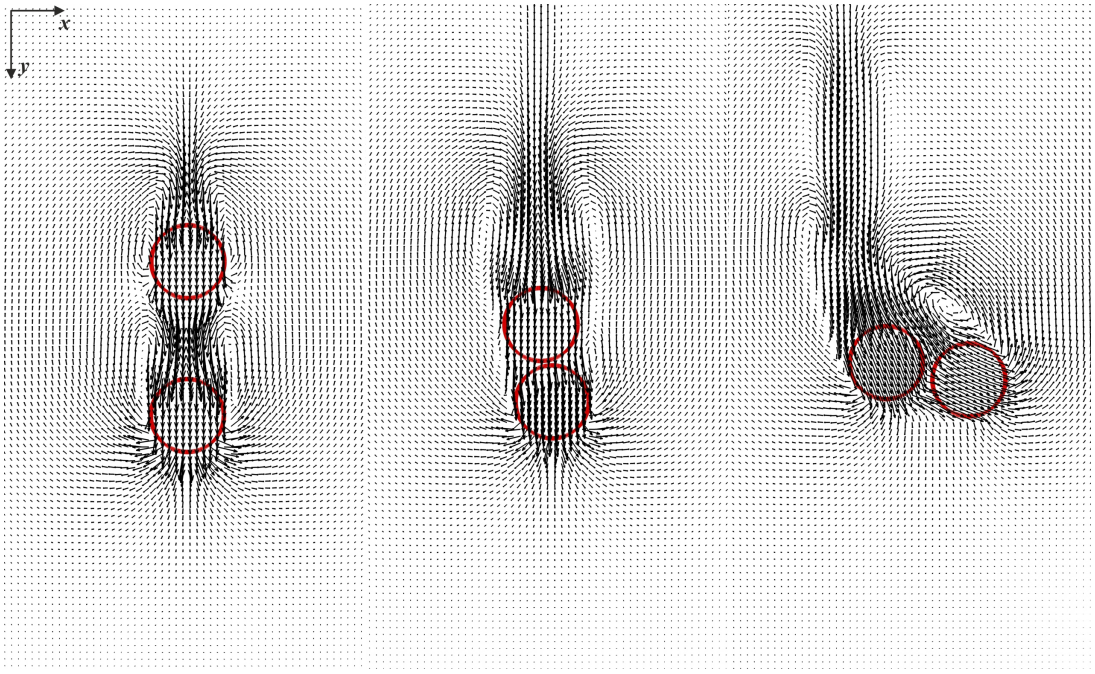
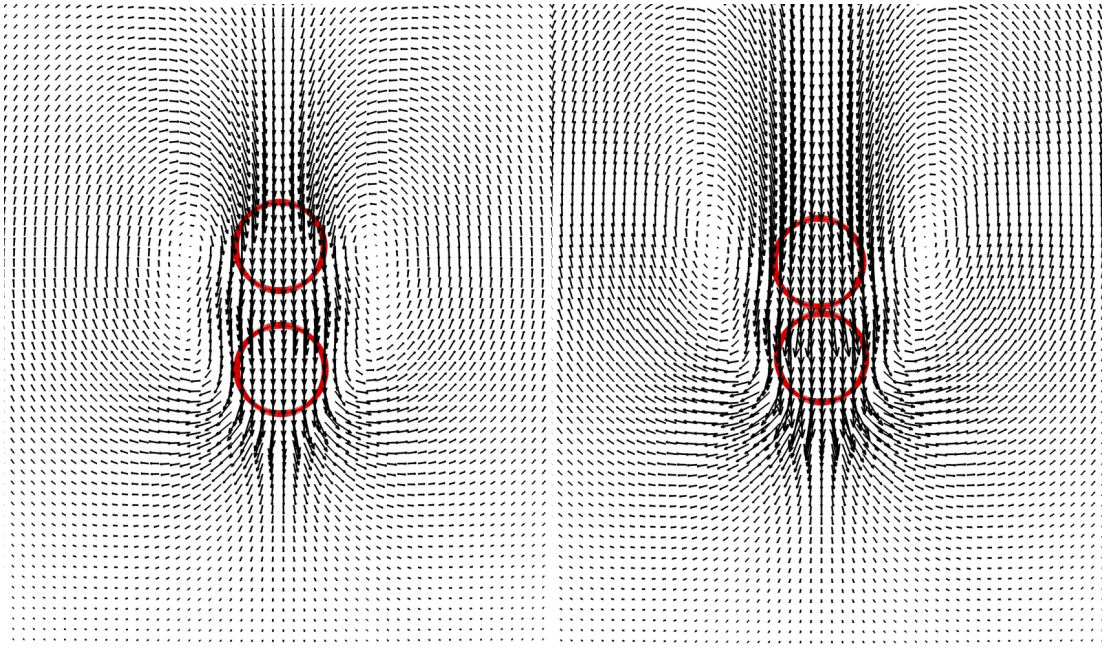


Figure 5.3: Angular velocity of the two particles - see the caption of Figure 5.2.

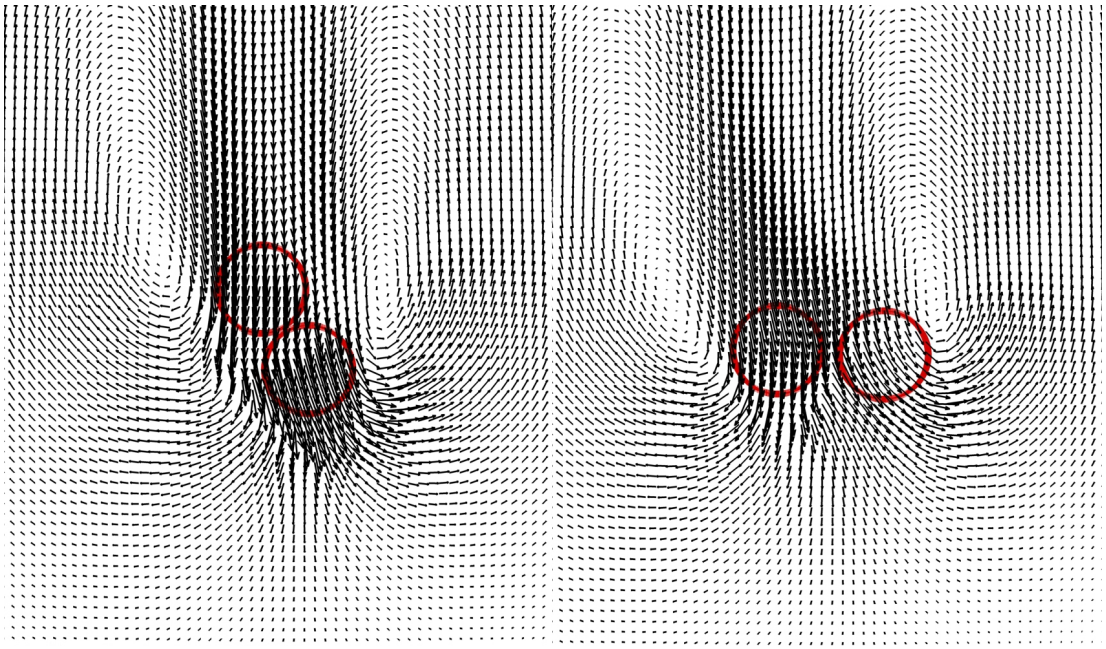


(a) $t = 0.1$ **(b)** $t = 0.2$ **(c)** $t = 0.225$

Figure 5.4: Vector plots of two colliding particles for $St_p = 84$ - Three different stages of drafting, kissing and tumbling are presented at $t = 0.1$, $t = 0.2$ and $t = 0.225$ respectively.

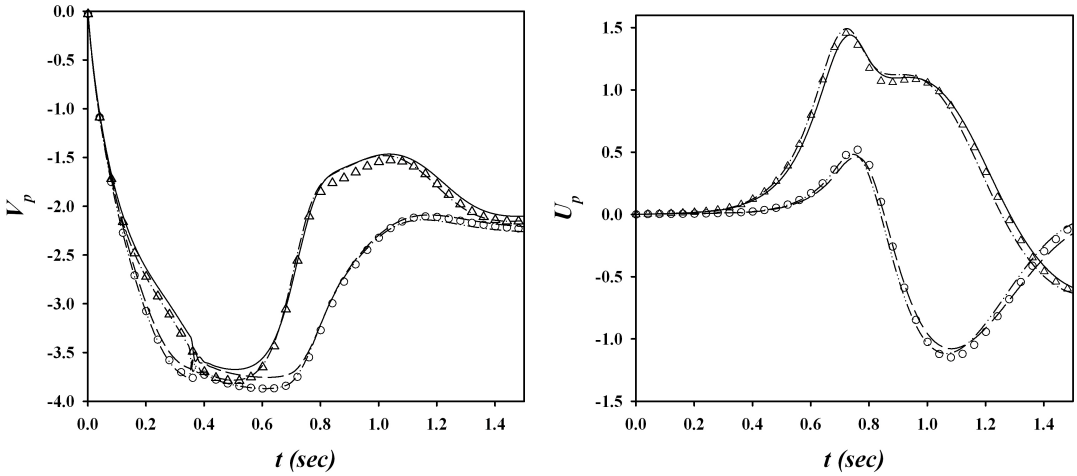


(a) $t = 0.2$ **(b)** $t = 0.35$

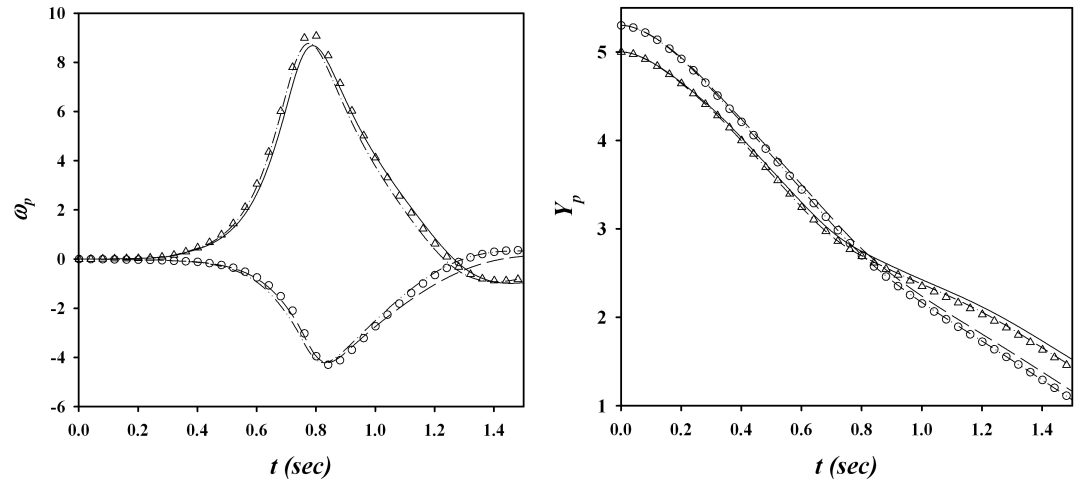


(c) $t = 0.65$ **(d)** $t = 0.8$

Figure 5.5: Vector plots of two colliding particles for $St_p = 1.18$ - Three different stages of drafting, kissing and tumbling are presented for a small Stokes number case.



(a) Vertical velocity **(b)** Lateral velocity



(c) Angular velocity **(d)** Vertical position

Figure 5.6: Linear and angular velocity and vertical position of the two particles at $St_p = 1.18$ - Time and grid independence of the collision strategy is tested. — leading particle ($\Delta t = 0.002$, $h = 1/256$), --- Trailing particle ($\Delta t = 0.002$, $h = 1/256$), -·-·- leading particle ($\Delta t = 0.002$, $h = 1/192$), -···- Trailing particle ($\Delta t = 0.002$, $h = 1/192$), $\triangle$ leading particle ($\Delta t = 0.001$, $h = 1/256$), $\circ$ Trailing particle ($\Delta t = 0.001$, $h = 1/256$)

body conformal mesh it is worth mentioning here that although the body fitted grids are assumed more accurate in the case of stationary objects, they are not necessarily more accurate in case of moving object due to frequent remeshing and interpolation of the old solution to the new grid points which rapidly deteriorates the accuracy, see Section 3.1. In this section following Yu et al. [173] a velocity scale is defined viz

$$U_c = \sqrt{\pi D_p/2(\rho_p/\rho_f - 1)g}, \tag{5.20}$$

as the characteristic velocity which allows convenient calculation of the drag coefficient by $C_D = (U_c/U_T)^2$. In this case no buoyancy effects are considered inside the particle domain i.e. $\beta_p = 0$. The set of non-dimensional numbers for this problem consists of Gr, Re and Pr. Re is based on the particle diameter and the characteristic velocity U_c, Grashof number is defined by

$$Gr = \frac{g\beta_f(T_p - T_\infty)D_p^3\rho_f^2}{\mu_f^2}, \tag{5.21}$$

where where T_∞ is the far field hot fluid temperature. It is useful to define another Reynolds number based on the terminal particle velocity which can be written by $Re_T = ReU_T/U_c$. In the following simulations time step size, particle diameter, density ratio and Prandtl number are set to $\Delta t = 5 \times 10^{-3}$, $D_p = 1$, $\rho_p/\rho_f = 1.00232$ and $Pr = 0.7$ respectively which are chosen according to Gan et al. [71].

A long domain with a width of $4D_p$ is considered such that the particle is always more that $8D_p$ away from the top and bottom walls during the simulation which is enough to ensure boundary independence of the results [71, 173, 197]. The particle is released $0.5D_p$ away from the centre-line and mesh spacing is set to $h = 1/32$ which is equivalent to 32 Eulerian grid points inside the particle domain. Gan et al. [71] only mentioned that they performed the calculation at $Re_T = 21$ for $Gr = 0$, this information is not enough to identify all the physical parameters however several numerical experiments show that setting $Re = 40.5$ yields $Re_T = 21.1$ which agrees to $Re_T = 21.2$ reported by Yu et al. [173] for the same Re. Gan et al. [71] identified five regions depending on the Gr number:

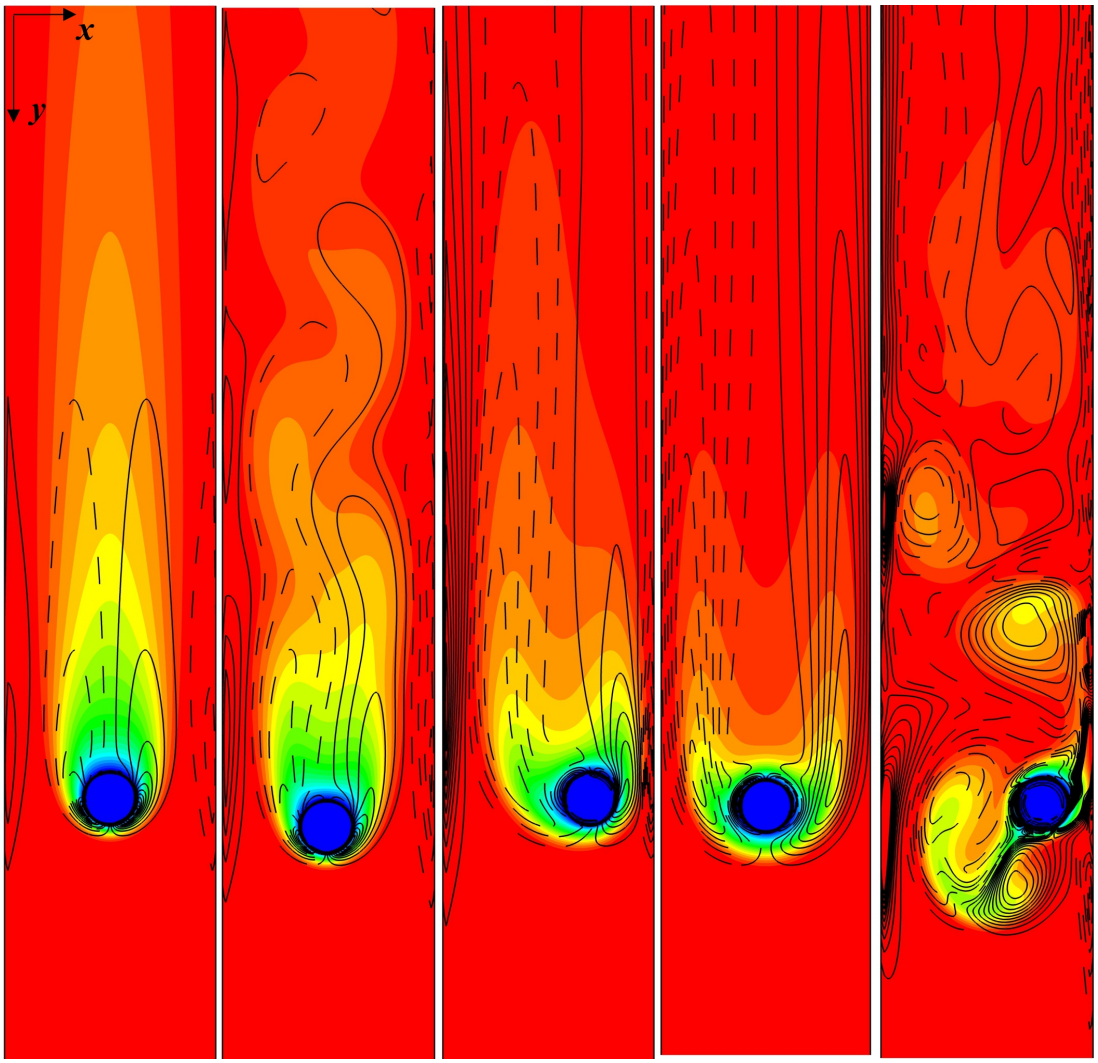
- $0 < Gr < 500$: Particle settles along the centre-line with a symmetric and steady wake.

- $500 < Gr < 810$: Periodic vortex shedding is observed at the rear of cylinder which causes regular oscillations around the centre-line.
- $810 < Gr < 2150$: Particle steadily settles off the centre-line near one of the walls.
- $2150 < Gr < 4500$: Particle once again reaches an equilibrium on the centre-line with symmetric wakes.
- $4500 < Gr$: Large amplitude oscillation of the particle with turbulent like vortex shedding which was ascribed by the the authors to the Kelvin-Helmholtz instability.

Later Yu et al. [173] observed a periodic vortex shedding and oscillation around the centre-line for $3500 < Gr < 4300$ and large amplitude oscillations for $4300 < Gr$ and consequently 6 regions were identified. Figure 5.7 shows the different regimes simulated using the current method. Similar to Gan et al. [71] large scale oscillations are only observed for $Gr > 4500$ and symmetric wakes were observed for $2150 < Gr < 4500$. Figure 5.8 shows the time history of the lateral distance of the centre of the particle to the left wall. At $Gr = 100$ a smooth transition to the centre-line and at $Gr = 564$ and $Gr = 6000$ periodic oscillation around the centre-line is correctly simulated. For $Gr = 1000$ and $Gr = 2000$ particle settles near the right wall, however for $Gr = 1000$ particle periodically approaches the wall and it takes much longer for the particle to reach the stationary state. For $Gr = 3500$ some initial oscillations around the centre-line is observed which are damped quite quickly and the particle settles on the centre line. Similar behavior is also observed for Grashof numbers up to 4500. Figure 5.9 compares the value of the Re_T at different values of Gr with previous studies where satisfactory agreement is observed given the complicated nature of the problem and that different computational procedures are used. Also it is worth mentioning that the values of Re_T at zero Grashof number are slightly different.

5.2.4 Motion of a single catalyst particle in a cavity

In the previous section motion of a single isothermal particle was studied in a long channel. The main concern in Section 5.2.3 was to study the long term behavior of a single catalyst particle and the types of instabilities that can arise in the presence of natural convection from a cold particle. In a real applications however, the particle will



(a) $Gr = 100$ **(b)** $Gr = 564$ **(c)** $Gr = 2000$ **(d)** $Gr = 3500$ **(e)** $Gr = 6000$

Figure 5.7: Sedimentation of a single cold particle in a vertical channel at different Grashof numbers - Vorticity contours in z-direction, ω_z, superimposed on the temperature contours for different Grashof numbers. Similar to Gan et al. [71] only 5 different regimes are observed in this range of Grashof numbers.

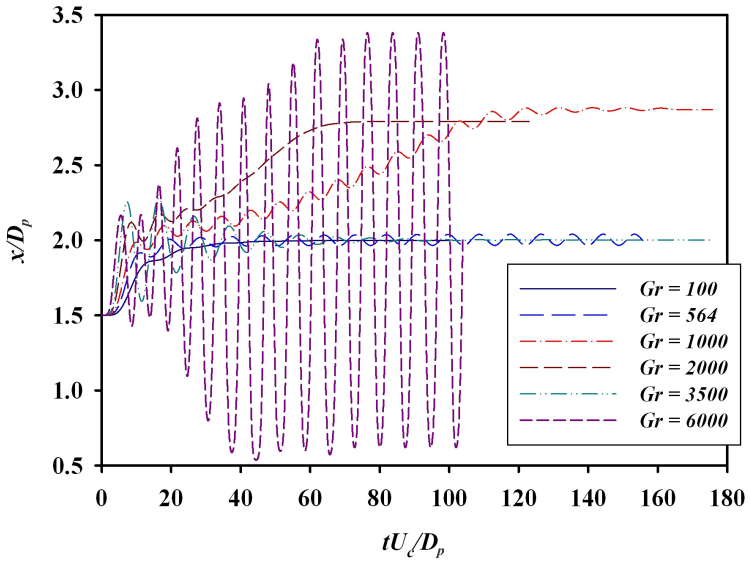


Figure 5.8: Time history of the particle lateral position - Lateral distance of the centre of the particle to the left wall for different Grashof numbers is plotted against the non–dimensional time tU_c/D_p.

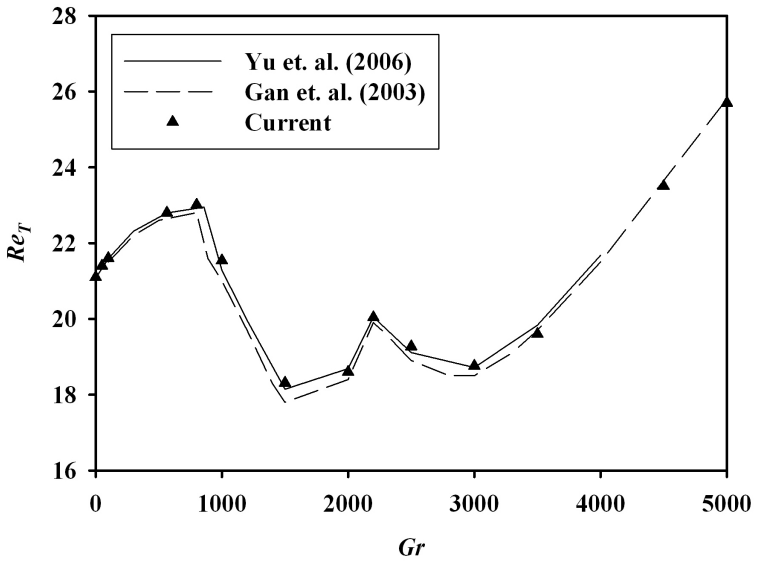


Figure 5.9: Comparison of the Re_T with previous studies - The value of Re_T is compared to the results reported by Gan et al. [71] and Yu et al. [173] for different values of Gr. Although Yu et al. [173] observed different regimes for $3500 < Gr < 4300$ it seems it does not have any significant effect on the terminal velocity.

eventually attain the temperature of the fluid and any instability induced by natural convection will diminish in a long run. In this section the motion of a single catalyst particle moving in an enclosure will be discussed. This problem is important both from a numerical and a practical point of view. The problem is important numerically since in this section the temperature of the particle freely varies and is time dependent and this indicates the true benefit of the current heat transfer formulation which allows simple treatment of conjugate phenomena.

It is assumed that heat is generated homogeneously inside the particle at constant rate for example due to some reactions taking place inside the particle domain. Similar to the cold particle case $\beta_p = 0$ but Gr is defined by temperature difference $(T_{p_m} - T_0)$ where T_{p_m} is another temperature reference, for example maximum particle temperature during the simulation, and T_0 is the initial temperature. However since T_{p_m} is not known a priori $(T_{p_m} - T_0) = 1$ is used for the simulations in this section. The only effect of this choice is that the final temperature distribution will not be normalized. Similarly Eq. (5.20) is used as the characteristic velocity. Other parameters are set to $Re = 40$, $Pr = 0.7$, $Gr = 1000$, $\rho_p/\rho_f = 1.1$ and $q_p = 12.414$. Note that q_p is dimensional in this study however a non-dimensional value $\overline{q}_p$ can be defined by

$$\overline{q}_p = \frac{q_p D}{\rho_f C_{pf} U_c (T_{p_m} - T_0)}. \tag{5.22}$$

Both the fluid–particle C_p and κ ratios are important in these simulation and hence a heat capacity ration $C_{pr} = C_{pp}/C_{pf}$ and a conductivity ratio $\kappa_r = \kappa_p/\kappa_f$ are defined. Computational domain is a large $8D \times 16D$ box where $D = 1$ and 32 node are placed along the particle diameter except for grid independence studies. $\Delta t = 2 \times 10^{-3}$ is chosen for all test cases except when performing time independence simulations. For the first simulation heat capacity ration is set to $C_{pr} = 1$ and $\kappa_r = 5$ to compare our results to those presented by Yu et al. [173]. It is worth mentioning that similar simulations are also performed by Wachs [175] however they assumed a homogeneous temperature inside the domain. Figure 5.10 compares the evolution of the non-dimensional velocity in time. Simulation is performed using two different Δt and three grid levels to assess the time and grid independence of the simulations. Figure 5.10 shows that the results with $h = 1/32$ and $\Delta t = 2 \times 10^{-3}$ are satisfactory with at most 5% error in calculating the time to reach the maximum velocity. This accuracy is enough for the rest of simulations

and discussions in this section and hence the computational time is saved by using these parameters which are similar to the values suggested by Wachs [175]. In this figure the same simulation using a very large density ratio of $\rho_p/\rho_f = 100$ is also presented with other parameters being the same. Unlike the method proposed by Yu et al. [173] no special treatment is required for this simulation and the method converges without any issues.

The physics of the problem is presented in Figure 5.11. Initially particle with higher density starts falling in the fluid due to the buoyancy forces while the temperature increases inside the particle domain due to the presence of a constant source term. Heat transfer to the surrounding fluid changes the fluid density which in turn induces a buoyancy-driven upward flow, Figure 5.11a, in the fluid domain. Although particle buoyant forces increase because of the decreasing fluid density, upward buoyancy-driven flow due to the temperature difference in the fluid eventually dominates the particle buoyant force and starts to lift the particle, Figure 5.11b. Particle gains a significant amount of momentum initially but by getting closer to the upper wall the strength of the upward wake quickly diminishes and particle never collides the upper wall, Figure 5.11c. Figure 5.11 shows that the upward velocity vanishes near the top wall at a non-dimensional time $tU_c/D = 21$ and particle starts falling. When the particle moves away from the wall, the upward lift forces strengthen once again and the particle moves upward which causes a mild oscillation near the top wall. Figure 5.12 shows the same simulation at different heat capacity ratios. At lower C_{pr} the particle heats much faster and consequently the buoyancy driven flow forms in a shorter time which in turn causes the particle to reach its maximum velocities faster.

Several tests are performed to show how the particle trajectory changes by increasing the internal heat generation rate q_p. Five values of $\overline{q}_p = 1 \cdots 5$ are chosen for this case with constant $C_{pr} = 0.2$ and $\kappa_r = 5$. Figure 5.13 show the evolution of the particle velocity and temperature in time. Velocity curves are similar to those in Figure 5.12 for $\overline{q}_p = 1$ however at larger values of $\overline{q}_p$ the trajectory of the particle changes markedly. A local maximum in the trajectory curve happens early in the motion of the particle for $\overline{q}_p \geq 2$. The velocity is then reduced for a short interval and increases rapidly to another local maximum, (see Figure 5.13a) after which it falls rapidly to zero. This behavior can easily be interpreted using the mean temperature (volume average) of the particle defined by $\overline{T}_p = \sum_{m=1}^{N} T_m \upsilon_m / \mathcal{V}_p$.

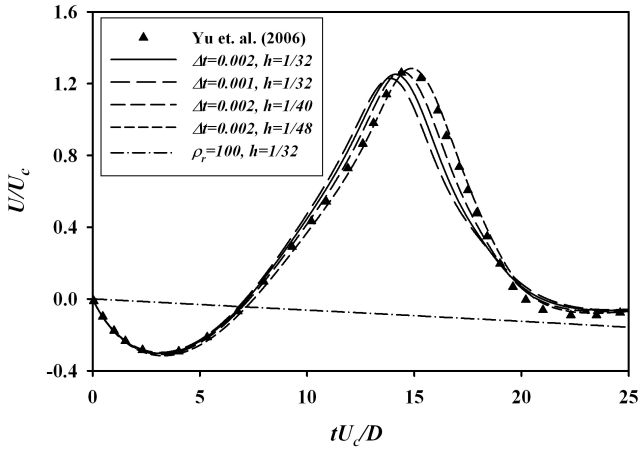
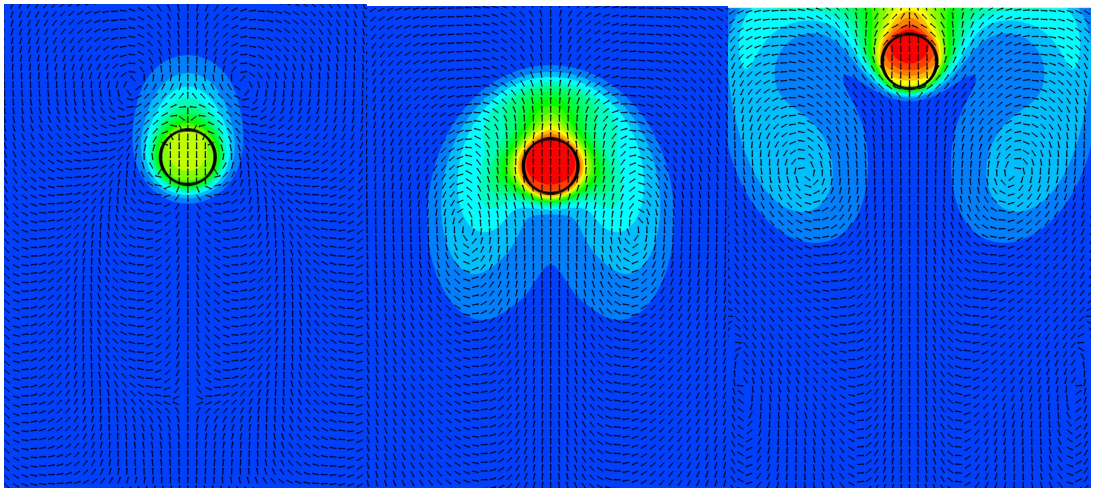


Figure 5.10: Evolution of the non-dimensional velocity of the particle in time for $C_{pr} = 1$ and $\kappa_r = 5$ - Particle upward velocity vanishes near the top wall because of the weakening of the wake responsible for lifting the particle due to the wall effects. $\rho_p/\rho_f = 1.1$ is used for time and grid independence studies. $\rho_p/\rho_f = 100$ is used to show that the method can easily handle large density ratios.



(a) $tU_c/D = 5$ **(b)** $tU_c/D = 15$ **(c)** $tU_c/D = 20$

Figure 5.11: Velocity vector plots (with uniform size) superimposed on the temperature contours - Results on the fine grid with $h = 1/48$, $C_{pr} = 1$ and $\kappa_r = 5$. Downward motion at $tU_c/D = 5$ and appearance of a pair of vortices followed by an upward motion at $tU_c/D = 15$. Finally particle halts at $tU_c/D \approx 20$.

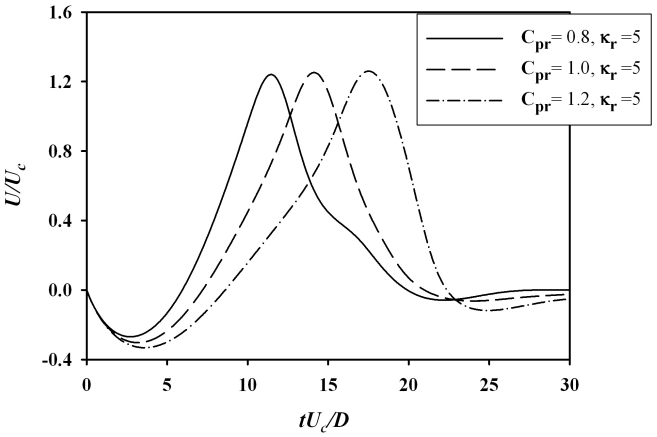
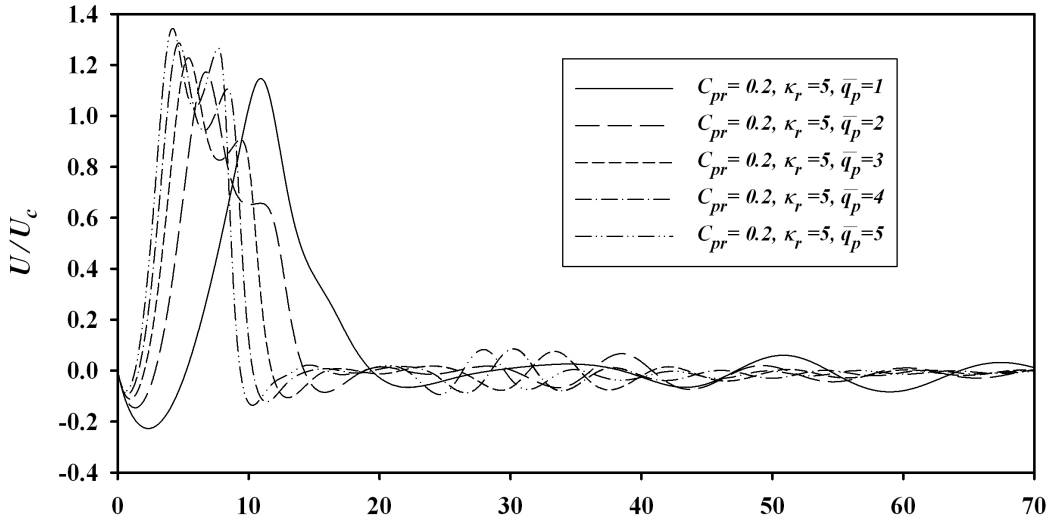


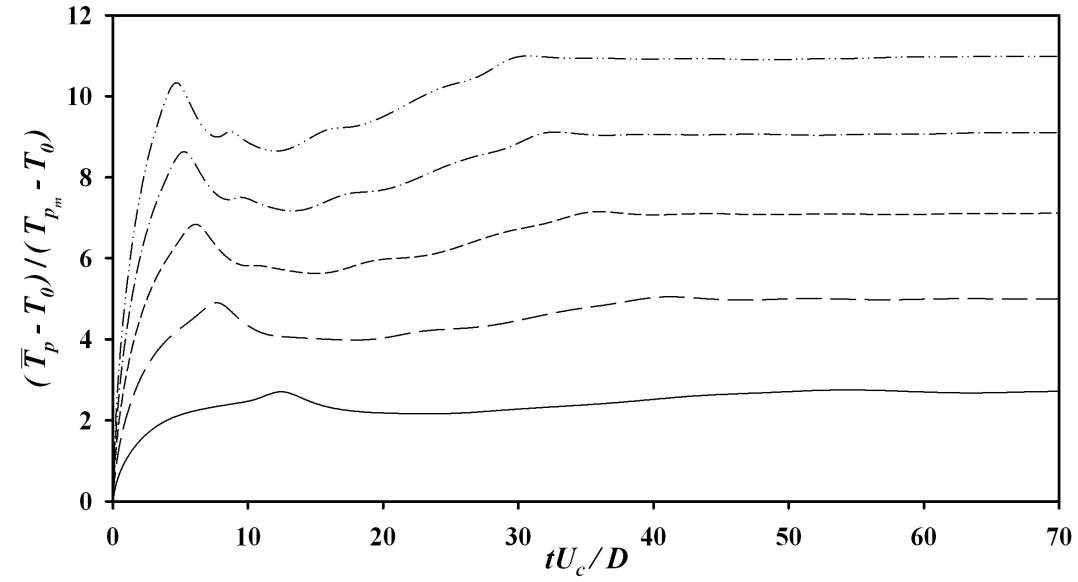
Figure 5.12: Evolution of the non-dimensional velocity of the catalyst particle in time - for the lower values of C_{pr} the rigid particle heats faster and consequently the buoyancy–driven flow forms faster and the particle reaches its maximum velocity in a shorter time.

The evolution of the non–dimensional mean temperature of the particle is plotted in Figure 5.13b. Due to a large internal heat generation and small heat capacity ratios, the temperature inside the particle increases rapidly which consequently induces a large upward current in the domain. Particle rapid acceleration in the colder surrounding fluid results in a temperature drop in the particle domain and consequently deteriorates the upward current momentarily. However due to lower velocities internal temperature increases once again which strengthens the buoyancy–driven flow and particle accelerated upward until it reaches the top wall. As explained earlier, a similar mechanism produces an oscillatory motion near the top wall. In Figure 5.13a the velocity evolution for a long non–dimensional time $tU_c/D = 70$ is plotted. Evidently the frequency of oscillations near the top wall is much higher for larger $\overline{q}_p$ and reduces by reducing $\overline{q}_p$.

