

Imperial College London
Department of Mechanical Engineering

Fundamental Understanding of Turbulent Gas-Solid Flows

Marian Zastawny

April 2013

Supervised by Dr Berend van Wachem

Submitted in part fulfillment of the requirements for the degree of
Doctor of Philosophy in Mechanical Engineering of Imperial College London
and the Diploma of Imperial College London

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution-Non Commercial-No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or distribution, researchers must make clear to others the licence terms of this work.

Declaration of originality

I hereby declare that this dissertation is the result of my own work. anything else included that is not a product of my own work is appropriately referenced in the text.

Abstract

Gas-solid flows are abundant both in nature and in industrial applications, therefore the ability to accurately predict their behaviour is of crucial importance. The main goal of the research project presented in this thesis was to develop a methodology for efficient true direct numerical simulations (DNS) of turbulent flows with solid particles. True DNS (TDNS) in this case implies that not only all the spatial and temporal scales of the flow field are directly computed, but also that the appropriate boundary conditions are imposed on surfaces of the particles allowing for the boundary layer development.

The designed approach is strongly based on the ideas of the Immersed Boundary Method (IBM). In this technique, two separate computational grids are used: a fixed fluid grid and a moving triangulated one, representing the body surface. The flow equations are modified in the regions where the grids overlap. Various implementations of the IBM are discussed, along with the most common difficulties encountered while using this approach. These challenges include accurate imposition of the boundary conditions, evaluation of the fluid-particle momentum transfer and spurious pressure oscillations observed in the case of moving bodies. A number of improvements, designed for addressing the main IBM challenges, are proposed and evaluated on a set of test cases. Additionally, a parallel triangulation library, MFTL, designed in the course of the research project is presented.

The IBM technique is subsequently adopted for the investigation of flows past non-spherical particles at a range of Reynolds numbers and orientations. The results of this study lead to the development of shape-specific correlations evaluating the drag, lift and torques on non-spherical particles as functions of Reynolds number and the angle of incidence. Also, an approach for describing the motion of such particles is presented as well.

Acknowledgements

The successful completion of this doctoral thesis would not have been possible without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

I take this opportunity to express my utmost gratitude to my supervisor, Dr Berend van Wachem, who has been my inspiration as I hurdle all the obstacles in the completion of my research, and who offered invaluable assistance and useful suggestions throughout my doctoral studies. I cannot find words to show enough appreciation for his guidance and steadfast encouragement. Without his help and unfailing support, this project would not have materialized.

It is with immense gratitude that I also acknowledge the Engineering and Physical Sciences Research Council (EPSRC) for their sponsorship and the Jan Dzienisiewicz trust for providing me with a bursary. Had it not been for their financial support, this thesis would have remained a dream.

I would also like to thank my colleagues at Imperial College who shared with me the PhD student experience. I consider it an honour to work in such a stimulating academic and social environment, and I am indebted to my fellow postgraduate students for all the inspirational discussions, for sharing knowledge and ideas, and most importantly for their invaluable friendship.

I also am most grateful to my beloved family and friends for their continued personal support, endless patience and faith in me. They have given me their unequivocal support throughout, as always, for which my mere expression of thanks likewise does not suffice.

Last, but by no means least, I would like to thank Ewa, for being the best support I could dream of.

To my Dad...

Contents

1	Introduction	15
1.1	Multiphase flows	15
1.2	Modelling of gas-solid flows	17
1.2.1	Particle-fluid interaction	17
1.2.2	Continuous phase modelling	21
1.3	Techniques for TDNS of flows with particles	23
1.3.1	Body-fitted grids - Arbitrary Lagrangian-Eulerian methods	23
1.3.2	Fictitious Domain methods	23
1.3.3	Lattice-Boltzmann methods	23
1.3.4	Immersed Boundary methods	24
1.4	Thesis outline	25
2	Immersed Boundary Method	26
2.1	Continuous forcing approach	29
2.2	Discrete forcing approach	31
2.2.1	Discrete direct forcing	32
2.2.2	Ghost cell methods	38
2.2.3	Cut-cell methods	41
2.3	Applying Immersed Boundary Method to the moving interfaces	47
2.3.1	Fresh-cell treatment	48
2.3.2	Spurious pressure oscillations	52
2.4	Additional remarks	60
2.5	Summary	61
3	Numerical method	63
3.1	Single phase flow	63
3.1.1	Momentum equation discretisation	63
3.1.2	Continuity equation discretisation	67
3.2	Immersed Boundary Method implementation	70
3.2.1	The simulation procedure	71
3.2.2	Enforcing the velocity boundary conditions	73
3.2.3	Additional modifications to the flow equations	75

3.2.4	Calculation of the forces on the IB	79
4	MultiFlow Triangulated Library	85
4.1	Concept and data structure	86
4.1.1	Data structure	86
4.1.2	Tolerance for floating point comparisons	88
4.1.3	Parallel implementation of MFTL	90
4.2	Main functions	92
4.2.1	Bounding box tree generation	92
4.2.2	Point-triangle distance	96
4.2.3	Point inclusion test	98
4.2.4	Intersection of two surfaces	99
4.2.5	Convex hull operations	103
4.3	Integration with MULTIFLOW	106
4.3.1	Parallel point inclusion test and cell tagging	106
4.3.2	Calculation of cut-cell properties	107
5	Validation and performance study	110
5.1	Force calculation technique	111
5.1.1	Simulation set-up	112
5.1.2	Results and discussion	114
5.2	Flow past a stationary sphere	118
5.2.1	Simulation set-up	119
5.2.2	Results and discussion	121
5.3	Oscillating sphere - spurious pressure oscillations	128
5.3.1	Simulation set-up	129
5.3.2	Results and discussion	130
5.4	Instantaneously accelerated sphere	135
5.4.1	Simulation set-up	137
5.4.2	Results and discussion	138
5.5	Sedimenting sphere	142
5.5.1	Simulation set-up	143
5.5.2	Results and discussion	144
5.6	Summary	148
6	TDNS with IB application - modelling flows with non-spherical particles	150
6.1	Drag, lift, torque for non-spherical particles	150
6.1.1	Introduction	150
6.1.2	Forces on particles	152
6.1.3	Numerical framework	156

Contents

6.1.4	Simulation set-up	159
6.1.5	Results and discussion	160
6.1.6	Conclusions	171
6.2	Modelling the motion of axis-symmetric particles	173
6.2.1	Equations of motion	174
6.2.2	Forces on the particles	174
7	Summary and outlook	178
7.1	Summary	178
7.2	Outlook	179
7.2.1	Improvements to the computational method	179
7.2.2	Additional capabilities	180
7.2.3	Possible applications	181
	Bibliography	182

List of Tables

3.1	Options for the convective terms discretisation	66
5.1	Simulation parameters used for force calculation evaluation	113
5.2	Sphere refinement level study	113
5.3	Predicted drag coefficients	121
5.4	Time-steps for simulations of an oscillating sphere	130
5.5	Time-steps for simulations of a suddenly accelerated sphere	137
5.6	Fluid properties used in simulations of a sedimenting sphere	143
6.1	Non-spherical particle shapes considered	152
6.2	Fit parameters for the drag of non-spherical particles	164
6.3	Fit parameters for the lift of non-spherical particles	168
6.4	Fit parameters for the pitching torque of non-spherical particles	168
6.5	Fit parameters for the mode 1 rotational torque	171
6.6	Fit parameters for the mode 2 rotational torque	172

List of Figures

1.1	Examples of multiphase flows	16
1.2	Modelling regimes of multiphase flows	18
1.3	Grid used for TDNS	19
1.4	Grid used for point particle model	20
1.5	Euler-Euler simulation of a fluidized bed	21
2.1	Schematic flow illustration	26
2.2	Body-fitted mesh for multiple particles	27
2.3	Immersed Boundary Method concept	28
2.4	Immersed Boundary Method classification	28
2.5	Illustration of continuous forcing	30
2.6	Distribution functions	30
2.7	The forcing procedure of Mohd-Yusof	32
2.8	Strategies for the interpolation of the forcing term	33
2.9	Scenarios for interpolation to the forcing point <i>ib</i>	34
2.10	Stencil for a 3D reconstruction of flow	34
2.11	Reconstruction points for the IB	35
2.12	Points selection applied by Choi et al.	36
2.13	Interpolation stencil used by Choi et al.	36
2.14	Least squares stencil for a 2D case	37
2.15	Mass source/sink term	38
2.16	Flow interpolation for the ghost cell method	38
2.17	Avoiding numerical instability at the ghost nodes	40
2.18	Imaginary points at constant distance from the IB	41
2.19	Trapezoidal cells for the cut-cell method	42
2.20	Calculation of gradient at the IB surface	43
2.21	Extrapolation to the IB surface	44
2.22	Cell linking idea	45
2.23	Cell mixing	46
2.24	<i>Ghost cell for cut-cell</i> formulation	47
2.25	Fresh/Dead cells	48
2.26	Fresh cell merging	49

List of Figures

2.27	Flow reconstruction at <i>fresh</i> cells	50
2.28	Field extension technique	51
2.29	Backward time integration	51
2.30	Pressure distribution for the stationary cylinder	53
2.31	Oscillating square cylinder cell transition	53
2.32	Time trace of flow variables for fresh cell	54
2.33	Time trace of flow variables for dead cell	54
2.34	IB on a Cartesian grid	55
2.35	Virtual cell merging	57
2.36	Effect of forcing strategies on the pressure oscillations	58
2.37	Amplitude spectra for forcing strategies	59
3.1	2D fluid cell with east neighbour	65
3.2	IB simulation procedure	72
3.3	IB cell tagging	73
3.4	Imaginary point IP used for the boundary condition enforcement	74
3.5	Excluding the flow inside the body from the continuity equation	77
3.6	Illustration of the cut-cell approach	78
3.7	Illustration of the imaginary surface for the force calculation	80
3.8	Auxiliary points for the force calculation	82
3.9	Velcotiy gradient reconstruction at the surface	83
4.1	MFTL surfaces	85
4.2	MFTL object structure	87
4.3	Orthogonal bounding box tree	89
4.4	Bounding box	89
4.5	Parallel simulation with multiple particles	91
4.6	Distribution of the surface over multiple processors	91
4.7	Flowchart showing the procedure for bounding box tree generation	94
4.8	Dividing the bounding box.	95
4.9	Reordering of triangle id's in the array	95
4.10	The point and triangle in $x y z$ coordinate system	96
4.11	Zones around a triangle	97
4.12	Ray tracing method	98
4.13	Point in triangle test	99
4.14	The fast overlap test illustration	100
4.15	Two intersecting triangles	101
4.16	Two triangles intersection variants	103
4.17	Points on XY plane	104
4.18	Points on XY plane after sorting	105

List of Figures

4.19 A convex hull	105
4.20 Point inclusion test	107
4.21 Cut-cell in 2D	108
5.1 Flow past a rotating sphere	112
5.2 Force coefficients for rotating sphere	114
5.3 Pressure force coefficients for rotating sphere	114
5.4 Viscous force coefficients for rotating sphere	115
5.5 Force coefficients convergence rate	116
5.6 Pressure force coefficients convergence rate	117
5.7 Viscous force coefficients convergence rate	117
5.8 Surface refinement influence	118
5.9 Force coefficient obtained by direct extrapolation	119
5.10 Flow past a stationary sphere	120
5.11 Time development of the drag coefficient	122
5.12 Drag coefficients for flow past a stationary sphere ($Re = 1.5$)	123
5.13 Drag coefficients for flow past a stationary sphere ($Re = 31.9$)	123
5.14 U Velocity profiles along the x axis ($Re = 31.9, N/D = 10$)	124
5.15 U Velocity profiles along the y axis ($Re = 31.9, N/D = 10$)	124
5.16 Pressure profiles along the x axis ($Re = 31.9, N/D = 10$)	125
5.17 Pressure profiles along the y axis ($Re = 31.9, N/D = 10$)	126
5.18 Imaginary point position influence on the drag ($Re = 1.5$)	127
5.19 Imaginary point position influence on the drag ($Re = 31.9$)	127
5.20 Contours for different IP positions	128
5.21 Oscillation sphere simulation set-up	129
5.22 Pressure forces on oscillating sphere	131
5.23 Detailed analysis of the oscillating sphere	132
5.24 Amplitude spectrum of the C_{DP}	133
5.25 Comparison of the RMS of C_P^2 for different methods	133
5.26 Comparison of the RMS of the high frequency C_{DP} oscillations	134
5.27 Pressure forces with reduced oscillations	135
5.28 Detailed study of low oscillations case	136
5.29 Instantaneously accelerated sphere set-up	137
5.30 Drag coefficient experienced by a instantaneously accelerated sphere at $t = 0.125$ ms	138
5.31 [Drag coefficient experienced by a instantaneously accelerated sphere at $t = 5$ ms	139
5.32 RMS of the C_D^2 for different IB implementations	140
5.33 C_D^2 discontinuity for the basic IB	140
5.34 C_D^2 discontinuity for the zero pressure gradient technique	141
5.35 C_D^2 discontinuity for ghost cell exclusion technique	141

List of Figures

5.36 C_D^2 discontinuity for the cut-cell technique	142
5.37 Sedimenting sphere set-up	143
5.38 Velocity profiles of the sedimenting sphere	144
5.39 Particle trajectories	145
5.40 Velocity profiles of the sedimenting sphere	146
5.41 Forces acting on a settling sphere	146
5.42 Forces acting on a settling sphere	147
5.43 Influence of the grid refinement on the velocity profiles	147
6.1 System of coordinates for non-spherical particles	153
6.2 Mirroring principle used for non-spherical particles	158
6.3 Pressure extrapolation to the surface	159
6.4 Velocity gradient reconstruction at the surface	160
6.5 Computational setup for DNS of non-spherical particles	161
6.6 Flow past ellipsoid at $Re = 200$	161
6.7 Comparison of the results with theoretical prediction	162
6.8 Forces in time for low Re flow	162
6.9 Forces in time for high Re flow	163
6.10 Drag Coefficients of non-spherical particles	165
6.11 Angular dependence of drag on non-spherical particles	166
6.12 Angular dependence of the lift coefficient	167
6.13 Angular dependence of the pitching torque	169
6.14 Mode 1 rotational torque coefficients	170
6.15 Steady state mode 2 rotational torque coefficients	171
6.16 Mode 2 rotational torque coefficients at <i>onset</i> of rotation	172
6.17 The inertial and particle coordinate systems	173
6.18 The velocity-based coordinate system	176

1 Introduction

This thesis summarises work performed on the *Fundamental understanding of turbulent gas-solid flows* research project. The main goal of the project was to develop a methodology for efficient true Direct Numerical Simulations (TDNS) of flows with particles. The designed approach was subsequently adopted for investigation of various multiphase flow cases in order to establish physical models governing the studied flows.

Flows with particles are widely encountered in nature as well as in various industrial applications. The ability to accurately predict behaviour of such flows is therefore of a high importance. Detailed numerical analysis of particulate flows is currently limited to simple cases, due to the high computational cost of such simulations. A number of simplified modelling approaches has been developed, allowing for simulations of flows on a larger, more applied scales. Still, these techniques require closure models based on physical principles in order to be functional.

True direct numerical simulations of particulate flows can be applied to develop engineering models which are necessary for large scale simulations using more sophisticated modelling techniques. For example, the force coefficient correlations, established for non-spherical particles during the course of this research, enabled an accurate analysis of a channel flow with a large number of particles treated as point sources. As a result of the development of the detailed models, simulations on practical scales can be performed and applied, for example, to optimize industrial devices.

1.1 Multiphase flows

Multiphase flows are abundant both in nature and in many industrial applications. In general, the term *multiphase flows* refers to flows, the components of which have more than one phase. Additionally, some authors, *e.g.* [6], add a condition that the components should be distinguishable at scales well above the molecular level. Additionally, Clift *et al.* [15] specify that biologically active and self propelling bodies, for instance a rocket flying in the atmosphere, should be excluded from consideration.

Even if the conditions outlined above are taken into account, multiphase flows still encompass a vast range of flow cases with numerous applications, *e.g.* cloud formation, boiling, cavitation, fuel injection, particle separation in filters, and many others. In fact, multiphase flows are encountered much more often than their single phase counterparts. One of the common features of all multiphase flows is that they contain a continuous phase (gas or a liquid) and separated particles (solid particles, bubbles or drops). Crowe *et al.* [17] distinguish four groups of multiphase flows depending on their components:

Gas-liquid flows - *e.g.* spray formation



(a) Spray formation, a gas-liquid flow.



(b) Dust storm, a gas-solid flow.

Figure 1.1: Examples of multiphase flows¹.

Gas-solid flows - *e.g.* pneumatic conveying

Liquid-solid flows - *e.g.* sediment transport in rivers

Three-phase flows - *e.g.* bubbles in a slurry flow

The physical behaviour of multiphase flows is highly complicated. It usually involves coupling between particles and fluid along with various additional phenomena depending on the investigated case. These involve, but are not limited to, bubble formation, drop coalescence, particle-particle collisions, combustion etc. At present, no general model to characterise behaviour of multiphase flows is available, therefore they have to be treated on one-to-one basis.

The focus of this particular project is on the gas-solid flows, where solid particles are immersed in the gaseous phase. While this is one of the simplest cases of multiphase flows, which does not have to take into account such complex processes as chemical reactions, large deformation of particles, or phase changes, a physical description of the gas-solid flows is not straightforward. Coupling between the fluid and the solid phase, along with particle-particle and particle-wall interactions (collisions/agglomeration), need to be considered. Moreover, the turbulence in the fluid phase introduces an additional level of complexity since the bodies immersed in the flow can either dampen or increase the fluctuations in the fluid.

An ability to understand the gas-solid flows and to predict their behaviour is vital, as they occur in various applications in fields ranging from geophysics, biotechnology, bioengineering, chemical processing, to numerous applications in mechanical engineering. Cyclone separators and fluidised beds are among the most common devices where gas-solid flows are present.

¹ a - <http://en.wikipedia.org/wiki/File:Aerosol.png>

b - <http://en.wikipedia.org/wiki/File:Dust-storm-Texas-1935.png>

1.2 Modelling of gas-solid flows

As mentioned above, a comprehensive physical model for studying multiphase flows has not been developed yet, due to the inherent complexity of such flows. It is also unlikely that such model will be introduced in a foreseeable future. The same is true about the gas-solid flows. Since there is no general physical description available, CFD is often employed in order to predict the flow behaviour. Still, the underlying physical phenomena are highly complicated and the computational power of modern CPU's is limited, therefore no single CFD technique is applicable for analysis of all important flow cases. There is, however, a number of numerical strategies which allow to study both particle-fluid interactions and to model the continuous phase. Some of the most important modelling approaches are described below.

1.2.1 Particle-fluid interaction

Flows with particles can be classified into different interaction regimes, depending on the volume fraction, defined as the fraction of volume occupied by particles in a unit volume [105]:

$$P = \frac{\sum_i N_i V_{P_i}}{V} \quad (1.1)$$

where N_i is the total number of particles in the size fraction i . The particle volume V_{P_i} can be calculated as $V_{P_i} = \frac{\pi}{6} D_{P_i}^3$. D_{P_i} is the diameter of a volume equivalent sphere, *i.e.* a sphere having the same volume as the investigated particle.

The volume fraction is related to the inter-particle spacing, L , by the formula:

$$\frac{L}{D_P} = \frac{1}{6} \frac{1}{P} \quad (1.2)$$

Simple analysis of the above equation shows that even for a relatively low volume fraction, for instance $P = 1\%$, the particles are only 3.74 diameters apart. Since the particles are close to each other, it can be expected, that the particle-particle interactions are non-negligible and have to be taken into account. For very high volume fractions, the interaction between the particles might even dominate the flow behaviour. Still, in many engineering applications, the volume fraction has much smaller values, ranging from 10^{-5} to 10^{-4} . In such cases, the resulting particle spacing is bigger than 20 diameters, hence the particles can be treated as isolated bodies.

A classification of flow regimes, depending on the particle volume fraction has been proposed by Elghobashi [20]. As shown in Fig. 1.2, the volume fraction along with the inter-particle spacing can be used to divide the flows into dense and dispersed two-phase flows. When $P < 10^{-3}$ the flow can be treated as dilute, hence the particle-particle interactions are ignored. A distinction of two additional modelling regimes can be made for the dilute flows. If the volume fraction is very low, $P < 10^{-6}$, the influence of particles on the fluid behaviour is negligible, therefore one-way coupling approach is applied, *i.e.* only the effect of the fluid on the particles is accounted for. On the other hand, for higher values of P , the particle influence can no longer be neglected, and it has to be considered in the flow

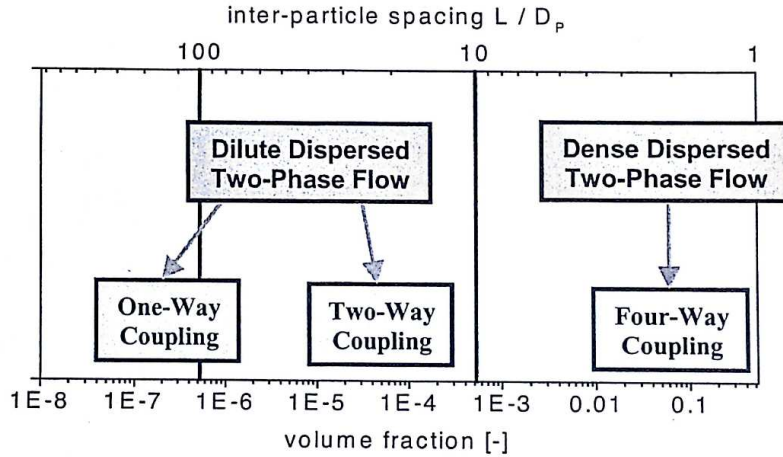


Figure 1.2: Modelling regimes of multiphase flows as the function of the volume fraction and the inter-particle spacing [105]. More complicated techniques have to be adopted as the volume fraction is increased.

equations.

On the opposite end of spectrum, *i.e.* $P > 10^3$ there is the dense flow regime. Here, the particle spacing is smaller than 10, which implies that particle collisions occur more frequently. Moreover, closely spaced particles modify the local flow field, indirectly affecting other particles. This regime is often referred to as the flow with a four-way coupling.

As discussed above, different physical phenomena need to be accounted for depending on flow regime, hence a special strategy has to be applied for each investigated case. There is a number of approaches designed for analysing each of the flow regimes. A comprehensive review of the available modelling techniques can be found in [114]. A brief description of the popular strategies used for numerical analysis of multiphase flows is given below.

Resolved particle interface (True DNS)

The most accurate technique which allows to simulate multiphase flows is to compute the exact flow around each particle. This requires resolving each particle by a number of fluid cells (usually a number of cells per diameter is of order $O(10)$) and imposing the appropriate boundary conditions at their surfaces, as illustrated in Fig. 1.3. This approach is referred to as a “True” Direct Numerical Simulation (TDNS), to distinguish from the traditional DNS which can also be used in the point particle modelling technique.

Due to high computational cost of such simulations, CFD analysis which use TDNS are limited to simple cases. For example, a simulation of a flow with 1000 spherical particles, each discretised by 10 cells per diameter, requires 10^6 cells only to compute the flow around the particles. Therefore, this approach is of little use for practical applications where the number of particles is usually greater than 10^5 .

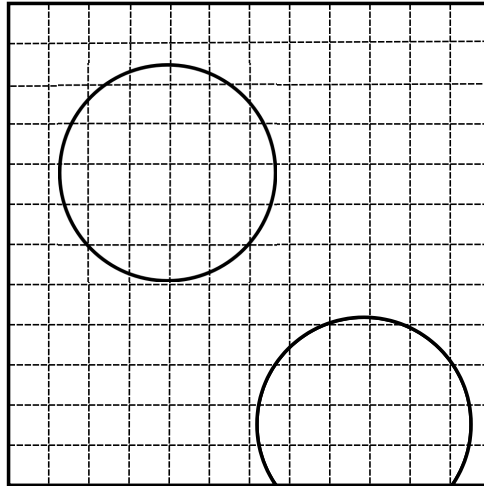


Figure 1.3: An example of grid used for TDNS. The grid spacing is smaller than the particle dimensions.

Even though the computation of the exact flow past particles is not feasible in large scale simulations, it is still a very useful approach. Results obtained this way help to better understand the physical processes governing the particle-fluid interaction and enable to design closure models required for simulations at larger scales.

Since the main motivation of the research project described in this thesis is to develop a deeper understanding of the gas-solid flow behaviour, TDNS has been chosen as the most accurate approach to resolve gas-solid flows. Several techniques for efficient simulations of flows with particles are either already available or under intense development. Significant part of this particular project was devoted to designing a robust methodology for performing simulations with fully resolved particles. The effects of this study are presented in chapters 3 and 5.

Point particle model

Resolving the accurate flow past each particle is a sub-category of a modelling approach called the Euler-Lagrange technique. In the Euler-Lagrange framework each particle is treated individually, what requires computation of both particle-particle and particle-fluid interaction for every body immersed in the flow. Moreover, the motion of every particle has to be calculated separately.

A point particle model is a “less expensive” alternative to TDNS. In this approach, the particles are treated as point sources, which are smaller than the grid spacing as shown in Fig. 1.4. Each particle can be identified and traced throughout the simulation. The forces on particles can be evaluated with the use of the force coefficient correlations, depending on the local flow variables. The two-way coupling is introduced by adding local source terms (equal to the force acting on each particle) and volume fraction influence to the Navier-Stokes equations of the fluid.

Applying the point particle model requires significantly less computational cost than the exact resolution of the boundary layer around each particle. Therefore, it enables simulations of multiphase flows on much larger scales with a high level of accuracy. Still, the precision of the obtained results will strongly

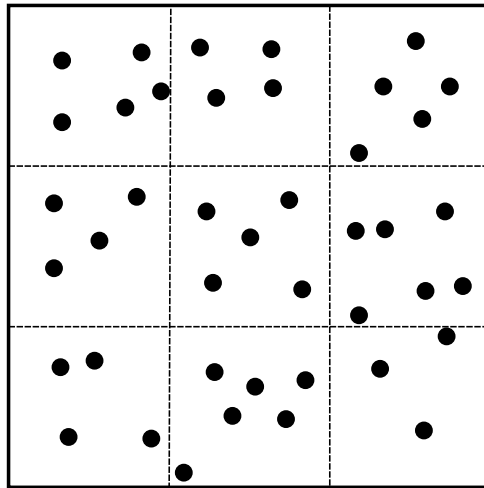


Figure 1.4: An example of grid used for point particle model, several particles can be present in a single fluid cell.

depend on the quality of models used for computing the particle-fluid interaction.

Euler-Euler modelling

At present, the only technique capable of simulating flows with a very high number of particles is the Eulerian-Eulerian modelling also known as the two-fluid method. Unlike in the approaches presented above, here the particles are not treated individually but are modelled as a quasi-fluid inter-penetrating with the flow. Hence, two sets of continuity and Navier-Stokes equations, linked by the volume fraction, have to be solved: one for the fluid and one for the averaged particles.

The Eulerian-Eulerian approach is a highly efficient technique, independent of the number of particles, which can be used in large scale simulations of dense multiphase flows. An illustration of a fluidized bed resolved in a two-fluid framework is shown in Fig. 1.5. Each phase can be distinguished by its volume fraction; the red colour corresponds to the regions of high particle accumulation, while the blue colour represents the volume occupied entirely by the fluid.

The challenges related to the closure models observed in the Euler-Euler technique are more severe than in the point-particle method. Not only the interaction of particles with both the fluid and each other has to be resolved but also the particle-turbulence coupling has to be evaluated. This framework is only applicable for dense flows, since meaningful averages have to be found, *i.e.* the mean values have to be calculated on scales much smaller than the flow domain. Also, the averaging process constrains the two-fluid approach to simulations with RANS turbulence modelling, as the exact flow details required for DNS or LES cannot be resolved. This is due to the fact that in the Euler-Euler approach the detailed interaction between the particles and the flow is averaged out and the small flow structures are not computed directly.

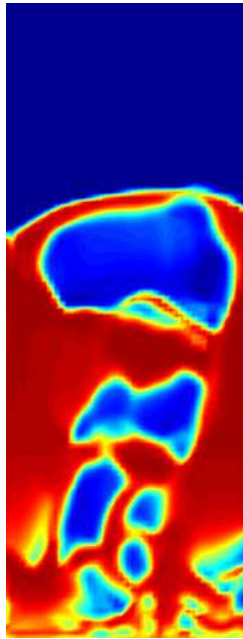


Figure 1.5: Euler-Euler simulation of a fluidized bed. Red colour represents the region of high particle concentration, while blue corresponds to the volume occupied entirely by the fluid.

1.2.2 Continuous phase modelling

In practice, most of the flows encountered in real-world applications are inherently turbulent, what poses significant challenges to the flow modelling. By definition, the turbulent flows are unsteady, of a chaotic character and three-dimensional. They involve strong fluid mixing along with fluctuations on a broad range of time and length scales. Due to the properties listed above, simulations of turbulent flows require special treatment. Several methodologies have been developed for this purpose.

Direct Numerical Simulation

Direct Numerical Simulation (DNS) is the most accurate way to simulate turbulent flows. It requires resolving all the fine details of the flow, both in space and in time. Because turbulence involves velocity and pressure fluctuations on a range of scales, the computational grid has to be fine enough to resolve even the smallest oscillations and capture the energy dissipating eddies. On the other hand, the domain size has to be large enough to encompass the large scale structures of the flow. The need to satisfy both conditions simultaneously makes DNS a very expensive computational method.

The ratio of the domain size to the grid resolution increases proportionally to the Reynolds number as $L \propto Re_L^{3/4}$ [24], where Re_L is the Reynolds number depending on the size of the largest eddies, L and λ are the dimensions of the large and small scale eddies respectively. Thus, as the Reynolds number becomes larger, the domain size and the grid refinement have to increase as well. This limits DNS to flows with low Re , what means that most cases of engineering applicability are not solvable with this approach.

Although DNS is limited to analyses of only simple problems at low Reynolds numbers, it is still a very powerful scientific tool, which allows to understand the physics governing the observed processes. DNS can be used to quantify various flow-particle interaction properties, along with the mechanisms of turbulence production and dissipation. The low Reynolds number limit is not such a severe constraint for particulate flows as the Re depends on the particle dimensions, which are usually small, hence in most cases $Re < 10^3$. Due to its ability to resolve the fine details of the flow, DNS is the underlying framework in this particular research project.

Large Eddy Simulation

As discussed above, Direct Numerical Simulation is a very expensive approach which generally produces much more data than is needed, therefore it has a limited applicability for analysis of practical flow cases at high Reynolds numbers. Large Eddy Simulation (LES) is a more feasible technique when general behaviour of the flow is to be determined.

In LES methodology only large scale, more energetic, vortices are resolved directly, while the small eddies are filtered out. Energy dissipation occurring in the small scale eddies is approximated by sub-grid scale models such as the Smagorinsky dissipation model. In general, LES is less expensive than DNS, however it is less accurate. Still, LES allows to perform relatively detailed simulations for complex geometries at Re too high to be resolved by means of Direct Numerical Simulations.

Reynolds Averaged Navier-Stokes

In engineering practice, only the global flow variables need to be computed, whilst the fine flow details are required only occasionally. Thus, applying expensive methods such as DNS or LES is usually not feasible. Reynolds-Averaged Navier-Stokes (RANS) equations are more often used for general purposes, since they allow for a solution of the flow in an average sense concentrating on the most important flow parameters.

In RANS, the flow variables are decomposed into the averaged and fluctuating components. While the mean values are resolved accurately, the oscillating components are averaged out and require special treatment. Several semi-empirical turbulence models (*e.g.* $k-\epsilon$, $k-\omega$) are available for estimation of the contribution of the flow fluctuations. These models usually introduce additional equations with empirical constants.

Although the number of equations solved by RANS is larger than in the case of LES or DNS, the simulations are still less expensive since a coarser grid and larger time-steps can be applied. The validity of closure models is based on comparison with experiments or LES/DNS simulations, however extra care needs to be taken when new flow cases are under investigation.

1.3 Techniques for TDNS of flows with particles

As mentioned earlier, TDNS of flows with particles can be an invaluable tool for understanding the processes occurring in multiphase flows. Since all the flow features are resolved with a high precision, TDNS is the closest thing to an experiment, but with much more control over the flow parameters and with more comprehensive data output.

In this thesis the concept of True Direct Numerical Simulations has two meanings. Firstly, it describes the technique applied for modelling the particle-fluid interaction. This involves resolving accurate flow around the body including the boundary layer with appropriate boundary conditions.

Secondly, the TDNS term refers to the treatment of the Navier-Stokes equations. As specified in section 1.2.2, performing Direct Numerical Simulations involves computation of the fluid behaviour at all spatial and temporal scales. Therefore no turbulence modelling is required, as the oscillations will be naturally triggered by small numerical instabilities.

Performing TDNS of gas-solid flows requires considerable computational power and an appropriate numerical technique to effectively model the bodies immersed in the fluid. A number of techniques for TDNS with solid particles is available and a brief outline of the most common approaches is given in this section.

1.3.1 Body-fitted grids - Arbitrary Lagrangian-Eulerian methods

This is conceptually the easiest approach. The domain is discretised on a boundary fitted grid and the boundary conditions are applied exactly at the domain boundaries. Although this methodology is extremely precise, the necessity of the grid re-meshing after every time-step significantly increases computational cost of the simulations.

Examples of applications of the body fitted grid approach for particulate flows simulation can be found in [50, 56, 57, 116].

1.3.2 Fictitious Domain methods

The fictitious domain method is a technique where a complex domain is embedded in a larger, simpler domain (fictitious domain). The boundary conditions of the original boundary are enforced in the new domain. A single set of momentum and continuity equations is solved, however a variable density (described by a step function) is used. Additionally, a rigidity constraint is enforced on the fluid, which belongs to the particle domain. This method is applied, for example in [2, 32, 93, 107]. The Immersed Boundary method, discussed later on, can also be treated as a variant of the fictitious domain approach.

1.3.3 Lattice-Boltzmann methods

Lattice-Boltzmann methods (LBM) significantly differ from other techniques described in this section. Instead of solving fluid equations using the finite volume or the finite difference approach, the Lattice Boltzmann method solves the Boltzmann equation on a discrete system of fictitious fluid “particles” (not

to be confused with bodies immersed in the fluid). The fluid “particles” are positioned on a regular lattice. They are allowed to move with discrete steps and to collide with other “particles” according to specific rules. If appropriate lattice topology and collision rules are chosen, the resulting solution resembles the Navier-Stokes equations.

The Lattice-Boltzmann method is very efficient and easily implemented in parallel computations. It also avoids the re-meshing issue encountered by body-fitted grids. The main disadvantage of this approach is the time-step limitation, which is much more strict than in the case of finite volume discretisation. LBM was applied for the DNS of particulate flows in [66, 67] and used in the study of a flow past non-spherical particles in [48].

1.3.4 Immersed Boundary methods

Immersed Boundary methods (IBM) consist of a class of techniques using separate grids for the fluid domain and for description of the particles’ surfaces. The fluid equations are modified in the regions where the two grids overlap. Since its introduction by Peskin in [95], the Immersed Boundary method has been applied by numerous researchers in various fields. Numerous variants of the approach are available.

The Immersed Boundary Method is particularly well suited for TDNS of gas-solid flows as it combines a high degree of accuracy with computational efficiency. Additionally, the IBM has several features that make it particularly attractive technique:

Contrary to the ALE approach, in IBM there is no need for the grid re-meshing associated with particle movement. This allows for higher computational efficiency and limits the loss of accuracy related to required interpolation.

The IB technique is conceptually simple and can be easily implemented on both structured and unstructured grids. It can be applied to any CFD code without the need of modification of the underlying solver.

The method is well suited for parallelization which can additionally improve its performance.

Application of separate meshes for the simulated bodies allows for efficient analysis of flows past complex bodies of an arbitrary shape as shown in chapter 6.

The IBM can also be extended to include the body deformation and the aero- or hydro-elastic effects.

Additionally, the IB approach can be modified in order to take into account various physical phenomena such as particle collisions, sub-grid scale forces, etc.

The aforementioned features of Immersed Boundary Method motivated its choice as the basis of the technique applied in the current research project. As mentioned earlier, there exists a number of IBM implementations, a detailed discussion of which is given in chapter 2. Nevertheless, although the IBM has been successfully applied in various scenarios, the current implementations of the method generally

experience problems related to the lack of mass conservation in the vicinity of the simulated bodies which lead to spurious pressure oscillations when moving bodies are simulated.

The methodology developed during the research project described in this thesis is designed to address those issues. The proposed approach aims at strict imposition of the no-slip boundary condition at the surface of the particles while maintaining global as well as local mass conservation. This allows for enhanced accuracy and a significant reduction of the spurious pressure fluctuations.

1.4 Thesis outline

The main goal of this particular research project is to design a methodology for efficient TDNS of gas-solid flows. The specific technique used for fluid-particle coupling is the Immersed Boundary Method, and its new implementation developed during the course of the project. The resulting technique can be applied for detailed analysis of flows with particles, in order to increase the understanding of the underlying physical processes governing the behaviour of such flows.

The thesis is structured in the following way. First, an overview of existing IBM implementations is presented in chapter 2. The main properties, strengths and weaknesses of the available methods, along with the challenges encountered by the IB approach are discussed. Next, the description of the developed numerical technique and its variants is given in chapter 3. This involves specification of the single phase flow discretisation and the details of the proposed IB implementation. Chapter 4 presents MFTL, the MULTIFLOW Triangulation Library designed to handle triangulated surfaces used in the current IBM.

The validation process of the proposed numerical technique is presented in chapter 5, where the results of several test cases are discussed. Then an example of TDNS application to model the forces acting on non-spherical particles in the flow is demonstrated. The thesis concludes with a summary and a future outlook.

2 Immersed Boundary Method

In general, the solution of an incompressible flow past a number of particles, as illustrated in Fig. 2.1, can be obtained by solving the Navier-Stokes equations given by:

$$\frac{\partial u^j}{\partial t} + \frac{\partial}{\partial x^i} u^i u^j = -\frac{\partial p}{\partial x^j} + \frac{\partial}{\partial x^i} \tau^{ij} \quad \text{in } \Omega_f \quad (2.1)$$

$$\frac{\partial}{\partial x^i} u^i = 0 \quad \text{in } \Omega_f \quad (2.2)$$

with appropriate boundary conditions at the particle surfaces:

$$u^i = u^i_{k} \quad \text{on } \Gamma_k \quad (2.3)$$

The Einstein summation convention is used in the above equations. The first expression represents the momentum balance, while the second one is the continuity conservation principle. Equation 2.3 is applied to specify the no-slip velocity boundary condition for each particle (note that the immersed bodies can have different velocities). The corresponding terms are:

u^j - velocity component in j direction

p - pressure,

ρ - fluid density

τ^{ij} - fluid stress tensor

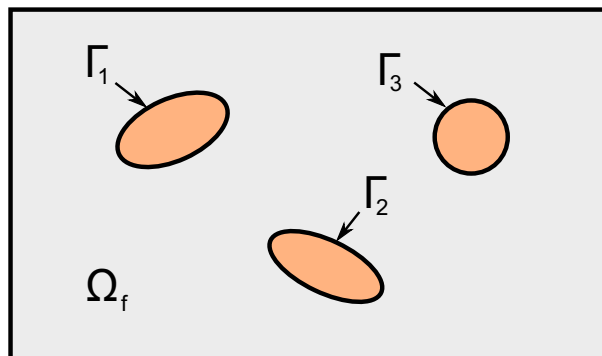


Figure 2.1: A schematic illustration of a flow of fluid Ω_f past multiple bodies Γ_k .

For Newtonian fluids the stress tensor is defined as

$$\tau_{ij} = \mu \left(\frac{\partial u^i}{\partial x^j} + \frac{\partial u^j}{\partial x^i} \right) \quad (2.4)$$

where μ is the dynamic viscosity of the fluid.

As stated in the previous chapter, a conventional approach for CFD simulations is to generate a body-fitted grid in the flow domain, such as the one shown in Fig. 2.2, and to solve the discretised flow equations with appropriate boundary conditions on the boundaries of the mesh. It is a well established technique which allows to obtain very accurate results, even for complex cases. This approach is also referred to as Arbitrary Lagrangian-Eulerian (ALE).

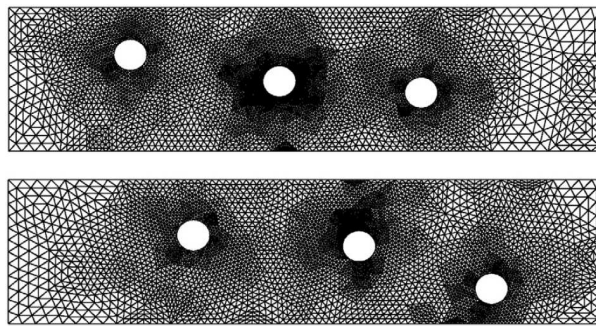


Figure 2.2: Body-fitted mesh for flow with multiple moving particles at two consecutive time-steps [39]. Re-meshing is required every time the particles move.

Although the ALE approach has obvious advantages related to the precision of results, its applicability to particulate flows is limited. The main drawback of this method is the need of grid regeneration every time the particles move. Despite the recent advances in the grid generation field, meshing is still a complicated process, which very often requires a considerable user input. Moreover, re-meshing may not only lead to a significant increase of computational cost of the simulation, but it can also introduce problems with mapping of the old solution to the newly generated grid [86].

The main motivation for designing the Immersed Boundary Method (IBM) was to avoid the aforementioned problems by applying a fixed grid (preferably Cartesian) for the fluid, and separate grids for the simulated bodies. Fig. 2.3 illustrates the IBM concept. The fluid is discretised on a static, boundary non-conformal mesh and the flow equations are modified in the highlighted regions around the particles in order to satisfy the boundary conditions on the bodies' surfaces.

Using a Cartesian grid for the flow simulations has numerous advantages. Firstly, the generation of a Cartesian mesh is very simple and can be easily automated. Additionally, as opposed to body-conformal grids, Cartesian meshes usually require less operations per grid cell and have an advantage of using powerful solving techniques, which enable to achieve the flow solution in shorter time-spans [86]. On the other hand, body-fitted meshes offer a higher level of control over the mesh design, because of the local grid refinement around the particles. Due to their ability to locally enhance the quality of the mesh, they scale better with increasing Reynolds numbers. Still, the capability to perform simulations

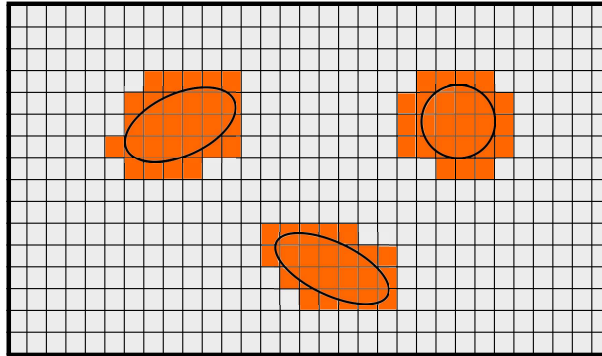


Figure 2.3: Illustration of the main idea behind the Immersed Boundary Method. Only highlighted cells are directly affected by the IBM, while the unmodified Navier-Stokes equations are solved in the remaining fluid cells.

of moving bodies in a time efficient manner, remains the main strength of the techniques adopting body-independent grids, such as the Immersed Boundary Method.

As stated earlier, the main idea of the IBM is to use separate grids for the fluid and for the simulated particles. The flow equations are discretised on the fluid mesh and are modified in the vicinity of the bodies to ensure that the proper boundary conditions are satisfied. A number of implementations of the IBM has been proposed since Peskin [95] introduced the idea of the Immersed Boundary (IB) in 1972. Although the main concept remains the same, the available techniques differ in the way the fluid-body coupling is enforced.

Following the classification suggested by Mittal and Iaccarino [86], the available approaches can be divided into two main groups (Fig. 2.4):

Continuous forcing approach

Discrete forcing approach

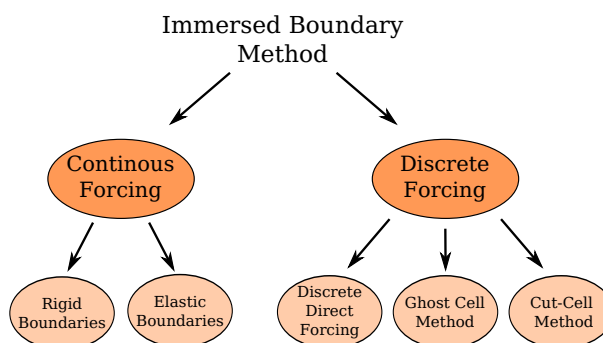


Figure 2.4: Classification of the Immersed Boundary Methods.

In short, the techniques in the first group calculate the force exerted by the particle on the fluid for each body element and spread the influence of the IB over a number of neighbouring fluid cells using

a distribution function. The methods in the second group, on the other hand, aim at calculating the influence of the boundary directly in the fluid cells occupied by the interface.

As stated by Gilmanov and Sotiropoulos [30], regardless of the adopted methodology, the success of the method depends on:

1. A proper description of the moving bodies topology;
2. Establishing appropriate relationships between the bodies and the underlying fluid grid;
3. Accurate enforcement of the boundary conditions;

Comprehensive reviews of the existing approaches can be found in paper by Haeri and Shrimpton [39] as well as in an article by Mittal and Iaccarino [86]. The main properties of each computational approach, along with their various implementations, are presented in the subsequent sections.

2.1 Continuous forcing approach

The original Immersed Boundary Method was developed by Peskin [95] in order to simulate blood flow in a beating human heart, which can be classified as a flow with elastic boundaries. In his technique, Peskin modified the Navier-Stokes equations by addition of a localized forcing term to the momentum equations:

$$\frac{u^j}{t} + \frac{u^i u^j}{x^i} = \frac{p}{x^j} + \frac{ij}{x^i} + f^j \quad (2.5)$$

where the forcing term, f^j , is given by:

$$f^j = \sum_k F_k^j (x^j - X_k^j) \quad (2.6)$$

F_k^j is the j 'th component of stress on the surface element k , while $(x^j - X_k^j)$ is the Dirac delta function which is non-zero when cell centre x^j coincides with surface element X_k^j :

$$(x^j - X_k^j) = \begin{cases} 1 & x^j = X_k^j \\ 0 & x^j \neq X_k^j \end{cases} \quad (2.7)$$

Hence f^j represents the force exerted on the flow by an elastic boundary. Since in practice, the cell centres rarely coincide with the surface elements, a distribution function needs to be applied instead of the delta function. Therefore the influence of the surface on the fluid is distributed over a range of cells, what leads to a smeared representation of the interface (Fig. 2.5).

It can be easily observed that the selection of an appropriate distribution function is the major component of this technique. A number of distribution functions, with different level of accuracy and complexity, was designed throughout the years *e.g.* [36, 68, 96, 99]. Fig. 2.6 shows some of the available

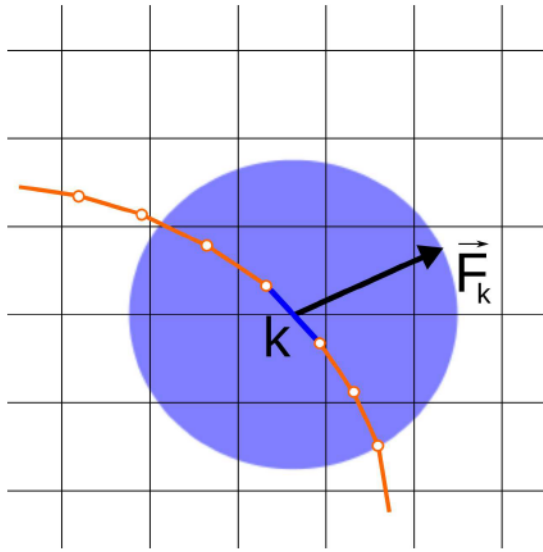


Figure 2.5: Illustration of the range of influence of the surface element k forcing term \vec{F}_k on the fluid grid.

distribution functions and their regions of influence. Haeri and Shrimpton [39] evaluate the performance of various distribution functions in their review, showing that smooth functions generally produce better results. They indicate, however, that a large number of points required for smoothing can significantly increase the computational cost associated with the forcing distribution and lead to a very smoothed representation of the interface.

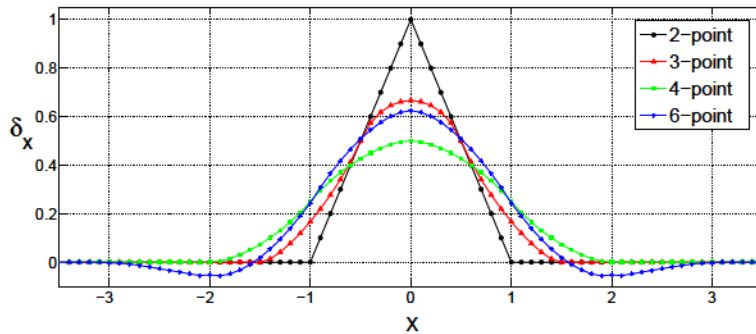


Figure 2.6: Range of influence for various distribution functions: 2-point hat[68], compact 3-point [99], 4-point [96], 6-point [36].

Continuous forcing approach was designed to handle simulations where the boundary is elastic and the stress on surface elements F_k^j can be determined using constitutive relationships, for example the Hooke's law. The technique exhibits, however, some serious drawbacks in the rigid body limit.

Adapting the approach suggested by Peskin requires retaining elasticity of the body, while assuming it is very stiff. This may lead to a stiff system of equations with stability constraints ([68], [106]). An

alternative model to describe the body stress was therefore suggested by Goldstein *et al.* [33]:

$$F_k^j = \int_0^t u^j(\cdot) d\tau + u^j(t) \quad (2.8)$$

where parameters α and β are adjusted to ensure proper boundary condition enforcement (usually both parameters have large negative values). This approach was applied to simulate a start-up flow past a circular cylinder and a low *Re* flow in a channel with stream-wise grooves [33]. The main disadvantage of the technique suggested by Goldstein *et al.* [33], sometimes called Virtual Boundary method is the limiting time-step, which is generally much smaller than the CFL number of the flow. Also, neither α nor β parameters are known before the simulation of a new problem and inappropriate choice of their values leads to stability problems for the flows with high Reynolds numbers [86].

A different approach based on application of porous medium equation was also developed ([1, 59]). In this case, the stress term F_k^j is active only in the solid region, which is equivalent to setting the coefficient to zero and adjusting only β [86].

To sum up, the continuous forcing approach is well suited for the flows with elastic boundaries, however it may suffer from several problems in the rigid limit. Moreover, the application of distribution functions, used instead of the δ function, prevents sharp representation of the boundary and allows for a non-physical mass flux through the surface [82]. Also, since the method requires solving the flow inside of the IB as well, its computational cost will increase proportionally the grid refinement, which can be avoided in some of the other discrete forcing techniques.

2.2 Discrete forcing approach

Discrete forcing methods have been designed in order to cope with the main difficulties observed in the continuous forcing approach. In methods belonging to this group, the forcing is applied directly in the fluid cells, hence no distribution function is necessary, however some interpolation of fluid/body values is usually required.

Three different categories which fall into the discrete forcing approach can be identified, namely *discrete direct forcing*, *ghost cell* method and the *cut-cell* method. The *discrete direct forcing* approach modifies the flow equations in the vicinity of the interface to ensure that the boundary condition is satisfied. In this technique, two methods are at hand. The forcing term, added to the fluid cells in the vicinity of the IB, can be evaluated on the pre-solved flow, as a force required to ensure that the no-slip velocity boundary condition is satisfied at the surface. Alternatively, the fluid velocity can be enforced in the fluid cells by reconstruction of the local flow field with the imposed boundary condition. The *ghost cell* method, on the other hand, is setting the velocities in cells inside of the body (so called *ghost cells*) in a way, that the interpolated velocity at the surface corresponds to the velocity of the body. Finally, the *cut-cell* methods are designed to accurately preserve the conservation laws both in terms of the momentum and the continuity, without the necessity of using cells inside the Immersed Boundary.

2.2.1 Discrete direct forcing

Modified Navier-Stokes equation 2.5 introduced by Peskin [95], applies also in the *discrete direct forcing* approach. In this case, however, generally the forcing term has to be calculated in the fluid cell from the available solution:

$$f^i = \frac{u^i u^j}{x^i} + \frac{p}{x^j} - \frac{u^j}{x^i} + \frac{u_{IB}^j - u^j}{t} \quad (2.9)$$

The forcing term is therefore used to enforce the u^j velocity at the current time-step, so that it is equal to the velocity of the Immersed Boundary. This technique depends heavily on the time advancement scheme adopted to evaluate the forcing term [3]. What is more, the main challenge in the *discrete direct forcing* approach is to calculate the forcing term in a general case, when the grid points do not coincide with the body surface.

Mohd-Yusof [87] was one of the first researchers to apply this technique. The forcing in [87] was calculated from the explicit solution at a previous time step and advanced in time using the Crank-Nicholson scheme for viscous terms and the 3rd-order Runge-Kutta for the non-linear terms. In order to preserve the smoothness of the velocity field, the forcing was applied in the interior cells as shown in Fig. 2.7. Application of the forcing changed the flow direction in the interior cell (b), so that the interpolated velocity satisfied the boundary condition at the interface. Then, after the diffusion step, the flow reached its final state (c). The main advantage of the method is that the velocity gradient on the surface can be reconstructed. On the other hand, the mirroring introduces a non-physical flow inside of the body which may lead to problems with mass conservation.

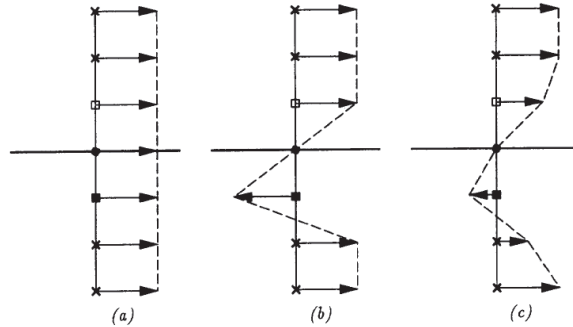


Figure 2.7: The forcing procedure of Mohd-Yusof [87]. The velocity of the fluid inside the IB is set to ensure that a no-slip boundary condition is satisfied at the interface. a) initial flow field, b) flow after addition of the forcing term, c) flow after a diffusion step.

The idea by Mohd-Yusof was further investigated by Fadlun *et al.* [22], who studied various approaches for calculating the body velocity u_{IB}^j in equation 2.9. Predicting the accurate value of u_{IB}^j was found to be critical for the convergence of the method. The adopted techniques are illustrated in Fig. 2.8. In the first approach, the forcing is applied directly in the cell that contains the surface, what leads to a stair-step representation of the IB. On the other hand, if the volume average is used the forcing is scaled by the volume of the fluid in the cut-cell. The most promising technique, however, is the linear

interpolation, where the u_{IB}^j in the equation 2.9 is calculated as a value interpolated from the IB and the neighbouring node. Additionally, Fadlun *et al.* have shown that the treatment applied to the cells inside the body has nearly no effect on the external flow field.

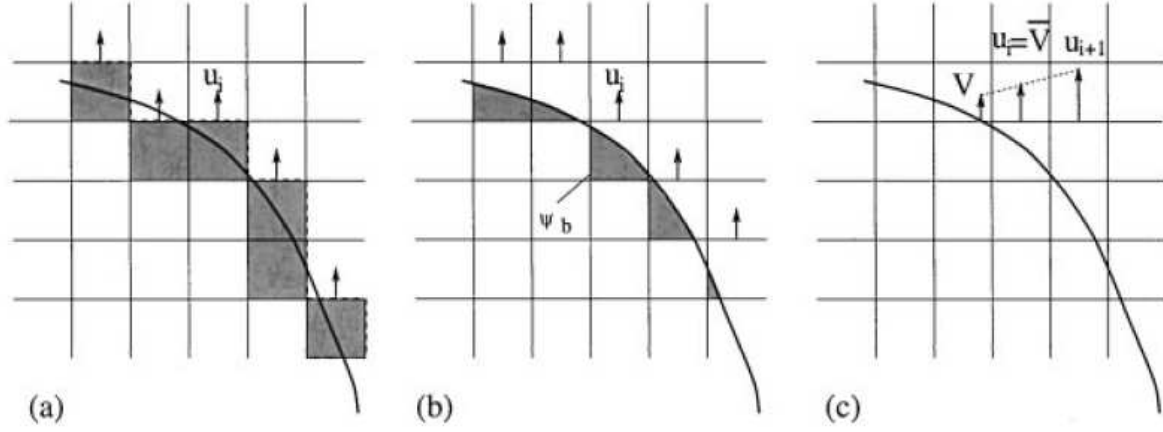


Figure 2.8: Strategies for the interpolation of the forcing term [22]. a) stair step, b) volume average, c) linear interpolation.

Although using the linear interpolation procedure results in a second-order method, the choice of the interpolation direction used by Fadlun *et al.* is somewhat arbitrary. As pointed by Balaras [3], this can lead to problems for complex geometries. In order to avoid this issue, Balaras suggested using the local surface normal, as the interpolation direction. In this method, the forcing was applied to external fluid points (ib in Fig. 2.9), while the value of velocity at those points was obtained through interpolation between the surface point and the “virtual” point (empty circle in the figure) according to:

$$u_{ib}^j = \frac{h_2}{h} u^j + \frac{h_1}{h} u_{virtual}^j \quad (2.10)$$

The velocity values at the “virtual” points were established by a bi-linear interpolation from the neighbouring fluid cells 1-4:

$$u_{virtual}^j = \sum_{i=1}^4 \alpha_i u_i^j \quad (2.11)$$

where α_i is the geometric interpolation coefficient from the cell i .

For the purpose of the calculation of the forcing term, both viscous and diffusive terms were advanced with the Adams-Bashforth scheme. There was no need to impose the pressure boundary condition as it was implicit in the right-hand side of the Poisson equation. Since the velocity components were linearised in vicinity of the IB, the pressure boundary condition reduced to $\frac{p}{n} = 0$ at the particle surface, as discussed in [22].