It is also interesting to test whether there is a limiting value for $\overline{q}_p$ to lift the particle to the top wall. Six more simulation were performed reducing the $\overline{q}_p$ from 0.9 to 0.4. Up to a $\overline{q}_p = 0.5$ similar behavior is observed, i.e the particle accelerates and reaches the top wall producing a maximum in the velocity curve, although the position of the maximum is deferred. Figure 5.14a shows this behavior for $\overline{q}_p = 0.5$. However for smaller values of $\overline{q}_p$ no significant maximum can be observed, see Figure 5.14a. Particle initially falls due to gravitational forces, but eventually buoyancy–driven flows start to



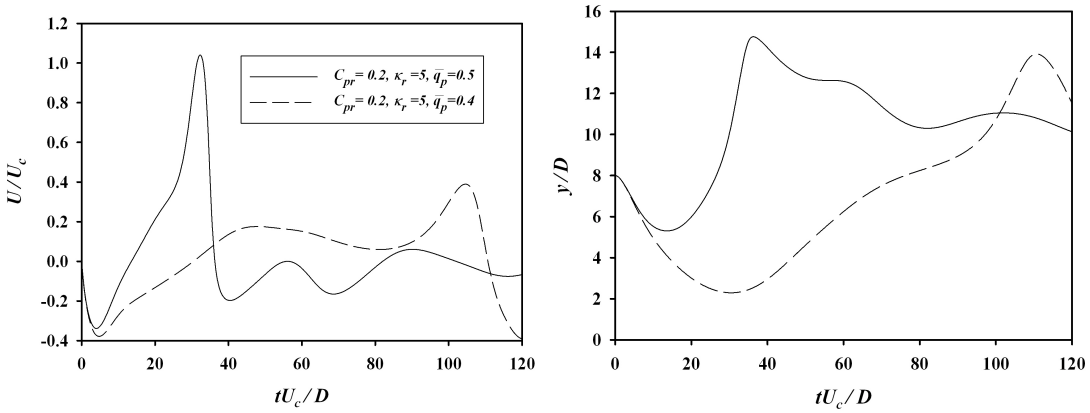
(a) Particle Velocity



(b) Particle Temperature

Figure 5.13: Evolution of the non-dimensional velocity and mean particle Temperature in time. - Due to rapid acceleration of the particle another maximum can be observed in the velocity graphs for $\overline{q}_p \geq 2$.

build up and the velocity increases smoothly and remains constant for a large period of time. Figure 5.14b shows that the particle will eventually reach the top wall but the large velocity maximum does not characterize the motion anymore.



(a) Particle Velocity **(b)** Particle position

Figure 5.14: Evolution of the non-dimensional velocity and particle position in time. - particle behavior changed markedly for $\overline{q}_p < 0.5$.

5.2.5 Flow around a stationary sphere

Flow around a stationary sphere is studied in this section to test the basic 3D implementation of the proposed algorithm. Reynolds number of up to $Re = 150$ are considered and key flow parameters are compared to previous numerical studies. In this range of Reynolds number the flow remains steady and axisymmetric [198]. Pressure and momentum under-relaxation factors used in this simulation are $a_p = 0.2$ and $a_u = 0.8$ respectively. BiCGSTAB solver with SIP preconditioner is used for the solution of both momentum and pressure correction equations which provides acceptable performance given the size of the problem. Maximum number of iteration per time step is set to 15 to attain a tolerance of $5e-7$ for the mass balance. Maximum solver iterations is set to 5 for the momentum equations and 20 for the pressure correction equation with a tolerance of $1e-7$ and a backward error analysis similar to one discussed in Section 2.7.3 is used to detect solver convergence.

A stair–step type material grid point is used and the width of the material control

volume to the background fixed mesh is set to $\frac{h}{h_m} = 4$ for all simulations. Fluid and particle densities are set to $\rho_f = 1000$ and ρ_p is irrelevant in fixed sphere calculations. The number of surface elements used for force calculation is 5120 and the sphere diameter is set to 0.1. With this configuration sphere volume is 5.24×10^{-4} and the discretized volume is 5.25×10^{-4} with relative error of 0.2%. Fluid viscosity is used to control the Reynolds number with far field fluid velocity set to $u_\infty = 1$.

A large domain with sides $15D_p \times 15D_p \times 15D_p$ similar to those used by Mittal et al. [122] and later Apte et al. [151] is used to remove the effects of the walls on the flows around the sphere. The finest mesh consists of 1.22048×10^6 elements with $h = 0.01$ around the object which gives about 10 Eulerian cells along the diameter of the sphere. Standard inlet and outlet boundary conditions are used in the direction of the flow and slip wall used for the other boundaries. Inlet velocity is set to $u_\infty = 1$ and inside the domain is initialized to the same velocity. A grid refinement strategy is also used to assess the accuracy of the total stress tensor interpolation in 3D, see Section 3.3.1.1.

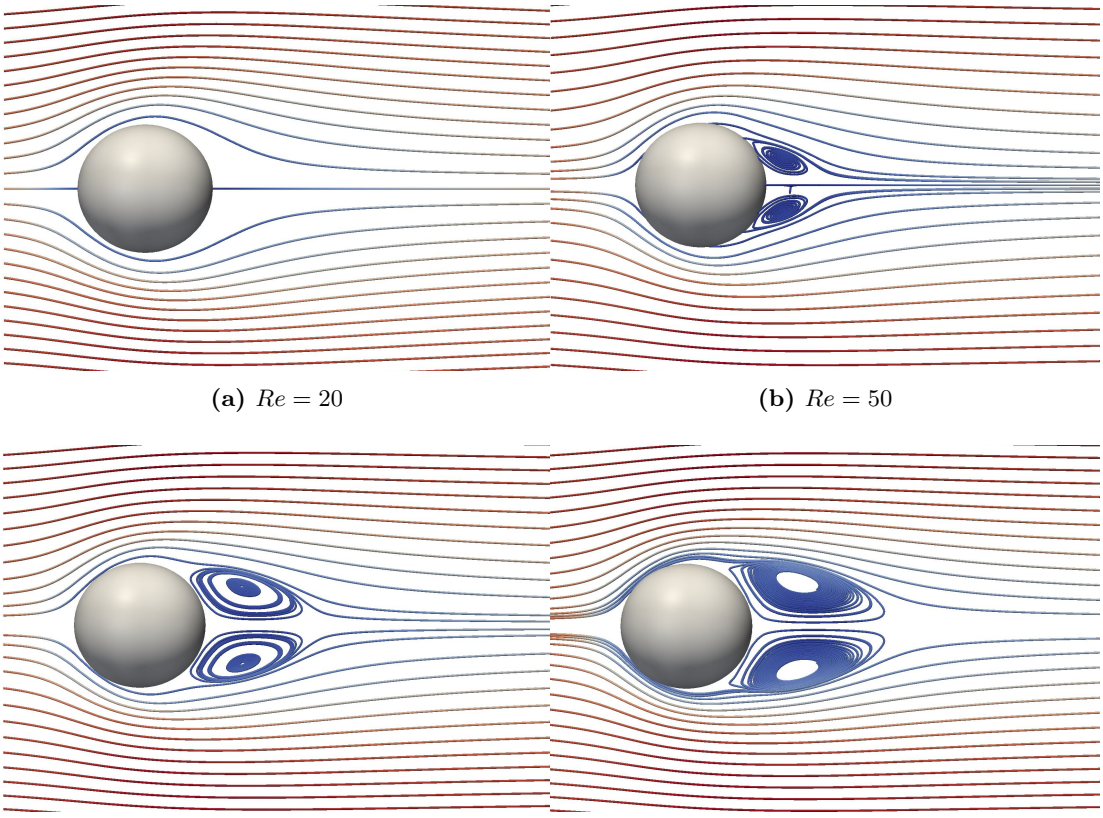
Figure 5.15 shows the streamlines around the sphere at the four simulated Reynolds numbers. Evidently the wake bubble forms around the sphere at $Re = 50$ and no bubble is formed at lower Reynolds numbers. Key parameters in this flow are the location and the size of the bubble and the Drag coefficient which is defined by

$$C_D = \frac{8F_D}{\rho_f u_\infty^2 \pi D^2}. \tag{5.23}$$

Table 5.1 compares the key wake bubble properties. These include the location of the centre of the bubble, $(x_c/D, y_c/D)$, in addition to the length of the bubble L_w. Good agreement is observed for all parameters for the three Reynolds numbers considered.

Re	50			100			150		
	x_c/D	y_c/D	L_w/D	x_c/D	y_c/D	L_w/D	x_c/D	y_c/D	L_w/D
Current	0.60	0.19	0.35	0.75	0.29	0.83	0.30	0.32	1.2
Apte et al. [151]	0.617	0.204	0.382	0.757	0.287	0.866	0.324	0.32	1.2
Mittal et al. [122]	-	-	-	0.742	0.278	0.84	0.31	0.3	1.17
Marella et al. [162]	-	-	0.39	-	-	0.88	-	-	1.19
Johnson and Patel [198]	-	-	0.4	0.75	0.29	0.88	0.32	0.29	1.2

Table 5.1: Comparison of the key wake bubble properties for flow around a sphere



(a) $Re = 20$ **(b)** $Re = 50$

(c) $Re = 100$ **(d)** $Re = 150$

Figure 5.15: Flow around a single stationary sphere. - Streamlines for the four Reynolds number considered are presented. Wake bubble form at $Re = 50$.

The other key parameter in this simulation is the drag coefficient defined by Eq. (5.23). Table 5.2 compares the drag coefficient calculated using the method explained in Section 3.3.1.1, to other numerical studies. The drag coefficient is slightly underestimated specially for lower Reynolds numbers, where the error compared to those reported in [151], is as large as 12% for $Re = 20$. Slightly understimated value for lower Reynolds numbers might be caused by a coarser grid used in this study compared to [122, 151, 162, 198]. However Mark and van Wachem [135] used the same resolution as this study and reported a smaller value for the drag coeffient. To ensure that the results are convergent three grid levels with $h = 0.015$, $h = 0.02$ and $h = 0.03$ are considered for the flow around a stationary sphere at $Re = 20$ and drag coefficients are calculated. Figure 5.16 shows the result of the grid convergence test which confirms the second order accuracy of the interpolation scheme and also the convergence of the method.

Study	C_D			
Re	20	50	100	150
Current Study	2.31	1.41	1.04	0.899
Mark and van Wachem [135]	2.19	-	0.85	-
Apte et al. [151]	2.62	1.55	1.10	0.9
Mittal et al. [122]	-	-	1.08	0.88
Marella et al. [162]	-	1.56	1.06	0.85
Johnson and Patel [198]	-	1.56	1.08	0.9

Table 5.2: Comparison of the drag coefficient C_D for the flow around a stationary sphere.

5.2.6 Sedimentation of a single sphere

Sedimentation of a single sphere is studied in this section. The particle density is set to $\rho_p = 1120\ \frac{kg}{m^3}$ and three different particle Reynolds numbers are considered similar to experimental data of Ten Cate et al. [199]. Table 5.3 summarizes the parameters used for three different cases considered where Re_T is calculated using the particle diameters and terminal velocity and the fluid physical properties. Numerical domain consists of a box with sides $10 \times 10 \times 16\, cm^3$ which is discretized into $100 \times 100 \times 160$ points similar to numerical simulations performed in [151]. The surface of the sphere is discretized using

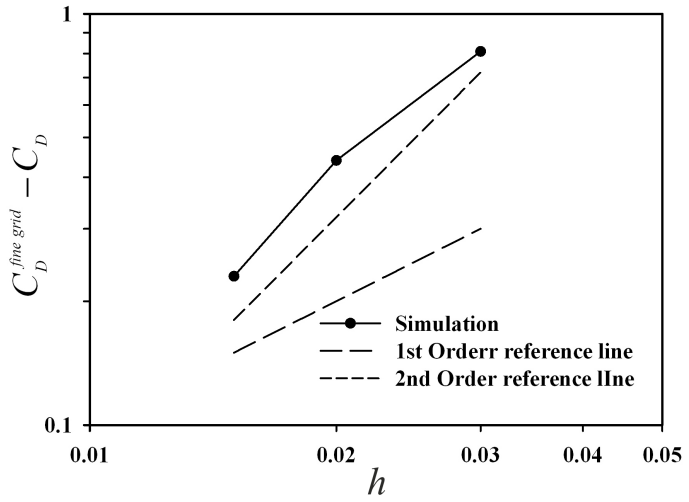


Figure 5.16: Convergence of the drag force - second order accuracy of the force calculation method for stationary objects, reference lines are given for first and second order accuracies for comparison.

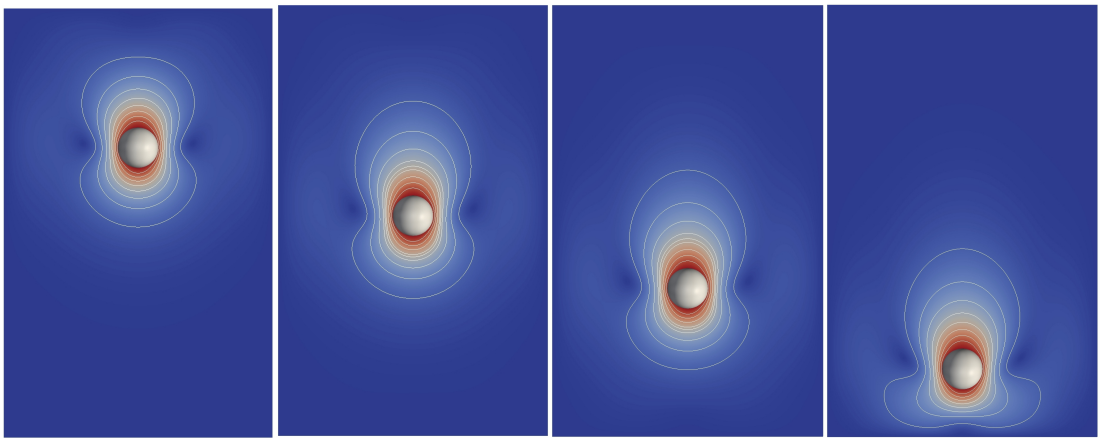
2562 triangles and the sphere volume is discretized using s stair–step grid with $\frac{h}{h_m} = 4$. Figures 5.17 and 5.18 show the normalized velocity contours for the single falling sphere at different times which are comparable to the experimental results presented in [199] and also the numerical results in [151]. Figure 5.19 compares the vertical position of the particle $\overline{H} = \frac{y_c - 0.5D}{D}$ to the experimental data of Ten Cate et al. [199] which shows good agreement between the two dataset.

Case	ρ_f $(\frac{kg}{m^3})$	μ_f $(\frac{Ns}{s})$	Re_T (Ten Cate et al. [199])	Re_T (Current study)
$S1$	970	0.373	1.5	1.3
$S2$	965	0.212	4.1	3.7
$S3$	962	0.113	11.5	10.2

Table 5.3: Simulation parameters for three different cases considered.

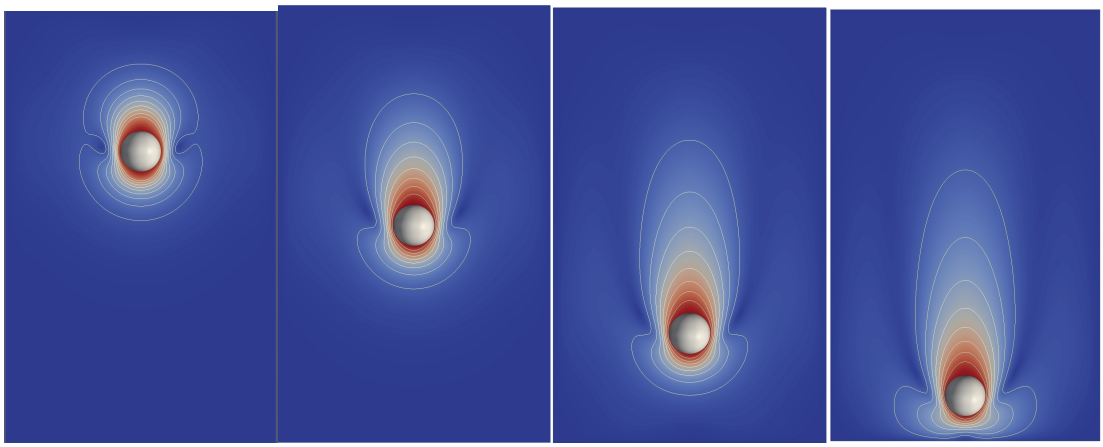
5.3 Conclusion

In this section the numerical algorithm is extended to moving objects and both the mathematical formulation and numerical implementation are discussed in detail. A



(a) $t = 0.5$ **(b)** $t = 1.3$ **(c)** $t = 2.1$ **(d)** $t = 3.0$

Figure 5.17: Contours of normalized velocity $\|u_{p,i}\|/U_T$**.** - contours are presented for four different times for case $S1$.



(a) $t = 0.3$ **(b)** $t = 0.7$ **(c)** $t = 1.2$ **(d)** $t = 1.5$

Figure 5.18: Contours of normalized velocity $\|u_{p.i}\|/U_T$**.** - contours are presented for four different times for case $S3$.

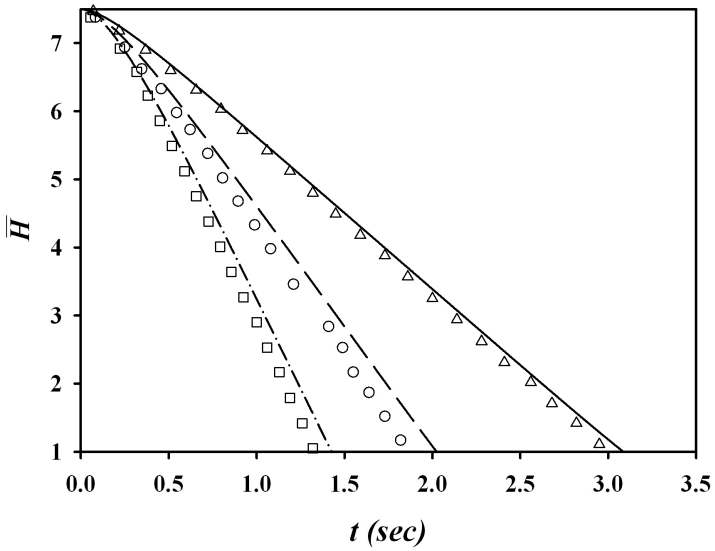


Figure 5.19: Non-dimensional position of the centre of the sphere $\overline{H} = \frac{y_c - 0.5D}{D}$ - — $Re_T = 1.3$, --- $Re_T = 3.7$, -·-·- $Re = 10.2$, symbols present the experimental data of Ten Cate et al. [199] for the corresponding simulations.

particle–particle collision strategy is implemented to stabilize the particle collision process and finally the extension to 3D problems is succinctly discussed. Several problems are discussed which are important from both a numerical and a practical point of view. Particle–particle collision is studied and the process of drafting, kissing and tumbling is investigated in detail for collisions with different Stokes numbers. Long term behavior of a single cold particle subject to natural convection in a long channel is studied next and the limiting Grashof numbers where the particle behavior changes considerably are identified and compared to previous studies. Motion of a single catalyst particle in a cavity is studied next, where heat is generated homogeneously inside the particle. The problem is extensively studied and limiting values of internal heat sources where the particle behavior changes are identified. Finally current 3D implementation is validated by considering flow around a stationary sphere and also by considering a single falling sphere in a Newtonian fluid.

6

A review of the polydispersed Eulerian–Eulerian turbulent models

Following the hierarchy of models discussed in Section 1.3 Eulerian-Eulerian based modeling for turbulent polydispersed multiphase flows are discussed in this chapter. Polydispersity is the norm rather than the exception in many engineering and environmental applications such as spray processes, soot formation, manufacturing nano–particles and reactive precipitation [200]. In this chapter the current state of Eulerian transport models for turbulent polydispersed particulate flows without size class discretization are summarized. The stochastic nature of both carrier and dispersed phase justifies a stochastic approach to describe the behavior of such systems. Understanding the stochastic tools and mathematical framework based on Langevin equation is compulsory and a short discussion is provided. A short discussion on the derivation of the transport equation up to the second order statistics without binning the particle diameter is provided based on the corresponding Fokker-Planck equation and extensive paper of Minier and Peirano [201]. Finally several terms that require closure models are discussed and classified. However writing the actual closures and several proposals are discussed in Chapter 7.

6.1 Introduction

For a monosized particle phase both the EE and EL approaches are studied extensively in the literature. The EL extension to polydispersed systems is intuitive: the particles

with prescribed size are tracked and the only difference is that more particles are needed to get reasonable statistics whereas the extension of the EE approach to polydispersed systems is more involved. In this chapter a unique modelling approach for deriving the field equations for both the dispersed and carrier phases is summarized where the assumptions can be introduced into the equations at a mesoscopic level making the derivation process transparent and intuitive. During the course of the chapter the equivalence of the Lagrangian approach and a pure statistical approach for mathematical description of polydispersed particles in flows will be clarified. A pure statistical method results in the population balance equations (PBE) which is the standard name for a Liouville equation in the engineering community and a recent review of the methods can be found in [200].

The process of deriving the field equation is general and applicable to any type of dense or dilute system and one only needs to identify the significance of each stochastic process and include it in the formulations. However in this chapter only the formulation for dilute systems is discussed which is applicable for example to the spray processes. The subject of sprays is very broad involving aspects such as atomization, single droplet burning, droplet coalescence and break-up, combustion etc., each of which could be studied as a subject of its own. Excellent reviews are available for different aspects of spray processes [202–205]. In these systems even an initially mono sized population undergoes complicated heat and mass transfer processes which causes the population to evolve to a polydispersed system in time. For example a mono sized spray in a flow with a temperature gradient will experience different evaporation rates resulting in a non-uniform size distribution. The size distribution can be the property of the final product as in production of nano-particles or can be an intermediate process variable as in droplets and bubbles where size distribution determines the interfacial area and consequently the rate of heat and mass transfer. Therefore inclusion of the changes in the particle diameter coherently and naturally in the derivation process is crucial.

In general polydispersed methods can be categorized as either sectional or non-sectional methods [206]. In this approach which was first introduced by Tambour [207], the particle population is sectioned according to particular diameter ranges $[D_p, D_p + \delta D_p]$. There are several improvements and formalisms to this approach [208–212] however although in this method some particle diameter statistic can be generated, it has a main drawback. Reliable cross correlation statistics can only be generated if the number

of sections approaches infinity and since the set of transport equations should be solved for each section, this approach is computationally not feasible.

There are much less information on the non–sectional approaches in the literature. Beck and Watkins [213, 214, 215] apparently present a unique approach to modeling polydispersed sprays without resorting to sectional methods. However a close inspection of the method shows that the model has many similarities to the sectional model of Archambault et al. [212], see Scott [206] for more details. The seminal paper of Minier and Peirano [201] discusses in detail the stochastic framework for the derivation of the EE field equations for the polydispersed multiphase flows however the paper is rather mathematical and does not concern the closure of different terms or the physical significance of the terms appearing in the equations. Later Scott [206] puts forward maximum entropy method (MEM) for the closure of the non–integer moments and writes the third order moment transport equations as the starting point for deriving closures for third order moments. They discussed that the maximum entropy method produces errors that in case of a uniform starting distribution exceed 500% since the method simply reduce any probability density functions to a normal distribution given only the first two moments. However good results were reported using an initially normal size distribution.

In this chapter the stochastic framework based on the Fokker-Plank Equation is discussed using numerical examples. Modeling approach is then discussed succinctly concentrating on the physical aspects of the equations by comparing the terms with single phase RANS equations or with their mono-dispersed counterparts when possible.

6.2 PDF Method: A Stochastic Framework

To model the desired macroscopic quantities the easiest way is to write closed PDEs for specific quantities where additional unknowns are added to the system. The success of this method depends on the possibility of expressing the unclosed terms explicitly based on the macroscopic laws. Therefore the applicability of this method to complicated multiphase flows is limited. A full or direct numerical simulation (in the sense of calculating direct interactions between very large number of fluid and particle elements) is also not feasible for complicated flows due to the computational limitations. Thus the only reasonable solution would be what usually referred to as a mesoscopic approach.

Here the mesoscopic approach based on probability density function will be discussed and hence a brief introduction to existing stochastic and statistical methods and the theory in a form suitable for modeling polydispersed particulate flows is required and is provided in the next section.

6.2.1 Definition of a stochastic process

A stochastic variable, X, in applied sciences is usually defined directly from its probability density function (PDF). Without loss of generality it can be assumed that this random (stochastic) variable, X, is a scalar e.g. one-dimensional velocity in a Brownian motion or temperature of a particle. This stochastic variable, X, can take a range of possible values $x \in S$ where S can be a discrete or continuous set such as $\mathbb{N}$ or $\mathbb{R}$ then the probability that X takes the values between x and $x + dx$ is:

$$\Pr(x < X < x + dx) = \mathcal{P}(x)dx. \tag{6.1}$$

In the same fashion it is possible to define a multivariate distribution also called joint probability distribution of r variables $X_1 \dots X_r$. Taking a subset s of r, $(s < r)$, marginal distribution of s variables is the probability that $X_1 \dots X_s$ take the values $x_1 \dots x_s$ regardless of values of $X_{s+1} \dots X_r$, i.e:

$$\mathcal{P}(x_1, \dots, x_s) = \int \mathcal{P}(x_1, \dots, x_s, x_{s+1}, \dots, x_r) dx_{s+1} \dots dx_r \tag{6.2}$$

It is also possible to assign fixed values to $X_{s+1} \dots X_r$ and ask for the joint probability of the remaining variables which is called conditional probability of $X_1 \dots X_s$ and defined by $P(x_1, \dots, x_s | x_{s+1}, \dots, x_r)$. The joint probability of $X_1 \dots X_r$ is equal to the marginal probability of $X_{s+1} \dots X_r$, to have the values $x_{s+1} \dots x_r$, times the conditional probability of $X_1 \dots X_s$, given the values of $X_{s+1} \dots X_r$:

$$\mathcal{P}(x_1, \dots, x_r) = \mathcal{P}(x_{s+1}, \dots, x_r) \mathcal{P}(x_1, \dots, x_s | x_{s+1}, \dots, x_r) \tag{6.3}$$

Having the stochastic variable X defined, other stochastic variables, namely Y, can be derived from it by a mapping, f, these new variables can also be functions of time, t, and hence:

$$Y_X(t) = f(X, t). \tag{6.4}$$

By replacing X by x, one possible value, we get $Y_x(t) = f(x,t)$, an ordinary function called a realization of the process. Therefore the stochastic process in physical sense is an ensemble of these realizations. By measuring the values, $x_0, \ldots, x_n$ at times $t_0, \ldots, t_n$ where $t_0 \leq \cdots \leq t_n$, it is possible to completely describe the process by the joint probability density function $\mathcal{P}(x_n, t_n; \ldots; x_0, t_0)$ which in case of complete independence can be written by

$$\mathcal{P}(x_n, t_n; \ldots; x_0, t_0) = \prod_i \mathcal{P}(x_i, t_i). \tag{6.5}$$

This means that the values of X at time t is independent of its values in the past or future. Also note that the x_i values could each be a vector, e.g velocity components or the whole phase space vector, i.e all the velocity and position components in addition to other scalars. Bold symbols such as $\mathbf{X}$ or $\mathbf{Z}$, and their corresponding values $\mathbf{x}$ and $\mathbf{z}$, are used to indicate that these stochastic variables are in fact vectors, which are also functions of time, defining the whole phase space.

6.2.2 Markov Process

The general process defined by

$$\mathcal{P}(\mathbf{x}_n, t_n; \ldots; \mathbf{x}_0, t_0), \tag{6.6}$$

is very difficult to handle since the knowledge of all previous points is required to describe such a process, thus the problem is usually restricted to a family of processes known as Markov processes where the knowledge of present state of the system completely describes the whole process. In other words the past history of process has no effect on the future evolution of the process or mathematically

$$\mathcal{P}(\mathbf{x}_n, t_n | \mathbf{x}_{n-1}, t_{n-1}; \ldots; \mathbf{x}_0, t_0) = \mathcal{P}(\mathbf{x}_n, t_n | \mathbf{x}_{n-1}, t_{n-1}). \tag{6.7}$$

In Eq. (6.7), $\mathcal{P}(\mathbf{x}_n, t_n | \mathbf{x}_{n-1}, t_{n-1})$ is also called the transition probability. This assumption is very powerful and means everything can be defined in terms of a transition probability $\mathcal{P}(\mathbf{x}_n, t_n | \mathbf{x}_{n-1}, t_{n-1})$ and an initial probability $\mathcal{P}(\mathbf{x}_0, t_0)$. Thus for example:

$$\mathcal{P}(\mathbf{x}_2, t_2; \mathbf{x}_1, t_1; \mathbf{x}_0, t_0) = \mathcal{P}(\mathbf{x}_2, t_2 | \mathbf{x}_1, t_1) \mathcal{P}(\mathbf{x}_1, t_1 | \mathbf{x}_0, t_0) \mathcal{P}(\mathbf{x}_0, t_0). \tag{6.8}$$

Note that the process can be continuous or discontinuous regardless of the nature of the variable $\mathbf{X}$. For example the sample space of classical Brownian motion of a particle immersed in a collection of light molecules, with assumption of hard sphere collisions, because of instant jumps, is not continuous in spite of the fact that the range of velocities is continuous.

6.2.3 The Chapman-Kolmogorov (CK) Equation

For a general stochastic process it is possible to write

$$\mathcal{P}(\mathbf{x}_2, t_2|\mathbf{x}_0, t_0) = \int \mathcal{P}(\mathbf{x}_2, t_2|\mathbf{x}_1, t_1; \mathbf{x}_0, t_0)\mathcal{P}(\mathbf{x}_1, t_1|\mathbf{x}_0, t_0)\mathrm{d}\mathbf{x}_1. \tag{6.9}$$

Introducing the Markov property, i.e Eq. (6.7) into Eq. (6.9) results in the celebrated CK equation:

$$\mathcal{P}(\mathbf{x}_2, t_2|\mathbf{x}_0, t_0) = \int \mathcal{P}(\mathbf{x}_2, t_2|\mathbf{x}_1, t_1)\mathcal{P}(\mathbf{x}_1, t_1|\mathbf{x}_0, t_0)\mathrm{d}\mathbf{x}_1. \tag{6.10}$$

Eq. (6.10) simply states that the probability of a process ending in state $(\mathbf{x}_2, t_2)$ given the initial state $(\mathbf{x}_0, t_0)$ is equal to the sum of all possible paths from $(\mathbf{x}_0, t_0) \rightarrow (\mathbf{x}_2, t_2)$. The CK equation is a complex non–linear equation relating all conditional probabilities to each other. Therefore it is convenient to derive a differential form of CK equation which is easier to handle and physically easier to interpret. It can be derived based on a trajectory point of view [216]. Considering the time evolution of the expectation of a twice differentiable function and using the CK equation, Eq. (6.10), the differential CK equation [201, 216, 217] can be derived as

$$\begin{aligned}\frac{\partial \mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = &-\sum_i \frac{\partial}{\partial x_i}\big(A_i(\mathbf{x}, t)\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)\big) \\ &+ \sum_{ij} \frac{1}{2}\frac{\partial^2}{\partial x_i \partial x_j}\big(B_{ij}(\mathbf{x}, t)\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)\big) \\ &+ \int \big(J(\mathbf{x}|\mathbf{z}, t)\mathcal{P}(\mathbf{z}, t|\mathbf{x}_0, t_0) - J(\mathbf{z}|\mathbf{x}, t)\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)\big)\mathrm{d}\mathbf{z}.\end{aligned} \tag{6.11}$$

In Eq. (6.11), A_i and B_{ij} are the drift vector and the the diffusion matrix which are first and second moment of the transition probability $\mathcal{P}(\mathbf{x}_2, t_1 + \Delta t|\mathbf{x}_1, t_1)$. In differential

CK equation, Eq. (6.11), the first and second terms define the drift and diffusion processes respectively and the last integral defines a jump process. Jump processes will be discussed thoroughly in Section 6.2.6. Pure drift process which is also known as the deterministic process (or Liouville's Equation) and also diffusion process (or Fokker-Plank Equation) are discussed in Section 6.2.5. The differential CK equation simply states that having the initial distribution of some variable $\mathbf{x}$ at time t_0 the final distribution at any time in the future can be found using this equation. Sometimes the probability of a system at state $(\mathbf{x}, t_0)$ to finish in a specific subset of states S at $t + \Delta t$ is required. It can alternatively be said that an equation is required to calculate the probability of ending up in the subset S for every possible state $(\mathbf{x}, t < t_{final})$. In this case the differential CK equation can be restated as the backward CK equation:

$$\begin{aligned}\frac{\partial \mathcal{P}(\mathbf{x}_0, t_0|\mathbf{x}, t)}{\partial t} = &- \sum_i A_i(\mathbf{x}, t) \frac{\partial}{\partial x_i}\big(\mathcal{P}(\mathbf{x}_0, t_0|\mathbf{x}, t)\big) \\ &- \frac{1}{2}\sum_{ij} B_{ij}(\mathbf{x}, t) \frac{\partial^2}{\partial x_i \partial x_j}\big(\mathcal{P}(\mathbf{x}_0, t_0|\mathbf{x}, t)\big) \\ &+ \int \big(J(\mathbf{z}|\mathbf{x}, t)\mathcal{P}(\mathbf{x}_0, t_0|\mathbf{x}, t) - J(\mathbf{z}|\mathbf{x}, t)\mathcal{P}(\mathbf{x}_0, t_0|\mathbf{z}, t)\big)\mathrm{d}\mathbf{z}. \quad (6.12)\end{aligned}$$

Next the most basic stochastic process (Wiener process) is defined and the solution of forward CK equations for this simple case is discussed. Then in Section 6.2.7 a solution to the backward equation is presented.

6.2.4 Wiener process

The Wiener process is a diffusion process and is the solution of the differential CK equation with $A_i = 0, B_{ij} = 1$ and $J = 0$. It is easy to use the characteristic functions [216] to show that the transitional probability of a Wiener process, $\mathcal{P}(\mathbf{w}, t|\mathbf{w}_0, t_0)$, are Gaussian with mean w_0 and variance $t - t_0$. Then the following properties ensue: (i) The sample path is irregular and is not re-producible. (ii) sample path is continuous but not differentiable. (iii) Increments of $W(t)$ defined by $dW_i = W_{i+1} - W_i$ are independent and are Gaussian. (iv) Using the previous property covariance $\langle W(t_2)W(t_1)|(w_0, t_0)\rangle$ is in general given by $\min(t_2 - t_0, t_1 - t_0) + w_0^2$. Simulating Wiener process is easy by using the property (iii), one only needs to sample from a normal distribution and add this to the current state of the system using the relation $W_{i+1} = W_i + dW_i$ and scale the

results. The Wiener process physically translates to a pure Brownian motion without any frictional coefficient defined by [218, 219]:

$$\frac{dU_{p,x}}{dt} = W_{p,x}(t), \tag{6.13}$$

with initial condition:

$$U_{p,x}|_{t=0} = U_{p,x0}, \tag{6.14}$$

where $W_{p,x}$ is a random rapidly fluctuating force per unit mass exerted on particle p, due to collision with other smaller particles. This type of differential equations with a stochastic function on one side are known as Langevin equation, c.f [216, 220]. The RHS of Eq. (6.13) and so the Langevin equation can include other terms, such as a friction ($\mu U_{p,x}$) term [218], which are not stochastic. Figure 6.1 shows a sample path of such motion simulated by integrating Eq. (6.13) directly. The pure stochastic motion without any friction is shown with $\mu = 0$ in this figure, evidently the solution deviates from the initial conditions significantly. However setting the frictional coefficient to $\mu = -1$ acts as a restoring force and the solution only vibrates around the initial value which physically translates to a random motion but with a bounded velocity and is a better model for the physical phenomenon.

The solution of the forward differential CK equation for the Wiener process is given in Figure 6.2. For the forward equation the proper initial condition is $\mathcal{P}(\mathbf{x}, t_0|\mathbf{x_0}, t_0) = \delta(\mathbf{x} - \mathbf{x}_0)$ which simply means that the solution at $t = 0$ is known with probability one. Figure 6.3 shows the effect of the restoring force, this force prevents the PDF from rapid evolution and the value of the velocity will remain around the initial conditions with probability one. This process with a linear drift added to the Wiener process is also known as Ornstein-Uhlenbeck process. Figure 6.4 shows the changes in the standard deviation, σ, of the PDF in time, smaller standard deviation corresponds to the smaller probability of occurrence of velocities too far from the mean while large values show that the extreme velocities are more probable. This simple example clarifies the connection between the trajectory and PDF points of view and also shows why the trajectory point of view resolves more information than the PDF point of view.

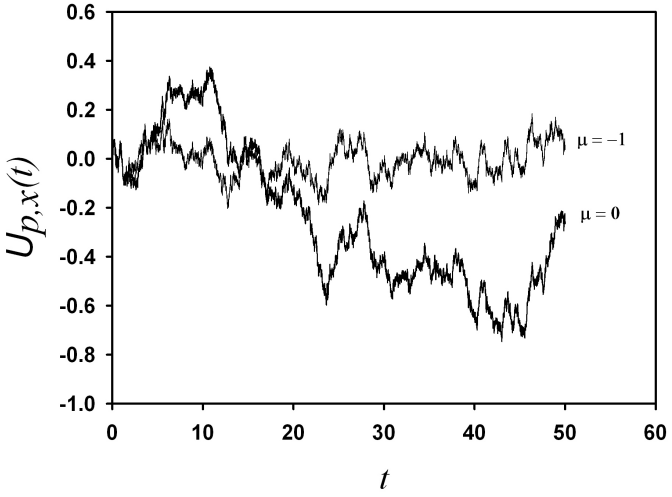


Figure 6.1: 1d-brownian-motion - Two sample path of the Brownian motion with and without friction coefficient μ. Friction force acts as a restoring force that keeps the velocity deviations bounded.

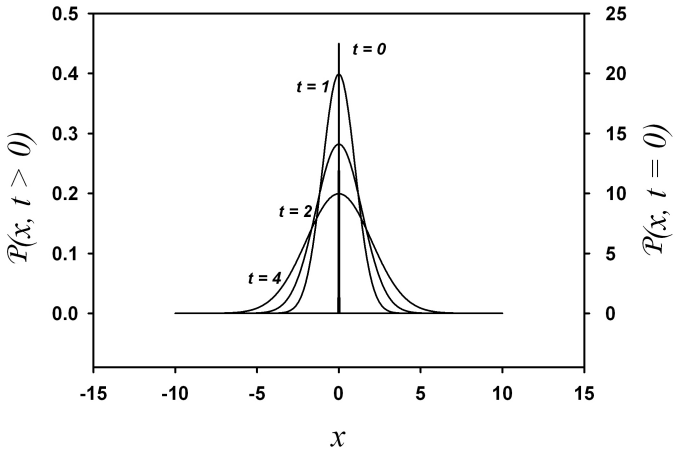


Figure 6.2: Solution of the forward CK equation for a Wiener process - The initial condition is known with probability one, thus is a scaled delta function at t = 0, the PDF then evolves and becomes flatter in time, increasing the uncertainty in the solution.

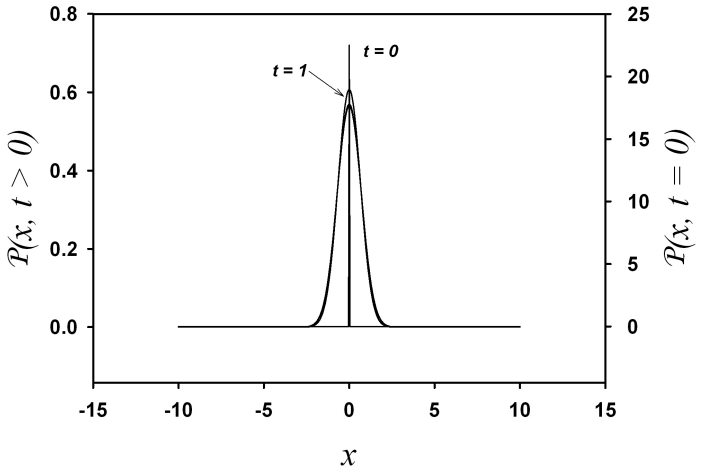


Figure 6.3: Solution of the forward CK equation for the Brownian motion - A Brownian motion with $\mu = -1$. Due to the certainty of the initial condition, it takes the form of the delta function. In this case restoring force keeps the solution around a mean and the PDF remains constant for $t > 2$.

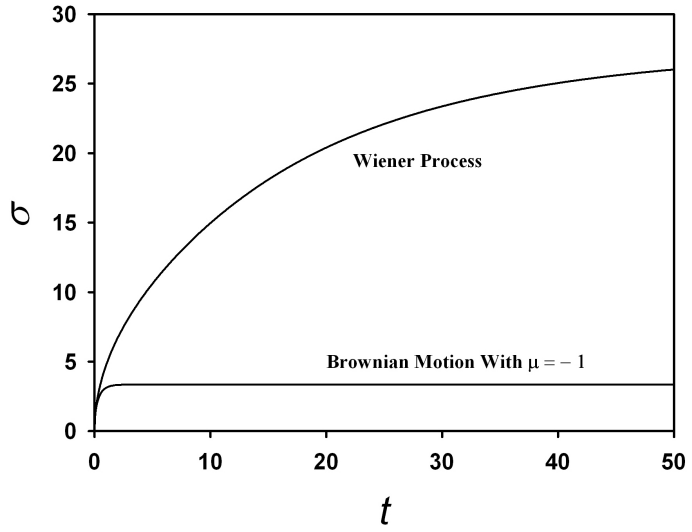


Figure 6.4: Comparison between Standard deviations - Changes in standard deviation of the PDF of the Wiener process and Brownian motion with $\mu = -1$ for $t \in [0, 50]$. It increases significantly for the Wiener process while it rapidly reaches a small constant value for the Brownian motion with $\mu = -1$.

6.2.5 Diffusion Process

Diffusion process is a subset of Markov process applicable to many physical system where the sample path is continuous. Obviously J should be zero in differential CK equation, Eq. (6.11), which stands for discontinuities in this equation and hence the form of the equation describing this process can be written as

$$\frac{\partial \mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = -\sum_i \frac{\partial}{\partial x_i}\big(A_i(\mathbf{x}, t)\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)\big) + \sum_{ij} \frac{1}{2}\frac{\partial^2}{\partial x_i \partial x_j}\big(B_{ij}(\mathbf{x}, t)\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)\big). \quad (6.15)$$

6.2.6 Jump Process

Considering the case where $A_i(\mathbf{x}, t) = B_{ij}(\mathbf{x}, t) = 0$, differential form of CK equation, Eq. (6.11), reduces to:

$$\frac{\partial \mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = \int \big(J(\mathbf{x}|\mathbf{z}, t)\mathcal{P}(\mathbf{z}, t|\mathbf{x}_0, t_0) - J(\mathbf{z}|\mathbf{x}, t)\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0)\big)\mathrm{d}\mathbf{z}. \quad (6.16)$$

Eq. (6.16) is usually known as the Master equation. It should be noted that the Master equation can also be interpreted as a simple gain-loss of probabilities such that the first term in the integral is the gain due to transition from other states and the second is the loss due to transition to other states. It can be said that every diffusion process can be approximated by a jump process. This means that in the limit of infinitely small jump sizes the Master equation becomes a Fokker-Planck equation. Figure 6.5 shows the sample path of a jump process and this scaling assumption, see also [220].

6.2.7 Stochastic differential equations

A stochastic differential equation is a differential equation in which one or more terms are stochastic processes resulting in a solution which itself is a stochastic process. A simple SDE would be that of a Brownian motion introduced in Section 6.2.4. Here only the relations between SDEs and the Fokker-Planck equation is considered.

A general stochastic differential equation also known as Langevin equation has the form:

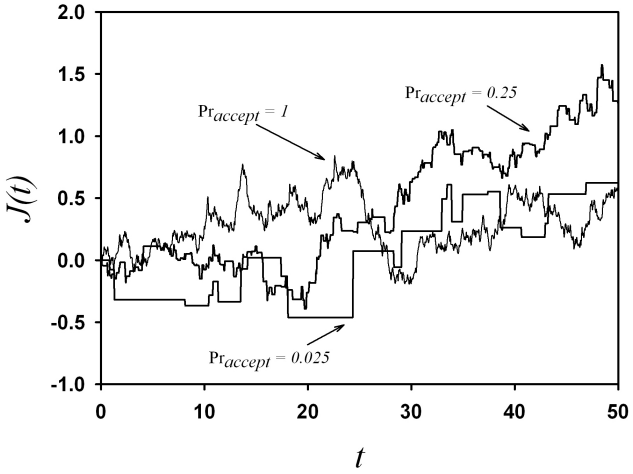


Figure 6.5: Jump process - A typical jump process (labeled by $Pr_{accept} = 0.025$). By increasing the the acceptance rate and using a scaled jump size a diffusion process can always be estimated by a jump process. Note the similarities between the graph for $Pr_{accept} = 1$ and Figure 6.1.

$$\mathrm{d}Z_i(t) = A_i(\mathbf{Z}(t), t)\mathrm{d}t + B_{ij}(\mathbf{Z}(t), t)\mathrm{d}W_j(t), \tag{6.17}$$

where W_j are a set of independent Wiener processes. It is important to notice that both the drift vector A_i and the diffusion matrix B_{ij} are functions of state vector variables $Z_i(t)$. To get from this equation to the corresponding Fokker-Planck equation the time development of an arbitrary $f(\mathbf{Z}(\mathbf{t}))$ should be considered, then using the rules of Itô calculus it is easy to show [221] that the corresponding Fokker-Planck equation in n-dimensional sample space is

$$\frac{\partial \mathcal{P}}{\partial t} = -\frac{\partial}{\partial z_i}[A_i(\mathbf{z}; t)\mathcal{P}] + \frac{1}{2}\frac{\partial^2}{\partial z_i \partial z_j}[(\mathbf{B}\mathbf{B}^T)_{ij}(\mathbf{z}; t)\mathcal{P}]. \tag{6.18}$$

In this equation $\mathbf{B}^T$ is the transpose of $\mathbf{B}$. If the diffusion matrix $B_{ij} = 0$ then the SDE reduces to the deterministic process:

$$\mathrm{d}Z_i(t) = A_i(\mathbf{Z}(t), t)\mathrm{d}t, \tag{6.19}$$

with the corresponding Liouville equation:

$$\frac{\partial \mathcal{P}}{\partial t} = -\frac{\partial}{\partial z_i}[A_i(\mathbf{z};t)\mathcal{P}]. \tag{6.20}$$

This is a completely deterministic system i.e if $z_i(\mathbf{x}_0, t)$ is the solution to Eq. (6.19) with initial conditions $z_i(\mathbf{x}_0, t_0) = x_{i,0}$, then $\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0) = \delta(\mathbf{x}-\mathbf{z}(\mathbf{x}_0, t))$ is the solution to Eq. (6.20) with initial conditions $\mathcal{P}(\mathbf{x}, t_0|\mathbf{x}_0, t_0) = \delta(\mathbf{x} - \mathbf{x}_0)$. The proof is best obtained by direct substitution of

$$\mathcal{P}(\mathbf{x}, t|\mathbf{x}_0, t_0) = \delta(\mathbf{x} - \mathbf{z}(\mathbf{x}_0, t)), \tag{6.21}$$

into the RHS of Eq. (6.20) (Note that x_i is the independent variables):

$$\begin{aligned} -\frac{\partial}{\partial x_i}[A_i(\mathbf{x};t)\delta(\mathbf{x} - \mathbf{z}(\mathbf{x}_0, t)] & \\ = -\frac{\partial}{\partial x_i}[A_i(\mathbf{z};t)\delta(\mathbf{x} - \mathbf{z}(\mathbf{x}_0, t)] & \\ = -A_i(\mathbf{z};t)\frac{\partial}{\partial x_i}[\delta(\mathbf{x} - \mathbf{z}(\mathbf{x}_0, t)]. & \end{aligned} \tag{6.22}$$

The property of delta function is used in moving from first to second line and the fact that A_i is not a function of $\mathbf{x}$ anymore, to move from second to third line. Also substituting Eq. (6.21) into the LHS of Eq. (6.20) and using the chain rule and again using the properties of the delta function, gives

$$\frac{\partial}{\partial t}\left[\delta\left(\mathbf{x} - \mathbf{z}(\mathbf{x}_0, t)\right)\right] = -\frac{\partial}{\partial x_i}\left[\delta\left(\mathbf{x} - \mathbf{z}(\mathbf{x}_0, t)\right)\right]\frac{\mathrm{d}z_i(t)}{\mathrm{d}t}. \tag{6.23}$$

Now using $\mathrm{d}Z_i(t) = A_i(\mathbf{Z}(t), t)\mathrm{d}t$ in Eq. (6.23) and changing to phase space notation Eq. (6.22) is restored.

Using an approach based on the Fokker-Planck equation provides a unique opportunity to add many physical phenomena to the model very conveniently and also lots of information about the model can be inferred at this stage. For example it is well known that inhomogeneous turbulence results in non-Gaussian PDFs [222] this behavior can easily be included in the model using a non-linear drift term. Figure 6.6 shows the evolution of the PDF of a SDE derived by adding the non–linear drift $sin(X_t)$ to a Wiener process. An appropriate final condition for the backward CK equation can be the scaled box function $\mathbb{1}_{[-5,5]}$. The solution shows that the non-Gaussian behavior is

easily captured using this non–linear drift term and the probability of a solution starting at any previous time ending up in state $\mathbb{1}_{[-5,5]}$ is the solution to the backward equation.

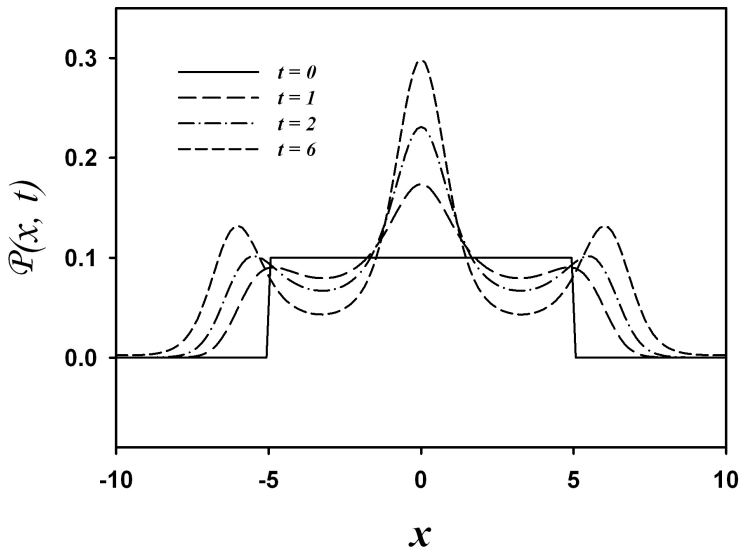


Figure 6.6: Solution of the backward CK equation - A non–linear drift term $sin(X_t)$ produces a non-Gaussian behavior of the PDF showing the benefits of modeling at a mesoscopic level.

6.3 Fluid-Particle systems

In this section a short discussion is provided on how the results of the probability theory and stochastic calculus presented in the Section 6.2 can be used to derive the various equations for modeling polydispersed particulate flows. Characterization of individual particles (solid or fluid) requires detailed knowledge of particle position, velocity, size, shape, rotation and temperature at a specified time. The equations derived in this manner are not ordinary PDEs but are PDEs in a hyperspace of several dimension and this makes the direct solution to these equations a formidable task [212]. Therefore it is necessary to reduce the dimensionality of the problem and make some simplifying assumptions to reduce the solution cost and also the model development process. Following assumptions are used to derive the equations [201, 206]: *(i)* Only spherical particles are considered and hence the only geometric characteristic is the diameter.

(ii) No collision is considered. *(iii)* Processes are assumed Markovian. *(iv)* A fluid particle is a small fluid element with length scale smaller than the Kolmogorov length scale but much larger than the molecular free path. Regarding the second assumption, it should be obvious at this stage that these forces can be included as a jump process in this general framework. This can be done by defining $W(\mathbf{x}|\mathbf{z},t) = \lambda(\mathbf{z},t)g(\mathbf{x}|\mathbf{z},t)$ where $\lambda(\mathbf{z},t)$ is the probability of occurrence of a jump at state $\mathbf{z}$ and $g(\mathbf{x}|\mathbf{z},t)$ is the probability of a jump to have a specified amplitude in going from state $\mathbf{z}$ to state $\mathbf{x}$. This is exactly the process used to produce Figure 6.5, however the problem is how to model g and λ based on the underlying physical phenomena [201]. Williams and Crane [223] studied the particle collision rate in turbulent flows for both solid particles and bubbles which could be a good starting point for such models. Multi particle statistics in the context of kinetic theory for dense particulate flows can be found in [10, 224, 225] but are outside the scope of the current study.

6.3.1 Eulerian and Lagrangian descriptions

There are two possible view points of the fluid-particle system. A Lagrangian point of view describes the probability of finding two particles (fluid and discrete) at a given state $\mathbf{Z}_{fp} = (\mathbf{X}_f, \mathbf{U}_f, \mathbf{\Psi}_f, \mathbf{X}_p, \mathbf{U}_p, \mathbf{\Psi}_p)$ where $\mathbf{X}$, $\mathbf{U}$ and $\mathbf{\Psi}$ are position, velocity and an arbitrary property vector[1]. The Lagrangian PDF is defined by $\mathcal{P}^L_{fp}(\mathbf{y}_f, \mathbf{u}_f, \boldsymbol{\psi}_f, \mathbf{y}_p, \mathbf{u}_p, \boldsymbol{\psi}_p; t)$ and the probability of finding a fluid/particle pair in the range $[\mathbf{u}_k, \mathbf{u}_k + \mathrm{d}\mathbf{u}_k], [\mathbf{y}_k, \mathbf{y}_k + \mathrm{d}\mathbf{y}_k], [\boldsymbol{\psi}_k, \boldsymbol{\psi}_k + \mathrm{d}\boldsymbol{\psi}_k]$ at time t (where $k = f$ for fluid and $k = p$ for particle) is simply given by

$$\mathcal{P}^L_{fp}(\mathbf{y}_f, \mathbf{u}_f, \boldsymbol{\psi}_f, \mathbf{y}_p, \mathbf{u}_p, \boldsymbol{\psi}_p; t)\mathrm{d}\mathbf{y}_f\mathrm{d}\mathbf{u}_f\mathrm{d}\boldsymbol{\psi}_f\mathrm{d}\mathbf{y}_p\mathrm{d}\mathbf{u}_p\mathrm{d}\boldsymbol{\psi}_p, \tag{6.24}$$

which conforms to the usual normalization constraint, i.e $\int \mathcal{P}^L_{fp}\mathrm{d}\mathbf{z}_{fp} = 1$. An Eulerian (field) point of view describes the probability of finding the fluid–particle mixture in a given state $\mathbf{z}_{fp} = (\mathbf{u}_f, \boldsymbol{\psi}_f, \mathbf{u}_p, \boldsymbol{\psi}_p)$ at two fixed points $(\mathbf{x}_f, \mathbf{x}_p)$ and fixed time. Here $\mathbf{x}$, $\mathbf{u}$ and $\boldsymbol{\psi}$ have the same meaning as in their Lagrangian counterparts and distinction is made by using upper and lower case letters. Correspondingly the probability of finding the system (at time t and position $\mathbf{x}_f, \mathbf{x}_p$) in the given state in the range $[\mathbf{u}_k, \mathbf{u}_k +$

[1] $\mathbf{\Psi}$ can be a combination of different scalars, therefore a vector notation is used for its presentation. However here only the diameter is considered and consequently this actually is only a scalar.

$\mathrm{d}\mathbf{u}_k], [\mathbf{x}_k, \mathbf{x}_k + \mathrm{d}\mathbf{x}_k], [\boldsymbol{\psi}_k, \boldsymbol{\psi}_k + \mathrm{d}\boldsymbol{\psi}_k]$ is

$$\mathcal{P}^E_{fp}(\mathbf{u}_f, \boldsymbol{\psi}_f, \mathbf{u}_p, \boldsymbol{\psi}_p; \mathbf{x}_f, \mathbf{x}_p, t)\mathrm{d}\mathbf{u}_f\mathrm{d}\boldsymbol{\psi}_f\mathrm{d}\mathbf{u}_p\mathrm{d}\boldsymbol{\psi}_p. \tag{6.25}$$

However $\int \mathcal{P}^E_{fp}\mathrm{d}\mathbf{z}_{fp} < 1$, because of the fact that positions are no longer a property of the state vector. To reduce the dimensionality a one-point description can be defined by integrating the joint Eulerian probability density of the system

$$\mathcal{P}^E_f(\mathbf{u}_f, \boldsymbol{\psi}_f; \mathbf{x}_f, t) = \int \mathcal{P}^E_{fp}(\mathbf{u}_f, \boldsymbol{\psi}_f, \mathbf{u}_p, \boldsymbol{\psi}_p; \mathbf{x}_f, \mathbf{x}_p, t)\mathrm{d}\mathbf{x}_p\mathrm{d}\mathbf{u}_p\mathrm{d}\boldsymbol{\psi}_p, \tag{6.26}$$

$$\mathcal{P}^E_p(\mathbf{u}_p, \boldsymbol{\psi}_p; \mathbf{x}_p, t) = \int \mathcal{P}^E_{fp}(\mathbf{u}_f, \boldsymbol{\psi}_f, \mathbf{u}_p, \boldsymbol{\psi}_p; \mathbf{x}_f, \mathbf{x}_p, t)\mathrm{d}\mathbf{x}_f\mathrm{d}\mathbf{u}_f\mathrm{d}\boldsymbol{\psi}_f. \tag{6.27}$$

This is a one–point, two–particle description which means that the probability of finding a fluid and a discrete particle separately at two different fixed points at a fixed time is considered. This definition obviously contains less information than the two–point, two–particle description where the joint probability of finding a fluid and a discrete particle is considered. This can also be mathematically justified by noting that the marginal PDFs (one–point PDFs) can always be constructed from the joint PDFs by integration while the reverse operation is not always possible. As a result of the fact that a fluid and a particle can not co–exist at the same position at the same time the following equation can be written for the one–point Eulerian PDFs

$$\int \mathcal{P}^E_f(\mathbf{u}_f, \boldsymbol{\psi}_f; \mathbf{x}_f, t)\mathrm{d}\mathbf{u}_f\mathrm{d}\boldsymbol{\psi}_f + \int \mathcal{P}^E_p(\mathbf{u}_p, \boldsymbol{\psi}_p; \mathbf{x}_p, t)\mathrm{d}\mathbf{u}_p\mathrm{d}\boldsymbol{\psi}_p = 1, \tag{6.28}$$

$\mathcal{P}^E_f$ and $\mathcal{P}^E_p$ can be normalized by defining normalization factors $\alpha_f(\mathbf{x}, t)$ and $\alpha_p(\mathbf{x}, t)$ which can be interpreted as phase volume fractions.

6.3.2 Mass Density Function (MDF)

It is more convenient to work with MDFs due to their intuitive physical interpretation. The Eulerian and Lagrangian MDFs can be defined by

$$F^L_f(\mathbf{y}_f, \mathbf{u}_f, \boldsymbol{\psi}_f; t) = M_f(t)\mathcal{P}^L_f(\mathbf{y}_f, \mathbf{u}_f, \boldsymbol{\psi}_f; t), \tag{6.29}$$

and

$$F_p^L(\mathbf{y}_p, \mathbf{u}_p, \boldsymbol{\psi}_p; t) = M_p(t)\mathcal{P}_p^L(\mathbf{y}_p, \mathbf{u}_p, \boldsymbol{\psi}_p; t). \tag{6.30}$$

In these equations F_k^L, with $k = f$ or p for fluid and particle respectively, can be interpreted as the probable mass of fluid or particle at the given state $\mathbf{z_k} = (\mathbf{y}_k, \mathbf{u}_k, \boldsymbol{\psi}_k)$. M_k is the normalization constant for F_k or the total mass of phase k which physically can be calculated by $\int_{\mathcal{V}_f} \rho_f(\mathbf{x}_f, t)\mathrm{d}\mathbf{x}_f$, $\mathcal{V}_f$ being the total fluid volume, for the fluid phase and $\sum_{i=1}^{N_p} m_{p,i}$ for the particle phase. By integrating the Lagrangian MDFs, Eulerian MDFs can be derived and are given by

$$F_f^E(\mathbf{u}_f, \boldsymbol{\psi}_f; \mathbf{x}_f, t) = \int M_f(t)\mathcal{P}_f^L(\mathbf{y}_f, \mathbf{u}_f, \boldsymbol{\psi}_f; t)\delta(\mathbf{x}_f - \mathbf{y}_f)\mathrm{d}\mathbf{y}_f \tag{6.31}$$

$$F_p^E(\mathbf{u}_p, \boldsymbol{\psi}_p; \mathbf{x}_p, t) = \int M_p(t)\mathcal{P}_p^L(\mathbf{y}_p, \mathbf{u}_p, \boldsymbol{\psi}_p; t)\delta(\mathbf{x}_p - \mathbf{y}_p)\mathrm{d}\mathbf{y}_p \tag{6.32}$$

6.4 PDF evolution equation

In Section 6.2 the weak equivalence, meaning Lagrangian equations contain more information, between Lagrangian stochastic description and evolution of underlying PDF is demonstrated. The starting point for the model derivation is the specification of the SDEs that describe the joint evolution of the particle and fluid element pair which at the most general case should contain the inter–phase mechanisms such as mass and momentum transfer. One advantage of this approach is the explicit addition of these terms to the equations as discussed in Section 6.3. In this section the Lagrangian equations and derivation of the moment equations of a general property based on [201, 206] which can be done by integrating the PDF evolution equation [201] is succinctly explained.

6.4.1 Deterministic description

Using a one–point, two–particle description and defining fluid and particle state vectors by $\mathbf{Z}_f^+ = \left(X_{f,i}^+, U_{f,i}^+, \Psi_{f,m}^+\right)$ and $\mathbf{Z}_p^+ = \left(X_{p,i}^+, U_{p,i}^+, \Psi_{p,n}^+\right)$, where the '+' is used to indicate the deterministic nature of this process, following differential equations can be written for the fluid–particle system

$$\mathrm{d}X^+_{f,i} = U^+_{f,i}\mathrm{d}t \tag{6.33}$$

$$\mathrm{d}X^+_{p,i} = U^+_{p,i}\mathrm{d}t \tag{6.34}$$

$$\mathrm{d}U^+_{f,i} = A^+_{f,i}\mathrm{d}t + A^+_{p\to f,i}\mathrm{d}t \tag{6.35}$$

$$\mathrm{d}U^+_{p,i} = A^+_{p,i}\mathrm{d}t \tag{6.36}$$

The problem with the deterministic assumption is that accelerations $A_{f,i}$ should be expressed as functions of the state vector specified. However this state vector is only a reduced state vector meaning that only one fluid and one discrete particle are considered at a specific time, and hence models for $A_{f,i}$ are required. One can assume that the accelerations are fast variables and model them using a Wiener process and utilize a stochastic description. This is equivalent to assuming that the accelerations are composed of two terms, a term which can be written as a function of the current reduced state vectors and a stochastic term that constitutes the net effects of interaction of this element with other fluid or discrete elements, see [226] for a discussion on how to identify the fast variables.

6.4.2 Stochastic One-point, Two-particle description

It is customary to define the fluid-particle state vector $\mathbf{Z}_{fp}$ by

$$\mathbf{Z}_{fp} = \left(X_{f,i}, U_{f,i}, X_{p,i}, U_{p,i}, U_{s,i}, \Phi_p\right), \tag{6.37}$$

with the corresponding phase space vector $\mathbf{z}_{fp}$

$$\mathbf{z}_{fp} = \left(y_{f,i}, u_{f,i}, y_{p,i}, u_{p,i}, u_{s,i}, \phi_p\right). \tag{6.38}$$

Here $U_{s,i}$ is the velocity encountered or 'seen' by the discrete particle. Then for the fluid element one can write the following generalized system of SDEs [201, 226]:

$$\mathrm{d}X_{f,i} = U_{f,i}\mathrm{d}t \tag{6.39}$$

$$\mathrm{d}U_{f,i} = A_{f,i}\mathrm{d}t + A_{p\to f,i}\mathrm{d}t + B_{f,ij}\mathrm{d}W_{f,j}, \tag{6.40}$$

where $A_{f,i}$ is the fluid acceleration term, $A_{p\to f,i}$ accounts for momentum transfer from the particulate to the fluid phase and finally $B_{f,ij}$ is the diffusion terms corresponding to

the stochastic Wiener processes $\mathrm{d}W_{f,j}$. Traditionally only velocity and position vectors are kept in the fluid state vector [227] and while this PDF describes the distribution of velocities, its description of the turbulent motion is seriously deficient [228], in that it provides no information on the length or time scale of the turbulent motion. Therefore another stochastic equation for fluid turbulent frequency Ω_f along the fluid element trajectory are included in the state vector as discussed in [229, 230]. The inclusion of Ω_f in the context of the current framework was first suggested and derived by Scott [206]. The field equation for Ω_f contains the unclosed turbulent flux $\langle \omega'_f u'_{f,i} \rangle$ which requires modeling [231]. Pope and Chen [229], Pope [230] derived a model for this term and there is no need to further consider this equation for writing higher order moments [206] and is not discussed here. The discrete particle trajectories considering a non-evaporating dispersed phase, $\mathrm{d}\Phi_p = 0$, are as follows

$$\mathrm{d}X_{p,i} = U_{p,i}\mathrm{d}t \tag{6.41}$$

$$\mathrm{d}U_{p,i} = A_{p,i}\mathrm{d}t \tag{6.42}$$

$$\mathrm{d}U_{s,i} = A_{s,i}\mathrm{d}t + A_{p\to s,i}\mathrm{d}t + B_{s,ij}\mathrm{d}W_{s,j}, \tag{6.43}$$

where the particle drift coefficient $A_{p,i}$ is taken as the particle acceleration due to aerodynamic drag, $A_{s,i}$ is the drift coefficient for the fluid element sampled or seen by the particle and $A_{p\to s,i}$ accounts for the momentum transfer between the particulate and sampled velocity phases. Also note that if a two point description were used then we were able to calculate $A_{s,i}$, but here $\mathrm{d}W_{s,j}$ is added which contains the effects of such reduction in the stated vector. Momentum transfer terms between phases $A_{p\to s,i}$and $A_{p\to f,i}$ are defined by [201]:

$$A_{p\to s,i} = -\frac{\alpha_p \rho_p}{\alpha_f \rho_f}\left(\frac{U_{s,i} - U_{p,i}}{\tau_p}\right), \; A_{p\to f,i} = -\frac{\alpha_p \rho_p}{\alpha_f \rho_f}\left(\frac{U_{s,i} - U_{p,i}}{\tau_p}\right), \tag{6.44}$$

Note that the form of $A_{p\to s,i}$ is rather intuitive, since both variables are available in the same location. However derivation of $A_{p\to f,i}$ in this form requires definition of a function that takes into account the probability of finding a fluid element at the particle location. This definition although required does not change the final result [201]. In Eqs. (6.44), $\alpha_p \rho_p$ is the expected particle mass at position x_i and $\alpha_f \rho_f$ is the

expected mass of the fluid and τ_p is the particle time scale and $A_{f,i}$ is fluid acceleration term [226, 230, 232].

Assuming F^E_{fp} be the MDF of the fluid–particle system, the evolution equations corresponding to the system of SDEs (Eq. (6.39)–(6.43)) can be derived using the Fokker-Planck equation which after integration over all particle or fluid properties results in the following equations for fluid and particle MDF evolution equations [206]:

$$\frac{\partial F^E_f}{\partial t} + \frac{\partial}{\partial x_i}\left[v_{f,i}F^E_f\right] = -\frac{\partial}{\partial v_{f,i}}\left[\left(A_{f,i} + \langle\, A_{p\to f,i}|\mathbf{Z}_f = \mathbf{z}_f\,\rangle\right) F^E_f\right] + \frac{1}{2}\frac{\partial^2}{\partial v_{f,i}\partial v_{f,j}}\left[\left(B_f B^T_f\right)_{ij} F^E_f\right] \quad (6.45)$$

$$\frac{\partial F^E_p}{\partial t} + \frac{\partial}{\partial x_i}\left[v_{p,i}F^E_p\right] = -\frac{\partial}{\partial v_{p,i}}\left[A_{p,i}F^E_p\right] - \frac{\partial}{\partial v_{s,i}}\left[\left(A_{s,i} + \langle A_{p\to s,i}|\mathbf{Z}_p = \mathbf{z}_p\rangle\right) F^E_f\right] + \frac{1}{2}\frac{\partial^2}{\partial v_{s,i}\partial v_{s,j}}\left[\left(B_s B^T_s\right)_{ij} F^E_p\right] \quad (6.46)$$

where $\langle A|\mathbf{Z} = \mathbf{z}\rangle$ appear here, as a result of choosing reduced state vector, i.e a one–point description, and is defined by

$$\langle A|\mathbf{Z}^r = \mathbf{z}^r\rangle = \int A(\mathbf{z}, t)\mathcal{P}(\mathbf{z}^{n-r}|\mathbf{Z}^r = \mathbf{z}^r; t)\mathrm{d}\mathbf{z}^{n-r}. \quad (6.47)$$

6.4.3 Transport Equation of a General Property

Macroscopic field equations for a general fluid property $\mathscr{H}_f$ are derived in [206] by multiplying Eq. (6.45) by $\mathscr{H}_f$ and integrating by parts when necessary and assuming the boundary points go to zero at $\pm\infty$. The equations are given as follows:

$$\frac{\partial}{\partial t}\left[\alpha_f\rho_f\left\langle\mathscr{H}_f\right\rangle\right] + \frac{\partial}{\partial x_j}\left[\alpha_f\rho_f\left\langle u_{f,j}\mathscr{H}_f\right\rangle\right] = \alpha_f\rho_f\left\langle A_{f,j}\frac{\partial\mathscr{H}_f}{\partial u_{f,j}}\right\rangle + \int_{-\infty}^{+\infty}\langle A_{p\to f,j}|\mathbf{Z}_f = \mathbf{z}_f\rangle\frac{\partial\mathscr{H}_f}{\partial u_{f,j}}F^E_f\mathrm{d}\mathbf{v}_f\mathrm{d}\theta_f + \frac{1}{2}\alpha_f\rho_f\left\langle\left(B_f B^T_f\right)_{ij}\frac{\partial^2\mathscr{H}_f}{\partial u_{f,i}\partial u_{f,j}}\right\rangle. \quad (6.48)$$

The boundary assumption used to derive Eq. (6.48) is equivalent to assuming that probability of finding a point with infinite velocity in the phase space vector is zero meaning that the PDFs converge to zero when at least one component of the velocity goes to infinity and this should be included in the construction of the PDFs.

Writing Eq. (6.48) in terms of transformed coordinates $\mathbf{v}_f \rightarrow \mathbf{v}'_f \left(= \mathbf{v}_f - \langle \mathbf{u}_f \rangle (\mathbf{x}, t)\right)$ makes the derivation of fluctuating field transport equations straightforward by defining the derivatives in the transformed coordinates this equation is give as follows [206]:

$$\frac{\mathrm{D}_f}{\mathrm{D}t}\left[\alpha_f \rho_f \left\langle \mathscr{H}'_f \right\rangle\right] + \frac{\partial}{\partial x_m}\left[\alpha_f \rho_f \left\langle u'_{f,m} \mathscr{H}'_f \right\rangle\right] = \alpha_f \rho_f \left\langle A_{f,m} \frac{\partial \mathscr{H}'_f}{\partial u'_{f,m}} \right\rangle$$
$$-\alpha_f \rho_f \frac{\mathrm{D}_f \langle u_{f,m} \rangle}{\mathrm{D}t} \left\langle \frac{\partial \mathscr{H}'_f}{\partial u'_{f,m}} \right\rangle - \alpha_f \rho_f \frac{\partial \langle u_{f,n} \rangle}{\partial x_m} \left\langle \frac{\partial u'_{f,m} \mathscr{H}'_f}{\partial u'_{f,n}} \right\rangle + \frac{1}{2} \alpha_f \rho_f \left\langle \left(B_f B_f^T\right)_{mn} \frac{\partial^2 \mathscr{H}'_f}{\partial u'_{f,m} \partial u'_{f,n}} \right\rangle$$
$$+ \int_{-\infty}^{+\infty} \left(A_{f,m} + \langle A_{p \rightarrow f,m} | \mathbf{Z}_f = \mathbf{z}_f \rangle\right) \frac{\partial \mathscr{H}'_f}{\partial v'_{f,m}} F'^E_f \mathrm{d}\mathbf{v}'_f. \quad (6.49)$$

In Eq. (6.49), $\mathrm{D}_k \langle \cdot \rangle / \mathrm{D}t = \partial \langle \cdot \rangle / \partial t + \langle u_{k,m} \rangle \, \partial \langle \cdot \rangle / \partial x_m$. A similar procedure results in the following macroscopic equations for a general particle property $\mathscr{H}_p$ and a general particle fluctuating property $\mathscr{H}'_p$ [206]

$$\frac{\partial}{\partial t}\left[\alpha_p \rho_p \langle \mathscr{H}_p \rangle\right] + \frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \langle u_{p,j} \mathscr{H}_p \rangle\right] = \alpha_p \rho_p \left\langle A_{p,j} \frac{\partial \mathscr{H}_p}{\partial u_{p,j}} \right\rangle$$
$$+ \alpha_p \rho_p \left\langle A_{s,j} \frac{\partial \mathscr{H}_p}{\partial u_{s,j}} \right\rangle + \frac{1}{2} \alpha_p \rho_p \left\langle \left(B_s B_s^T\right)_{ij} \frac{\partial^2 \mathscr{H}_p}{\partial u_{s,i} \partial u_{s,j}} \right\rangle$$
$$+ \int_{-\infty}^{+\infty} \langle A_{p \rightarrow s,j} | \mathbf{Z}_p = \mathbf{z}_p \rangle \frac{\partial \mathscr{H}_p}{\partial v_{s,j}} F_p^E \mathrm{d}\mathbf{v}_p \mathrm{d}\mathbf{v}_s \mathrm{d}\delta_p, \quad (6.50)$$

$$\begin{aligned}
&\frac{\mathrm{D}_p}{\mathrm{D}t}\left[\alpha_p\rho_p\left\langle\mathscr{H}_p'\right\rangle\right]+\frac{\partial}{\partial x_m}\left[\alpha_p\rho_p\left\langle u_{p,m}'\mathscr{H}_p'\right\rangle\right]= \\
&\quad-\alpha_p\rho_p\left[\frac{\mathrm{D}_p\left\langle u_{p,m}\right\rangle}{\mathrm{D}t}\left\langle\frac{\partial\mathscr{H}_p'}{\partial u_{p,m}'}\right\rangle+\frac{\mathrm{D}_p\left\langle u_{s,m}\right\rangle}{\mathrm{D}t}\left\langle\frac{\partial\mathscr{H}_p'}{\partial u_{s,m}'}\right\rangle\mathrm{D}_p\left\langle\phi_p\right\rangle\mathrm{D}t\left\langle\frac{\partial\mathscr{H}_p'}{\partial\phi_p'}\right\rangle\right] \\
&\quad+\alpha_p\rho_p\left[\left\langle A_{p,m}\frac{\partial\mathscr{H}_p'}{\partial u_{p,m}'}\right\rangle+\left\langle A_{s,m}\frac{\partial\mathscr{H}_p'}{\partial u_{s,m}'}\right\rangle\right] \\
&-\alpha_p\rho_p\left[\frac{\partial\langle u_{p,n}\rangle}{\partial x_m}\left\langle\frac{\partial}{\partial u_{p,n}'}\left[u_{p,m}'\mathscr{H}_p'\right]\right\rangle+\frac{\partial\langle u_{s,n}\rangle}{\partial x_m}\left\langle\frac{\partial}{\partial u_{s,n}'}\left[u_{p,m}'\mathscr{H}_p'\right]\right\rangle+\frac{\partial\langle\phi_p\rangle}{\partial x_m}\left\langle\frac{\partial}{\partial\phi_p'}\left[u_{p,m}'\mathscr{H}_p'\right]\right\rangle\right] \\
&\quad+\frac{1}{2}\alpha_p\rho_p\left\langle\left(B_sB_s^T\right)_{mn}\frac{\partial^2\mathscr{H}_p'}{\partial u_{s,m}'\partial u_{s,n}'}\right\rangle+\int_{-\infty}^{+\infty}\left\langle A_{p\to s,m}|\mathbf{Z}_p=\mathbf{z}_p\right\rangle\frac{\partial\mathscr{H}_p'}{\partial u_{s,m}'}F'^E_p\mathrm{d}\mathbf{v}_p'\mathrm{d}\mathbf{v}_s'\mathrm{d}\delta_p'.
\end{aligned}\tag{6.51}$$

6.5 Fluid field equations

In this section various equations for the fluid phase are presented which can all be derived using equations (6.48) and (6.49) for fluid and (6.50) and (6.51) for particle phase directly. These equation were first reported in [201]. These equations are reiterated here, since an attempt is made here to describe the different terms appearing in these equations. In addition details of derivation of these equation from equations (6.48)–(6.51) is not provided before and is given in Appendix C for fluid phase Reynold stresses and particle momentum equation.