Gilmanov *et al.* [31] extended the methodology discussed above to simulations of a single, convex Immersed Boundary in a three-dimensional flow. Here, the flow was solved on a non-staggered grid, with a second-order accurate finite-difference approach. The QUICK upwind scheme was applied for the

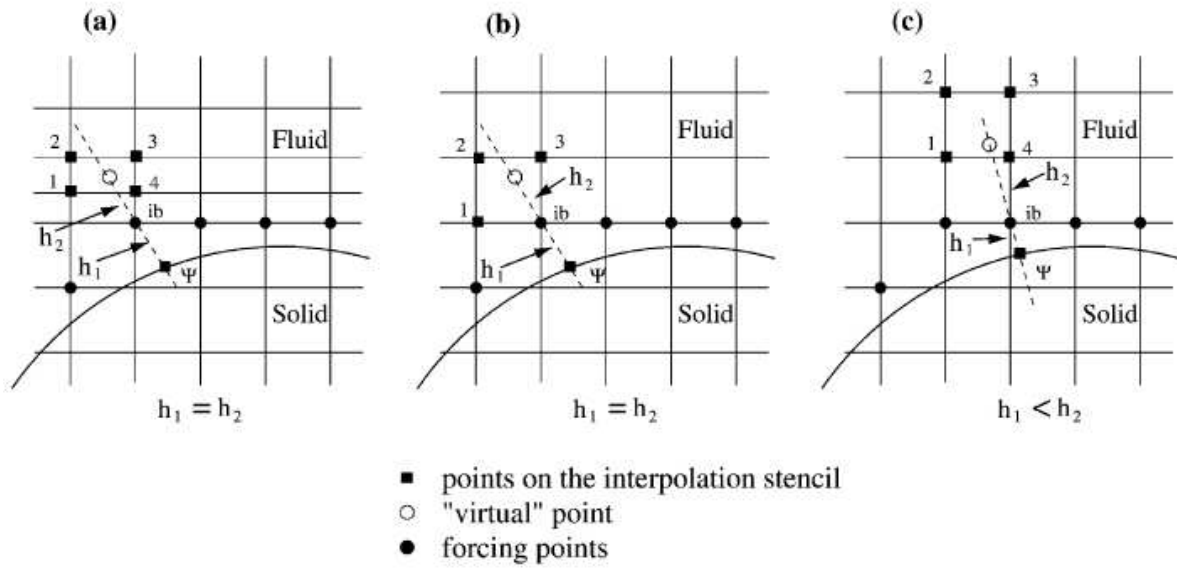


Figure 2.9: Various scenarios for the interpolation of the flow variables to the forcing point ib [3]. Figures a-c show different interpolation stencil, used depending on the “virtual” point position.

convective terms, while time advancement was achieved by a four-stage, explicit Runge-Kutta scheme enhanced with implicit residual smoothing and local (dual) time stepping.

The interpolation stencil is shown in Fig. 2.10. Similarly as before, the forcing term was calculated on the fluid points next to the interface (point b in the figure). In this case however, the virtual point c was found at the intersection of the normal to the IB crossing the forcing point with the nearest plane containing four fluid nodes a , d , e , and f . Also, a Neumann boundary condition was applied on the pressure at the interface.

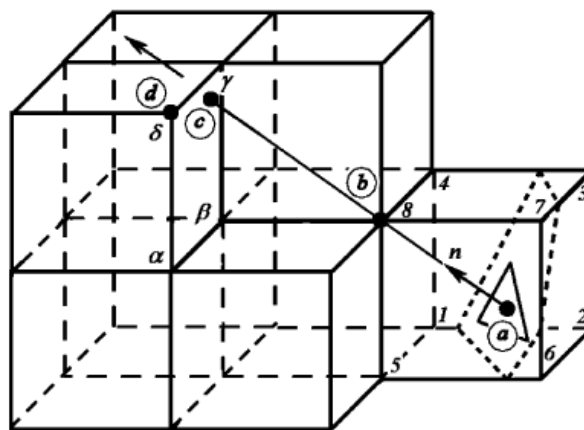


Figure 2.10: Reconstruction of the solution at point b [31]. The flow variables at b are found by interpolation from a , at the IB surface, and c lying on the x, y, z plane.

Application of the aforementioned technique to more complex problems involving multiple moving

bodies of arbitrary shapes was introduced in [30]. The governing equations were discretized using a hybrid staggered/non-staggered grid arrangement. The interpolation stencil remained the same as the one illustrated in Fig. 2.10, however a quadratic interpolation was used instead of the linear one. The structure of points used for the IB method is shown in Figure 2.11. The technique conserves the mass throughout the flow (global conservation), but not in the region of the forcing points. Applications of this technique can be found in [29, 104].

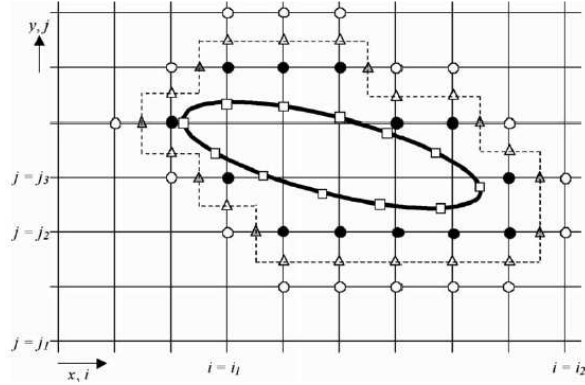


Figure 2.11: Illustration of the points used for the flow reconstruction [30]. Open squares are the body points, black circles are the forcing points. Open circles represent the last fluid layer where the continuity is still satisfied.

Further developments of the discrete forcing technique were suggested by Choi *et al.* [12]. The flow was solved in a generalized coordinate system using the finite volume approach. Time integration of the discrete Navier-Stokes equations was achieved by the artificial compressibility approach. The methodology applied in this method was similar to the one of Fadlun *et al.* [22], with the difference that the forcing was imposed on points on the both sides of the boundary (Fig. 2.12). The velocity of the interior points was set to the value at the nearest surface point while the pressure was set to the free-stream value.

The velocities at band points were decomposed into normal and tangential components, and evaluated using high-order functions. The velocity components were defined as a function of normal coordinate n as shown in Fig. 2.13.

The velocity was found using the following expression:

$$u_B^i(n) = u_T^i(n) + u_N^i(n) + u_S^i \quad (2.12)$$

subscripts T, N represent the tangential and normal components respectively. The tangential component was evaluated using a power law function while a cubic function was used for the normal component, to ensure that its second derivative vanishes at the surface. The pressure gradient was set at the surface assuming that a local pressure distribution can be given by a second-order polynomial. For non-accelerating bodies this simplified to a zero normal pressure gradient condition.

The interpolation to the “virtual” point I from the neighbouring field points was based on a merit

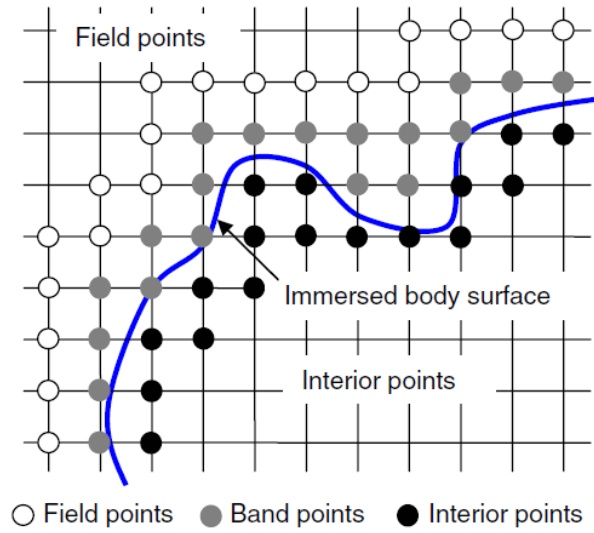


Figure 2.12: In method suggested by Choi *et al.* [12], the forcing is applied both to the band and interior points.

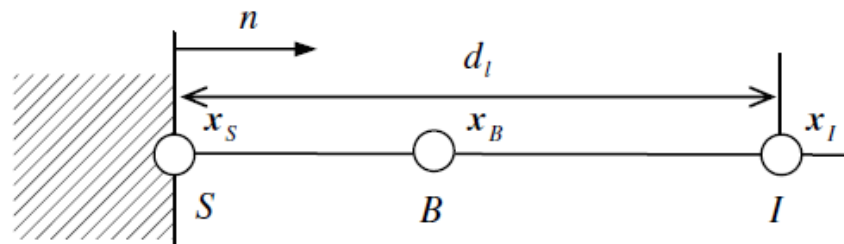


Figure 2.13: The value at the band point B is evaluated as an interpolation between surface point S and the “virtual” point I .

function. If the point lied close to the surface normal, which was crossing the corresponding band point, its interpolation coefficient were very large, and decreased with increasing distance in the tangential direction.

Higher order interpolation procedures were investigated by Peller *et al.* [94], who studied Lagrange and the least-square interpolation. It was found that least-square fit of a 3-point stencil provided accuracy improvement with a good stability, as opposed to Lagrange interpolation, which introduced stability problems. The 3-point stencil applied in [94] for a 2D case is illustrated in Fig. 2.14. For a three-dimensional analysis, a directional weighting was applied.

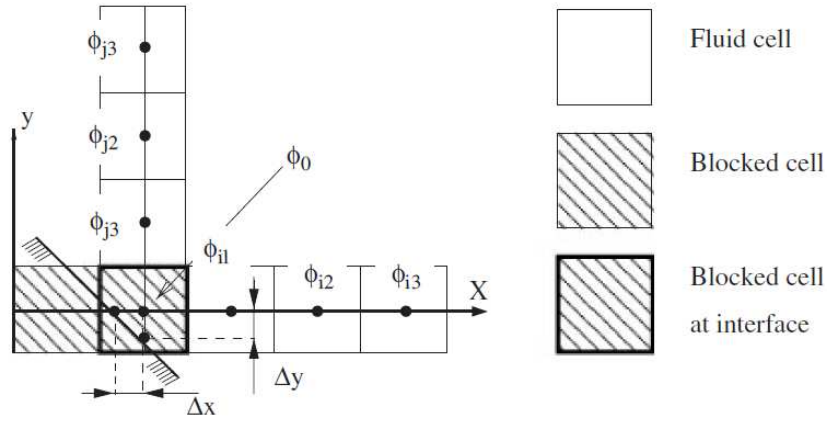


Figure 2.14: The configuration for 2D least-squares interpolation stencil [94].

Kim *et al.* [61] found that in general, the direct forcing methods do not satisfy the continuity around the particles. Explicit mass source/sink term was introduced to the continuity equation in order to compensate for the non-physical flow through the IB surface. The idea was further evolved by Huang and Sung [52], who proposed adding additional terms to the mass source/sink term to take into account the actual flow regions in the forcing cells. The resulting source term q added to the continuity equation was given as:

$$q = \frac{u_2}{x} - \frac{v_1}{y} + \frac{1}{x}u_{1c} - \frac{2}{x}u_{2c} \quad (2.13)$$

with velocities defined in the Fig. 2.15. The first two terms on the right hand side correspond to the source term given by Kim *et al.* [61], while the second pair is the source term refinement introduced in [52]. x_1, x_2 are the distance parameters.

Several other methods were also introduced. Zhang and Zheng [124], for example, utilized the direct forcing method based on interpolation of the forcing at the surface and extrapolation to the neighbouring fluid cells. Li and Wang [72], on the other hand, focused on designing forcing method for LES applications. Other researchers, like Deng [18] or Pinelli *et al.* [97] solved the forcing directly on the body elements and distributed it to the surrounding fluid cells.

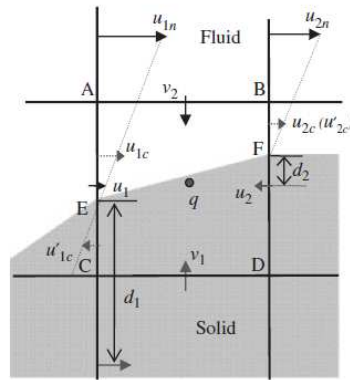


Figure 2.15: Diagram for the mass source term (q) evaluation [52]. The velocities u_{1c} and u_{2c} are evaluated using the interpolation from the neighbouring face centres.

2.2.2 Ghost cell methods

While the techniques described above usually modify the Navier-Stokes equations by adding a forcing term in the cells in the neighbourhood of the interface, the *ghost cell* methods impose the boundary conditions by setting the flow properties in the so called *ghost cells*. These are the cells that are inside the body but close to the interface.

The *ghost cell* method was first introduced by Majumdar [77] for RANS simulations on non-body fitted grids. The motivation was to use the cells outside the computational domain to impose the flow whilst the full Navier-Stokes equations would be solved for the fluid cells. This is especially important, since applying *discrete direct forcing* for RANS or LES can be problematic, as the *discrete direct forcing* modifies the flow equations directly in the fluid cells close to the IB surface.

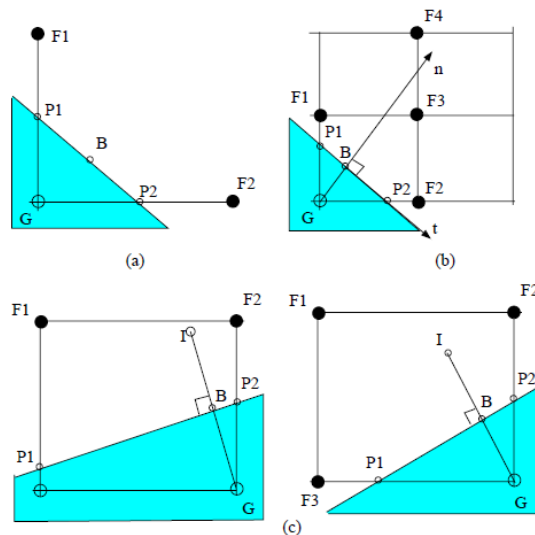


Figure 2.16: Interpolation procedures adopted by Majumdar [77]. a) linear interpolation, b) quadratic interpolation, c) bilinear interpolation.

The velocities in the *ghost cells* are set, ensuring that the reconstructed velocity at the surface of the body satisfies the no-slip boundary condition. Similarly as with the *discrete direct forcing* approach, the key element of this procedure is the choice of the interpolation method.

Several reconstruction procedures, shown in Fig. 2.16, have been investigated in [77]. In the first case (a), two neighbouring points F1 and F2 were chosen to enforce boundary condition at the boundary point B. Linear interpolation was used, resulting in:

$$u_G^j = \omega_1 u_1^j + \omega_2 u_2^j + \omega_B u_B^j \quad (2.14)$$

where ω_i are the weighting coefficients depending on the geometry. This procedure suffered from the fact that large, negative weighting factors were often observed, what had a disadvantageous impact on the solution stability.

As an alternative, an interpolation along the surface normal, that is linear in the tangential direction and quadratic along the normal was used. This resulted in the following expression for the u velocity:

$$u_G^j = a_1 n^2 + a_2 n + a_3 t + a_4 n t + a_5 \quad (2.15)$$

where a_i are the interpolation coefficients, while n, t are the distances from the surface in the normal and tangential directions respectively. The coefficients were evaluated from points F1-F4 as well as point B, represented as filled circles in the Fig. 2.16b. Still, there was no guarantee that the coefficients remain positive and smaller than one.

In order to ensure that the interpolation coefficients remain positive and smaller than unity, a so called imaginary point (IP), a projection of the *ghost node* through the IB surface, was applied (Fig. 2.16c). The value at the imaginary point I was evaluated by a bi-linear interpolation:

$$u_I^j = a_1 x + a_2 y + a_3 xy + a_4 \quad (2.16)$$

The coefficients a_i were calculated from the quadratic stencil surrounding the imaginary point. If the interpolation node lied inside the body, a corresponding body point was used instead (P1 and P2 in Fig. 2.16c). When the value at I was calculated, it was related to the velocity at the *ghost node* by:

$$u_G = 2u_{IB} - u_I \quad (2.17)$$

The *ghost cell* method was extended to three-dimensional applications by Tseng and Ferziger [110]. Two reconstruction techniques were investigated in order to avoid large negative extrapolation coefficients when some of the fluid interpolation nodes were close to the body surface. Two alternatives were suggested. One being the already mentioned application of an imaginary point created by projecting the *ghost node* into the fluid domain. In the second approach the fluid points very close to the boundary (e.g. $< 0.1 \Delta x$) were treated as boundary points with a specified velocity. Both approaches are illustrated in Fig. 2.17.

Tseng and Ferziger [110] also discussed the techniques for imposing various boundary conditions. For

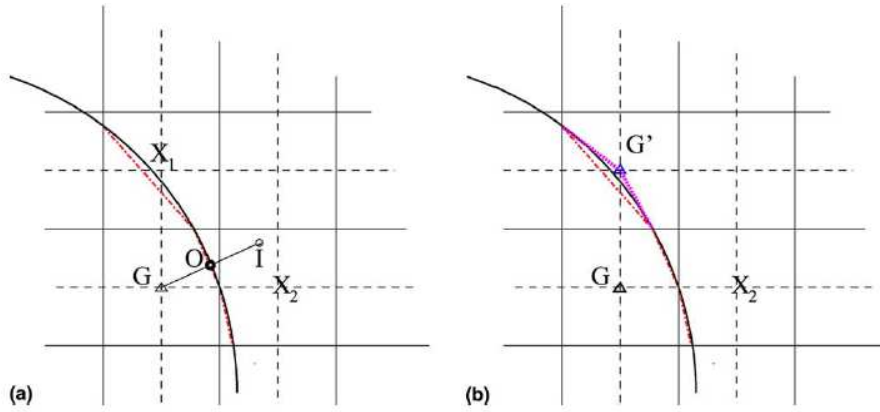


Figure 2.17: Treatments of the *ghost nodes* to prevent numerical instability from the extrapolation [110].
 a) generation of an imaginary point I , b) forcing G to be treated as a boundary point.

example, they suggested that the pressure gradient can be found by the vector decomposition in the x , y , z directions.

The *ghost cell* method has been also applied to the compressible flows by Ghias *et al.* [28]. The methodology presented in their paper was essentially similar to the ones described above although special treatment was applied in the cases where the imaginary point was surrounded by cells which belong to the body. If one of the points in the interpolation stencil was a *ghost cell* itself, a boundary point was used instead. Other ghost nodes, however, were still used for interpolation, as they did not affect the well posed behaviour of the interpolation.

Mittal *et al.* [85] adopted the technique from [28] to a general incompressible flow with complex bodies. A collocated grid was used, since it enabled a much easier implementation of the IB condition. The issue associated with generation of imaginary points for surfaces with complicated geometries was alleviated by projecting the ghost node through the surface element which contained the closest vertex. Additionally a zero-pressure gradient was imposed on the surface. Also, a problem of freshly cleared cells was discussed (see section 2.3 for more details).

The *ghost cell* method was also used by Pan and Shen [92]. In this implementation however, the *ghost cells* instead of being simply mirrored as in the other approaches, were projected through the surface and placed at a constant distance from the IB (Fig. 2.18). The distance was equal to $= \sqrt{2} x$ in 2D and $= \sqrt{3} x$ in 3D. Forcing a constant radius from the surface allowed to avoid the influence of IB on the interpolation to the imaginary point.

Mark and van Wachem [82] observed that a non-physical mass flow through the body interface is present in the *ghost cell* methods. The lack of local mass conservation decreases the accuracy of the technique and leads to spurious pressure oscillations observed in the simulations of moving particles, discussed in further sections of this chapter.

The method proposed in [82], referred to as the Mirroring Immersed Boundary Method, used the *ghost* and imaginary point pairs to enforce the velocity no-slip boundary condition at the particle surface. Additionally, in order to ensure a local mass conservation, the *ghost cell* velocities were excluded from

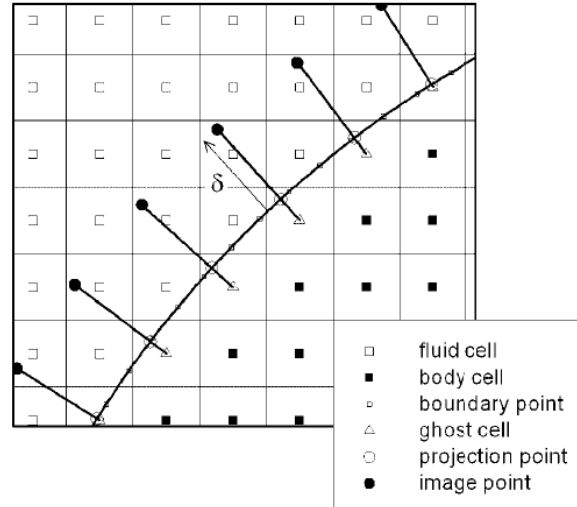


Figure 2.18: Imaginary points at constant distance δ from the IB [92]. This approach ensures that the IB surface points are not used for velocity interpolation to the imaginary points.

the continuity equation. Similar approach was proposed by Shinn *et al.* [104].

The technique introduced in [82] was extended to curvilinear grids in [90] and further developed in [123]. Also, a variation of the technique which combines setting velocity at *ghost nodes* and at fluid points close to the surface is suggested by Mark *et al.* [81].

The Mirroring Immersed Boundary Method forms the base of the technique developed during the current research project. The main idea of the developed approach is to use the *ghost* and imaginary point pairs to enforce the velocity no-slip boundary condition at the particle surface and to appropriately modify the continuity equation in the vicinity of the IB to ensure local mass conservation. Detailed discussion of the method's implementation is presented in chapter 3.

2.2.3 Cut-cell methods

The last group of techniques classified as a discrete forcing approach are the *cut-cell* methods, sometimes referred to as Cartesian methods. Contrary to the techniques presented above, the main principle of the *cut-cell* methods is to locally modify the fluid mesh. The cells cut by the Immersed Boundary are reshaped, so that a locally unstructured grid is formed around the body. The first reported attempts applying this methodology for inviscid flows are found in [14].

In principle, the *cut-cell* methods attempt to solve the Navier-Stokes and continuity equations in all the fluid cells without the need to use an additional forcing term. Also, there is no need to set velocity in the cells lying outside the flow domain. The main challenge for the *cut-cell* method is to accurately describe the modified fluid cells. Additionally, it is of critical importance to use the proper interpolation procedures, in order to evaluate the flow variables on cell faces and at the Immersed Boundary.

Ye *et al.* [120] were among the first researchers to adopt the *cut-cell* approach for simulation of viscous flows. The numerical procedure presented in [120] was designed for two-dimensional cases and

was shown to be second-order accurate. Extension of this method to moving cases was proposed in [111], and is discussed further in section 2.3. In order to avoid numerical instabilities related to the small cells, a cell merging procedure was proposed. If the centre of the original cell was inside the IB, the modified cell was merged to one of its fluid neighbours to form a trapezoidal cell as shown in Fig. 2.19a.

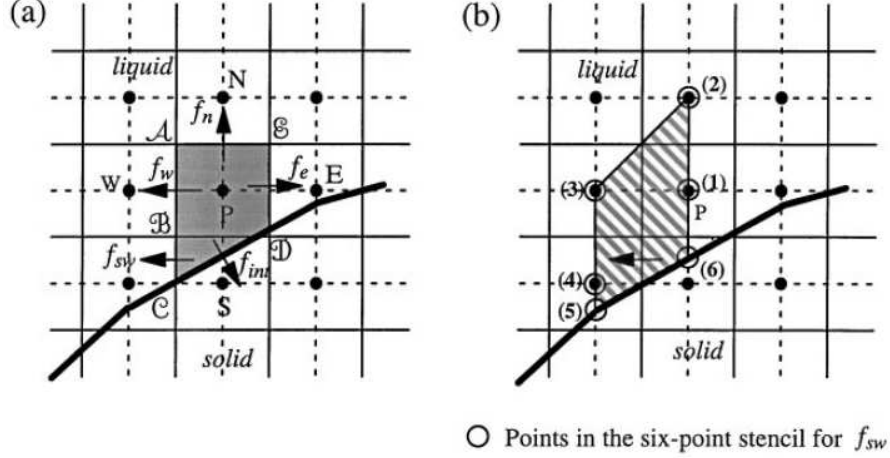


Figure 2.19: Key features of the *cut-cell* approach [120]. a) an example of trapezoidal cell formed through merging. b) stencil used for interpolation of f_{sw} velocity.

An accurate discretisation in this procedure is critical for achieving a precise flow solution. The values at faces, which are not cut by the boundary (e.g. face f_w in the figure), were evaluated by a simple linear interpolation:

$$w = W \frac{x_P - x_w}{x_P - x_W} + P \frac{x_P - x_w}{x_P - x_W} \quad (2.18)$$

where W is:

$$W = \frac{x_P - x_w}{x_P - x_W} \quad (2.19)$$

Special treatment was applied for those faces, which are cut by the boundary. In this case, the flow variables were expressed by a two-dimensional polynomial quadratic in the y -direction and linear in x evaluated using a larger cell stencil (Fig. 2.19b):

$$= c_1 x y^2 + c_2 y^2 + c_3 x y + c_4 y c_5 x + c_6 \quad (2.20)$$

while the normal gradient (in x direction) was evaluated as:

$$\frac{\partial}{\partial x} = c_1 y^2 + c_3 y + c_5 \quad (2.21)$$

c_i are the interpolation coefficients depending on the geometry of the 6-point stencil illustrated in the figure.

Apart from the special interpolation procedure for the fluxes at fluid cell faces, the *cut-cell* method

required evaluation of the boundary contribution as well. The velocity values for the Dirichlet boundary condition were given by the velocity of the IB surface, however the gradients had to be calculated by interpolation. Ye *et al.* [120] suggested decomposing the normal derivative at the surface into:

$$\frac{\partial}{\partial n}_{IB} = \frac{\partial}{\partial x} n_{x IB} + \frac{\partial}{\partial y} n_{y IB} \quad (2.22)$$

The gradient in the y direction was calculated by taking a 3-point stencil in the y direction as illustrated in Fig. 2.20a. Evaluation of the gradient in the x direction was more complicated and required a larger number of cells used for interpolation. The procedure was quite similar to the one adopted for evaluation of the flow values at the south-west face f_{SW} . Finally, the normal gradient at the IB surface was expressed as a sum of 9 point stencil:

$$\frac{\partial}{\partial n}_{IB} = \sum_{i=1}^9 \alpha_i \quad (2.23)$$

where α_i is the interpolation coefficient of the i 'th component of the stencil.

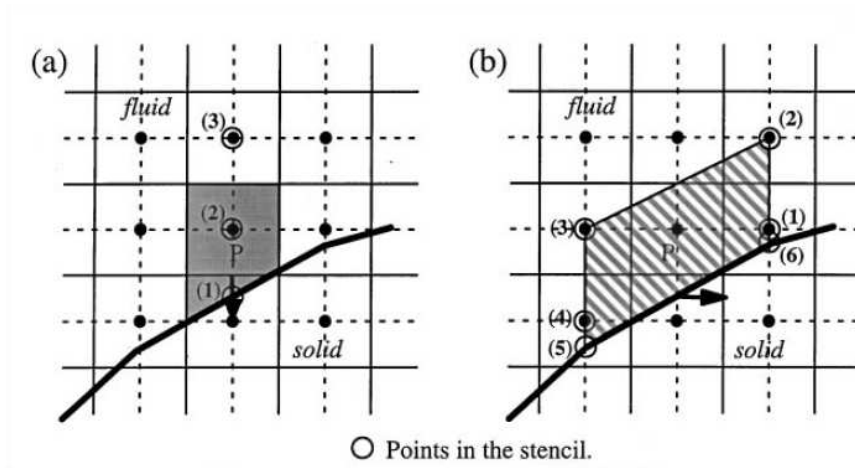


Figure 2.20: Calculation of gradient at the IB surface [120]. a) stencil used for calculating $\frac{\partial}{\partial y}$. b) trapezoidal stencil for calculating the gradient at the IB in the x direction.

Another implementation of the *cut-cell* approach was proposed by Chung [13]. Similarly as in [111, 120], small *cut-cells* were merged with their neighbours to avoid the stability problems. This method, however, differs somewhat in the treatment of the body surface and interpolation to the cut faces procedure. The viscous stress on the interface were evaluated based on the normal gradient of the tangential velocity component u^T . u^T was obtained by using the flow velocities of the interface and velocities calculated at auxiliary points b_1, b_2 shown in Fig. 2.21. The tangential velocity gradient in normal direction was therefore:

$$\frac{u^T}{\partial n}_{IB} = s^j (8u_{IB}^j + 9u_{b_1}^j - u_{b_2}^j) / 6h \quad (2.24)$$

where s^j is the tangential vector. Values at the auxiliary points were calculated using bilinear interpola-

tion from the surrounding fluid nodes or the points at the interface.

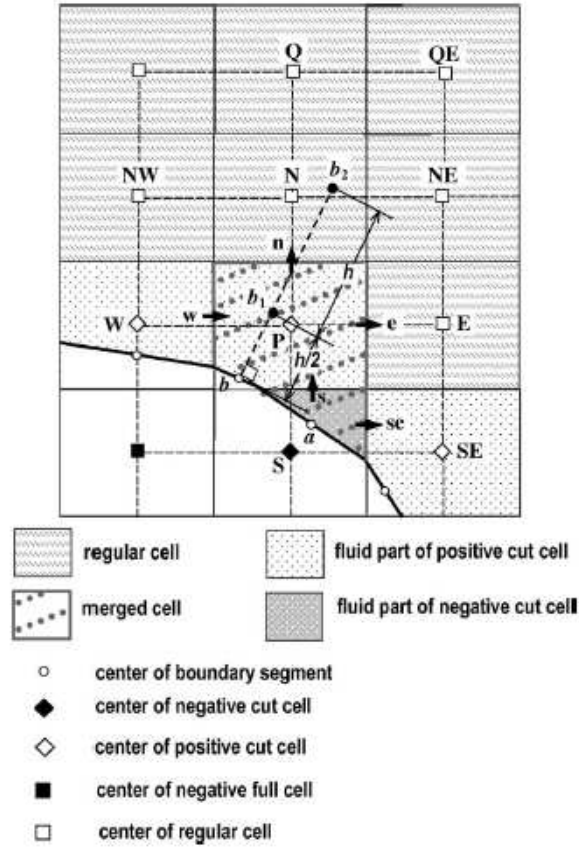


Figure 2.21: Schematic illustration of interpolation stencil used in [13]. The tangential velocity gradient at the IB surface is calculated based on the velocity values at points b , b_1 , b_2 .

The Neumann boundary condition was set for the Poisson pressure equation, resulting in a zero-pressure gradient at the surface. The corresponding expression is:

$$p_{IB} = p_{b_1} \quad (2.25)$$

Calculation of the flow variables at *cut-cell* faces (e.g. face se in the Fig. 2.21) utilized the local stencil around the face with surface points being used instead of the interior cells. Hence, the value of se was found as:

$$se = a_P P + a_E E + a_{SE} SE + a_{IB} IB \quad (2.26)$$

where a_i are the geometric interpolation coefficients. Flow values at the interface were obtained from either the Dirichlet boundary condition (velocity) or from the equation 2.25 (pressure).

As previously stated, one of the major obstacles for *cut-cell* methods is the existence of cells with a very small fluid volume fraction, often referred to as small cells. Small cells can introduce various difficulties as explained by Kirkpatrick *et al.* [62]. Firstly, very small time step has to be used in order to satisfy the condition of the CFL number smaller than unity, since the cell dimension is very small as

well. Also, the variation of cell size can lead to high stiffness of the linearised set of equations used to resolve the flow field. Finally, as far as staggered grids are concerned, sometimes the mass conservation cannot be solved since a velocity cell might not have two corresponding pressure nodes.

A number of approaches which address the issue of small cells is at hand. So far what has been discussed is the cell merging technique, in which the small cells are merged to their fluid neighbours. Although conceptually simple, cell merging can be hard to implement, since the discretisation stencil for the merged cell becomes different to that of a standard cell. A requirement to use the diagonally adjacent cells considerably increases the complexity of the code and has a negative impact on the parallelization. Additionally, consistent merging for the three-dimensional problems proves to be a major challenge.

Kirkpatrick *et al.* [62] suggested an alternative approach that is suitable for 3D problems. In the cell linking technique, a small cell is related to the neighbouring fluid cell by a master-slave relationship. Instead of forming a new larger cell, the small cell is moved to the same position as a master node as illustrated in Fig. 2.22. Both cells still remain as separate entities. Because of this property, all Navier-Stokes and continuity equations terms can be calculated in the same way that is applied for standard fluid cells. Cell linking allows to apply the pressure gradient and the velocity correction evaluated at the master cell to the slave node.

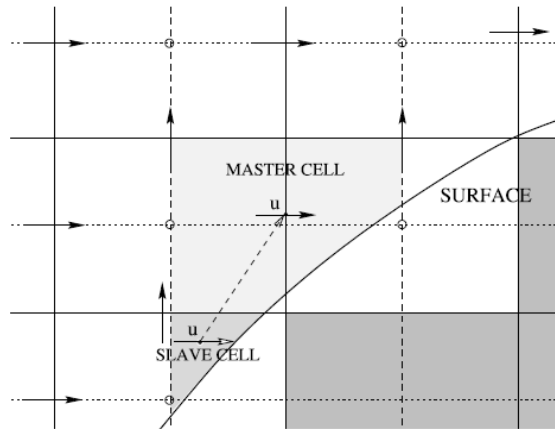


Figure 2.22: An illustration of the slave cell being moved, so that its centre is coincident with the master node [62].

It was suggested that for exceptionally small cells, with the largest area in any direction being smaller than 1% of the original face area, a merging procedure is applied. A consistent procedure for choosing master-slave pair was also discussed.

A different strategy which allows to deal with small cells, called cell mixing was introduced by Hu *et al.* [51]. The procedure was applied for simulation of compressible inviscid flows with multiple interacting fluids or with rigid bodies. This approach was also used to solve incompressible viscous flows by Meyer *et al.* [84]. Here, the flow equations were discretised on staggered grid. The idea of cell mixing can be summarised as follows. Firstly, the target cells are determined based on the local surface normal. Only fluid neighbours in the x and y directions are used in [51], while all the fluid neighbours

are treated as target cells in [84], what is seen in Fig. 2.23.

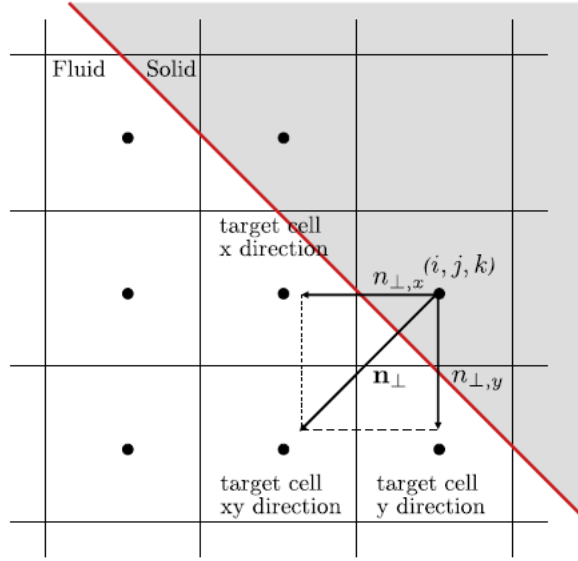


Figure 2.23: Illustration of the cell mixing concept for the $i j k$ cell [84]. Fraction of the momentum imbalance in the source cell $i j k$ is transferred to target cells - fluid neighbours of the $i j k$.

The momentum equations of the target cells are modified by the addition of the conservative exchange X^j from the small cell. The exact value of X^j depends on the orientation of the cells with respect to each other and their volume fraction:

$$X^j = \frac{j}{j} \quad q^j da^j_{src} \quad (2.27)$$

where q^j is the conserved quantity and a^j is the the surface vector of the cell face. Coefficients j are:

$$\begin{aligned} x &= n_x^2 \text{tgt} & y &= n_y^2 \text{tgt} & z &= n_z^2 \text{tgt} \\ xy &= n_x n_y \text{tgt} & xz &= n_x n_z \text{tgt} & yz &= n_y n_z \text{tgt} \\ xyz &= n_x n_y n_z \text{tgt} \end{aligned} \quad (2.28)$$

where tgt is the volume fraction of the target cell.

The X^j term added to the target cell has the same value as the one subtracted from the small cell hence the property is conserved. The mixing is applied before the boundary conditions are imposed and the pressure correction equation is solved.

Another development in the *cut-cell* method was proposed by Hartmann *et al.* [43]. The technique was designed to solve the Navier-Stokes equations for compressible flows. Small cells were treated using a mixed cell linking/merging approach. Unlike the in the aforementioned formulations, here the gradients of the flow variables were evaluated at the cell centres using the least-squares reconstruction, which is

very useful on unstructured grids. The property value at cell face was therefore calculated as:

$$f = P + \frac{x_{Pf}^j}{x_P^j} \quad (2.29)$$

where P was the current cell centre, while x_{Pf}^j was the distance between cell centre and the face centre.

Ghost nodes were introduced in order to allow gradient calculation using the least-squares formulation in the boundary nodes. These additional cells were projections of the boundary cell centres through the interface outside the fluid domain (Fig. 2.24). The flow variables in the *ghost cells* were extrapolated from the boundary cells and used only for the gradient calculations. Also, a discussion on calculation of the gradients at the boundary was given. The method was further developed in [37, 38, 42, 44].

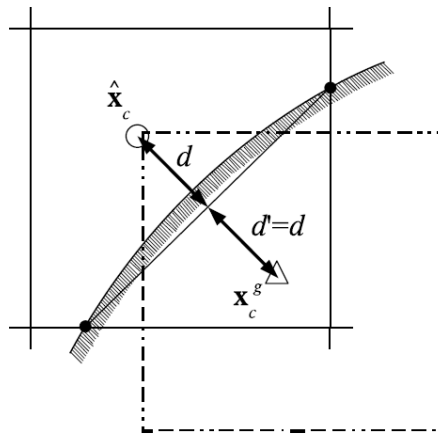


Figure 2.24: The location of the *ghost cell* used for the gradient calculations using the least-square formulation [43]. The flow variables in the *ghost node* are calculated by extrapolation from the fluid domain.

Other *cut-cell* examples can be found in e.g. [9], [55], [27] or [80]. In general, the *cut-cell* method is the only approach that strictly conserves both mass and momentum of the flow both locally and globally. The main drawback of this method, however, is its complexity and the high computational cost associated with the additional operations required for accurate cell description. Moreover, an extension of the *cut-cell* method ideas to three-dimensional problems considerably complicates the technique implementation.

2.3 Applying Immersed Boundary Method to the moving interfaces

As stated at the beginning of this chapter, one of the main motivations for development of the Immersed Boundary Method, is its ability to efficiently simulate flows with moving boundaries, associated with the fact that there is no need for additional fluid grid re-meshing.

This thesis concentrates on those techniques which use the discrete forcing approach, due to their

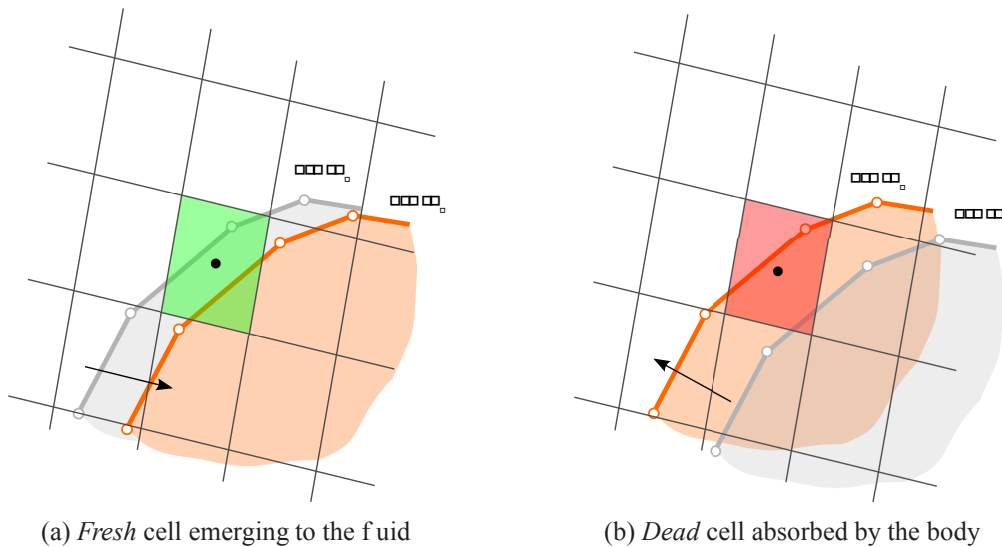


Figure 2.25: Change of the cell behaviour as the IB moves from the location occupied at $t = t_0$ to a new position at $t = t_1$. Cells can either emerge into the fluid domain (a) or be absorbed by the body (b).

ability to preserve a sharp representation of the body surface and the fact that no additional constraints, encountered in the continuous forcing procedure, are imposed on the numerical method. Extensions of the methods discussed in the previous section to the simulations with moving bodies is fairly straightforward.

One possible strategy involves using a non-inertial reference frame for the moving body as suggested by Kim and Choi [60], who applied the IB formulation with source/sink terms proposed in [61]. Applying a non-inertial coordinate system requires introduction of additional terms to the system of equations, such as the Coriolis force. What is more important, however, is that the technique proposed in [60] is limited only to the cases of single particles and is therefore not applicable to general particulate flows.

A more common approach for moving body problems is to simply translate the particle grid and recalculate the IB interaction with the underlying fluid mesh (*i.e.* the interpolation coefficients etc.). Still, some additional attention has to be given to the so called *fresh cells*, which are the fluid cells that used to be inside of the IB in the previous time-step. Also, it is very common to observe spurious pressure oscillations for the moving particles when the discrete forcing is used [81, 92]. Although in general these oscillations do not affect the overall solution (*i.e.* the average drag coefficient remains constant and realistic), they may have significant implications in certain flow cases.

2.3.1 Fresh-cell treatment

When the Immersed Boundary is moving through the flow domain, some cells which were previously inside the surface will emerge into the fluid, while other cells, previously in the fluid will be absorbed by the particle. Such cells are referred to as *fresh* and *dead* cells respectively, and are illustrated in Fig. 2.25.

The time advancement of Navier-Stokes equation requires the explicit fluid velocity from the previous time-step, however the flow variables in the *fresh* cells have no physical history term, therefore they need special treatment.

Udaykumar [111] suggested that the cells emerging into the fluid domain can be temporarily merged to their fluid neighbour. Instead of solving the Navier-Stokes equation for such a cell, the velocity was extrapolated from the neighbouring fluid cells. Fig. 2.26 illustrates the procedure.

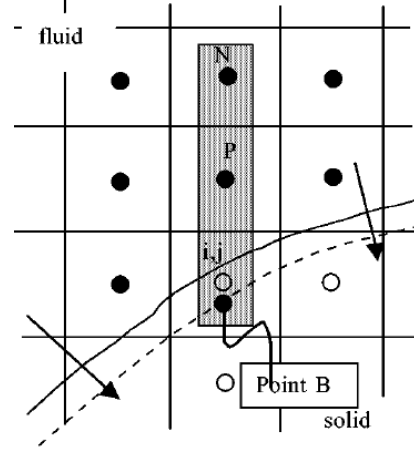


Figure 2.26: The stencil used for evaluation of the flow velocity in the *fresh cell* i, j [111]. The choice of stencil depends on the dominant direction of the IB, y direction is chosen in this case, therefore cells P and N along with the surface point B are used for the calculation of the velocities at i, j .

The velocity was therefore calculated as:

$$u^j(y) = a_0^j + a_1^j y + a_2^j y^2 \quad (2.30)$$

where a_i^j are the extrapolation coefficients depending on the geometry and the flow values at points P , N , B applied for the procedure. The direction of the interpolation depends on the dominant direction of the IB (horizontal or vertical). The temporary merging is applied only in the time-steps when the cell emerges into the flow. A conventional momentum equation was used in the subsequent steps.

A different approach was applied by Gilmanov *et al.* [30]. Taking advantage of the dual-time stepping scheme and the fact that the boundary conditions were enforced implicitly, the issue of *fresh* cells was avoided by simply ensuring that a body did not move more than one cell during a time-step. This resulted in a maximum time-step for a moving particle simulation given as:

$$t = \frac{x}{\max(u_{IB}^x, u_{IB}^y, u_{IB}^z)} \quad (2.31)$$

where x is the local grid spacing around the boundary.

A similar time-step limit was imposed in the method used in [85]. In this case, however, since the *ghost cell* method was used, there was a need for the flow reconstruction in the *fresh* cells. This was

achieved by interpolation from neighbouring fluid cells as illustrated in Fig. 2.27. The value in the *fresh* cell was evaluated using the same stencil which was used to calculate the flow velocity in the imaginary point IP at the previous time-step. In order to increase the accuracy of the prediction the velocity value of a boundary point BI at a current time was taken into account as well.

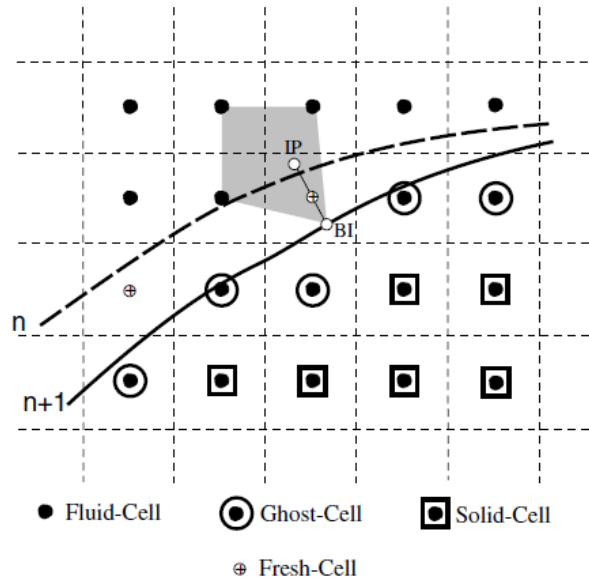


Figure 2.27: Flow reconstruction at *fresh* cells [85]. The velocity in the *fresh* cell is interpolated from the stencil used for the velocity interpolation to the IP at time n and a boundary point BI at time $n + 1$.

If forcing is imposed on the external fluid layer as in [118], there is also a need to predict the flow variables values of the *fresh* cells, since their history term, *i.e.* the velocity at previous time-step, is not known. Yang and Balaras [118] proposed a method called “field extension” in order to determine the “old” flow prediction in such cells. This technique reduces to extrapolation of the resolved flow variables into the solid cells at the end of the time-step. The concept is shown in Fig. 2.28. First the interior cells were identified (they corresponded to the *ghost cells* in other methods). Then the values were obtained by interpolation from the stencil shown in the figure. A simplified version of this approach was proposed recently in [119].

An alternative approach has been developed recently by Lee and You [71]. Instead of reconstructing the velocities at *fresh* cells, they proposed to change the time-step for those cells to perform a backward time integration. The idea can be illustrated by Fig. 2.29. During a single time-step, while advancing from time t^n to time t^{n+1} the Immersed Boundary intersects with two cell centres at times t^{n+1} and t^{n+2} . The flow velocity at the time of intersection is known and has to correspond to the velocity of the boundary at that time.

The discretised time advancement term in momentum equation is:

$$\frac{w^{j,n+1} - w^{j,n}}{t} = RHS^j \quad (2.32)$$

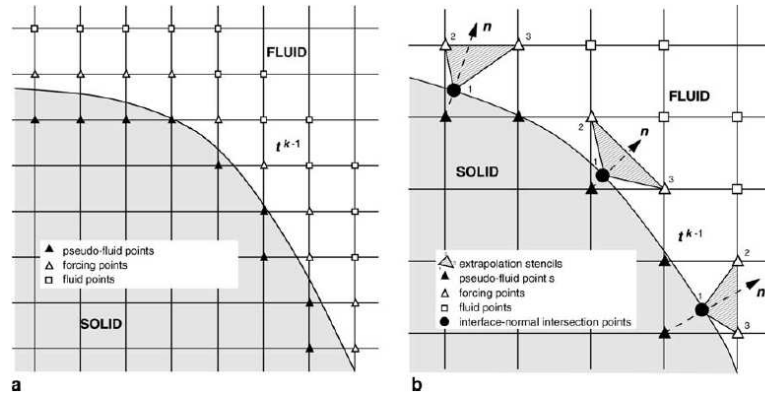


Figure 2.28: Field extension technique [118]. The velocities of the interior cells (black triangles) are extrapolated from the resolved flow field using the interface and the forcing points.

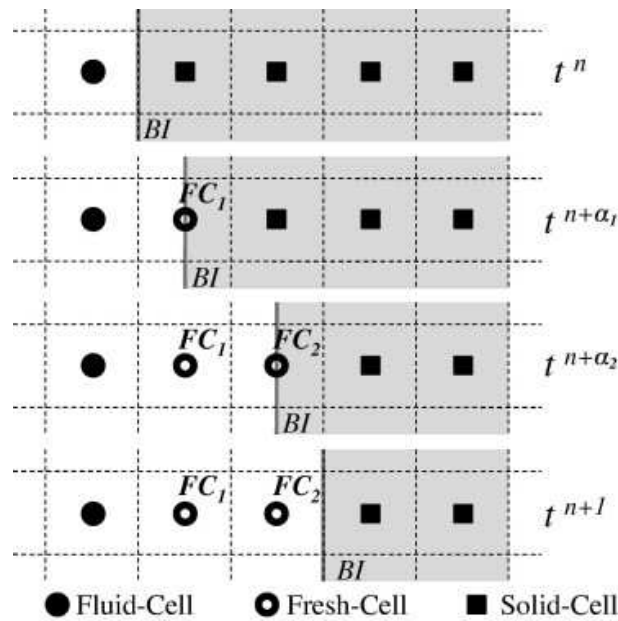


Figure 2.29: Backward time integration [71]. The boundary interface BI intersects FC_1 and FC_2 cell centres as it moves from time t^n to t^{n+1} . The time of intersection can be used to locally modify the time-step size for *fresh cells*.

where RHS^j are remaining terms of the momentum equation. It was proposed to re-write the equation 2.32 as:

$$\frac{u^{j\ n+1} - u_{IB}^{j\ n+ \ k}}{t^{\ k}} = RHS^j \quad (2.33)$$

where $u_{IB}^{j\ n+ \ k}$ is the body velocity at time $t^{n+ \ k}$ and $t^{\ k} = t^{n+1} - t^{n+ \ k}$. This modified momentum equation applies only to the *fresh* cells. The main advantage of this approach is that it can handle multiple layers of the *fresh* cells, therefore relaxing the maximum time-step condition on the particle motion.

2.3.2 Spurious pressure oscillations

Non-physical oscillations in the pressure field have been observed by a number of researchers in the simulations with moving bodies. The pressure fluctuations are commonly encountered, when the fluid-particle interactions are modelled by means of the discrete forcing approach (e.g. [81, 92, 103, 113]). Although these oscillations usually have a limited effect on the main simulation parameters (e.g. terminal velocity of a sedimenting particle), their existence can have a strong impact on unsteady flow simulations. Furthermore, the magnitude of the oscillations tends to grow as the time-step of the simulation is decreased, what is somewhat counter-intuitive. This undesirable property of the Immersed Boundary Method has been recently investigated by numerous researchers [70, 73, 76, 103], who propose various techniques to decrease the oscillations. Still, a coherent explanation for the source of the issue is yet to be described.

Uhlmann [112] reported that force oscillations were present in simulations of body motion when methods designed by Fadlun *et al.* [22] or Kim *et al.* [61] had been applied. In the subsequent paper [113] an argument is made, which relates the oscillations to insufficient smoothing. A suggested solution of the issue involves calculating the forcing term directly at the surface and applying it to the fluid through a distribution function. This, however, results in a diffused interface, what is undesirable from the point of accuracy.

An attempt to identify the source of the pressure oscillations was made by Lee *et al.* [70], who performed their study using the *discrete direct forcing* inside the IB on a staggered grid. The pressure oscillations were attributed to two factors: spatial discontinuity in pressure and a temporal discontinuity in velocity.

Spatial discontinuity of the pressure was illustrated by means of simulation of a stationary circular cylinder at $Re = 40$ with 50 grid cells per diameter. Time development of the pressure along the cylinder centerline is shown in Fig. 2.30. It can be clearly seen that the pressure inside the body develops to values much higher than that of the surrounding fluid in order to satisfy the global continuity equation. This, however, does not have a major influence on simulation of the stationary body, as the pressure inside of the body is not used to update the velocity in the fluid grid points.

Still, the pressure discontinuity at the interface, illustrated in Fig. 2.30, may have significant implications for the moving particle case. This was investigated in [70] by simulating a square cylinder, described by 50 grid cells, moving periodically in the fluid. At non-dimensional time $t_{u/d} = 4.280 - 4.281$ the cylinder interface passed through a grid face, resulting in an emergence of a *fresh* cell

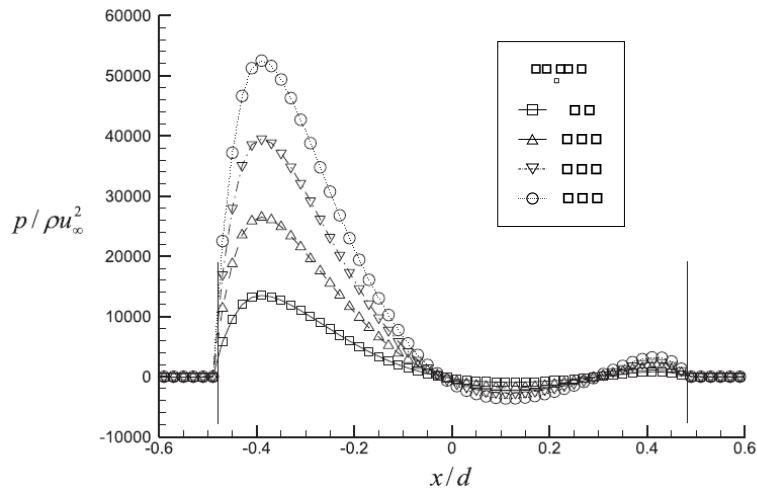


Figure 2.30: Pressure distribution along the centerline of a stationary cylinder at $Re = 40$ at different time-steps [70].

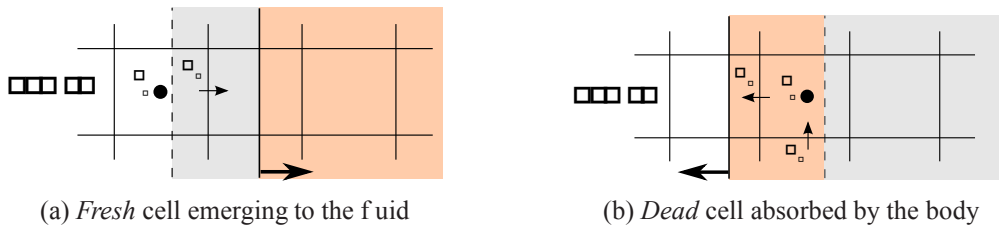


Figure 2.31: The position of the measurement points on the staggered grid used in the oscillating square cylinder case [70].

depicted in Fig. 2.31a. Time traces of the pressure and intermediate velocity in the vicinity of the cylinder surface, with and without source/sink term, are presented in Fig. 2.32. The fluid pressure increased as the body approached the grid cell and experienced strong oscillations afterwards. The velocity field, as it was affected by the non-physical pressure behaviour also showed strong oscillations. An addition of the appropriate mass source/sink term significantly decreased the magnitude of the oscillations.

While the spatial discontinuity in the pressure manifested itself in the case of *fresh* cells, the temporal discontinuity in velocity was observed when the fluid cell had been absorbed by the body (*dead* cell). The time traces of velocities and pressures for this case are illustrated in Fig. 2.33 with the measurement point definitions given in Fig. 2.31b. Due to the spatial discretisation, there was a jump when the inside velocity was adjusted to the body velocity. This introduced oscillation in the pressure field as observed in Fig. 2.33.

Lee *et al.* [70] investigated also the effect of both the grid spacing and the time-step size on the magnitude of the pressure oscillations. It has been found that the oscillations become stronger as the time-step decreases, however they can be decreased by using finer grid spacing. The formula relating the

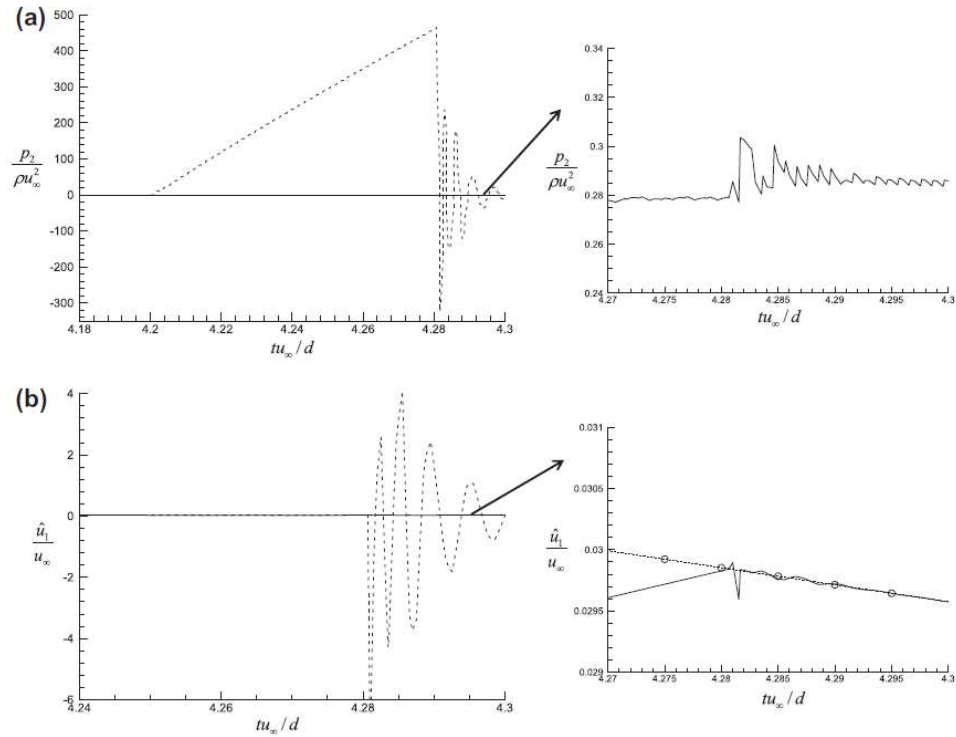


Figure 2.32: Time traces of flow variables for a *fresh* cell [70]. Right hand side plots show traces with application of mass source/sink term. Location of measurement points are defined in Fig. 2.31a.

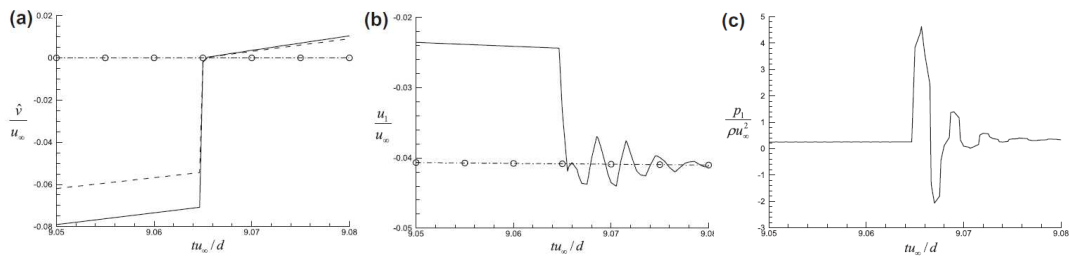


Figure 2.33: Time traces of flow variables for a *dead* cell [70]. Location of measurement points are defined in Fig. 2.31b.

magnitude of oscillations to the time-step size and the grid refinement can be expressed as:

$$\begin{aligned}
 P_{osc} & \sim \frac{x^2}{t} & \text{in 2D} \\
 P_{osc} & \sim \frac{x^3}{t} & \text{in 3D}
 \end{aligned} \tag{2.34}$$

where P_{osc} is the magnitude of the pressure oscillations, x is the local grid spacing, while t is the simulation time-step.