6.5.1 Fluid continuity equation

Setting $\mathscr{H}_f=1$ in Eq. (6.48) results in the following equation for fluid continuity

$$\frac{\partial}{\partial t}\left(\alpha_f\rho_f\right)+\frac{\partial}{\partial x_i}\left[\alpha_f\rho_f\left\langle u_{f,i}\right\rangle\right]=0.\tag{6.52}$$

This is the usual continuity equation considering the volume fraction of fluid phase.

6.5.2 Fluid phase momentum equation

Setting $\mathscr{H}_f=u_{f,i}$ in Eq. (6.48) results in the following equation for fluid momentum using material derivative defined in previous section [201]:

$$\alpha_f \rho_f \frac{\mathrm{D}_f \langle u_{f,i} \rangle}{\mathrm{D}t} = -\frac{\partial}{\partial x_j} \left[\alpha_f \rho_f \langle u'_{f,i} u'_{f,j} \rangle \right] + \alpha_f \rho_f \langle A_{f,i} \rangle + S^{\langle \mathbf{u}_f \rangle}_{p \to f,i}, \tag{6.53}$$

where $S^{\langle \mathbf{u}_f \rangle}_{p \to f,i}$ is the momentum source because of the fluid–particle coupling. First term in the RHS of Eq. (6.53) is the Reynolds stress tensor which can be interpreted similar to the single phase case [230] as the mean momentum flux due to the fluctuating velocity on the boundary of a control volume. Second term is the fluid acceleration term which contains mean pressure gradient and viscous stresses. Momentum source terms need modeling and closures can be found in [224].

6.5.3 Fluid phase Reynolds stresses

Using Eq. (6.49) with $\mathscr{H}_f = \langle u'_{f,i} u'_{f,j} \rangle$ using material derivatives results in (see Appendix C.1)

$$\begin{aligned} \alpha_f \rho_f \frac{\mathrm{D}_f}{\mathrm{D}t} \left[\langle u'_{f,i} u'_{f,j} \rangle \right] = &-\frac{\partial}{\partial x_k} \left[\alpha_f \rho_f \langle u'_{f,i} u'_{f,j} u'_{f,k} \rangle \right] \\ &- \alpha_f \rho_f \langle u'_{f,i} u'_{f,k} \rangle \frac{\partial \langle u_{f,j} \rangle}{\partial x_k} - \alpha_f \rho_f \langle u'_{f,j} u'_{f,k} \rangle \frac{\partial \langle u_{f,i} \rangle}{\partial x_k} + \alpha_f \rho_f \langle A_{f,i} u'_{f,j} + A_{f,j} u'_{f,i} \rangle \\ &+ \alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{ij} \right\rangle + S^{\langle u'_{f,i} u'_{f,j} \rangle}_{p \to f,ij}, \end{aligned} \tag{6.54}$$

where an analytical treatment for the source term $S^{\langle u'_{f,i} u'_{f,j} \rangle}_{p \to f,ij}$ is provided in [201]. In Eq. (6.54) the first term is the triple correlation of the fluctuating velocities, second and third are the production terms, fourth term is the fluid acceleration term that requires modeling and contains the effects of both pressure gradient and fluctuating viscous terms [233]. The product of diffusion tensor is where the dissipation tensor, ϵ_{ij}, should be modeled [233–235]. The triple correlation encountered in Eq. (6.54) is an unclosed term and the logical starting point for a closure expression would be its transport equation similar to the single phase counterparts, see [236], and hence Scott [206] derives the transport equations for these triple correlations.

6.6 Particle phase mean equations

In this section, particle field equations are presented which can be derived using the transport equations for a general property. In addition the significance of each term in

the resulting equations are discussed in detail.

6.6.1 Particulate phase continuity equation

Setting $\mathscr{H}_p = 1$ in Eq. (6.50) results in [201]

$$\frac{\partial}{\partial t}\left[\alpha_p \rho_p\right] + \frac{\partial}{\partial x_i}\left[\alpha_p \rho_p \left\langle u_{p,i}\right\rangle\right] = 0. \tag{6.55}$$

This equation is simply the mass balance equation for a particle phase without any inter-phase mass transfer or transfer due to coalescence and break–ups.

6.6.2 Particulate phase and seen momentum equations

Setting $\mathscr{H}_p = u_{p,i}$ results in (see Appendix C.2)

$$\alpha_p \rho_p \frac{\mathrm{D}_p \left\langle u_{p,i}\right\rangle}{\mathrm{D}t} = -\frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \left\langle u'_{p,i} u'_{p,j}\right\rangle\right] + \alpha_p \rho_p \left\langle A_{p,i}\right\rangle. \tag{6.56}$$

The first term on the RHS of Eq. (6.56) is the particle velocity Reynolds stress tensor which can be interpreted similar to the fluid Reynolds stress tensor as the momentum flux due to the fluctuating velocity. These terms require closures for which explicit transport equations are obtained. Second term is the effect of all acceleration terms such as drag force, effects of mean pressure gradient and any external body forces such as gravity [237]. Similarly setting $\mathscr{H}_p = u_{s,i}$ for the seen velocity, we have

$$\alpha_p \rho_p \frac{\mathrm{D}_p \left\langle u_{s,i}\right\rangle}{\mathrm{D}t} = -\frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \left\langle u'_{s,i} u'_{p,j}\right\rangle\right] + \alpha_p \rho_p \left\langle A_{s,i}\right\rangle + S^{\langle u_{s,i}\rangle}_{p\to s,i}, \tag{6.57}$$

where $S^{\langle u_{s,i}\rangle}_{p\to s,i}$ accounts for the two-way coupling between the phases.

6.6.3 Particle Mean Diameter

Setting $\mathscr{H}_p = \phi_p$ results in [201]

$$\alpha_p \rho_p \frac{\mathrm{D}_p \left\langle \phi_p\right\rangle}{\mathrm{D}t} = -\frac{\partial}{\partial x_i}\left[\alpha_p \rho_p \left\langle \phi'_p u'_{p,i}\right\rangle\right]. \tag{6.58}$$

6.6.4 Particle Velocity Reynolds Stresses

Setting $\mathscr{H}_p' = u'_{p,i}u'_{p,j}$ in Eq. (6.51) produces [201]

$$\alpha_p\rho_p\frac{\mathrm{D}_p}{\mathrm{D}t}\left[\left\langle u'_{p,i}u'_{p,j}\right\rangle\right] = -\frac{\partial}{\partial x_k}\left[\alpha_p\rho_p\left\langle u'_{p,i}u'_{p,j}u'_{p,k}\right\rangle\right] - \alpha_p\rho_p\left\langle u'_{p,i}u'_{p,k}\right\rangle\frac{\partial\left\langle u_{p,j}\right\rangle}{\partial x_k} - \alpha_p\rho_p\left\langle u'_{p,j}u'_{p,k}\right\rangle\frac{\partial\left\langle u_{p,i}\right\rangle}{\partial x_k} + \alpha_p\rho_p\left\langle A_{p,i}u'_{p,j} + A_{p,j}u'_{p,i}\right\rangle, \quad (6.59)$$

for the particulate phase Reynolds stresses. First term on the RHS of Eq. (6.59) represents the transport of the stress by the particle fluctuating velocity which is diffusive in nature. Second and third terms are the production by the mean particle velocity gradient. The last term represents the effects of particle dragging along the fluid turbulence. A_p coefficients are the drag forces and can be simply written as $\frac{1}{\tau_p}\left(u_{s,i} - u_{p,i}\right)$, where τ_p is the time scale of the particles which depends on the diameter of the particle. If the diameters were constant then it would be possible to write:

$$\alpha_p\rho_p\left\langle A_{p,i}u'_{p,j} + A_{p,j}u'_{p,i}\right\rangle = \alpha_p\rho_p\left\langle\frac{1}{\tau_p}\left(u_{s,i} - u_{p,i}\right)u'_{p,j} + \frac{1}{\tau_p}\left(u_{u,j} - u_{u,j}\right)u'_{p,i}\right\rangle = \frac{\alpha_p\rho_p}{\tau_p}\left[\left\langle\left(u'_{s,i} + \left\langle u_{s,i}\right\rangle\right)u'_{p,j}\right\rangle - \left\langle\left(u'_{p,i} + \left\langle u_{p,i}\right\rangle\right)u'_{p,j}\right\rangle + \left\langle\left(u'_{s,j} + \left\langle u_{s,j}\right\rangle\right)u'_{p,i}\right\rangle - \left\langle\left(u'_{p,j} + \left\langle u_{p,j}\right\rangle\right)u'_{p,i}\right\rangle\right] = -\frac{2\alpha_p\rho_p}{\tau_p}\left\langle u'_{p,j}u'_{p,i}\right\rangle + \frac{\alpha_p\rho_p}{\tau_p}\left[\left\langle u'_{s,i}u'_{p,j}\right\rangle + \left\langle u'_{s,j}u'_{p,i}\right\rangle\right]. \quad (6.60)$$

Using Eq.(6.60), Eq.(6.59) is reduced to the mono–dispersed equation derived by Simonin et al. [238] and Simonin et al. [239]. Evidently for the mono-dispersed systems there is no need to include the field equations for any correlation containing particle diameter. However here mixed moments of diameter and seen velocity, also diameter and particle velocity are needed to close the equations at the second order moment level.

6.6.5 Particle Velocity / Seen Velocity Correlation

By setting $\mathscr{H}'_p = u'_{s,i}u'_{p,j}$ we have [201]:

$$\begin{aligned}\alpha_p\rho_p\frac{\mathrm{D}_p}{\mathrm{D}t}\left[\left\langle u'_{s,i}u'_{p,j}\right\rangle\right] = &-\frac{\partial}{\partial x_k}\left[\alpha_p\rho_p\left\langle u'_{s,i}u'_{p,j}u'_{p,k}\right\rangle\right]\\ &-\alpha_p\rho_p\left\langle u'_{s,i}u'_{p,k}\right\rangle\frac{\partial\left\langle u_{p,j}\right\rangle}{\partial x_k}-\alpha_p\rho_p\left\langle u'_{p,j}u'_{p,k}\right\rangle\frac{\partial\left\langle u_{s,i}\right\rangle}{\partial x_k}\\ &+\alpha_p\rho_p\left\langle A_{s,i}u'_{p,j}+A_{p,j}u'_{s,i}\right\rangle+S_{p\to s,ij}^{\left\langle u'_{s,i}u'_{p,j}\right\rangle},\end{aligned}\tag{6.61}$$

where once again $S_{p\to s,ij}^{\left\langle u'_{s,i}u'_{p,j}\right\rangle}$ is due to the fluid–particle coupling. The first term in the RHS of Eq. (6.61) is the seen-particle velocity correlation transported by the means of particle fluctuating velocity. Second and third terms are the production terms due to mean particle and seen velocity gradient. The fourth term contains different effects which are easier to interpret in the case of mono-dispersed particulate flows. Using analysis similar to Eq. (6.60), it is possible to express the fourth term in Eq. (6.61) as a dissipation by the pressure-strain correlations $-\frac{1}{\tau_f}\left\langle u'_{s,i}u'_{p,j}\right\rangle$ where τ_f is the fluid Lagrangian integral time scale [238] and the following interphase momentum transfer term

$$-\frac{1}{\tau_p}\left(\left\langle u'_{s,i}u'_{p,j}\right\rangle-\left\langle u'_{s,i}u'_{s,j}\right\rangle\right),\tag{6.62}$$

which tends to bring the fluid-particle covariance, $\left\langle u'_{s,i}u'_{p,j}\right\rangle$, near to the fluid Reynolds stress tensor.

6.6.6 Particle Diameter / Particle Velocity Correlation

By setting $\mathscr{H}'_p = \phi'_p u'_{p,i}$ we have [201]

$$\begin{aligned}\alpha_p\rho_p\frac{\mathrm{D}_p}{\mathrm{D}t}\left[\left\langle\phi'_p u'_{p,i}\right\rangle\right] = &-\frac{\partial}{\partial x_j}\left[\alpha_p\rho_p\left\langle\phi'_p u'_{p,i}u'_{p,j}\right\rangle\right]\\ &-\alpha_p\rho_p\left\langle\phi'_p u'_{p,j}\right\rangle\frac{\partial\left\langle u_{p,i}\right\rangle}{\partial x_j}-\alpha_p\rho_p\left\langle u'_{p,i}u'_{p,j}\right\rangle\frac{\partial\left\langle\phi_p\right\rangle}{\partial x_j}+\alpha_p\rho_p\left\langle A_{p,i}\phi'_p\right\rangle.\end{aligned}\tag{6.63}$$

Again this equation contains the triple correlation that can be interpreted as the transport of the diameter-particle velocity correlation, $\left\langle\phi'_p u'_{p,i}\right\rangle$, by the means of the particle fluctuating velocity. Second and third terms are the again production terms by the mean field gradients. The last term is the mixed effects of particle drag and the fluctuations in its diameter.

6.6.7 Seen Velocity Reynolds Stresses

Eq. (6.61) contains the term $\left\langle A_{p,j}u'_{s,i}\right\rangle$ which in turn will involve the seen velocity Reynolds stresses $\left\langle u'_{s,i}u'_{s,j}\right\rangle$ for which the transport equation is written as follows [201]:

$$\begin{aligned}\alpha_p\rho_p\frac{\mathrm{D}_p}{\mathrm{D}t}\left[\left\langle u'_{s,i}u'_{s,j}\right\rangle\right] = &-\frac{\partial}{\partial x_k}\left[\alpha_p\rho_p\left\langle u'_{s,i}u'_{s,j}u'_{p,k}\right\rangle\right]\\ &-\alpha_p\rho_p\left\langle u'_{s,i}u'_{s,k}\right\rangle\frac{\partial\left\langle u_{s,j}\right\rangle}{\partial x_k}-\alpha_p\rho_p\left\langle u'_{s,j}u'_{s,k}\right\rangle\frac{\partial\left\langle u_{s,i}\right\rangle}{\partial x_k}\\ &+\alpha_p\rho_p\left\langle A_{s,i}u'_{s,j}+A_{s,j}u'_{s,i}\right\rangle+\alpha_p\rho_p\left\langle(B_sB_s^T)_{ij}\right\rangle+S_{p\to s,ij}^{\left\langle u'_{s,i}u'_{s,j}\right\rangle},\end{aligned}\tag{6.64}$$

where $S_{p\to s,ij}^{\left\langle u'_{s,i}u'_{s,j}\right\rangle}$, the coupling term, and other terms on the RHS of Eq. (6.64) can be interpreted similar to Eq. (6.54).

6.6.8 Particle Diameter / Seen Velocity Correlation

Eq. (6.63) contains the term $\left\langle A_{p,j}\phi'_p\right\rangle$ which in turn will involve the seen correlation $\left\langle\phi'_pu'_{s,i}\right\rangle$ and the transport equation for $\left\langle\phi'_pu'_{s,i}\right\rangle$ is [201]:

$$\begin{aligned}\alpha_p\rho_p\frac{\mathrm{D}_p}{\mathrm{D}t}\left[\left\langle\phi'_pu'_{s,i}\right\rangle\right] = &-\frac{\partial}{\partial x_j}\left[\alpha_p\rho_p\left\langle\phi'_pu'_{s,i}u'_{p,j}\right\rangle\right]\\ &-\alpha_p\rho_p\left\langle\phi'_pu'_{p,j}\right\rangle\frac{\partial\left\langle u_{s,i}\right\rangle}{\partial x_j}-\alpha_p\rho_p\left\langle u'_{s,i}u'_{p,j}\right\rangle\frac{\partial\left\langle\phi_p\right\rangle}{\partial x_j}\\ &+\alpha_p\rho_p\left\langle A_{s,i}\phi'_p\right\rangle+S_{p\to s,i}^{\left\langle\phi'_pu'_{s,j}\right\rangle}.\end{aligned}\tag{6.65}$$

The first term is again transport of the diameter-seen velocity correlation, $\left\langle\phi'_pu'_{s,i}\right\rangle$, by the particle velocity fluctuations, second and third terms are the production terms and the fourth is the mixed effects of fluid seen acceleration and diameter fluctuations. The source term in this equation is again is a result of fluid–particle coupling and simply accounts for the effects of fluctuations in the particle diameter on the seen velocity.

6.6.9 Particle Diameter Variance

The particle diameter variance does not appear in any of the previous equations as an unclosed term. However it is required to describe the polydispersed nature of the particulate phase. This variance transport equation is obtained by setting $\mathscr{H}'_p=\phi'_p\phi'_p$ as follows [201]

$$\alpha_p \rho_p \frac{\mathrm{D}_p}{\mathrm{D}t} \left[\left\langle \phi_p' \phi_p' \right\rangle \right] = -\frac{\partial}{\partial x_i} \left[\alpha_p \rho_p \left\langle \phi_p' \phi_p' u_{p,i}' \right\rangle \right] - 2\alpha_p \rho_p \left\langle \phi_p' u_{p,i}' \right\rangle \frac{\partial \left\langle \phi_p \right\rangle}{\partial x_i}. \tag{6.66}$$

6.7 Non–integer closure problem

Terms containing the drag force $A_{p,i}$ appear in Equations (6.56), (6.59), (6.61) and (6.64). These terms depending on the form of the drag law used, generate complicated non-integer moments. Considering Eq. (6.56) which is the simplest case, $\langle A_{p.i} \rangle$ can be expanded for a poly–sized particle population as follows

$$\langle A_{p,i} \rangle = \left\langle \frac{1}{\tau_p} (u_{s,i} - u_{p,i}) \right\rangle, \tag{6.67}$$

Where the particle response time is given by

$$\tau_p = \frac{\rho_p}{18\mu_f} \frac{\phi_p^2}{f_1}, \tag{6.68}$$

and $f_1 = 1$ for Stokes flow. Substituting Eq. (6.68) into Eq. (6.67) and assuming Stokes flow results in

$$\langle A_{p,i} \rangle = \frac{18\mu_f}{\rho_p} \left\langle u_{s,i} \phi_p^{-2} - u_{p,i} \phi_p^{-2} \right\rangle. \tag{6.69}$$

If the flow were mono–dispersed the particle diameters would be independent of integration variables and Eq. (6.69) would simplify to

$$\langle A_{p,i} \rangle = \frac{18\mu_f}{\rho_p \phi_p^2} \left\langle u_{s,i} - u_{p,i} \right\rangle, \tag{6.70}$$

and no further closures were required. If the assumption of Stokes drag (for $\mathrm{Re}_p > 1$) is to be further relaxed, a value for the parameter f_1 should be prescribed. A common relation in terms of particle Reynolds number for f_1 is given by the following equation [240]

$$f_1 = 1 + 0.15 Re_p^{0.687}, \tag{6.71}$$

Now substituting Eq. (6.71) into Eq. (6.67) results in

$$\langle A_{p,i}\rangle = \frac{18\mu_f}{\rho_p}[\langle (u_{s,i}-u_{p,i})\phi_p^{-2}\rangle + 0.15\left(\frac{\rho_f}{\mu_f}\right)^{0.687}\langle \phi_p^{-1.313}\|u_{s,i}-u_{p,i}\|^{0.687}(u_{s,i}-u_{p,i})\rangle], \quad (6.72)$$

which contains non–integer moments. Even more complicated closures are generated in Equations (6.59), (6.61) and (6.64) and closure of these moments in terms of available moments is not trivial. It is required for any proposed closure to be consistent with the presented stochastic framework and also to maintain the generality of the approach. If the form of the PDF is known any moment can directly be calculated by simple integrations over the state space. However the only information available after solving the transport equations is a set of integer moments. The reconstruction of a PDF from a set of moments and also other possible approaches is the subject of Chapter 7.

6.8 Conclusion

In this Chapter a mathematically rigorous framework to derive the EE field equations based on the SDEs and corresponding Fokker-Planck equation is reviewed. The mathematical framework and different fundamental stochastic processes are discussed in detail using several numerical experiments. The process of deriving field equations for a polydispersed mixture is reviewed and summarized from different sources. The field equations for up to second order moments are presented and an attempt has been made to physically interpret different terms appearing in these equations by comparing them to their single phase or mono–dispersed counterparts. Several terms containing unclosed terms are identified and the non–integer closure problem is explained in detail. Several proposals for this problem will be considered in the next chapter.

6. A REVIEW OF THE POLYDISPERSED EULERIAN–EULERIAN TURBULENT MODELS

7

Closure of non–Integer moments

In this chapter the closure problem for non-integer moments as discussed in Section 6.7 will be considered. In this chapter only one dimensional moments of form $\langle x^p \rangle$, $p \in \mathbb{R}$ are considered. It is worth mentioning that although the closure problem discussed in Section 6.7 is a multivariate problem in nature this preliminarily one–dimensional treatment can conceivably provide insight to the general multivariate case. In addition univariate case is encountered in multiphase flow systems if heat and mass transfer, chemical reactions, agglomeration or break-up are added to the equations and hence the problem is also of practical interest. Two different categories of methods are considered: the first method is based on the reconstruction of the underlying PDF using Laguerre polynomials and the other is based on the direct calculation of non-integer moments using the fractional derivatives of moment generating function (MGF). By applying the results of fractional calculus an explicit equation is derived to express non-integer moments as a function of any arbitrary number of integer moments. The proposed methods are tested on several highly non-Gaussian analytical PDFs in addition to experimental agglomeration data and direct numerical simulation of fluid-particle turbulent multiphase flows.

7.1 Introduction

The governing equations for many physical phenomena are precisely known, however numerical simulation by direct discretisation of the governing equations is not generally

feasible when several complicated phenomena such as turbulence, chemical reaction, agglomeration and break-up are involved. The starting point to tackle such complicated phenomena is usually to write the averaged equations. The process starts by defining an arbitrary probability density function (PDF) for the process and deriving an equation for the evolution of the PDF as discussed in Chapter 6. Rigopoulos [200] also discusses similar methods from an engineering point of view using the population balance equation (PBE) for reactive flows.

Non-integer moments can be introduced in the equations if the size distribution of particles is retained in Eulerian field equations (moment evolution) as discussed in the previous chapter. Also if Phenomena such as coagulation, nucleation, coalescence and breakup are included in the PBE [241–245] or heat and mass transfer considered [214, 246]. There are some ad-hoc solutions to the problem such as interpolation between the moments. Beck and Watkins [214, 246] used geometric interpolation and Frenklach and Harris [241] used Laguerre interpolation between the logarithms of the integer moments. Note that interpolation between the moments is not possible for negative fractional moments since usually only equations for positive moments are available. Fractional negative moments can also appear in the moment evolution equations as discussed in Section 6.7. PDF reconstruction methods are also attempted [242–244] which will be discussed in more detail below.

In this chapter the problem is examined from a mathematical point of view and the moments $\mu_p = \langle x^p \rangle$ of a general univariate PDF are considered. The moments are given by

$$\mu_p = \int_\Omega x^p \mathcal{P}(x) dx, \quad p \in \mathbb{R}, \tag{7.1}$$

where Ω is the domain on which x is defined. It is obvious that having the PDF any moment can readily be calculated, however in many physical phenomena it is easy to determine the moments but it is extremely difficult to determine the distributions themselves [247]. In addition during a numerical simulation the only information available are the solved variables and in this case the solved variables are the first few integer moments not the evolution equation of the PDF itself. Thus a method is required to enable us to estimate the real order moments $\mu_p,\ p \in \mathbb{R}$, using limited number of integer moments $\mu_i,\ i = 1, 2, \cdots, n$. To accomplish this two general methods will be contemplated: A PDF reconstruction method based on Laguerre polynomials and a direct

fractional method of moments (DFMM) based on derivatives of the moment generating function (MGF).

The first obvious approach is to try to reconstruct the PDF using integer moments μ_i which leads to the well-known finite moment problem [247] and can be regarded as a finite dimensional version of the Hausdorff moment problem [248–251] which is in general ill-posed lacking one or more conditions of a well posed problem (i.e existence, uniqueness and stability) [252]. The reconstruction methods can be classified under linear and non-linear methods as discussed by Volpe and Baganoff [253]. They classified the maximum entropy method (MEM) as a non-linear method and reconstruction methods based on expansion around some parent distribution as linear methods.

The MEM was first utilized by Koopman [254] based on the idea of information theory of Shannon [255]. The MEM (Koopman) method is attractive for several reasons [256]: (i) Sound conceptual foundations; (ii) interdependence between even and odd order moments; (iii) non-negative probabilities, (iv) produces the most unbiased distribution possible, i.e if a distribution with less entropy (uncertainty) were used that would imply the existence of additional knowledge [212]. The last property of MEM (i.e most unbiased distribution) might be interesting in some applications but for the current problem where only the first few moments (usually 2) are available, the method effectively only generates Gaussian PDFs.

PDF reconstruction based on some priori simple shape is the other option. However this method has a global realizability issue, in that an assumed PDF, valid at one location in space and time may evolve into a form that violates the assumption of the PDF at another position [214]. A more advanced method in this category is expansion using orthogonal polynomials. Two basic methods in this category which are extensively used in the literature are Gram-Charlier and Edgeworth series expansion [257]. In both methods the PDF $\mathcal{P}$ is evaluated using a truncated expansion in terms of Hermite's polynomials $H_n(x)$:

$$\mathcal{P}(x) = \sum_{n=1}^{N} C_n H_n(x) \mathcal{N}(\mu, \sigma), \tag{7.2}$$

where $\mathcal{N}(\mu, \sigma)$ is a Gaussian distribution with parameters μ and σ, C_n are the coefficients containing the higher order moments (Gram-Charlier) or higher order cumulants

(Edgeworth) and H_n is the Hermite polynomial which can explicitly be written [258] by

$$H_n(x) = n! \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{(-1)^k x^{n-2k}}{k!(n-2k)!2^k}. \tag{7.3}$$

Majumdar et al. [259] among others [260, 261] discussed that both methods can be divergent due to the fact that the series expansion is sensitive to the behavior of $\mathcal{P}(x)$ at infinity. For the series to be convergent $\mathcal{P}(x)$ should fall faster to zero than $e^{-x^2/4}$. Generalized Laguerre polynomial expansion is the more promising approach in this category which does not have any oscillatory or divergence problems of Edgeworth or Gram-Charlier expansions [259]. Although Laguerre polynomial expansion has been used in the literature to reconstruct different types of PDFs, its accuracy in estimating non-integer moments encountered in multiphase flow systems has not been tested and will be discussed in this chapter. There are other methods which cannot be considered general such as spline reconstruction method proposed by John et al. [247] which fits cubic splines to the available moments however for the method to be effective usually a large number of moments are required.

The methods discussed above are all indirect methods i.e first a PDF is reconstructed using one of the discussed methods then the PDF should numerically be integrated (Eq. (7.1)) to calculate the required moments. In this chapter also a direct method using fractional derivatives of the MGF is formulated. The property of the moment generating function is that its kth derivative calculated at $s = 0$ is equal to the kth moment of the distribution $\mathcal{P}(x)$ [262]:

$$\mu_k = \left. \frac{d^k G}{ds^k} \right|_{s=0}. \tag{7.4}$$

Now if this equation could be extended to non positive non integer values of k then it would provide a way of expressing non integer moments as a function of moment generating function. This extension is formalized in the field of fractional calculus and will be discussed in Section 7.3. Then several test cases will be discussed using moments of analytical PDFs, moments of agglomeration phenomena measured experimentally and moments extracted from direct numerical simulation of isotropic fluid-particle turbulent systems.

7.2 Generalized Laguerre polynomial expansion

The Laguerre expansion of a PDF $\mathcal{P}$ can be written by [259, 263]

$$\mathcal{P}(x) = \sum_{n=0}^{\infty} r_n L_n^a(bx)\mathcal{G}(x; a+1, b), \tag{7.5}$$

where L_n^a is the generalized Laguerre polynomial with parameter a. $\mathcal{G}(x; a, b)$ is the Gamma distribution with parameters a and b given by

$$\mathcal{G}(x; a, b) = \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx), \quad x \geq 0,\ a, b > 0, \tag{7.6}$$

where $\Gamma(a)$ is the gamma function given by

$$\Gamma(a) = \int_0^{\infty} t^{a-1} \exp(-t) dt. \tag{7.7}$$

If the argument is an integer then $\Gamma(a) = (a-1)!$. The function can be approximated by directly calculating the integral which is not a stable method and a better approach would be the approximation method proposed by Cody [264] which is adopted for the current study. Generalized Laguerre polynomials are orthogonal with respect to the measure $\mathcal{G}$ therefore using the Rodriguez formula [265], L_n can be defined by

$$L_n^a(x)\mathcal{G}(x; a+1, 1) = \frac{1}{n!}(\frac{d}{dx})^n[x^n \mathcal{G}(x; a+1, 1)]. \tag{7.8}$$

Using the Leibniz's theorem for differentiation of product it can explicitly be written by [265]

$$L_n^a(x) = \sum_{m=0}^{n} \binom{n+a}{n-m} \frac{(-x)^m}{m!}, \tag{7.9}$$

where the binomial coefficients are generalized using the gamma function.

The coefficients r_n can be found using the orthogonality of the Laguerre polynomials. A sequence of polynomials $Q_1(x), \cdots, Q_n(x)$ are said to be orthogonal on the interval $[x_1, x_2]$ with respect to a weight function $W(x)$ if

$$\int_{x_1}^{x_2} Q_n Q_m W(x) dx = 0 \quad \forall m \neq n. \tag{7.10}$$

For Laguerre polynomials it can easily be shown that

$$\int_0^\infty L_n^a(bx)L_m^a(bx)\mathcal{G}(x;a+1,b)dx = \frac{\Gamma(n+a+1)}{n!\Gamma(a+1)}\delta_{nm}, \tag{7.11}$$

where $\delta_{nm} = 1$ if $m = n$ and is 0 otherwise. Now multiplying the right and left hand side of Eq.(7.5) by $L_m^a(x)$ and using Eq.(7.11) results in

$$r_n = \frac{n!\Gamma(a+1)}{\Gamma(n+a+1)}\int_0^\infty \mathcal{P}(x)L_n^a(bx)dx. \tag{7.12}$$

By introducing Eq.(7.9) into Eq.(7.12) and using the definition of Γ function and Eq.(7.1) we have

$$\begin{aligned} r_n &= \frac{n!\Gamma(a+1)}{\Gamma(n+a+1)}\int_0^\infty \mathcal{P}(x)\sum_{i=0}^{n}\binom{n+a}{n-i}\frac{(-bx)^i}{i!}dx \\ &= n!\Gamma(a+1)\sum_{i=0}^{n}\frac{(-b)^i\mu_i}{i!(n-i)!\Gamma(a+i+1)}. \end{aligned} \tag{7.13}$$

In this equation μ_i are the available moments of order i. The only remaining issue is the calculation of the parameters a and b which are defined such that the expansion (7.5) has the same mean and variance as the original distribution $\mathcal{P}(x)$. A simple derivation of these parameters can be found in [263] and are given by

$$a = \frac{2\mu_1^2 - \mu_2}{\mu_2 - \mu_1^2}, \quad \text{and} \quad b = \frac{\mu_1}{\mu_2 - \mu_1^2}. \tag{7.14}$$

7.3 Direct fractional method of moments

In this section an approach to estimate fractional moments as a series of the integer moments is formulated. The MGF of a positive valued distribution can be defined by [262]:

$$G(s) = \int_0^\infty \mathcal{P}(u)e^{su}du. \tag{7.15}$$

The main property of the MGF is that the moments of the original PDF $\mathcal{P}(x)$ can be derived from this function by Eq. (7.4). Extension of this equation to non-integer values will follow next.

7.3.1 Fractional Derivatives and Integrals

The derivative of arbitrary real order p can be considered as an interpolation to the operators of a sequence of n-fold integration and nth order derivative. Fractional derivatives are presented by ${}_aD_t^p(t)$ where a and t are the limits related to the operation of the fractional differentiation and are commonly called terminals of the fractional differentiation. These are essential to avoid ambiguities in application of fractional derivative to real numbers [266–268]. There are two equivalent approaches to the definition of fractional differentiation namely the Grunwald-Letnikov (GL) approach and the Rienmann-Liouville (RL) approach. It is customary [267] to use the RL formulation for problem setup and use GL approach to obtain a numerical solution. Following the same approach RL definition is used to establish the relation between the evaluation of the moments and fractional calculus and GL approach is used to provide an explicit equation for fractional moments as a function of integer moments.

7.3.2 RL fractional derivative

Consider the integral

$$f^{-1}(t) = \int_a^t f(\tau)d\tau, \tag{7.16}$$

and define

$$f^{-2}(t) = \int_a^t d\tau_1 \int_a^{\tau_1} f(\tau)d\tau. \tag{7.17}$$

It can then be shown that, see Appendix D.1,

$$f^{-2}(t) = \int_a^t (t-\tau)f(\tau)d\tau. \tag{7.18}$$

Then by the method of induction we have

$$f^{-n}(t) = \frac{1}{(n-1)!}\int_a^t (t-\tau)^{n-1}f(\tau)d\tau, \quad n \geq 1. \tag{7.19}$$

This can be extended to non-integer values of n using the Gamma function:

$${}_aD_t^{-p}f(t) = \frac{1}{\Gamma(p)}\int_a^t (t-\tau)^{p-1}f(\tau)d\tau, \quad p > 0. \tag{7.20}$$

The fractional derivative can then be defined by [269, 270]

$$ {}_aD_t^p f(t) = \frac{1}{\Gamma(k-p)} \left(\frac{d}{dt}\right)^k \int_a^t (t-\tau)^{k-p-1} f(\tau) d\tau, \quad k-1 \leq p \leq k. \tag{7.21} $$

Note that this definition is not arbitrary and it is defined to be equivalent to the GL definition which is defined as an extension to backward differences. Having the differentiation extended to arbitrary order, it can be shown (Appendix D.2),

$$ \mu_p = {}_{-\infty}D_0^p G(0), \tag{7.22} $$

where μ_p is the moment of the density $\mathcal{P}(x)$ of order p. Although the RL definition of fractional derivatives and integrals is more convenient to mathematically link the fractional moments to fractional derivatives it does not provide a numerical method for calculating the moments. However by reverting to the GL definition an expression can be written for the fractional moments.

7.3.3 GL fractional derivatives

GL definition of fractional derivatives is more intuitive and starts by observing the series of backward differences and writing a general series for derivatives of order n:

$$ f^{(n)}(t) = \frac{d^n f}{dt^n} = \lim_{h\to 0} \frac{1}{h^n} \sum_{r=0}^{n} (-1)^r \binom{n}{r} f(t-rh). \tag{7.23} $$

Then by considering the following generalization

$$ f_h^{(p)}(t) = \frac{1}{h^p} \sum_{r=0}^{n} (-1)^r \binom{p}{r} f(t-rh), \tag{7.24} $$

for arbitrary natural numbers p and n, such that $p \leq n$, we have

$$ \lim_{h\to 0} f_h^p(t) = f^{(p)}(t) = \frac{d^p f}{dt^p}, \tag{7.25} $$

because all the coefficients in the numerator after $\binom{p}{p}$ are identically zero. Eq. (7.24) can be inverted (see Appendix D.3) to get

$$ f(t-ih) = \sum_{r=0}^{i} (-1)^r \binom{i}{r} h^r f^{(r)}(t). \tag{7.26} $$

Eq. (7.26) is the key for writing the non-integer moments as functions of integer moments and it will become clear in the course of this section. To extend the definition to negative values a careful definition of the upper bound of the summation is required otherwise the limit in Eq. (7.24) would be strictly zero for any n. Therefore by taking $h = \frac{t-a}{n}$, a being a real number, we can define

$$\lim_{\substack{h\to 0 \\ nh=t-a}} f_h^{(-p)}(t) = {}_aD_t^{-p} f(t). \tag{7.27}$$

Note that Eq. (7.27) is actually the definition of an integral, for example writing the series for $p = 1$ the limit is simply the definition of $\int_a^t f(\tau)d\tau$. By the method of induction it is possible to show [267, 268]

$${}_aD_t^{-p} f(t) = \lim_{\substack{h\to 0 \\ nh=t-a}} h^p \sum_{r=0}^{n} (-1)^r \binom{p}{r} f(t-rh) = \frac{1}{(p-1)!}\int_a^t (t-\tau)^{p-1} f(\tau)d\tau, \tag{7.28}$$

and consequently [267]

$${}_aD_t^{-p} f(t) = \underbrace{\int_a^t d\tau_1 \int_a^{\tau_1} d\tau_2 \cdots \int_a^{\tau_{p-1}}}_{p \text{ times}} f(\tau_p)d\tau_p. \tag{7.29}$$

Therefore

$${}_aD_t^{p} f(t) = \lim_{\substack{h\to 0 \\ nh=t-a}} h^{-p} \sum_{r=0}^{n} (-1)^r \binom{p}{r} f(t-rh), \tag{7.30}$$

is indeed a general expression for p-fold integration and derivatives of order n. Eq. (7.30) can be extended to non-integer p values. By direct calculation of the limits it can be shown that [267, 270]

$${}_aD_t^{-p} f(t) = \frac{1}{\Gamma(p)}\int_a^t (t-\tau)^{p-1} f(\tau)d\tau, \tag{7.31}$$

and

$$\begin{aligned} {}_aD_t^{p} f(t) = &\sum_{r=0}^{m} \frac{f^{(r)}(a)(t-a)^{-p+r}}{\Gamma(-p+r+1)} \\ &+ \frac{1}{\Gamma(-p+m+1)}\int_a^t (t-\tau)^{m-p} f^{(m+1)}(\tau)d\tau, \quad \forall m > p-1. \end{aligned} \tag{7.32}$$

It is important to note the equivalence between Eq. (7.20) and Eq. (7.31). The equivalence between Eq. (7.32) and Eq. (7.21) is harder to note but direct differentiation and integration by parts shows that both definitions are indeed equivalent [268]. Having the framework established an equation will be derived to explicitly write the non-integer moments as a function of integer moments in the next section.

7.3.4 Estimating the non-integer moments

A first order approximation to p-order derivative using the GL definition can be written as

$$ {}_aD_t^p f(t) = \lim_{h\to 0} h^{-p}\, {}_a\Delta_t^p f(t) \approx h^{-p} \sum_{r=0}^{\lfloor \frac{t-a}{h} \rfloor} (-1)^r \binom{p}{r} f(t-rh). \tag{7.33} $$

The number of addends in Eq. (7.33) becomes very large for $t \gg a$. This series can be truncated using the short memory principle; taking into account the behavior of $f(t)$ only in the recent past. This means that the truncated series (7.33) is in particular a very good approximation for calculating fractions that are approximately equal to the number of integer moments retained in the expansion and the error can be quantified as suggested by Deng [271]. In Eq. (7.33), $f(t-rh)$ can be considered as the MGF but since the function is not explicitly available the equation is not usable in this form. Equation (7.26) provides an equation for this term, now inserting Eq. (7.26) into Eq. (7.33) noting the definitions (7.22) and (7.4) we get

$$ \mu_p \approx \sum_{r=0}^{N} (-1)^r \binom{p}{r} \sum_{j=0}^{r} (-1)^j \binom{r}{j} h^{j-p} \mu_j, \tag{7.34} $$

where N is the number of moments retained in the expansion. A similar equation is recently derived by Gzyl and Tagliani [272] using Taylor expansions and also by Alexiadisa et al. [273] using Weyl fractional derivatives. However they applied the series to larger number of integer moments ($O(100)$ and $O(10)$ respectively) which is not applicable to the problems considered in this study. The only remaining issues are to provide an equation to calculate the coefficient $w_r^p = (-1)^r \binom{p}{r}$ and propose a step size h. The coefficients, w_r^p, are generalizations of binomial coefficients to non-integer values and can be calculated using the definition of the gamma function. However another possible approach is to use the recursive relations [267]

$$w_0^p = 1, \quad w_r^p = (1 - \frac{p+1}{r})w_{r-1}^p, \quad r = 1, 2, \cdots \quad , \tag{7.35}$$

which eliminates the need for the evaluation of gamma functions and provides better computational efficiency and stability. In addition it provides a unique opportunity for higher order estimations of the fractional derivative which is discussed next. Considering the coefficients of power series for the function $(1 - z)^p$:

$$(1 - z)^p = \sum_{r=0}^{\infty} w_r^p z^r. \tag{7.36}$$

Substituting $z = e^{-i\theta}$ coefficients can be expressed in terms of Fourier transforms:

$$w_r^p = \frac{1}{2\pi i} \int_0^{2\pi} (1 - e^{-i\theta})^p e^{ik\theta} d\theta. \tag{7.37}$$

Since always only a finite number of moments are used any FFT library can be used to calculate the coefficients using Eq. (7.37). Equations (7.35) and (7.37) are only first order approximations however higher order approximations can be constructed using higher order polynomials. Lubich [274] suggests polynomials up to 6th order which can be used in conjunction with Eq. (7.37) and an FFT library to find coefficients for the higher order approximation to the fractional derivatives. However only first order approximation (Eq. (7.35)) is used in this study due to its simplicity and efficiency.

The step size h should ideally be very small. Prescribing a value for $H = h^{-1}$ is equivalent to the width of the PDF considered in the calculations. Gzyl and Tagliani [272] showed that the series is always convergent for $1/2 < h \leq 1$ or $1 \leq H < 2$ however as will be discussed in the next section this value causes a severe under-estimation when only a few moments are used. It seems to be reasonable to relate H to the mean and the variance of the available data:

$$H = \mu + \lambda\sigma, \tag{7.38}$$

where μ is the mean of the data, σ is the standard deviation and λ is an adjustable parameter. Relating the step size to the statistics of the data was first explained in [267] and later used by [273] for the simulation of agglomeration problem using the method of moments (MOM) to solve the PBE however different forms with different parameters are possible. To the authors experience it seems to be valid that if the fractional moment

of interest is not much larger than the number of moments retained in the series setting $\lambda = \frac{1}{p+1}$ provides very accurate estimates. This hypothesis ($\lambda = \frac{1}{p+1}$) will be elaborated further in the next section where discussing the results.

7.4 Results and Discussions

7.4.1 Log-normal distribution

To test the results an analytic Log-normal distribution given by

$$\mathcal{P}(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{(\log x - \bar{\mu})^2}{2\bar{\sigma}^2}}, \tag{7.39}$$

is considered. In Eq. (7.39), $\bar{\sigma}$ and $\bar{\mu}$ are scale and location parameters respectively. Higher order moments are analytically given by

$$\mu_j = e^{(\frac{1}{2}j^2\sigma^2 + j\bar{\mu})}. \tag{7.40}$$

Figure 7.1 shows the simulation of the log-normal distribution using the Laguerre polynomials where the PDF is reconstructed for 200 equidistant points in range $[0, 12]$. Although our goal is to simulate the non-integer moments the accuracy of the calculated moments are directly proportional to the accuracy of the fit. Generally a Laguerre polynomial is able to produce a very good fit to this distribution with a limited number of moments because of the similarity between the shape of gamma and log-normal distributions.

Figure 7.2 shows the fractional moments $\mu_p,\ p = \{1.2, 2.3, 3.4, 4.5\}$ calculated using the fractional moments approach and the Laguerre polynomial approach. Intentionally only the first two moments (normalized on $[0, 12]$) are used to test the ability of the two proposed methods in estimating the non-integer moments using only limited data. In this section all errors are calculated by

$$Err = \frac{|\mu_a - \mu_e|}{\mu_a}, \tag{7.41}$$

where subscripts a and e stand for the analytical and estimated values respectively and also $\%Err = 100 \times Err$. In Figure 7.2 error bars, for $\pm 100 \times Err$, are also provided with maximum error of 27.5% in calculating the $\mu_{4.5}$ using the fractional moments method

with all other errors being less than 5%. The reason for this large error will be discussed in the next section and a remedy will be suggested.

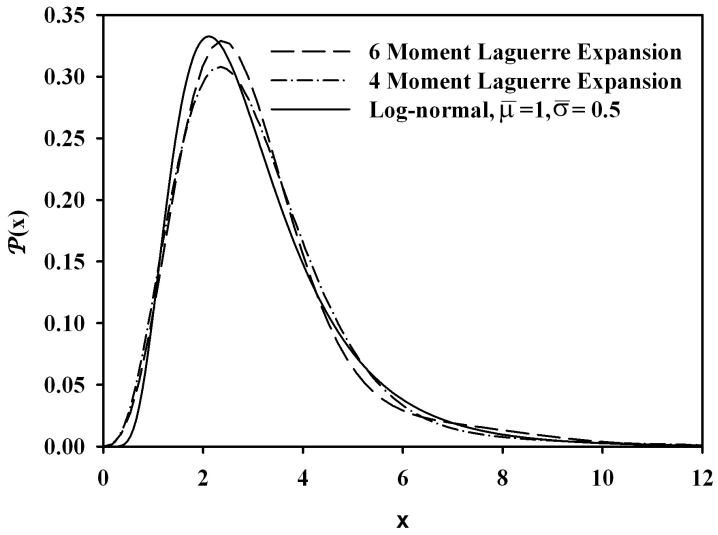


Figure 7.1: Log-normal distribution - scale and position parameters are0.5 and 1 respectively. This distribution can accurately be reproduced using small number of moments.

7.4.2 Mixture of normal distributions

A mixture of normal distributions can be created using

$$\mathcal{P}(x) = \sum_{i=1}^{N} w_i \mathcal{N}(\mu_i, \sigma_i), \quad \sum_{i-1}^{N} w_i = 1, \tag{7.42}$$

where $\mathcal{N}(\mu, \sigma)$ is the normal distribution with mean μ and variance σ. The following mixture distribution with two terms is considered

$$\mathcal{P}(x) = \alpha(0.8\mathcal{N}(0.5, 1) + 0.2\mathcal{N}(4, 1))\mathbb{1}_{[0,8]}, \tag{7.43}$$

where $\mathbb{1}_A$ is the box function which is equal to 1 if $x \in A$ and is zero otherwise. The parameters are chosen such that the final mixture be a bimodal distribution, see Behboodian [275] and Schilling et al. [276] for the necessary and sufficient conditions. The normalization constant, $a = 1.327743884718793$, ensures $\int_0^8 P(x)dx = 1$. First 12 moments of this function are given in Table 7.1 which are calculated by direct integration

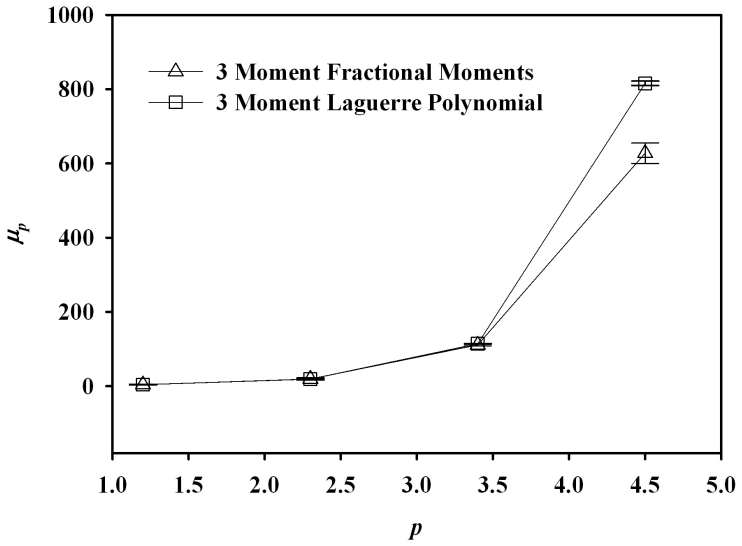


Figure 7.2: Non–integer moment estimation - LPM and DFMM methods are used and error bars are presented for each calculation.

using adaptive quadrature method [277] with relative and absolute tolerance of $1e-8$ and $1e-15$ respectively. Note that the PDF is normalized and therefore $\mu_0 = 1$.

Moment	Value	Moment	Value
μ_1	1.803323915192423	μ_2	5.618825039917679
μ_3	22.21194221994151	μ_4	98.56381003061070
μ_5	467.7931304312276	μ_6	2326.755938195312
μ_7	12010.60922300377	μ_8	63991.84370378382
μ_9	350667.3348484823	μ_{10}	1971324.861855061
μ_{11}	11345464.98539287	μ_{12}	66731974.28785729

Table 7.1: First 12 moments of mixture Gaussian distribution.

Figure 7.3 shows the simulation of the mixture distribution using the Laguerre polynomials for 200 discrete points. More moments are needed in this example to correctly capture the tail of the distribution.

Table 7.2 shows the fractional moments of the mixture Gaussian distribution using Laguerre polynomials method (LPM) and DFMM. DFMM estimates the fractional moments particularly well as long as the number of integer moments is near the value

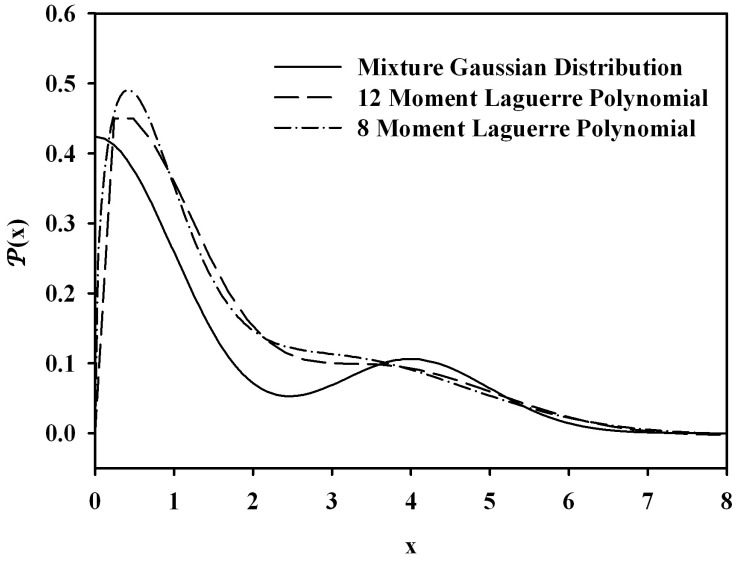


Figure 7.3: A mixture normal distribution - parameters in Eq. (7.43) are chosen such that the distribution is actually a bimodal distribution. The PDF is then reconstructed using 8 and 12 moments.

of the fractional moments. For example using first 3 integer moments the estimated values for $\mu_{1.2}, \mu_{2.3}, \mu_{3.4}$ are particularly precise with maximum error of 5.2% for $\mu_{3.4}$ which are better than the estimates provided by LPM. However for $\mu_{4.5}$ a large error is detected using the DFMM. Same calculations are performed using the first 5 integer moments and the results are listed in Table 7.3. In this example very precise estimates are provided using DFMM which are all better than those calculated using LPM.

Moment	Value	LPM	DFMM	LPM %Err	DFMM %Err
$\mu_{1.2}$	2.1974	2.1093	2.1333	4.0093	2.9171
$\mu_{2.3}$	8.3418	7.5806	8.5486	9.1249	2.4793
$\mu_{3.4}$	39.9035	37.1028	37.8250	7.0187	5.2088
$\mu_{4.5}$	213.2800	219.6070	140.6065	2.9665	34.0742

Table 7.2: Fractional Moments estimated using 3 integer moments (μ_0, μ_1, μ_2).

In the LPM case increasing the number of moments to 5 actually increases the error. The test with 7 and 9 moments is performed and a mild oscillatory convergence

Moment	Value	LPM	DFMM	LPM $\%Err$	DFMM $\%Err$
$\mu_{1.2}$	2.1974	2.2203	2.1804	1.0431	0.7741
$\mu_{2.3}$	8.3418	8.7174	8.3543	4.5029	0.1501
$\mu_{3.4}$	39.9035	45.4930	39.9454	14.0075	0.1050
$\mu_{4.5}$	213.2800	271.8476	209.9239	27.4604	1.5736

Table 7.3: Fractional Moments of a mixture Gaussian distribution, estimated using 5 integer moments $(\mu_0, \mu_1, \mu_2, \mu_3, \mu_4)$

is detected which is not reported in other studies using Laguerre series. On the other hand DFMM is based on a firm mathematical ground with predictable behavior which is a direct consequence of the short memory principle discussed in Section 7.3.4. Parameter λ can easily be adjusted to provide better results, for example setting $\lambda = \frac{2}{p+1}$ results in $\mu_{4.5} = 209.9239$ with $\%Err = 1.57$, see Table 7.3.

7.4.3 Rice-Nakagami distribution

A Rice-Nakagami distribution can be written by [278]:

$$\mathcal{P}(x; I_c, \sigma) = \frac{1}{\sigma^2} \exp\left(-\frac{x + I_c}{\sigma^2}\right) I_0\left(2\frac{\sqrt{xI_c}}{\sigma^2}\right), \tag{7.44}$$

where σ is the standard deviation and $I_c = \mu - \sigma^2$, with μ being the mean. I_0 is the zeroth order modified Bessel function of the first kind. Higher order moments are analytically given by [279]:

$$\mu_j = \sigma^{2j} \exp\left(-\frac{I_c}{\sigma^2}\right) \Gamma(j+1) \mathcal{M}\left(j+1, 1, \frac{I_c}{\sigma^2}\right), \tag{7.45}$$

where $\mathcal{M}$ is the confluent hyper-geometric function (Kummar function). Parameters I_c and σ^2 are set to 1 and 1.9 respectively. Figure 7.4 shows the reconstructed PDF (Eq. (7.44)) in $[0, 12]$ for 200 discrete points with the normalization constant calculated to be 0.99994848708433 on this interval. Evidently a Laguerre expansion can capture the features of this distribution with very high accuracy even with very limited number of moments. Note that only the first 3 and 5 moments are used to produce Figure 7.4 whereas in Figures 7.1 and 7.3 a larger set of moments was used. Figure 7.5 shows the exact values of the fractional moments and the values calculated using LPM and DFMM methods. Since this distribution can accurately be reconstructed using LPM,

very accurate estimations for the fractional moments can be achieved with errors never exceeding 2.5%. Despite this DFMM still produces acceptable results with the maximum error of 22% in calculating $\mu_{2.3}$ and error for $\mu_{1.2}, \mu_{3.4}, \mu_{4.5}$ being in the same range as those calculated by LP method.

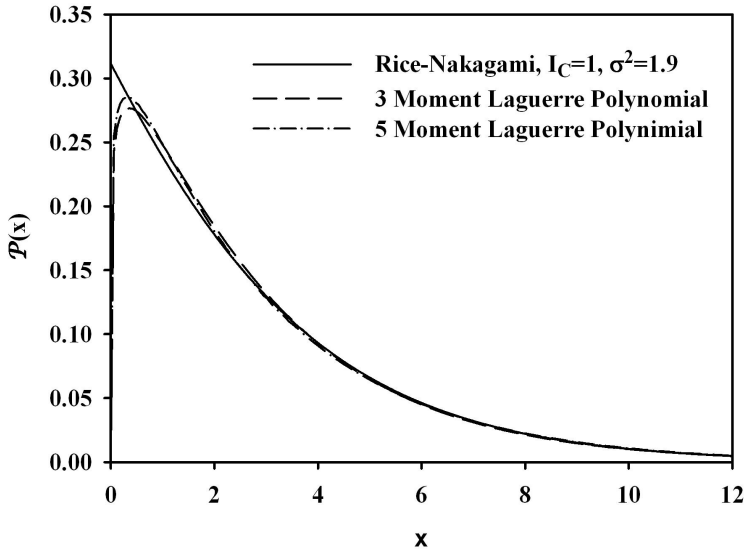


Figure 7.4: Rice-Nakagami distribution - reconstructed on 200 points using 3 and 5 moments.

Estimation of the fractional moments using a constant value as suggested by Gzyl and Tagliani [272] is also attempted. The results are plotted in Figure 7.5 and the only reliable results are those calculated between the available integer moments, i.e $\mu_{1.2}, \mu_{2.3}$. The values for $\mu_{3.4}, \mu_{4.5}$ are severely under-estimated and practically unusable. Note also that in this example the maximum of the suggested interval is used, i.e. $H = 2$, using smaller values causes even larger under-estimations. However it should be stated that the adaptive value for H is selected to work with very small number of integer moments, $3 \leq N \leq 5$, which has practical applications, for example, when solving Eulerian field equations. Therefor if a larger number of integer moments are available in the model one should use a more conservative value for H or include the number of integer moments, N, in the definition of λ.

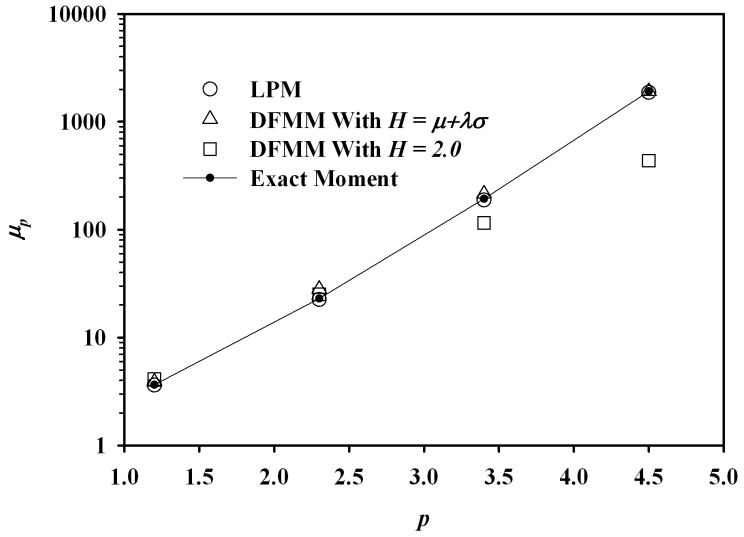


Figure 7.5: Fractional moments of the Rice-Nakagami distribution - the moments are calculated using LPM and DFMM methods using both an adaptive and constant value for H.

7.4.4 Crystallization process: experimental data

In this section experimental data of Oncul et al. [280], see also John et al. [247], for a crystallization process are used to further test the two proposed methods. The moments of the particle mass distribution (PMD) are provided in Table 7.4. Figure 7.6 shows the reconstructed PMD using 11 moments by Laguerre polynomials which cannot capture the first sharp peak accurately. Only the experimental moments are provided in [247, 280] and not the PMD data. The PDM is needed to analytically calculate the fractional moments and compare them to the values estimated either by the LPM or DFMM. The *exact* values of the fractional moments are calculated using MATLAB and the following graphical procedure: the graph of the data is sampled using 10^3 points, these digitized points are then re-sampled using cubic spline interpolation to produce a fine uniform sample of 5000 points and then integrated to get the fractional moments. Comparison of the integer moments available in the paper with recalculated integer moments from the aforementioned process shows that the error is always less that 1% for the first five moments and therefore the same process is used to calculate the fractional moments $\mu_{1.2}, \cdots, \mu_{4.5}$ assuming the calculated values from this integration procedure to be exact.

Table 7.5 shows the results of fractional moments calculation using only first 3 moments. The LP method produces large errors for $\mu_{4.5}$ since the method is not able to capture the sharp peak in the distribution. DFMM still produces acceptable results which are better than the estimates produced by LP method. Note that $\lambda = \frac{1}{1+p}$ is used for $p \in 1.2, 2.3, 3.4$ and $\lambda = \frac{2}{1+p}$ for $p = 4.5$.

Moment	Value	Moment	Value
μ_0	1.028571810801850e-02	μ_1	1.627619528771249e-05
μ_2	3.731662761082810e-08	μ_3	1.005399673163290e-10
μ_4	2.961317428763996e-13	μ_5	9.324232740417577e-16
μ_6	3.106813835529885e-18	μ_7	1.087507001932166e-20
μ_8	3.973765578013985e-23	μ_9	1.507036647558311e-25
μ_{10}	5.901352427828747e-28	μ_{11}	NA

Table 7.4: Moments of experimental PMD - First 11 moments of particle mass distribution (PMD) measured by [280]

Moment	Value	LPM	DFMM	LPM $\%Err$	DFMM $\%Err$
$\mu_{1.2}$	4.6577E-06	4.4901E-06	4.6738E-06	3.5979	0.3461
$\mu_{2.3}$	6.0937E-09	5.6016E-09	6.3647E-09	8.0759	4.4469
$\mu_{3.4}$	9.3137E-12	8.2381E-12	1.0012E-11	11.5492	7.4976
$\mu_{4.5}$	1.5563E-14	1.0276E-14	1.8520E-14	33.9737	18.9984

Table 7.5: Fractional moments of experimental PMD estimated using first 3 integer moments (μ_0, μ_1, μ_2)

7.4.5 Polydispersed fluid-particle system: DNS simulation

A pseudo-spectral DNS code capable of two-way coupled fluid-particle, homogeneous turbulence simulations, 'PANDORA' [3, 206], is used in this section to provide the particle distribution data. Two sample simulations were used with 32^3 and 128^3 grid points equivalent to a Taylor Reynolds number, Re_λ, of 24.24 and 83.40 respectively. More details on these simulation can be found in [3]. Here E_s is the enstrophy sampled along the particle path and $\langle e \rangle$ is the mean fluid enstrophy. The problem and related simulations is not discussed in more detail here and the PDFs in this section can simply

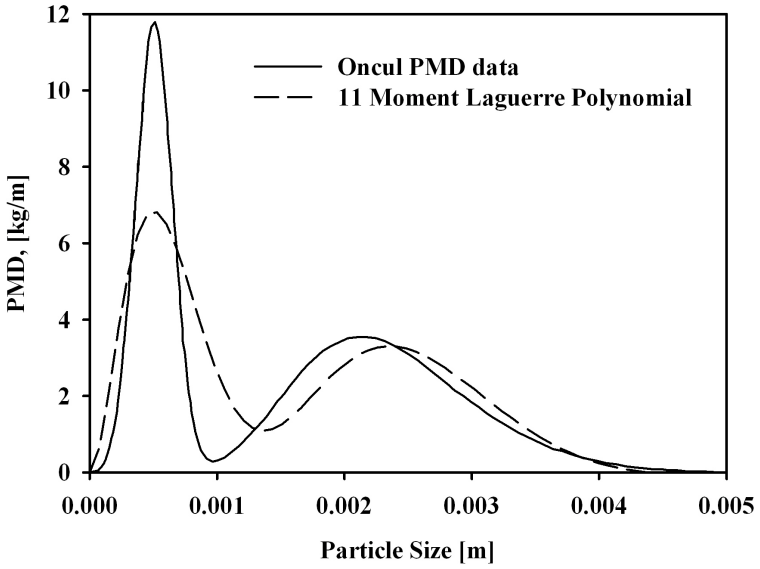


Figure 7.6: Reconstruction of PMD - Laguerre polynomials are used with 11 moments to reconstruct the experimental PDM of Oncul et al. [280]

be considered as some form of conditional distributions of a globally uniform particle size distribution. A deviation in the conditional PDFs from a globally uniform distribution is observed due to a well studied phenomenon known as preferential accumulation [281–284].

Despite the very simple shape of these PDFs using traditional methods such as MEM to reconstruct them is absolutely hopeless due to the fact that they do not tend to zero near the limits of the phase space. Figure 7.7 shows the reconstruction results of the diameter distribution of a globally uniform diameter size distribution conditioned on the samples enstrophy for 32^3 and 128^3 runs. 11 moments are used to generate these figures. Although LPM can capture the general behavior of the conditional particle size distribution, it does not provide a good fit specially near both ends of the distribution even with a large set of integer moments $\mu_1 \cdots \mu_{11}$. Another issue is that even if a very larger number of moments were available, the computation is restricted by the truncation errors since usually higher order moments are extremely small (large) which are multiplied by large (small) coefficients in the series. This results in error that can be as large as all the significant digits in the calculation. Double precision arithmetic

was sufficient for all the calculations in this study however if larger number of moments are required one can consider using arbitrary precision libraries such as the GMP [285].

Figures 7.8 and 7.9 summarize the fractional moment calculations using both methods for 32^3 simulation. In this section exact values are directly calculated from the DNS data and relative errors are calculated using these values. Similarly $\lambda = \frac{1}{p+1}$ is used for $p \in \{1.2, 2.3, 3.4\}$ and $\frac{2}{1+p}$ used for $p = 4.5$. LPM produces errors that can be as large as 45% when only first three moments (μ_0, μ_1, μ_2) are used, which is not surprising since the PDF fit is not accurate. DFMM produces accurate results even when the first three moments are used with errors that never exceeds 20%. Adding 2 more moments and DFMM produces extremely accurate results with errors all under 1% but LPM still produces errors as large as 30%. Figure 7.10 shows the same calculation using the 128^3 simulation. Only the results for $4\langle e \rangle < E_s < 5\langle e \rangle$ are presented which shows the same trend as the 32^3 simulation with the errors in DFMM results never exceeding 10% in this case, even for estimation with 3 moments.

To test the ability of the proposed methods in calculation of non-integer negative moments following non-integer negative moments are used $p \in \{-0.5, -1.5, -2.5\}$. The same functional form for the λ is used and the results are summarized in Table 7.6.

No. Integer Moments	Moment	Value	LPM	DFMM	LPM %Err	DFMM %Err
3	$\mu_{-0.5}$	4.3310	4.0025	4.6390	7.58	7.11
	$\mu_{-1.5}$	118.4843	90.0237	105.9396	24.20	10.58
	$\mu_{-2.5}$	5260.7300	2643.1820	3139.4362	49.76	40.32
5	$\mu_{-0.5}$	4.3310	4.1165	4.7404	4.95	9.45
	$\mu_{-1.5}$	118.4843	107.2110	119.72330	9.51	1.05
	$\mu_{-2.5}$	5260.7300	4108.2148	4677.2128	21.90	11.09

Table 7.6: Negative Fractional Moments estimated using 3 and 5 integer moments - Fractional moments of the DNS simulation estimated using first 3 and 5 integer moments including μ_0.

It is also possible to write λ as a function of both the number of retained integer moments and fractional order to guarantee both convergence properties and accuracy of the series as discussed in Section 7.4.3. DFMM generally produces better estimates with results that are in particular accurate for $p > -2$. However one of the advantages of

the DFMM is the possibility of adjusting the value of the free parameter λ for a specific application to yield exceptionally accurate values. It is worth mentioning that having an adjustable parameter is usually not considered as an advantage since it is not possible in practice to compare the calculated results to a known solution. However since this series is derived analytically the parameter H (or λ) only depends on the truncation of the series. This is directly related to the number of integer moments retained in the calculation but not on a specific form of the PDF or application. Therefore one can tune the parameter on standard test cases for a specific range of available moments and safely use it for any problem. For example $\lambda = \frac{1}{p+1}$ suggested in this chapter can safely be used for any problem as long as $|p| \leq N + 1 - \mathbb{1}_{p<0}$ and for slightly large values up to $|p| \leq N + 2 - \mathbb{1}_{p<0}$, $\lambda = \frac{2}{p+1}$ is a better choice. This range also covers the problem considered in Section 6.7.

7.5 Conclusion

In this chapter the non-integer moments closure problem which is encountered in many multiphase flow systems is considered. Considered two methods were extensively tested, a direct method and a method based on reconstruction of the underlying PDF. Different PDF reconstruction methods were considered and LPM is suggested due to its non-oscillatory behavior and better convergence properties compared to other orthogonal polynomial fitting methods. A direct method base on the results of fractional calculus and the fact that the MGF can also generate the non-integer moments is also formulated. Several problems are considered, including analytical, experimental and numerical simulations. All the test cases involve highly non-Gaussian distributions and DFMM method produced better results in almost all the test cases.

DFMM method involves a free parameter that should be adjusted for different problems. This parameter is directly related to the assumptions made in the mathematical derivation of DFMM equation and has simple mathematical explanation similar to the step size used in the finite difference approximations. A simple general expression is provided for the free parameter λ which produces accurate results in the provided test cases, however it can be adjusted for different problems to yield higher accuracies.

Generally both methods can be considered general with DFMM possessing slightly better properties, which in addition to its higher accuracy observed in the test cases,

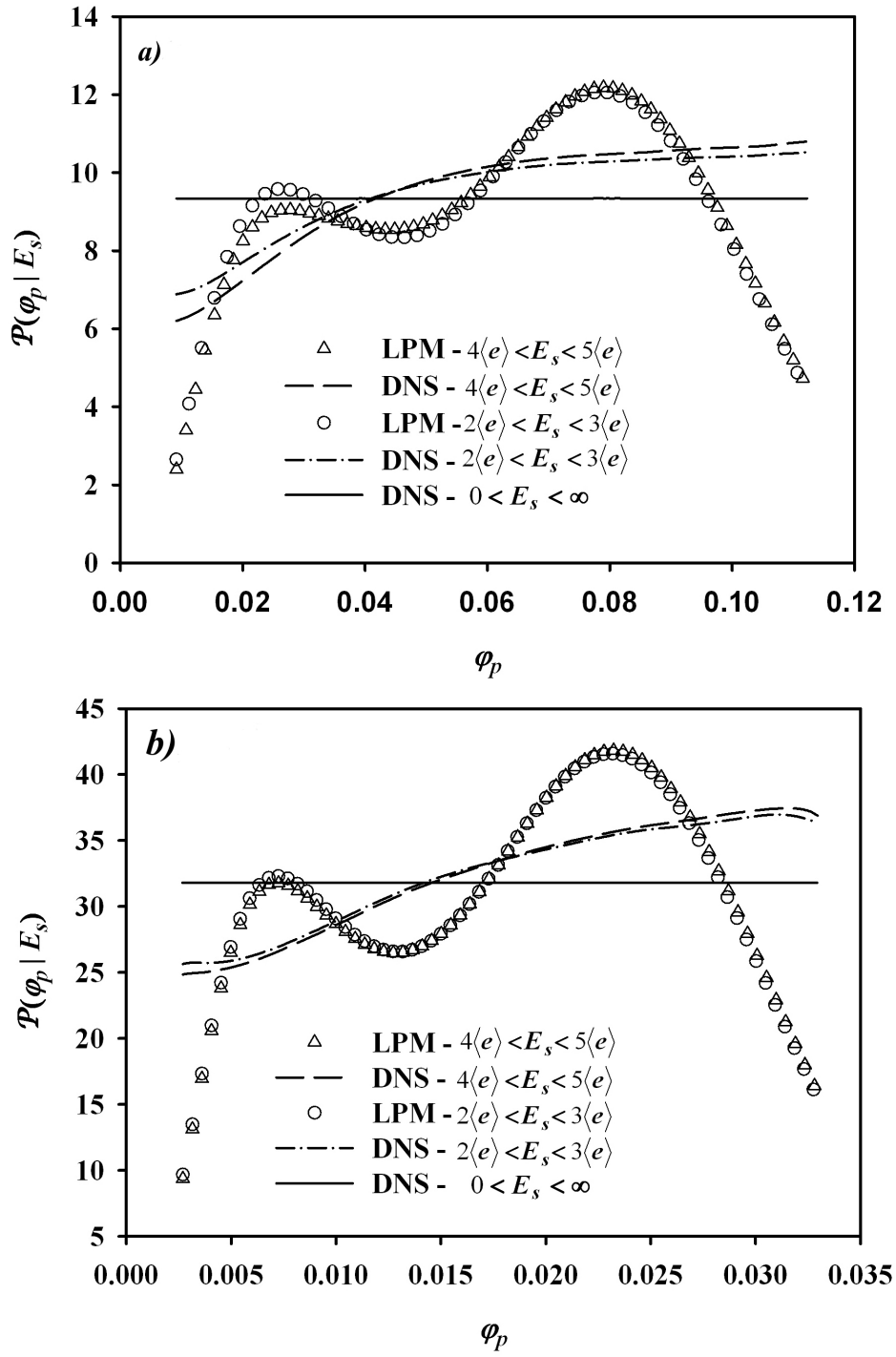


Figure 7.7: Conditional PDF reconstruction - Conditional PDF reconstructed using 11 moments, a. 32^3, b. 128^3 simulation.