Additionally, it was shown that applying both the field reconstruction suggested by Yang and Balaras [118], or adding a mass source/sink proposed by Kim *et al.* [61], had a positive effect and resulted in decreasing the magnitude of oscillations. This observation agrees with the methodology applied in [71].

An investigation of the source of the pressure oscillations was also performed by Seo and Mittal [103], who related it to a violation of the geometrical conservation law, *i.e.* the fact that the number of grid cells occupied by the IB changes rapidly in time. This leads to the lack of global mass conservation in time. An analysis of the continuity equation for the moving Immersed Boundary is required to understand the problem (Fig. 2.34):

$$\begin{aligned}
 \frac{dV_{IB}}{dt} + \int_{IB} u^j n^j dA = \\
 \frac{V}{t} (q^{n+1} - q^n) + \int_{IB} u_{IB}^j n^j dA - \int_{SS} u_f^j n^j dA
 \end{aligned} \tag{2.35}$$

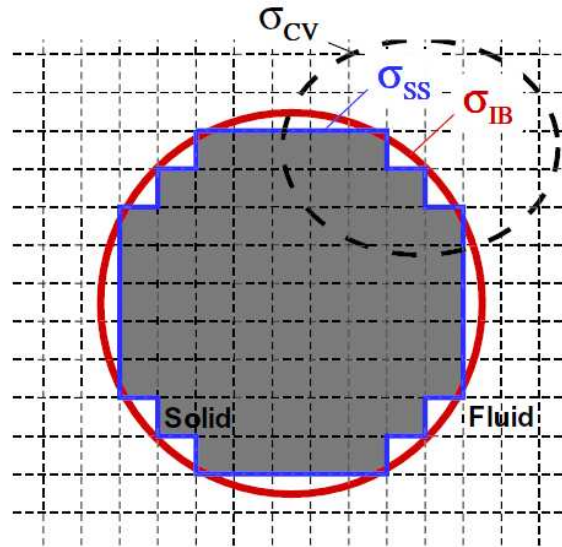


Figure 2.34: Immersed Boundary on a Cartesian grid [103]. IB corresponds to the IB surface, SS is the stair-step representation of the IB.

where V_{IB} is the volume enclosed by the IB surface IB , q^k corresponds to the number of solid cells with volume V at time k ; u_{IB}^j , u_f^j are the velocities of the body and the velocity calculated at the cell

face respectively, while S_S is the surface enclosed by the “stair-step” boundary around the solid cells. Terms on the right hand side of the equation correspond to the error in the mass conservation, which can be discretised as follows:

$$\frac{dV_{IB}}{dt} + \sum_f u^j n^j dA = \frac{V}{t} (q^{n+1} - q^n) - \sum_f A u_{IB}^j n_f^j n_{IB}^j (n_f^i n_{IB}^i) + d^i \frac{u^j}{x^i} n_f^j \quad (2.36)$$

here, A is the cell face area, n_f is the face normal, n_{IB} is the IB surface normal and d^i is the projection distance from the surface element to the corresponding face center.

The first term on the right hand side was interpreted as the change of volume related to the *fresh/dead* cells appearing in the simulation. The remaining terms describe the error associated with leakage through the interface. The second term was associated with the difference of the real surface and the “stair-step” boundary, while the last term corresponds to the mismatch between the IB and the location where the continuity is enforced.

It was observed that for a stationary body ($u_{IB}^j = 0$) only the last term remains. Moreover, the first term related to the *fresh/dead* cells, is expected to experience high discontinuity in time as the particle moves. As shown by Seo and Mittal [103], this is the main source of the pressure fluctuations observed in IB simulations.

An interesting observation can be made if the aforementioned continuity equation is applied to a single cell. Here the volume conservation error becomes:

$$-\frac{V}{t} \sum_f u_{IB}^j n^j A = -\frac{V}{t} 1 - CFL_{IB} \quad (2.37)$$

where $CFL_{IB} = \sum_f u_{IB}^j n^j A / V$. This implies that the pressure oscillations are proportional to the cell face area and inversely proportional to the time-step, what agrees with the earlier observations from [70]. Equation 2.37 also suggests, that if the body moves exactly one cell in a single time-step *i. e.* $CFL_{IB} = 1$, the oscillations should disappear. Seo and Mittal report confirms this observation.

Having established that the pressure fluctuations are caused by a rapid change of volume encompassed by the Immersed Boundary, Seo and Mittal suggested a technique to eliminate the problem. The proposed idea was to correct the *ghost cell* flow solution with the mass fluxes obtained using the *cut-cell* method.

Both momentum and continuity equations of cells, whose centres lie in the fluid domain, were solved using the finite-volume discretisation described in [120]. The *ghost cell* values were, however, set using the velocity and the pressure gradient mirroring. A procedure similar to cell-mixing, called “virtual cell merging”, was applied to the small *cut cells* in order to satisfy the continuity equation of those cells.

This method essentially transferred the error in mass conservation of the small cells to the neighbouring fluid cells as shown in Fig. 2.35. The mass flux X_j transferred to the neighbouring cell j was evaluated as:

$$X_j = \frac{X_j}{j} \sum_{src} u^j n^j dA \quad (2.38)$$

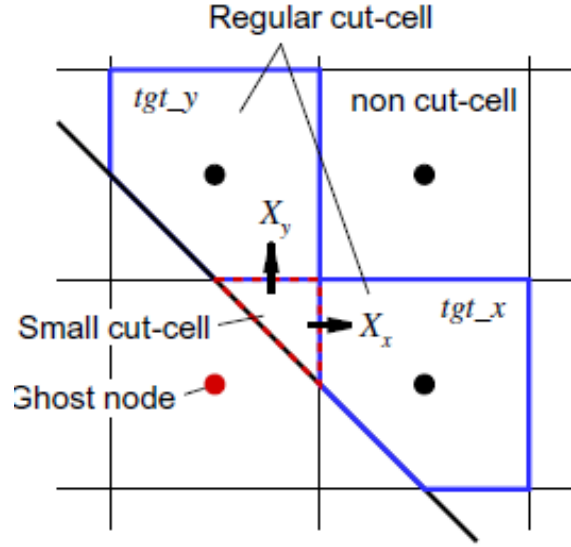


Figure 2.35: Schematic illustration of the virtual cell merging procedure [103]. Mass conservation error from the small *cut-cell* is transferred to its neighbours by means of source term X_j .

$$X_j = \begin{cases} (n^j)^2 A & t_{gt} > 0.5 \\ 0 & t_{gt} \leq 0.5 \end{cases} \quad (2.39)$$

A is the flow face area, while t_{gt} is the fluid volume fraction of the target cell. The corrected continuity equations for the small cells and the neighbouring fluid cells will then become:

$$u^j n^j dA_{src} = u^j n^j dA_{src} \quad X_j = 0 \quad (2.40)$$

$$u^j n^j dA_{tgt} = u^j n^j dA_{tgt} + X_j \quad (2.41)$$

The subsequent results using the proposed approach showed a significant reduction in the amplitude of the pressure oscillations by approximately six times, compared to the unmodified case. Furthermore, a quantitative analysis of the results allowed to observe that the scaling properties of the fluctuations remained practically the same as the ones described by equation 2.34.

Several mechanisms responsible for the remaining oscillations were proposed:

The equations for the *fresh* cells were ill-posed, more accurate prediction of the “old” time-step velocity is needed.

A temporal discontinuity in the velocity was observed when the particle moved through the domain.

Some errors in the momentum conservation in the *cut-cells* might have still been present.

The surface was locally linearised, what could lead to inaccuracies.

The effect of applying different forcing strategies on the pressure oscillations of a moving particle has been investigated by Liao *et al.* [73]. Three possible scenarios were considered: forcing in the cells inside of the body (extrapolation), forcing in the fluid cells in the near vicinity of the IB (interpolation) and a combination of interpolation and solid body forcing inside of the interface. The results obtained for a circular cylinder oscillating in a fluid are shown in Fig. 2.36.

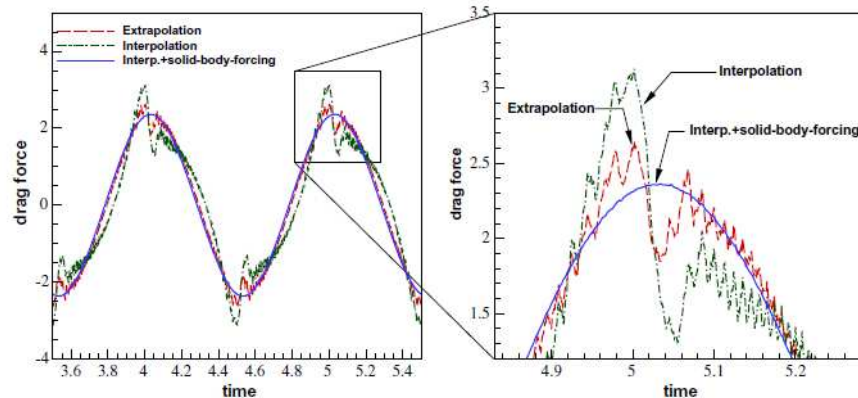


Figure 2.36: Effect of different forcing strategies on the pressure oscillations for an oscillating cylinder case [73].

Fourier analysis was used in order to quantify the amplitude and frequencies of the oscillations. The resulting amplitude spectra show that the behaviour of the first two forcing strategies is almost the same, although they differ in the magnitude of the oscillations (Fig. 2.37a-b). On the other hand, when a combination of interpolation and solid body forcing was applied, the only visible mode corresponds to the frequency of body motion (Fig. 2.37c).

These observations can be related to the aforementioned studies, especially to the simulations performed by Lee *et al.* [70]. It appears to be the case, that taking into account solid body forcing inside the IB ensures a smooth transition of the flow variables as the interface passes through the grid faces, therefore reducing the spurious oscillations.

The technique presented in [73] was subsequently adopted to study the natural and forced convection for the flows with moving particles in [74].

A study on the artificial pressure oscillations was also conducted by Luo *et al.* [76], who attributed them to the sudden changes of the numerical description of the computational cells as the body moves through the fluid. It was indicated that if a cell changes its character, *i.e.* a cell with an applied forcing becomes a cell governed by the discretised Navier-Stokes equation or vice-versa, the two subsequent descriptions (forcing/momentum equation) will be generally not consistent with each other.

A smooth transition of the numerical stencils was proposed to solve this issue. A combination of *discrete direct forcing* based on [118] with the *ghost cell* method [85] was applied. The forcing cells in the fluid domain were however treated in a hybrid way. Depending on the distance from the boundary,

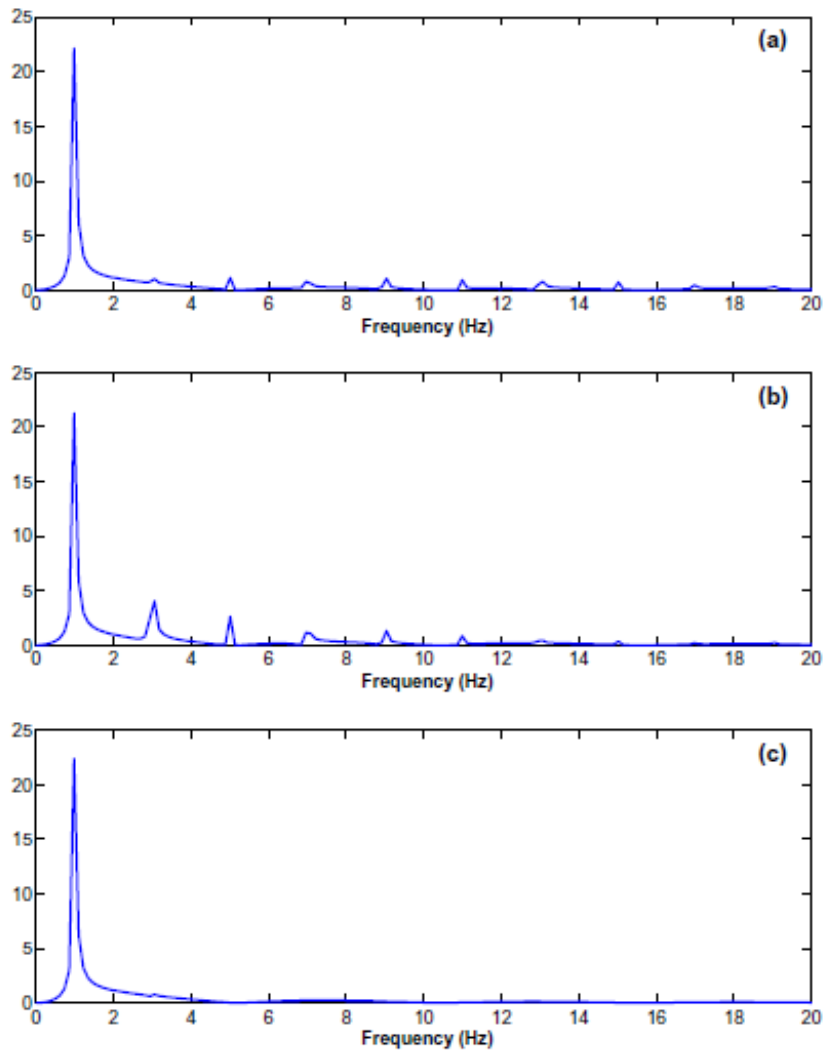


Figure 2.37: Oscillation amplitude spectra for different forcing strategies [73].

the cell was mostly resolved by the forcing scheme or the Navier-Stokes equation. The equation for the velocity at such points was written as:

$$w^j = (1 - \alpha_i) NS^j + \alpha_i u_i^j \quad (2.42)$$

where NS^j are the coefficients of the discretised Navier-Stokes equation for the cell, while $\alpha_i u_i^j$ corresponds to the velocity forcing, which ensured a no-slip velocity boundary condition. The parameter describing the magnitude of the forcing contribution was given by:

$$\begin{aligned} &= \frac{1}{x^2} + \frac{2}{y^2} && \text{in 2D} \\ &= \frac{1}{x^2} + \frac{2}{y^2} + \frac{3}{z^2} && \text{in 3D} \end{aligned} \quad (2.43)$$

with x, y, z being the grid spacing in x, y, z directions respectively and α_i the distances to interface in the i directions.

This approach ensured that there was no rapid change in the description of the cells as the body moves. The results presented in [76] show that the oscillations were in fact mostly suppressed, however some strange behaviour was observed in one of the investigated cases.

To summarise, there exist a consensus that the source of the spurious pressure oscillations lies in the fact, that as the body moves, some cells change their type. Also, the behaviour of the amplitude of fluctuations with the grid refinement level and time-step size is well described.

Various explanations and techniques to resolve the issue of pressure oscillations have been proposed. Although significant improvements have been made the problem has not been solved yet. Recent simulations performed during this research project shed a new light on the subject. These findings, along with their implications, are discussed in the chapters 3 and 5, in which the numerical method developed during the project is presented and validated.

2.4 Additional remarks

The current review focused on the way the forcing is applied at the Immersed Boundary, which is the most important feature of any IB method. Other features of the method are, however, also explored in the literature. Improving the technique adopted to describe the body surface is one of them. Generally, two approaches are at hand. One of the very popular ideas, is describing the body surface by means of a Level-Set method. Here, the interface is tracked by a Level-Set function which is calculated in the entire

fluid domain. A Level-Set function is defined as:

$$\begin{aligned}
 d(x^j, x_{IB}^j) & > 0 & x^j \text{ in the fluid region} \\
 d(x^j, x_{IB}^j) & = 0 & x^j \text{ on the interface} \\
 d(x^j, x_{IB}^j) & < 0 & x^j \text{ in the solid region}
 \end{aligned}
 \tag{2.44}$$

where $d(x^j, x_{IB}^j)$ is the distance from the point x^j to the nearest surface point x_{IB}^j . This method is applied by a number of researchers ([43], [53], [118]). The main advantage of the Level-Set method is its simplicity and ability to efficiently describe even complex bodies. On the other hand, time advancement of the Level-Set function may pose some problems. Also, the function resolution must be fine enough to allow for description of surfaces smaller than the fluid grid spacing.

Even if the Level-Set method is used, it has to be initialised by means of either an analytical function or a triangulated representation of the surface topology. In fact, using a triangulated surface representation is much more common due to its independence of the underlying grid refinement. A body of an arbitrary shape is given by a set of surface triangles with known coordinates and outward pointing normals. The main challenge here lies in the identification whether a given point falls inside or outside of the surface and determination of the body-fluid grid intersection points.

A novel parallel triangulation library MFTL has been developed during this research project to enable simulations with the Immersed Boundary Method. The library can describe particles of arbitrary shapes and allows for efficient calculation of the body-fluid interaction. The details of the MFTL are presented in chapter 4.

2.5 Summary

To conclude, there exists a number of the Immersed Boundary Method implementations and the technique is still under an extensive development. The potential of this technique has been recognized and it lies mainly in the ability to efficiently simulate flows with either moving or deforming bodies, although it can be also successfully applied to analysis of flows around bodies with complex shapes. Numerous advancements have been proposed for the improvement of the IBM and were discussed in this chapter.

The continuous forcing approach is well suited for simulations with deforming bodies, however it suffers from stability issues when applied to the solid particles. Also, the application of distribution functions prevents sharp representation of the surface boundaries in this technique.

Alternative method, the discrete forcing, eliminates the problems encountered by the continuous forcing approach by ensuring a sharp representation of the interface. There are three types of the discrete forcing techniques:

discrete direct forcing where the fluid cells outside the body are modified in order to enforce the no-slip velocity boundary condition at the IB surface

ghost cell method where velocities in the cells inside the IB are set, so that the interpolated velocity

at the IB interface matches the velocity of the body

cut-cell method where the momentum and continuity equations are solved only for the flow part of the fluid cells

From the above, only the *cut-cell* approach ensures a strict, global and local mass and momentum conservation. This approach is however computationally more expensive since it requires to implicitly calculate the velocity gradients and the pressures at the surface of the Immersed Boundary. It also requires special treatment of the small cells, where the fraction of the fluid is less than half.

On the other hand, the *ghost cell* method and the *discrete direct forcing* experience pressure oscillations in the flow when moving particles are simulated, what is one of the main issues with these IB implementations. The fluctuations are attributed to the lack of mass conservation in the vicinity of the simulated bodies.

The numerical method presented in this thesis aims at decreasing those oscillations, while maintaining the high quality of the solution at a low computational cost. The main idea behind the current technique is based on the concepts introduced in [82], *i.e.* a combination of a *ghost cell* method used to set the no-slip velocity boundary condition with a *cut-cell* technique adopted to the continuity equation in the vicinity of the particle. This allows to take advantage of both approaches and to minimize their negative features.

The *ghost cell* method is easy to implement, and unlike the *discrete direct forcing*, it does not affect the Navier-Stokes equations in the fluid cells directly, but it enforces the no-slip boundary condition by setting the velocities inside the simulated bodies. The main advantage of the *ghost cell* method in comparison to the *cut-cell* technique is the fact that it eliminates the computationally expensive and hard to implement necessity to implicitly calculate the velocity gradients and the pressures at the surface of the Immersed Boundary as well as the necessity of small cell treatment.

On the other hand, application of the *cut-cell* technique only to the continuity equation ensures that the mass flux is conserved both locally and globally. The *cut-cell* approach, when applied to the continuity equation only, is much easier to implement and does not suffer from its main limitations, *i.e.* the small cell constraint and the necessity to implicitly calculate the pressures and velocity gradients at the IB surfaces.

The rest of this thesis will concentrate on the explanation of the proposed numerical technique, its validation and application to study of flows past non-spherical particles.

3 Numerical method

This chapter describes the numerical methodology developed during the research project presented in this thesis. First, the approach used to simulate the single phase flow is outlined. Afterwards the various suggested Immersed Boundary Method implementations are discussed. The IB technique adopted in this thesis is based on a modified *ghost cell* approach combined with the *cut-cell* technique adopted to the continuity equation. The motivation for this approach is described in section 2.5.

3.1 Single phase flow

In the current research the flow equations are solved using MULTIFLOW, a curvilinear, multiblock, multiprocessor flow solver designed by the Berend van Wachem research group [90]. The solver uses a collocated grid arrangement, where both pressure and velocities are specified at the fluid cell centres. Additionally momentum weighted interpolation introduced by Rhie and Chow [98] is applied to discretise the continuity equation.

The fluid behaviour is described by the momentum (2.1) and continuity (2.2) equations. Discretisation of these equations results in a set of coupled linear equations solving the three components of velocity along with the pressure for each fluid cell in the computational domain. The solved system of equations has the following form:

$$\begin{aligned} u_1 &= RHS_{u_1} \\ u_2 &= RHS_{u_2} \\ u_3 &= RHS_{u_3} \\ p &= RHS_p \end{aligned} \quad (3.1)$$

The exact coefficients of the matrix will depend on the discretisation scheme outlined below.

3.1.1 Momentum equation discretisation

The discretised form of the momentum equation has the following structure:

$$\begin{aligned} \underbrace{\frac{V_P}{t} u_P^j}_{\text{transient}} + \underbrace{c_{(N+P)} u_{(N+P)}^j}_{\text{convection}} = \\ \underbrace{b_{(N+P)} p_{(N+P)}}_{\text{pressure gradient}} + \underbrace{t_{(N+P)} u_{(N+P)}^j}_{\text{viscous shear}} + \text{RHS discretisation terms} \end{aligned} \quad (3.2)$$

where u_P is the value of variable u at the current cell centre, while u_N is the value of u at the cell centre of the face neighbour of current cell. V_P is the cell volume, Δt is the time-step and u_P^{jO} is the j velocity component at the cell centre evaluated at the previous time-step. c , t and b are the discretisation coefficients for the convection, shear and pressure gradient terms respectively.

The above equation can be rearranged to obtain:

$$\begin{aligned} & \underbrace{\frac{V_P}{t}}_{\text{central coefficient}} + \underbrace{c_P}_{\text{convection coefficient}} \underbrace{t_P}_{\text{viscous coefficient}} u_P^j + \underbrace{c_N}_{\text{convection coefficient}} \underbrace{u_N^j}_{\text{viscous coefficient}} + \underbrace{t_N}_{\text{viscous coefficient}} \underbrace{u_N^j}_{\text{viscous coefficient}} + \underbrace{b_{(N+P)}}_{\text{pressure coefficient}} p_{N+P} \\ = & \underbrace{\frac{V_P}{t}}_{\text{RHS transient}} \underbrace{u_P^{jO}}_P + \text{RHS discretisation terms} \end{aligned} \quad (3.3)$$

The coefficients on the left hand side of the equation 3.3 depend on the velocities and pressures at the current cell centre (u_P) and its neighbours (u_N). The right hand side elements represent the explicit contributions from the previous time-steps. The approach applied for evaluation of the discretisation coefficients is described below.

Convection terms

The discretisation coefficients for the convection terms of the momentum equation, $\frac{1}{x^i} u^i u^j$, are found by evaluating the volume integral by means of Gauss law:

$$\int_V \frac{1}{x^i} u^i u^j dV = \sum_A u^i u^j n^i dA = \sum_f u_f^j u_f^i a_f^i \quad (3.4)$$

where a_f^i is the outward face normal vector ($a_f^i = A_f n^i$; A_f - face area, n^i - unit normal face vector). Equation 3.4 states that the volume integral of the convective term can be approximated by a sum of the fluxes through the cell faces f , multiplied by the flow velocity. When determining the convective fluxes, the values must be calculated exactly at the center of the cell face and a linear variation of the velocity over the cell volume is assumed. The magnitude of the convection coefficient is simply the mass flow through the face:

$$M_F = \sum_f u_f^i a_f^i \quad (3.5)$$

Several numerical schemes to calculate the values of the convective terms at the cell faces are available, with upwind and central schemes being among the most established ones.

The **central** scheme calculates the value of the property at the face by linear interpolation between two neighbouring points. Depending on desired accuracy, the value of the variable at the face centre can be approximated by computing the variable at different points e , e_e , e_e , as illustrated in Fig. 3.1. The resulting coefficients, evaluated at different points are given in Table 3.1, where the interpolation

coefficients l_P and l_N are:

$$l_P = \frac{x_f^i x_P^i n^i}{x_f^i x_P^i + x_f^i x_N^i n^i} \quad (3.6)$$

$$l_N = \frac{x_f^i x_N^i n^i}{x_f^i x_P^i + x_f^i x_N^i n^i} \quad (3.7)$$

x_P^i, x_N^i, x_f^i are the position vectors of cell P , its neighbour N and the face f respectively.

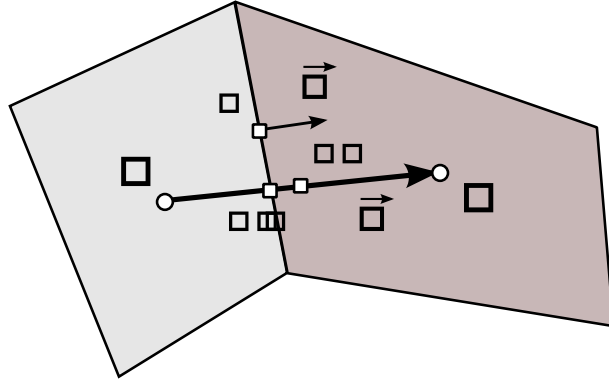


Figure 3.1: Points available for evaluation of the variable value at the cell face centre e . e is the point halfway between P and E , e is the point on the intersection of q and the face centre. n is the face normal.

In general, the central scheme is second-order accurate, however it tends to generate wiggles for flows with large time-steps ($CFL > 1$). **Upwind** scheme was designed to overcome the time-step limitation of the central scheme. Although it is only first-order accurate, the upwind scheme is unconditionally stable and allows for simulations with large time-steps. In principle, the upwind scheme specifies whether the coefficient will be applied to the diagonal term or to the neighbouring cell depending on the sign of the mass flux M_F (Table 3.1). There are other discretisation schemes ([24]), but they are currently not implemented in the MULTIFLOW IBM solver, therefore they are not discussed here.

Viscous stress discretisation

The Gauss law applied to the viscous term in the momentum equation results in:

$$\int_V \frac{\partial}{\partial x^i} \tau^{ij} dV = \int_A \tau^{ij} n^i dA = \int_f \tau_f^{ij} A_f n_f^i \quad (3.8)$$

where A_f is the face area.

Using the definition of τ^{ij} given in equation (2.4) the discretised stress component at the face results

Table 3.1: Options for the convective terms discretisation.

Scheme	Central			Upwind
	e	e	e	
c_P	$\frac{1}{2}M_f$	$l_P M_f$	$\frac{1}{2}M_f$	M_f if $M_f \geq 0$
c_N	$\frac{1}{2}M_f$	$l_N M_f$	$\frac{1}{2}M_f$	M_f if $M_f < 0$
RHS	0	0	$M_f \frac{\overline{u_f^i}}{x_f^i}$	0

in a form:

$$A_f n_f^i = A_f n_f^i \frac{u_f^j}{x_f^i} + \frac{u_f^i}{x_f^j} \quad (3.9)$$

Usually the first term of the right-hand side is dominating element, therefore it requires the most attention. Since it is important to use the correct face normal when calculating the $\frac{u}{x^i} n^i$ gradient, a special discretisation is performed:

$$\frac{u_f^j}{x_f^i} n_f^i = \frac{u_f^j}{x_f^i} q_f^i + \frac{u_f^j}{x_f^i} n_f^i \quad (3.10)$$

where q^i is the vector between the cell centres of current and neighbouring cells (see Fig. 3.1). Due to stability reasons, a scaling parameter $\frac{u_f^j}{x_f^i} n_f^i$ is introduced, what leads to an equation in taking a following form:

$$\frac{u_f^j}{x_f^i} n_f^i = \frac{u_N^j}{x_f^i} + \frac{u_P^j}{x_f^i} n_f^i \quad (3.11)$$

The second term of the viscous stress, given in equation 3.9, can be implemented in an explicit way using the velocity gradients from the previous time-step:

$$A_f n_f^i \frac{u_f^i}{x_f^j} = A_f n_f^i \frac{\overline{u_f^i}}{x_f^j} \quad (3.12)$$

Equations 3.11 and 3.12 will result in following discretisation coefficients for the viscous stress terms:

$$t_P^j = + \frac{A_f}{x_f^j} \quad (3.13)$$

$$t_N^j = - \frac{A_f}{x_f^j} \quad (3.14)$$

$$RHS_f^j = A_f \overline{\frac{u^j}{x^i}} n^i \frac{q^i}{x^j} + \frac{u^i}{x^j} n^i \quad (\text{per face}) \quad (3.15)$$

Pressure gradient discretisation

Similarly as above, the Gauss law applied to the pressure gradient results in:

$$\int_V \frac{p}{x^i} dV = \int_A p n_f^i dA - p_f a_f^i \quad (3.16)$$

Pressure at the cell face can be calculated using central scheme analogously as in the case of the convection terms. In the current approach pressure is calculated at point e (Fig. 3.1), with l_P and l_N coefficients calculated from expressions 3.6 and 3.7. Hence, the pressure gradient discretisation coefficient read:

$$b_P^j = l_P a_f^j \quad (3.17)$$

$$b_N^j = l_N a_f^j \quad (3.18)$$

This completes the discretisation of the momentum equation.

3.1.2 Continuity equation discretisation

The continuity equation can be written as:

$$\frac{\partial u^i}{\partial x^i} = 0 \quad (3.19)$$

In the discretised form, after applying the Gauss law one obtains:

$$\int_f u_f^i a_f^i = \int_f u_f^i a_f^i - M_f = 0 \quad (3.20)$$

Equation 3.20 is simply the mass conservation principle, stating that the sum of the mass fluxes over the cell surface must be equal to zero. M_f is the same mass flow through the face, which is used in the convective terms of the momentum equation.

The face normal velocity needs to be expressed by means of velocities and pressures at both current and the neighbouring nodes. This is proposed by the momentum weighted interpolation [98] which is described in this section. The continuity equation is hence discretised as follows:

$$g_P P_P + \sum_N g_N P_N + \sum_{N+P} h_{(N+P)}^i u_{(N+P)}^i = RHS \quad (3.21)$$

where the pressure coefficient g_P lies on the diagonal of the matrix and needs to be non-zero.

Momentum weighted interpolation

The discretised momentum equation 3.3 can be rewritten in a following form:

$$\frac{V_P}{t} + a_P u_P^j = \frac{a_N u_N^j}{N} + \frac{b_{(N+P)} p_{(N+P)}}{N+P} + \frac{V_P}{t} u_P^{jO} \quad (3.22)$$

where $a_k = c_k + t_k$. Dividing the above equation by a_P and defining:

$$c = \frac{V_P}{t} \quad (3.23)$$

$$d = \frac{V_P}{a_P} \quad (3.24)$$

$$u^j = \frac{a_N u_N^j}{a_P} \quad (3.25)$$

$$\frac{b_{(N+P)} p_{(N+P)}}{N+P} = V_P \frac{p}{x_j} \quad (3.26)$$

results in:

$$(1 + c_P d_P) u_P^j = u_P^j + d_P \frac{p}{x_j} + c_P d_P u_P^{jO} \quad (3.27)$$

Analogous equation can be written for any point. For example for the point e in the Fig. 3.1. Expression 3.27 can also be used to evaluate u_P^j as:

$$u_P^j = (1 + c_P d_P) u_P^j + d_P \frac{p}{x_j} + c_P d_P u_P^{jO} \quad (3.28)$$

The value of u^j at point e can be calculated as the linear interpolation between points P and E and reads:

$$u_e^j = \frac{1}{2} (u_P^j + u_E^j) \quad (3.29)$$

$$\begin{aligned} &= \frac{1}{2} (1 + c_P d_P) u_P^j + d_P \frac{p}{x_j} + c_P d_P u_P^{jO} \\ &+ \frac{1}{2} (1 + c_E d_E) u_E^j + d_E \frac{p}{x_j} + c_E d_E u_E^{jO} \end{aligned} \quad (3.30)$$

It is reasonable to assume that the coefficients at points P , E , e have the same values, and to divide the expression 3.27 by $(1 + c d)$. Substituting the above equation to the equation 3.27 for u_e^j and performing some simplifications leads to:

$$\begin{aligned}
 u_e^j &= \frac{u_P^j + u_E^j}{2} \\
 &\quad \frac{d_e}{1 + c_e d_e} \frac{p}{x_j} \left(\frac{1}{2} \frac{p}{x_j} \right)_P \left(\frac{1}{2} \frac{p}{x_j} \right)_E \\
 &\quad + \frac{c_e}{1 + c_e d_e} u_e^{jO} \left(\frac{1}{2} u_P^{jO} \right) \left(\frac{1}{2} u_E^{jO} \right)
 \end{aligned} \tag{3.31}$$

The form of the above equation can be further simplified by introduction of averages:

$$\bar{u}_e^j = \frac{P + E}{2} \tag{3.32}$$

$$d_e = \overline{d_e} = \text{avg} \left[\frac{d_e^{(x)}}{1 + c_e d_e^{(x)}} \frac{d_e^{(y)}}{1 + c_e d_e^{(y)}} \frac{d_e^{(z)}}{1 + c_e d_e^{(z)}} \right] \tag{3.33}$$

This gives the final form for the velocity at point e :

$$u_e^j = \bar{u}_e^j \frac{d_e}{d_f} \frac{p}{x_j} \left(\frac{p}{x_j} \right)_e + c_e u_e^{jO} \left(u_e^{jO} \right) \tag{3.34}$$

The pressure terms in the equation 3.34 play a role of a 3rd order filter, *i.e.* their value is non-zero only if the pressure profile is of a cubic or higher order. The “old” velocity terms ensure that the steady state solution is time-step independent. In practice, however, their influence is small and are very often neglected.

Since the continuity equation is evaluated at cell face centres, u_e has to be found instead of u_e . Similarly as for convection or pressure terms in the momentum equations various versions of central scheme can be applied. The pressure will be however still calculated at point e . This results in:

$$u_e^j = \bar{u}_e^j \frac{d_f}{d_e} \frac{p}{x_j} \left(\frac{p}{x_j} \right)_e + c_e u_e^{jO} \left(u_e^{jO} \right) \tag{3.35}$$

On an arbitrary grid the cell face normal does not necessarily point in the same direction as the line connecting two neighbouring cell centres. Therefore additional correction term might need to be employed (as in the viscous stress discretisation).

Also, since the continuity equation solves only the balance of fluxes through the faces, it is more appropriate to evaluate $u_e^j a_e^j$ what leads to:

$$u_e^j a_e^j = \bar{u}_e^j a_e^j \frac{d_f a_e^j}{d_e a_e^j} \frac{p}{x_j} \left(\frac{p}{x_j} \right)_e + c_e a_e^j u_e^{jO} \left(u_e^{jO} \right) \tag{3.36}$$

The pressure term at the cell face can be evaluated using direct pressure gradient:

$$\frac{p}{x_j} = \frac{p_E - p_P}{e} \quad (3.37)$$

while the pressure gradient in each cell can be obtained in the same way as in the momentum equation:

$$\int_V \frac{p}{x^i} dV = \int_A p n_f^i dA \quad (l_{PPP} + l_{NPN}) a_f^i \quad (3.38)$$

This implies that not only the current cell P neighbours are used in the discretised linear equation, but their neighbours need to be taken into account as well. Finally, the resulting continuity discretisation coefficients are:

$$h_P^i = l_N a_f^i \quad (3.39)$$

$$h_N^i = l_N a_f^i \quad (3.40)$$

$$g_P = d_f \frac{A_f}{d_f} \quad \text{large pressure stencil} \quad (3.41)$$

$$g_N = d_f \frac{A_f}{d_f} + \text{large pressure stencil} \quad (3.42)$$

$$RHS_f = d_f A_f \frac{p}{x^i} n_f^i \quad q^i \quad (\text{per face}) \quad (3.43)$$

3.2 Immersed Boundary Method implementation

As shown in chapter 2, the solution of an incompressible flow past a number of particles requires solving the momentum (2.1) and continuity (2.2) equations for the fluid with the no-slip velocity boundary condition (2.3) applied at the particle surfaces.

The discretisation of the flow equations has been discussed in the previous section of this chapter. Here, the implementation of the particle-fluid interaction is presented. This involves enforcement of appropriate boundary conditions on the particle-fluid interfaces, and calculation of the forces experienced by the bodies.

A number of implementations of the Immersed Boundary Method has been proposed, a detailed review of which is presented in chapter 2. The IB method presented in this thesis is based on the ideas presented in [82]; a combination of a *ghost cell* approach, where the fluid velocities are set inside the surface in order to satisfy the boundary conditions at the interface with additional modifications to the continuity equation. The presence of the IB does not modify the flow region directly, as the Navier-Stokes equations are solved for all fluid cells.

A successful implementation of the *ghost cell* Immersed Boundary Method needs to satisfy the following conditions:

Accurate boundary conditions enforcement on the fluid-body interface;

Ensuring that there is no mass flux through the interface, and that no mass is generated in the flow;

Calculating the accurate forces acting on the IB from the resolved flow field;

A detailed description of the adopted approach for each of those aspects is given below.

3.2.1 The simulation procedure

A flowchart illustrating the numerical procedure adopted for the True Direct Numerical Simulations of flows with solid particles during the current research is shown in the Fig. 3.2. At the beginning of the simulation a triangulated grid is generated for each of the simulated bodies. Also, the auxiliary data, needed for the force calculation routines is prepared. Each body has its own surface mesh and therefore its motion and interaction with the surrounding fluid is calculated independently from other particles.

Once the surface meshes are generated, the type of every cell in the fluid domain has to be determined. There are 4 cell types, depending on the cell centre location with respect to the Immersed Boundary:

type **0** - fluid cell

type **2** - fluid cell with a neighbouring cells inside the body

type **3 + 2 NIB** - solid cell inside the IB

type **4 + 2 NIB** - *ghost cell* with a fluid neighbour

$NIB = 0, 1, 2, \dots$ is the particle identifier. The type of the cells inside the body depends on the IB number, e.g. type **6** cell is a *ghost node* of the second particle. An illustration of the cell tagging is shown in Fig. 3.3.

Optionally the properties of the *cut-cells*, such as the accurate flow face areas, modified face centres and the volume of the fluid inside the *cut-cells* are evaluated, see chapter 4 for the comprehensive description of the process.

Next, the *ghost nodes* are projected through the surface to generate imaginary points, used for imposing the boundary conditions at the IB surfaces. Since the flow variables are interpolated to the imaginary points in the matrix set-up stage, the respective interpolation coefficients are calculated and stored for further use.

Afterwards, the discretisation coefficients of the flow equations are evaluated. During this step, the flow equations are modified in order to account for the IB presence. This involves setting the velocity of the *ghost cells* along with modifications to the continuity equation in the vicinity of the particle. Details of these steps are given in the next sections of this chapter.

The matrix coefficients are normalised prior to the solution. Once all the coefficients of the main matrix are known, the flow equations are solved by the PETSc KSP solver [4]. The resolved flow variables are then used to calculate the mass fluxes through the cell faces. These mass fluxes are used in the convection term discretisation in the subsequent time-step.

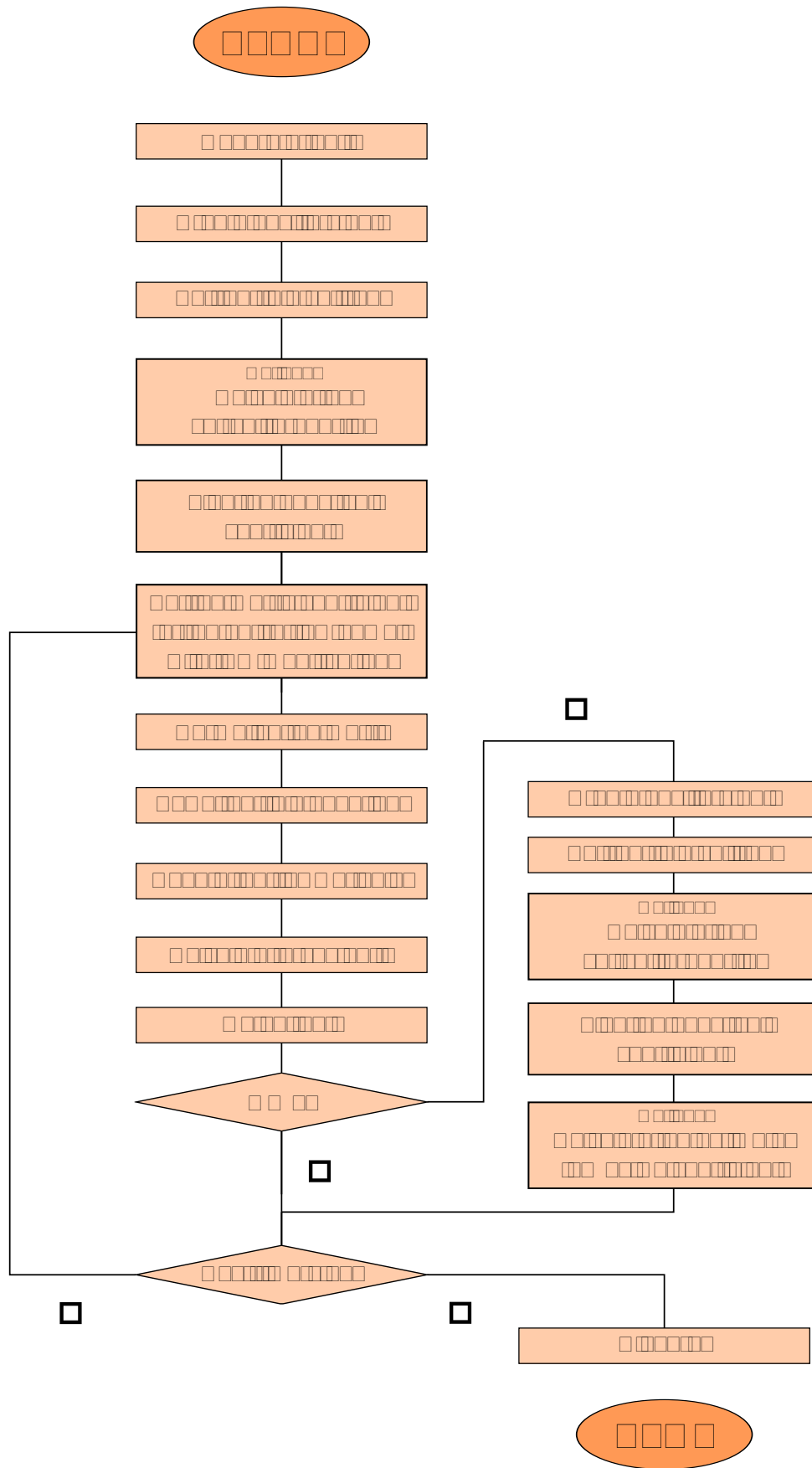


Figure 3.2: IB simulation procedure flowchart.

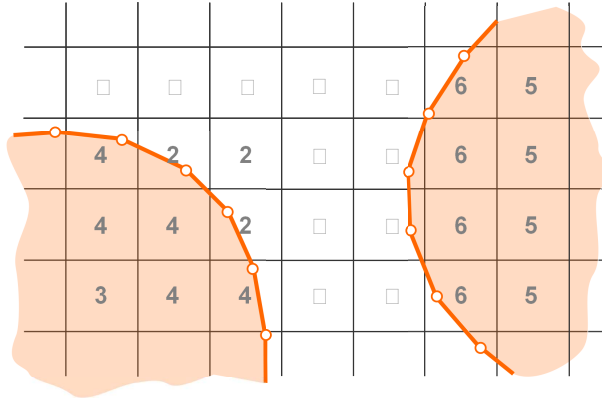


Figure 3.3: Illustration of cell tagging for the current IBM. *Ghost cells* have tags **4** and **6** respectively.

The forces and torques acting on the particle are evaluated as a sum of pressure and viscous components, of the updated flow field, as described in section 3.2.4.

If the particle motion is enabled, its acceleration is calculated. Both linear and angular acceleration are used to update the IB velocities. The motion of the particle is then computed by means of the following equations:

$$x^j = u_p^j t + \frac{1}{2} a_p^j (t)^2 \quad (3.44)$$

$$\dot{\theta} = \dot{\theta}_p t \quad (3.45)$$

where x^j is the translation step, $\dot{\theta}^j$ is the rotational step, while u_p^j is the particle linear velocity, $\dot{\theta}_p^j$ the angular velocity and a_p^j is the linear acceleration of the particle.

In the simulations with moving bodies, the cell tagging has to be recomputed each time particles move. Also, the auxiliary data and the imaginary points have to be regenerated before advancing to the next time-step. If a simulation of a flow past a stationary body is performed, the cell tagging remains the same, therefore the program can advance to the next time-step directly.

If the *cut-cell* method is used to modify the continuity equation, the mass fluxes and the velocities in the *fresh* cells need to be recomputed after the body is moved. This step involves solving the flow equations, with the IB at its new position, similarly as during the main flow solution step. The main difference is the fact that here, only the mass fluxes along with the *fresh* cell velocities are updated, while the flow variables (velocity and pressure) remain unchanged.

Once all the time-steps are computed the used memory is released and the simulation is concluded.

3.2.2 Enforcing the velocity boundary conditions

In the *ghost cell* method, used as the basis of the current IBM implementation, the no-slip velocity boundary condition is enforced by setting the velocities of the *ghost nodes* in a way that ensures that the fluid velocity at the IB surface matches the particle velocity at the surface.

For every *ghost cell* an imaginary point, IP, is created by projecting the cell centre along the normal of

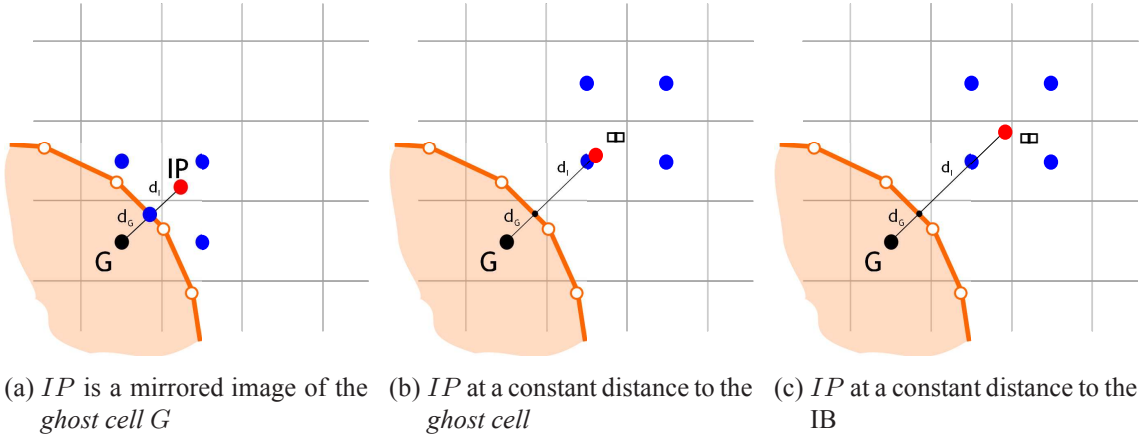


Figure 3.4: Imaginary point, IP , used for the boundary condition enforcement (red), along with the corresponding interpolation stencil (blue).

the closest surface triangle as illustrated in Fig. 3.4. Three positioning variants are considered:

- a) IP as a mirrored projection of G - $d_I = d_G$
- b) IP at a constant distance to the ghost cell
- c) IP at a constant distance to the Immersed Boundary

The velocity at the imaginary point can be implicitly calculated by a tri-linear interpolation from the surrounding fluid cells (blue circles in Fig. 3.4). If one of the interpolation points coincides with the current ghost node the known value at the surface can be used instead. It has been observed, however, that use of the surface point has little impact on the results.

The velocity at the imaginary point is calculated as:

$$w_{IP}^j = \sum_i \alpha_i w_i^j + \beta w_{IB}^j \quad (3.46)$$

where α_i are the tri-linear interpolation coefficients (see [82] for the detailed derivation). If the surface point is not used, the value of β is set to zero.

The interpolated velocity at the IB surface, evaluated using the ghost and imaginary point, is given as follows:

$$w_{IB}^j = \frac{1}{d_I + d_G} (d_I w_G^j + d_G w_{IP}^j) \quad (3.47)$$

The equation 3.47 can be transformed to obtain the expression of for the velocity at the ghost cell:

$$\begin{aligned} d_I w_G^j &= (d_I + d_G) w_{IB}^j - d_G w_{IP}^j \\ &= (d_I + d_G) w_{IB}^j - d_G \sum_i \alpha_i w_i^j - \beta d_G w_{IB}^j \end{aligned} \quad (3.48)$$

If the point is mirrored through the surface (Fig. 3.4a), the equation 3.48 becomes:

$$u_G^j = 2u_{IB}^j - u_I^j \quad (3.49)$$

The equation 3.48 is used to specify the velocity boundary condition at the surface. The u_{IP}^j velocity is calculated in an implicit way, *i.e.* the coefficients a_i are placed directly in the main matrix, containing the coefficients obtained by linearising the Navier-Stokes equations.

The imaginary points along with the interpolation coefficients have to be re-computed every time the IB moves.

3.2.3 Additional modifications to the flow equations

The Immersed Boundary Method, used to satisfy the no-slip velocity boundary condition, modifies the momentum equation in the *ghost cells*. As a result, a non-physical flow inside the particle is enforced.

The continuity equation will therefore adjust the pressure to appropriate values, ensuring that the mass conservation principle is not violated. This may lead to unrealistic flow behaviour inside the IB, which should not have any effect on the outside flow. Nevertheless, due to the misalignment of the fluid and the particle grids, a non-zero mass flux through the boundary can be sometimes observed.

If the aforementioned observations are taken into account, it seems reasonable to include the IB influence not only in the momentum but in the continuity equation as well.

The simulations performed during the course of the current research project show that the impact of modifying the continuity equation is very small in the case of flow past a stationary particle. On the other hand, when moving particles are concerned, the treatment of the continuity equation can have a significant influence on the magnitude of the spurious pressure oscillations observed in the IBM.

In fact, some researchers investigating the issue of the non-physical pressure fluctuations, link them directly to the local violation of the continuity equation [103]. Also, it has been shown that preventing the mass flux through the surface leads to a significant reduction of the pressure oscillations [70, 71, 103].

In this thesis various strategies for enforcing the local satisfaction of the continuity equation are explored, and the performance of these approaches is evaluated. The analysed techniques involve:

- Limiting the IB influence only to the modification of the momentum equation in the *ghost cells*

- Setting the velocity of cells far inside the surface to the velocity of the body

- Applying a zero normal pressure gradient boundary condition at the surface of the particle

- Excluding the velocities inside the IB from the continuity equation

- Excluding the flow inside the IB from the continuity equation

- Using a *cut-cell* approach to solve the continuity equation in the vicinity of the IB along with the convective coefficient recalculation

In all the techniques mentioned above, the *ghost cell* velocity is set by the Immersed Boundary condition described in section 3.2.2. The additional modifications applied in each of the implementations are summarised below, while the assessment of the behaviour of the investigated methods is performed in chapter 5.

Setting the velocity inside the body

One possible strategy to limit the non-physical flow inside the body is to set the velocities of the cells far inside the body (type 3 and 5 in Fig. 3.3) to the velocity of the solid body motion. This leads to:

$$u_{IN} = u_{IB} + x_{IN} \times \omega_{IB} \quad (3.50)$$

where u_{IB} and ω_{IB} are the linear and angular velocities of the IB respectively. x_{IN} is the position of the cell centre with respect to the body centre.

Setting the velocity in the entire region inside the body will have a strong effect on the pressure of the fluid, as it has to adjust accordingly, in order for the continuity equation to be satisfied.

Zero normal pressure gradient at the surface

A common strategy for the treatment of the continuity equation is to specify a zero normal pressure gradient at the IB surface. In such case, the continuity equation for the *ghost cells* has the following form:

$$p_G = p_{IP} = \sum_i \alpha_i p_i \quad (3.51)$$

α_i are the same interpolation coefficients used for calculating the velocity of the imaginary point used for setting the no-slip velocity boundary condition, while p_i are the pressures of the fluid cells used for interpolation.

Since there is no pressure gradient in the direction normal to the surface and the no-slip velocity boundary condition is applied, there should be no mass flux through the interface. The explicit convective coefficients in the momentum equations are therefore set to zero for faces between *ghost* and fluid cells.

Excluding the flow inside the body from the continuity equation

Mark and van Wachem [82] made an argument, that since an artificial flow is generated inside the body by the IB condition, it should be excluded from the continuity equation to prevent mass flow through the surface. The reasoning behind this approach is that the continuity should be solved only in the fluid domain, while the flow inside the body should have no effect on the behaviour of the fluid. This leads to a stair-step representation of the body in the continuity equation as illustrated in Fig. 3.5 (the hatched area corresponds to the cells excluded from the continuity equation).

Two variants of this approach are available. In the first technique, only the fluid velocities are excluded from the continuity, while the pressure correction term remains active inside the body, as it acts only as

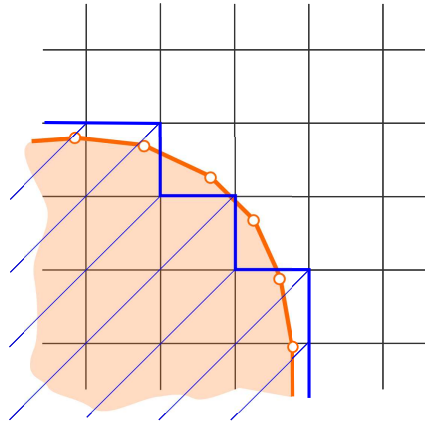


Figure 3.5: Excluding the flow inside the body from the continuity equation. A stair-step representation of IB is achieved in the continuity equation.

a third-order filter. Excluding the velocities from the continuity leads to the following expression for the mass flux through the face (e.g. for east face):

$$M_f = d_f a_e^j \left(\frac{p}{x_j} \right)_e - \frac{p}{x_j} + a_e^j u_{IB}^j \quad (3.52)$$

u_{IB}^j is the known velocity of the Immersed Boundary that becomes an explicit right-hand side term of the continuity equation. The continuity equation, with flow velocities set to zero has to be solved in the cells inside the body as well.

Alternatively, the pressure inside the body can also be excluded from the consideration. In such case, the equation for the mass flux at the cell becomes:

$$M_f = d_f a_e^j \left(\frac{p}{x_j} \right)_e - \frac{p}{x_j} + a_e^j u_{IB}^j \quad (3.53)$$

The pressures in the *ghost cells* are still used in this implementation, however the pressures of the remaining cells, far inside the IB have no impact on the flow and can be set to an arbitrary value.

Solving the continuity using the *cut-cell* approach

In practice, the only way to ensure a strict mass conservation, is to solve the continuity equation for the fluid in the entire domain, with the flow inside the body removed from consideration. This requires solving the mass conservation equation for the fluid in the exterior body cells, along with evaluation of fluxes through the cell face areas available to flow in the *ghost cells*.

Fig. 3.6 illustrates the procedure for calculation of the flux through the face cut by the IB. First, the exact topology of every cell has to be determined. This involves not only calculation of the fluid face areas and face centres, but also the exact areas of the surface triangles in each cell along with the volume

occupied by the fluid. The exact procedure adopted for the topology calculation is outlined in chapter 4.

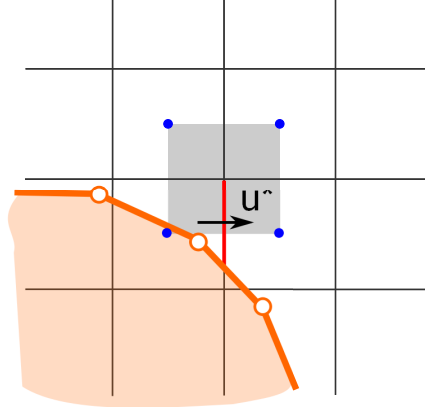


Figure 3.6: Illustration of the *cut-cell* approach to evaluate the flow velocity u . A 4-point stencil (8-point in 3D), blue dots, is used to calculate the u interpolation coefficients. The choice of points depends on the location of the face centre.

The mass flux through a cut-face e has the following form:

$$M_e = u_e^j a_e^j - d_f a_e^j \frac{p}{x_j} - \frac{p}{x_j} \quad (3.54)$$

here, the u_e^j and a_e^j are the velocity at the centre of the cut-face and the modified flow area at the face respectively. Note that the pressure terms remain unaffected by the *cut-cell* approach, as they play only the role of a third-order filter. The velocities of the cells far inside the body have to remain zero however, as they are not supposed to contribute to the solution of the pressure field.

The velocities at the centres of the cut faces are evaluated using a tri-linear interpolation based on 8-point stencil (4 in 2D), illustrated as the blue circles in Fig. 3.6. The choice of the stencil is unique and depends on the position of the face centre of the cut-face. Exact calculation of the face velocities is essential for the accuracy of the method. It is of major importance, especially when the flow between two neighbouring *ghost cells* is considered. The velocities in those cells, if considered alone, are not physical, however when used for interpolation they enhance the quality of the velocity evaluation. Using a tri-linear interpolation for all cut-faces ensures a smooth change of the velocity as the body moves.

Volume occupied by the fluid in the *ghost cells* can be very small, what usually causes significant problems for the *cut-cell* methods. Due to several factors, this issue is however avoided in the current implementation. Firstly, since the flow equations are normalised prior to the solution, the small cells will not introduce additional stiffness to the equation as their coefficients will have the same order of magnitude as those of the non-cut cells. Also, the CFL constraint is relaxed as well, because it applies only to the momentum equation.

When the approach outlined above is applied, the continuity equation for the *cut-cells* has the following

form:

$$M_f = \sum_{IB} a_{IB}^j u_{IB}^j \quad (3.55)$$

The right hand side of equation 3.55 is the contribution of the body motion, *i.e.* sum of the mass flux induced by the velocity of every IB element present in the cell.

Solving the continuity equation with the *cut-cell* approach has also significant implications on the solution of the momentum equation, especially in terms of the convection term discretisation. The modified momentum equation of the fluid cells needs also to take into account the mass flux induced by the body motion, therefore the convective term discretisation is calculated as:

$$\int_V \frac{1}{x^i} u^i u^j dV = \int_f u_f^j u_f^i a_f^i + \sum_{IB} a_{IB}^i u_{IB}^i u_{IB}^j \quad (3.56)$$

Additionally, it was found that the *cut-cell* approach is more sensitive to the accuracy of the convective coefficient prediction than the other methods discussed in this section. Inaccurate prediction of the mass flux through face associated with the motion of the body may lead to a wrong flow solution, therefore a more precise estimation of the convective terms in the momentum equation is needed.