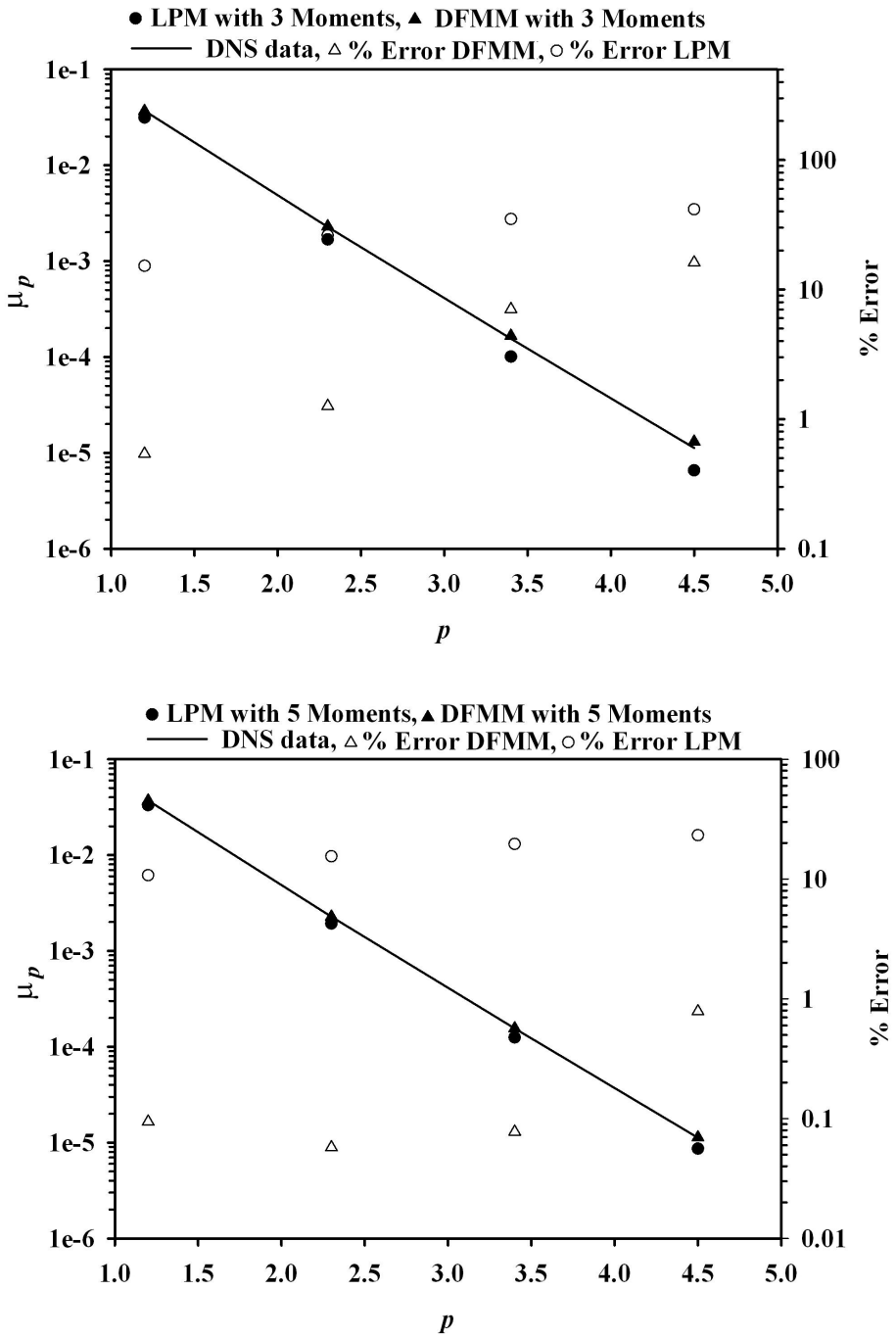


Figure 7.8: Calculation of fractional moments for 32^3 simulation with $2\langle e\rangle < E_s < 3\langle e\rangle$ - fractional moments ($\mu_{1.2}, \mu_{2.3}, \mu_{3.4}, \mu_{4.5}$) are calculated using first 3 and 5 moments using LPM and DFMM. In all cases better accuracy for DFMM is evident.

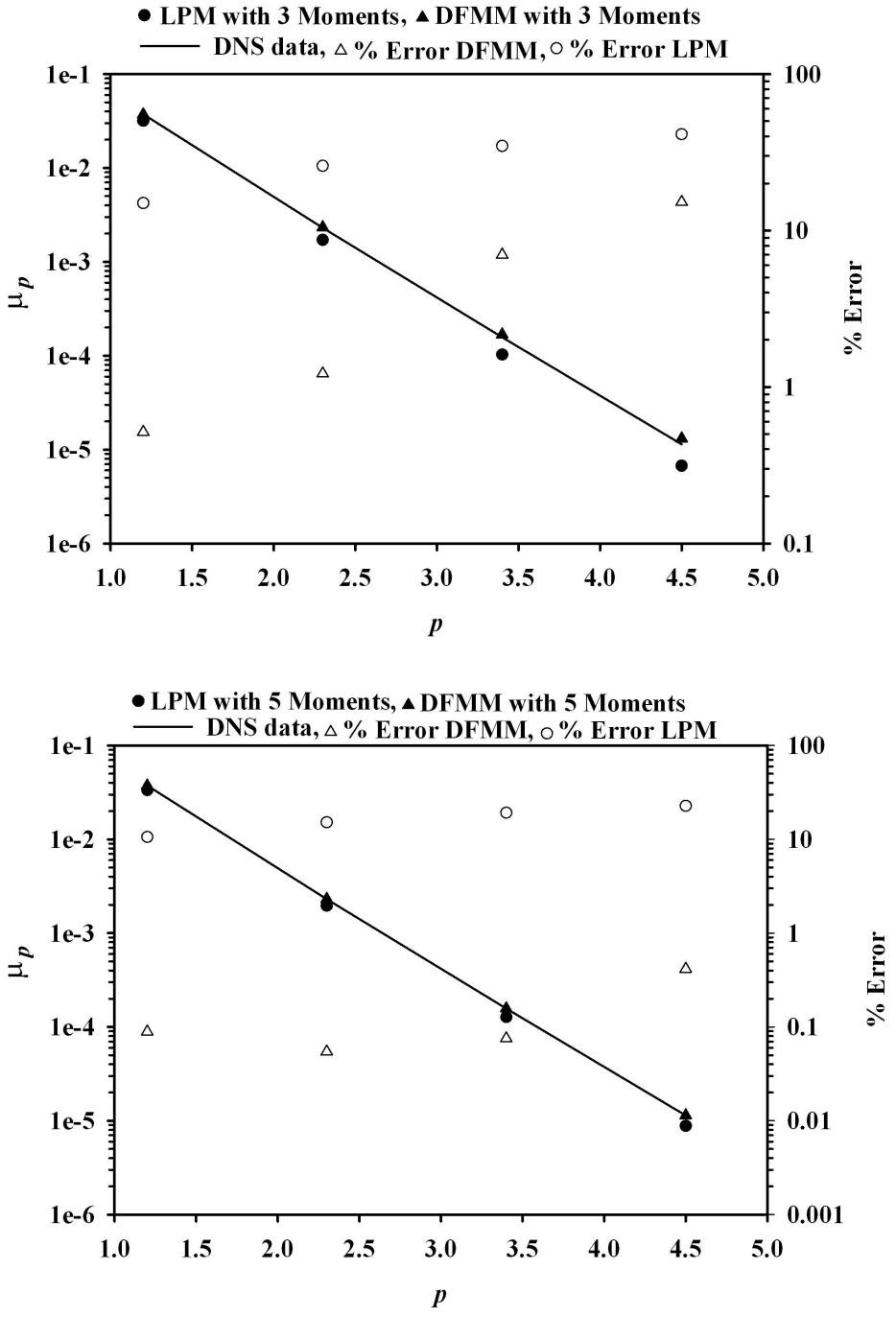


Figure 7.9: Calculation of fractional moments for 32^3 simulation with $4\langle e\rangle < E_s < 5\langle e\rangle$ - fractional moments ($\mu_{1.2}, \mu_{2.3}, \mu_{3.4}, \mu_{4.5}$) calculated using first 3 and 5 moments using LPM and DFMM for. In all cases better accuracy for DFMM is evident.

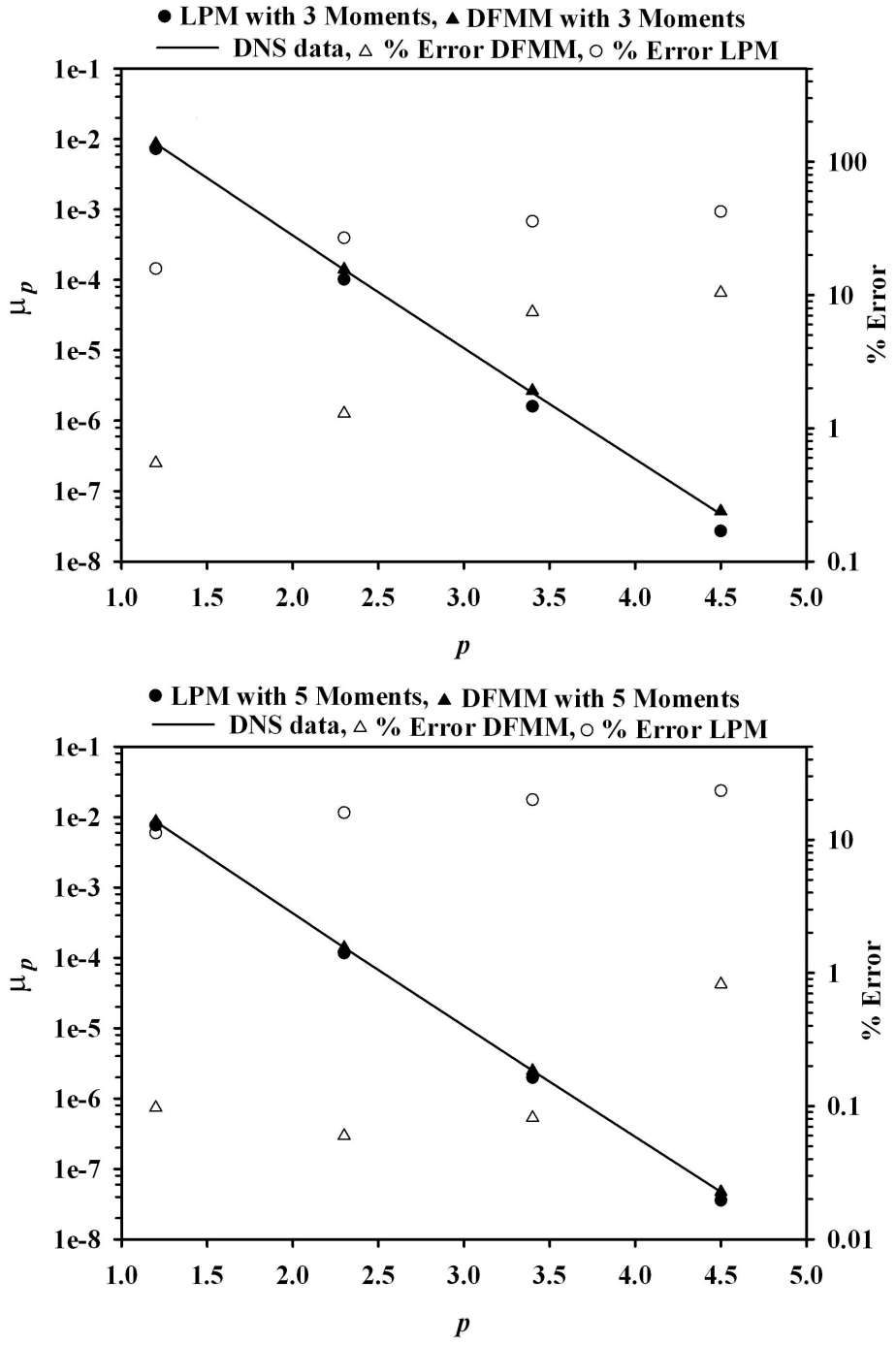


Figure 7.10: Calculation of fractional moments for 128^3 - Fractional moments $(\mu_{1.2}, \mu_{2.3}, \mu_{3.4}, \mu_{4.5})$ calculated using first 3 and 5 moments using LPM and DFMM. Maximum error never exceeds 10% for DFMM.

can be summarized as follow:

- No intermediate PDF is involved. This is very important property since one should always consider the possibility of producing non-realizable PDFs (negative probabilities) when reconstructing a PDF. There is no guarantee that LPM always produces realizable PDFs specially near the tails however it is easy to check for this issue and a simple spline fit to the small negative region can correct the behavior.
- DFMM is computationally much more efficient than any indirect method involving an intermediate PDF. If an indirect method is used in addition to reconstructing the intermediate PDF one also needs to perform numerical integration to calculate the value of the non-integer moment whereas in DFMM method a simple explicit equation can be derived which can be implemented in less than 100 lines of code.
- Coordinate shifts are needed if the random variable assume negative values in LPM whereas no such manipulations are needed for DFMM.

However it should be noted that since the LPM method is merely a series expansion, extension to higher dimensions is straightforward but in the case of the DFMM method a rigorous extension of the theory is required.

8

Future work plan

8.1 Code development and parallelization

On the Eulerian side the 3D code is fully parallelized with all the requirements in place and several libraries required are extensively tested. However to be able to perform large scale fully resolved simulations in 3D, parallelization of the particle phase is also required. This proves to be more difficult for a general multi block code because of the number of connections between the blocks. If we let the blocks to be connected arbitrarily the number of block dependencies for a single block increases from 6 for a purely Eulerian simulation to 26 including the overlaps from the edges and corners. For such a general case in addition to complicated implementation the scalability and performance of the final scheme is questionable. Some restriction might be needed on the domain decomposition topologies such as simple pencil or slice decomposition. Although there are some studies regarding the performance and scalability of parallel CFD codes for simulation of Eulerian fields there is a gap in the literature on the parallel Lagrangian particle simulations, therefore such a study can be a valuable contribution to the literature.

Several iterative linear solver are implemented, tested and parallelized in this study, except for the parallelization of the multigrid accelerator which is not attempted. Although a parallelization of the multigrid accelerator is straight–forward and easy to implement having all the communication and data structures in place, the performance deteriorated rapidly by increasing the number of grid levels due to a large number of communication in coarse grid levels and further research is required for a scalable im-

plementation. One such improvement can be to solve one or two coarsest levels on all the processors independently.

8.2 Direct Numerical simulations

Several fixed grid methods are compared and the FD method is found to be superior to IB methods especially if other transfer phenomena are considered. The method is extended to heat transfer problems and is tested extensively in 2D. Possible extensions to the algorithm can be

- Addition of other phenomena such as mass transfer by solving other scalar equations and using a similar assumption of a fluid with variable physical property.
- Implementation of molecular level phenomena such as thermophoresis is straight forward by performing a molecular dynamics simulation on the cells containing the boundaries of the immersed object and integrating the extra forces on the boundary of the object.
- Particle collision strategy can be refined to remove the ad hoc collision parameter from the model. It is possible to exactly impose the collision condition between the particles by imposing the rigidity criteria on the system of two particles that are about to collide and then using the conservation of momentum to calculate the velocities in the normal direction. However this is only possible for collision involving two particles and for collision of larger number of particles is not feasible. More rigorous models are also available that consider both tangential and normal forces which of course are computational much more expensive. This is an active research area which can easily be the subject of an independent study.

Another objective of the thesis is also to establish a hierarchical framework for fluid–particle systems where the data from more accurate simulations are used to provide models for more engineering inclined simulations. This has been demonstrated by providing a very high quality correlation for the local heat transfer coefficients around a circular cylinder at low to moderate Reynolds numbers and a wide range of Prandtl numbers. Fully resolved simulations are particularly useful for this type of studies. For

example simulations can be performed to study the particle history force and to provide more accurate correlations for the drag and lift forces or heat transfer coefficients around a single or a bed of particles.

8.3 EE models

Modelling process for a general polydispersed fluid-particle system is reviewed in this study. Different terms requiring closures are identified and a problem with non–integer moments is studied extensively. Two different classes of methods for the calculation of the non–integer moments based on the available integer moments are implemented and compared. The first method is based on the reconstruction of the underlying PDF using orthogonal polynomials and the other one is based on the fractional derivatives of the moment generating function. It is found that the method based on the moment generating functions generally produces better results.

The main improvement here would be to extend the methods to higher dimensional cases. This would be easier for the reconstruction method since it is merely a series expansion and the extension to higher dimensional cases is straight forward. However the method is based on the Laguerre polynomials and multi–dimensional Laguerre polynomials are usually built upon independent one dimensional kernels. Therefore this extension requires very careful examination for its capabilities of capturing the cross correlations. The method based on fractional derivatives can be improved by examining other functional forms for λ. In this study it is assumed that λ is a function of order of the non–integer moment. However this works very well with very small number of moments and especially when the required moments fall between the available moments. A better proposal would be to choose λ to be a function of both the number of integer moments and also the order of non–integer moments. It is also possible to consider the extension of the method to higher dimensional cases but a careful study of the theory of partial fractional derivatives might be requires.

Following our hierarchical modelling strategy, again data from spectral point–particle simulations are used in this section to assess the validity of the proposed estimation methods for real particulate systems. The available database can also be used to validate the proposed closures for multi–dimensional cases in the future.

Appendix A

Linear Solvers

A short description of the implemented linear solvers will be provided in this Appendix in addition to a short discussion on multigrid methods and the implemented algebraic multigrid strategy.

A.1 Basic iterative solvers

The simple iterative methods are characterized by the basic operation

$$\mathbf{x}^k = \mathbf{B^{-1}}\mathbf{x}^{k-1} + \mathbf{c}, \tag{A.1}$$

where $\mathbf{B}$ is an estimate to the full factorization of the coefficient matrix and $\mathbf{c}$ is the right hand side of the equation multiplied by B^{-1}. These are very simple methods to implement but also the convergence rate is extremely slow. The main reason for implementing them is that they can effectively be used as preconditioners for the Krylov subspace methods and also smoothers for the multigrid methods.

A.1.1 The Jacobi method

The Jacobi method is easily derived by examining each of the equations in the linear system $\mathbf{Ax} = \mathbf{b}$ in isolation. The method can be written in matrix form

$$\mathbf{x}^k = \mathbf{D}^{-1}\mathbf{b} - \mathbf{D}^{-1}\left(\mathbf{L} + \mathbf{U}\right)\mathbf{x}^{k-1}. \tag{A.2}$$

In this equation $\mathbf{L}$ and $\mathbf{U}$ are strictly lower and upper triangular parts of the coefficient matrix $\mathbf{A}$ and $\mathbf{D}$ is its diagonal. Note that no extra calculation is needed for this

factorization. Also note that the order in which the equations are examined is irrelevant, since the Jacobi method treats them independently.

A.1.2 The Successive Over Relaxation (SOR) method

If the equations in A.2 are solved in sequence so that the updated values of $\mathbf{x}$ are used as soon as they are available, then we obtain the Gauss–Seidel algorithm which can be written in matrix form

$$\mathbf{x}^k = (\mathbf{D} + \mathbf{L})^{-1}\,\mathbf{b} - (\mathbf{D} + \mathbf{L})^{-1}\left(\mathbf{U}\mathbf{x}^{k-1}\right). \tag{A.3}$$

The SOR method is derived by applying a weighted average between the previous iterate and the computed Gauss-Seidel iterate successively for each component which can be written in matrix form

$$\mathbf{x}^k = \omega\,(\mathbf{D} + \omega\mathbf{L})^{-1}\,\mathbf{b} - (\mathbf{D} + \omega\mathbf{L})^{-1}\,(\omega\mathbf{U} - (1-\omega)\,\mathbf{D})\,\mathbf{x}^{k-1}. \tag{A.4}$$

The choice of the value of ω is crucial to the convergence of the SOR method. A well known theorem due to Kahan [286] shows that SOR fails to converge if ω is outside the interval $(0, 2)$. If the relaxation term $\omega = 1$, SOR reduces to the Gauss-Seidel method. There are variations of the method that change the order of the solution which is useful for efficient parallel implementation. This can degrade or enhance the rate of convergence and usually there is a trade off between the rate of convergence and the parallelism [55]. We will use these methods only as preconditioners for a block decomposition of the domain where the preconditioner is usually applied locally [7], therefore we will not discuss these methods further.

A.2 Incomplete factorization methods

These methods stand between the complete factorization methods used in direct solvers where the coefficient matrix is completely factorized into its lower $\mathbf{L}$ and upper $\mathbf{U}$ triangular matrices and the basic solvers (Jacobi, SOR) where $\mathbf{L}$ and $\mathbf{U}$ are directly extracted from $\mathbf{A}$ without any calculation. One can think of it as a trade off between the effort put into the factorization and the number of iterations needed for the convergence. Here instead of doing a complete factorization only a restricted number of off diagonal

elements of the **U** and **L** matrices are allowed to take on a non-zero value. Typically only the diagonals that are non-zero in the matrix **A** and one or two further bands. Among these methods IC, ILU, SIP and MSI methods are considered. The methods are efficient for small to medium size problems also can be used as sophisticated preconditioners (smoothers) for Krylov (multigrid) methods.

A.2.1 Incomplete Cholesky and incomplete LU factorization methods

The simplest incomplete factorization scheme is the Incomplete Cholesky (IC) method, where a symmetric matrix **A** is factored into a $\widehat{\mathbf{L}}\widehat{\mathbf{L}}^T$ system, with non-zero values only when $a_{i,j} \neq 0$. Assuming

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{L}^T, \tag{A.5}$$

an incomplete factorization of **A** can be written by

$$\mathbf{M} = \left(\mathbf{\Delta} + \widehat{\mathbf{L}}\right)\left(\mathbf{\Delta} + \widehat{\mathbf{L}}^T\right), \tag{A.6}$$

where $\mathbf{\Delta}$ is a diagonal matrix. Now assuming $m_{i,j} \neq 0$ only where $a_{i,j} \neq 0$ it can be shown that the $\widehat{\mathbf{L}} = \mathbf{L}$ and

$$\delta_{i,i} = \frac{1}{\sqrt{a_{i,i} - \Sigma_{j=1}^{i-1}(a_{i,j}\delta_{j,j})^2}}. \tag{A.7}$$

The system can then be solved using the following procedure

$$(\mathbf{\Delta} + \mathbf{L})\mathbf{y} = \mathbf{r}^{k-1} \tag{A.8}$$

$$(\mathbf{\Delta} + \mathbf{L}^T)\mathbf{z} = \mathbf{y} \tag{A.9}$$

$$\mathbf{x}^k = \mathbf{x}^{k-1} + \mathbf{z}. \tag{A.10}$$

The method is not often used as a solver. However it has frequently been used as a preconditioner for the Conjugate Gradient method, the combination being referred to as the Incomplete Cholesky-Conjugate Gradient method or ICCG. The solver can be modified to eliminate the square roots in the factorization of the equations resulting in the Incomplete LDL method. It can also be modified for the application to non-symmetric systems resulting in the incomplete LU (ILU) method [287].

A.2.2 Strongly Implicit Procedure (SIP)

The Strongly Implicit Procedure (SIP) of Stone [7, 288] is a more advanced version of the Incomplete LU scheme, but unlike the IC and ILU methods it is only applicable to the equations resulting from a finite volume discretization of a PDE. The approximate LU decomposition of $\mathbf{A}$ that is used is

$$\mathbf{M} = \mathbf{LU} = \mathbf{A} + \mathbf{N}, \tag{A.11}$$

where $\mathbf{N}$ is the error between the exact and approximate factorizations. If $\mathbf{L}$ and $\mathbf{U}$ are constrained to be only non-zero in the locations that $\mathbf{A}$ is non-zero then the matrix $\mathbf{M}$ will have six extra non-zero diagonals than $\mathbf{A}$ if it is a seven-diagonal matrix from a three dimensional PDE. To make $\mathbf{M}$ a good approximation of $\mathbf{A}$, the $\mathbf{N}$ array is set such that $\mathbf{Nx} \approx \mathbf{0}$. This is done by recognizing that the system being solved is from a finite volume approximation of a PDE. Thus the values of the $\mathbf{x}$ field in the extra diagonals of $\mathbf{N}$ can be approximated by a second order extrapolation of the values of $\mathbf{x}$. By putting the terms for the extrapolation into the elements of $\mathbf{N}$ and canceling with the values of $\mathbf{x}$ in the extra diagonals of $\mathbf{N}$ then the system can be made to approximate $\mathbf{Nx} \approx \mathbf{0}$. Finally, to make the LU factorization unique the diagonal elements of $\mathbf{U}$ are set to 1, see [7] for a detailed derivation.

A.2.3 Modified Strongly Implicit procedure (MSI)

The Strongly Implicit Procedure was further developed by Schneider and Zedan [289, 290] into the Modified Strongly Implicit procedure (MSI). In this method the $\mathbf{L}$ and $\mathbf{U}$ arrays chosen in the decomposition in Equation A.11 are allowed to have more non-zero elements. The code for the MSI method is more complicated than for the SIP scheme and it requires more memory, using $14n$ words of storage for three dimensional PDEs. However it converges much faster than the other incomplete factorization methods. It is infrequently reported in the literature and seems to be less frequently used, presumably due the increased complexity of the implementation and the extra storage used.

A.3 Krylov subspace methods

A.3.1 GMRES method

The idea of GMRES method to solve a system of equations $\mathbf{Ax} = \mathbf{b}$ is to approximate the true solution by a vector $\mathbf{x}^n \in \mathscr{K}_n$ that minimizes the norm of the residual $\|\mathbf{r}\|_n$. Here $\mathscr{K}_n$ is the Krylov subspace generated by $\mathbf{A}$ and $\mathbf{b}$ which is defined as follows

$$\mathscr{K}_n = \langle \mathbf{b}, \mathbf{Ab}, \cdots, \mathbf{A}^{n-1}\mathbf{b} \rangle. \tag{A.12}$$

This is the spaces generated by successively larger group of vectors $\mathbf{b}, \mathbf{Ab}, \cdots$. In other words the problem is to find a vector $\mathbf{c} \in \mathbb{R}^n$ such that $\|\mathbf{AK}_n\mathbf{c} - \mathbf{b}\| = minimum$ where $\mathbf{K}_n$ is the Krylov matrix constructed by using vectors belonging to the Krylov subspaces. This minimization is not stable unless the a special procedure named Arnoldi iteration [53] is used to construct a sequence of Krylov matrices $\mathbf{Q}_n$ whose columns span the successive Krylov subspaces $\mathscr{K}_n$. Therefore writing $\mathbf{x}_n = \mathbf{Q}_n\mathbf{y}$ results in the following least squares problem

$$\begin{aligned} \|\mathbf{AQ}_n\mathbf{y} - \mathbf{b}\| &= \|\mathbf{Q}_{n+1}\mathbf{H}_n\mathbf{y} - \mathbf{b}\| \\ &= \|\mathbf{H}_n\mathbf{y} - \mathbf{Q}_{n+1}^T\mathbf{b}\| \\ &= \|\mathbf{H}_n\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1\|. \end{aligned} \tag{A.13}$$

First line is based on a Hessenberg[1] reduction of form $\mathbf{AQ}_n = \mathbf{Q}_{n+1}\mathbf{H}_n$ where $\mathbf{H}_n$ is a Hessenberg matrix and Arnoldi iteration should be used for the reduction. The fact that the norm does not change by multiplying by an orthogonal matrix is used for going from the first to second line and the last line results from the method of construction of the Krylov matrices by Arnoldi iteration. Therefor the n^{th} step of the GMRES algorithm consists of step n of the Arnoldi iteration and minimization of Equation A.13 and finally calculation of $\mathbf{x}_n = \mathbf{Q}_n\mathbf{y}$. To place an upper limit on the storage required by the scheme, the solver is commonly implemented with a restart after $k_{restart}$ iterations, see [291] for the complete algorithm.

[1] A Hessenberg matrix has zeros below the first sub-diagonal.

A.3.2 The Conjugate Gradient (CG) method

CG method is only applicable to symmetric positive definite matrices. The method is a system of recurrence formula that generates a sequence of iterate $\mathbf{x}_n \in \mathscr{K}_n$ with property that at step n, $\|\mathbf{t}_n\|_A$ is minimized. Here $\|\mathbf{t}_n\|_A = \sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}$ and has the properties of a norm as defined by Equation 2.23. A complete algorithm can be found in [7].

Since it requires a symmetric system, any preconditioner used with the solver must preserve this symmetry. Thus Jacobi method can be used as preconditioners, but not SOR. ICCG (Incomplete Cholesky-Conjugate Gradient) method as explained in Section A.2.1 is common choice and the solver is specifically applicable to pressure correction equation in SIMPLE [29] type algorithms.

A.3.3 The BCG stabilized method

The Bi-Conjugate Gradient Stabilized method was developed by van der Vorst [292] from the CGS, BiCG and GMRES methods, in order to solve non symmetric systems of equations whilst avoiding the often highly irregular convergence patterns of CGS and BiCG, and the large storage requirements of GMRES. The price to be paid for the non symmetric problem is that one must work with two Krylov subspaces rather than one, generate by multiplication by $\mathbf{A}^T$ as well as $\mathbf{A}$.

The method can be applied to non-symmetric systems, and is quite robust and efficient. Like the conjugate gradient method it only stores a limited number of vectors at any iteration, and hence unlike the GMRES solver does not increase its per-iteration memory use and operation count with increasing number of iterations.

A.4 Multigrid method

Multigrid methods are mainly used to overcome the scalability issues of the simple iterative and incomplete factorization solvers. As explained in Section 2.7.4 the scalability issue is mainly due to the inability of the basic iterative solvers in eliminating the long wavelength components of the error field. There are several good texts on the subject which are mostly involved in theoretical aspects [293–296]. In this section the mechanisms of a typical multigrid solver is discussed and a simple algebraic strategy is explained to be applied to a sparse system of equation from a finite volume discretization.

There are generally two types of multigrid techniques namely algebraic and geometric. In geometric multigrid a set of successively coarser meshes is explicitly generated and equations are explicitly discretized on each level. In contrast algebraic multigrid methods only operate on the coefficient matrix. The algebraic multigrid scheme then transforms the coefficient matrix onto a series of progressively coarser matrices and finally precisely solves the system on the coarsest level. The solution is then solved on the series of successively finer meshes, using the solution from the previous (coarser) levels as an initial estimate, finally solving on the finest mesh. By transferring from a fine to a coarse system the medium wavelength errors in the fine mesh solution are transformed into short wavelength errors on the coarse mesh which are much easier to smooth out. The computation cost for a precise solution to the coarsest level is low, and the cost for solving on the finer meshes is reduced by using the coarse mesh solution as an initial estimate.

The three basic operators for the multigrid technique are the smoother, which improves the current estimate of the solution on a given mesh, and the restriction and prolongation operators, which map a set of equations and a solution between a fine and a coarse mesh.

For all but the coarsest mesh the multigrid solver is recursively applied to solve the restricted system of equations. On the coarsest mesh the restricted system is solved either by an iterative method, solving the equations to full convergence, or by the use of a direct method.

On the finest mesh the norm of the residual is taken and compared to a supplied tolerance. If the norm of the residual is less than the tolerance then the solver is taken to have converged and the process terminates. Otherwise the solver applied recursively to the system.

The smoother for the multigrid method is typically a simple iterative scheme such as SOR, Gauss Seidel or Jacobi iteration, or an incomplete factorization method such as SIP or MSI. At each recursive call of the multigrid solver the solution is smoothed by applying several iterations of the smoother to the solution. The smoother does not have to be applied to convergence since the aim is not to solve the equations but rather to smooth the solution on the current mesh.

A.4.1 Prolongation and Restriction operations

Geometric multigrid are not general since they are closely tied to the underlying solver and the discretization of the equations on each mesh level. The multigrid method discussed here conforms to the ideas of algebraic multigrid where the solver is not directly coupled to the discretization process and hence the prolongation and restriction operators must be derived from the fine mesh equations rather than re–discretizing the underlying equations.

To illustrate the process consider the finite volume discretization of a simple one-dimensional PDE using 3 grid points with an explicit implementation of boundary condition into the right hand vector. The resulting system of equation

$$a_{W_i} x_{i-1} + a_{P_i} x_i + a_{E_i} x_{i+1} = b_i, \tag{A.14}$$

would have the form

$$\begin{bmatrix} a_{P_1} & a_{E_1} & & & \\ a_{W_2} & a_{P_2} & a_{E_2} & & \\ & a_{W_3} & a_{P_3} & a_{E_3} & \\ & & a_{W_4} & a_{P_4} & a_{E_4} \\ & & & a_{W_5} & a_{P_5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}.$$

For an elliptic PDE the solution must be reasonably smooth, and so the solution values at the even numbered points in the mesh can be estimated from a second order centred interpolation from the odd numbered points,

$$x_2 = \frac{1}{2}(x_1 + x_3), \quad x_4 = \frac{1}{2}(x_3 + x_5). \tag{A.15}$$

Substituting Equation A.15 into Equation A.14 results in the system of equation

$$\begin{bmatrix} 2a_{P_1} + a_{E_1} & a_{E_1} & \\ a_{W_3} & 2a_{P_3} + a_{W_3} + a_{E_3} & a_{E_3} \\ & a_{W_5} & 2a_{P_5} + a_{W_5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} 2b_1 \\ 2b_3 \\ 2b_5 \end{bmatrix},$$

which can be rewritten as

$$\begin{bmatrix} a_{P_1}^{2h} & a_{E_1}^{2h} & \\ a_{W_3}^{2h} & a_{P_3}^{2h} & a_{E_3}^{2h} \\ & a_{W_5}^{2h} & a_{P_5}^{2h} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1^{2h} \\ b_3^{2h} \\ b_5^{2h} \end{bmatrix},$$

with

$$\begin{aligned} a_{P_i}^{2h} &= 2a_{P_{2i-1}} + a_{E_{2i-1}} + a_{W_{2i-1}}, \\ a_{E_i}^{2h} &= a_{E_{2i-1}}, a_{W_i}^{2h} = a_{W_{2i-1}} \\ b_i^{2h} &= 2b_{2i-1}. \end{aligned} \tag{A.16}$$

For these equations the restriction can be accomplished by a simple injection,

$$x_i^{2h} = x_{2i-1}. \tag{A.17}$$

The corresponding prolongation can be performed using,

$$x_i = x_{(i+1)/2}^{2h} \quad i = 1, 3, \cdots, \quad x_i = \frac{1}{2}(x_{\frac{i}{2}+1} + x_{\frac{i}{2}}) \quad i = 2, 4, \cdots \tag{A.18}$$

Same principles can be used to write similar equation in 2 or 3 dimensions where bi– or tri–linear interpolations are used for prolongation and restriction phases.

A.4.2 Multigrid cycles

The number of iterations on each level of multigrid mesh can be changed to yield different types of multigrid cycles. If the number of iterations on each level is set to one it means that the algorithm visits each level of mesh just once, see Figure A.1, which is called the "v-cycle". On the other hand if we set the number of inner iterations to 2, then each level of the mesh is visited two times creating a w-cycle which is also illustrated in Figure A.1. For a parallel run the most efficient cycle is a v-cycle [295].

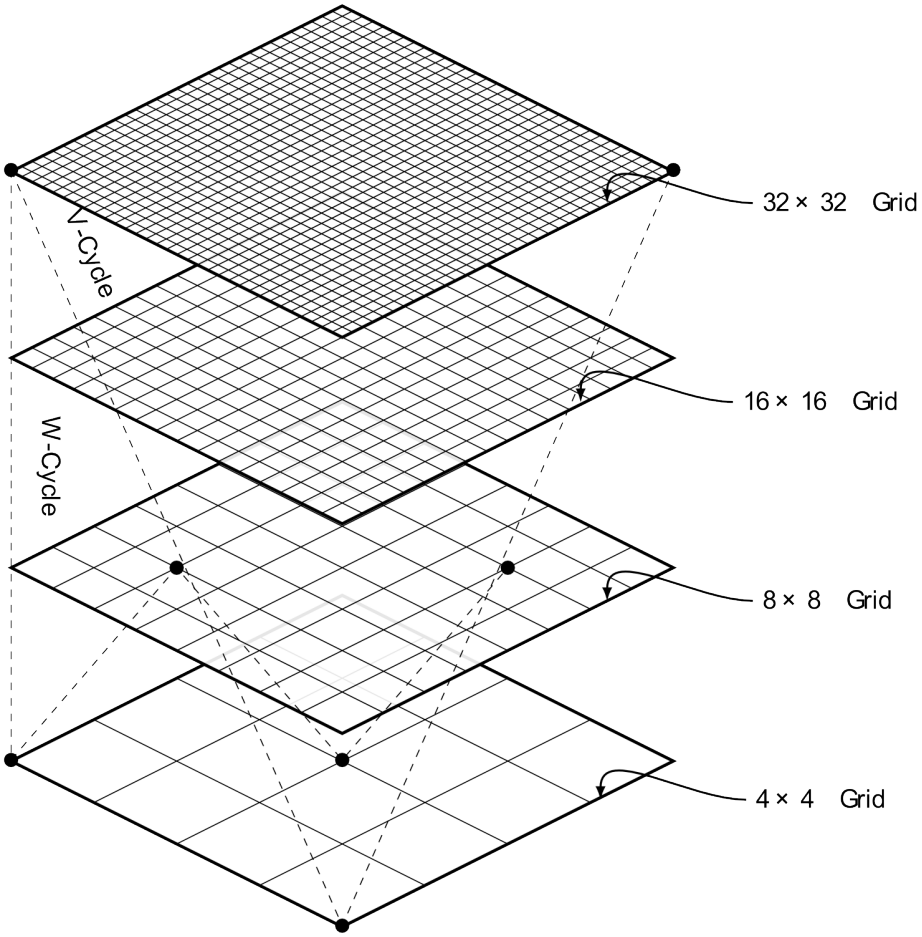


Figure A.1: Different multi-grid cycles - Four levels of mesh for a multigrid method. In a v-cycle each level of mesh is visited just once whereas in a w-cycle each level of the mesh is visited two times.