It was proposed to perform additional flow field calculation sub-step during the same time-step, after the body is moved. This operation is performed in order to obtain a more accurate prediction of the mass fluxes through the cell faces in the vicinity of the body, taking into account the new position of the IB. The additional flow calculation is also used to estimate the velocities in the *fresh* cells. Unlike in the case of the main flow solution step, here the flow variables are not updated but only used to estimate the respective mass fluxes, so that the explicit convective term coefficients in the momentum equation of the next time-step take into account change in the mass flux associated with the new body location.

Summarising the *cut-cell* method developed during the course of the research project described in this thesis, along with setting the velocities in the *ghost* cells, involves three modifications to the flow equations. Firstly, it modifies the continuity equation to take into account the accurate mass fluxes through the cell faces and the mass flux associated with the motion of the IB. Additionally the IB influence is included in the discretisation of the convective terms in the momentum equation. Finally the *cut-cell* method requires the convective terms to be recalculated for the simulation of moving bodies.

The performance and the accuracy of the methods described above is analysed on a number of cases in chapter 5.

3.2.4 Calculation of the forces on the IB

Contrary to other Immersed Boundary Methods, where the forces are calculated during the process of imposing the boundary conditions, in the *ghost cell* method, the forces have to be calculated from the resolved flow field. This requires performing additional step after the Navier-Stokes equations have been solved.

The force calculation technique will use the flow field variables to evaluate the momentum exchange

between the fluid and the particle. Two strategies for this operation are investigated in this thesis. In the first method, the force on the particle is calculated by applying the momentum conservation principle to a surface around the body. In the second approach, on the other hand, the flow variables are extrapolated to the surface and the total force is calculated as the sum of force acting on each of the IB surface triangles.

Imaginary surface around the IB

One possible strategy for calculating the force acting on the particle is to compute the momentum transfer through an arbitrary, imaginary surface around the particle. This is a very common approach used to evaluate the momentum loss experienced by the flow past an arbitrary object. Derivation of this technique can be found in most fluid mechanics textbooks (e.g. [63]).

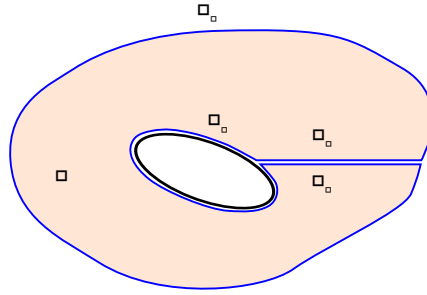


Figure 3.7: Illustration of the imaginary surface S around the particle used for the force calculation.

In case of any flow past a body, an imaginary closed surface S can be created as shown in Fig. 3.7. Surface S is formed by 4 segments: S_1 is the outer boundary, S_2 is closely fitted to the body, while S_3 and S_4 are identical surfaces with opposite normal directions. Assuming that no body forces are present, the momentum conservation principle for the volume encompassed by the surface S is given by:

$$\int_V \frac{du^j}{dt} dV + \int_S u^j u^i n^i dA + \int_S p n^j dA - \int_S u^i n^i dA = 0 \quad (3.57)$$

Since the segment S_2 is closely fitted to the body the momentum flux through this element of the surface is equivalent to the force acting on the body. Moreover, as the elements S_3 and S_4 are separated by an infinitesimally small distance, the momentum transfer through both segments cancels out. Hence the force F^j acting on the particle becomes:

$$F^j = \int_V \frac{du^j}{dt} dV - \int_{S_1} u^j u^i n^i dA - \int_{S_1} p n^j dA + \int_{S_1} u^i n^i dA \quad (3.58)$$

Usually the surface S_1 is assumed to be in the region of the undisturbed flow. If this assumption is considered the equation 3.58 can be further simplified. The streamlines in the undisturbed flow field are parallel, hence the shearing term can be neglected. Also, the pressure on the surface is equal to p_∞ , and its integral is zero, if the surface S_1 is stretched far enough from the body. Finally, if the steady state is considered the volume integral term is also zero. With aforementioned assumptions the force on the

body can be calculated as:

$$F^j = \int_{S_1} u^j u^i n^i dA \quad (3.59)$$

In the current research, a modified version of the technique presented above is applied. Instead of stretching the surface S_1 far from the body, it is assumed to lie in the close vicinity of the particle. It is formed by the cell faces lying between cells of type **0** and **2** (Fig. 3.3). In this case, the complete version of discretised equation 3.58 needs to be solved. Therefore the force on the particle is the sum of the momentum fluxes through every cell face forming the surface S_1 :

$$F^j = \int_c \frac{du^j}{dt} dV_c + \int_f u_f^j u_f^i n_f^i dA_f - \int_f p_f n_f^j dA_f + \int_f \left(\frac{u^j}{x^i} + \frac{u^i}{x^j} \right) n_f^i dA_f \quad (3.60)$$

The volumetric term in the equation 3.60 is calculated by adding the contribution of all the fluid cells in the close vicinity of the IB (type **2**). Other terms have to be evaluated at a specific cell faces. The velocities, pressures, and normal velocity gradients are calculated using the same discretisation scheme as in the case of the single-phase flow discretisation process described in section 3.1.

Calculation of the cross-terms in the velocity gradient tensor is performed using a Taylor series approximation in order to avoid the influence from the interior of the body:

$$u_f^j = u_c^j + \frac{u^j}{x^i} r^j \quad (3.61)$$

where r^j is the distance from the centre of the cell c to the centre of the considered face. The exact gradient is found using a least-squares fit of the above equation evaluated for all the fluid cells surrounding the particular cell face.

The torque acting on the particle can be found as sum of the cross products of the position vector of the relevant face with the force acting on that face evaluated by means of equation 3.60.

The main advantage of adopting the imaginary surface for the force calculation is the fact that it has the same accuracy level as the resolved flow field. The values of the flow variables, except the cross-terms of the velocity gradient, are obtained with the same discretisation technique, as the one applied to solve the flow equations. The method is therefore very accurate in predicting the forces for the cases of flows past stationary bodies.

On the other hand, the presence of volumetric term introduces additional challenges when moving bodies are considered. If the motion of the particle is unsteady the volumetric term will have a strong contribution to the total momentum transfer. Also the total volume of the type **2** cells in the surface does not have to match the volume of the fluid in the surface.

Additionally, the method has limited applicability in cases when the imaginary surfaces around two particles overlap. For example, a special treatment would be required for calculation of particle collision forces. It is necessary therefore to develop a method using the triangulated particle surface for the force calculation in such circumstances.

Force calculated directly on the IB surface

The total force acting on the Immersed Boundary in the j direction can be found by considering the sum of forces acting on each of the surface elements:

$$F^j = \int_{IB} (p_{ij} + \tau_{ij}) n^j dA = \int_{IB} p_{ij} + \frac{u^j}{x^i} + \frac{u^i}{x^j} n^j dA \quad (3.62)$$

where δ_{ij} is the Kronecker delta - a unit tensor, with 1 on the diagonal and 0 elsewhere.

The force is therefore a sum of the pressure and viscous forces acting on each of the particle surface elements. The surface components need to be integrated over the entire IB in order to obtain the total force acting on the body.

Neither the pressure value nor the velocity gradients are explicitly known at the IB surface, therefore they have to be determined from the resolved flow field using extrapolation.

For every surface triangle three equally spaced, auxiliary points are generated in the normal direction of the triangle as depicted in Fig. 3.8. The flow variables can be then interpolated to those points from the surrounding fluid cells using a tri-linear interpolation.

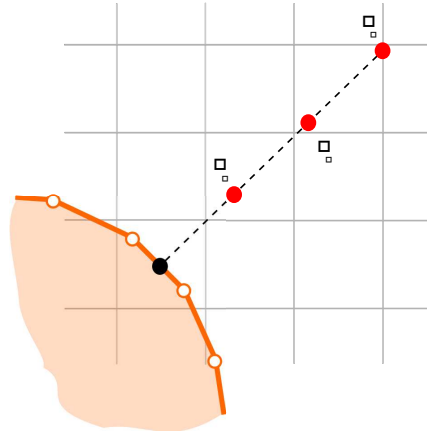


Figure 3.8: Position of auxiliary points P_i used to extrapolate the flow properties to the IB surface element.

The second-order extrapolation scheme from the auxiliary points is used to find the pressure on the surface according to the formula:

$$p_{IB} = p_1 + \frac{p}{n} = \frac{1}{2}(5p_1 - 4p_2 + p_3) \quad (3.63)$$

Several strategies are investigated for the calculation of the velocity gradients at the IB surface. Following the argument made by Kirkpatrick [62], it is assumed that only the velocity gradient in the normal direction contributes to the viscous stress at the IB surface. The velocity gradient in the normal direction can be found as:

$$\frac{u^j}{n} = \frac{u_1^j - u_{IB}^j}{n} \quad (3.64)$$

where u_1^j is the velocity at the auxiliary point P_1 in the Fig. 3.8. u_{IB}^j is the velocity at the surface triangle centre, while n is the distance from the auxiliary point to the IB. The velocity gradient obtained using the equation 3.64 is calculated with a first-order accuracy. The precision of the gradient determination can be increased to second-order by taking into account the velocity at the second auxiliary point u_2^j :

$$\frac{u^j}{n} = \frac{1.5u_{IB}^j + 2u_1^j - 0.5u_2^j}{n} \quad (3.65)$$

Although the normal velocity gradient is usually sufficient to calculate the magnitude of the forces acting on the particle, the cross term $\frac{u^i}{x^j}$ needs to be considered in order to calculate accurately the torque acting on the particle as will be shown in the chapter 5.

The complete velocity gradient tensor can be evaluated using the Taylor series expansion from the neighbouring fluid cells:

$$u_{IB}^j = u_n^j + \frac{u^j}{x^i} r^j \quad (3.66)$$

where u_n^j is the velocity at neighbouring cell and r^j is the distance from the current IB surface triangle centre to the centre of a neighbouring cell. In order to obtain the most accurate representation of the velocity gradients, a least-square fit, weighted by the distance from the triangle centre to the neighbouring cell, is applied. A 2D stencil used for the velocity gradient calculation is shown in Fig. 3.9.

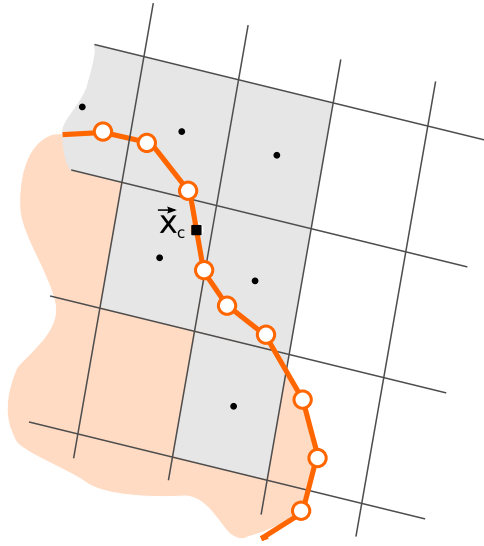


Figure 3.9: A two-dimensional view of neighbouring cells used to reconstruct the velocity gradient using the least-square fit.

Similarly as in the case of the imaginary surface, the total torque acting on the particle is calculated as the sum of the cross products of the position vectors of the relevant IB triangles with the forces acting on those triangles.

The main advantage of using the IB surface for the force calculation is the fact that all the components are evaluated directly at the IB surface, hence their contribution can be directly assessed. Moreover, this

approach is very practical for simulations with multiple bodies in close proximity, as different regions of the surface can have appropriate treatment. Collision modelling and sub-grid forces can be therefore implemented easily. Also, apart from the necessity of recomputing the auxiliary points, the method does not pose any additional problems for the moving particle case.

Still, the combined interpolation/extrapolation introduces additional errors and may lead to greater inaccuracy, as compared to the imaginary surface approach described earlier. Also, the current technique depends strongly on the surface resolution, *i.e.* the surface triangle grid has to be fine enough to accurately evaluate the forces.

4 MultiFlow Triangulated Library

MultiFlow Triangulated Library (MFTL) is a parallel triangulation library designed as part of the current research project for CFD simulations with Immersed Boundaries in MULTIFLOW solver. Although MFTL was created to be used as part of the MULTIFLOW package, it is a stand-alone library that can be adopted to any C code. The library is responsible for generating and performing operations on arbitrarily shaped triangulated bodies, like those illustrated in Fig. 4.1. MFTL is designed to operate on multiple processors.

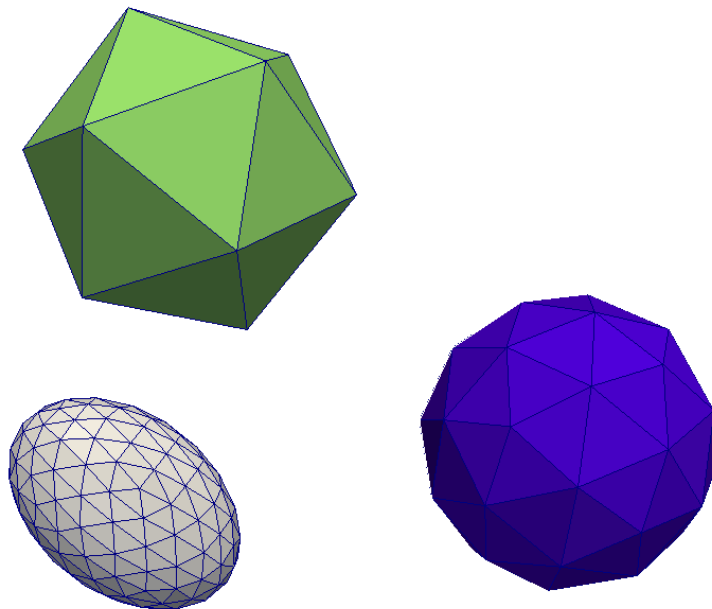


Figure 4.1: Examples of surface topologies used in MFTL.

The basic functionality of the library includes:

- Generation of spheres or reading a surface saved in a `.gts` format

- Calculation of geometrical properties of the surface, *e.g.* volume, centroid, triangle properties, etc.

- Performing basic geometrical operations on the surface: scaling, moving, rotation

- Determination whether a point of given coordinates is inside/outside the surface (point inclusion test)

Finding a list of intersection points of two triangulated bodies

Calculation of various distances in 3D: point-point, point-triangle, point-surface

Generating a convex hull from a set of planar points

Saving the body in `.vtk` or `.gts` format

MFTL is written in the C programming language and can be easily incorporated into any C code by addition of the `"mftl.h"` header file. The header file contains the definitions of all the data structures used in the library along with definitions of external MFTL functions. The internal MFTL functions, on the other hand, remain hidden to prevent user distraction. In order to access MFTL internal functions in the code, which uses the library, a specific header file (e.g. `"mftl-functions.h"` or `"geometricops.h"`) needs to be included.

MFTL uses the MPI protocol for communication between processes and UThash for the list operations.

This chapter presents the description of the library structure along with the implementation details. The algorithms for the most important functions covered as well. Finally, the integration of MFTL with MULTIFLOW and its effects on the Immersed Boundary simulations are examined.

4.1 Concept and data structure

CFD simulations with Immersed Boundaries follow a certain procedure, which has been described in chapter 3. The main motivation for designing the MFTL was to enable IB simulations with triangulated surfaces. A description of the data structures used in the library is presented, followed by a discussion of some of the most important concepts.

4.1.1 Data structure

MFTL is a library designed to handle multiple triangulated surfaces at the same time. For this to be achievable, the surface information has to be managed efficiently. Each particle is saved as a separate entity in a data structure called `MFTL_Object`, which stores all the necessary information about the surface such as the surface points coordinates, body distribution over processors and the hierarchical structures like bounding box trees.

Additionally, an independent `MFTL_Point` structure is available. This structure is used for several purposes in the library. Its most important applications are: operations with fluid grid points from the CFD code, storing the list of two surface intersection points.

Hierarchical structures (bounding box trees) are applied in MFTL for fast determination of two triangulated objects' intersection and for performing point inclusion tests. Bounding boxes are extremely useful as they enable to reject large groups of elements without the necessity of testing every one of them.

MFTL_Object

The entire information about a triangulated body is stored in a data structure called `MFTL_Object`, which is graphically depicted in Fig. 4.2. It can be seen that surfaces in MFTL are formed by inter-linked hash tables of triangles, edges and vertices. `MFTL_Object` contains the pointers only to the first element of each table.

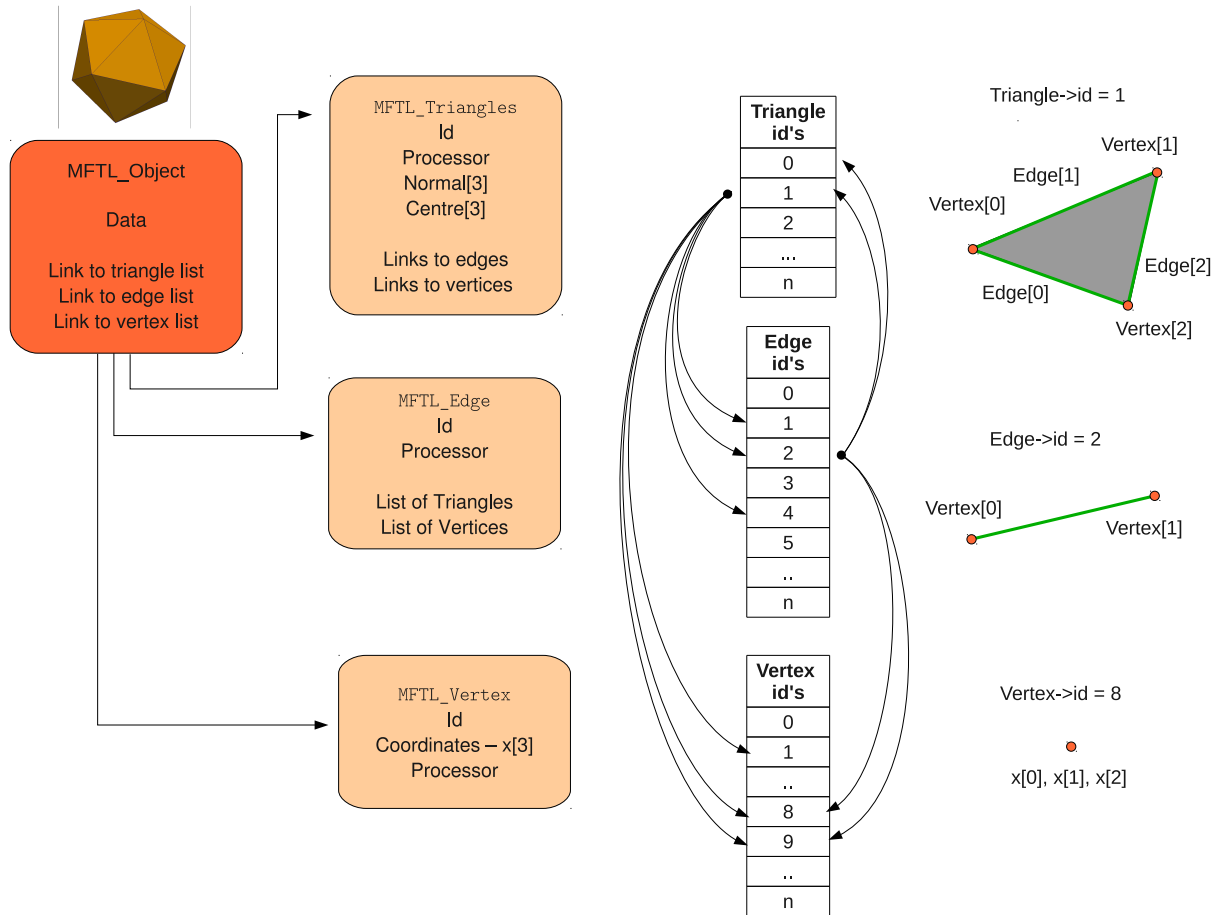


Figure 4.2: Structure of a general MFTL object.

The `MFTL_Vertex` table, which stores all the surface vertices, can be interpreted as a cloud of points defining the surface. A vertex structure contains only its id's, coordinates and the master processor. The actual surface is formed by connecting the vertices by means of triangles and edges.

The edge structure does not have any geometrical information, but it stores the pointers to its vertices along with the pointers to the two triangles it belongs to. In general, the main application of an edge is to allow identification of two neighbouring triangles.

The most important geometrical entity in the `MFTL_Object` structure is the `MFTL_Triangle`. A triangle stores not only the information about its edges, but it also points directly to the vertices as well. Apart from that, the triangle keeps some basic geometrical data about itself including data about its center, outward pointing normal and the area. Additionally, variables which are required to describe the

body-fluid interaction, such as auxiliary points, surface pressure and viscous stress, are also defined in the `MFTL_Triangle` structure.

Apart from the geometrical entities, the `MFTL_Object` also contains information about the distribution of the body over processors. This includes the range of coordinates used in the simulation, the number of processor cuts in each direction and the ownership ranges of each processor. Each object has its own parallel communicator as well.

MFTL_Point

`MFTL_Point` structure is designed for operations on points that do not belong to the surface, e.g. fluid mesh cell centres or vertices. Although such points are not part of the surface, they may be needed for certain procedures such as for determining whether a fluid grid point is inside the surface etc.

`MFTL_Point` differs from the `MFTL_Vertex` structure. Whereas the vertex contains only the information about its position in space, the `MFTL_Point` also stores its grid addressing.

Hierarchical structures

Each surface has a single global bounding box available on all processors which is defined by the span of the surface, i.e. x_{min}^j and x_{max}^j coordinates of the global box correspond to the x_{min}^j and x_{max}^j of the underlying surface object.

Two additional, local bounding box tree structures are also generated. A bounding box tree consecutively divides the object's triangle tables into groups until each group contains a single triangle, as illustrated in Fig. 4.3. The main reason for adopting the hierarchical structures is that they allow for fast rejection tests. For example, if two bounding boxes containing a number of surface triangles do not intersect, none of the triangles from the boxes will intersect, hence there is no need to investigate each triangle pair separately.

MFTL generates two types of bounding boxes for every object. Both types, orthogonal and oriented, are shown in Fig. 4.4. Orthogonal bounding boxes are applied in the point inclusion test, which uses the ray tracing method. The oriented boxes (OBBTree), introduced in [35], are designed for the fast surface intersection points determination.

Regardless of its type, a bounding box is characterised by a position vector of its centre (*Displacement*), the dimensions along its principal axes (x_{BB} , y_{BB} , z_{BB}) and the rotation matrix R^{ij} describing orientation of the box. For orthogonal boxes the rotation matrix is identity. Additionally, the bounding box stores a number of encompassed triangles as well as a link to an array with their id's, i.e. the box structure does not contain the triangles themselves, only their location in the `MFTL_Object` structure. The bounding box tree generation is discussed in more detail in section 4.2.1.

4.1.2 Tolerance for floating point comparisons

On numerous occasions MFTL, has to decide whether points are coincident, lie on the same line or are coplanar. One of the challenges for this type of assessment in the computational geometry is

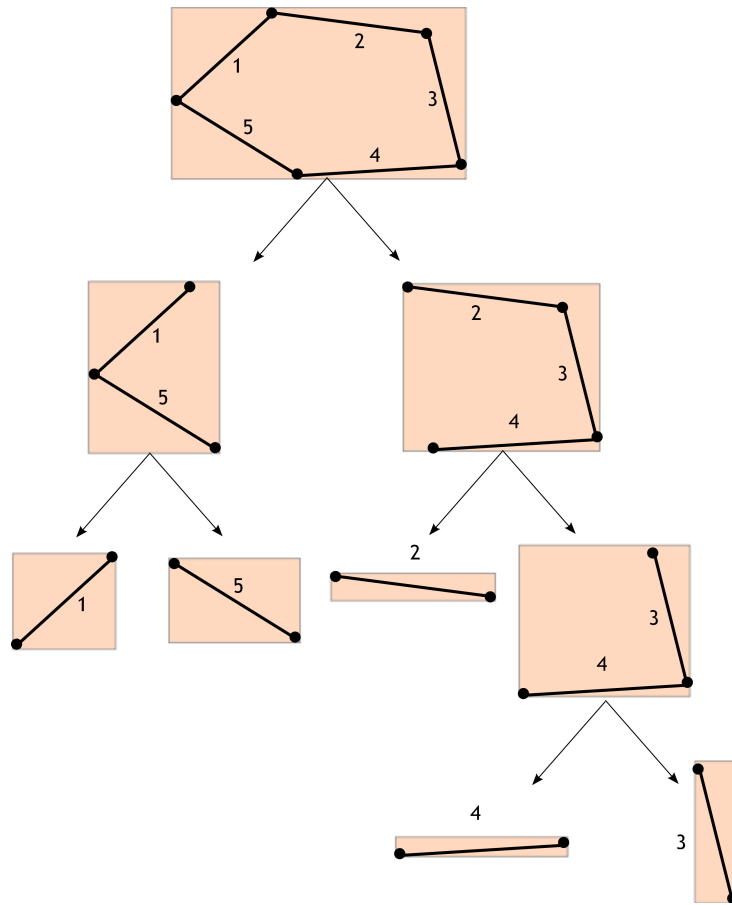


Figure 4.3: Illustration of an orthogonal bounding box tree for a polygon from five edges.

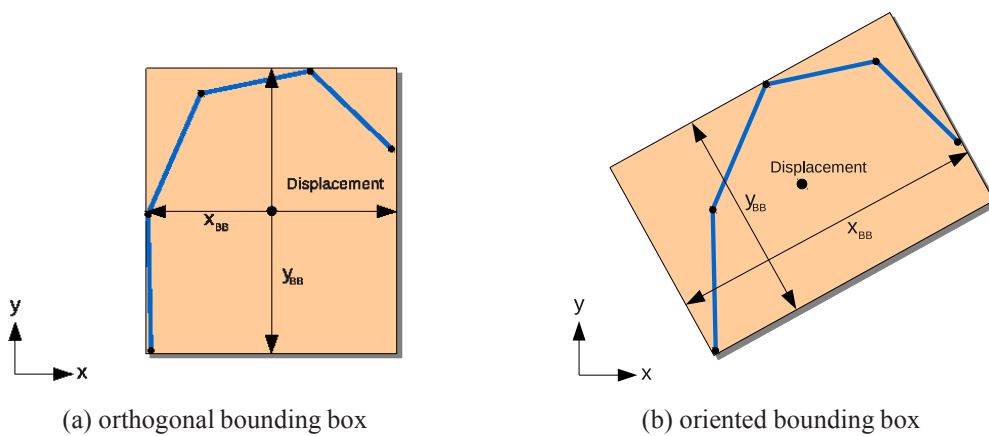


Figure 4.4: Schematic illustrations of different bounding box types.

related to the floating point accuracy. Numbers, in the C programming language, are stored in a binary format with fixed accuracy. Mathematical operations can therefore introduce a round-off error, which can lead to a situation, where for instance $3.0/3.0 = 0.999999999996$. Hence, a comparison of two floating-point numbers is a non-trivial task.

Usually, a test whether the value of variable a is equal to the value of b has to be written as $a - b < TOL$, where TOL is the pre-defined tolerance. Similar approach is applied in MFTL, however a variable tolerance value is used. This is because the library operates on surfaces of different length-scales and needs to retain the same accuracy level for each body.

The tolerance value in MFTL is based on the object geometry. Once the surface is generated, a special function is called to evaluate the desired tolerance level. By default, the value of TOL is equal to 10^{-6} times the length of the smallest triangle edge of the surface. This means that if a distance between any two points is smaller than the tolerance value, they will be treated as coincident.

The following method is applied, in order to determine whether three points A, B, C (with different coordinates) are collinear. First, the normal vector to the plane defined by the points is calculated as a cross product of 2 vectors $V_{BA}^j = B^j - A^j$ and $V_{CA}^j = C^j - A^j$. Then, the length of the resulting vector $V_{BACA} = V_{BA} \times V_{CA}$ is compared to the TOL value. If the length of the normal vector V_{BACA} is smaller than the MFTL tolerance level, the three investigated points are treated as collinear.

4.1.3 Parallel implementation of MFTL

One of the main motivations for designing MFTL was to enable parallel operations in the simulations with triangulated surfaces. This means that during a parallel simulation with multiple surfaces, each processor will store and operate only on a fraction of the data. An example of such a case is depicted in Fig. 4.5. Here, the fluid grid on the first processor interacts with two triangulated bodies, while the fluid on the second processor needs access to three surfaces. In fact, the surface present on both processors can also be divided, so that each processor uses only the necessary data. Therefore, after the bodies are redistributed over the processors, the processor 1 will store information about one full body and half of the second one, while the processor 2 will have access to two bodies and a half of the third one.

As mentioned earlier, the information about the processor count and the ownership range is stored in the `MFTL_Object` structure. Also, each geometrical entity (triangle, edge, vertex) is assigned to a single processor. The range of ownership of a processor n is defined by variables:

$$\text{ownershiprangemin}[\text{dir}][n]$$

$$\text{ownershiprangemax}[\text{dir}][n]$$

in the `MFTL_Object` structure, where $\text{dir} = 0, 1, 2$ is the direction. This allows to determine the processor owning each vertex in a unique manner. The main processor of edges and triangles is set to be the same as the processor of the first vertex of the element.

Fig. 4.6 illustrates sample distribution of a single body over two processors. Each processor stores only the elements that lie within its ownership range. If at least one vertex of any triangle lies inside

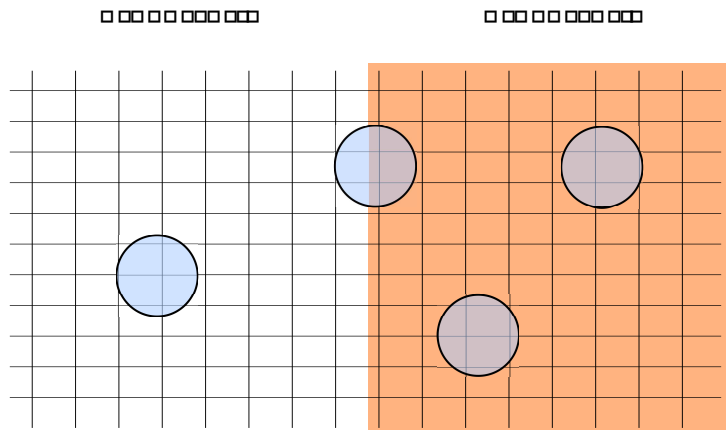


Figure 4.5: Parallel simulation with multiple particles. Some of the particles may belong to two or more processors.

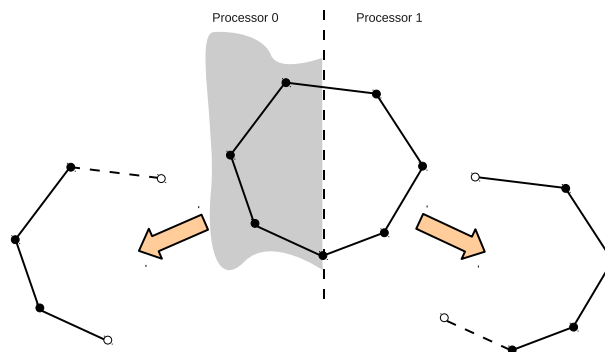


Figure 4.6: Distribution of the surface over multiple processors. Solid lines and full circles represent edges and points that belong to the current processor, while dashed lines and empty circles correspond to *ghost* entities.

the processor ownership range, the processor will also store this triangle along with its edges and other vertices. However, if the current processor does not own these elements, they will be marked as *ghosts* - empty dots and dashed lines in Fig. 4.6.

Surfaces are distributed over processors directly after their generation. Additionally, the redistribution function has to be called whenever the body is moved. The following steps are performed in order to divide the body over multiple processors:

1. Determine the processor owning each vertex
2. Re-label the edges of the object according to processors of their “first” vertex
3. Loop over triangles of the surface at the current CPU:
 - a) Delete *ghost* triangles, edges and vertices.
 - b) If any vertex of the triangle is to be moved to a different processor, add the triangle and its data to transmission array
 - c) If all vertices of the triangle are moved, remove the triangle from the current processor
4. Transmit the data over processors
5. Decode the transmission arrays, regenerate the object structure
6. Remove the edges and vertexes that do not belong to any triangle on the current CPU.

The parallel implementation of the library has strong implications for various operations which are to be performed. Generally, the MFTL functions can be divided into two groups: serial and parallel. The functions in the first group, such as calculation of triangle surface areas, are performed locally and do not require any communication between the processors. On the other hand, the parallel operations will need to transmit the data at some point. For instance, when the total body surface area is calculated, the partial surface areas evaluated on every processor have to be added in the final stage. Details of the parallel implementation of specific functions are discussed in the next sections, where the algorithms applied for various MFTL operations are presented.

4.2 Main functions

This section describes in detail the procedures adopted to perform some of the more advanced operations enabled by MFTL. An overview of each algorithm is discussed, followed by the explanation of the specific steps performed by the routines.

4.2.1 Bounding box tree generation

The bounding box trees are local structures, meaning that they store only the information about the triangles owned by the current processor. The main goal of generation of a bounding box tree is to create

a structure that allows to easily and efficiently find a specific surface triangle. The algorithm describing the procedure adopted in order to create a bounding box tree is illustrated in Fig. 4.7.

In the first stage, the function calculates the total number of triangles in the box (including *ghosts*) and checks if it is non-zero. This step is necessary, since the function is called by all CPU's, even those which do not have any elements of the surface. If the number of triangles equals to zero, the function returns an empty bounding box, otherwise it allocates memory for the head node of the bounding box and saves the id's of encompassed triangles in an array. Afterwards, the routine recursively splits the current bounding box into smaller boxes to create a tree structure (see Fig. 4.3).

The process of splitting starts by checking whether a leaf node is found (*i.e.* a box with only one triangle). If this is the case, the function performs the final splitting operation described later on. Otherwise, the function calculates the bounding box properties.

Next, the bounding box is cut in half along its longest dimension and the function divides its triangles into two groups which are transmitted to the next tree level. In the last stage, the routine calls itself with the children of the current node being the argument. The operation is performed from left to right until each box is broken down into indivisible leaf nodes.

Basic geometrical parameters, such as dimensions, displacement and rotation matrix are calculated for each bounding box in the tree structure. The most important part of this operation is the evaluation of the rotation matrix \mathbf{R}_{ij} which describes the box orientation with respect to the main coordinate system. The rotation matrix is found by calculating the eigenvectors of the covariance matrix \mathbf{C}_{ij} :

$$\mathbf{C}_{ij} = \frac{1}{12} (9A_t c_t^i c_t^j + V_0^i V_0^j + V_1^i V_1^j + V_2^i V_2^j) - \frac{c^i c^j}{A} \quad (4.1)$$

where V_i^j is the j coordinate of the i -th vertex of triangle, A_t is the triangle area, while c_t^j is the j coordinate of the triangle centre. Non-indexed variables c^j and A are the mean vertex coordinate and sum of triangle areas in the box respectively. They are calculated as:

$$A = \sum_t A_t \quad (4.2)$$

$$c^j = \frac{\sum_t A_t c_t^j}{A} \quad (4.3)$$

The eigenvectors computed from covariance matrix tend to align with the box axes [35], hence they are used to define the rotation matrix. First row of the matrix corresponds to the dominant orientation, *i.e.* the one with the largest dimension. In case of orthogonal boxes, the rotation matrix is set to identity.

When the rotation matrix is known, the maximum and minimum points of the box in the coordinate system, defined by the rotation matrix, are calculated. The extreme points are used to determine the box dimensions in the oriented coordinate system:

$$d^i = \frac{1}{2} (\text{Max}^i - \text{Min}^i) \quad (4.4)$$

4.2. MAIN FUNCTIONS

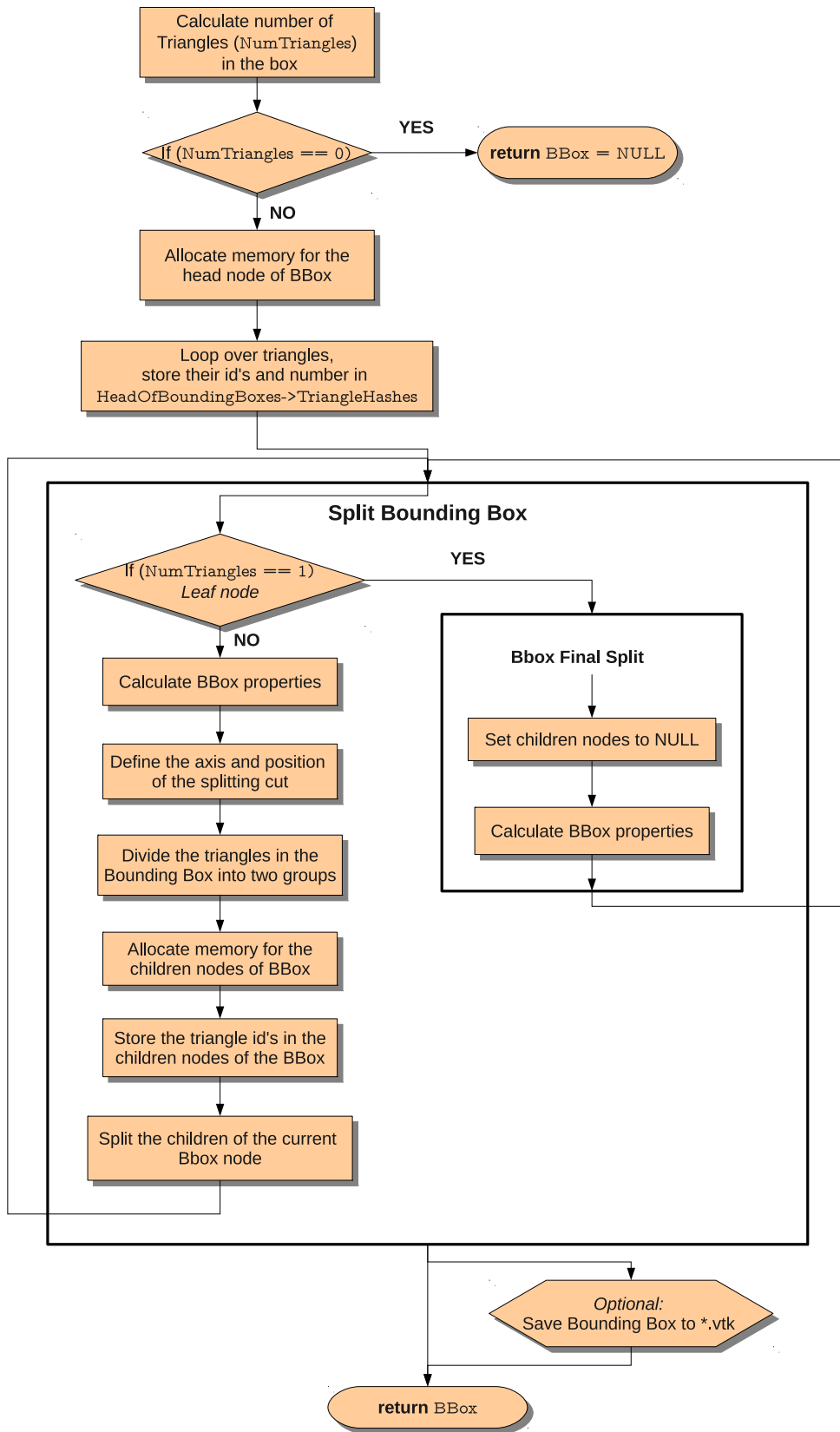


Figure 4.7: Flowchart showing the procedure for bounding box tree generation.

and the centre of the box:

$$c^i = \frac{1}{2}(\text{Max}^i + \text{Min}^i)R_{ij} \tag{4.5}$$

After the box properties are evaluated, its triangles can be divided into two groups that will be passed on to the next tree level. First, a cut along the longest principal axis of the box has to be defined, as illustrated in Fig. 4.8.

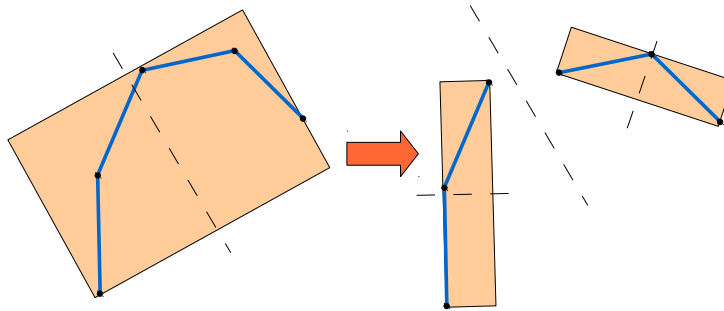


Figure 4.8: Dividing the bounding box.

Next, the function loops over the triangles in the box and checks whether the triangle centre lies on the left or on the right hand side of the cut. The array with the id's of the triangles (stored in the head node of the tree) is then rearranged, so that triangles on the left-hand side of the cut stay in the first part of an array, while the triangles on the right-hand side of the cut are moved to the second part of the array (see Fig. 4.9). If all triangles are on one side of the cut, the function arbitrarily divides the array in half. The number of triangles on each side of the cut is calculated during the rearrangement and is saved as the number of triangles in the children nodes. Finally, pointers to specific parts of the main array are assigned and stored in the children nodes. Note that there is only one array with the triangle id's on each processor which is stored in the head node of the tree, while the children nodes only link to its specific parts.

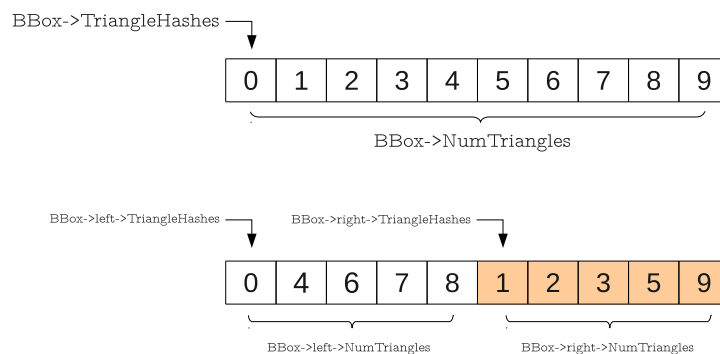


Figure 4.9: Reordering of triangle id's in the array. The children nodes of the box point to the specific part of the array.

Treatment of the leaf nodes with only one triangle is relatively easy. The box properties are calculated without using the covariance matrix. Instead, the major axis of the box is aligned with the longest edge

of the triangle, while the minor axis is simply the normal of the triangle. The third axis is a vector cross product of the major and the minor axes. The box dimensions and centre are evaluated using the same formulas as for non-leaf nodes (equations 4.4 and 4.5 respectively).

4.2.2 Point-triangle distance

The minimal distance d between a point P and an arbitrary surface triangle is calculated using a 2D projection method, described by Jones [58].

The procedure requires establishing a new coordinate system $x y z$, in which the entire triangle lies on the $y z$ plane, with $V[0]$ vertex of triangle being the origin and $V[1]$ lying on z axis. The original point P is transformed into P' . Fig. 4.10 illustrates the problem formulation.

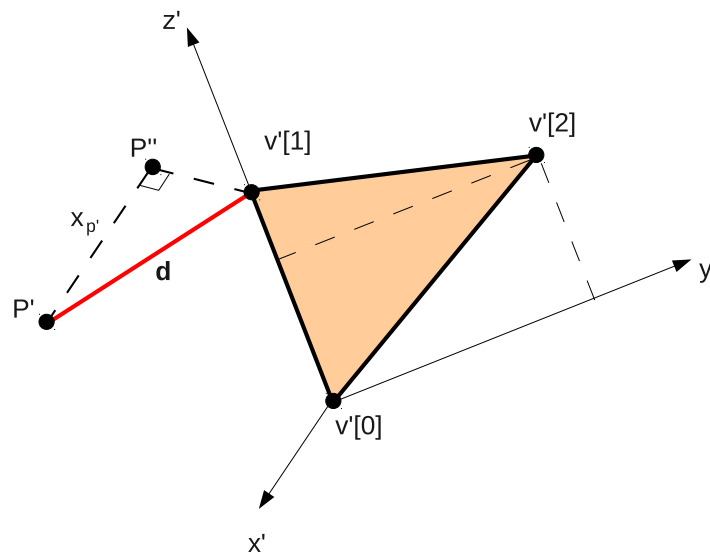


Figure 4.10: The point and triangle in $x y z$ coordinate system.

In the new coordinate system, point P can be easily projected onto the triangle plane by ignoring the x component of its coordinates. In the second stage, the function checks which element of the triangle is closest to the projected point P' . Seven zones around the triangle can be distinguished, as shown in Fig. 4.11. Zone 0 lies inside the triangle. If P' lies in zones 1, 3 or 5, the closest element to the triangle is the corresponding vertex, while for other zones the corresponding edge is the closest triangle element. Zones 2, 4 and 6 are bounded by the triangle edges and perpendicular lines, crossing the respective triangle vertices.

The function checks the orientation of the point with respect to the edges in order to determine the zone, which point P' falls into. This is performed by evaluating $E_{edge}(P')$. For the edge defined by vertices $V[0]$ and $V[2]$ it reads:

$$E_{02}(P') = (y_P - y_0)(z_2 - z_0) - (z_P - z_0)(y_2 - y_0) \quad (4.6)$$

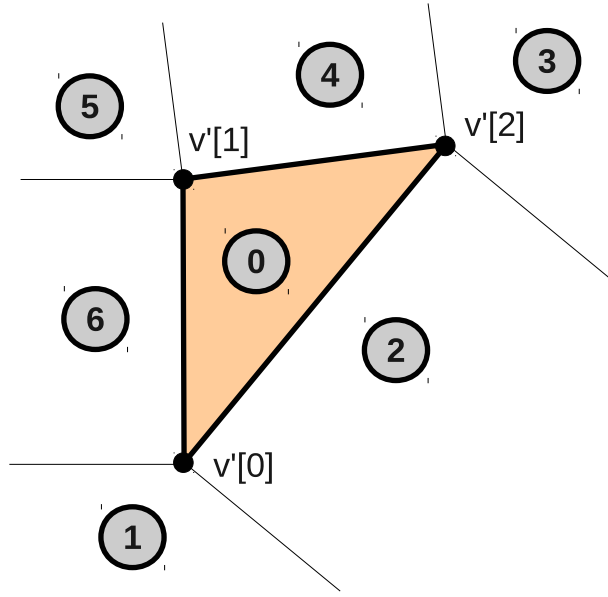


Figure 4.11: Various zones describing the closest element of the triangle.

If $E_{02}(P) > 0$, point P lies on the right hand side of the edge, *i.e.* in zones 1, 2 or 3. On the other hand, if $E_{02}(P) < 0$, it is located on the left hand side of the edge. Similar check has to be done for other edges. When $E_{02}(P) = 0$, the point lies on the line defined by the edge. The position relative to the perpendicular lines is determined in a similar fashion. Finally, the distance from point P to the triangle is calculated as the square root of the sum of the squared distance between point P and the closest element of the triangle added to the square of the x coordinate of point P .

The procedure can be illustrated with a simple example. Consider a triangle defined in $x y z$ coordinates by vertices $(0 0 0)$, $(0 0 2)$, $(0 1 1)$ and point $P = (2 2 1)$. First, the position of P relative to edge 0 2 is determined using equation 4.6. In this case, $E_{02}(P) = 1$, hence the projected point P is on the right hand side of edge defined by $V[0]$ and $V[2]$. Now, the orientation of the point with respect to the line perpendicular to that edge and crossing vertex $V[0]$ has to be determined:

$$E_{02}(P) = (y_P - y_0)(y_2 - y_0) - (z_P - z_0)(z_2 - z_0) \quad (4.7)$$

$E_{02}(P) = 1$, so P lies on the right hand side of the $E_{02}(V[0])$ line. An identical test for the perpendicular line crossing vertex $V[2]$ shows that the point P is located in zone 3. Therefore the distance from the point P to the triangle is:

$$d = \sqrt{x_p^2 + [(y_p - y_2)^2 + (z_p - z_2)^2]} \quad (4.8)$$

If the projected point P is inside the triangle, the distance would be simply the x coordinate of P .

4.2.3 Point inclusion test

Determination whether a given point is inside the surface is a parallel function using the ray tracing technique. The principle of the method is illustrated in Fig. 4.12. A number of intersections between the surface and a random ray starting at point P is calculated. For a closed surface, an odd number of intersection points means that point P lies inside the body, otherwise it is outside. Since in MFTL parts of the surface may belong to different processors, additional communication between the CPU's is necessary for performing the point inclusion test.

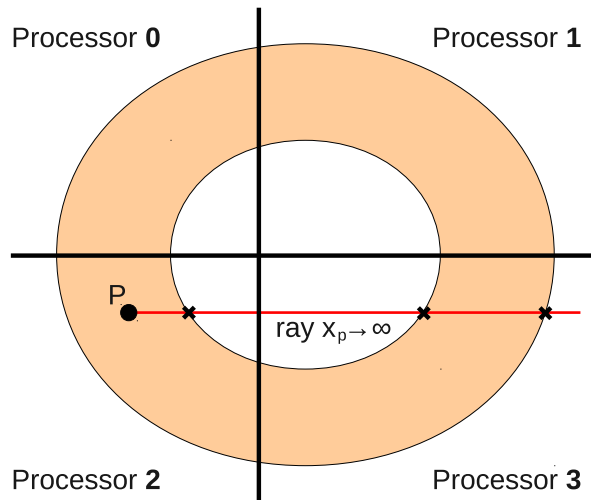


Figure 4.12: Illustration of the ray tracing method on multiple processors. There is one intersection point between the ray and the surface on processor 0, and 2 intersections on processor 3. When the number of the points is added it becomes 3, an odd number, so the point is inside the surface.

At the beginning of the process, a random ray has to be generated and broadcast to all processors. Next, the function navigates the orthogonal bounding box tree, checking if the ray stabs the bounding box at the current level. When the leaf node is reached, the tested point is projected onto the triangle plane along the ray direction and the routine tests, if the projected point is inside the triangle.

This is done by checking, for each triangle edge, whether the cross products of vectors defined by the edge vertices and one of the vertices and the point, have the same direction as the reference edge (Fig. 4.13).

If the point stabs the triangle, it is added to the list of intersection points. Once the function traverses through entire tree structure, the number of intersection points along with their coordinates is known, and can be transmitted to other processors. However, in a the case when the projected point lies either on the triangle edge or coincides with one of the vertices, a degenerate case flag is assigned. For a degenerate case, a new ray has to be generated and the procedure repeated.

In the final stage of the point inclusion test, the processors count the total number of intersection points. If the number of stabbed triangles is odd (for a closed surface), it means that the tested point is inside the surface, otherwise it is outside.

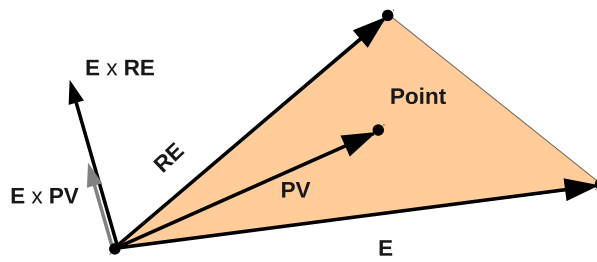


Figure 4.13: Determination if the point is inside of a triangle. If $(\mathbf{E} \times \mathbf{PV}) \cdot (\mathbf{E} \times \mathbf{RE}) > 0$, for every edge the point lies inside of the triangle.

4.2.4 Intersection of two surfaces

The function calculating the intersection of two surfaces returns a list of intersection points of two bodies on the current processor. The list is available only on the CPU which calculated the intersection. If the entire intersection is needed, the point list has to be gathered across the processors.

The process of determining the intersection has 2 stages:

1. Find the list of overlapping triangles.
2. Calculate the intersection points of individual triangle pairs.

The routine applies bounding box tree structures, based on the method described in [35], for fast completion of the first stage. The second stage is performed using the concept introduced by Troop *et al.* [109].

Before the actual intersection points can be found, MFTL has to efficiently determine which surface triangles can be in contact. Application of oriented bounding box tree structures described earlier is a convenient method which allows to perform this operation. For any two bounding boxes, a fast overlap test can be performed. If the boxes overlap, a similar test is performed for their children and so on, until the function reaches a pair of leaf nodes (*i.e.* indivisible bounding boxes). When two indivisible boxes overlap, the triangles they encompass are stored in a list of triangle pairs. This approach enables a fast rejection of large groups of triangles, without the need of looping over all triangles in each body.

The overlap test is illustrated in Fig. 4.14. Vector \mathbf{T} is the displacement of the centres of the bounding boxes A and B. The principle of the test is to project the boxes onto a separating line defined by vector \mathbf{L} , in order to obtain two intervals defined by object centres and radii \mathbf{r} . If the two intervals do not overlap, the boxes are disjoint. Mathematically, this corresponds to performing the following test:

$$\mathbf{T} \cdot \mathbf{L} > r_A + r_B \quad (4.9)$$

If the above inequality is true, the bodies are disjoint and no further tests are required.

15 tests with different separating axes \mathbf{L} are necessary, in order to ensure that two oriented bounding boxes overlap. If, in all cases, the inequality 4.9 does not hold, it means that the boxes are not disjoint. The directions of separating axes are defined by 3 faces of the first box, 3 faces of the second box and 9

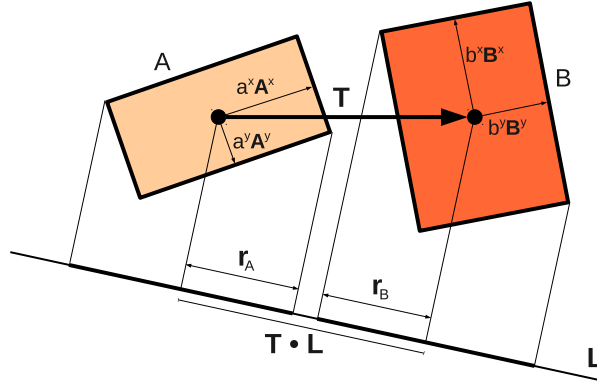


Figure 4.14: The fast overlap test illustration. The boxes overlap if their projected radii r_A and r_B cross each other.

combinations of the edges:

$$L_i^j = A_j^T i, \quad j = 1 \ 2 \ 3 \quad (4.10)$$

$$L_i^j = B_j^T i, \quad j = 4 \ 5 \ 6 \quad (4.11)$$

$$L_i^j = A_m^T i \ B_n^T i, \quad j = 7 \ 15 \ m = 1 \ 3 \ n = 1 \ 3 \quad (4.12)$$

$$(4.13)$$

where \mathbf{A} and \mathbf{B} are the rotation matrices of the boxes that need to be transposed. When the separation axis is known, the interval radii r_A and r_B can be calculated. r_A is defined as:

$$r_A = \sqrt{\sum_i a_i^2 \mathbf{A}^T \mathbf{L}^2} \quad (4.14)$$

where a_i is the dimension of the box in the i direction in the box coordinate frame. Second box radius, r_B can be found in an analogous way.

If the bodies overlap, the function will test boxes in the next level in the first tree structure. If this is the leaf node, the second tree needs to be traversed. When both tree nodes are indivisible, the id's of the triangles are added to the list of potentially intersecting triangle pairs. After traversing through the both trees is finished, a list of potentially intersecting triangle pairs is completed, and the routine can move to the next stage - determination of the intersection points.

In the second stage, the routine loops over the list of overlapping triangle pairs and adds their intersection points to the global intersection list. Fig. 4.15 illustrates the variables used in the process.

Calculation of the intersection points can be described by the following steps:

1. Find the parameters.
2. Use the parameters to check if the triangle A intersects the plane of the triangle B.
3. Create t vector - a segment of the triangle A intersecting the plane of the triangle B.

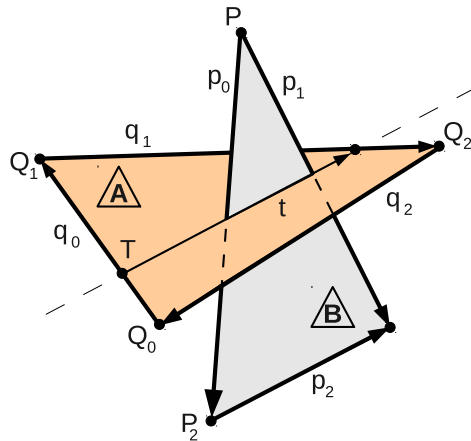


Figure 4.15: Two intersecting triangles.

4. Find the points of intersection of the line defined by t with the edges of triangle B.
5. Determine whether the intersection points belong to both triangles. Also, check if the new points coincide with any point already in the list. Finally, add the non-coinciding points to the intersection list.

The intersection point between the plane defined by vectors p_0 and p_1 , and the edge q_i , ($i = 0, 1, 2$) can be found by solving:

$$P + \alpha p_0 + \beta p_1 = Q_i + \gamma q_i \quad (4.15)$$

The equation can be rewritten in a matrix form as:

$$A \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = r_i \quad (4.16)$$

where $r_i = Q_i - P$, and A is a matrix defined as $A = (p_0 \ p_1 \ q_i)$. γ is a parameter specifying the position of the intersection point on the vector q_i . If $0 \leq \gamma \leq 1$, parameter γ is called legal and the point lies on the i 'th edge of triangle A. Therefore knowledge of γ_i for all edges is sufficient to determine if triangle A intersects the plane of triangle B.

Calculation of γ_i is done in the first step. This is performed by using determinants of matrix A :

$$\gamma_i = \frac{A(q_i)}{A(r_i)} \quad (4.17)$$

Next, the function checks whether triangle A intersects the plane defined by triangle B. This is done by evaluating if $0 \leq \gamma_i \leq 1$. There are 3 possible scenarios:

None of the found γ_i satisfies the inequality - the triangles are disjoint, no further steps are needed, move to the next triangle pair.

At least two γ_i satisfy the inequality - triangle A is intersected by the plane of Triangle B, therefore function moves to step 3.

All the determinants $A(\mathbf{q}_i)$ are zero, so the triangles are coplanar - special treatment is needed to find the intersection.

For the second option, the function creates a vector \mathbf{t} , a segment of triangle A intersected by the plane of triangle B, and a point \mathbf{T} to which the segment is attached. \mathbf{T} is calculated as:

$$\mathbf{T} = \mathbf{Q}_i + \gamma_i \mathbf{q}_i \quad (4.18)$$

where i is either 1 or 2, depending on which γ_i satisfies the 0 to 1 condition. The vector \mathbf{t} depends on the second legal γ_i and can be:

$$0,1 \text{ legal: } \mathbf{t} = \mathbf{q}_0(1 - \gamma_0) + \gamma_1 \mathbf{q}_1$$

$$0,2 \text{ legal: } \mathbf{t} = \mathbf{q}_2(1 - \gamma_2) + \gamma_0 \mathbf{q}_0$$

$$1,2 \text{ legal: } \mathbf{t} = \mathbf{q}_1(1 - \gamma_1) + \gamma_2 \mathbf{q}_2$$

the last case applies also if all the γ_i 's are legal, but 1 and 2 show the same vertex.

In the next step the exact intersection point between the \mathbf{t} vector and the edges of triangle B is found by solving a parametric equation:

$$\mathbf{P} + \gamma_i \mathbf{p}_i = \mathbf{T} + \gamma_j \mathbf{t} \quad (4.19)$$

The equation calculates the intersection point of two lines: one defined by \mathbf{t} and the second given by \mathbf{p}_i edge of B, and can be used to find both γ_i and γ_j for each edge of triangle B. The parameter γ_j is equal to:

$$\gamma_j = \frac{(\mathbf{p}_i - \mathbf{t}) \cdot [(\mathbf{T} - \mathbf{P}_i) \times \mathbf{t}]}{(\mathbf{p}_i - \mathbf{t}) \cdot [(\mathbf{T} - \mathbf{P}_i) \times \mathbf{t}]} \cdot \frac{(\mathbf{T} - \mathbf{P}_i) \cdot \mathbf{t}}{(\mathbf{p}_i - \mathbf{t})} \quad (4.20)$$

with conditions that $[(\mathbf{T} - \mathbf{P}_i) \times \mathbf{t}] \cdot (\mathbf{p}_i - \mathbf{t}) = 0$ and $(\mathbf{p}_i - \mathbf{t}) \cdot (\mathbf{p}_i - \mathbf{t}) > 0$. The second parameter can be obtained in a similar way as:

$$\gamma_i = \frac{(\mathbf{t} - \mathbf{p}_i) \cdot [(\mathbf{P}_i - \mathbf{T}) \times \mathbf{p}_i]}{(\mathbf{t} - \mathbf{p}_i) \cdot [(\mathbf{P}_i - \mathbf{T}) \times \mathbf{p}_i]} \cdot \frac{(\mathbf{P}_i - \mathbf{T}) \cdot \mathbf{p}_i}{(\mathbf{t} - \mathbf{p}_i)} \quad (4.21)$$

with conditions that $(\mathbf{t} - \mathbf{p}_i) \cdot [(\mathbf{P}_i - \mathbf{T}) \times \mathbf{p}_i] = 0$ and $(\mathbf{t} - \mathbf{p}_i) \cdot (\mathbf{t} - \mathbf{p}_i) > 0$. If the conditions are not satisfied the parameter is set to a very large value.

The parameters γ_i and γ_j can be used to find the intersection points of the edges of triangle B and the vector \mathbf{t} . Some possible triangle arrangements are illustrated in Fig. 4.16. If $0 < \gamma_i < 1$, the function checks the corresponding γ_j . If γ_j is also in the same interval, the i edge of triangle B intersects \mathbf{t} inside triangle A. If this happens for two edges, the \mathbf{t} vector is fully contained in triangle B. Alternatively, if there are two γ 's in the $< 0, 1 >$ interval, but one of the gammas does not fit inside, the edge point will be used for intersection (middle image of the figure). If two γ 's are outside the $< 0, 1 >$ interval and

have the same sign, there is no intersection, however if they have different signs, the intersection will be defined by points T and $T + t$. Cases in which there is only one intersection point are also included in the above considerations.

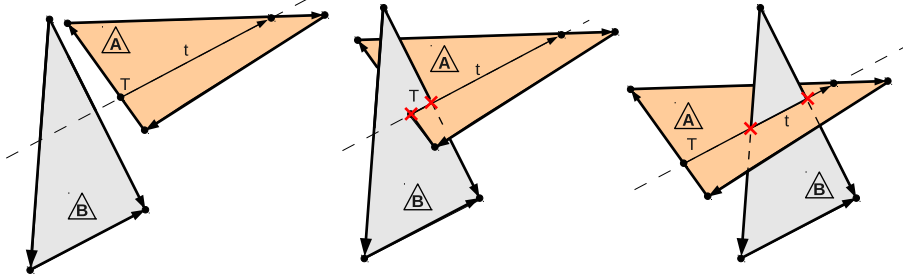


Figure 4.16: Some possible variations of intersection of two triangles.

Similar approach as the one described above is applied to coplanar triangles. In this case, however, only points that lie on the intersection of edges are considered to be the intersection. The function defines point T and vector t as the edges of the triangle A. Three checks are performed. Then i and i are calculated to find the exact intersection points.

The final step of the second stage of the routine calculating the intersection points is to check, whether the found point is already in the list of intersection points by looping over the list and determining if the distance between new and each of the existing points is smaller than the tolerance level. If the point is not present yet, it is added to the list.

The product of the surface intersection function is an unsorted list of points that belong to both surfaces. A list of intersection points is available locally on the current processor and can be easily transmitted to other processors.

4.2.5 Convex hull operations

The convex hull can be defined as the smallest convex polygon that contains a set of N points. This mathematical concept is widely used in the computational geometry. In MFTL, convex hulls are applied for calculation of the area of intersecting objects. For instance, a convex hull allows to easily compute the fraction of a triangle area which is inside a fluid cell.

Convex hull generation

A convex hull can be generated for any set of N points lying on the same plane. MFTL offers two possible scenarios for the convex hull treatment. Given a list of unsorted planar points, MFTL will either return a list of sorted points which form a hull, or generate a planar surface defining the hull from the sorted points.

The algorithm for convex hull generation is based on the procedure proposed by O'Rourke [91]. For a given set of planar points X , the following steps are performed:

1. Copy the original point list to X_C and align the points in the new list with the XY plane

2. Find the rightmost lowest point and label it as p_0
3. Sort the remaining points angularly about p_0 in the counter-clockwise direction
4. Perform the Graham Scan to exclude the points inside the hull
5. Re-arrange the original list according to the order of X_C
6. *optional* Create a surface from the ordered point list X

Note that both sorting and the Graham Scan are performed on the copy of the original point set, which remains unchanged until the fifth step.

In the first stage of the algorithm, X_C - a copy of the original point set is made. Then, points in X_C are aligned with the XY plane. This is achieved by finding a rotation axis U^j and a rotation angle θ . The U^j axis is evaluated as a cross product of N^j , the normal vector to the plane defined by points in the set, with the z axis vector ($Z^j = [0 \ 0 \ 1]$). The cosine of the angle of rotation is $\cos(\theta) = \frac{N^j \cdot Z^j}{|N^j| |Z^j|}$. Once the rotation axis and the rotation angle are determined, the points can be aligned with the XY plane.

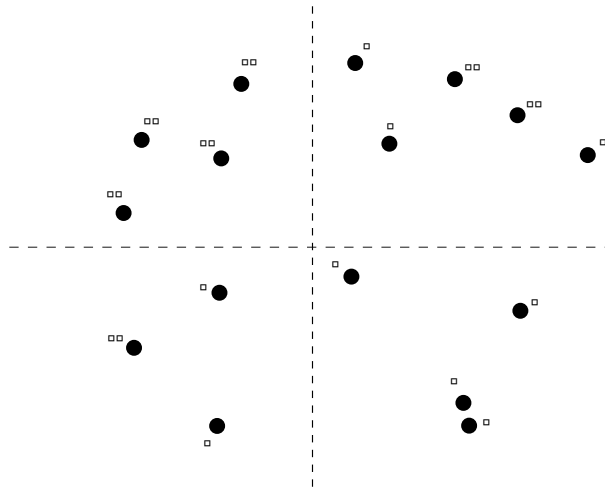


Figure 4.17: An example of set of points used for convex hull generation.

Fig. 4.17 shows an example set of points on the XY plane which can be encountered during the process of convex hull generation. The second step of the algorithm requires finding the rightmost lowest point p_0 . For this purpose, the point with the lowest y coordinate is determined. If several points with the lowest y value are found (e.g. points 1 and 7 in the figure), p_0 is set to be the one with the highest value of the x coordinate. Here, $p_0 = 7$.

Next, all the remaining points are sorted angularly around the point p_0 in the counter-clockwise direction. In order to increase the algorithm performance and avoid some of the inaccuracies introduced by floating point arithmetic, the sorting routine applies the signed area to determine the points order. Consider points 14 and 15 in Fig. 4.18. The signed area of the triangle defined by points 7, 14 and 15 is negative, therefore the points need to be swapped. If the signed area is zero (e.g. for points 8 and 9), the points are collinear, and the point closer to the p_0 will be deleted (8 in this example).

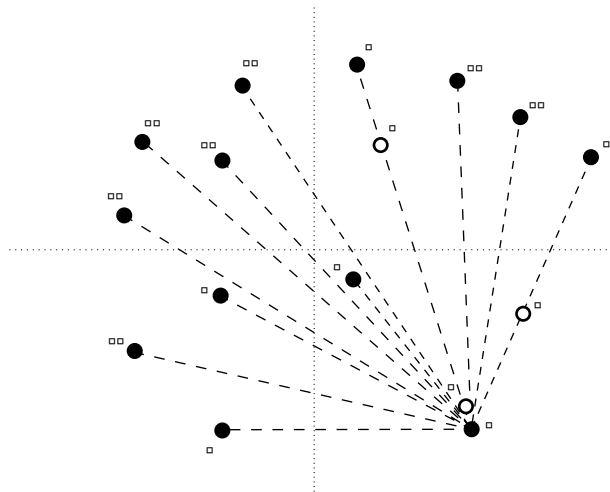


Figure 4.18: Points on XY plane after sorting. Empty circles represent points that are removed from the original data set.

At the end of the second step an ordered list of points is available. Fig. 4.18 illustrates the state of X_C after sorting. The empty circles correspond to the points that have been removed from the list.

The Graham Scan algorithm can be applied on the set of sorted points. The algorithm removes the points that are inside the hull from the list. This is achieved by determining whether a signed area of the triangle formed by three consecutive points is greater than zero. If this is not the case, the middle point of the triangle is removed. Consider points 4, 15 and 2, which illustrate the procedure. The signed area of the triangle formed by these points is positive, therefore the algorithm can move to the next three points: 15, 2, 14. Here, the triangle area is negative, so point 2 is removed from the list, as will be point 14 in the next step. The resulting convex hull is depicted in Fig. 4.19. The empty circles represent the points removed from the original data set.

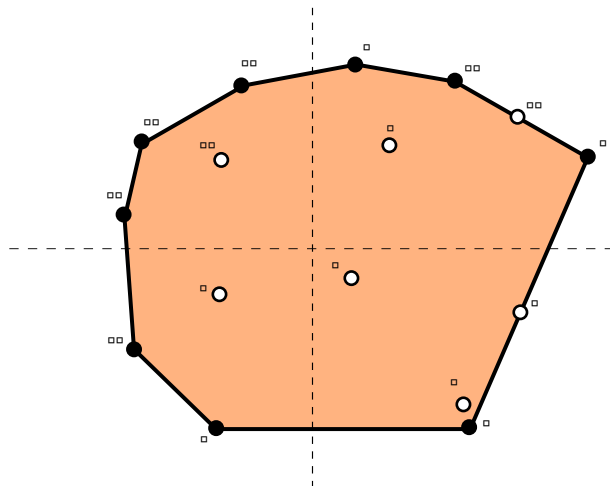


Figure 4.19: A convex hull generated after the Graham Scan. Empty circles represent points that are removed from the original data set.

Finally, the original list is re-arranged using the information from the X_C . This results in a convex hull spanned on an arbitrary three-dimensional plane. Optionally, a triangulated surface can be created using the sorted point list. The triangles of such a surface will be defined by point p_0 and consecutive hull vertices pairs.

4.3 Integration with MULTIFLOW

This section describes some details of the way MFTL is integrated into MULTIFLOW. The implementation of a point inclusion test function on multiple processors is described, along with the procedure which allows to calculate the geometrical properties of the *cut-cells*.

4.3.1 Parallel point inclusion test and cell tagging

One of the most important procedures performed during a simulation is setting appropriate tags to the fluid cells, based on their position with respect to the particle grid. Tagging values prescribed for the current numerical method are described in detail in chapter 3.2.1. Here, an outline of the tagging procedure is given.

The procedure requires the following steps:

1. Initialise the tags of all cells to 0
2. Loop over the fluid domain and store the points that fall into the global bounding box of the particle
3. Gather the points over the processors, so that they are accessible to every CPU
4. Perform the parallel inclusion test
5. Loop over the domain for the second time, and set the tags of the fluid points close to the boundary and the solid points far inside the surface.

The basic concept of the algorithm can be illustrated by means of Fig. 4.20. The first step of the routine involves a fast rejection test. If a fluid cell is inside the global bounding box of the particle, available on each processor, then it will be stored in a list of test points (points 2, and 3), otherwise it is left unchanged (point 1).

After the first looping is finished, the lists of test points from each processor are gathered, so that all CPU's have access to the same list. Now, the point inclusion test, described in section 4.2.3 is done for each point in the test list. Fluid cells outside the surface (*e.g.* point 2 in the figure), along with the points not belonging to the current processor, are removed from the list, while the cells inside the surface are assigned a tag corresponding to the body number.

Afterwards, another looping over the fluid domain is performed, where the cell types are modified according to their neighbours as prescribed in chapter 3.2.1. Also, the list of points inside the particle on each processor is saved for further use.

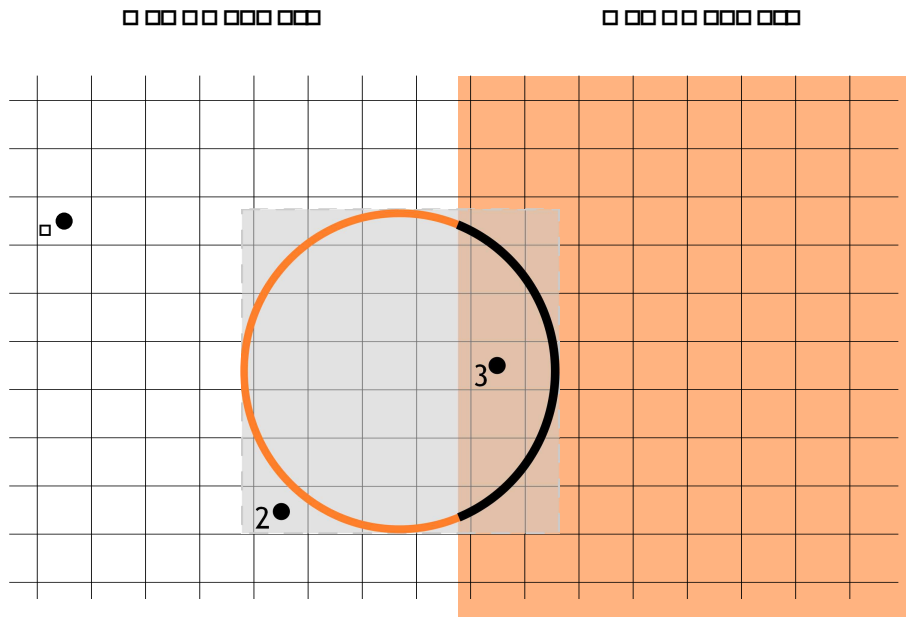


Figure 4.20: Graphical illustration of the point inclusion test on multiple processors. The shaded area represents the global bounding box of the surface.

4.3.2 Calculation of cut-cell properties

MFTL also enables calculation of geometrical parameters of fluid cells cut by a triangulated surface. A two-dimensional example of such a cell is illustrated in Fig. 4.21. Evaluation of the *cut-cell* properties involves finding the surface triangles in the cell and computing the fraction of their area which is inside the cell (green segments in the figure), calculating accurate flow face areas of the cell (n , s , w) and determining the volume fraction of the fluid in the cell (V).

Due to the fact that the processor cuts in the fluid domain do not always match the processor cuts of the triangulated surface, the process of *cut-cell* properties determination requires extensive communication between CPU's. The following procedure is adopted:

1. Loop over the fluid domain and save the *cut-cells* (based on their tagging) in a list
2. Gather the list from all processors, so that each CPU has access to all *cut-cells*
3. Loop over global list of *cut-cells*,
 - a) Find the triangles in each cell, calculate the fraction of the triangle area that falls inside the cell.
 - b) Find the intersection points between the surface and the cell faces.
4. Re-distribute the information about the triangles in the cells and the surface-face intersection points, ensuring that they are available to the processor owning the cell
5. Loop over fluid domain, calculate accurate flow face areas for each *cut-cell*

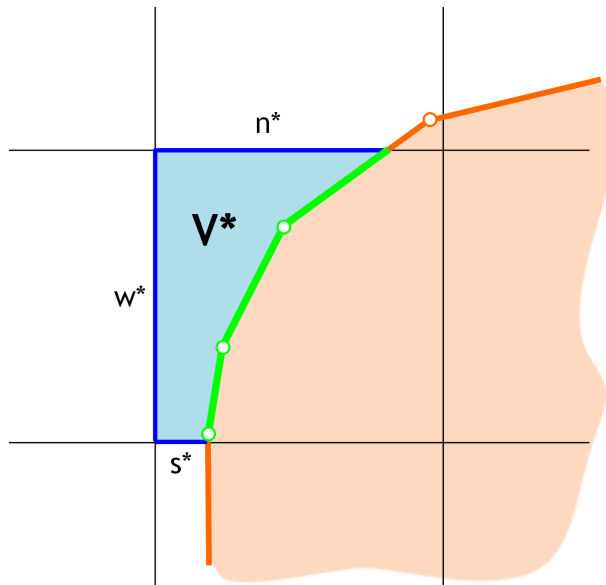


Figure 4.21: A two-dimensional fluid cell cut by a triangulated surface. The blue region represents the modified fluid volume V^* , w , s and n are the modified fluid cell faces, while the green edges correspond to the cut surface triangles.

6. Evaluate the fluid volume fraction in the *cut-cells*

The first two steps of the routine ensure that the *cut-cell* list is available on each processor. Then, each CPU loops over the global *cut-cell* list to find which of its triangles are inside the cell. This is performed by checking whether the oriented bounding box of the triangle overlaps the bounding box of the cell. If this is the case, the function determines how many triangle vertices fall inside the cell. When all vertices are in, the entire triangle is assigned to the cell. Otherwise, the triangle-cell intersection points are found by using the procedure described in section 4.2.4. The triangle-cell intersection points are applied to evaluate the active triangle area in the cell, using the convex hull algorithm (see 4.2.5).

While looping over the global *cut-cell* list, the function also determines the intersection points between the surface and the cell faces. Once all the inside triangles and the particle-cell intersection points are found, the information is redistributed over the processors. As a result, each processor will again have only the data needed for further calculations, *i.e.* the list of triangles with their “active” areas and the particle-face intersection points for each cell owned by the current CPU.

In the fifth step the routine loops over the fluid domain once again. This time, the information from the previous steps is used to evaluate the fluid flow areas in each cell. Similarly as for the cut triangles, the convex hull algorithm is applied for this purpose.

Finally, the accurate flow face areas and the triangle active areas are used to evaluate the volume of

the *cut-cell* according to the formula:

$$\begin{aligned}
 V &= \int_V dV = \frac{1}{3} \int_A (x_f^i - x_c^i) n_f^i dA \\
 &= \frac{1}{3} \int_f (x_f^i - x_c^i) a_f^i + \int_t (x_t^i - x_c^i) a_t^i
 \end{aligned} \tag{4.22}$$

The volume of fluid in the *cut-cell* is calculated as a third of the sum of the scalar product of $(x_f^i - x_c^i)$, the distance from the cell centre to the cell face multiplied by a_f^i , the corrected face area vector. Contribution from the surface triangles has to be added in order to close the surface integral.

5 Validation and performance study

This chapter presents the results of a number of tests performed in order to validate and evaluate the performance of the Immersed Boundary Method implementation designed during the course of the current research project. The study was conducted on several benchmark test cases, ranging from calculation of the forces from the analytical flow field to a simulation of a sphere settling in a fluid under the influence of gravity. Each test was designed to focus on evaluation of a specific property of the developed method.

Every test presented in this chapter is described in a similar fashion. First, a short motivation for the test is given, combined with a general summary of the simulation. Then, the computational setup is presented. Finally, the obtained results for various IB implementations are shown, followed by a discussion and general observations.

All the numerical methods applied during the validation process are presented in detail in chapter 3, where both the single phase discretisation along with the IB implementations are discussed. Unless specified otherwise, the simulation results are obtained with following assumptions and settings applied in the flow equations:

central discretisation scheme of the convective terms

the RHS terms of viscous stress term (equation 3.15) are neglected due to their small size

the influence of the convection is neglected when calculating the d term in the momentum weighted interpolation procedure

the explicit velocity terms (u_e^{jO} , w_e^{jO} in equation 3.35) are neglected when calculating the mass fluxes in the continuity equation

It is also useful to introduce some concepts, that are going to be used in this chapter. Since all the simulations involve flows with spherical bodies, R will represent the particle radius, while D will be its diameter. Additionally the Reynolds number of the simulation is defined as $Re = \frac{\rho DU}{\mu}$, where ρ and μ are the fluid density and viscosity respectively, while U is the characteristic velocity of the simulation described in each case.

The grid refinement level is evaluated by the means of N/D parameter, which represents the number of grid cells per particle diameter. Fluid meshes used in this chapter are equidistant Cartesian grids.

The non-dimensional drag (C_D), lift (C_L) and torque (C_T) coefficients are defined as:

$$C_D = \frac{F_D}{\frac{1}{2} \rho U^2 D^2} \quad (5.1)$$

$$C_L = \frac{F_L}{\frac{1}{2} R^2 U^2} \quad (5.2)$$

$$C_T^i = \frac{T^i}{\frac{1}{2} R^3 U^2} \quad (5.3)$$

F_D is the drag force, F_L the lift, while T^i is the i th component of the torque vector. The drag acts in the positive flow direction along the fluid velocity vector, lift is the transverse component of the total fluid force on the IB. The torque on the other hand is usually acting on the counter-rotational direction.

5.1 Force calculation technique

As discussed in chapter 3, the current Immersed Boundary Method, based on the *ghost cell* technique, solves the flow equations by imposing appropriate boundary conditions on the fluid-particle interface. This is achieved by setting the velocities of the *ghost cells* inside the particle in a way that ensures that the interpolated velocity at the boundary is equal to the body velocity.

The resulting forces and moments acting on the body have to be calculated from the resolved flow field. As this operation is independent from the flow field solution method, it is reasonable to investigate the performance of the force calculating technique using a prescribed analytical flow field. This allows to determine the force calculation accuracy in an ideal case, where the IB method does not introduce additional errors.

The main goal of this test is therefore to compare various force calculation techniques introduced in chapter 3.2.4 and to verify their ability to obtain the force predicted by the analytical solution. Not only the accuracy of the prediction, but also the convergence rate and the dependence on the surface refinement level are going to be investigated. Also, the behaviour of the force components is going to be analysed.

The analytical solution of the flow field past a rotating sphere, has been chosen as the most suitable base for the analysis. The solution of a low Reynolds number flow past a transversely rotating sphere has been derived by Rubinow and Keller [101]. They analytically solved the Navier-Stokes equations by asymptotically matching the flow near the sphere with the undisturbed flow at infinity. As a result an analytical formula for the Magnus lift force was derived.

The viscous flow past a rotating sphere has several properties that make it a good test for calculating the force. Firstly, an analytical solution of the entire flow field is available, therefore both velocities and pressures of the fluid are known everywhere. This allows for exact evaluation of force and moments coefficients acting on the particle. Moreover, the rotation introduces additional effects, namely the Magnus lift force and a counter-rotational torque, which can have much smaller magnitude than the drag. The presence of the additional force components make this set-up more rigorous than for example a Stokes flow past a sphere.

5.1.1 Simulation set-up

The following computational set-up has been chosen for the study of the force calculation technique performance. A sphere of diameter D is placed at the center of cubical flow domain with edge length $a = 1$ m as shown in Fig. 5.1. The fluid domain is discretised by 40 cells in each direction, what results in $\Delta x = 0.025$ m.

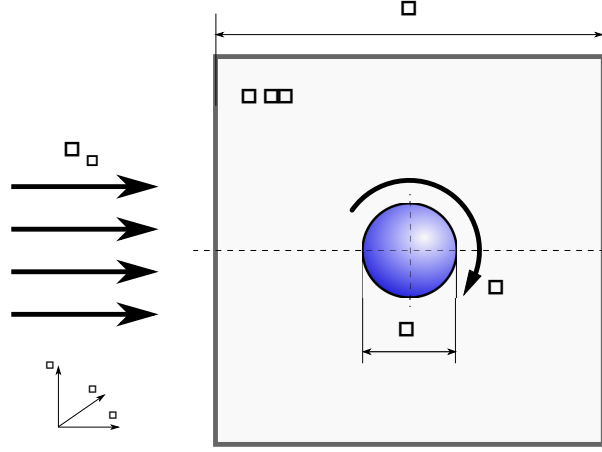


Figure 5.1: A viscous flow past a rotating sphere.

The fluid of density $\rho = 1$ kg/m³ travels in the positive x direction with the free stream velocity $U = 1$ m/s. The position of the sphere is fixed, however it is assumed to rotate with an angular velocity ω , vector of which is co-linear with the y axis. Hence the boundary condition at the IB surface is given by:

$$u_{IB} = \omega \times r \quad (5.4)$$

The non-dimensional parameters describing the flow are the Reynolds number $Re = \frac{U D}{\nu} = 0.5$ and the non-dimensional angular velocity $\Omega = R \omega = 1.5$. The exact values of the fluid velocities and pressure, at each cell in the fluid domain, are calculated using the formulas derived in [101]. The fluid variables inside the body are set to zero. According to the analytical solution the particle will experience a drag force in x direction, rotational lift in z direction and a counter rotational torque in y direction. The analytically determined forces and torque coefficients values are:

$$C_D = \frac{24}{Re} \left(1 + \frac{3}{16} Re \right) \quad (5.5)$$

$$C_L = 2 \quad (5.6)$$

$$C_M = \frac{32}{Re} \quad (5.7)$$

Note that the lift coefficient (C_L) does not depend on the flow Reynolds number, it is a function of the non-dimensional rotation only. For the current set-up the coefficient values, calculated from the

analytical solution, are:

$$C_D = 52.5 \quad C_{DP} = 17.5 \quad C_{DV} = 35.0 \quad (5.8)$$

$$C_L = 3.0 \quad C_{LP} = 1.5 \quad C_{LV} = 1.5 \quad (5.9)$$

$$C_M = 96.0 \quad (5.10)$$

where C_P is the pressure component of the coefficient, while C_V is the viscous part.

Since the number of the grid cells in the domain is constant, the refinement level, N_D , is specified by modifying the radius of the sphere. The angular velocity of the particle and the fluid viscosity need to be adjusted, in order to preserve the constant values of Re and Ω . The specific values used in the simulations are listed in the table 5.1.

Table 5.1: The simulation parameters used for force calculation evaluation.

N_D	6	8	10	12	16	20	24
R [m]	0.075	0.100	0.125	0.150	0.200	0.2500	0.300
μ [Ns/m ²]	0.3	0.4	0.5	0.6	0.8	1.0	1.2
Ω [1/s]	20.0	15.0	12.0	10.0	7.5	6.0	5.0

Apart from the grid refinement study, a set of tests quantifying the influence of the amount of the triangles describing the sphere's surface is performed. Both spheres and ellipsoids in MFTL, are initially generated as unit icosahedra with 20 triangular faces. The surface mesh is then refined by splitting each triangle edge into two new segments, what results in generation of 4 new triangles for every surface face. Finally, the surface is scaled the desired dimensions and moved into the specified location. Four different refinement levels, shown in table 5.2, are considered in this test.

Table 5.2: Sphere refinement level study.

Refinement Level	2	3	4	5
Number of triangles	320	1280	5120	20480
Surface Area (%)	98.12	99.52	99.88	99.97

The surface refinement study was performed with at the grid refinement level being equal to $N_D = 12$. It can be seen that at the 4th level of refinement the surface area is 99.88 % of the corresponding sphere's surface area. In general, this is a sufficient level of accuracy, and is used in other simulations in the present chapter.

Also, it needs to be noted that the refinement level has an effect the local sphere radius. Although the radius from the sphere centre to the triangle vertices is always calculated in a correct way, the radius to the triangle centre is smaller. Once the refinement level is increased, the difference between the spheres radius and the distance from the body centre to the centre of the triangle is decreased.

5.1.2 Results and discussion

The forces and torques obtained from the simulations are converted to the respective coefficients using equations 5.1, 5.2, 5.3. The calculated values are plotted as a function of N/D in Fig. 5.2. The black dashed line corresponds to the coefficient value predicted by the analytical solution. Additionally the pressure and viscous components of the coefficients as a function of the grid refinement level are shown in Figs. 5.3 and 5.4 respectively. Since the mass flux term contribution to the force calculation using the surface around the particle is negligible, it is omitted in the current analysis.

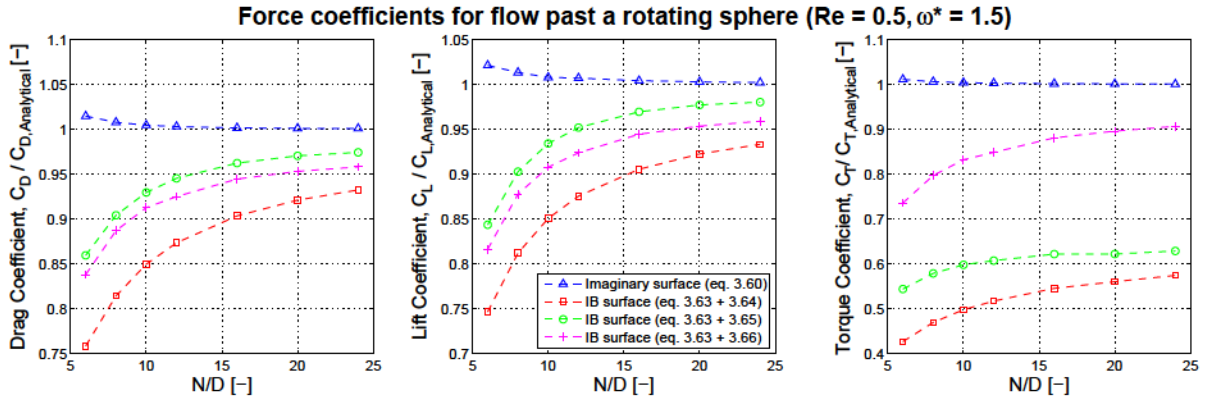


Figure 5.2: Force coefficients for a viscous flow past a rotating sphere at $Re = 0.5$, $\omega^* = 1.5$ as a function of the refinement level N/D . 3-point pressure extrapolation (eq. 3.63) is applied to calculate the pressure contribution when using the IB surface. Viscous term is calculated by: a) first-order normal velocity gradient extrapolation (eq. 3.64), b) second-order normal velocity gradient extrapolation (eq. 3.65), c) least-square fit for the velocity gradient tensor calculation (eq. 3.66).

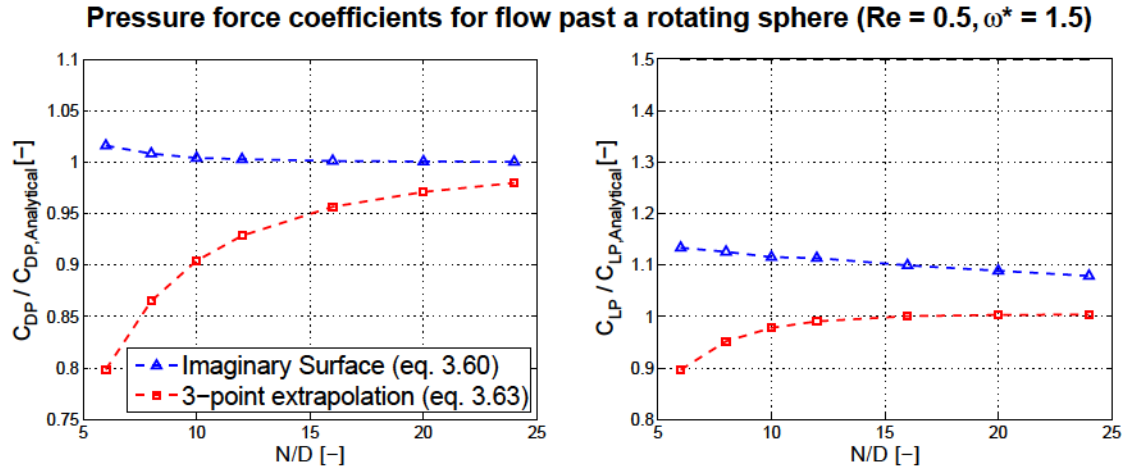


Figure 5.3: Pressure force coefficients for a viscous flow past a rotating sphere at $Re = 0.5$, $\omega^* = 1.5$ as a function of the refinement level N/D .

It can be clearly seen that calculating the force using the imaginary surface around the investigated

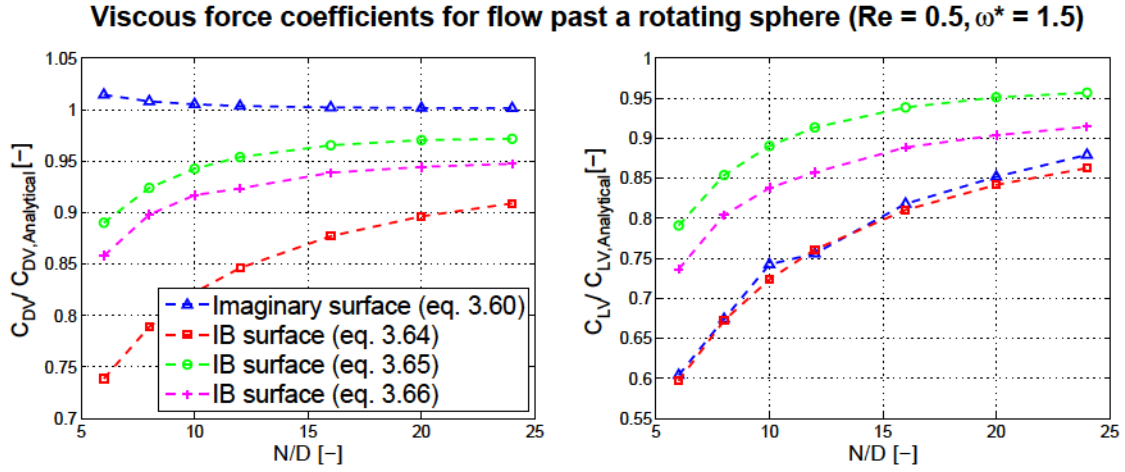


Figure 5.4: Viscous force coefficients for a viscous flow past a rotating sphere at $Re = 0.5, \omega^* = 1.5$ as a function of the refinement level N/D . Viscous term for IB surface is calculated by: a) first-order normal velocity gradient extrapolation (eq. 3.64), b) second-order normal velocity gradient extrapolation (eq. 3.65), c) least-square fit for the velocity gradient tensor calculation (eq. 3.66).

body results in a very good agreement with the theoretical prediction, even for low values of the grid refinement. This is because the imaginary surface technique uses the fluid variables, obtained from the analytical flow field, for the interpolation to the cell faces, therefore it has the same precision as the discretised flow field. On the other hand, methods using the IB surface require combination of interpolation-extrapolation procedure, which add additional inaccuracy. Still, all methods converge to the expected force coefficient values as the number of grid cells per diameter increases. The values obtained using the IB surface, however, remain slightly under-predicted. The method using the second-order normal viscous gradient calculation seems to give the best estimate from all the techniques applying the IB surface directly.

A very important observation can be made by studying the counter rotational torque coefficient values. The techniques applying the normal viscous gradient for torque calculation seem to converge to a different value than expected. In fact the torque predicted by these methods corresponds to two-thirds of the theoretical torque coefficient. Closer inspection of the analytical solution proposed by Rubinow and Keller [101] reveals that even though the viscous forces depend only on the normal velocity gradient, the torque is also influenced by the cross terms of the viscous stress tensor. Moreover, contribution of the non-diagonal terms of the viscous stress tensor correspond to exactly one third of the counter rotational torque. Therefore they need to be taken into account for an accurate torque prediction.

Inspection of the respective force components provides additional information about the performance of the different force calculation techniques. Since the flow is highly non-linear in the vicinity of the particle, all methods using the IB surface struggle to predict the exact force coefficient value, due to the embedded properties of the extrapolation procedure. However, a test where the values of the pressure and the velocity gradient tensor are computed at the surface from the analytical solution (*i.e.* no extrapolation

is used), reveals that the coefficients are equal to the theoretical prediction. This suggests, that the technique can be improved by modifying the extrapolation procedures.

Also, in case of the imaginary surface method, the pressure lift component is over-predicted, while the viscous lift is under predicted (Figs. 5.3 and 5.4). Nevertheless the total value of the lift is evaluated correctly. This is because the respective components are calculated on the imaginary surface and not on the IB directly. Hence, they cannot be treated as the real values of the pressure/viscous terms on the sphere's surface, but only as estimates.

Fig. 5.5 shows the convergence rate of the forces and torques for various calculation techniques, while Figs. 5.6 and 5.7 illustrate the convergence of the respective force components. It can be observed that the relative error, in all cases, decreases with an approximately second-order slope. No clear distinction can be made between the methods in terms of convergence rate of the drag coefficient. Again, the imaginary surface method has much smaller starting error, however its convergence rate is similar to other methods. A different observation can be made for the convergence rates of the lift force components shown in Fig. 5.6. Here, the relative error of the imaginary surface technique is comparable to other methods. However, the convergence rate for the pressure lift component calculated with 3-point extrapolation method is considerably better than for the imaginary surface.

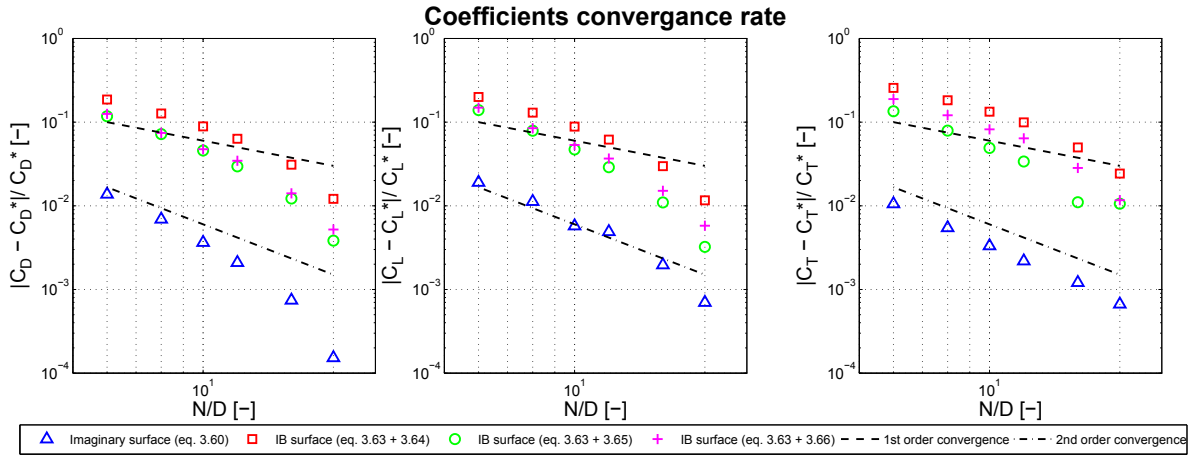


Figure 5.5: Force coefficients convergence rate for a viscous flow past a rotating sphere at $Re = 0.5$, $\Omega = 1.5$ as a function of refinement level N/D . 3-point pressure extrapolation (eq. 3.63) is applied to calculate the pressure contribution when using the IB surface. Viscous term is calculated by: a) first-order normal velocity gradient extrapolation (eq. 3.64), b) second-order normal velocity gradient extrapolation (eq. 3.65), c) least-square fit for the velocity gradient tensor calculation (eq. 3.66).

The results of the influence of the surface mesh refinement study for $N/D = 12$, are presented in Fig. 5.8. It can be easily seen that the imaginary surface prediction is not influenced by the change in the refinement level. On the other hand, the IB surface-based force calculation methods show a certain amount of sensitivity to the surface refinement level. Also, even though the 3rd refinement level corresponding to 1280 surface triangles gives acceptable results, the 4th level is more suitable - a compromise between

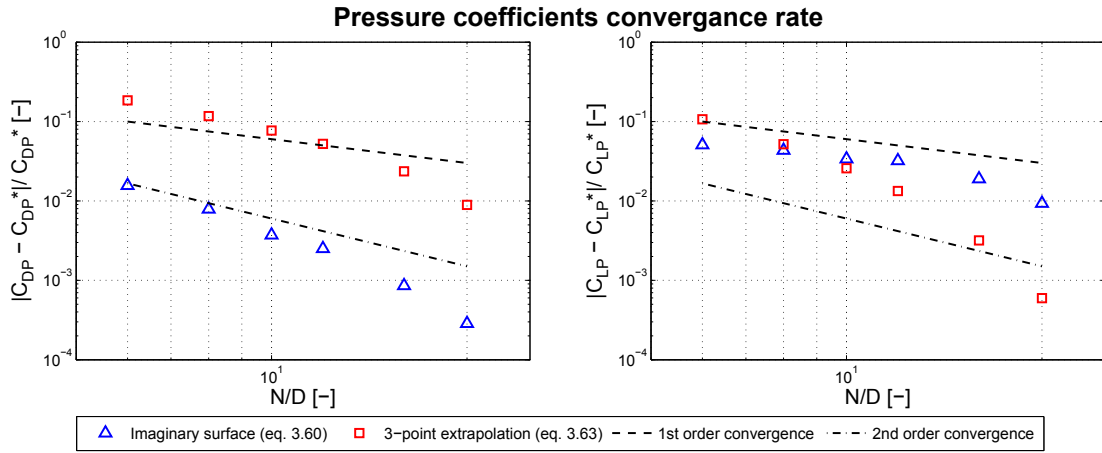


Figure 5.6: Pressure force convergence rate coefficients for a viscous flow past a rotating sphere at $Re = 0.5$, $\Omega = 1.5$ as a function of refinement level N/D .

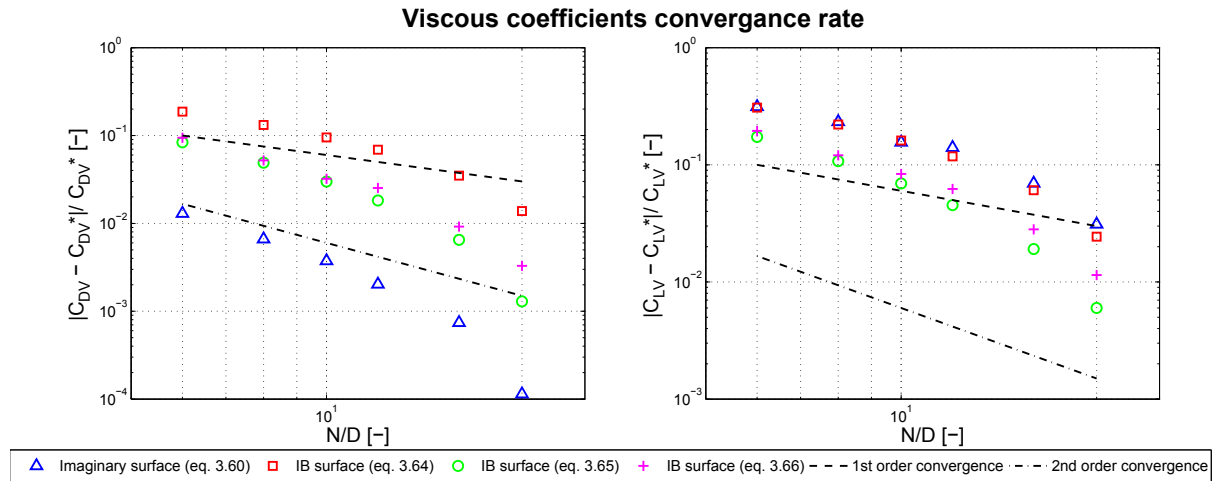


Figure 5.7: Viscous force coefficients convergence rate for a viscous flow past a rotating sphere at $Re = 0.5$, $\Omega = 1.5$ as a function of refinement level N/D . Viscous term for IB surface is calculated by: a) first-order normal velocity gradient extrapolation (eq. 3.64), b) second-order normal velocity gradient extrapolation (eq. 3.65), c) least-square fit for the velocity gradient tensor calculation (eq. 3.66)

the computational cost and the accuracy.

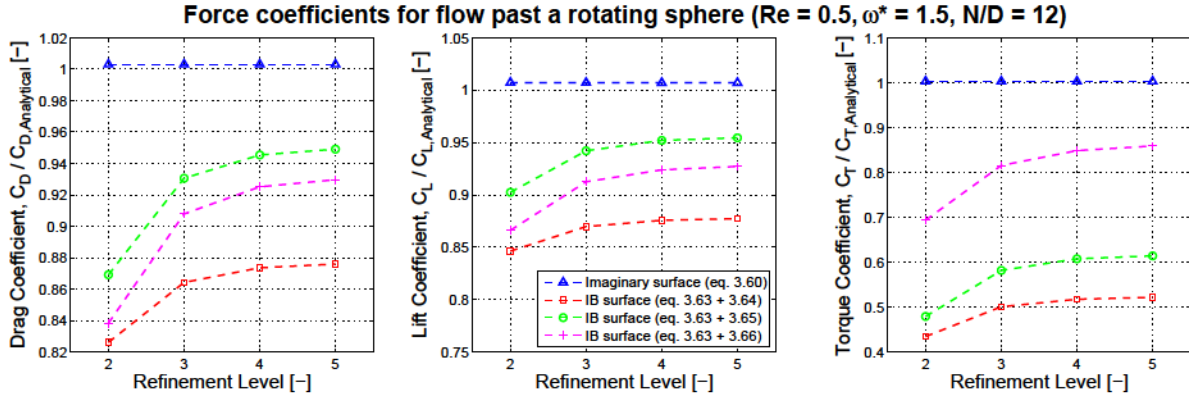


Figure 5.8: Influence of the surface refinement level on the force coefficients ($Re = 0.5$, $\omega^* = 1.5$, $N/D = 12$). Viscous term calculated by: a) first-order normal velocity gradient extrapolation (eq. 3.64), b) second-order normal velocity gradient extrapolation (eq. 3.65), c) least-square fit for the velocity gradient tensor calculation (eq. 3.66)

Summarising, it has been shown that in an analysis of a flow past a stationary object the imaginary surface force calculation technique provides the best results, both in terms of accuracy, as well as in the case of the required surface refinement level. Other investigated techniques, even at the highest refinement level are not able to achieve the same accuracy level, however they still can be used for the force prediction.

Although the viscous force depends mainly on the normal velocity gradient, the torque experienced by the particle has to take the cross terms of the velocity gradient tensor into account as well.

It is believed that the IB surface based force calculation methods can be further enhanced by improving the extrapolation procedure and by eliminating the interpolation step, *i.e.* using the fluid points to directly extrapolate the flow field properties to the body surface. The extrapolation can be performed using a least-square fit of the flow field properties. This approach is being currently under development in the research group and is showing some promising results (Fig. 5.9).

5.2 Flow past a stationary sphere

The second stage of the validation process requires evaluation of the boundary condition enforcement technique. This can be performed by simulating a flow past stationary objects at various Reynolds numbers. Both the accuracy and the convergence rates of different Immersed Boundary approaches are going to be investigated. The ability of the IB implementation to identify various flow features will be also studied.

Although using the analytical solution of the flow is very useful for analysis of the force calculation technique performance, it is not suitable for investigation of the complete IB method behaviour. This is due to the fact, that usually analytical solutions of the free flows assume the boundary conditions at

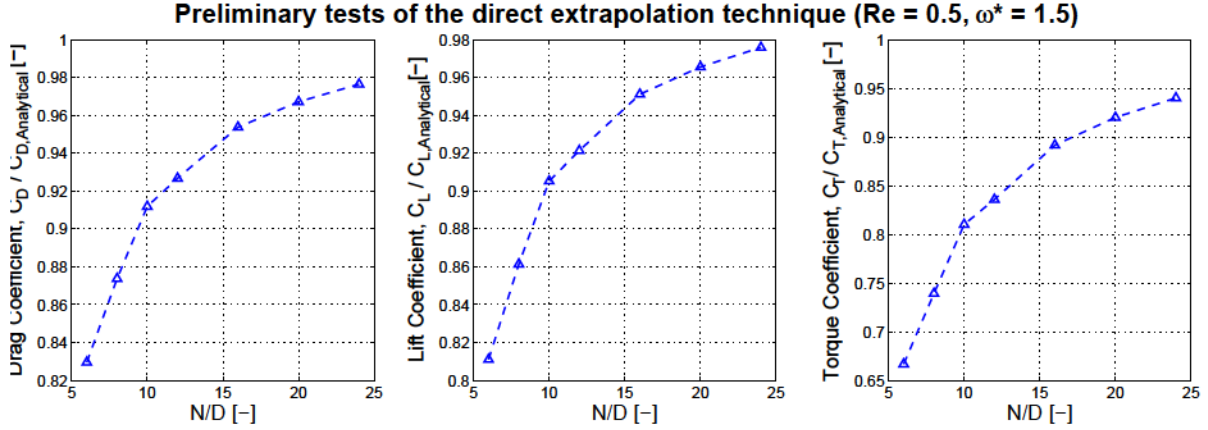


Figure 5.9: Force coefficient obtained by direct extrapolation of the flow variables ($Re = 0.5$, $\omega^* = 1.5$) as a function of refinement level N/D .

infinite distance from the particle, which requires very large and non-practical computational domains. Also, a low Reynolds number ($Re < 1$) is often assumed in the flows solved analytically.

A reasonable test to evaluate the performance of the method could be a comparison against established force coefficient correlations. The main issue with this approach is the fact that the correlations are based on a large set of data and can differ strongly between each other. Additionally, the data for the empirical correlations is usually gathered using various techniques and at different conditions. Generally it can be assumed that the accuracy is satisfactory if the relative difference between the obtained and predicted results is within 5 per cent.

The best approach is, however, to perform a direct comparison with experimental results at known flow conditions. Ten Cate *et al.* [108] proposed an experiment to measure the velocity of a sphere sedimenting in a fluid under the influence of gravity. This experiment has been well documented and will form a basis for further analysis in this chapter.

Although the experiment described in [108] was performed to measure the velocity of the sphere, it can be also used to quantify the force acting on the spherical particle. When the sphere reaches its terminal velocity, its weight is balanced by the buoyancy and drag forces. As both the particle weight and the buoyancy force are known, the drag can be easily determined.

In the experimental set-up the sphere is moving in a stationary fluid, however in this test the fluid is moving and the particle is stationary. Similar flow behaviour can be expected if the Reynolds number of the simulation corresponds to the one measured in the experiment. Still, the boundary conditions of the flow will have a certain influence on the results, therefore the computed force values can be treated only as estimates.

5.2.1 Simulation set-up

The main goal of the analysis presented in this section is to obtain similar conditions for a flow past a stationary sphere as the ones observed in the experiment. The computational domain used for simulations

is illustrated in Fig. 5.10. The fluid container (flow domain) is 160 mm long with a cross-section of 100x100 mm.

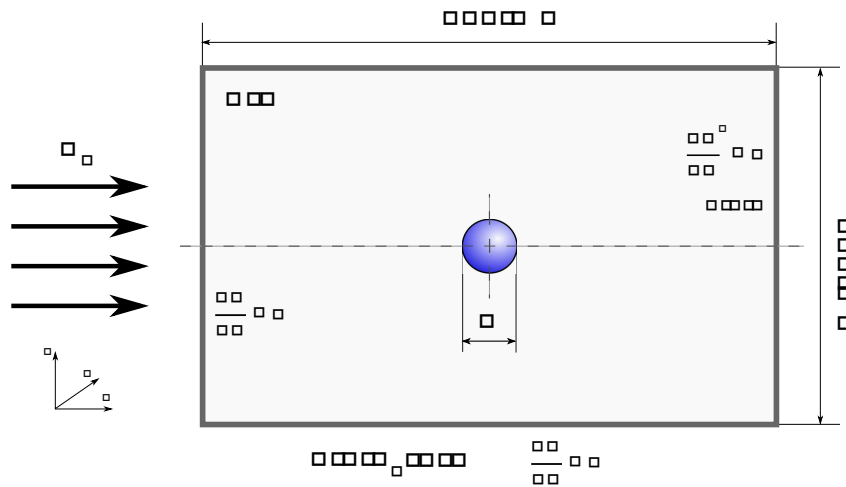


Figure 5.10: Set-up for the flow past a stationary sphere.

The sphere of a diameter $D = 15$ mm is placed on the symmetry axis of the domain, 80 mm from the left boundary. The fluid with density ρ and viscosity μ travels in the positive x direction with velocity U . The boundary conditions are:

$$\text{INLET} - u = (U \ 0 \ 0), \quad p = n = 0$$

$$\text{WALLS} - u = (U \ 0 \ 0), \quad p = n = 0$$

$$\text{OUTLET} - u^j = n = 0, \quad p = 0$$

The present set-up allows to investigate some of the properties of various IB techniques, such as their convergence rate and the flow field structure. All the IB implementations presented in chapter 3 are going to be studied at different grid refinement levels and Reynolds numbers.

Simulations at four grid refinement levels are performed. The number of cells per sphere diameter being $N_D = 6 \ 8 \ 10 \ 12$, while the entire domain is discretised by $64 \times 40 \times 40$, $86 \times 54 \times 54$, $107 \times 67 \times 67$ and $128 \times 80 \times 80$ cells respectively. The Reynolds numbers of the simulations, correspond to the extreme cases studied in the experiment, *i.e.* $Re_1 = 1.5$, $Re_4 = 31.9$. The simulation parameters are then:

$$Re_1 = 1.5 - \rho = 970 \text{ kg/m}^3, \quad \mu = 0.373 \text{ Ns/m}^2, \quad U = 0.038 \text{ m/s}$$

$$Re_4 = 31.9 - \rho = 960 \text{ kg/m}^3, \quad \mu = 0.058 \text{ Ns/m}^2, \quad U = 0.128 \text{ m/s}$$

In order to decrease the time needed for a single simulation, the computation is divided into two steps. First, the flow is solved with a very high time-step ($\Delta t = 10^6$ s) using the upwind discretisation scheme for the convective terms, to obtain a reasonable estimate of the flow field. The obtained results are used as the initial condition in the more accurate simulations with central difference scheme with

$t = 2.5 \cdot 10^{-3}$ s. For every case the flow is allowed to develop for 80 time-steps so that a steady state solution is achieved.

The resulting drag coefficient is calculated using equation 5.1. The force on the sphere is evaluated using both the imaginary surface and the IB surface, with the second-order extrapolation of the normal velocity gradient. The results are compared to the correlations for the drag coefficient presented by Schiller and Neumann [102]:

$$C_D = \frac{24}{Re} (1 + 0.15 Re^{0.687}) \quad (5.11)$$

and by Cheng [8]:

$$C_D = \frac{24}{Re} (1 + 0.27Re)^{0.43} + 0.47 \cdot \exp(-0.04Re^{0.38}) \quad (5.12)$$

Additionally a comparison with a drag coefficient, obtained from the experimental measurements of the settling sphere, is performed. The drag coefficient corresponding to the sphere's terminal velocity is calculated as:

$$C_D = \frac{g(\rho_p - \rho_f) 4 R^3}{1.2 U^2 R^2} \quad (5.13)$$

where $\rho_p = 1120 \text{ kg/m}^3$ is the particle density.

The expected drag coefficients for the investigated Reynolds numbers are summarised in the Table 5.3.

Table 5.3: Predicted drag coefficients.

Re	S& N [102]	Cheng [8]	Experiment [108]
1.5	19.17	18.5	21.00
31.9	1.97	2.06	2.00

5.2.2 Results and discussion

After a rapid adjustment, resulting from the change of the discretisation scheme, a steady state solution is achieved within 80 time-steps, *i.e.* the drag coefficient remains almost constant. Fig. 5.11 shows the time development of the drag coefficient along with the change in the C_D calculated as $C_D = C_D^{n+1} - C_D^n$ for one of the investigated cases. It can be observed that after 0.2 s the change in the C_D value is less than 0.001. Although the obtained values differ slightly depending on the applied force calculation technique, they both reach values similar to the ones predicted by the experiment and the empirical correlations.

The drag coefficients achieved at the final step of each simulation are used in the analysis of the convergence of the methods. Figs. 5.12 and 5.13 show the change of the drag coefficient with the grid refinement for flows at $Re = 1.5$ and $Re = 31.9$ respectively. The figures show the coefficients achieved using the IB surface for the force calculation, as well as the C_D evaluated by means of the imaginary surface. It can be observed that, in most IB implementations, the drag coefficient converge to a single

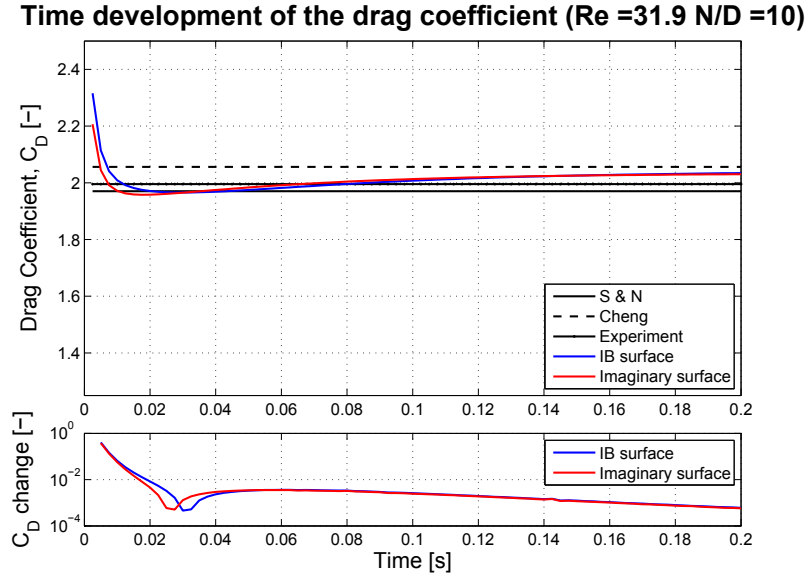


Figure 5.11: Illustration of the time development of the drag coefficient, along with its change ($C_D \text{ change} = C_D^{n+1} - C_D^n$) between time-steps $n + 1$ and n , for a flow past a stationary sphere at $Re = 31.9$.

value as the number of grid cells describing the particle's surface is increased. The convergence is not that clear in the cases of line *a* and *c* where there is no treatment of the continuity equation in the vicinity of the IB. This indicates the importance of modifying the continuity equation in order to prevent the mass flux through the surface. Similarly as in the previous test case, the forces obtained by means of the imaginary surface reach their converged values at low refinement level.