Appendix B

Coefficients for the polynomial fit

For the stability of the fitting process it is required to use the z–scores of the independent variables, i.e the normalized variable by mean and standard deviation, therefore following variables are defined

$$X_1 = \frac{Re - 130}{72.231512} \tag{B.1}$$

$$X_2 = \frac{Re - 130}{72.222222} \tag{B.2}$$

$$Y_1 = \frac{Pr_{0.1\leq \mathrm{Pr} \leq 5} - 1.041667}{1.287617} \tag{B.3}$$

$$Y_2 = \frac{Pr_{5.0< \mathrm{Pr} \leq 40} - 22.923080}{11.121256}. \tag{B.4}$$

Note that for each coefficient in the Trigonometric6 model two sets of equations are required one for $0.1 \leq \mathrm{Pr} \leq 5$ which should be used with X_1 and Y_1 and one for $5.0 < \mathrm{Pr} < 40$ which should be used with X_2 and Y_2. These equations have the following form where the coefficients $p_{00} \cdots p_{23}$ should be read from the Tables B.1–B.3.

$$\begin{aligned} c = {} & p_{00} + p_{10}X + p_{01}Y + p_{20}X^2 + p_{11}XY \\ & + p_{02}Y^2 + p_{30}X^3 + p_{21}X^2Y + p_{12}XY^2 + p_{03}Y^3 \\ & + p_{40}X^4 + p_{31}X^3Y + p_{22}X^2Y^2 + p_{13}XY^3 + p_{50}X^5 \\ & + p_{41}X^4Y + p_{32}X^3Y^2 + p_{23}X^2Y^3, \end{aligned} \tag{B.5}$$

where

$$c \in \{a_{01}, a_{11}, b_{11}, a_{21}, b_{21}, a_{31}, b_{31}, a_{41}, b_{41}, a_{51}, b_{51}, a_{61}, b_{61}\}, \tag{B.6}$$

or

$$c \in \{a_{02}, a_{12}, b_{12}, a_{22}, b_{22}, a_{32}, b_{32}, a_{42}, b_{42}, a_{52}, b_{52}, a_{62}, b_{62}\}, \tag{B.7}$$

and $(X, Y) \in \{(X_1, Y_1), (X_2, Y_2)\}$ depending on the value of the Prandtl number as described earlier.

	a_{01}	a_{02}	a_{11}	a_{12}	a_{21}	a_{22}	a_{31}	a_{32}	a_{41}
p_{00}	24.46	765.36	-31.33	-1347.09	5.26	-45.03	11.77	583.73	-2.49
p_{10}	14.09	1318.02	-21.66	-2372.03	-4.31	157.08	9.13	1014.47	2.72
p_{01}	-48.31	440.83	94.03	-790.06	-7.45	64.39	-41.07	335.53	4.67
p_{20}	21.79	-179.58	-39.94	323.15	-3.88	73.36	17.70	-142.93	2.08
p_{11}	-29.13	184.32	54.38	-331.52	-18.36	69.97	-21.96	139.19	10.38
p_{02}	-56.61	-75.94	98.24	135.81	-14.05	-0.94	-40.66	-58.53	7.70
p_{30}	-5.60	-376.46	10.17	679.42	0.15	-113.35	-4.42	-283.08	-0.08
p_{21}	48.89	-330.74	-88.96	596.59	-1.69	-29.48	38.84	-253.81	0.70
p_{12}	-12.13	-177.61	20.26	319.97	-10.03	-36.39	-7.18	-135.15	5.24
p_{03}	17.71	5.67	-31.09	-10.09	2.98	-14.38	13.09	5.57	-1.67
p_{40}	-6.76	119.18	12.15	-215.87	-0.28	-11.96	-5.21	93.96	0.17
p_{31}	7.17	-30.25	-12.85	54.56	1.81	-22.24	5.39	-22.14	-1.03
p_{22}	14.28	15.79	-25.73	-28.70	1.50	-13.08	11.15	13.20	-0.95
p_{13}	12.08	53.73	-21.46	-97.00	2.77	1.39	8.87	41.76	-1.50
p_{50}	2.99	38.16	-5.38	-68.50	0.03	23.82	2.34	27.20	-0.03
p_{41}	-13.32	93.39	24.06	-168.76	-1.62	4.80	-10.21	72.12	0.93
p_{32}	-3.05	61.16	5.55	-110.44	0.65	7.76	-2.49	46.86	-0.32
p_{23}	-0.60	25.56	1.09	-46.06	0.24	9.09	-0.57	19.13	-0.10

Table B.1: Polynomial Coefficients to calculate a01–a42

	a_{42}	a_{51}	a_{52}	a_{61}	a_{62}	b_{11}	b_{12}	b_{21}	b_{22}
p_{00}	22.82	-1.51	-86.95	0.12	-1.34	1.45	42.30	-20.66	-986.35
p_{10}	-90.22	-1.52	-141.53	-0.30	9.83	4.03	-101.66	-14.98	-1739.09
p_{01}	-36.72	5.56	-46.29	-0.62	3.76	7.26	-41.80	70.74	-577.36
p_{20}	-40.13	-2.71	22.19	-0.14	3.30	2.45	-48.84	-29.77	238.01
p_{11}	-39.26	2.24	-18.39	-0.99	3.68	12.62	-46.79	39.77	-242.12
p_{02}	0.75	5.24	8.37	-0.72	-0.18	7.48	0.14	70.87	99.56
p_{30}	62.85	0.62	36.19	0.00	-6.11	-0.04	75.17	7.50	494.67
p_{21}	16.79	-5.71	34.82	0.05	-2.01	1.24	19.87	-65.92	436.36
p_{12}	20.50	0.46	18.04	-0.39	-2.10	6.37	24.27	13.63	233.67
p_{03}	7.88	-1.76	-1.31	0.17	-0.70	-1.52	9.84	-22.62	-8.00
p_{40}	6.32	0.73	-13.75	-0.03	-0.39	0.11	7.68	8.94	-159.07
p_{31}	12.42	-0.71	2.52	0.11	-1.19	-1.07	14.89	-9.39	39.53
p_{22}	7.22	-1.61	-2.21	0.14	-0.66	-0.79	8.80	-19.00	-21.48
p_{13}	-0.89	-1.11	-5.88	0.13	0.15	-1.74	-0.86	-15.49	-71.26
p_{50}	-12.99	-0.33	-2.96	0.01	1.18	0.02	-15.70	-3.98	-49.13
p_{41}	-2.74	1.38	-10.02	-0.10	0.36	1.01	-3.30	17.62	-123.67
p_{32}	-4.37	0.40	-6.33	0.02	0.46	-0.48	-5.20	4.15	-80.78
p_{23}	-5.04	0.13	-2.40	-0.01	0.48	-0.22	-6.11	0.87	-33.47

Table B.2: Polynomial Coefficients to calculate a51–b22

	b_{31}	b_{32}	b_{41}	b_{42}	b_{51}	b_{52}	b_{61}	b_{62}
p_{00}	3.46	-42.64	5.31	266.92	-0.90	9.77	-0.33	-16.72
p_{10}	-4.71	143.08	4.46	452.05	1.23	-37.93	-0.38	-23.99
p_{01}	-7.92	58.26	-17.95	149.01	2.18	-15.20	0.78	-7.78
p_{20}	-3.38	66.72	8.14	-66.57	0.82	-16.15	-0.51	4.41
p_{11}	-17.06	63.70	-8.84	60.79	4.44	-16.10	0.11	-2.82
p_{02}	-12.25	-0.83	-17.73	-26.29	3.24	0.35	0.78	1.49
p_{30}	0.14	-102.16	-1.99	-122.50	-0.03	26.42	0.10	5.13
p_{21}	-1.24	-26.43	17.60	-112.56	0.18	7.33	-1.01	5.72
p_{12}	-8.51	-32.94	-2.69	-59.25	2.10	8.58	-0.03	2.84
p_{03}	2.63	-12.94	5.78	3.11	-0.72	3.24	-0.28	-0.37
p_{40}	-0.28	-10.65	-2.32	42.65	0.09	2.45	0.12	-2.50
p_{31}	1.66	-20.14	2.35	-9.21	-0.45	5.17	-0.10	0.24
p_{22}	1.46	-11.78	5.02	6.33	-0.45	2.95	-0.27	-0.48
p_{13}	2.41	1.31	3.84	18.68	-0.62	-0.41	-0.16	-1.02
p_{50}	0.04	21.29	1.05	11.20	-0.02	-5.40	-0.06	-0.27
p_{41}	-1.49	4.32	-4.50	32.13	0.41	-1.23	0.22	-1.69
p_{32}	0.55	7.04	-1.16	20.65	-0.12	-1.86	0.08	-1.02
p_{23}	0.18	8.19	-0.30	8.21	-0.03	-2.09	0.03	-0.33

Table B.3: Polynomial Coefficients to calculate b21–b62

Appendix C

Derivation of the transport equations

C.1 Transport equation for fluid phase Reynolds stresses

In this appendix explicit derivation of Eq. (6.54) from the general PDF transport equation is provided. Starting from Eq. (6.49), with $\mathscr{H}'_f = \left\langle u'_{f,i} u'_{f,j} \right\rangle$ it is possible to write

$$\frac{\mathrm{D}_f}{\mathrm{D}t}\left[\alpha_f \rho_f \left\langle u'_{f,i} u'_{f,j} \right\rangle\right] = \alpha_f \rho_f \frac{\mathrm{D}_f}{\mathrm{D}t}\left[\left\langle u'_{f,i} u'_{f,j} \right\rangle\right] + \left\langle u'_{f,i} u'_{f,j} \right\rangle \frac{\mathrm{D}_f}{\mathrm{D}t}[\alpha_f \rho_f]. \tag{C.1}$$

Using the definition of the material derivative, continuity equation for the fluid phase (Eq. (6.52)) can be written as

$$\frac{\mathrm{D}_f}{\mathrm{D}t}\left(\alpha_f \rho_f\right) = -\alpha_f \rho_f \frac{\partial u_{f,k}}{\partial x_k}. \tag{C.2}$$

Replacing $\frac{\mathrm{D}_f}{\mathrm{D}t}\left(\alpha_f \rho_f\right)$ in Eq. (C.1) using Eq. (C.2) gives

$$\frac{\mathrm{D}_f}{\mathrm{D}t}\left[\alpha_f \rho_f \left\langle u'_{f,i} u'_{f,j} \right\rangle\right] = \alpha_f \rho_f \frac{\mathrm{D}_f}{\mathrm{D}t}\left[\left\langle u'_{f,i} u'_{f,j} \right\rangle\right] - \alpha_f \rho_f \left\langle u'_{f,i} u'_{f,j} \right\rangle \frac{\partial u_{f,k}}{\partial x_k}. \tag{C.3}$$

The first term in the RHS of Eq. (C.3) is that appearing in the LHS of Eq. (6.54) and second term will be discussed later. Derivation of other terms is more involved and derivations are as follows:

C. DERIVATION OF THE TRANSPORT EQUATIONS

1. Drift term $\alpha_f \rho_f \left\langle A_{f,i} u'_{f,j} + A_{f,j} u'_{f,i} \right\rangle$: Firs note that the velocity components are independent variables, therefore $\frac{\partial u'_{f,n}}{\partial u'_{f,m}} = \delta_{nm}$, using this fact it is possible to write:

$$\alpha_f \rho_f \left\langle A_{f,m} \frac{\partial \mathscr{H}'_f}{\partial u'_{f,m}} \right\rangle = \alpha_f \rho_f \left\langle A_{f,m} \frac{\partial u'_{f,i} u'_{f,j}}{\partial u'_{f,m}} \right\rangle = \alpha_f \rho_f \left\langle A_{f,m} u'_{f,j} \delta_{im} + A_{f,m} u'_{f,i} \delta_{jm} \right\rangle = \alpha_f \rho_f \left\langle A_{f,i} u'_{f,j} + A_{f,j} u'_{f,i} \right\rangle .$$

2. Diffusion terms $\alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{ij} \right\rangle$:

$$\frac{1}{2} \alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{mn} \frac{\partial^2 \mathscr{H}'_f}{\partial u'_{f,m} \partial u'_{f,n}} \right\rangle = \frac{1}{2} \alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{mn} \frac{\partial^2 u'_{f,i} u'_{f,j}}{\partial u'_{f,m} \partial u'_{f,n}} \right\rangle = \frac{1}{2} \alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{mn} \frac{\partial}{\partial u'_{f,m}} (u'_{f,i} \delta_{jn} + u'_{f,j} \delta_{in}) \right\rangle = \frac{1}{2} \alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{mn} (\delta_{im} \delta_{jn} + \delta_{jm} \delta_{in}) \right\rangle = \frac{1}{2} \alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{ij} + \left(B_f B_f^T \right)_{ji} \right\rangle = \alpha_f \rho_f \left\langle \left(B_f B_f^T \right)_{ij} \right\rangle .$$

   Where the symmetry property of $\left(B_f B_f^T \right)$ is used to move to the last line.

3. Convection terms $-\alpha_f \rho_f \left\langle u'_{f,i} u'_{f,k} \right\rangle \frac{\partial \langle u_{f,j} \rangle}{\partial x_k} - \alpha_f \rho_f \left\langle u'_{f,j} u'_{f,k} \right\rangle \frac{\partial \langle u_{f,i} \rangle}{\partial x_k}$:

$$- \alpha_f \rho_f \frac{\partial \langle u_{f,n} \rangle}{\partial x_m} \left\langle \frac{\partial u'_{f,m} \mathscr{H}'_f}{\partial u'_{f,n}} \right\rangle = -\alpha_f \rho_f \frac{\partial \langle u_{f,n} \rangle}{\partial x_m} \left\langle \frac{\partial u'_{f,m} u'_{f,i} u'_{f,j}}{\partial u'_{f,n}} \right\rangle = - \alpha_f \rho_f \frac{\partial \langle u_{f,n} \rangle}{\partial x_m} \left\langle u'_{f,m} u'_{f,i} \delta_{jn} + u'_{f,m} u'_{f,j} \delta_{in} + u'_{f,i} u'_{f,j} \delta_{mn} \right\rangle = - \alpha_f \rho_f \left\langle u'_{f,i} u'_{f,k} \right\rangle \frac{\partial \langle u_{f,j} \rangle}{\partial x_k} - \alpha_f \rho_f \left\langle u'_{f,j} u'_{f,k} \right\rangle \frac{\partial \langle u_{f,i} \rangle}{\partial x_k}.$$

   Changing the dummy index m to k, the term $-\alpha_f \rho_f \left\langle u'_{f,i} u'_{f,j} \right\rangle \frac{\partial \langle u_{f,k} \rangle}{\partial x_k}$ cancels the corresponding term in the LHS (Eq. (C.3)).

4. Setting $\mathscr{H}'_f = u'_{f,i} u'_{f,j}$ in $\alpha_f \rho_f \frac{\mathrm{D}_f \langle u_{f,m} \rangle}{\mathrm{D}t} \left\langle \frac{\partial \mathscr{H}'_f}{\partial u'_{f,m}} \right\rangle$ will produce first order moments of fluctuating velocity which is zero by definition.

C.2 Transport equation for the particle phase momentum

In this appendix explicit derivation of Eq. (6.56) from the general PDF transport equation is provided. Starting from Eq. (6.50), with $\mathscr{H}_p = u_{p,i}$ the first term in the LHS of Eq. (6.50) can be written as

$$\frac{\partial}{\partial t}[\alpha_p \rho_p \langle u_{p,i} \rangle] = \alpha_p \rho_p \frac{\partial}{\partial t}[\langle u_{p,i} \rangle] + \langle u_{p,i} \rangle \frac{\partial}{\partial t}[\alpha_p \rho_p]. \tag{C.4}$$

Noting $u_{p,i} = \langle u_{p,i} \rangle + u'_{p,i}$ and that by definition the first order moments of the fluctuating component are zero, second term in Eq.(6.50) can be written by

$$\begin{aligned} \frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \langle u_{p,i} u_{p,j} \rangle\right] &= \frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \langle u'_{p,i} u'_{p,j} \rangle\right] + \frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \langle u_{p,i} \rangle \langle u_{p,j} \rangle\right] \\ &= \frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \langle u'_{p,i} u'_{p,j} \rangle\right] + \langle u_{p,i} \rangle \frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \langle u_{p,j} \rangle\right] + \alpha_p \rho_p \langle u_{p,j} \rangle \frac{\partial}{\partial x_j}\left[\langle u_{p,i} \rangle\right]. \end{aligned} \tag{C.5}$$

By multiplying the particle continuity equation (Eq.(6.55)) by $\langle u_{p,i} \rangle$ it is possible to write

$$\langle u_{p,i} \rangle \frac{\partial}{\partial x_j}\left[\alpha_p \rho_p \langle u_{p,j} \rangle\right] = -\langle u_{p,i} \rangle \frac{\partial}{\partial t}[\alpha_p \rho_p], \tag{C.6}$$

which cancels the second term in Eq. (C.4). Additional terms by definition reduce to the first term in the LHS and first term in the RHS of Eq.(6.56). By noting $u_{p,i}$ are independent variables first term in the RHS of Eq. (6.50) can be simplified:

$$\alpha_p \rho_p \left\langle A_{p,j} \frac{\partial u_{p,i}}{\partial u_{p,j}} \right\rangle = \alpha_p \rho_p \langle A_{p,j} \delta_{ij} \rangle = \alpha_p \rho_p \langle A_{p,i} \rangle . \tag{C.7}$$

Other terms in the LHS of Eq. (6.50) are zero since the variables are independent and this concludes the derivation of the Eq. (6.56).

Appendix D

Fractional Calculus

D.1 Proof of n-fold integration formula

In this section a proof for Eq.(7.18) which is used to extend the n-fold integration to arbitrary orders, is provided. By setting $h(\tau_1) = \int_a^{\tau_1} f(\tau)d\tau$ and $g'(\tau_1) = 1$ we have

$$\begin{aligned}\int_a^t d\tau_1 \int_a^{\tau_1} f(\tau)d\tau &= \int_a^t h(\tau_1)g'(\tau_1)d\tau_1 \\ &= h(\tau_1)g(\tau_1)|_a^t - \int_a^t \tau_1 f(\tau_1)d\tau_1 \\ &= t\int_a^t f(\tau)d\tau - \int_a^t \tau f(\tau)d\tau \\ &= \int_a^t (t-\tau)f(\tau)d\tau. \end{aligned} \tag{D.1}$$

It is now easy to use Eq.(D.1) n times to show by induction that Eq.(7.19) holds for any integer n.

D.2 Generation of arbitrary order moments from the MGF

First note that using Eq. (7.21) with $\nu = k - p$ we have:

$${}_{-\infty}D_t^p f(t) = \frac{1}{\Gamma(\nu)}\left(\frac{d}{dt}\right)^k \int_{-\infty}^t (t-\tau)^{\nu-1} f(\tau)d\tau. \tag{D.2}$$

Using $\eta = t - \tau$:

$$
\begin{aligned}
{}_{-\infty}D_t^p f(t) &= \frac{1}{\Gamma(\nu)}\left(\frac{d}{dt}\right)^k \int_0^\infty \eta^{\nu-1} f(t-\eta)d\eta \\
&= \frac{1}{\Gamma(\nu)} \int_0^\infty \eta^{\nu-1} \left(\frac{d}{dt}\right)^k f(t-\eta)d\eta \\
&= \frac{1}{\Gamma(\nu)} \int_{-\infty}^t (t-\tau)^{\nu-1} \left(\frac{d}{dt}\right)^k f(\tau)d\tau. \qquad \text{(D.3)}
\end{aligned}
$$

The last line of Eq. (D.3) is the Caputo [297] definition of the fractional derivative with $a \to -\infty$, i.e ${}_{-\infty}^{C}D_t^p f(t)$. The RL and Caputa definitions are not generally equivalent however Eq. (D.3) shows that both definitions are equivalent for the limit $a \to -\infty$. Now we can use [298]

$$
{}_{-\infty}^{C}D_t^p e^{ct} = c^p e^{ct} \quad \forall t,\ -\infty < t < \infty,\ \forall c > 0, \qquad \text{(D.4)}
$$

and consequently ${}_{-\infty}D_t^p e^{ct} = c^p e^{ct}$. Assuming $G(s)$ is analytic in $(-\infty, 0]$ and defining $\nu = k - p$, one can use Eq. (7.21) and change the order of integration to get

$$
\begin{aligned}
{}_{-\infty}D_s^p G(s)|_{s=0} &= \Gamma(\nu)^{-1} \frac{d^k}{ds^k} \int_{-\infty}^s (s-\tau)^{\nu-1} \left[\int_0^\infty e^{\tau u} P(u)du\right] d\tau \Big|_{s=0} \\
&= \int_0^\infty \left[\Gamma(\nu)^{-1} \frac{d^k}{ds^k} \int_{-\infty}^s (s-\tau)^{\nu-1} e^{\tau u} d\tau\right] \Big|_{s=0} P(u)du \\
&= \int_0^\infty {}_{-\infty}D_s^p(e^{su})\Big|_{s=0} P(u)du = \int_0^\infty u^p e^{su}\Big|_{s=0} P(u)du \\
&= \int_0^\infty u^p P(u)du = \mu_p. \qquad \text{(D.5)}
\end{aligned}
$$

D.3 Inversion of the GL series

Eq. (7.24) can be inverted by expanding the series for the first few terms:

$$
f^{(2)}(t) = h^{-2} f(t) - 2h^{-2} f(t-h) + h^{-2} f(t-2h) \qquad \text{(D.6)}
$$

$$
f^{(1)}(t) = h^{-1} f(t) - h^{-1} f(t-h) \qquad \text{(D.7)}
$$

$$
f^{(0)}(t) = f(t). \qquad \text{(D.8)}
$$

Then by introducing Eq. (D.8) into Eq. (D.7) to eliminate $h^{-1} f(t)$ we get

$$f(t-h) = f(t) - hf^{(1)}(t), \tag{D.9}$$

similarly by introducing Eq. (D.8) and Eq. (D.7) into Eq. (D.6) to eliminate first two terms on the RHS we have

$$f(t-2h) = f(t) - 2hf^{(1)}(t) + h^2 f^{(2)}(t). \tag{D.10}$$

Eq.(7.26) easily follows by induction.

References

[1] M. Gorokhovski and M. Herrmann. Modeling primary atomization. *Annu Rev Fluid Mech*, 40:343–366, 2008. 2

[2] C.T. Crowe. On models for turbulence modulation in fluid/particle flows. *International Journal of Multiphase Flow*, 26:719–727, 2000. 2

[3] S. J. Scott, A. U. Karnik, and J. S. Shrimpton. On the quantification of preferential accumulation. *International Journal of Heat and Fluid Flow*, 30:789–795, 2009. 2, 209

[4] R.A Shaw. Particle-turbulence interactions in atmospheric clouds. *Annu. Rev. Fluid Mech*, 35:183–227, 2003. 2

[5] J.S Shrimpton. Pulsed charged sprays: Application ro disi engines during early injection. *Int. J. Numer. Meth. Engng.*, 58:513–536, 2003. 3

[6] G.M. Feath. Evaporation and combustion in sprays. *IEEE Transactions*, 1A-19:754–758, 1983. 3

[7] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer, 2002. 3, 13, 15, 48, 49, 74, 78, 87, 137, 224, 226, 228

[8] T. Ishihara, T. Gotoh, and Y. Kaneda. Study of high-reynolds number isotropic turbulence by direct numerical simulation. *Annual Review of Fluid Mechanics*, 41:165–180, 2009. 3

[9] J.R Grace. Contacting modes and behaviour classification of gas-solid and other two-phase suspensions. *Can. J. of Chem. Eng.*, 64:1953–1966, 1986. 4, 5

[10] Gidaspow. *Multiphase Flow and Fluidization*. Academic Press, 1994. 5, 175

[11] D. Geldart. Types of gas fluidization. *Powder Technol*, 7:185–195, 1973.

[12] R. Clift, J.R. Grace, and M.E. Weber. *Bubbles, Drops and particles*. Academic Press, 1978. 4

[13] H. T. Bi and J. R. Grace. Flow regime diagrams for gas-solid fluidization and upward transport. *Int. J. Multiphase Flow*, 21:1229–1236, 1995. 5

[14] H. T. Bi, J. R. Grace, and K. S. Lim. Transition from bubbling to turbulent fluidization. *Ind. Eng. Chem. Res.*, 34:4003–4008, 1995. 5

[15] M. Mandhane, G. A. Gregory, and K. Aziz. A flow pattern map for gas - liquid flow in horizontal pipes. *International Journal of Multiphase Flow*, 1:537–553, 1974. 5, 7

[16] D.H. Beggs and J.P. Brill. A study of two-phase flow in inclined pipes. *Journal of Petroleum Technology*, 25:607–617, 1973. 5

[17] J.L. Xu, P. Cheng, and T.S. Zhao. Gas liquid two-phase flow regimes in rectangular channels with mini/micro gaps. *International Journal of Multiphase Flow*, 25: 411–432, 1999. 5

[18] T. Furukawa and T. Fukano. Effect of lquid viscosity on flow patterns in vertical upward gas-liquid two-phase flow. *International Journal of Multiphase Flow*, 27: 1109–1126, 2001. 5

[19] Y. Taitel, D. Bornea, and A. E. Dukler. Modelling flow pattern transitions for steady upward gas-liquid flow in vertical tubes. *AIChE Journal*, 26:345–354, 1980. 6

[20] D. Barnea, O. Shoham, Y. Taitel, and A. E. Dukler. Flow pattern transition for gas-liquid flow in horizontal and inclined pipes. *Int. J. Multiphase Flow*, 6: 217–225, 1980. 7

[21] Y. Liu, W. Yang, and J. Wang. Experimental study for the stratified to slug flow regime transition mechanism of gas-oil two-phase flow in horizontal pipe. *Frontiers of Energy and Power Engineering in China*, 2:152–157, 2008. 7

[22] M.N. Pantzali, A.A. Mouza, and S.V. Paras. Countercurrent gas-liquid flow and incipient flooding in inclined small diameter tubes. *Chemical Engineering Science*, 63:3966–3978, 2008. 7

[23] A. Dyment and A. Boudlal. A theoretical model for gas-liquid slug flow in down inclined ducts. *International Journal of Multiphase Flow*, 30:521–550, 2004.

[24] A. Zapke and D. G. Kröger. Countercurrent gas-liquid flow in inclined and vertical ducts - i: Flow patterns, pressure drop characteristics and flooding. *International Journal of Multiphase Flow*, 26:1439–1455, 2000.

[25] E. Grolman and J. M. H. Fortuin. Gas-liquid flow in slightly inclined pipes. *Chemical Engineering Science*, 52:4461–4471, 1997. 7

[26] J. Lavieville, E. Deutsch, and O Simonin. Large eddy simulation of interactions between colliding particles and a homogeneous isotropic turbulence field. gas-particle flows. *ASME*, 228:359–369, 1995. 8

[27] Q. Wang, K.D. Squires, H. Chen, and J.B McLaughlin. On the role of the lift force in turbulence simulations of particle deposition. *Int. J. Multiphase Flow*, 23: 749–763, 1997. 8

[28] S. Patankar. *Numerical heat transfer and flow*. New York: Hemisphere, 1980. 13, 17, 48

[29] H. Versteeg and W. Malalasekra. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Prentice Hall, 2007. 13, 15, 74, 78, 80, 228

[30] J. Stewart. *Calculus: early transcendentals*. Thomson Brooks/Cole, 2008. 14

[31] H.P. Langtangen, K-A. Mardal, and R. Winther. Numerical methods for incompressible viscous flow. *Advances in Water Resources*, 25:1125–1146, 2002. 17, 18

[32] R. Brown, D.L. Cortez and M.L. Minionz. Accurate projection methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 168: 464–499, 2001. 18

[33] P.M. Gresho and R.L. Sani. *Incompressible flow and the finite element method*. New York: Wiley, 1998. 18

[34] J. Kim and P. Moin. Application of a fractional-step method to incompressible navier-stokes equations. *Journal Of Computational Physics*, 59:308–323, 1985. 18

[35] H. Le and P. Moin. An improvement of fractional step method for the incompressible navier-stokes equations. *Journal Of Computational Physics*, 92:369–379, 1991. 72

[36] M. Rai and P. Moin. Direct simulation of turbulent flow using finite-difference schemes. *J. Comput. Phys.*, 96: 15–53, 1991. 18

[37] F. Chalot, G. Chevalier, Q. V. Dinh, and L. Giraud. Some investigations of domain decomposition techniques in parallel cfd. *Lecture Notes in Computer Science*, 1685:595–602, 1999. 18

[38] G. Karypis and V. Kumar. Metis 4.0: Unstructure graph partitioning and sparse matrix ordering system. Technical report, Department of Computer Science, University of Minnesota, 1998. 19

[39] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *International Conference on Parallel Processing*, pages 113–122, 1995. 19, 20

[40] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal Parallel and Distributed Computing*, 48:96–129, 1998. 19, 20

[41] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison Wesley, 2005. 19

[42] A. Pothen, H. D. Simon, and .K-P. KLiou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11: 430–452, 1990. 19

[43] M. T. Heath and P. Raghavan. A cartesian nested dissection algorithm. *SIAM Journal on Matrix Analysis and Applications*, 16:235–253, 1995. 20

[44] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *Supercomputing, San Diego, CA*, 1995. 20

[45] E. Coffman, M.J. Elphick, and A. Shoshani. System deadlocks. *ACM Computing Surveys*, 3:67–78, 1971. 24

[46] Peter. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, 1996. 24, 26

[47] W. Gropp. *Using Mpi: Portable Parallel Programming With The Message-Passing Interface*. MIT Press; 2nd Revised edition, 2000. 25, 26

[48] M. Snir. *MPI: The Complete Reference*. MIT Press; 2nd Revised edition, 1998. 25

[49] J. Hauser, M. Spel, J. Muylaert, and R. Williams. Parnss: an efficient parallel navier-stokes solver for complex geometries. In *Proceedings of the 25th AIAA Fluid Dynamics Conference*, pages 94–2263, 1994. 26

[50] C. de Nicola, R. Tognaccini, and P. Visingardi. Multiblock structured algorithms in parallel cfd. In *Proceedings of the Parallel Computational Fluid Dynamics Conference*, pages 1–8, 1995.

[51] A. M. Wissink and R. R. Meakin. Computational fluid dynamics with adaptive overset grids on parallel and distributed computer platforms. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1628–1634, 1998. 26

[52] M. J. Djomehri and R. Biswas. Performance enhancement strategies for multi-block overset grid cfd applications. *Parallel Computing*, 29:1791–1810, 2003. 26

[53] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997. 35, 227

[54] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996. 35, 37, 38, 39

[55] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems*. SIAM, second edition edition, 1994. 39, 40, 224

[56] A.J.C. Ladd. Numerical simulations of particulate suspensions via a discretized boltzmann equation. part i. theoretical foundation. *J. Fluid Mech*, 271:285–310, 1994. 47

[57] A.J.C. Ladd. Numerical simulations of particulate suspensions via a discretized boltzmann equation. part ii: numerical results. *J. Fluid Mech*, 271:311–339, 1994. 47

[58] Z. G. Feng and E. E. Michaelides. The immersed boundary-lattice boltzmann method for solving fluid-particles interaction problems. *J. Comput. Phys.*, 195: 602–628, 2004. 47

[59] Z. G. Feng and E. E. Michaelides. Proteus: a direct forcing method in the simulations of particulate flows. *J. Comput. Phys.*, 202:20–51, 2005. 47

[60] C.M. Rhie and W.L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21:1525–1532, 1983. 48

[61] T.E. Tezduyar, M. Behr, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces - the deforming-spatial-domain/ space-time procedure: I. the concept and the preliminary numerical tests. *Comput. Methods Appl. Mech. Engrg.*, 94:339–351, 1992. 49, 50

[62] T.E. Tezduyar, M. Behr, S. Mittal, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces - the deforming-spatial-domain/space-time procedure: Ii. computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Comput. Methods Appl. Mech. Engrg.*, 94: 353–371, 1992. 49, 50

[63] A.A. Johnson and T.E. Tezduyar. Simulation of multiple spheres falling in a liquid-filled tube. *Comput. Methods Appl. Mech. Engrg.*, 134:351–373, 1996. 50, 51, 52

[64] A.A. Johnson and T.E. Tezduyar. 3d simulation of fluid-particle interactions with the number of particles reaching 100. *Comput. Methods Appl. Mech. Engrg.*, 145:301–321, 1997. 50, 51, 52

[65] A.A. Johnson and T.E. Tezduyar. Advanced mesh generation and update methods for 3d flow simulations. *Comput. Mech*, 23:130–143, 1999. 50, 51, 52

[66] TE Tezduyar and S. Sathe. Modeling of fluid-structure interactions with the space-time finite elements: solution techniques. *International Journal for Numerical Methods in Fluids*, 54:855–900, 2007. 50

[67] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T.E. Tezduyar. Computer modeling of cardiovascular fluid-structure interactions with the deforming-spatial-domain/stabilized space-time formulation. *Comput. Method Appl. M.*, 195(13-16):1885 – 1895, 2006.

[68] M. Manguoglu, K. Takizawa, A.H. Sameh, and T.E. Tezduyar. Nested and parallel sparse algorithms for arterial fluid mechanics computations with boundary layer mesh refinement. *Int. J. Numer. Meth. Fluids*, 65:135–149, 2011. 50

[69] H.H. Hu, D.D. Joseph, and M.J. Crochet. Direct simulation of fluid particle motion. *Thero. Comput. Fluid Dyn.*, 3:285, 1992. 50

[70] H.H. Hu, N.A. Patankar, and M.Y. Zhu. Direct numerical simulations of fluid - solid systems using the arbitrary lagrangian - eulerian technique. *J. Comput. Phys.*, 169:427–462, 2003. 50, 51, 52

[71] H. Gan, J.Z. Chang, J.J. Feng, and H.H. Hu. Direct numerical simulation of the sedimentation of solid particles with thermal convection. *J. Fluid Mech.*, 481: 385–411, 2003. 50, 52, 107, 137, 140, 145, 146, 147, 148

[72] C.R. Choi and C.N. Kim. Direct numerical simulations of the dynamics of particles with arbitrary shapes in shear flows. *J. Hydrodyn.*, 22:456–465, 2010. 50

[73] D.C. Wan and S. Turek. Direct numerical simulation of particulate flow via multigrid fem techniques and the fictitious boundary method. *Int. J. Numer. Method Fluids*, 51:531–566, 2006. 50

[74] D. Wan and S. Turek. Modeling of liquid-solid flows with large number of moving particles by multigrid fictitious boundary method. *J. Hydrodyn. Ser. B*, 18: 93–100, 2006. 50

[75] D. Wan and S. Turek. An efficient multigrid-fem method for the simulation of solid-liquid two phase flows. *J. Comput. Appl. Math.*, 203:561–580, 2007. 50, 52

[76] D. Wan and S. Turek. Fictitious boundary and moving mesh methods for the numerical simulation of rigid particulate flows. *Journal of Computational Physics*, 222:28–56, 2007. 50, 60

[77] S.O Unverdi and G. Truggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *J. Comput. Phys.*, 100:25–37, 1992. 50

[78] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y-J. Jan. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169:708–759, 2001. 50

[79] P. Zhao, J.C. Heinrich, and D.R. Poirier. Numerical modeling of fluid-particle interactions. *Comput. Methods Appl. Mech. Engrg*, 195:5780–5796, 2006. 50

[80] P. Zhao, J.C. Heinrich, and D.R. Poirier. Dendritic solidification of binary alloys with free and forced convection. *Int. J. Numer. Methods Fluids*, 49:233–266, 2005.

[81] P. Zhao, M. Venere, J.C. Heinrich, and D.R. Poirier. Modeling dendritic growth of a binary alloy. *J. Comput. Phys.*, 188:434–461, 2003. 50

[82] A.A. Johnson and T.E. Tezduyar. Models for 3d computations of fluid - particle interactions in spatially periodic flows. *Comput. Methods Appl. Mech. Engrg.*, 190:3201–3221, 2001. 51, 52

[83] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–61, 2005. 53, 54, 62, 76

[84] C.S. Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 10:252–271, 1972. 55, 69

[85] C.S. Peskin. The fluid dynamics of heart valves: Experimental, theoretical and computational methods. *Annu. Rev. Fluid Mech.*, 14:235–259, 1981. 55

[86] A.A. Mayo and C.S. Peskin. An implicit numerical method for fluid dynamics problems with immersed elastic boundaries. In *Fluid Dynamics in Biology: Proceedings of an AMS-IMS-SIAM Joint Summer Research Conference, Contemporary Mathematics*, volume 140, pages 261–277, 1993. 55

[87] E. P. Newren, A. L. Fogelson, R. D. Guy, and R. M. Kirby. A comparison of implicit solvers for the immersed boundary equations. *Comput. Methods Appl. Mech. Engrg.*, 197:2290–2304, 2008. 55

[88] E. P. Newren, A. L. Fogelson, R. D. Guy, and R. M. Kirby. Unconditionally stable discretizations of the immersed boundary equations. *Journal of Computational Physics*, 222:702–719, 2007.