The force predicted for the flow at $Re = 1.5$ is higher than the one predicted by the experiment or the empirical correlations. This can be attributed to the influence of the boundary conditions, as the computational domain is relatively narrow. On the other hand, the forces obtained for the simulations at higher Reynolds number fall within the acceptable bounds specified by the correlations. The modifications applied to the pressure equation seem to have little influence on the magnitude of the predicted forces, especially for the simulations with higher grid refinement level.

It is also worth noting that the values obtained for the line *c*, where the velocity of the cells inside the body is set to IB velocity, are much different than those obtained by the other implementations. This issue will be addressed later on when the exact pressure and velocity profiles are analysed.

Investigation of the pressure and U velocity profiles along the x and y axes shown in Figs. 5.14, 5.15, 5.16 and 5.17, can provide additional valuable information. Apart from the velocity and pressure profiles the figures show the exact location of the IB interface (black lines). The presented profiles correspond to the simulation of the flow at $Re = 31.9$ and $N/D = 10$.

As expected the U velocity outside the body behaves similarly for all investigated cases (Figs. 5.14, 5.15). The no-slip velocity boundary condition is strictly conserved on the IB surface for all the cases. The lines *a* and *c* follow a slightly different pattern, with U velocity being smaller for case *c* than for all

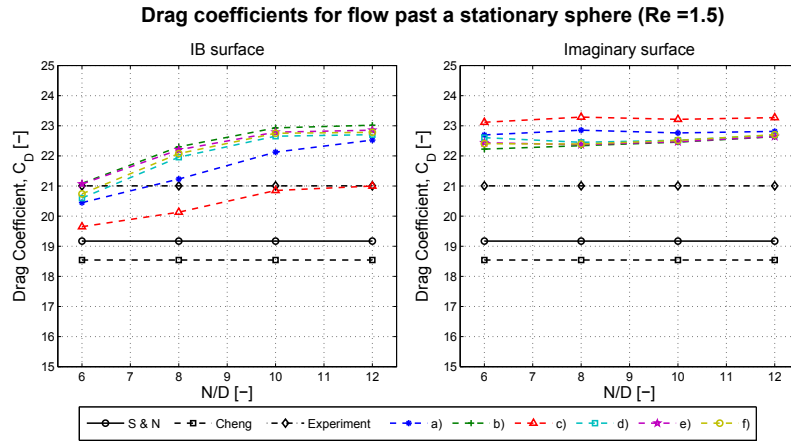


Figure 5.12: Drag coefficients for flow past a stationary sphere ($Re = 1.5$) as a function of the refinement level N/D . a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) velocity inside the body equal to IB velocity (eq. 3.50), d) excluding the *ghost cells* from the continuity (eq. 3.53), e) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), f) continuity solved with *cut-cell* approach (eq. 3.54).

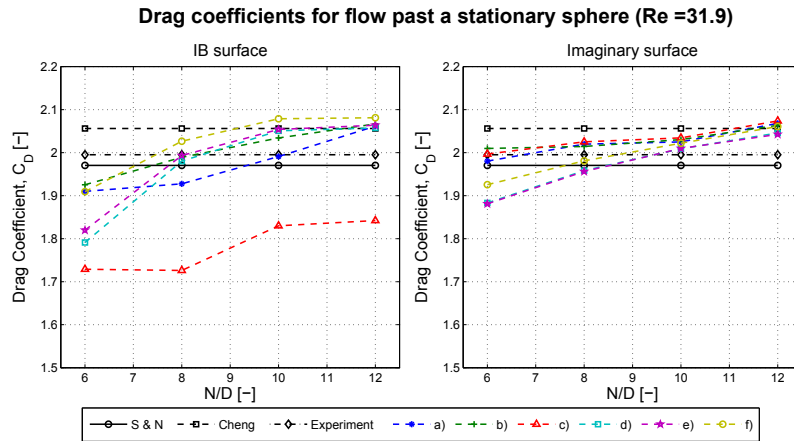


Figure 5.13: Drag coefficients for flow past a stationary sphere ($Re = 31.9$) as a function of the refinement level N/D . a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) velocity inside the body equal to IB velocity (eq. 3.50), d) excluding the *ghost cells* from the continuity (eq. 3.53), e) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), f) continuity solved with *cut-cell* approach (eq. 3.54).

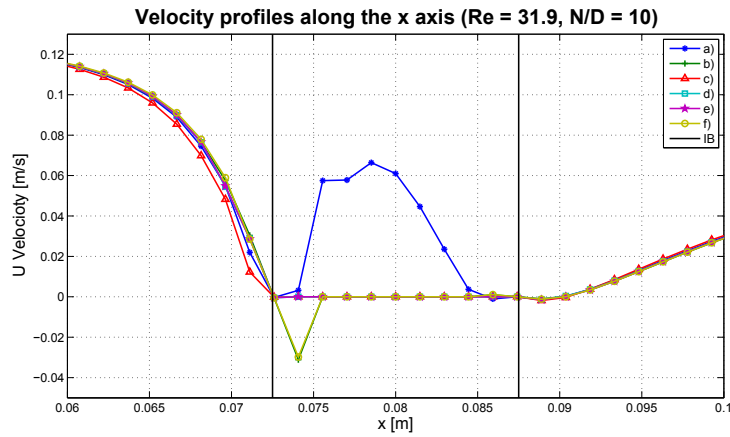


Figure 5.14: U Velocity profiles along the x axis ($Re = 31.9$, $N/D = 10$). The black vertical lines indicate the position of the IB interface. a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) velocity inside the body equal to IB velocity (eq. 3.50), d) excluding the *ghost cells* from the continuity (eq. 3.53), e) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), f) continuity solved with *cut-cell* approach (eq. 3.54).

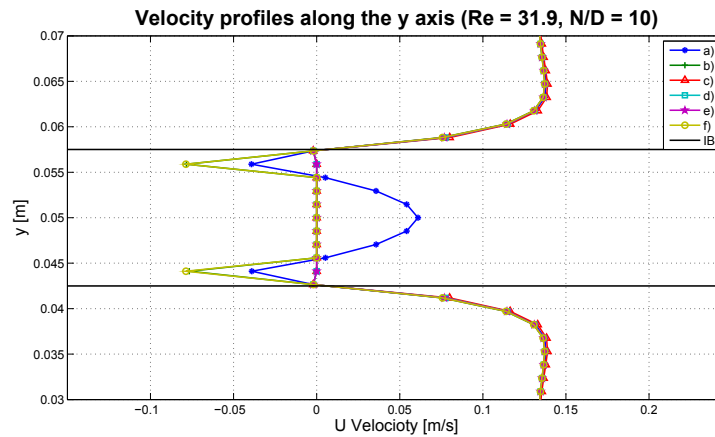


Figure 5.15: U Velocity profiles along the y axis ($Re = 31.9$, $N/D = 10$). The black horizontal lines indicate the position of the IB interface. a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) velocity inside the body equal to IB velocity (eq. 3.50), d) excluding the *ghost cells* from the continuity (eq. 3.53), e) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), f) continuity solved with *cut-cell* approach (eq. 3.54).

other cases.

The velocity behaviour inside the particle differs strongly depending on the implementation. If the momentum equation is solved inside the particle (line *a*), a non-zero velocity is observed inside the IB. Otherwise, the velocities far inside the body are zero, as set by the method. It is also worth noting that the *ghost cell* position for lines *b* and *f* is different from the one observed in other implementations. This is because that in the *b* and *f* implementations the *ghost cells* are defined as the fluid cells inside the body with at least one fluid neighbour in the entire 29-cell surrounding stencil. In other cases only face neighbours are used for determining the cell character.

Study of the pressure profiles leads to a number of interesting observations, as they differ strongly both inside and outside the particle. Closer inspection of the region in front of the particle (Fig. 5.16) reveals the main properties of the investigated IB implementations.

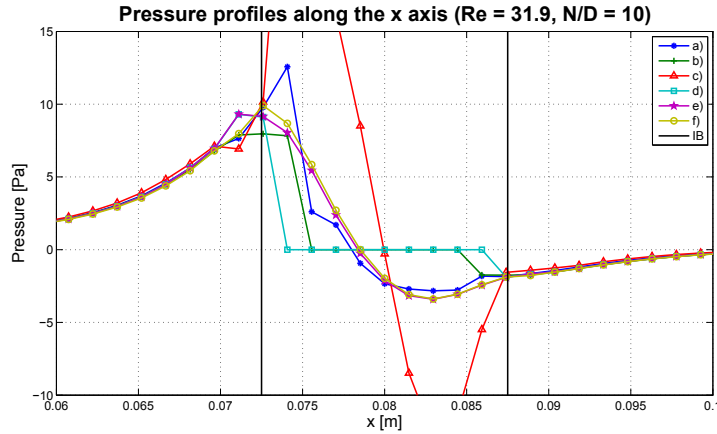


Figure 5.16: Pressure profiles along the x axis ($Re = 31.9$, $N/D = 10$). The black vertical lines indicate the position of the IB interface. a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) velocity inside the body equal to IB velocity (eq. 3.50), d) excluding the *ghost cells* from the continuity (eq. 3.53), e) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), f) continuity solved with *cut-cell* approach (eq. 3.54).

For the case when only the momentum equation in the *ghost cells* is modified (*a*), the profile outside the particle is smooth, with maximum values achieved inside the particle. The pressure far inside the IB adjust to the values required by the momentum and continuity equations.

On the other hand, in the case where the zero-pressure gradient is set on the IB interface (*b*), the pressure profile is no longer smooth in the direct neighbourhood of the particle, which is counter-intuitive. The $\frac{p}{n} = 0$ condition enforces that the pressure of the *ghost cell* is equal to the interpolated pressure at the imaginary point. This requirement however has no effect on the global pressure behaviour. Also, the pressure values far inside the body are equal to zero, though it does not have any affect on the solution, as they are excluded from the flow equations.

The pressures observed in the case *c* profile, are significantly different from the ones observed in the other cases. The values inside the body become very large in order to satisfy the continuity equation,

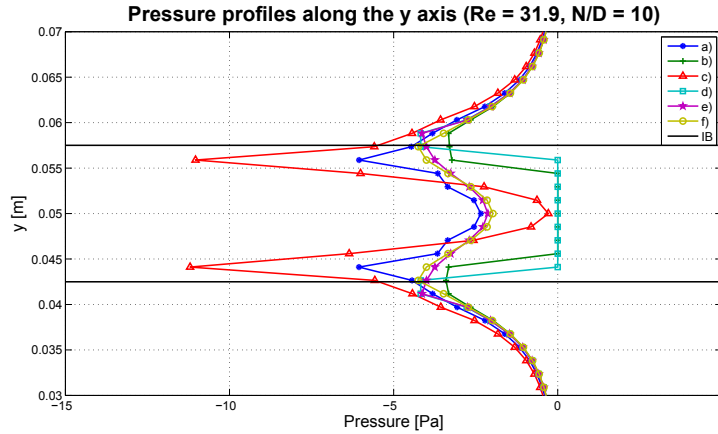


Figure 5.17: Pressure profiles along the y axis ($Re = 31.9$, $N/D = 10$). The black horizontal lines indicate the position of the IB interface. a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) velocity inside the body equal to IB velocity (eq. 3.50), d) excluding the *ghost cells* from the continuity (eq. 3.53), e) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), f) continuity solved with *cut-cell* approach (eq. 3.54).

where the velocities are set to zero. Moreover, there is a non-physical drop of pressure ahead of the sphere. Inspection of this profile explains why the force values calculated by extrapolation of the flow properties to the IB surface differ strongly from the imaginary surface prediction. Since the extrapolation techniques use multiple points to calculate the pressure at the IB surface, they are more sensitive to local discontinuities that corrupt the flow field. At this stage it can be concluded that setting the velocity inside the body equal to IB velocity is not a reasonable approach and will not be pursued any further.

Profiles d and e have nearly identical behaviour in the fluid region, however they differ inside the body, since d excludes both the pressures and velocities inside the body from the continuity equation. Two important observations can be made for these approaches. First, the pressure inside the body has a very limited effect on the continuity equation, which is governed mainly by the velocity flow field. Secondly, excluding the velocities of the *ghost cells* from the continuity equations leads to the increased values of the pressure ahead of particle. The values of the pressures inside the body remain small. Also, the pressure profile is not smooth in the vicinity of the body.

The pressure profile obtained with the *cut-cell* approach for the continuity equation (f) is very similar to a and b . It however remains smooth until it reaches the IB interface. Here, the *ghost cell* pressure adjusts to ensure that there is no velocity flux through the IB. Contrary to the other methods, (f) solves the continuity only for the fluid part of the *ghost cells*, ensuring mass conservation independent of the underlying grid.

The influence of the position of the imaginary point IP, used to set the *ghost cell* velocity, has also been investigated. Figs. 5.18 and 5.19 show the drag force for the flow past a stationary sphere at different grid refinement levels. The flow equations are solved using the *cut-cell* approach for the continuity equation.

Changing the position of the imaginary point applied for setting the velocity boundary conditions,

5.2. FLOW PAST A STATIONARY SPHERE

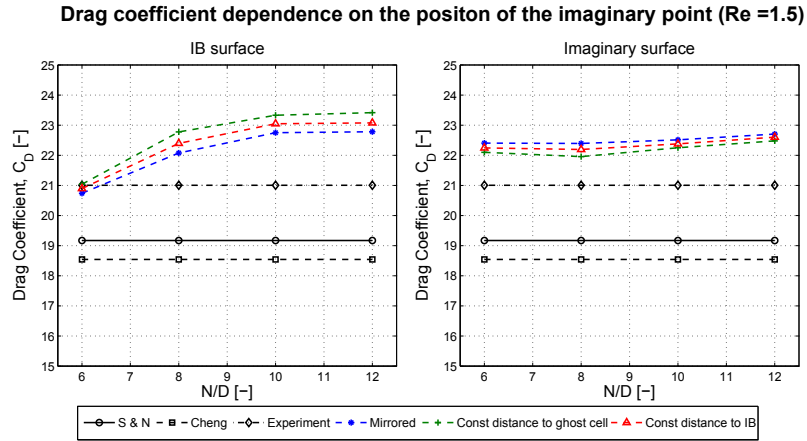


Figure 5.18: Drag Coefficient as a function of the refinement level N/D for different imaginary point positions ($Re = 1.5$). Continuity solved with *cut-cell* approach.

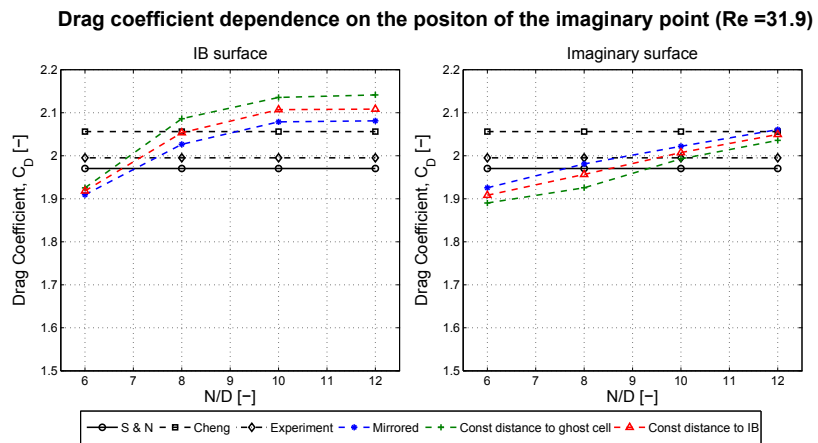


Figure 5.19: Drag Coefficient as a function of the refinement level N/D for different imaginary point positions ($Re = 31.9$). Continuity solved with *cut-cell* approach.

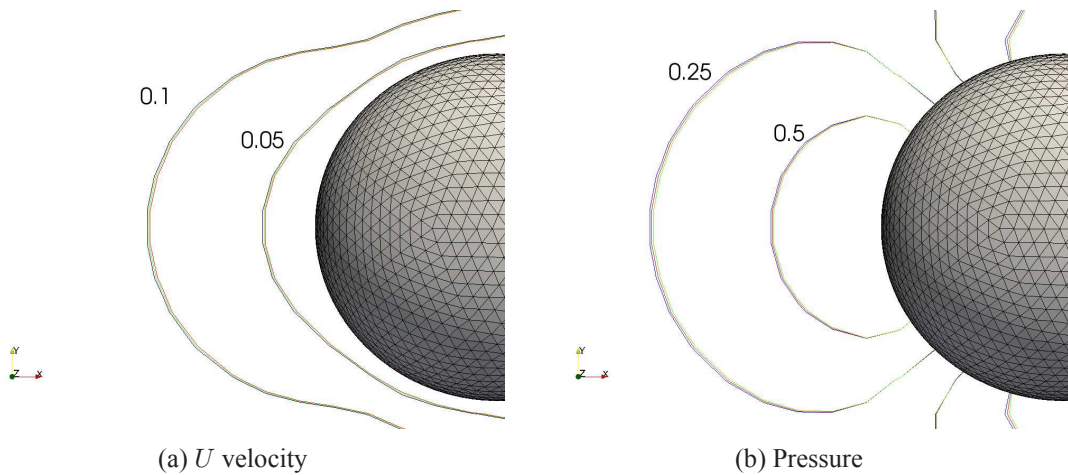


Figure 5.20: Contours of U velocity and the pressure ahead of the sphere ($Re = 31.9$, $N/D = 10$) for different positions of the imaginary point. **blue** - IP mirrored, **red** - constant distance to the *ghost cell*, **green** - constant distance to the IB.

has a small, but visible effect on the magnitude of calculated forces. The drag predicted when the IP is at the constant distance from the IB is higher than when the IP is a mirrored projection of the *ghost-cell*. Even higher values are achieved if the IP point is placed at the constant distance from the *ghost cell*. Interestingly an opposite behaviour is observed when the imaginary surface is applied for the force evaluation.

Investigation of the U velocity and the pressure contours around the particle (Fig. 5.20) reveals little more information. Although a similar pattern, as in the case of the force calculation is observed, the flow behaviour is nearly identical for all the cases. The constant pressure and velocity lines for the mirrored IP are at the largest distance from the sphere surface. The red line representing the values obtained by a fixed distance between *ghost cells* and the imaginary points are the closest. Nonetheless, no definite conclusion on the effect of the position of the imaginary point on the performance of the method can be made at this stage.

Summarising, the study of the flow past a stationary sphere results in a number of observations. Treatment of the continuity equation is necessary to ensure smooth convergence of the force magnitude with the grid refinement. Setting the velocity inside the body without appropriate modification of the continuity equation leads to a non-physical pressure behaviour in the vicinity of the particle. All other methods are able to calculate the force within an acceptable level of accuracy, although they differ in the behaviour of the pressure close to the body boundaries. Further tests are going to be applied in order to evaluate other properties of the investigated IB implementations.

5.3 Oscillating sphere - spurious pressure oscillations

One of the main challenges faced by the discrete forcing Immersed Boundary Methods, such as the one developed in the current research is the presence of spurious pressure oscillations observed in the

simulations with moving particles. According to Seo and Mittal [103], these oscillations can be attributed to the violation of the geometrical conservation law as discussed previously in chapter 2. Seo and Mittal investigated the issue by simulating a cylinder oscillating in a fluid. A similar approach is adopted here to evaluate the magnitude of the oscillations and to quantify the possible improvements.

Instead of simulating an oscillating cylinder, an oscillating sphere is analysed, as this case is of more interest from the particulate flow point of view. The purpose of the simulations presented in this section is to investigate the general flow behaviour - identify the source of the oscillations and quantify their magnitude.

5.3.1 Simulation set-up

The set-up applied for the simulations presented in this section is derived from the configuration proposed by Seo and Mittal [103] to investigate the pressure behaviour in the case of oscillating cylinder in 2D. In this work sphere with diameter $D = 0.25$ m is placed at the center of the cubical flow domain of size 1 m, as shown in Fig. 5.21. The domain is discretised by 40 or 80 cells in each direction resulting in $N/D = 10$ or $N/D = 20$ cells per particle diameter.

A zero pressure and zero normal velocity gradient boundary conditions are applied at both the left and right-hand side walls of the domain. No-slip velocity and zero pressure gradient boundary conditions are enforced on all other fluid domain boundaries.

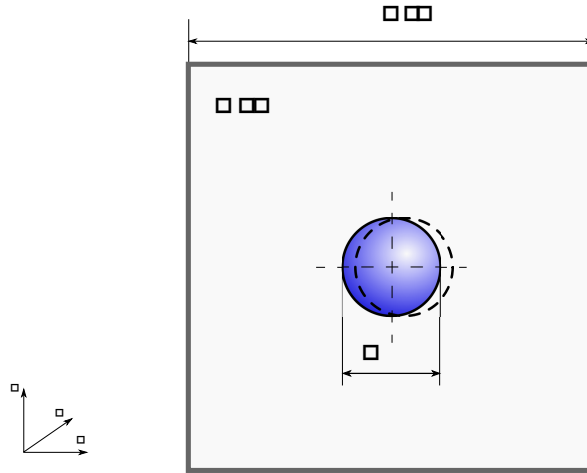


Figure 5.21: Set-up for the simulation of an oscillating sphere.

The oscillatory motion of the sphere in the x direction is described by following equations:

$$x_{IB}(t) = x_{IB}(0) + X_o [1 - \cos(2\pi f_o t)] \quad (5.14)$$

$$u_{IB}(t) = U_o \sin(2\pi f_o t) \quad (5.15)$$

where $x_{IB}(t)$ and $u_{IB}(t)$ are the x components of the body position and velocity at time t respectively.

f_o is the oscillation frequency, while $U_o = 2 f_o X_o$. The maximum distance travelled by the sphere X_o is specified to be $X_o = 0.125D$. Similarly as in the original configuration ([103]), the Reynolds number and Strouhal number are specified as $Re = U_o D = 31$, $St = f_o D / U_o = 3.2$.

The frequency f_o is set to be $f_o = 8$ Hz, hence the maximum velocity is $U_o = 0.6283$ m/s, while the fluid viscosity and density are $\mu = 0.05067$ Ns/m² and $\rho = 10$ kg/m³ respectively. The simulation is performed with three different time-steps Δt and at two refinement levels $N/D = 10$ and $N/D = 20$. Table 5.4 shows the time-steps used and corresponding maximum $CFL = U_o \Delta t / \Delta x$ numbers. The proposed set-up is sufficient to observe easily identifiable oscillations in the force behaviour.

Table 5.4: Time-steps for simulations of an oscillating sphere.

N/D	10			20		
Δt [s]	0.0025	0.00125	0.000625	0.0025	0.00125	0.000625
CFL	0.0628	0.0314	0.0157	0.1257	0.0628	0.0314

The oscillations are quantified by means of two approaches. Seo and Mittal [103] proposed evaluating the oscillations using a 2nd discontinuity defined as:

$$C_p^2 = C_p^{n+1} - 2C_p^n + C_p^{n-1} \quad (5.16)$$

where C_p^i is the pressure drag coefficient at time i , which is calculated as:

$$C_p = \frac{F_P}{\frac{1}{4} \rho R^3 f_o^2} \quad (5.17)$$

Additionally a Fast-Fourier transform is applied to evaluate the relevant frequencies of the pressure oscillations.

The forces are calculated directly on the IB surface. The pressure component is evaluated using the 3-point pressure extrapolation, while the viscous stress term is obtained by means of second-order normal velocity gradient extrapolation.

5.3.2 Results and discussion

Both the influence of the grid refinement and the time-step size on the magnitude of pressure oscillations are analysed for every IB implementation studied. An example result, showing the pressure coefficient as the function of the period fraction for various time-steps and grid refinement levels, when the zero normal pressure gradient is enforced at the surface, is illustrated in Fig. 5.22. As expected, the pressure oscillations intensify as the time-step size is decreased and almost disappear for the heavily refined case. Also, the pressure drag coefficient is noticeably higher when N/D is increased, while a phase shift is observed between high and low values of Δt .

An in depth study of the pressure behaviour can be performed by plotting various parameters as the

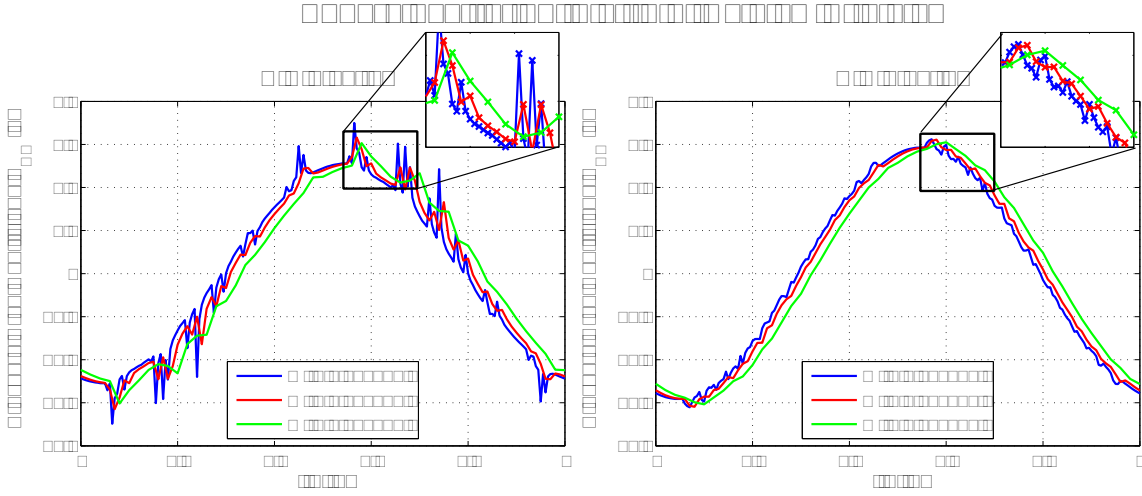


Figure 5.22: The pressure forces acting on the oscillating sphere as the function of the period fraction. A zero normal pressure gradient is enforced at the IB surface.

function of the period fraction. Fig. 5.23 illustrates not only the pressure coefficient, but also shows when the *fresh* and *dead* cells occur. Additionally, the graphs of instantaneous pressure discontinuity C_P^2 and the relative volume error are plotted. The volume error is calculated as:

$$v = \frac{V_{IB} - V_{IN}}{V_{IB}} \quad (5.18)$$

where V_{IB} is the exact IB volume, while V_{IN} is the sum of the volume of the fluid cells that are inside the particle.

It is clear from the Fig. 5.23 that the oscillations and their amplitude are linked to the cell transition, *i.e.* cells changing their behaviour in time. The exact correlation between those events is however not always clear, as sometimes the C_P^2 has higher values when the *fresh* cells emerge, while in other cases the oscillations are bigger when the cells are absorbed by the IB. Moreover, the magnitude of oscillations is not linked directly to the amount of cells changing their behaviour. Additionally, there is no correlation between the the volume error, which can reach 10 % of the sphere's volume, and the magnitude of the pressure oscillations.

The spectral analysis of the C_{DP} illustrated in Fig 5.24 reveal that the high amplitude oscillations occur at frequency $f = 8$ Hz, which corresponds to the frequency of the sphere's motion. Still, a non-zero signal is observed across the higher range of the frequency spectrum.

Comparison of the pressure oscillations observed in various IB implementations can be performed by analysis of the root-mean-square (RMS) of the pressure discontinuity, C_P^2 , or the RMS of the high frequency (> 8 Hz) oscillations. Fig. 5.25 illustrates the root-mean-square of the C_P^2 as the function of the time-step size (left plot, $t_o = 2.5$ ms) and the refinement level. All the investigated methods display similar behaviour. The C_P^2 magnitude increases as the t decreases, but it becomes smaller as the grid is refined.

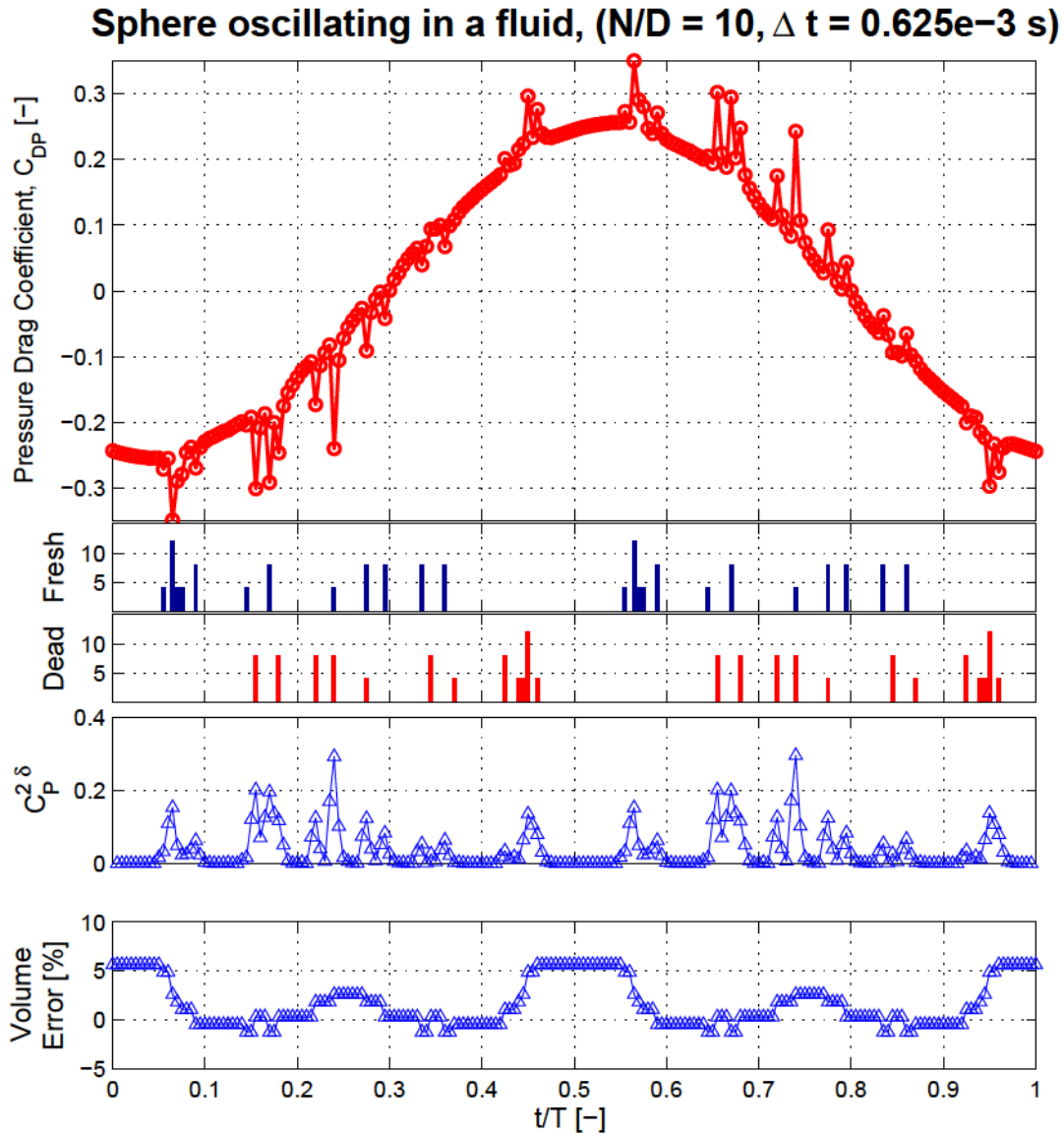


Figure 5.23: Pressure drag coefficient, number of *fresh/dead* cells, instantaneous pressure discontinuity and relative volume error as a function of the period fraction for a sphere oscillating in a fluid ($N/D = 10, \Delta t = 0.625$ ms). A zero normal pressure gradient is enforced at the IB surface.

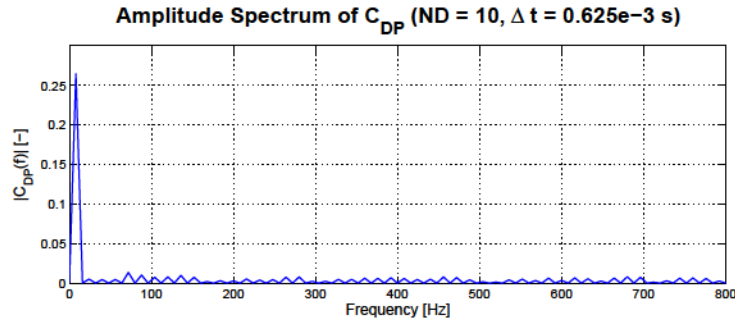


Figure 5.24: Amplitude spectrum of the C_{DP} . A zero normal pressure gradient is enforced at the IB surface.

It is clear that appropriate treatment of the continuity equation has a critical role in decreasing the magnitude of oscillations. In the case *a*, where only the *ghost cell* velocity is set, in the worst scenario the oscillations are almost of the same order as the predicted C_{DP} . The zero pressure gradient approach (*b*), investigated so far, has also a relatively high level of the oscillations, albeit smaller than for case *a*. Exclusion of the *ghost cell* velocities or the entire cells from the continuity equation (*d* and *c*) reduces the $C_P^{2\delta}$ even further, although no difference is observed between those methods. Finally, the *cut-cell* solution of the continuity equation (*e*) results in an approximately an order of magnitude decrease in the oscillations as compared to case *a*.

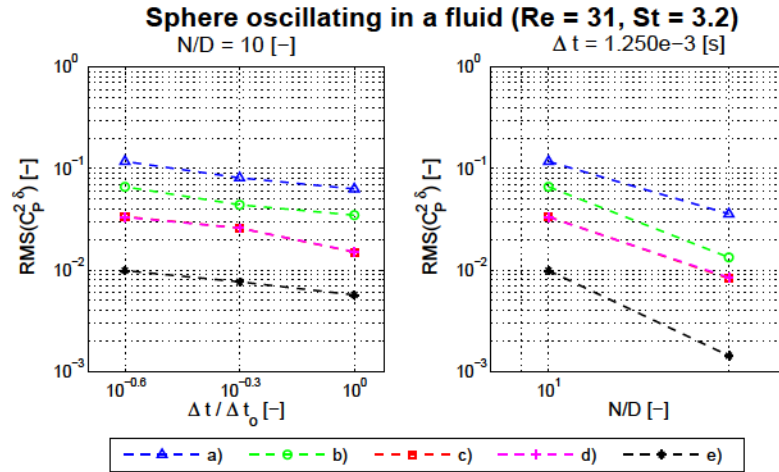


Figure 5.25: Comparison of the RMS of $C_P^{2\delta}$ for different IB implementations as a function of the relative time-step $\Delta t/\Delta t_o$ at $N/D = 10$ (left) and the refinement level N/D at $\Delta t = 1.25$ ms (right). a) IB setting the *ghost cell* velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) excluding the *ghost cells* from the continuity (eq. 3.53), d) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), e) continuity solved with *cut-cell* approach (eq. 3.54).

Analogous observations can be done by studying the RMS of the high frequencies C_{DP} amplitudes presented in Fig. 5.26. The oscillations decrease with the increasing time-step is nearly linear in the log-log scale. The Fig. 5.26 shows that the *cut-cell* approach (*e*) not only significantly reduces the oscillations,

but it also ensures that they disappear more quickly as the grid is refined.

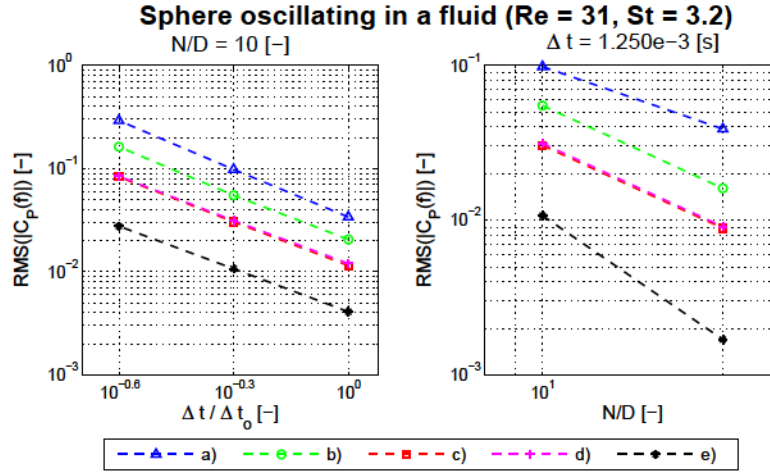


Figure 5.26: Comparison of the RMS of the high frequency C_{DP} oscillations as a function of the relative time-step $\Delta t/\Delta t_o$ at $N/D = 10$ (left) and the refinement level N/D at $\Delta t = 1.25$ ms (right). a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) excluding the *ghost cells* from the continuity (eq. 3.53), d) excluding the velocities of *ghost cells* from the continuity (eq. 3.52), e) continuity solved with *cut-cell* approach (eq. 3.54).

It can be concluded, from the analysis presented above, that the *cut-cell* treatment of the continuity equation (e) is the most effective oscillations reduction technique. If Fig. 5.27 is compared to Fig. 5.22, a clear improvement can be seen. Pressure oscillations are indistinguishable at the higher refinement level. They are still present for the small Δt case at $N/D = 10$, but are much smaller than those observed in Fig. 5.22.

Even though significant reduction of the $C_P^{2\delta}$ is achieved, some oscillations are still present. Similarly as before the source of these oscillations may be identified with the help of Fig 5.28, where the occurrence of the *fresh/dead* cells is marked along with the instantaneous values of $C_P^{2\delta}$ and relative volume error.

Although the low grid refinement level simulation with a small time-step is considered, the magnitude of the oscillations remains small. Also the relative volume error is almost zero throughout the simulation, and it never exceeds 0.01 %. Contrary to the previously analysed case, for the current case the pressure oscillations can be uniquely associated with the occurrence of the dead cells. A repeating pattern can be also seen for various combinations of the cell transitions. Still there is no direct correlation between the amount of *ghost cells* and the magnitude of $C_P^{2\delta}$.

These observations seem to indicate that the remaining oscillations are associated with the treatment of the explicit convective term in the discretised momentum equation, *i.e.* the magnitude of the mass flux calculated from the previous time-step solution with equation 3.5. Even though the mass flux term is recalculated after the body is moved, its exact value may still be inaccurate, what may lead to pressure oscillations. Initial study of this property has been performed and it has been observed that the flow solution is very sensitive to any modifications applied to the explicit mass flux term, however an appropriate

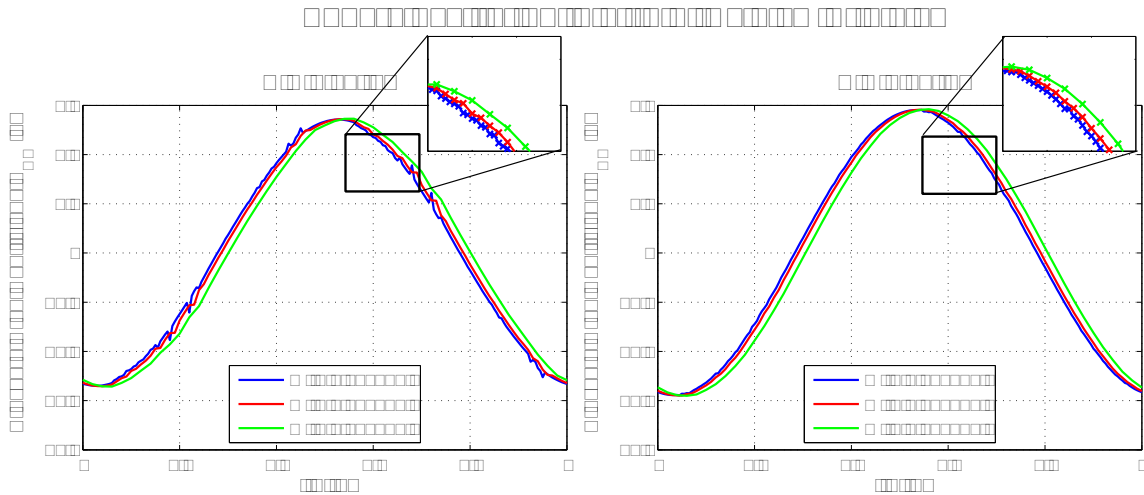


Figure 5.27: The pressure forces acting on the oscillating sphere as the function of the period fraction. Continuity equation solved with the *cut-cell* approach.

treatment is yet to be determined.

Apart from the aforementioned analysis the influence of various *fresh* cell treatment techniques has been investigated. Since the oscillations are no longer observed in Fig. 5.28 when the *fresh* cells emerge into the fluid, it can be expected that the effect of *fresh* cell treatment will be small, and that in fact is the case for the oscillating sphere. Still the interpolation of the velocity to the *fresh* cells is suggested as enhances the prediction of the “old” velocity in those cells.

Additionally the effect of repositioning the imaginary point IP, used to set the *ghost cell* velocity has been studied and shown that there is very little change in the pressure behaviour depending on the position of the IP.

The main conclusion from this test is that the pressure oscillations can be decreased by applying an appropriate treatment to the continuity equation. Even though a significant reduction of the C_P^2 is obtained, there is still possibility for improvement, as small oscillations are still observed when the fluid cells change their character.

5.4 Instantaneously accelerated sphere

Simulation of a instantaneously accelerating sphere is the fourth stage of the validation process. This is the first test case, where the direct comparison with the experimental force values can be performed. Whereas the simulation of the oscillating sphere was designed to investigate only the pressure oscillations, here both the magnitude of the pressure oscillations and the accuracy of the force prediction can be evaluated.

The experimental set-up proposed by ten Cate *et al.* [108] is used as a benchmark for the comparison. A sphere, initially at rest is instantaneously accelerated to the velocity equal to the terminal velocity achieved in one of the experiments presented in [108]. The drag force obtained from the simulation can

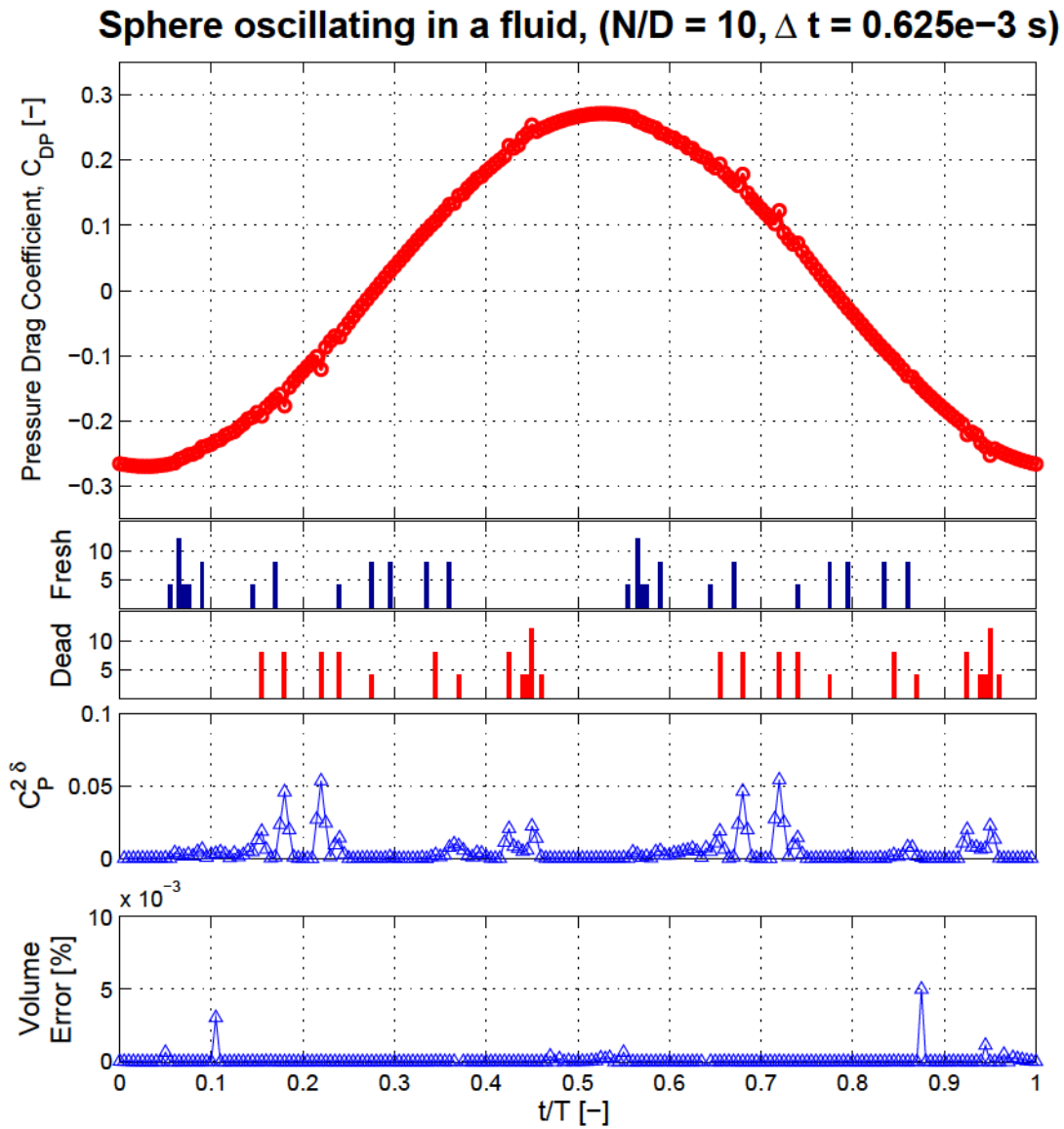


Figure 5.28: Pressure drag coefficient, number of *fresh/dead* cells, instantaneous pressure discontinuity and relative volume error as a function of the period fraction for a sphere oscillating in a fluid ($N/D = 10, \Delta t = 0.625$ ms). Continuity solved with *cut-cell* approach.

be therefore directly compared to the drag derived from the experimental measurements of the terminal settling velocity (equation 5.13).

The results of this test enable comparison of the different IB implementations both in the terms of their ability to suppress the pressure oscillations and their capability to predict the force experienced by a particle moving in a fluid.

5.4.1 Simulation set-up

In the test presented in this section a sphere of a diameter $D = 15$ mm has an initial position $(50\ 50\ 50)$ mm as illustrated in Fig. 5.29. The fluid domain is discretised by $107 \times 67 \times 67$ grid cells, what results in approximately $N/D = 10$, *i.e.* 10 cells per particle diameter. The sphere, initially at rest is instantaneously accelerated to velocity $U_{IB} = 0.128$ m/s in the positive x direction. The U_{IB} velocity is fixed throughout the simulation. The fluid density and viscosity are specified according to data from [108], and are $\rho = 960$ kg/m³ and $\mu = 0.058$ Ns/m² what results in $Re = 31.9$.

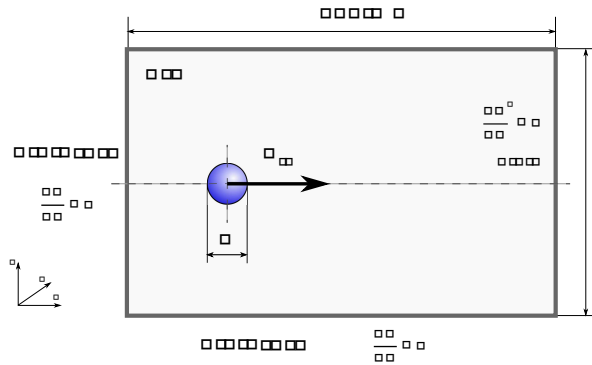


Figure 5.29: Instantaneously accelerated sphere set-up.

During the simulation, the particle travels distance $x_{IB} = 0.0512$ m, *i.e.* more than the half of the fluid domain. Three different time-step values are investigated for each IB implementation tested. The exact values and the corresponding particle CFL numbers are listed in the table 5.5 below.

Table 5.5: Time-steps for simulations of a suddenly accelerated sphere.

t [s]	0.00125	0.0025	0.0050
CFL	0.1072	0.2144	0.4288

A velocity no-slip and zero pressure gradient boundary conditions are applied at the walls, as well as at the inlet of the domain. Zero normal velocity gradient along with pressure equal to zero boundary conditions are specified at the outlet. The fluid is initially at rest. Similarly as before the flow equations are solved using the central discretisation scheme for the convection term in the momentum equation, while the force is calculated using the 3-point pressure extrapolation, and the second-order normal velocity gradient extrapolation

5.4.2 Results and discussion

The drag coefficient experienced by the sphere moving at a constant velocity throughout the simulation is shown, as a function of time, in Fig. 5.30. The time-step of the simulation is set to a low value, with a corresponding CFL number equal to 0.1072. The results obtained by four different IB implementations are compared to each other and the experimental prediction ($C_{D\ exp} = 2.0$).

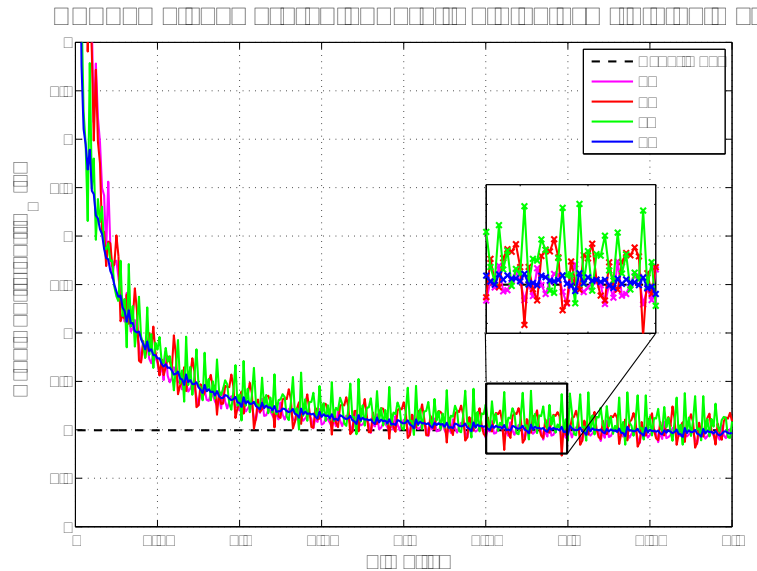


Figure 5.30: Drag coefficient experienced by an instantaneously accelerated sphere as a function of time ($t = 1.25$ ms). a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) excluding the *ghost cells* from the continuity (eq. 3.53), d) continuity solved with *cut-cell* approach (eq. 3.54).

As seen in Fig 5.30, all methods are able to predict the expected value of the drag coefficient within an acceptable level of accuracy (5%). After initial adjustment a constant mean value is achieved. A certain degree of fluctuations is observed in all results. Still, the oscillations of the drag in the solution, obtained through solving the continuity equation with the *cut-cell* approach (line *d*), are considerably smaller than in other cases. In the worst case scenario, case *c*, the magnitude of the oscillations may reach up to 20% of the nominal drag.

Increasing the simulation time-step from $t = 0.125$ ms to $t = 5$ ms leads to a reduction of the oscillations as shown in Fig. 5.31. Similarly as before, the drag coefficient development in time for various IB implementations is presented. Again, the smallest fluctuations are observed in the case where the continuity equation is solved using the *cut-cell* approach (*d*). On the other hand, the behaviour of the oscillations for other implementations is different from the one observed in the case of an oscillating sphere.

Investigation of the RMS of the C_D^2 , defined by equation 5.16, for the tested IBM implementations reveals that the fluctuation reduction with increasing time-step is no longer as clear as before. Although the *cut-cell* approach (*d*) still results in the smallest oscillation rate, the reduction of C_D^2 is no longer

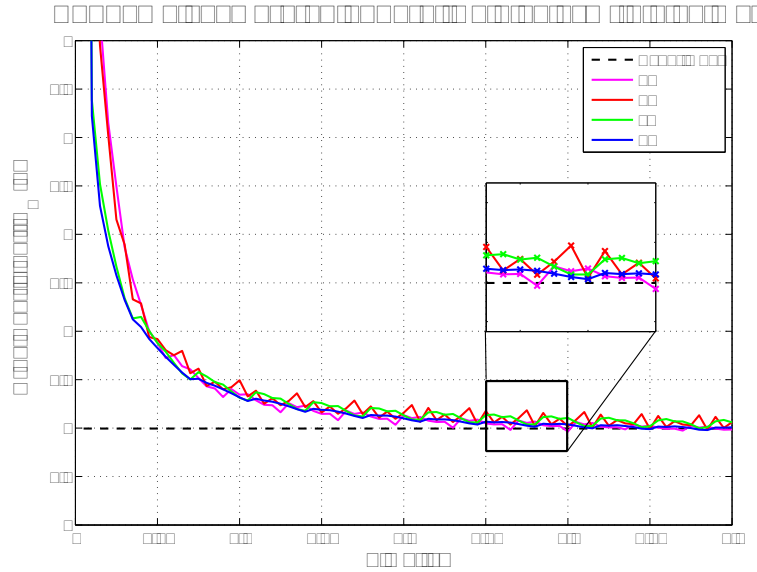


Figure 5.31: Drag coefficient experienced by an instantaneously accelerated sphere as a function of time ($t = 5$ ms). a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) excluding the *ghost cells* from the continuity (eq. 3.53), d) continuity solved with *cut-cell* approach (eq. 3.54).

linear in log-log scale. Contrary to the previously studied case, here the basic IB implementation (a, setting the *ghost cell* velocities only), allows to obtain a relatively low level of fluctuations compared to lines b and c.

Moreover, the root-mean-square values of the C_D^2 seem to be independent of the time-step size, if the zero pressure gradient boundary condition is specified at the surface of the IB (b). For large t , they are however bigger than the values obtained when the *ghost cell* velocities are excluded from the continuity equation (line c on the graph).

The instantaneous C_D^2 discontinuity for the investigated implementations is plotted for different time-step sizes in Figs. 5.33-5.36. A certain degree of periodicity can be observed in all cases. Also, a distinct reduction of the oscillations with increasing t is observed only when the *ghost cells* are excluded from the continuity (c) or the *cut-cell* approach is applied (d). No clear pattern is observed for other implementations.

The results of the analysis, presented in this section introduce a new question with regards to the IB method. Although the Reynolds number of the simulation of the instantaneously accelerated sphere is similar to the Re of the oscillating sphere simulation, investigated in the previous section, the flow behaviour is significantly different. This may be linked to the fact, that the velocity of the particle in the current test case is higher than the maximum velocity of the oscillating sphere. The velocity magnitude seems to affect the balance between various components of the momentum and continuity equations and therefore results in a modified dependence of the fluctuations on the time-step size.

Nevertheless the *cut-cell* method applied to the continuity equation allows to achieve a lower oscil-

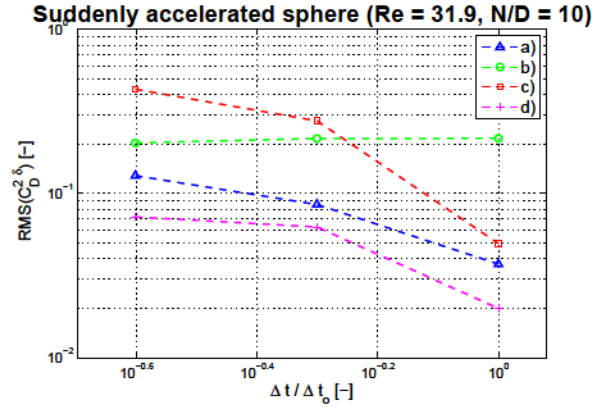


Figure 5.32: RMS of the $C_D^{2\delta}$ for different IB implementations as a function of relative time-step ($\Delta t / \Delta t_o$). a) IB setting the ghost cell velocity only, b) zero pressure gradient at the surface (eq. 3.51), c) excluding the *ghost cells* from the continuity (eq. 3.53), d) continuity solved with *cut-cell* approach (eq. 3.54).

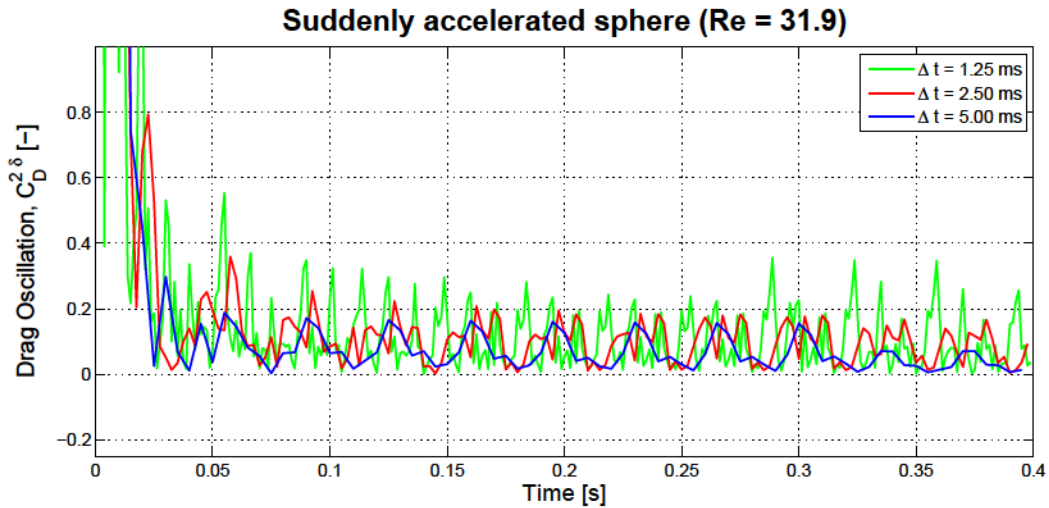


Figure 5.33: Instantaneous $C_D^{2\delta}$ discontinuity as a function of time for the simulations where only the *ghost cell* velocities are set.

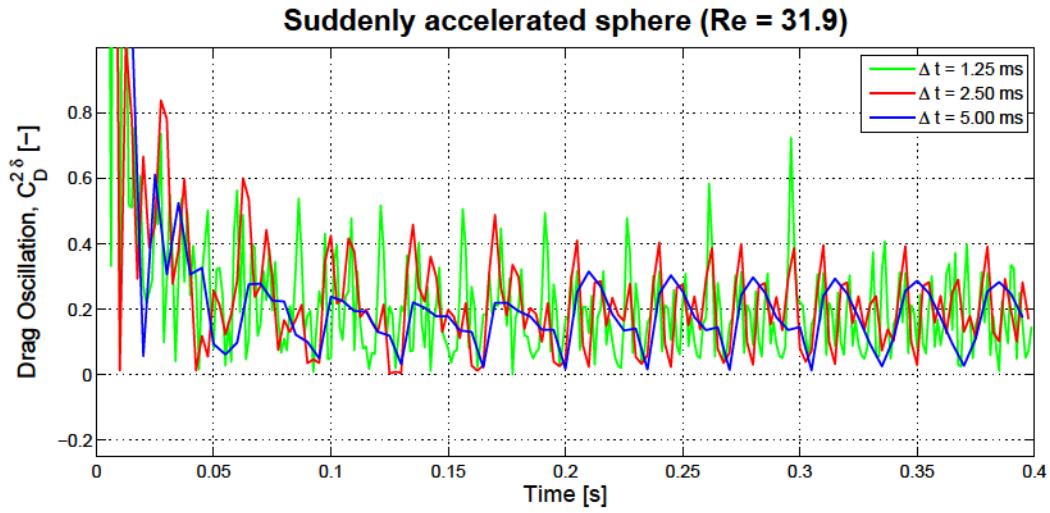


Figure 5.34: Instantaneous $C_D^{2\delta}$ discontinuity as a function of time for the simulations resolved with zero pressure gradient at the IB surface.