[89] Y. Mori and C.S. Peskin. Implicit second order immersed boundary methods with boundary mass. *Comput. Meth. Appl. Mech. Eng.*, 197:2049–2067, 2008. 55

[90] C. S. Peskin. The immersed boundary method. *Acta Numer.*, 11:479–517, 2002. 56, 57, 58

[91] B. E. Griffith, R. D. Hornung, D. M. McQueen, and C. S. Peskin. An adaptive, formally second order accurate version of the immersed boundary method. *Journal of Computational Physics*, 223:10–49, 2007. 56, 57

[92] M.C. Lai and C.S. Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *J. Comput. Phys.*, 160:705–719, 2000. 57, 60, 62

[93] A.M. Roma, C.S. Peskin, and M.J. Berger. An adaptive version of the immersed boundary method. *J. Comput. Phys.*, 153:509–534, 1999. 57

[94] X. Yang, X. Zhang, Z. Li, and G.-W. He. A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations. *Journal of Computational Physics*, 228:7821–7836, 2009. 58, 59

[95] E. Saiki and S. Biringen. Numerical simulation of a cylinder in uniform flow: Application of a virtual boundary method. *Journal of Computational Physics*, 123:450–463, 1996. 60, 62, 63

[96] E. Balaras. Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations. *Computers & Fluids*, 33:375–404, 2004. 60, 73

[97] L. Shen, E.S. Chan, and P. Lin. Calculation of hydrodynamic forces acting on a submerged moving object using immersed boundary method. *Computers & Fluids*, 38:691–702, 2009. 60, 74

[98] J. Blasco, M. C. Calzada, and M. Marín. A fictitious domain parallel numerical method for rigid particulate flows. *Journal of Computational Physics*, 228:7596–7613, 2009. 60

[99] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007. 61

[100] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, 1999. 62

[101] W.-X. Huang and H.J. Sung. An immersed boundary method for fluid-flexible structure interaction. *Computer Methods in Applied Mechanics and Engineering*, 198:2650–2661, 2009. 62

[102] D. Goldstein, Handler. R., and L. Sirovich. Modeling a no-slip boundary with an external force field. *J. Comput. Phys.*, 105:354–366, 1993. 62, 67

[103] S. J. Shin, W.-X. Huang, and H.J. Sung. Assessment of regularized delta functions and feedback forcing schemes for an immersed boundary method. *Int. J. Numer. Meth. Fluids*, 58:263–286, 2008. 62, 63

[104] M. Uhlmann. An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, 209:448–476, 2005. 63, 72, 73, 74, 135, 139, 140, 141

[105] A.W. Data. *Introduction to Computational Fluid Dynamics*. Cambridge University Press, 2005. 64

[106] T. J. Chung. *Computational Fluid Dynamics*. Cambridge University Press, 2002. 64

[107] M. Vinokur. On one-dimensional stretching functions for finite-difference calculations. Technical Report CR-3313, NASA, 1980. 64

[108] E. Guilmineau and P. Queutey. A numerical simulation of vortex shedding from an oscillating circular cylinder. *Journal of Fluids and Structures*, 16:773–794, 2002. 64, 68

[109] P. Angot, C.H. Bruneau, and P. Frabrie. A penalization method to take into account obstacles in viscous flows. *Numer. Math.*, 81:497–520, 1999. 68, 69

[110] K. Khadra, P. Angot, S. Parneixc, and J.-P. Caltagironec. Fictitious domain approach for numerical modelling of navier-stokes equations. *Int. J. Numer. Meth. Fluids*, 34:651–684, 2000. 68, 69

[111] A. L. Fogelson and C. S. Peskin. Afastnumericalmethod for solving the three-dimensionalstokes' equations in the presence of suspended particles. *Journal of Computational Physics*, 79:50–69, 1988. 69

[112] A. L. F. Lima E Silva, A. Silveira-Neto, and J. J. R. Damasceno. Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method. *Journal of Computational Physics*, 189:351–370, 2003. 69

[113] J. Mohd-Yosuf. Combined immersed boundary/b-spline methods for simulation of flow in complex geometries. *Annu. Res. Briefs, Cent. Turbul. Res*, pages 317–328, 1997. 72

[114] E. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for threedimensional complex flow simulations. *J. Comput. Phys.*, 161:35–60, 2000. 73, 74

[115] R. Verzicco, J. Mohd-Yusof, P. Orlandi, and D. Howarth. Les in complex geometries using boundary body forces. *AIAA J.*, 38:427–433, 2000.

[116] Z. Wang, J. Fan, and K. Luo. Combined multi-direct forcing and immersed boundary method for simulating flows with moving particles. *Int. J. Multiphase Flow*, 34:283–302, 2008. 72

[117] S. Majumdar, G. Iaccarino, and PA. Durbin. Rans solver with adaptive structured boundary non-conforming grids. *Annu. Res. Briefs, Cent. Turbul. Res.*, pages 353–364, 2001. 72, 76

[118] A. Gilmanov, F. Sotiropoulos, and E. Balaras. A general reconstruction algorithm for simulating flows with complex 3d immersed boundaries on cartesian grids. *J. Comput. Phys.*, 191:660–669, 2003. 76, 87

[119] A. Gilmanov and F. Sotiropoulos. A hybrid cartesian/immersed boundary method for simulating flows with 3d, geometrically complex, moving bodies. *Journal of Computational Physics*, 207:457–492, 2005. 87

[120] Y-H. Tseng and J.H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *J. Comput. Phys.*, 192:593–623, 2003. 76

[121] R. Ghias, R. Mittal, and H. Dong. A sharp interface immersed boundary method for viscous compressible flows. *J. Comput. Phys.*, 225:528–553, 2007. 76

[122] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, and A. von Loebbecke. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of Computational Physics*, 227:4825–4852, 2008. 72, 76, 77, 78, 94, 97, 155, 157

[123] M. P. Kirkpatrick, S. W. Armfield, and J. H. Kent. A representation of curved boundaries for the solution of the navier-stokes equations on a staggered three-dimensional cartesian grid. *J. Comput. Phys*, 184: 1–36, 2003. 72, 87

[124] P. G. Tucker and Z. Pan. Cartesian cut cell method for incompressible viscous flow. *Appl. Math. Modell*, 24: 591–606, 2000.

[125] H. S. Udaykumar, R. Mittal, and W. Shyy. Computation of solid-liquid phase fronts in the sharp interface limit on fixed grids. *Journal of Computational Physics*, 153:535–574, 1999.

[126] H. S. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna. A sharp interface cartesian grid method for simulating flows with complex moving boundaries. *Journal of Computational Physics*, 174:345–380, 2001.

[127] H. S. Udaykumar, R. Mittal, and P. Rampunggoon. Interface tracking finite volume method for complex solid-fluid interactions on fixed meshes. *Commun. Numer. Meth. Engng*, 18:89–97, 2002. 72

[128] A.S. Almgren, J.B. Bell, P. Colella, and T. Marthaler. A cartesian grid projection method for the incompressible euler equations in complex geometries. *SIAM J. Sci. Comput*, 18:1289–1309, 1997. 72

[129] H. Johansen and P. Colella. A cartesian grid embedded boundary method for poissonŠs equation on irregular domains. *J. Comput. Phys*, 147:60–85, 1998. 72

[130] S. Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Comput. Phys*, 190:572–600, 2003. 72

[131] J. Kim, D. Kim, and H. Choi. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *J. Comput. Phys.*, 171:132–150, 2001. 72

[132] D. Kim and H. Choi. Immersed boundary method for flow around an arbitrarily moving body. *J. Comput. Phys.*, 212:662–680, 2006. 72, 98, 103, 104

[133] L. Shen and E.S. Chan. Numerical simulation of fluid-structure interaction using a combined volume of fluid and immersed boundary method. *Ocean Engineering*, 35:939–952, 2008. 74

[134] R. Ghias, R. Mittal, and T.S. Lund. A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries. *AIAA*, page 80, 2004. 76

[135] A. Mark and B. G.M. van Wachem. Derivation and validation of a novel implicit second-oder accurate immersed boundary method. *Journal of Computational Physics*, 227:6660–6680, 2008. 76, 77, 78, 157

[136] T. Ye, R. Mittal, H.S. Udaykumar, and W. Shyy. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. *J. Comput. Phys.*, 156:209–240, 1999. 78, 90, 94, 97

[137] S. Takiguchi, T. Kajishima, and Y. Miyake. Numerical scheme to resolve the interaction between solid-particles and fluid-turbulence. *JSME Int. J. Ser. B*, 42 (3):411–418, 1999. 79

[138] T. Kajishima and S. Takiguchi. Interaction between particle clusters and particle-induced turbulence. *International Journal of Heat and Fluid Flow*, 23:639–646, 2002. 79

[139] T. Kajishima. Influence of particle rotation on the interaction between particle clusters and particle-induced turbulence. *International Journal of Heat and Fluid Flow*, 25:721–728, 2004. 79

[140] R. Glowinski, T-W. Pan, and J. Periaux. A fictitious domain method for dirichlet problems and applications. *Comp. Meth. Appl. Mech. Engng.*, 111: 283–303, 1994. 80

[141] R. Glowinski, T-W. Pan, and J. Periaux. A fictitious domain method for external incompressible viscous flow modeled by navier-stokes equations. *Comp. Meth. Appl. Mech. Engng.*, 112:133–148, 1994. 80

[142] R. Glowinski, T-W. Pan, and J. Periaux. Distributed lagrange multiplier methods for incompressible viscous flow around moving rigid bodies. *Comp. Meth. Appl. Mech. Engng*, 151:181–194, 1998. 80

[143] R. Glowinski, T.-W. Pan, T.I. Hesla, and D.D. Joseph. A distributed lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25:755–794, 1999. 80, 134, 135, 138, 139

[144] R. Glowinski, T.-W. Pan, T.I Hesla, D.D. Joseph, and J. Periaux. A distributed lagrange multiplier fictitious domain method for the simulation of flow around moving rigid bodies: application to particulate flow. *Comput. Methods Appl. Mech. Engrg*, 184:241–267, 2000. 80, 135, 138, 139

[145] N.A. Patankar, P. Singh, D.D. Joseph, R. Glowinski, and T.-W. Pan. A new formulation of the distributed lagrange multiplier/ fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 26:1509–1524, 2000. 80

[146] N. A. Patankar. A formulation for fast computations of rigid particulate flows. *Cent. Turb. Res. Annu. Res. Briefs*, 2001. 80, 133

[147] Z. Yu and X. Shao. A direct-forcing fictitious domain method for particulate flows. *J. Comput. Phys.*, 227: 292–314, 2007. 80, 82

[148] C. Diaz-Goano, P. Minev, and K. Nandakumar. A fictitious domain/finite element method for particulate flows. *J. Comput. Phys*, 192:105–123, 2003. 80

[149] N. Sharma and N.A. Patankar. A fast computation technique for the direct numerical simulation of rigid particulate flows. *Journal of Computational Physics*, 205:439–457, 2005. 80, 82, 88, 112

[150] C. Veeramani, P. Minev, and K. Nandakumar. A fictitious domain formulation for flow with rigid particles: a non-lagrangian multiplier version. *Journal of Computational Physics*, 205:439–457, 2007. 80, 133

[151] S. V. Apte, M. Martin, and N. A. Patankar. A numerical method for fully resolved simulation (frs) of rigid particle-flow interactions in complex flows. *Journal of Computational Physics*, 228:2712–2738, 2009. 80, 82, 97, 98, 101, 103, 133, 155, 157, 158

[152] A. M. Ardekani, S. Dabiri, and R. H Rangel. Collision of multi-particle and general shape objects in a viscous fluid. *Journal of Computational Physics*, 227:10094–10107, 2008. 85, 86, 139

[153] J.B. Bell, P. Colella, and H.M Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 85: 257–283, 1989. 87

[154] R. Mittal and S. Balachandar. effect of three-dimensionality on the lift and drag of nominally two-dimensional cylinders. *Phys. Fluids*, 7:1841–1865, 1995. 88, 98, 103, 115

[155] C. H. K. Williamson. Vortex dynamics in the cylinder wake. *Annu. Rev. Fluid. Mech.*, 28:477–539, 1996. 88

[156] M. Behr, D. Hastreiter, S. Mittal, and T.E Tezduyar. Incompressible flow past a circular cylinder: dependence of the computed flow field on the location of the lateral boundaries. *Computer Methods in Applied Mechanics and Engineering*, 123:309–316, 1995. 88

[157] G. Z. Dennis, S. C. R.and Chang. Numerical solution for steady flow past a circular cylinder at reynolds number up to 100. *J. Fluid Mech*, 42:471–489, 1970. 90

[158] B. Fornberg. A numerical study of steady viscous-flow past a circular-cylinder. *J. Fluid Mech.*, 98:819–855, 1980. 90

[159] C. H. K. Williamson. The natural and forced formation of spot-like Śvortex dislocationsŠ in the transition of a wake. *Journal of Fluid Mechanics*, 243:393–441, 1992. 94, 97

[160] Y. Zang, R. L. Street, and J. R. Koseff. A non-staggered grid, fractional step method for time-dependent incompressible navier-stokes equations in curvilinear coordinates. *Journal of Computational Physics*, 114:18–33, 1994. 94, 97

[161] D. J. Tritton. Experiments on the flow past a circular cylinder at low reynolds number. *J. Fluid Mech.*, 6: 547–567, 1959. 97

[162] S. Marella, S. Krishnan, H. Liu, and H. S Udaykumar. Sharp interface cartesian grid method i: An easily implemented technique for 3d moving boundary computations. *Journal of Computational Physics*, 210:1–31, 2005. 97, 155, 157

[163] H. Dutsch, F. Durst, S. Becker, and H. Lienhart. Low-reynolds-number flow around an oscillating circular cylinder at low keulegan-carpenter numbers,. *Journal of Fluid Mechanics*, 360:249–271, 1998. 98

[164] J. Feng, H. H. Hu, and D. D. Joseph. Direct simulation of initial value problems for the motion of solid bodies in a newtonian fluid. part 1. sedimentation. *J. Fluid. Mech.*, 261:95–134, 1994. 107, 137, 138

[165] H. S. Yoon, Y. B. Lee, and H. H Chun. A numerical study on the fluid flow and heat transfer around a circular cylinder near a moving wall. *Int. J. Heat Mass Transfer*, 50:3507–3520, 2007. 107

[166] J. Kim and H. Choi. An immersed-boundary finite-volume method for simulation of heat transfer in complex geometries. *Korean Soc. Mech. Eng. Int. J.*, 18 .:1026–1035, 2004. 107, 125, 129

[167] J. R. Pacheco, T. Rodic, R. E. Peck, and A. Pacheco-Vega. Numerical simulations of heat transfer and fluid flow problems using an immersed-boundary finite-volume method on nonstaggered grids. *Numerical Heat Transfer, Part B*, 47:1–24, 2005. 107

[168] D. Pan. An immersed boundary method on unstructured cartesian meshes for incompressible flows with heat transfer. *Numerical Heat Transfer, Part B*, 49: 277–297, 2006. 107, 108

[169] B. S. Kim, D. S. Lee, M. Y. Ha, and H. S. Yoon. A numerical study of natural convection in a square enclosure with a circular cylinder at different vertical locations. *Int. J. Heat Mass Transfer*, 51:1888–1906, 2008. 107

[170] N. Zhang, Z. C. Zheng, and S. Eckels. Study of heat-transfer on the surface of a circular cylinder in flow using an immersed-boundary method. *International Journal of Heat and Fluid Flow*, 29:1558–1566, 2008. 108, 125, 129

[171] Z. Wang, J. Fan, K. Luo, and K. Cen. Immersed boundary method for the simulation of flows with heat transfer. *International Journal of Heat and Mass Transfer*, 52:4510–4518, 2009. 108, 125, 129

[172] Z. G. Feng and E. E. Michaelides. Heat transfer in particulate flows with direct numerical simulation (dns). *International Journal of Heat and Mass Transfer*, 52:777–786, 2009. 108

[173] Z. Yu, X. Shao, and A. Wachs. A fictitious domain method for particulate flows with heat transfer. *Journal of Computational Physics*, 217:424–452, 2006. 108, 137, 145, 146, 148, 149, 150

[174] Z. Yu, X. Shao, and A. Wachs. A fictitious domain method for particulate flows with heat transfer. *Journal of Computational Physics*, 217:424–452, 2006. 108

[175] A. Wachs. Rising of 3d catalyst particles in a natural convection dominated flow by a parallel dns method. *Computers and Chemical Engineering*, 35: 2169–2185, 2011. 108, 149, 150

[176] I. Demirdzic, Z Lilek, and M. Peric. Fluid flow and heat transfer test problems for nonorthogonal grids: benchmark solutions. *Int. J. Numer. Meth. Fluids*, 15:329–354, 1992. 112, 113

[177] A. A. Soares, J. M. Ferreira, and R. P. Chhabra. Flow and forced convection heat transfer in crossflow of non-newtonian fluids over a circular cylinder. *Ind Eng Chem Res*, 44:5815–5827, 2005. 114, 115

[178] R. P. Bharti, R. P. Chhabra, and V. Eswaran. A numerical study of the steady forced convection heat transfer from an unconfined circular cylinder. *Heat and Mass Transfer*, 43:639–648, 2007. 114, 115, 125, 129

[179] I. B. Celik, U. Ghia, P.J. Roache, C.J. Freitas, H. Coleman, and P. E. Raad. Procedure for estimation and reporting of uncertainty due to discretization in cfd applications. *Journal of Fluids Engineering*, 130:078001–4, 2008. 114

[180] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Publications Inc, 2003. 114

[181] K. Mahesh, G. Constantinescu, and P. Moin. A numerical method for large-eddy simulation in complex geometries. *Journal of Computational Physics*, 197: 215–240, 2004. 115

[182] H.C. Perkins. Forced convection heat transfer from a uniformly heated cylinder. *J. Heat Transfer*, 84:257–263, 1962. 115

[183] H.C. Perkins and G Leppert. Local heat-transfer coefficients on a uniformly heated cylinder. *Int. J. Heat Mass Transfer*, 7:143–158, 1964. 115

[184] R. M. Fand. Heat transfer by forced convection from a cylinder to water in crossflow. *Int. J. Heat Mass Transfer*, 8:995–1010, 1965. 115, 116, 117

[185] A. Zukauskas and J Ziugzda. *Heat Transfer of a Cylinder in Crossflow*. Hemishere Pub, 1985. 116

[186] S Whitaker. Forced convection heat transfer calculations for flow in pipes, past flat plate, single cylinder, and for flow in packed beds and tube bundles. *AIChE J*, 18:361–371, 1972. 116, 117

[187] S.W. Churchill and M Bernstein. A correlating equation for forced convection from gases and liquids to a circular cylinder in cross flow. *J. Heat Transfer*, 99: 300–306, 1977. 116

[188] S. Sanitjai and R.J Goldstein. Heat transfer from a circular cylinder to mixtures of water and ethylene glycol. *International Journal of Heat and Mass Transfer*, 47:4785–4794, 2004. 116

[189] S. Sanitjai and R. J Goldstein. Forced convection heat transfer from a circular cylinder in crossflow to air and liquids. *International Journal of Heat and Mass Transfer*, 47:4795–4805, 2004. 116

[190] E. M. Sparrow, J. P. Abraham, and J. C. K. Tong. Archival correlations for average heat transfer coefficients for non-circular and circular cylinders and for spheres in crossflow. *Int J Heat Mass Transf*, 47: 5285–5296, 2004. 116, 117

[191] J.B. Copas. Regression, prediction and shrinkage. *Journal of the Royal Statistical Society. Series B*, 45:311–354, 1983. 121

[192] L. Ricci. Adjusted r-squared type measure for exponential dispersion models. *Statistics and Probability Letters*, 80:1365–1368, 2010. 121

[193] E. R. G. Eckert and E. Soehngen. Distribution of heat-transfer coefficients around circular cylinders in crossflow at reynolds numbers from 20 to 500. *Trans. ASME*, 75:343–347, 1952. 125, 129

[194] H. Goldstein, C.P Poole, and J.L Safko. *Classical Mechanics*. Addison Wesley, 2000. 132, 133

[195] P. Singh, D. D. Joseph, T. I. Hesla, R. Glowinski, and T. W. Pan. Direct numerical simulation of viscoelastic particulate flows. *J. Non-Newtonian Fluid Mech.*, 91: 165–188, 2000. 134

[196] P. Gondret, M. Lance, and L. Petit. Bouncing motion of spherical particles in fluids. *PHYSICS OF FLUIDS*, 14:643–652, 2002. 134

[197] Z. Yu, N. Phan-Thien, and R. I. Tanner. Dynamical simulation of sphere motion in a vertical tube. *J. Fluid Mech*, 518:61–93, 2004. 137, 145

[198] T.A Johnson and V.C Patel. Flow past a sphere up to a reynolds number of 300. *J. Fluid Mech*, 378:19–70, 1999. 154, 155, 157

[199] A. Ten Cate, C. H. Nieuwstad, J. J. Derksen, and H. E. A. Van den Akker. Particle imaging velocimetry experiments and lattice-boltzmann simulations on a single sphere settling under gravity. *PHYSICS OF FLUIDS*, 14:4012–4025, 2002. 157, 158, 160

[200] S Rigopoulos. Population balance modelling of polydispersed particles in reactive flows. *Progress in Energy and Combustion Science*, 36:412–443, 2010. 161, 162, 192

[201] Jean-Pierre Minier and Eric Peirano. The pdf approach to turbulent polydispersed two-phase ows. *Physics Reports*, 352:1–214, 2001. 161, 163, 166, 174, 175, 177, 178, 179, 182, 183, 184, 185, 186, 187

[202] NA Chigier. The atomization and burning of liquid fuel sprays. *Progress in Energy and Combustion Science*, 2:97–114, 1976. 162

[203] NA Chigier. Challenges for future research in atomization and spray technology. *Automization and Sprays*, 16:727–736, 2006.

[204] WA Sirignano. Fuel droplet vaporization and spray combustion theory. *Progress in Energy and Combustion Science*, 9:291–322, 1983.

[205] M Faeth. Transport and combustion in sprays. *Progress in Energy and Combustion Science*, 13: 293–345, 1987. 162

[206] S. J. Scott. *A PDF Based Method forModelling Polysized Particle Laden Turbulent Flows Without Size Class Discretisation.* PhD thesis, Imperial College London, 2006. 162, 163, 174, 177, 179, 180, 181, 183, 209

[207] Y. Tambour. A sectional model for evaporation and combustion of sprays of liquid fuels. *Israel J. Tech*, 18:47–56, 1980. 162

[208] J. B. Greenberg, I. Silverman, and Y. Tambour. On the origins of spray sectional conservation equations. *Combustion and Flame*, 93:90–96, 1992. 162

[209] K. Domelevo. The kinetic sectional approach for noncolliding evaporating sprays. *Atomization and Sprays*, 11:291–303, 2001.

[210] F. Laurent, M. Massot, and P. Villedieu. Eulerian multi-fluid modeling for the numerical simulation of coalescence in polydisperse dense liquid sprays. *Journal of Computational Physics*, 194:505–543, 2004.

[211] F. Laurent and M. Massot. Multi-fluid modelling of laminar polydispersed spray flames: Origin, assumptions and comparrison of sectional and sampling methods. *Combustion Theory Modelling*, 5:537–572, 2001.

[212] M. R. Archambault, C. F. Edwards, and R. W. McCormack. Computation of spray dynamics bymoment transport equations i: Theory and development. *Atomization and Sprays*, 13:63–87, 2003. 162, 163, 174, 193

[213] J. C. Beck and A. P. Watkins. Spray modelling using the moments of the droplet size distribution. In *ILASS-Europe*, 1999. 163

[214] J. C. Beck and A. P. Watkins. The droplet number moments approach to spray modelling: The development of heat and mass transfer sub-models. *Int. J. Heat Fluid Flow*, 24:242–259, 2003. 163, 192, 193

[215] J. C. Beck and A. P. Watkins. Modelling polydispersed sprays without discretisation into droplet size classes. In *ILASS Pasadena*, 2000. 163

[216] C.W. Gardiner. *Handbook Of Stochastic Methods For Physics, Chemistry And Natural Sciences.* Springer-Verlag, 2004. 166, 167, 168

[217] P.E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations.* Springer - Verlag, 1992. 166

[218] P. Mazur. On theory of brownian motion. *Physica*, 25: 149–162, 1959. 168

[219] P. Mazur and I. Oppenheim. Molecular theory of brownian motion. *Physica*, 50:241–258, 1970. 168

[220] N.G. Van Kampen. *Stochastic Processes In Physics And Chemistry.* Elsevier, 2007. 168, 171

[221] B. Oksendal. *Stochastic Differential Equations, An Introduction with Applications.* Springer, 1995. 172

[222] S. B. Pope. Application of the velocity-dissipation probability density function model in inhomogeneous turbulent flows. *Phys. Fluids A*, 3:1947–1957, 1991. 173

[223] J. J. E. Williams and R. I. Crane. Particle collision rate in turbulent flow. *International Journal of Multiphase Flow*, 9:421–435, 1983. 175

[224] E. Peirano and B. Leckner. Fundamentals of turbulent gas-solid flows applied to circulating fluidized bed combustion. *Prog. Energy Combust. Sci*, 24:259–296, 1998. 175, 183

[225] Augusto Neri and Dimitri Gidaspow. Riser hydrodynamics: Simulation using kinetic theory. *AIChE Journal*, 46:52–67, 2000. 175

[226] E. Peirano, S. Chibbaro, J. Pozorski, and J.-P. Minier. Mean-field/pdf numerical approach for polydispersed turbulent two-phase flows. *Progress in Energy and Combustion Science*, 32:315–371, 2006. 178, 180

[227] S. B. Pope. Pdf methods for turbulent reactive flows. *Progress in Energy and Combustion Science*, 11: 119–192, 1985. 179

[228] S.B. Pope. Application of the velocity-dissipation probability density function model in inhomogeneous turbulent flows. *Phys. Fluids A*, 3(8):1947–1957, 1991. 179

[229] S. B. Pope and Y. L. Chen. The velocity-dissipation probability density function model for turbulent flows. *Phys. Fluids A*, 2(8):1437–1449, 1990. 179

[230] S.B. Pope. *Turbulent Flows.* Cambridge University Press, 2001. 179, 180, 183

[231] S. Haeri and J.S. Shrimpton. A mesoscopic description of polydispersed particle laden turbulent flows. *Progress in Energy and Combustion Science*, pages 1–25, 2011. 179

[232] J-P Minier and J. Pozorski. Derivation of a pdf model for turbulent flows based on principles from statistical physics. *Phys. Fluids*, 9(6):1748–1753, 1997. 180

[233] S.B Pope. On the relationship between stochastic lagrangian models of turbulence and second-moment closures. *Phys. Fluids*, 6:973–985, 1994. 183

[234] S. B. Pope. Lagrangian pdf methods for turbulent flows. *Annu. Rev. Fluid Mech.*, 26:23–63, 1994.

[235] H. A. Wouters, T. W. J. Peeters, and D. Roekaerts. On the existence of a generalized langevin model representation for second-moment closures. *Phys. Fluids*, 8: 1702–1704, 1996. 183

[236] K. Hanjalic and B. E. Launder. A reynolds stress model of turbulence and its application to thin shear flows. *J. Fluid. Mech*, 52:609–638, 1972. 183

[237] O. Simonin. *Statistical And Continuum modelling Of Turbulent Reactive Particulate Flows. Part 1: Theoretical Derivation Of Dispersed Phase Eulerian Modelling From Probability Density Function Kinetic Equation.* In Lecure Series Von-Karman Institute for Fluid Dynamics, 2000. 184

[238] O. Simonin, E. Deutsch, and M. Boivin. Comparison of large-eddy simulation and second-moment closure of particle fluctuating motion in two-phase turbulent shear flows. In *Turbulence and Shear Flows 9*, pages 85–115. Springer-Verlag, 1995. 185, 186

[239] O. Simonin, E. Deutsch, and M. Bovin. Large eddy simulation and second-moment closure model of particle fluctuatingmotion in two-phase turbulent shear flows. *Turbulent Shear Flows*, 9, 1993. 185

[240] L. I. Zaichik and V.M. Alipchenkov. A statistical model for transport and deposition of high-inertia colliding particles in turbulent flow. *International Journal of Heat and Fluid Flow*, 22:365–371, 2001. 188

[241] M. Frenklach and S. J Harris. Aerosol dynamics modeling using the method of moments. *Journal of Colloid and Interface Science*, 118:252–261, 1987. 192

[242] R. B. Diemer and J. H. Olson. A moment methodology for coagulation and breakage problems: Part 3-generalized daughter distribution functions. *Chemical Engineering Science*, 57:4187–4198, 2002. 192

[243] R. B. Diemer and J. H. Olson. A moment methodology for coagulation and breakage problems: Part 1-analytical solution of the steady-state population balance. *Chemical Engineering Science*, 57:2193–2209, 2002.

[244] R. B. Diemer and J. H. Olson. A moment methodology for coagulation and breakage problems: Part 2Ůmoment models and distribution reconstruction. *Chemical Engineering Science*, 57:2211–2228, 2002. 192

[245] R.B Diemer and J.H. Olson. Bivariate moment methods for simultaneous coagulation, coalescence and breakup. *Journal of Aerosol Science*, 37:363–385, 2006. 192

[246] J.C. Beck and A.P Watkins. On the development of spray submodels based on droplet size moments. *Journal of Computational Physics*, 182:586–621, 2002. 192

[247] V. John, I. Angelov, A. A. Oncul, and D. Thevenin. Techniques for the reconstruction of a distribution from a finite number of its moments. *Chemical Engineering Science*, 62:2890–2904, 2007. 192, 193, 194, 208

[248] A Tagliani. Hausdorff moment problem and maximum entropy: A unified approach. *Applied Mathematics and Computation*, 105:291–305, 1999. 193

[249] P.N. Inverardi, G. Pontuale, A. Petri, and A. Tagliani. Hausdorff moment problem via fractional moments. *Applied Mathematics and Computation*, 144:61–74, 2003.

[250] P.N. Inverardi, G. Pontuale, A. Petri, and A. Tagliani. Stieltjes moment problem via fractional moments. *Applied Mathematics and Computation*, 166:664–677, 2005.

[251] M.B. Pintarellia and F Vericat. Generalized hausdorff inverse moment problem. *Physica A: Statistical Mechanics and its Applications*, 324:568–588, 2003. 193

[252] G. Talenti. Recovering a function from a finite number of moments. *Inverse Problems*, 3:501–517, 1987. 193

[253] E. V. Volpe and D. Baganoff. Maximum entropy pdfs and the moment problem under near-gaussian conditions. *Probabilistic Engineering Mechanics*, 18:17–29, 2003. 193

[254] B. O. Koopman. Relaxed motion in irreverible molecular statistics. *Stochastic Processes in Chem. Phys.*, 15:37–63, 1969. 193

[255] C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–623, 1948. 193

[256] J. Paris and A. Vencovska. In defense of the maximum entropy inference process. *Int. J. Approx. Reasoning*, 17:77–103, 1997. 193

[257] S. Blinnikov and R. Moessner. Expansions for nearly gaussian distributions. *Astron. Astrophys. Suppl. Ser.*, 130:193–205, 1998. 193

[258] M Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1972. 194

[259] A. K Majumdar, C.E Luna, and P.S Idell. Reconstruction of probability density function of intensity fluctuations relevant to free-space laser communications through atmospheric turbulence. In *Proc. SPIE*, 2007. 194, 195

[260] M.G. Kendall, A. Stuart, and J.K Ord. *Kendall's Advanced Theory of Statistics: Vol.1: Distribution Theory*. John Wiley & Sons Inc, 1991. 194

[261] E. Gaztanaga, P. Fosalba, and E. Elizalde. Gravitational evolution of the large-scale probability density. *Astrophys. J.*, 539:522–531, 2000. 194

[262] A.M. Mood, F .A. Graybill, and D.C. Boes. *Introduction to the Theory of Statistics*. McGraw-Hill, 1974. 194, 196

[263] H. Mustapha and R. Dimitrakopoulos. Generalized laguerre expansions of multivariate probability densities with moments. *Computers & Mathematics with Applications*, 60:2178–2189, 2010. 195, 196

[264] W.J Cody. An overview of software development for special functions. *Lecture Notes in Mathematics*, 506: 38–48, 1976. 195

[265] N.N. Lebedev. *Special Functions and Their Applications*. Dover Publications, 1972. 195

[266] B. Ross. Fractiorial calculus: an historical apologia for i he devcloprrlent of a calculus using differentiation and antidifferentiation of non integral orders. *Mathematics Magazine*, 50:115–122, 1977. 197

[267] I Podlubny. *Fractional differential equations*. ACADEMIC PRESS, 1999. 197, 199, 200, 201

[268] R Hilfer. *Fractional calculus in physics*. World Scientific, 2000. 197, 199, 200

[269] S.E Schiavone and W. W. Lamb. A fractional power approach to fractional calculus. *Journal of mathematical analysis and applications*, 149:377–401, 1990. 197

[270] G. Cottone and M Di Paola. On the use of fractional calculus for the probabilistic characterization of random variables. *Probabilistic Engineering Mechanics*, 24:321–334, 2009. 197, 199

[271] W. Deng. Short memory principle and a predictor-corrector approach for fractional differential equations. *Journal of Computational and Applied Mathematics*, 206:174–188, 2007. 200

[272] H. Gzyl and A. Tagliani. Hausdorff moment problem and fractional moments. *Applied Mathematics and Computation*, 216:3319–3328, 2010. 200, 201, 207

[273] A. Alexiadisa, M. Vanni, and P. Gardin. Extension of the method of moments for population balances involving fractional moments and application to a typical agglomeration problem. *Journal of Colloid and Interface Science*, 276:106–112, 2004. 200, 201

[274] C.H. Lubich. Discretized fractiorial calculus. *SIAM J. Math. Anal.*, 17:704–719, 1986. 201

[275] J. Behboodian. On the modes of a mixture of two normal distributionson the modes of a mixture of two normal distributions. *Technometrics*, 12:131–139, 1970. 203

[276] M.F. Schilling, A.E. Watkins, and W. Watkins. Is human height bimodal? *The American Statistician*, 56: 223–229, 2002. 203

[277] L.F. Shampine. Vectorized adaptive quadrature in matlab. *Journal of Computational and Applied Mathematics*, 211:131–140, 2008. 204

[278] A. K. Majumdar and H. Gamo. Statistical measurements of irradiance fluctuations of a multipass laser beam propagated through laboratory-simulated atmospheric turbulence. *Appl. Opt.*, 21:2229–2235, 1982. 206

[279] A.K. Majumdar. Uniqueness of statistics derived from moments of irradiance fluctuations in atmospheric optical propagation. *Optics communications*, 50:1–7, 1984. 206

[280] A.A. Oncul, D. Thevenin, M.P. Elsner, and A. Seidel-Morgenstern. Numerical analysis of the preferential crystallization of enantiomers. In *11th Workshop on Two-Phase Flow Predictions*, 2005. 208, 209, 210

[281] A. Aliseda, A. Cartellier, F. Hainaux, and J.C. Lasheras. Effect of preferential concentration on the settling velocity of heavy particles in homogeneous isotropic turbulence. *J. Fluid Mech*, 468:77–105, 2002. 210

[282] S. Sundaram and L.R. Collins. Collision statistics in an isotropic particle-laden turbulent suspension part 1: Direct numerical simulations. *J. Fluid Mech*, 335:75–109, 1997.

[283] K.D. Squires and J.K Eaton. Preferential concentration of particles by turbulence. *Phys. Fluids*, 3:1169–1178, 1990.

[284] J.K. Eaton and J.R Fessler. Preferential concentration of particles by turbulence. *Int. J. Multiphase Flow*, 20:169–209, 1994. 210

[285] T. Granlund. *The GNU Multiple Precision Arithmetic Library*, 5.0.2 edition, 5 2011. 211

[286] W. Kahan. *Gauss-Seidel Methods of Solving Large Systems of Linear Equations*. PhD thesis, University of Toronto, 1958. 224

[287] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition edition, 2003. 225

[288] H. L. Stone. Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM Journal of Numerical Analysis*, 5:530–558, 1968. 226

[289] G. E. Schneider and M. Zedan. A three-dimensional modified strongly implicit procedure for heat conduction. *AIAA Journal*, 21:295–303, 1983. 226

[290] G. E. Schneider and M. Zedan. A modified strongly implicit procedure for the numerical solution of field problems. *Numerical Heat Transfer*, 4:1–19, 1981. 226

[291] Y. Saad and M. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing*, 7:856–869, 1986. 227

[292] H. A. van der Vorst. Iterative solution methods for certain sparse linear systems with a nonsymmetric matrix arising from pde problems. *Journal of Computational Physics*, 44:1–19, 1981. 228

[293] A. Brandt. Multi-level adaptive solutions to boundary value problems. *Mathematics of Computation*, 31: 333–390, 1977. 228

[294] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, 2000.

[295] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2001. 231

[296] P. Wesseling. *An Introduction to Multigrid Methods*. Wiley, 1992. 228

[297] M. Caputo. Linear models of dissipation whose q is almost frequency independent-ii. *Geophys. J . R. Astr. Soc*, 13:529–539, 1967. 242

[298] N. Cressie and M. Borkent. The moment generating function has its moments. *Journal of Statistical Planning and Inference*, 13:337–344, 1986. 242