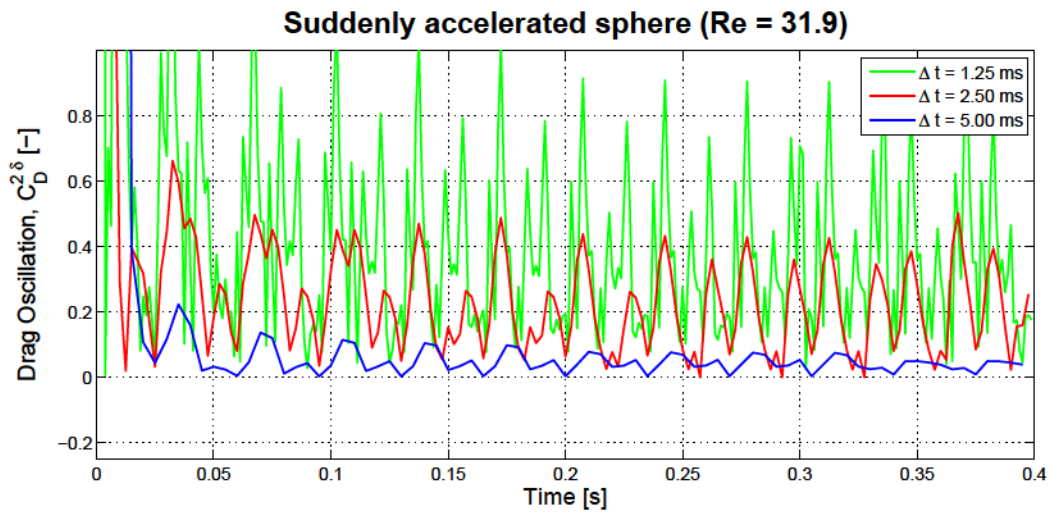


Figure 5.35: Instantaneous $C_D^{2\delta}$ discontinuity as a function of time for the simulations where the *ghost cell* velocities are excluded from the continuity equation.

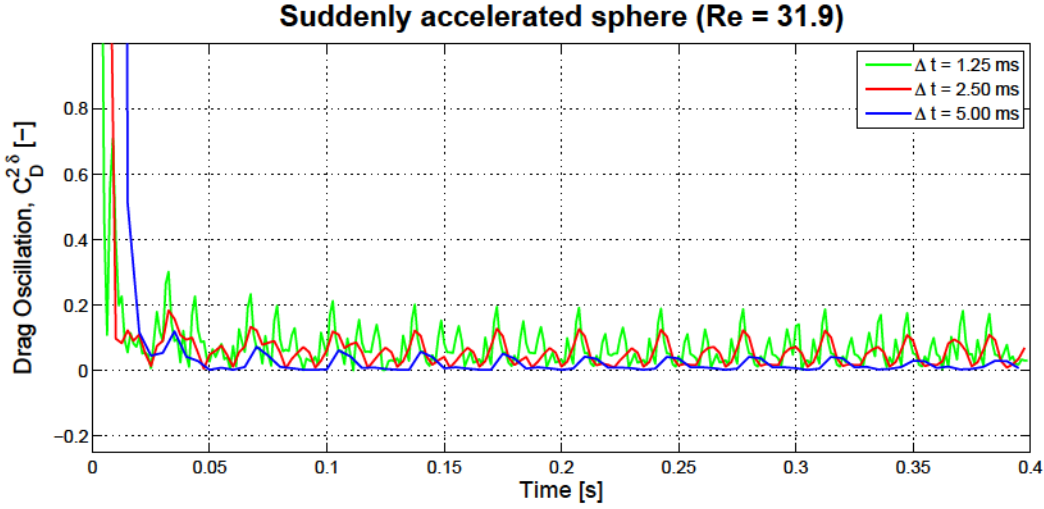


Figure 5.36: Instantaneous $C_D^{2\delta}$ discontinuity as a function of time for the simulation resolved with the *cut-cell* technique.

lation level than the other cases. It is also able to predict the drag coefficient values agreeing with the experimental prediction.

5.5 Sedimenting sphere

The final test performed during the validation process is the simulation of a sphere settling in a fluid tank under the influence of the gravity. The goal of this test is to show the ability of the developed technique to accurately simulate an arbitrary flow with a moving particle, where a two-way fluid-particle coupling is present. The simulation set-up is based on the well documented experiment proposed by ten Cate *et al.* [108]. The simulations are performed for every test case presented in [108], with the Reynolds numbers, based on the particle terminal velocity, ranging from 1.5 to 31.9.

For all tests presented in previous sections, the motion of the particle was predefined in the computational set-up. This is not the case for the simulation of a sedimenting sphere, where a two-way coupling is required. This means that the fluid equations are solved with appropriate boundary conditions applied at the IB surface, while the motion of the particle is determined by the net hydrodynamic forces computed from the resolved flow field as described in chapter 3.2.

Since the velocity of the body is determined by time integration of the acceleration, the influence of the spurious pressure oscillation on the velocity profile is small, as will be shown in the results section. Still, if the magnitude of the pressure oscillations is large enough, it can affect the velocity values. Strong pressure oscillations will result in an oscillatory behaviour of the velocity profile.

5.5.1 Simulation set-up

A sphere of diameter $D = 15$ mm is suspended 135 mm from the bottom of a fluid tank as shown in Fig. 5.37. Note that the gravity acts in the positive x direction. The fluid domain is discretised by $107 \times 67 \times 67$ grid cells resulting in $N/D = 10$. The particle density is specified to be $\rho_p = 1120$ kg/m³, while the gravitational acceleration is $g = 9.80665$ m/s². The fluid parameters (density, viscosity) are adjusted to obtain the desired flow conditions. The specific values are listed in the Table 5.6.

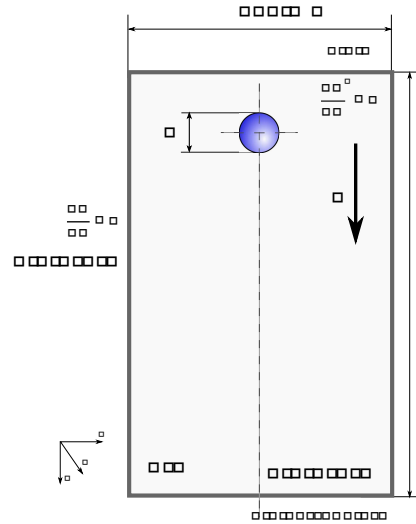


Figure 5.37: Computational set-up for simulations of a settling sphere, showing the initial position of the particle in the domain.

Table 5.6: Fluid properties used in simulations of a sedimenting sphere.

Experiment	[kg/m ³]	[Ns/m ²]	u [m/s]	Re []
E1	970	0.373	0.038	1.5
E2	965	0.212	0.060	4.1
E3	962	0.113	0.091	11.6
E4	960	0.058	0.128	31.9

Both fluid and the particle are initially at rest. The wall velocity boundary conditions are applied at all the domain boundaries, except the top face where zero normal velocity gradient is specified. The zero pressure gradient is applied on the side walls of the domain, zero pressure is specified at the top wall, while the pressure extrapolation is defined at the bottom face.

The motion of the sphere is driven by the gravity and the fluid forces. A constant force equal to the difference between the particle weight and the buoyancy acts on the sphere throughout the simulation. The gravity has no direct effect on the fluid, *i.e.* no additional source terms are added to the fluid equations, hence the pressure gradient due to the gravity is not observed. The fluid-particle interaction occurs through the momentum transfer between the phases. The boundary conditions imposed on the

IB surface introduces fluid motion, while the resulting hydrodynamic force affects the velocity of the particle.

The sphere is allowed to settle until it reaches the vicinity of the bottom wall. When the imaginary points used for setting the boundary conditions on the particle fall outside the flow domain the simulation is stopped. The simulation time-step $\Delta t = 0.0025$ s is the same for all investigated cases.

The force on the particle is evaluated in a similar way as in previous simulations, *i.e.* using a 3-point pressure, and second-order normal velocity gradient extrapolations. Central discretisation scheme is applied for the calculation of the convective terms in the momentum equation.

5.5.2 Results and discussion

The velocity profiles obtained by the simulations of four experimental sedimenting sphere cases and the corresponding experimental measurements are shown in Fig. 5.38. The flow field is resolved using the *cut-cell* approach for the continuity equation with recalculation of the mass fluxes sub-step.

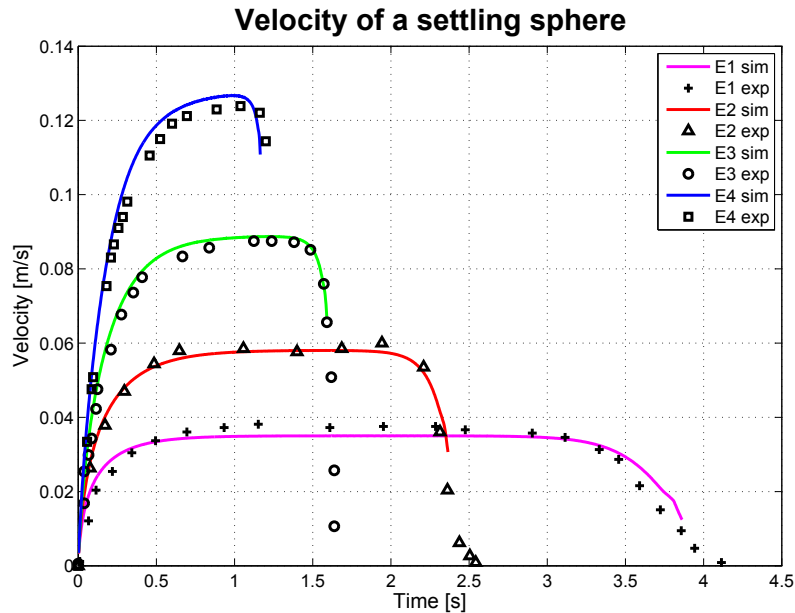


Figure 5.38: Comparison of particle settling velocities, plotted as a function of time, measured during the experiments (markers) and calculated during the simulation (lines). Flow field resolved using the *cut-cell* approach for the continuity equation.

In general, the velocity prediction is in good agreement with the experimental measurements. The velocity for the experiment 4 is slightly over-predicted, what may be related to the grid refinement level ($N/D = 10$).

The particle trajectories predicted by the simulations are also similar to the trajectories observed in [108], as seen in Fig. 5.39. The left axis of the figure shows the relative height of the sphere, normalised against the particle diameter. A small discrepancy is seen, when the sphere reaches the bottom wall, as currently the IB implementation does not support any sub-grid force modelling.

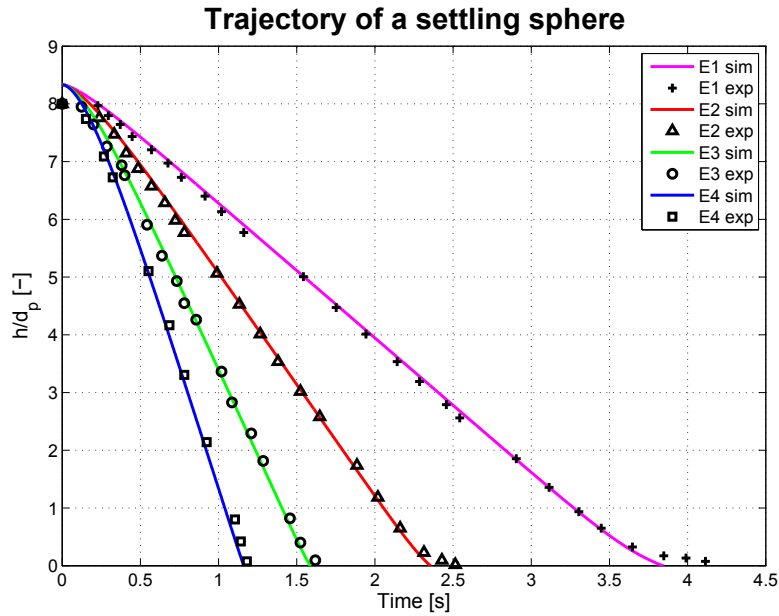


Figure 5.39: Comparison of particle settling trajectories, plotted as a function of time, measured during the experiments (markers) and calculated during the simulation (lines). Flow resolved using the *cut-cell* approach for the continuity equation.

The current results can be also compared to the predictions achieved by simulations where a zero pressure gradient boundary condition is imposed on the IB interface (Fig. 5.40). Both predictions are very similar. Still, some velocity fluctuations, which are not present in Fig. 5.38, can be observed in Fig. 5.40 for high Reynolds number flows.

Since the velocity of a settling particle depends on the integrated acceleration, which in turn is calculated from the balance of forces acting on the body, a force behaviour in time is worth investigating, as it is more sensitive to pressure field oscillations. Figs. 5.41 and 5.42 show a comparison of the force prediction achieved by different IB implementations, for the lowest and highest terminal velocity experiments.

In both cases the magnitude of the calculated drag forces are similar. Nevertheless the zero pressure gradient on the IB interface method, experiences much higher level of fluctuations than the *cut-cell* technique. Since the force oscillations for the low Re simulations are small, they do not have any visible effect on the velocity profiles seen in Fig. 5.40. On the other hand the magnitude of oscillations is considerably higher for the experiment 4 set-up ($Re = 31.9$). Those oscillations are therefore propagated to the velocity profiles (see line E4 in Fig. 5.40).

The effects of the grid refinement level on the velocity predicted by the *cut-cell* technique are also studied and illustrated in Fig. 5.43. Small velocity fluctuations can be observed for the coarse grid, however they almost disappear when the number of grid cells is increased. The velocity predicted at $N_D = 8$ is comparable to the velocity computed when $N_D = 10$, although further grid refinement is expected to result in even more accurate prediction.

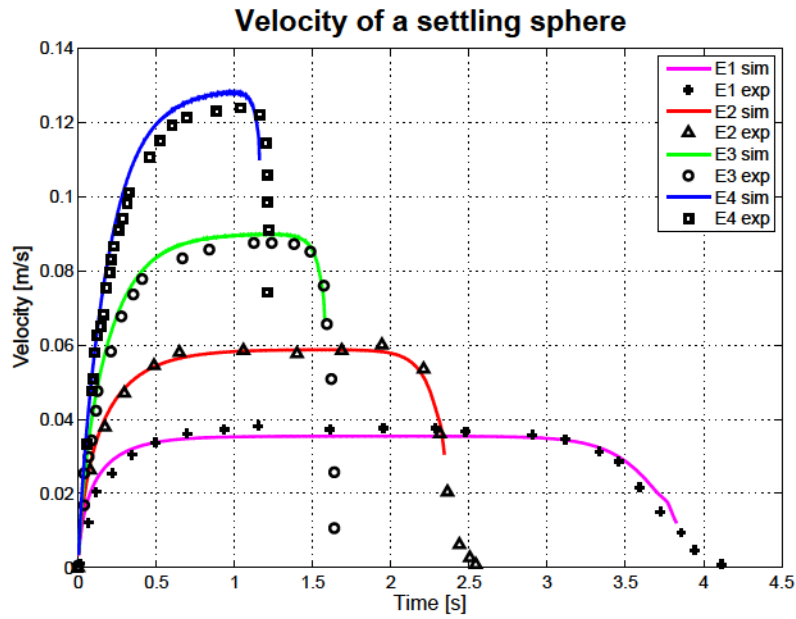


Figure 5.40: Comparison of particle settling velocities, plotted as a function of time, measured during the experiments (markers) and calculated during the simulation (lines). Flow resolved by setting the zero pressure gradient on the IB interface. Note small oscillations in the high Re case (E4).

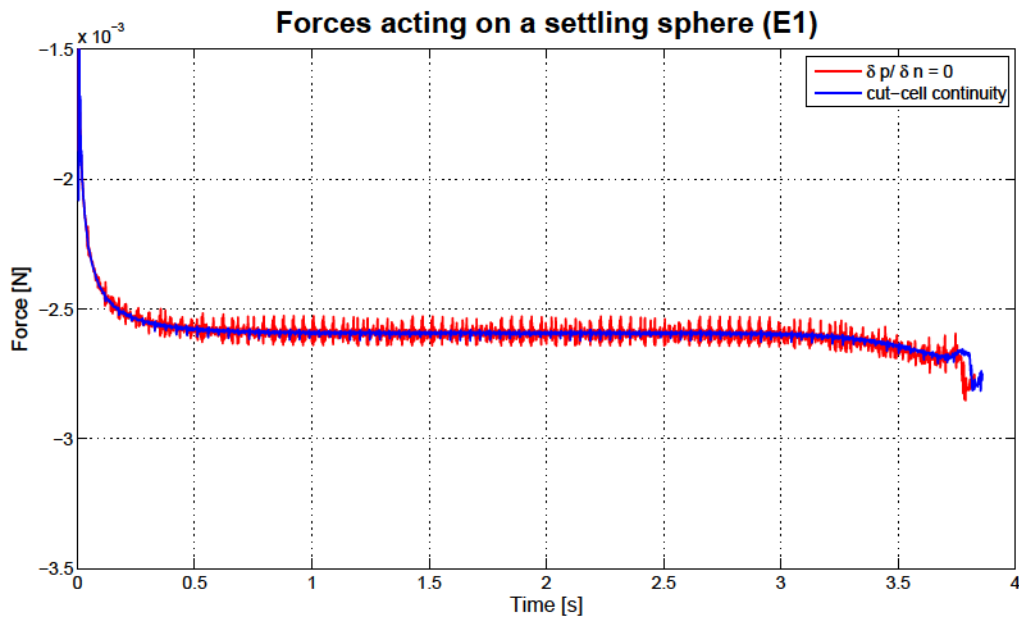


Figure 5.41: Comparison of the forces, plotted as a function of time, acting on a settling sphere ($Re = 1.5$), resolved by different IB implementations.

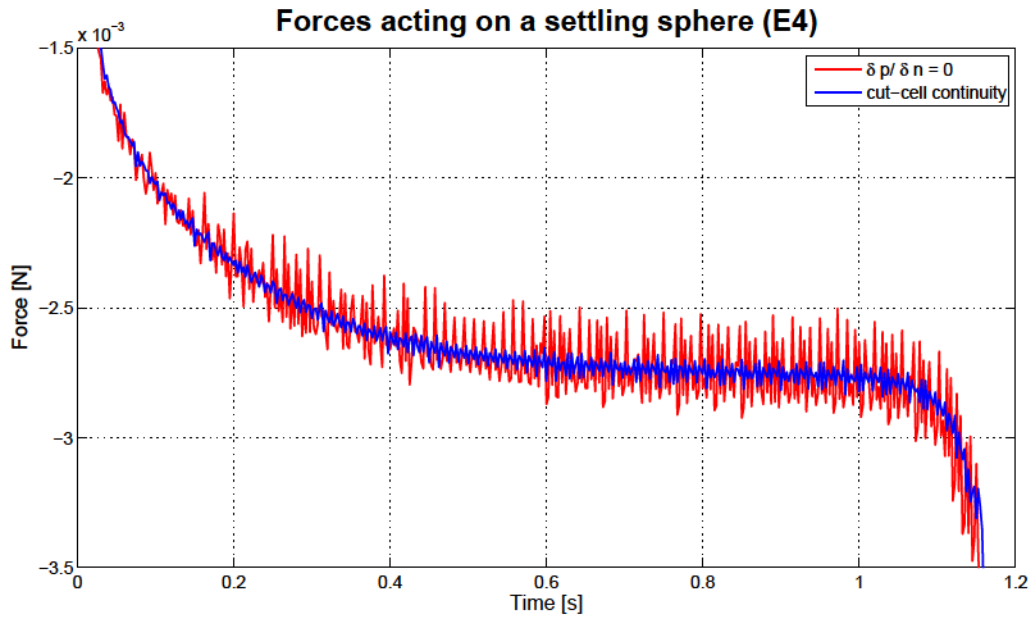


Figure 5.42: Comparison of the forces, plotted as a function of time, acting on a settling sphere ($Re = 31.9$), resolved by different IB implementations.

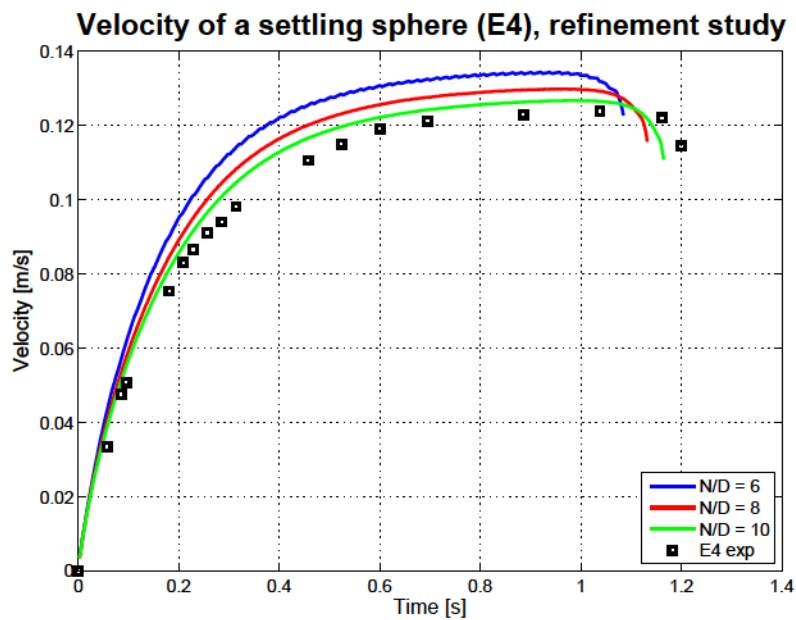


Figure 5.43: Influence of the grid refinement level on the velocity of the settling sphere for the experiment 4 ($u_\infty = 0.128$ m/s). Flow resolved using the *cut-cell* approach for the continuity equation.

5.6 Summary

This chapter focused on analysis of various properties of the Immersed Boundary Method developed during the current project. Results of a number of test cases, designed to independently assess the performance of different features of the IBM, were presented and discussed.

First, an evaluation of proposed force calculation techniques was performed, by investigating the ability of the force frame-works to predict the accurate forces and torques from the analytical flow field around a rotating sphere. Computing the force by means of an imaginary surface around the IB results in the most accurate force and torque prediction even for low grid refinement levels. However the applicability of the imaginary surface technique is limited only to flows with stationary objects.

A combination of a 3-point pressure extrapolation with a second-order normal velocity gradient extrapolation allows to obtain a satisfactory agreement with the expected force coefficients. Also, a full velocity gradient tensor has to be known at the surface of the particle in order to capture the correct torque, calculated from analytical solution, therefore a least-squares velocity gradient tensor estimation is preferred in cases when the torque is of importance.

Simulations of flow past a stationary sphere at $Re = 1.5$ and $Re = 31.9$ were performed in order to evaluate the effect of modifying the continuity equation near the IB on the flow behaviour. Convergence study of the investigated techniques indicates that appropriate treatment of the continuity equation is necessary to achieve well-behaved convergence of the forces.

Choice of the IB method implementation has little effect on the exact value of the force experienced by the particle. Still, the inspection of the pressure profiles in the vicinity of the particle shows a strong dependence on the type of modifications applied to the continuity equation. Smooth pressure profiles near the particle are observed only for the basic IB implementation, *i.e.* IB without any modifications to the continuity equation and when the *cut-cell* approach to the continuity equation was applied. On the other hand, setting zero-pressure gradient, by means of specifying $p_G = p_{IP}$, has a local effect but does not enforce appropriate pressure behaviour in the far-field.

It was also observed that the position of the imaginary point IP, used for setting the *ghost cell* velocities, has a very small influence on the flow field and the force prediction.

Analysis of a sphere oscillating in a fluid was performed in order to investigate the spurious pressure fluctuations associated with the simulations of moving bodies. Tests at different grid refinement levels and time-step sizes were performed. The simulations reveal that the pressure oscillations are caused by cells changing their type as the body moves (occurrence of the so called *fresh* and *dead* cells). Moreover, the fluctuations rate grows with decreasing time-step and decreases with increasing grid refinement level.

Appropriate treatment of continuity equation results in a significant reduction of the magnitude of the oscillations. Application of the *cut-cell* approach to the continuity equation results in an order of magnitude reduction of the oscillations compared to the basic IB implementation. Some oscillations are however still observed.

Combined analysis of the oscillations along with the accuracy of the force prediction was achieved, by means of simulation of an instantaneously accelerated sphere. Similarly as in the case of the flow past

a stationary sphere, all investigated implementations are able to predict a drag force agreeing with the experimental observation.

The *cut-cell* approach results in the smallest oscillation rate. Still, pressure fluctuations dependence on the time-step size is different than in the case of the oscillating sphere. Significantly smaller oscillations are observed in the case of the basic IB implementation, while in the approach where a zero pressure gradient is specified at the surface the magnitude of the oscillations is nearly independent of Δt . This behaviour is attributed to a larger magnitude of the velocity in the analysed case.

Finally, simulations of a sphere sedimenting in a tank were performed in order to analyse the overall performance of the developed method. Simulations of a settling body involve two-way coupling between the body and the fluid; *i.e.* the motion of the particle is influenced by the hydrodynamic force, while the fluid flow is driven by the moving particle. Four tests at terminal velocity Reynolds numbers, ranging from 1.5 to 31.9, were analysed.

Analysis of the results show the ability of the *cut-cell* method to accurately predict the behaviour of the particle both in terms of the velocity and its trajectory. Although the forces acting on the particle still retain the oscillatory character, the magnitude of oscillations is considerably smaller than in a case where a zero-pressure gradient is enforced on the particle surface. It is also shown that the small force fluctuations have no effect on the smoothness of the velocity profile.

Concluding, the *cut-cell* method improves the quality of the results provided by the IB method by significantly reducing the oscillations and being able to accurately predict various flow features.

6 Application of TDNS with Immersed Boundary Method - modelling flows with non-spherical particles

Even though rapid increase in the computational power has been observed in recent years, it is still not possible to perform Direct Numerical Simulations of multiphase flows on an industrial scale. Moreover, in the large scale simulations of turbulent gas-solid flows, the particles are usually modelled as spheres, whereas in reality they usually have different shapes. Because the simplified modelling approach may result in a loss of valuable data, more detailed physical models need to be developed.

This chapter presents work done on improving the modelling techniques for flows with non-spherical particles. Shape specific correlations for the forces and torques acting on non-spherical particles, for range of flow conditions and angles of incidence, are derived in the first section of this chapter. Next, the approach for modelling the motion of non-spherical particles, based on the derived correlations, is described.

6.1 Derivation of drag and lift force and torque coefficients for non-spherical particles in flows

Work presented in this section has been published in the International Journal of Multiphase Flow [122]

6.1.1 Introduction

The ability to predict the behaviour of turbulent gas-solid flows is vital for the successful design and determination of optimum operating conditions in numerous industrial applications, e.g. cyclone separators, fluidised beds, dust collectors, and pulverised-coal combustors to name a few. The dynamics of these type of systems can be investigated through experiments or through numerical simulations. The low cost and large amount of data and insight that can be obtained make the numerical approach a very convenient option. Still, performing large scale numerical study of complex multiphase flow requires some assumptions and empirical data describing the interactions between the fluid and the particles.

So far, nearly all studies performed on the gas-particle flows model particles as perfect spheres. This assumption is very convenient due to its simplicity, the fact that the behaviour of spheres is well known, and the availability of a number of models to describe the interaction with fluid flow. Dependence of the drag coefficient of a sphere on the Reynolds number can be found for example in [102]. Also behaviour

of rotating spheres and spheres moving in a shear flow have been studied with additional correlations for Magnus and Saffman forces available in [19, 65, 89].

Nevertheless, a vast number of the applications deals with non-spherical particles, which makes analysis of these type of flows more complicated. Spheres can be described by a single characteristic value, *i.e.* the diameter, whereas non-spherical particles require more parameters. Even very regular shapes, like ellipsoids or fibres, are described by at least two parameters. Moreover, the particles can have varying orientation with respect to the flow, what additionally complicates the description of their behaviour. Besides the drag force, a non-spherical particle also experiences a transverse lift force along with pitching and rotational torques.

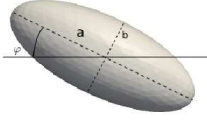
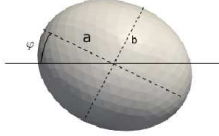
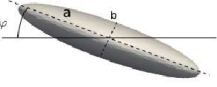
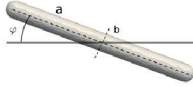
Even though most of the papers on gas-solid flows focus on spherical particles, the effects of non-sphericity had been addressed by some researchers. In order to account for the deviation from the idealised spherical shape, a so-called “sphericity factor”, ψ , has been introduced [64, 115]. Sphericity is defined as the ratio of the surface area of a sphere with equal volume as the non-spherical particle over the surface area of the non-spherical particle. By definition, the sphericity factor is less than or equal to one, where one corresponds with a sphere. In most engineering handbooks (*e.g.* [16]) the drag of a non-spherical particle is estimated using correlations for spherical particles and modified to take into account the sphericity factor.

Using sphericity to describe non-spherical particles may give promising results, nonetheless it is far from ideal solution. For instance, the same value of sphericity can be obtained for a needle like prolate ellipsoid and for a disc, while their behaviour in the flow will be different. Moreover, the sphericity does not account for the orientation of the non-spherical particle. In order to introduce orientation dependency in drag correlations, some researchers, like Hölzer and Sommerfeld [47], use two additional factors: the sphericity determined in the lengthwise direction and one in the crosswise direction, making the effective sphericity orientation dependent.

Other ways to describe the shape of the particle are proposed by Rosendahl [100] or by Loth [75]. Rosendahl uses the super-elliptic function to describe the particle shape and to predict the drag at two extreme orientations, *i.e.* aligned with the flow, and at 90° relative to incoming fluid velocity. On the other hand Loth describes the particle by its aspect ratio, which is applied to calculate shape correction factors both in parallel and cross-wise directions. The effects of orientation can be also included by modifying the reference area in the force coefficient expressions. The most complete overview of the existing methods for analysing non-spherical particles can be found in a paper by Mandl and Rosendahl [78]. So far, the majority of the research has focused on determining the drag coefficient, while the secondary motion resulting from the lift and torques has received very little attention.

The current thesis shows the results of True Direct Numerical Simulations of the flows past four different non-spherical particles presented in Table 6.1. The obtained forces are then used to design shape-specific correlations, that describe the interactions between the fluid and the particles. The equations can be used as a base of large scale analysis of complex flows with non-spherical particles.

Table 6.1: Four non-spherical particle shapes considered in this thesis.

Shape	Parameters	Shape	Parameters
Ellipsoid 1 	$\frac{a}{b} = 0.89$ $\frac{a}{b} = \frac{5}{2}$	Ellipsoid 2 	$\frac{a}{b} = 0.99$ $\frac{a}{b} = \frac{5}{4}$
Disc 	$\frac{a}{b} = 0.63$ $\frac{a}{b} = \frac{5}{1}$	Fibre 	$\frac{a}{b} = 0.69$ $\frac{a}{b} = \frac{5}{1}$

6.1.2 Forces on particles

Few closed models describing the motion of a non-spherical particle in the fluid are available. In a Lagrangian framework the translation motion of particles can be described by Newtonian equations of motion [121]:

$$m_p \frac{dv_p}{dt} = F_D + V_p(\rho_p - \rho_f)g + F_{PG} + F_{VM} + F_L \quad (6.1)$$

where m_p is the particle mass, V_p is the particle volume, ρ_p is the particle density, ρ_f is the fluid density, and v_p is the translational velocity of the particle centre of mass. The forces acting on the particle are given on the right hand side of the equation and correspond to drag, buoyancy, the force due to the fluid pressure gradient, the virtual mass force and the lift force. For heavy particles in dilute suspensions the drag, lift and inertia effects play dominant role in determining the motion of the particle [69], therefore these forces are the main focus of the current thesis.

When describing rotational motion of an axis-symmetric particle, it is convenient to use an additional coordinate system, fixed at the particle centre and aligned with the axis-symmetric axis of the particle. In such system the equations of rotational motion take the following form [121]:

$$I_x \frac{d\omega_x}{dt} - \omega_y \omega_z (I_y - I_z) = T_x \quad (6.2)$$

$$I_y \frac{d\omega_y}{dt} - \omega_z \omega_x (I_z - I_x) = T_y \quad (6.3)$$

$$I_z \frac{d\omega_z}{dt} - \omega_x \omega_y (I_x - I_y) = T_z \quad (6.4)$$

where I_x, I_y, I_z are the moments of inertia; $\omega_x, \omega_y, \omega_z$, the angular velocities, while T_x, T_y, T_z are the torques with respect to particle axes. It is clear that for a successful model of particle motion one needs to know the drag and lift forces along with the torques acting on the particle in various flow conditions.

There are few research papers concerning the analytical derivation of the flow past ellipsoids at very low Reynolds numbers. Brenner [7] established an expression for the forces acting on the prolate ellipsoids in a flow, while Jeffery [54] introduced the equations for the torques on the particle. These models have been applied in various analysis of flows with non-spherical particles, *e.g.* [25, 79, 88]. The aforementioned models have, however, limited use in the cases of flows where particle Reynolds numbers are larger than unity. Therefore, an alternative approach is necessary in order to calculate the forces on the particle over wider range of Re . The most suitable way is to perform an experimental study or numerical analysis to obtain the forces and torques on particles over a range of flow conditions.

In order to predict the fluid interaction with a general non-spherical particle, simulations over a range of Reynolds numbers and over the range of the two independent Euler angles, ranging from 0° to 360° and 0° to 180° respectively, are to be performed. This leads to a very large number of required simulations. However, many non-spherical shapes can be treated as axis-symmetric and modelled as prolate or oblate ellipsoids or as cylindrical fibres. For these shapes, by a rotation of coordinate system, any three-dimensional arrangement with the flow field can be converted into simplified 2D flow case as illustrated in Fig. 6.1.

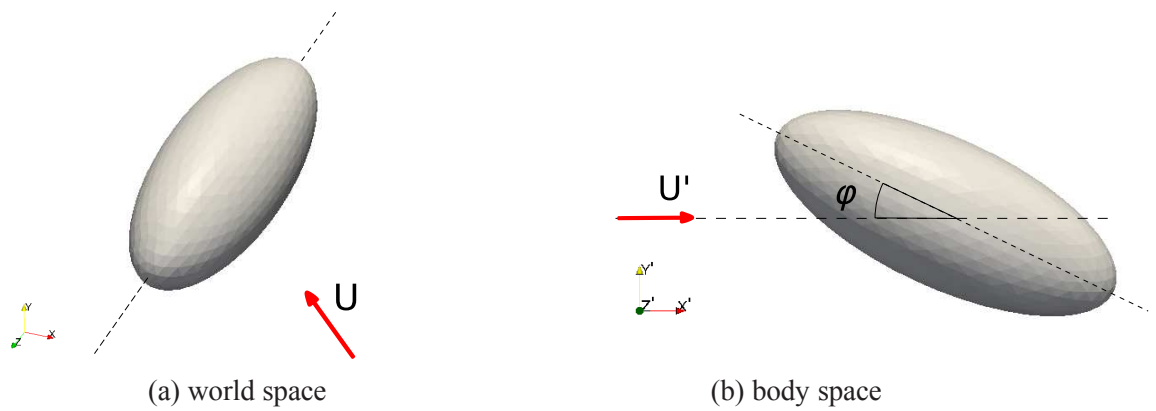


Figure 6.1: The fluid velocity in world space (left) and in body space (right). The angle of incidence is defined as the angle between the direction of the fluid velocity and the longest axis of the body.

This significantly simplifies the problem and allows to minimize the number of parameters needed to perform a full study of the particle behaviour. In a two-dimensional coordinate system the particle experiences a drag force acting in the direction of the flow, a transverse lift force, a pitching torque and torque counteracting the rotation. Details on these forces and the current state of knowledge are described below.

Drag force

The drag force experienced by a non-spherical particle acts in the direction of the flow velocity and is characterised by the drag coefficient, defined as:

$$C_D = \frac{F_D}{\frac{1}{2} \rho u^2 A_P} \quad (6.5)$$

where F_D is the actual drag force, $u = u - u_p$ is the velocity of the particle relative to the local undisturbed fluid velocity u , ρ is the fluid density, and $A_P = \frac{\pi}{4} d_p^2$ the reference area with d_p being the equivalent particle diameter, *i.e.* the diameter of a sphere with the same volume as the considered particle. The choice of the definition of the reference area is arbitrary, as long as it is used consistently. Defining the reference area as the cross-sectional area of volume equivalent sphere guarantees that it stays constant regardless of the angle of incidence. It also allows to easily compare the coefficient for various shapes and orientations.

The drag coefficient for non-spherical particles has been focus of research for some time. Similarly as in the case of spherical particles, the drag coefficient is determined empirically for particle Reynolds numbers ($Re = \frac{u d_p}{\nu}$) larger than unity; this can be done experimentally (*e.g.* [21, 117]) or computationally (*e.g.* [48, 75]). Haider and Levenspiel [40] correlate the experimentally obtained drag force for a number of particle shapes and particle Reynolds numbers using the sphericity coefficient. Hölzer and Sommerfeld [47] propose an improvement to the prediction by introducing crosswise sphericity factor ψ , which is the ratio of the crosswise area of the equivalent sphere to the crosswise area of the particle. Loth [75], on the other hand, uses shape correction factors based on the particle aspect ratio to determine the drag in the Stokes and Newton flow regimes, while the drag in the intermediate Re region is calculated by matching the drag coefficients from the two extreme regimes. A different approach is suggested by Rosendahl [100], who introduces a formula based on several correction factors using the super-elliptic description of the shape of the non-spherical particle. There are also a number of other correlations which have been proposed on similar grounds (*e.g.* [26, 41], and a review can be found in paper by Chhabra *et al.* [11]). Although all of these correlations are complex and contain 5 to 10 fit parameters, the errors in their prediction are still quite large; a typical mean error of 25%, with maximal errors exceeding 100% [11, 47].

It is worth noting that only few of the available correlations predict the change of the drag coefficient with the relative angle of the particle to the flow velocity, sometimes referred to as the angle of incidence. Rosendahl [100] heuristically suggests to correlate the drag coefficient with this angle using the following equation:

$$C_D(\alpha) = C_D(0) + (C_D(90) - C_D(0)) \sin^3 \alpha \quad (6.6)$$

where α is the angle of incidence. In other approaches the angle dependency is obtained by modifying the sphericity or the drag coefficient definition itself, taking into account the changing crosswise area of the particle.

Lift force

Lift force on a particle not aligned with the flow velocity is a result of non-axisymmetric flow field. It acts in the direction perpendicular to the fluid velocity and similarly as the drag can be characterised by lift coefficient, defined as:

$$C_L = \frac{F_L}{\frac{1}{2} \rho u^2 \frac{\pi}{4} d_p^2} \quad (6.7)$$

where F_L is the actual lift force - the fluid force acting orthogonal to the flow velocity.

Compared to the drag force, significantly less research work has been done to predict the lift exerted on a non-spherical particle by the fluid motion. There is some work on deriving equations for the lift coefficients for rotating spherical particles, or spheres in a shear flow (e.g. [10, 101]), but as the transverse forces in case of non-spherical particles are a product of different sources, the aforementioned studies are of a limited usage. A common assumption for the lift force, is that it is proportional to the drag force with the orientation of the non-spherical particle, by the so called ‘‘cross flow principle’’ [46],

$$\frac{C_L}{C_D} = \sin^2 \theta \quad (6.8)$$

There is an attempt to improve this equation by Mandel and Rosendahl [78], introducing a dependency on Reynolds number, however this equation does not enhance the quality of the prediction.

Pitching torque

Since the centre of pressure of the total aerodynamic force acting on the particle does not coincide with the particle’s centre of mass, a pitching torque is generated. It acts around the axis perpendicular to the plane where the forces are present and attempts to increase the particle’s angle of incidence. Analogously as the force coefficients, the torque coefficient can be defined as:

$$C_T = \frac{T_P}{\frac{1}{2} \rho u^2 \frac{\pi}{8} d_p^3} \quad (6.9)$$

where T_P is the pitching torque. Note the d_p^3 term in the denominator, instead of d_p^2 present for force coefficients.

A heuristic expression for the torque for low Reynolds number on a non-spherical particle is proposed in [5]. This expression originates from the limits of the lift force, its invariance under a 180° rotation of the particle and that it should vanish if the angle is 90°. This expression is employed in [100] to predict the behaviour of non-spherical particles in a swirling flow. This expression has, however, not been rigorously validated before. Other approaches include determining the centre of pressure on the particle as a function of angle of incidence and determining the resulting torque as the cross-product of the total force and the distance between the centre of gravity and the centre of pressure [78].

Rotational torque

The particle can gain rotation due to the pitching torque or by collisions with the walls of the domain or other particles. The rotation of non-spherical particle can occur in two modes: around the axis of symmetry and around the axis perpendicular to the axis of symmetry. In both cases the rotational torque can be characterised by the torque coefficient defined as:

$$C_R = \frac{T_R}{\frac{1}{2} \rho u_p^2 \frac{d_p^5}{2}} \quad (6.10)$$

where T_R is the rotational torque, u_p is the relative rotation with ω_p being the particle angular velocity. The rotational torque coefficient is different for every mode of rotation.

There is no easily available correlation describing the rotational torque of a non-spherical particle. The closest useful assumption is the formula for spheres, suggested by Dennis *et al.* [19], where the coefficient depends on rotational Reynolds number, which is defined as:

$$Re_R = \frac{\omega_p d_p^2}{\nu} \quad (6.11)$$

Some study on the secondary motion of the particle has been done. Lattice Boltzmann simulations were performed in [48] for various shapes and graphs of the obtained lift and torque coefficients are shown. However, no empirical correlations are presented.

Summarising, even though there exists a number of correlations allowing for calculation of forces on an arbitrary non-spherical particle, they have limited accuracy for an arbitrary particle. Therefore in this research work the drag, lift and torque coefficients are determined for each particle type. To determine the forces and torque on a non-spherical particle as a function of Reynolds number and particle orientation, a large number of True DNS (TDNS) simulations are performed, where an accurate flow field around the particle is determined. The term “true” emphasizes that not only all the flow scales are resolved but also a no-slip boundary condition is imposed at the surface of particle. The forces which are obtained from the TDNS are coarse-grained into semi-empirical models predicting the drag, lift and pitching torque coefficients as a function of particle Reynolds number and orientation, expressed by the angle of incidence. Additionally, the rotational torque coefficients for different modes of rotation are determined as a function of rotational Reynolds number Re_R .

6.1.3 Numerical framework

Because of the large number of simulations required, a computationally effective and efficient framework is desired to deal with flows including non-spherical particles. Moreover, the framework should be able to handle rotating particles. Throughout the years, various methods coupling the particles with surrounding fluid have been developed. Among the oldest ones is the arbitrary-Lagrangian-Eulerian method [49], where a two-dimensional unstructured grid is created around the bodies and the mesh is

adapted as they move. Although this method works very well when the deformations are small, like in aerodynamic problems, the necessity to create a new mesh every time step if the deformations are large is time consuming and may limit accuracy.

The Immersed Boundary method (IBM) tackles this problem by using Cartesian grid for the fluid, while the presence of particles is accounted for by modifying flow variables where the IB crosses the Eulerian mesh cells. Peskin [95] was among the first to propose the IB method. In this implementation, the coupling between the phases is achieved by calculating point forces representing the influence of the boundaries. The forces are distributed over the mesh using a distribution function. This method is first order accurate in space and time. The reason for this is the fact, that a blurred representation of the boundary is achieved.

An alternative way to implement the IB method, sometimes called immersed interface method, is to apply the boundary conditions directly at the interface of the particle. It is introduced in [87], where a smooth velocity gradient is applied over the IB. Despite being able to achieve some promising results, the method exhibits problems with mass conservation in the boundary cells. Therefore, this work employs an improved type of IB method from [87], the mirroring immersed boundary method.

Implicit immersed boundary method

The work outlined in this paper employs the implicit mirroring immersed boundary (MIB) method to resolve the flow surrounding the analysed particles. The first ideas on this method are introduced in [82], although a number of improvements have been made [123]. Due to the fact that it uses a non-boundary conforming grid, it is capable to resolve detailed flow around an arbitrary non-spherical particle. The surface of the particle is triangulated and an immersed boundary condition is applied at the cells where the particle surface triangles intersect the fluid Eulerian grid. The MIB method mirrors the velocity field through the surface triangles by adding a constraint to the Navier-Stokes equations and creating a fictitious flow inside the particle. This is done by setting a fictitious exterior normal point x_e , as depicted in Fig. 6.2. The mirrored point is defined as the x_{iib} is mirrored along the normal of its closest interface surface triangle. Hence, the smallest distance between the interface and the fictitious point, x_e , is equal:

$$x_e = x_{iib} + 2dn \quad (6.12)$$

where dn is the normal distance from the x_{iib} to the closest interface triangle.

If the exterior normal point coincides with a discrete velocity point, the Dirichlet condition for the velocity of the x_{iib} point is trivial. More generally, the exterior normal point lies between the discrete velocity points and therefore the velocity needs to be implicitly interpolated using the surrounding points. The mirrored velocity is set to the reversed interpolated velocity plus the boundary velocity. The interior velocities are set to the boundary (IB) velocity. In order to conserve mass, the velocity of the mirrored points x_{iib} is excluded from the continuity equation. The method has been shown to be second order accurate in predicting the drag, lift and torque forces on a rotating sphere [123].

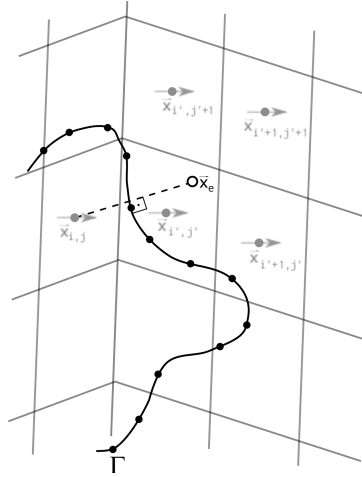


Figure 6.2: A two-dimensional representation of the MIB method, the exterior normal point x_e with its surrounding interpolation points.

Calculation of the forces

The force on the immersed particle is indirectly given by the numerical framework, as the solution of the Navier-Stokes equations implicitly accounts for the presence of the boundary of the non-spherical particle. The total force consists of two distinctive contributions, *i.e.* the pressure and the viscous forces. These forces are integrated over the interface Γ , giving

$$F_i = \int_{\Gamma} (p \delta_{ij} + \tau_{ij}) n_j dS \quad (6.13)$$

Neither the pressure nor the velocity derivatives, required to determine the surface forces, are explicitly known at the interface and therefore need to be determined from the resolved flow field. To determine the pressure at the interface, three auxiliary points, normal to the considered triangle, are applied to extrapolate the value of the pressure onto the surface. Their position with respect to the fluid mesh is shown in Fig. 6.3. The pressure at the surface is found with second order accuracy according to the approximation:

$$p_{x_c} = p_x + \frac{p}{x} \frac{1}{2} (5p_x - 4p_x + p_x) \quad (6.14)$$

The viscous force component, on the other hand, depends on the velocity gradients, which are evaluated at the surface. This can be written by Taylor series expansion as:

$$u_i(x_c) = u_i(x_{nb}) + \frac{u_i}{x_j} (x_{nb} - x_c) + O(x_{nb} - x_c)^2 \quad (6.15)$$

where $u_i(x_c)$ is the velocity at the triangle centre, $u_i(x_{nb})$, are the fluid velocities of neighbouring cells, $(x_{nb} - x_c)$ is the distance from triangle centre to the neighbouring cell and $\frac{u_i}{x_j}$ is the sought velocity gradient at the surface. To obtain the most accurate representation of the gradients, a weighted least square problem is solved, where the velocities are based on 17 neighbouring cells, as shown in Fig.

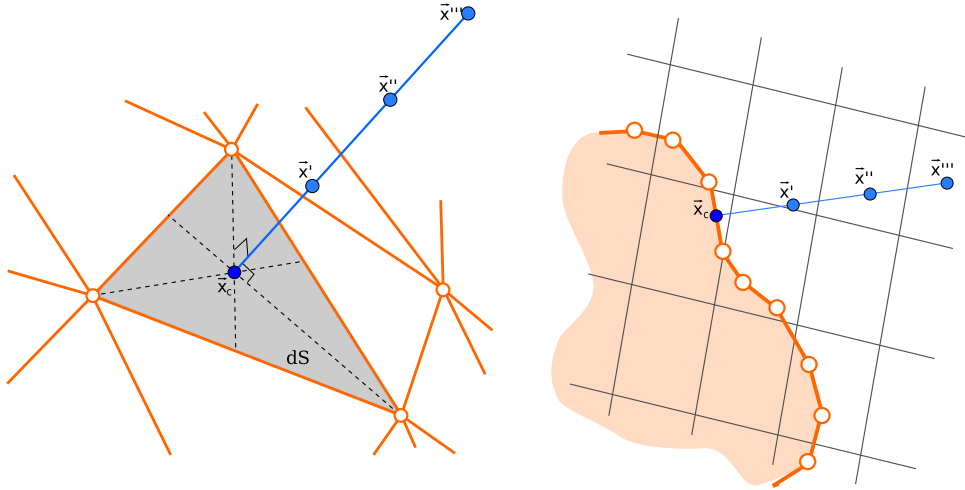


Figure 6.3: An illustration of the points used to extrapolate the pressure onto the particle surface and their position relatively to the fluid cells. Three-dimensional representation is shown on the left, while a two-dimensional projection is illustrated on the right hand side.

6.4. The velocity on the surface, $u_i(x_c)$, is obtained by interpolation, similarly as in the case of the x_e mirroring and pressure points.

6.1.4 Simulation set-up

The true direct numerical simulation (TDNS) framework has been used to determine the drag, lift and torque of the bodies shown in Table 6.1. A triangulated representation of the bodies is introduced in a domain with size $20 d_p \times 20 d_p \times 10 d_p$ for low Reynolds number $Re < 1$ and a cubical domain $10 d_p \times 10 d_p \times 10 d_p$ for high Re number simulations. The set-up is illustrated in Fig. 6.5.

A uniform flow with the velocity $U = 1.0 \text{ m/s}$ is set along the positive x axis. The fluid density is $\rho = 1 \text{ kg/m}^3$. A full slip (*i.e.* no velocity gradient) boundary condition is applied on the boundaries of the domain. The pressure is set to P_{out} at the outlet, and zero gradient in pressure is specified on other boundaries of the domain. The simulations are initially resolved using upwind scheme with very large time step in order to get the first approximation of the flow field. Afterwards, the time step is decreased to maintain the $CFL < 1$ condition and the discretisation scheme is changed to central for increased accuracy.

The number of mesh cells in the domain depends on the Reynolds number and varies between 8 to 12 cells on the diameter of equivalent sphere. The particle Reynolds number, Re , based upon the equivalent particle diameter, d_p , is varied between 0.1 and 300 by adjusting the fluid viscosity μ . Simulations at different particle angles of incidence ranging from 0° to 90° are performed. The particle is fixed and does not rotate. The mirroring immersed boundary is used to enforce the no-slip boundary condition on the surface of the particle and calculate the forces on the non-spherical particle. Obtained force values are applied to establish particle specific empirical relations for force coefficient as a function of Re and

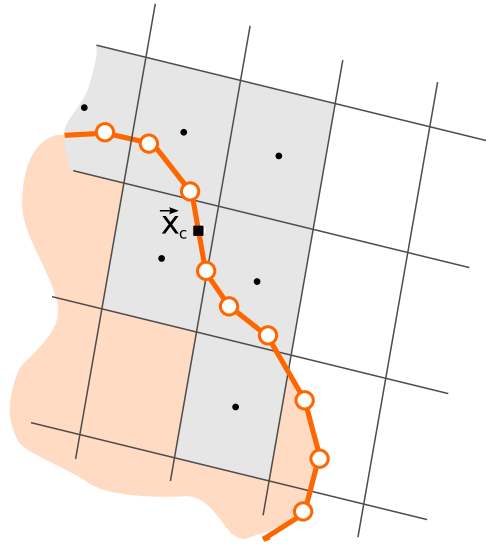


Figure 6.4: A two-dimensional view of neighbouring cells which are used to reconstruct the velocity gradient. Note that the IB points far inside the particle are not included in the velocity gradient determination, as they are forced to match the velocity of the particle and do not represent the fluid.

Additionally, a number of simulations with rotating particle is performed. The particle can rotate around its axis of symmetry (mode 1) or around perpendicular plane (mode 2). In rotational analysis the fluid is stationary, while the rotational Reynolds number Re_R is determined by adjusting angular velocity (ω_1 or ω_2) or the viscosity of the fluid μ .

6.1.5 Results and discussion

TDNS of the flow past each particle type is performed in order to obtain shape-specific drag, lift and torque characteristics as a function of Reynolds number and angle of incidence. The flow is calculated by enforcing a no-slip boundary condition at the particle surface by the mirroring immersed boundary condition presented in section 6.1.3. The length of simulations allows to obtain steady state solution for low Re or constant averages in case of unstable flows. A sample result is illustrated in Fig. 6.6, which shows an instantaneous flow field around the ellipsoid 1 at $Re = 200$.

Obtained values for force coefficients at low Reynolds numbers are compared to the models presented in [7, 25]. This comparison is shown in Fig. 6.7, where a reasonable agreement between the results is seen. The difference in the calculated drag coefficient results from the effects of the finite domain on the current simulations and the non-zero Re number. It is however worth noting that the results deviate quickly from each other as the Reynolds number increases. The comparison with the torque formulas given in [54] is not suitable, as this model applies only in the case of flows with gradients in incoming velocity, what is not the case in the current research.

Fig. 6.8 shows the behaviour of the force coefficients as the time progresses in simulations at low Reynolds number. After the fast readjustment of values of the coefficients, resulting from the change of

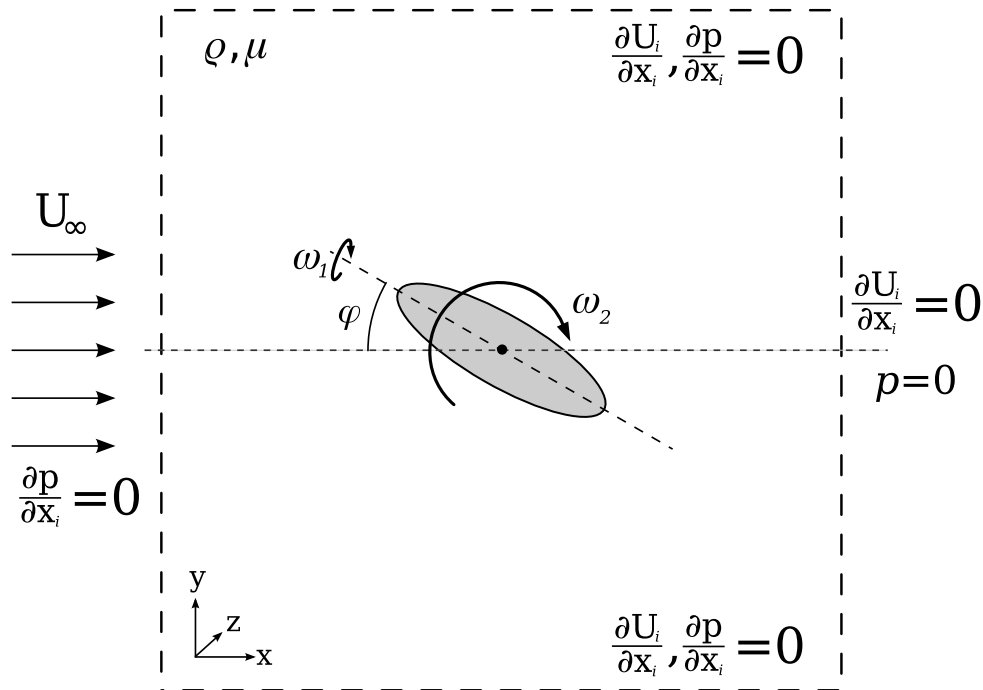


Figure 6.5: A schematic of the fluid domain used for analysis along with the boundary conditions applied at the walls.

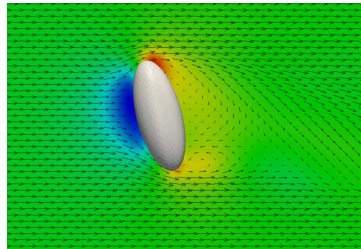


Figure 6.6: A snapshot of the velocity (arrows) and pressure field (colour) in a flow past ellipsoid 1 at $Re = 200$ and $\theta = 70$.

discretisation scheme, constant coefficients are instantly achieved. The difference in coefficient values for various particles can also be clearly seen. For higher Reynolds numbers, *e.g.* $Re = 300$, the flow behaviour sometimes becomes oscillatory, as illustrated in Fig. 6.9. It is worth noting that the oscillations in the flow are more visible when looking at the lift or torque coefficients than the drag. In steady flows, the small fluctuations of the force coefficients tend to be damped, as seen in the case of the fibre in the figure. However, in some cases the perturbations may grow and a transition to an unsteady mode will occur. The unsteady motion can be of nearly periodic or chaotic character. It should be noted that both the shape and the angle of incidence of the non-spherical particle play key role in the resulting flow behaviour, changing the value of the Reynolds number at which the transition occurs. In the current work the forces used for determining the correlations describing the coefficients are obtained by averaging the coefficients throughout the simulation after the initial period of time, allowing for readjustment to a new

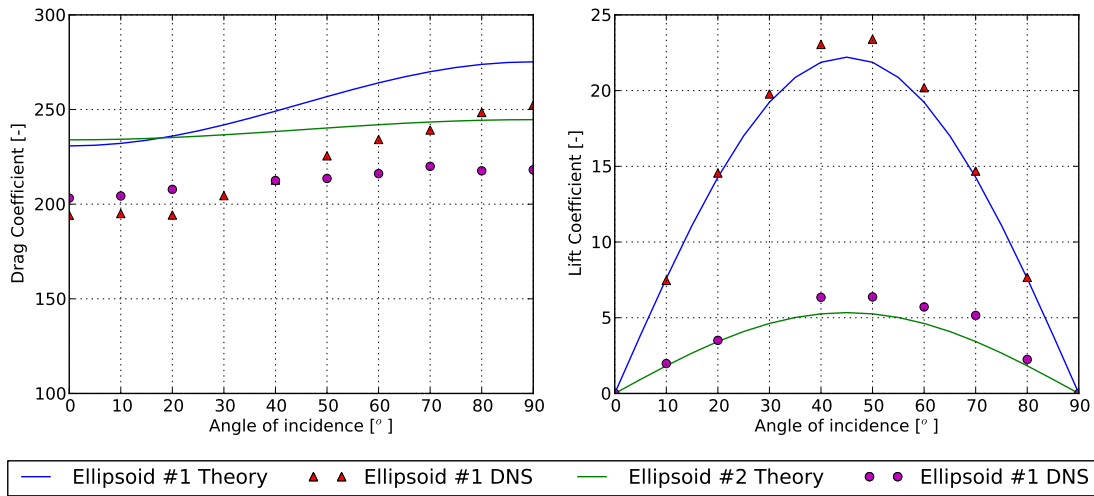


Figure 6.7: The comparison of the values for drag and lift coefficients obtained in [7, 25] with the current TDNS results for prolate ellipsoids at $Re_p = 0.1$.

discretisation scheme.

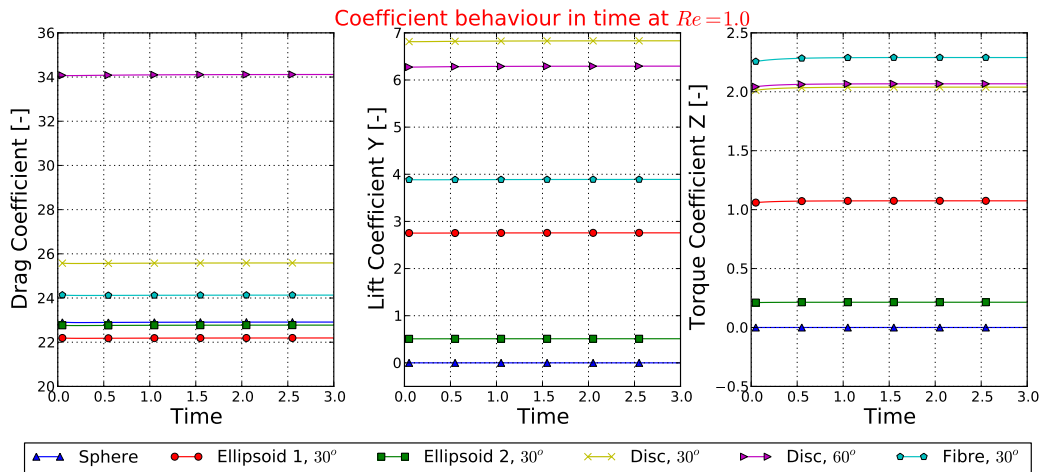


Figure 6.8: The force coefficients and torque coefficients as a function of time for the different particles with flow at $Re = 1$.

Drag

For each particle shape, TDNS simulations are done at various Re numbers, rotational Re numbers and various angles of incidence between the particle and the direction of mean flow. For each simulation, force and torque coefficient values are obtained. The obtained coefficient values are used to create shape-specific force correlations as functions of the Reynolds numbers and the angle of incidence. A new framework for the force coefficients is proposed, which leads to excellent fits for all of the simulated

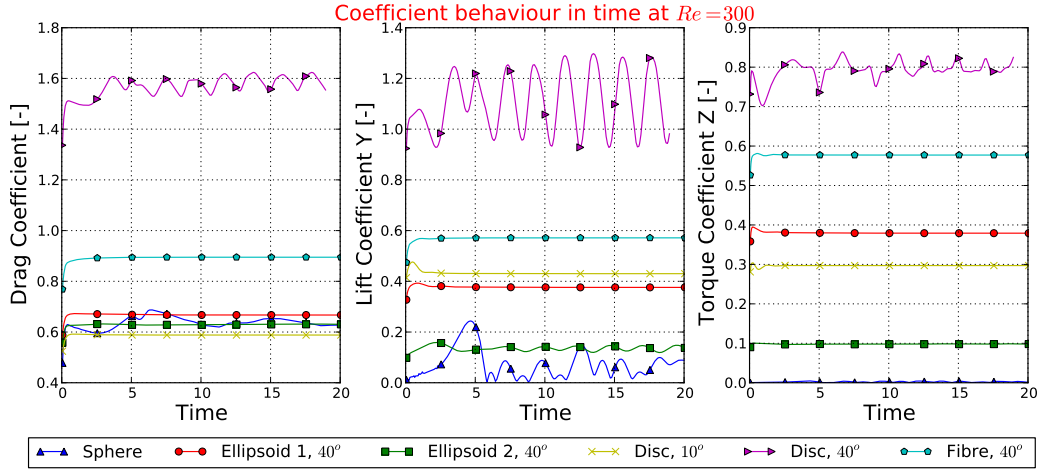


Figure 6.9: The force coefficients and torque coefficients as a function of time for the different particles with flow at $Re = 300$. The sphere is in the unsteady periodic flow regime, giving rise to an additional transverse lift force. Flow past a disc at 10° is steady, however large oscillations are present at 40° . On the other hand, the flow remains steady even at 40° for the fibre and ellipsoid 1, whereas for a nearly spherical ellipsoid 2 small oscillations are present in the lift coefficient.

particle shapes, fluid flows, and rotations. The suggested equation for the drag coefficient is

$$C_D(\alpha) = C_{D=0^\circ} + (C_{D=90^\circ} - C_{D=0^\circ}) \sin^{\alpha_0} \quad (6.16)$$

where

$$C_{D=0^\circ} = \frac{a_1}{Re^{a_2}} + \frac{a_3}{Re^{a_4}} \quad (6.17)$$

$$C_{D=90^\circ} = \frac{a_5}{Re^{a_6}} + \frac{a_7}{Re^{a_8}} \quad (6.18)$$

The resulting particle-specific values of the fit parameters obtained from the simulations are listed in the Table 6.2. The dependency of the drag coefficient on the Reynolds number is shown in Fig. 6.10. The drag coefficient decreases with the Reynolds number for all particle types. A considerable difference between various shapes can be observed. It is also worth noting the change between drag coefficients of the ellipsoid 1 at two extreme orientations. The ratio between the lowest and the highest drag coefficient grows as the Re is increased. Similar observations can be made for other types of particles.

The obtained results are compared with the expressions for drag found in [47, 78]:

$$C_D(\alpha) = C_{D=0^\circ} + (C_{D=90^\circ} - C_{D=0^\circ}) \sin^3(\alpha) \quad (6.19)$$

$$C_{D=0/90} = \frac{8}{Re} + \frac{16}{Re} + \frac{3}{Re} \frac{1}{0.75} + 0.4210^{0.4} (\log)^{0.2} \frac{1}{Re} \quad (6.20)$$

Fig. 6.11 presents the dependence of the drag coefficient values on the angle of incidence for different

Table 6.2: The values of fit parameters for equation for the drag coefficient (Equation 6.18) resulting from the DNS simulations of analysed particles.

Coefficient	Ellipsoid 1	Ellipsoid 2	Disc	Fibre
a_0	2.0	1.95	1.96	2.12
a_1	5.1	18.12	5.82	20.35
a_2	0.48	1.023	0.44	0.98
a_3	15.52	4.26	15.56	2.77
a_4	1.05	0.384	1.068	0.396
a_5	24.68	21.52	35.41	29.14
a_6	0.98	0.99	0.96	0.97
a_7	3.19	2.86	3.63	3.66
a_8	0.21	0.26	0.05	0.16

particles. The coefficient grows with increasing angle following an “s” shaped curve. The angle dependence has much higher influence for large Reynolds numbers, as the $C_{D \max} / C_{D \min}$ ratio increases. The equation proposed by Hölzer and Sommerfeld [47] shows generally good agreement with the current results, however it tends to under-predict the influence of orientation as well as over-predict the drag for particles aligned with the flow at low Reynolds numbers.

Lift

Because of the large amount of results, the lift experienced by the particle is correlated with the Reynolds number and the angle of incidence, θ . Usually, in literature, lift is directly correlated with drag as in equation 6.8. However the current results show that this approach does not seem appropriate as the relation of the lift coefficient to drag is more irregular than that of the lift coefficient alone.

The new equation for the lift coefficient proposed in this thesis is:

$$C_L = \frac{b_1}{Re^{b_2}} + \frac{b_3}{Re^{b_4}} \sin(\theta)^{b_5+b_6 Re^{b_7}} \cos(\theta)^{b_8+b_9 Re^{b_{10}}} \quad (6.21)$$

The first term describes the magnitude of the coefficient while the powers of the trigonometric functions characterise the varying dependence on the angle. As with the newly proposed equation for the drag coefficient, the above equation for the lift gives a very good fit for all of the particle shapes, Reynolds numbers and incident angles. The parameters resulting from fitting the simulations are given in Table 6.3.

The calculated lift is compared with literature lift related to the drag by the “cross-flow” principle given in equation 6.8. The lift coefficient behaviour at various orientations is illustrated in Fig. 6.12. The lift curve is not exactly symmetric with the highest values achieved at angles between 45 and 55 degrees. The big improvement over literature prediction is clearly seen as the current correlations account for the effect of the Reynolds number on the lift magnitude. Presented results are however in good agreement with

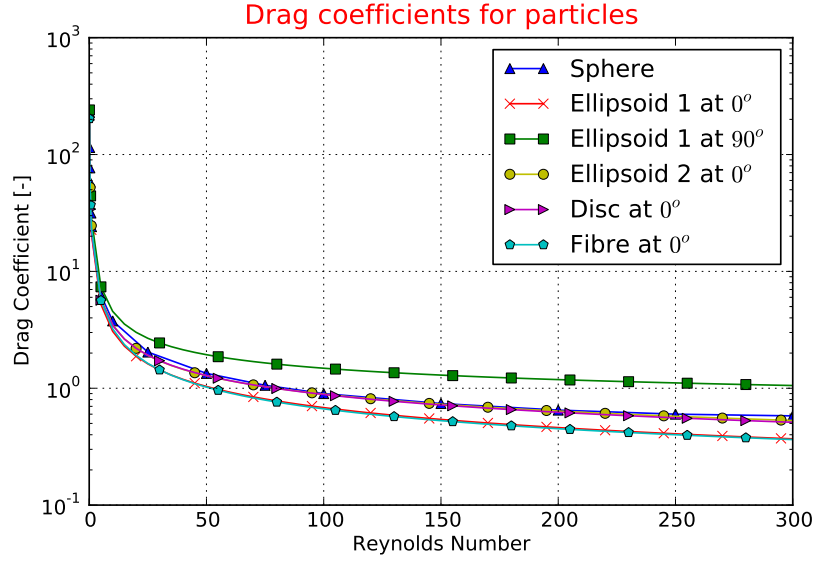


Figure 6.10: Drag coefficients for the analysed shapes as a function of Reynolds number.

those obtained through Lattice Boltzmann method in [48] both in terms of shape as well as magnitude.

Pitching torque

There are two types of torque which may act on a non-spherical particle, pitching torque and rotational torque. The pitching torque occurs when the fluid flow direction is not aligned with one of the symmetry axis of the particle. The behaviour of the coefficient originating from the pitching torque can be described in the same way as in the case of the lift coefficient, as the pitching torque coefficient C_T is zero at 0 and 90 degrees. Therefore following new equation for the pitching torque coefficient is proposed:

$$C_T = \frac{c_1}{Re^{c_2}} + \frac{c_3}{Re^{c_4}} \sin(\theta)^{c_5+c_6 Re^{c_7}} \cos(\theta)^{c_8+c_9 Re^{c_{10}}} \quad (6.22)$$

The parameters resulting from fitting the simulation results are given in Table 6.4. The quality of the fit is very good for all particle shapes and flow-particle incident angles. In the case of the pitching torque coefficient, it is not straightforward to make a direct comparison with literature. In most articles the torque is obtained as the product of the component of the aerodynamic force, which is normal to the non-spherical particle, by the distance from centre of pressure to the centre of gravity x_{cp} :

$$T = x_{cp} F_N \quad (6.23)$$

where the x_{cp} can be given for example by [100]:

$$x_{cp} = \frac{L}{4} (1 - \sin^3(\theta)) \quad (6.24)$$

where L is the length of the non-spherical particle, while F_N is the total aerodynamic force on the non-

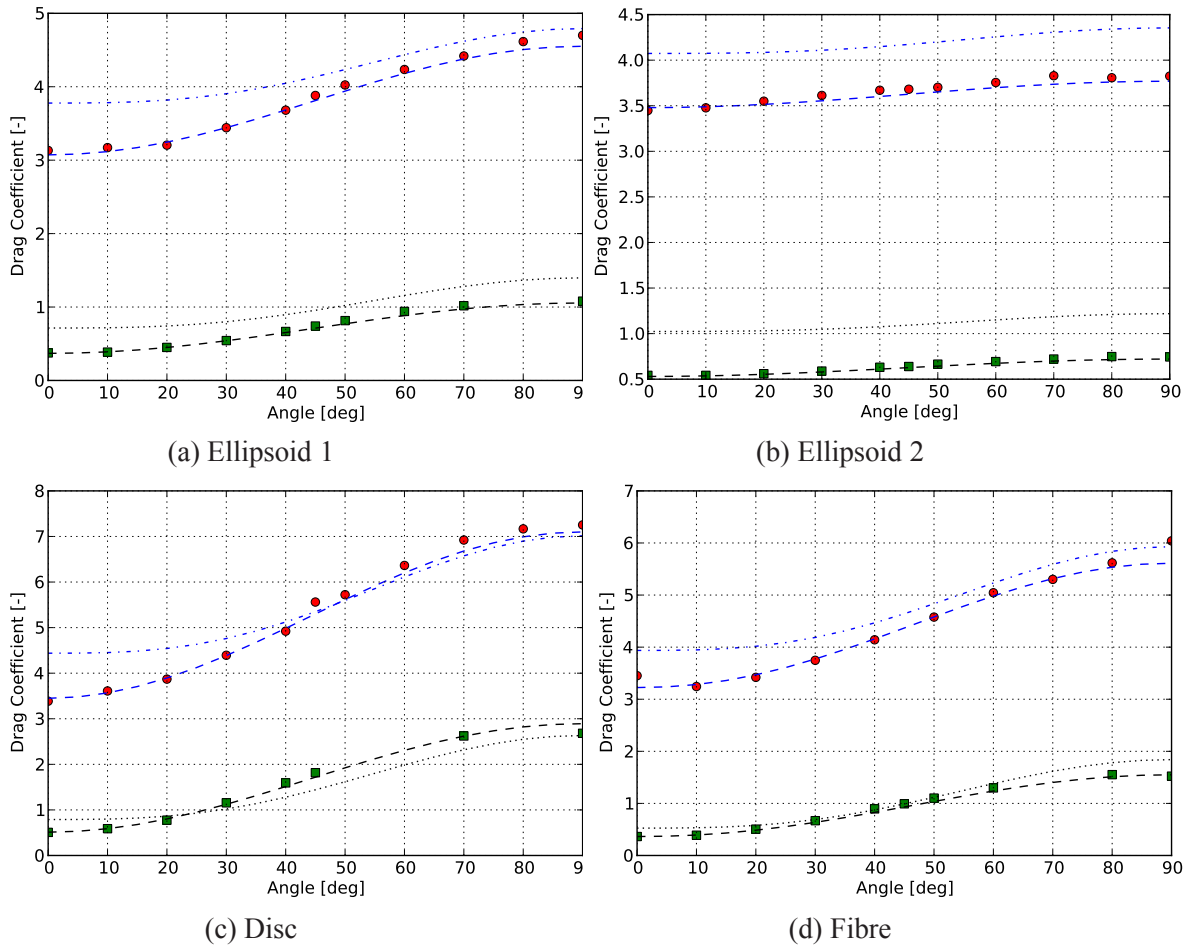


Figure 6.11: Drag coefficients of particles as a function of angle of incidence. (●) simulation at $Re = 10$; (---) present correlation (Eq. 6.18) at $Re = 10$; (---) correlation of [47] at $Re = 10$; (■) simulation at $Re = 300$; (---) present correlation (Eq. 6.18) at $Re = 300$; (---) correlation of [47] at $Re = 300$.

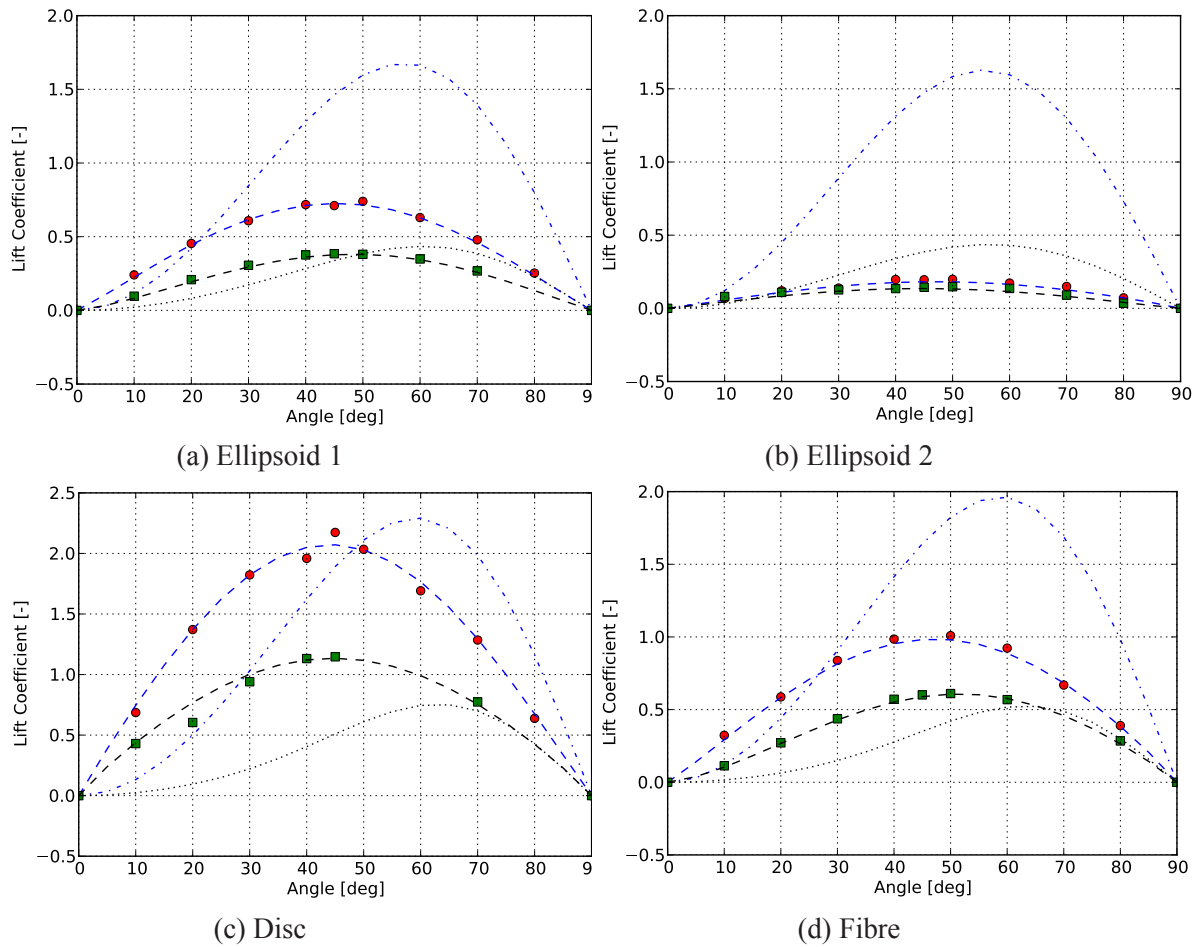


Figure 6.12: Lift coefficients of the particles as a function of angle of incidence. (●) simulation at $Re = 10$; (—) present correlation (Eq. 6.21) at $Re = 10$; (⋯) correlation of [46] at $Re = 10$; (□) simulation at $Re = 300$; (—) present correlation (Eq. 6.21) at $Re = 300$; (⋯) correlation of [46] at $Re = 300$.

Table 6.3: The values of lift parameters for equation for the lift coefficient (Equation 6.21) resulting from the DNS simulations of analysed particles.

Coefficient	Ellipsoid 1	Ellipsoid 2	Disc	Fibre
b_1	6.079	0.083	12.111	8.652
b_2	0.898	-0.21	1.036	0.815
b_3	0.704	1.582	3.887	0.407
b_4	-0.028	0.851	0.109	-0.197
b_5	1.067	1.842	0.812	0.978
b_6	0.0025	-0.802	0.249	0.036
b_7	0.818	-0.006	-0.198	0.451
b_8	1.049	0.874	5.821	1.359
b_9	0.0	0.009	-4.717	-0.43
b_{10}	0.0	0.57	0.007	0.007

spherical particle, with magnitude being the vector sum of the Hölzer and Sommerfeld drag [47] and the lift force predicted by the “cross flow” principle [46]:

$$F_N = \frac{\rho}{8} u^2 d_p^2 C_D \sin(\alpha) (1 + \sin(\alpha) \cos^2(\alpha)) \quad (6.25)$$

Table 6.4: The values of lift parameters for equation for the pitching torque coefficient (Equation 6.22) resulting from the DNS simulations of analysed particles.

Coefficient	Ellipsoid 1	Ellipsoid 2	Disc	Fibre
c_1	2.078	0.935	3.782	0.011
c_2	0.279	0.146	0.237	-0.656
c_3	0.372	-0.469	2.351	8.909
c_4	0.018	0.145	0.236	0.396
c_5	0.98	0.116	-0.394	2.926
c_6	0.0	0.748	1.615	-1.28
c_7	0.0	0.041	-0.044	0.037
c_8	1.0	0.221	-0.537	-15.236
c_9	0.0	0.657	1.805	16.757
c_{10}	0.0	0.044	-0.037	-0.006

The resulting torque can be transformed into the torque coefficient by using equation 6.9. The torque coefficients behaviour at various angles is represented in Fig. 6.13. The large discrepancy is the result of the fact that the errors in the lift prediction from equation 6.8 are propagated to the expression of lift (eq. 6.23), hence it is hard to evaluate the quality of the prediction of the position of the centre of pressure itself. The current results are however comparable to the torque coefficient results presented in [48]. Therefore it can be concluded that application of equations 6.21-6.22 greatly improves the accuracy

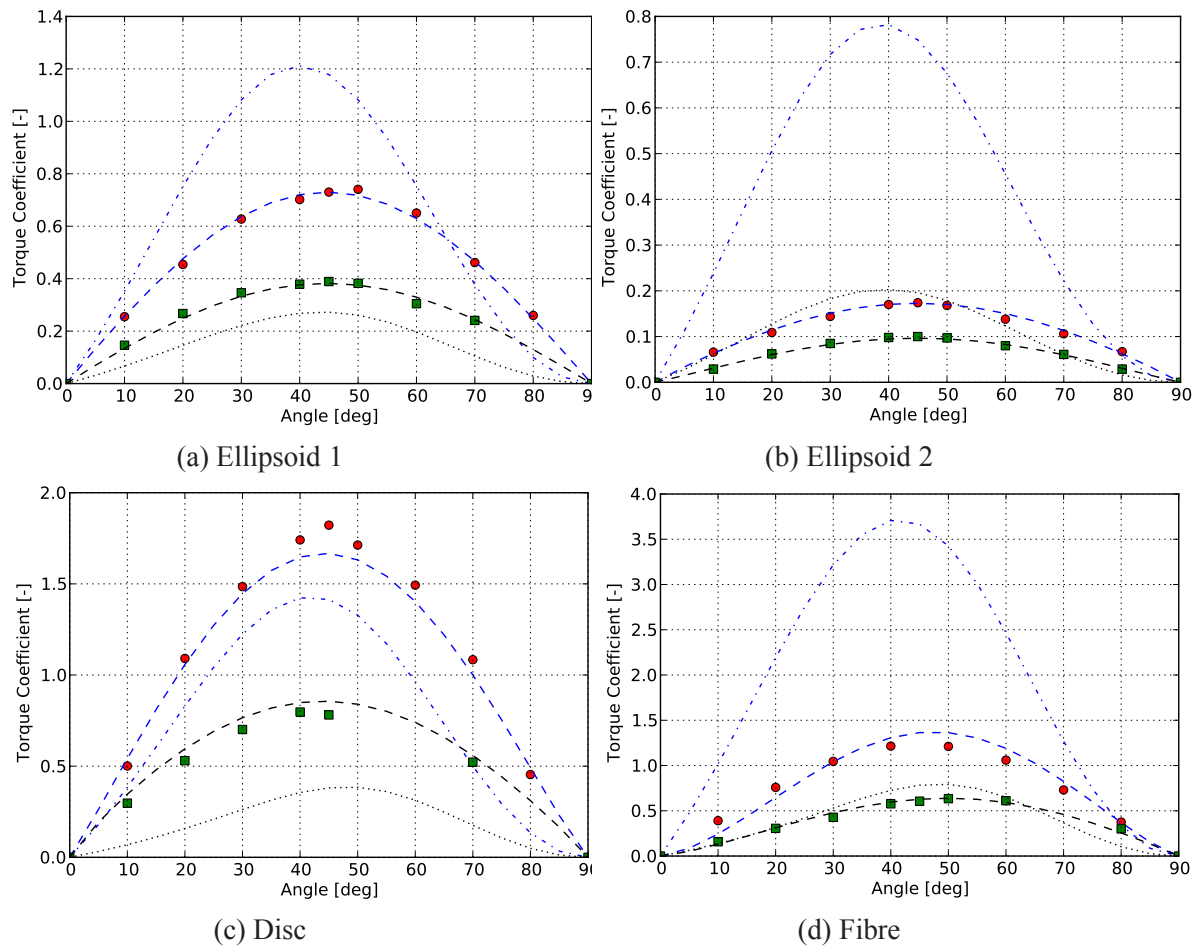


Figure 6.13: Torque coefficients of the particles as a function of angle of incidence. (\circ) simulation at $Re = 10$; ($- - -$) present correlation (Eq. 6.22 at $Re = 10$; (\cdots) equation 6.23 at $Re = 10$; (\square) simulation at $Re = 300$; ($- - -$) present correlation (Eq. 6.22 at $Re = 300$; (\cdots) equation 6.23 at $Re = 300$.

of the predictions of the non-spherical particles behaviour in a fluid.

Rotational torque

The second type of torque is a rotational torque, originating from the relative rotation of the particle with respect to the fluid. Simulations of the rotating particles have been performed in order to determine the counter-rotational torque coefficient correlations. Two modes of rotation have been analysed: around axis of symmetry (mode 1) and around a perpendicular axis (mode 2). The fluid in the simulations is initially at rest. The particle centre of gravity is fixed at the centre of the domain, while the non-spherical particle itself is allowed to rotate with fixed angular velocity.

Fig. 6.14 illustrates the behaviour of the rotational torque coefficient, defined in equation 6.10, at the steady state, with changing rotational Reynolds number Re_R for mode 1 of rotation. The obtained values are compared to correlation for rotating spheres proposed by Dennis *et al.* [19]. The non-spherical

particles show similar dependence on Re_R as the sphere. The magnitude of the coefficients corresponds to the cross-sectional area of particle, with disc's rotational coefficient being the highest, while fibre experiencing the smallest counter-rotational torque.

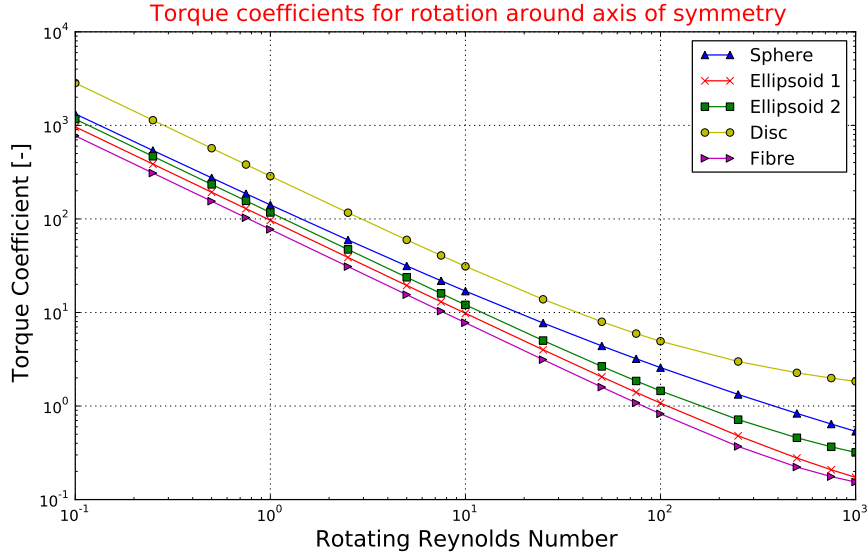


Figure 6.14: Mode 1 rotational torque coefficients for the analysed particles as a function of rotational Reynolds number. Dennis *et al.* [19] correlation used for the sphere.

An analogous analysis can be performed for the second mode of rotation. The torque coefficients after averaging for many time-steps are shown in Fig. 6.15. A further analysis of the results shows that this approach is not as straightforward as in the previous case. When a spherical particle rotates around its axis of symmetry, the effective volume it occupies remains at the same position. This is, however, not the case for non-spherical bodies rotating in the second mode of rotation, as they tend to create a volume of non-steady rotating fluid, much bigger than the volume of the particle. The resulting steady torque coefficient is then not only related to the torque associated with the *initial* rotation of the particle, but with the torque associated with the particle rotation given the surrounding fluid is rotating as well. Therefore, although being a somewhat arbitrary choice, the torque resulting from the *onset* of rotation of the particle is determined and used to determine the fit parameters.

The rotating torque coefficients resulting from the onset of rotation are presented in Fig. 6.16, and compared to the prediction for a rotating sphere [19]. The figure shows higher torque coefficients than those seen in Fig. 6.15, which is expected since in the previous case there is an established flow field around the particle. Therefore, direct comparison between results for non-spherical particles and correlation for rotating spheres given by Dennis *et al.* [19] cannot be made. Particles with higher aspect ratios are subjected to larger torques with the fibre having the highest torque coefficients in a given flow situation.

Both rotation modes show similar behaviour in terms of the change in magnitude of the rotational torque coefficient with growing Re_R . Therefore they can be correlated by the following relationship,

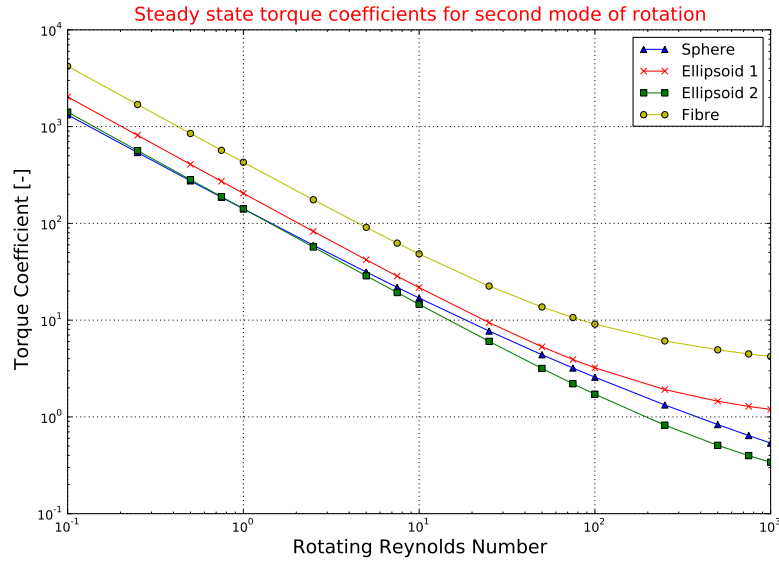


Figure 6.15: Mode 2 rotational torque coefficients for analysed shapes as a function of rotational Reynolds number for a steady-state flow situation. The Dennis et al. [19] correlation is shown for the spherical particle.

providing good data fit:

$$C_R = r_1(Re_R)^{r_2} + \frac{r_3}{(Re_R)^{r_4}} \quad (6.26)$$

The specific fit coefficients for each particle type rotating around its axis of symmetry are given in Table 6.5, while parameters for second mode of rotation are given in Table 6.6.

Table 6.5: The values of fit parameters for equation for the rotational torque coefficient along the axis of symmetry of the particle resulting from the DNS simulations of analysed particles.

Coefficient	Ellipsoid 1	Ellipsoid 2	Disc	Fibre
r_1	0.23	0.573	3.812	0.024
r_2	-0.116	-0.154	-0.13	0.168
r_3	96.378	116.61	283.03	77.314
r_4	1.0	1.0	1.0	1.0

6.1.6 Conclusions

The behaviour of the interaction of non-spherical particles with a fluid flow is a complex phenomenon, even for axis-symmetric particles in a uniform flow. Shape has a great influence on the behaviour of the particle, not only by changing the values of the experienced forces and torques but also by shifting the

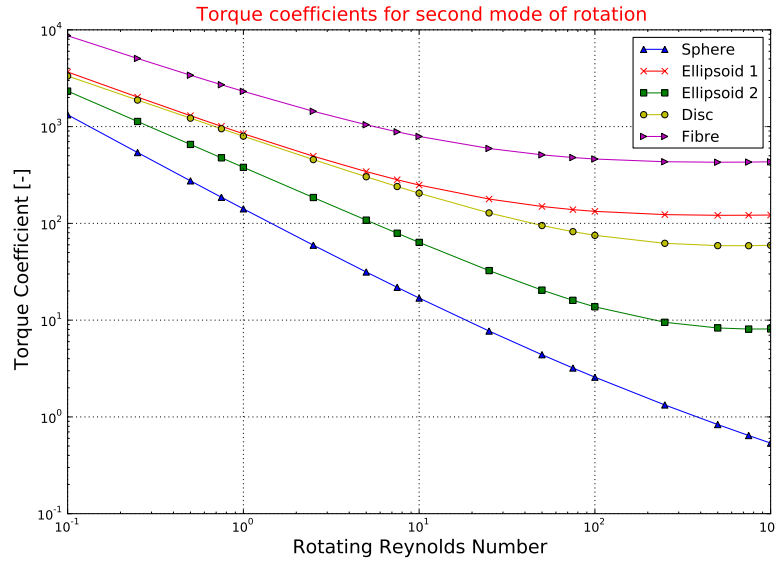


Figure 6.16: Mode 2 rotational torque coefficients for analysed shapes as a function of rotational Reynolds number for the *onset* of rotation. The Dennis et al. [19] correlation for the spherical particle is shown.

Table 6.6: The values of fit parameters for equation for the torque coefficients along the axis which are not axisymmetric of the particle resulting from the DNS simulations of analysed particles.

Coefficient	Ellipsoid 1	Ellipsoid 2	Disc	Fibre
r_1	71.03	1.244	13.31	239.76
r_2	0.069	0.239	0.189	0.075
r_3	773.04	378.12	783.05	2074.02
r_4	0.67	0.789	0.628	0.612

Reynolds number at which the transition to unsteady flow occurs. The flow field also strongly depends on the angle of incidence between the particle and the incoming fluid velocity.

Even though there have been some previous attempts to characterise the flow past an arbitrarily-shaped particle, no complete model is available. Literature provides relatively good predictions for the drag coefficient, but the predictions for both lift and torques have poor accuracy.

The present work analyses the results from a large number of TDNS of flows past various non-spherical particles at different angles of incidence and different rotation, in the range of Reynolds numbers typical for turbulent gas-solid flow. The numerical study was performed using the mirroring immersed boundary method, which allowed to obtain accurate results in an efficient way.

The analysis has resulted in newly proposed equations and fit parameters for four different non-spherical particles to predict the drag coefficient, the lift coefficient and two types of torque coefficients, which all show an excellent agreement with the TDNS results. These equations predicting the force and

torque coefficients can be applied to large scale simulations with many particles, to accurately elucidate their behaviour in a surrounding fluid flow field.

6.2 Modelling the motion of axis-symmetric particles

Simulation of particulate flows in Euler-Lagrange framework requires computation of motion of each particle in terms of translation as well as rotation. This usually involves introduction of additional coordinate systems x and x' illustrated in Fig. 6.17. The inertial coordinate system x has the same orientation as the original, fluid reference frame but is fixed at the particle centre of mass. The particle coordinate system x' is also fixed at the particle centre of mass, however it is aligned with the particle principle axes.

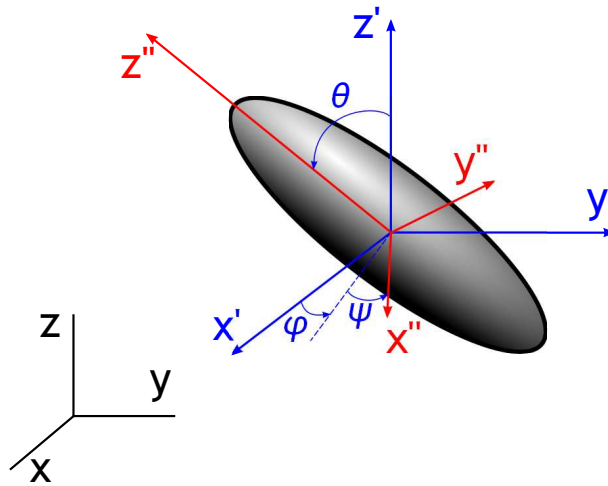


Figure 6.17: The inertial (x) and particle (x') coordinate systems.

The coordinate systems x and x' are related by a rotation matrix A as:

$$x' = Ax \tag{6.27}$$

where the transformation matrix is defined in [34] by means of Euler angles:

$$A = \begin{pmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & \sin \theta \\ \sin \theta \cos \psi & \sin \theta \sin \psi & \cos \theta \\ -\sin \psi & \cos \psi & 0 \end{pmatrix} \tag{6.28}$$

Introduction of additional reference frames for every particle is a popular approach widely used for simulation of flows with non-spherical particles [23, 25, 79, 121].

6.2.1 Equations of motion

The equations describing the motion of a non-spherical particle can be written, following the formulation given by [25], in the following form:

$$m_p \frac{dv}{dt} = F \quad (6.29)$$

$$\begin{aligned} I_x \frac{d}{dt} \begin{pmatrix} x \\ y \\ z \end{pmatrix} &= T_x \\ I_y \frac{d}{dt} \begin{pmatrix} y \\ z \\ x \end{pmatrix} &= T_y \\ I_z \frac{d}{dt} \begin{pmatrix} z \\ x \\ y \end{pmatrix} &= T_z \end{aligned} \quad (6.30)$$

where m_p is the particle mass, while I_j is the moment of inertia in the j direction. Note that the particle motion occurs in the non-inertial frame while the rotation is evaluated in the particle coordinate system.

6.2.2 Forces on the particles

Since both particle mass and the tensor of inertia are known simulation parameters, it is obvious that the quality of the prediction of the particle motion depends on the accuracy of the force and torque calculation.

In order to study the various components of the force vector F , equation 6.29 can be rewritten in a form suggested by Maxey and Riley [83]:

$$m_p \frac{dv}{dt} = F_{\text{aerodynamic}} + F_{\text{gravity}} + F_{\text{press gradient}} + F_{\text{virtual mass}} + F_{\text{history}} \quad (6.31)$$

According to the order-of-magnitude estimates presented by Lazaro and Lasheras [69], in a case of small, but heavy particles in a dilute suspension, only the aerodynamic and inertial terms have a significant contribution to the force vector. The remaining components, *i.e.* force due to the pressure gradient, the virtual mass force, and the Basset history force can be discarded in such simulations.

Calculation of the torque acting on the particle is somewhat easier, since it can be approximated as the sum of the pitching and counter-rotational/resistance torque. The pitching torque is generated if the centre of pressure, where the force is applied, does not coincide with the body centre of mass. The resistance torque on the other hand, describes the fluid response to the body rotation.

The main challenge when evaluating the forces and torques on a non-spherical particle is taking into account their dependence on the particle orientation with respect to the flow velocity. Several strategies are adopted for the computation of the force vector. Some researchers (*e.g.* [45]) focus only on the drag and buoyancy acting on the particles, neglecting the lift component of the aerodynamic force. Whereas this approach might be appropriate for dense flows it is not the preferred one for a more general case. Two different strategies computing the complete force vector are discussed below.

Resistance tensor approach

Evaluation of the force vector by means of the so called resistance tensor, defined by Brenner [7] is one of the most common approaches. The main advantage of this technique is its elegance and ease of implementation. The force vector is defined as $F = \mathbf{K}u_p$, where the resistance \mathbf{K} tensor is obtained in the particle reference frame:

$$\mathbf{K} = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{pmatrix} \quad (6.32)$$

The diagonal elements of the resistance tensor are [79]:

$$k_x = k_y = \frac{16(\alpha^2 - 1)^{3/2}}{[(2\alpha^2 - 3)\ln(\alpha + \frac{\alpha^2 - 1}{2})] + \frac{\alpha^2 - 1}{2}} \quad (6.33)$$

$$k_z = \frac{8(\alpha^2 - 1)^{3/2}}{[(2\alpha^2 - 1)\ln(\alpha + \frac{\alpha^2 - 1}{2})] + \frac{\alpha^2 - 1}{2}} \quad (6.34)$$

where α is the particle aspect ratio. The resistance tensor in the inertial coordinate frame is computed using the rotation matrix as $\mathbf{K} = \mathbf{A}^t \mathbf{K} \mathbf{A}$.

This approach not only allows to calculate the force in three-dimensions, but also takes into account the orientation of the particle. Using the resistance tensor assumes, however, that the magnitude of the force stays the same regardless of the particle orientation. While this may be valid for low Reynolds number flows, it is not true in a general case as shown in the simulations presented earlier in this chapter.

Nevertheless, it would be theoretically possible to adopt the resistance tensor for computation of the force for a general flow past a non-spherical particle. In this case, however, the resistance tensor would have to lose its attractive properties, independence on the orientation and symmetry. An alternative technique is therefore required.

Force decomposition approach

In aerodynamic problems the force is usually decomposed into the drag component, acting in the direction of the undisturbed velocity direction, and a perpendicular lift force. Similar strategy is proposed for simulation of the flow with non-spherical particles. This approach requires introduction of additional "velocity" coordinate system x_u where the z_u axis is aligned with the flow velocity, while x_u lies on the same plane as the velocity and one of the particle axes.

The force vector in such coordinate system is given by:

$$F_u = \frac{1}{2} \rho_f A_r u_p^2 \begin{pmatrix} C_{Lx}(u_u, u Re) \\ C_{Ly}(u_u, u Re) \\ C_D(u_u, u Re) \end{pmatrix} \quad (6.35)$$

where ρ_f is the fluid density, u_p is the particle velocity relative to the flow, while A_r is the reference area, usually based on equivalent diameter and calculated as $A_r = \frac{1}{4} d_{eq}^2$. The force coefficients C_D and C_L

are functions of the orientation angles θ_u , ϕ_u and the Reynolds number.

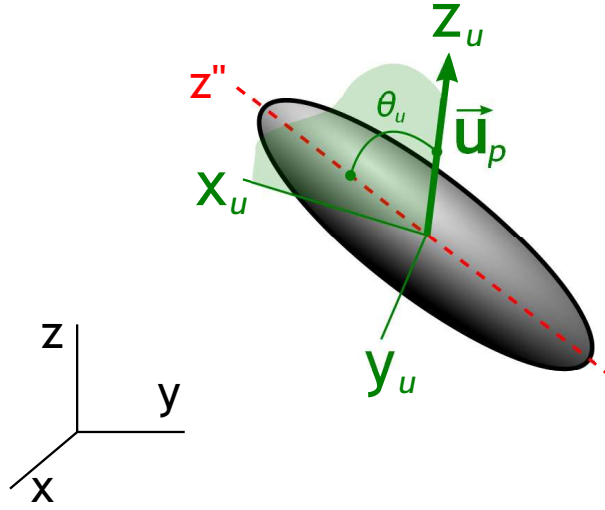


Figure 6.18: The velocity-based coordinate system.

This formulation can be simplified even further, if an axis-symmetric particle is considered. Whereas the z_u axis is still aligned with the flow velocity, the x_u axis lies on the same plane as the particle principle axis (Fig. 6.18). The force vector in such case has a following form:

$$F_u = \frac{1}{2} \rho A_r u_p^2 \begin{pmatrix} C_L(\theta_u, Re) \\ 0 \\ C_D(\theta_u, Re) \end{pmatrix} \quad (6.36)$$

As seen in the above equation the perpendicular lift has only one component and both force coefficients depend only on one angle, referred to as the angle of incidence. The aerodynamic torque, caused by the misalignment of the particle centre of mass and the centre of pressure, velocity coordinate system has only one component and is evaluated as:

$$T_{y_u} = \frac{1}{2} \rho A_r d_p^2 u_p^2 C_T(\theta_u, Re) \quad (6.37)$$

The velocity coordinate system is related to the particle coordinate system by a rotation matrix \mathbf{R}_u , hence the force vector in the particle reference frame is given by $F = \mathbf{R}_u^T F_u \mathbf{R}_u$. Similar procedure can be applied for calculation of aerodynamic torque in the particle coordinate system. Therefore the force in the inertial reference frame will have a form:

$$F = \mathbf{A}^T \mathbf{R}_u^T F_u \mathbf{R}_u \mathbf{A} \quad (6.38)$$

Since the rotation of the particle is computed in the particle coordinate system the counter rotational torques are added to the transformed aerodynamic torque and the rotation is evaluated in that reference frame.

The methodology presented above is compatible with the force and torque coefficients formulae derived in the first part of this chapter. The combination of the force decomposition approach together with the derived force and torques components were applied for simulations of a channel flow with non-spherical particles.

7 Summary and outlook

7.1 Summary

The current thesis presents the summary of the work performed on the *Fundamental understanding of turbulent gas-solid flows* research project. Gas-solid flows are abundant both in nature and in a range of industrial applications, therefore ability to accurately predict their behaviour is of high importance.

The main goal of the project is to develop a successful methodology for True Direct Numerical Simulations of flows with solid particles, where all the flow scales along with the particle boundary layers are resolved. TDNS approach allows for detailed analysis of various flow cases and can be adopted to develop better understanding of the physical processes driving particle-fluid interaction.

The developed approach is based on the principles of a *ghost cell* Immersed Boundary Method. The Immersed boundary Method uses separate computational grids for the fluid and the simulated particles, what makes it very efficient approach for analysis of flows with moving particles or complex shapes. The fluid-particle coupling is achieved by modifying the fluid equations in the vicinity of the particle.

A no-slip velocity boundary condition is imposed on the surface of the body by setting the velocities of the fluid cells inside the body, so called *ghost cells*. The value of the *ghost cell* velocity ensures that the interpolated velocity at the surface of the IB is equal to the velocity of the simulated body.

One of the main drawbacks experienced by the Immersed Boundary Method is the occurrence of spurious pressure oscillations, observed in simulations of moving bodies. These fluctuations are related to the change of the numerical treatment of the fluid cells as the body moves through the domain. The oscillations increase as the time-step of the simulation is decreased. Although the fluctuations have little effect on the average magnitude of predicted fluid-particle forces, they can lead to stability issues and non-physical flow behaviour.

Addressing the issue of the spurious pressure oscillations is the main motivation for development of the current numerical technique. The proposed approach combines setting the no-slip velocity boundary condition by means of mirroring the flow through the IB surface with a *cut-cell* approach applied to the continuity equation.

Application of the *cut-cell* approach to the continuity equation involves three significant modifications to the flow equations in the vicinity of the particle. First, the continuity equation in the fluid cells close to the IB is discretised using the accurate cell face areas available for the fluid flow. The exact value of the velocity at the *cut-cell* faces is calculated by means of 8-point tri-linear interpolation from

the surrounding cells. Additionally, the flow induced by the moving Immersed Boundary is taken into account.

The second modification enforced by the *cut-cell* technique involves addition of the influence of the IB motion into the discretisation of the convective terms in the momentum equations of the fluid cells cut by the IB interface. Finally, the *cut-cell* technique requires recalculation of the explicit convective coefficients in the momentum equations each time the particle moves, prior to the solution of the main flow equations.

Implementation of the *cut-cell* technique presented in this thesis required development of a new parallel triangulation library, MFTL, described in chapter 4. The library is responsible for all the operations involving the triangulated surface of the IB. It also enables determination of surface intersection points and calculation of accurate flow areas in the *cut-cells*.

The proposed method is compared to several different IB implementations, by means of a set of tests designed to evaluate the performance particular features of the analysed techniques. The *cut-cell* approach, designed in the current research project, is able to accurately predict the flow behaviour in all of the studied test cases. Moreover, a significant reduction of the spurious pressure oscillations is achieved by this technique.

One of the Immersed Boundary Methods implementations studied during the current research project is also applied to investigate flows past non-spherical particles in order to enhance the understanding of the flow-particle interaction. The results of a range of performed simulations are applied to establish shape specific correlations for the drag, lift and torques experienced by various non-spherical particles as functions of Reynolds numbers and incidence angles. A model for description of the motion of such particles in the point-particle framework is also developed.

7.2 Outlook

The work presented in this thesis shows a great potential for further improvements and applications. Although the proposed IB technique shows promising results, it requires further testing and optimisation. Also, implementation of additional features, such as collision modelling in order to achieve full functionality. Finally there is a number of applications where TDNS with IBM might provide valuable information about the flow behaviour. Discussion of aforementioned topics is presented below.

7.2.1 Improvements to the computational method

A novel Immersed Boundary Method implementation, applying a *cut-cell* approach to solve the continuity equation in the vicinity of the particle, is proposed in this thesis. Although the method is able to accurately predict the flow behaviour in number of test cases, there still exists an opportunity for further improvements both in terms of accuracy and the computational efficiency of the developed technique.

The precision of the force prediction when the flow variables are extrapolated to the surface depends strongly on the grid and IB surface refinement. The force calculation framework is also reported to under-predict the forces acting on the body, when calculating the force from an analytically prescribed

flow field (chapter 5.1). A new force calculation technique is currently under development and expected to improve the force prediction. This method is based on the direct extrapolation of the flow properties, *i.e.* pressure and velocity gradients, to the IB surface (omitting the additional interpolation to the auxiliary IB points step). It is expected that eliminating the interpolation to auxiliary points will increase the accuracy of the method by avoiding errors introduced by the interpolation.

Although the proposed methodology, based on the *cut-cell* approach for solution of the continuity equation, enables a significant reduction of the spurious pressure oscillations observed during simulations of moving bodies, the problem is not completely solved yet. Small oscillations, related mainly to the occurrence of *dead* cells are still observed in the presented results. When a cell is absorbed by a moving IB, the mass flux calculated in the previous time-step can no longer be used for the discretisation of the convective terms in the momentum equation. The current method accommodates for this fact by performing a sub-step calculation of the flow field and updating the mass fluxes. It can be expected that improving the accuracy of the mass fluxes prediction, for example by applying a *cut-cell* approach to the momentum equation, might decrease the magnitude of the pressure oscillations even further.

Additionally, the computational performance of the numerical procedure can be significantly improved by optimizing the numerical code. At the moment the *cut-cell* method implementation is limited to a one processor, single block simulation on Cartesian grids. A full multi-block multi-processor capability is currently under development, and will significantly increase the speed of the computation, which will have a strong impact on the capability of the code to conduct TDNS of large scale complex flow cases. Enabling multi-core functionality requires allowing each processor to access the flow information from the fluid cells, used for interpolation of fluid variables to cell centres, which belong to other CPU's.

Moreover, optimization of certain functions, *e.g.* auxiliary point generation or surface intersection determination, will lead to a considerable code speed-up. Currently the auxiliary and imaginary points are deleted and regenerated every time the IB moves. Regeneration only of the necessary points, *e.g.* occurring when the IB moves through a processor cut, and recalculating the remaining ones will eliminate the computationally expensive memory management operations.

Finally, because the explicit convection coefficients in the momentum equations need to be recalculated each time the particle moves, the proposed IB implementation solves the flow equations twice per each time-step, what has a great impact on computational cost of a simulation. An alternative approach where the mass fluxes, used for the convective terms discretisation, are estimated locally, without the necessity of recalculating the flow in the entire domain, is desirable. Application of a faster type of the convective terms estimation might reduce the computational cost related to this process and speed-up the code considerably.

7.2.2 Additional capabilities

Apart from the improvements in the performance and the speed of the developed method, implementation of additional functionality is strongly recommended. Currently, the *cut-cell* approach is only applicable for bodies of convex shapes (spheres, ellipsoids, cylinders, etc.), addition of more complex body treatment would be a valuable and easily implementable improvement.

More importantly, however, a frame-work to work with cases where particles are close to each other or close the domain boundaries needs to be developed and implemented. The numerical technique presented in this thesis can be extended to handle such cases. The main challenge is a treatment of instances, when either the imaginary points, used to set the *ghost cell* velocity, or the IB auxiliary points, applied in the force calculation technique, fall outside of the flow domain or inside another body. Such points can no longer be used for the calculations and require appropriate procedures, that involve either point repositioning or exclusion from the algorithm. Significant work is currently performed on this topic.

Additionally, implementation of collision model and contact forces is necessary for simulations of multiple moving particles. This might involve selection of a number of surface triangles to be treated in a special way in order to account for the local collision effects. Collisions both between the particles as well as with domain walls need to be considered. Apart from addition of the collision forces, a model of sub-grid forces experienced by particles in close proximity is desirable. This however requires establishment of an appropriate physical model to predict such forces.

7.2.3 Possible applications

The main motivation for development of the current Immersed Boundary Method is its capability to perform TDNS of particles moving in a flow. Although such simulations are limited in size due to their high computational costs, they can still provide valuable physical insight, as shown in chapter 6 of this thesis. Some possible applications of the current method are suggested below.

Shape-specific correlations for calculation of the forces and torques acting on four different non-spherical particles were developed during the current research project. Further extension of the obtained results to include shape parameters, *e.g.* ellipsoid aspect-ratio, into the correlations is suggested. Development of such formulas would be highly valuable, since the accuracy of currently available models is very limited.

As discussed in the previous section, implementation of the sub-grid forces experienced by bodies in close proximity would be an important addition to the current capabilities of the proposed method. Development of an appropriate force model requires performing a number of simulations of particles moving with respect to each other. A very fine grid is desired in this type of study in order to accurately capture the effects even if the particles are very close. Nonetheless, the current code capability is well suited for analysis of this phenomenon. Parametric study of magnitude of forces acting on particles moving with respect to another over a range of Reynolds numbers along with particle relative sizes and velocities, would provide sufficient data to develop force correlations describing the particle-particle interaction.

Another example of possible application of the current method is the investigation of shear flow past a particles of various shapes. Although shear flows past spherical particles have been extensively studied before, there is no simple expression to characterise influence of the non-uniform velocity profile on the forces acting on the particle, which can be applied across a range of Reynolds numbers. Development of such an expression would be very useful, since gas-solid flows with non-uniform velocity profiles are very common, for instance in pneumatic transport applications. Compact model of shear flow influence

on the particle behaviour would allow to enhance the prediction of the particle trajectories near the pipe walls, for example. Similarly as before a study of shear flows with immersed bodies over a range of parameters (*e.g.* Reynolds number, shear rate) can be performed in order to gather force predictions at various conditions. Then the data can be used to develop correlations describing forces acting on a particle in a shear flow.

Only a small selection of numerous possible uses of the current IBM are described above. Nevertheless, the potential of the method and importance of its applications can be easily seen.

Bibliography

- [1] P. Angot, C.-H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81:497–520, 1999.
- [2] 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(8):2712–2738, May 2009.
- [3] E. Balaras. Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations. *Computers and Fluids*, 33(3):375–404, March 2004.
- [4] Satish Balay, Jed Brown, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Users Manual. 2011.
- [5] D. Besnard and F. H. Harlow. Nonspherical particles in two-phase flow. *International journal of multiphase flow*, 12(6):891–912, 1986.
- [6] C. E. Brennen. *Fundamentals of Multiphase Flows*. Cambridge University Press 2005, Pasadena, California, 2005.
- [7] H. Brenner. The Stokes resistance of an arbitrary particle. *Chemical Engineering Science*, 18(1):1–25, 1963.
- [8] Nian-Sheng Cheng. Comparison of formulas for drag coefficient and settling velocity of spherical particles. *Powder Technology*, 189(3):395–398, February 2009.
- [9] Y. Chen and O. Botella. The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. *Journal of Computational Physics*, 229(4):1043–1076, February 2010.
- [10] P. Cherukat. A computational study of the inertial lift on a sphere in a linear shear flow field. *International Journal of Multiphase Flow*, 25(1):15–33, February 1999.
- [11] R. P. Chhabra, L. Agarwal, and N. K. Sinha. Drag on non-spherical particles: an evaluation of available methods. *Powder Technology*, 101(3):288–295, March 1999.
- [12] J.-I. Choi, R. C. Oberoi, J. R. Edwards, and J. A. Rosati. An immersed boundary method for complex incompressible flows. *Journal of Computational Physics*, 224(2):757–784, June 2007.

- [13] M.-H. Chung. Cartesian cut cell approach for simulating incompressible flows with rigid bodies of arbitrary shape. *Computers and Fluids*, 35(6):607–623, July 2006.
- [14] D. K. Clarke, H. A. Hassan, and M. D. Salas. Euler Calculations for Multielement Airfoils Using Cartesian Grids. *AIAA Journal*, 24(3):1986, 1985.
- [15] R. Clift, J. R. Grace, and M. E. Weber. *Bubbles, Drops and Particles*. Academic Press, New York, 1978.
- [16] C. T. Crowe. *Multiphase flow handbook*. CRC, 2005.
- [17] C. T. Crowe, M. Sommerfeld, and Y. Tsuji. *Multiphase Flows with Droplets and Particles*. CRC Press, 1998.
- [18] J. Deng, X.-M. Shao, and A.-L. Ren. A new modification of the immersed-boundary method for simulating flows with complex moving boundaries. *Int. J. Numer. Meth. Fluids*, pages 1195–1213, 2006.
- [19] S. C. R. Dennis, S. N. Singh, and D. B. Ingham. The steady flow due to a rotating sphere at low and moderate Reynolds numbers. *Journal of Fluid Mechanics*, 101(02):257, April 1980.
- [20] S. Elghobashi. On predicting particle-laden turbulent flows. *Applied Scientific Research*, 52(4):309–329, June 1994.
- [21] S. Ergun. Fluid flow through packed columns. *Chem. ENG. Prog.*, 48:89–94, 1952.
- [22] E. A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations. *Journal of Computational Physics*, 161(1):35–60, June 2000.
- [23] F.-G. Fan and G. Ahmadi. Wall Deposition of Small Ellipsoids From Turbulent Air Flows—a Brownian Dynamics Simulation. *Journal of Aerosol Science*, 31(10):1205–1229, October 2000.
- [24] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer Verlag, Berlin Heidelberg New York, 3. edition, 2002.
- [25] I. Gallily and A. H. Cohen. On the orderly nature of the motion of nonspherical aerosol particles. II. Inertial collision between a spherical large droplet and an axially symmetrical elongated particle. *Journal of Colloid and Interface Science*, 68(2):338–356, 1979.
- [26] G. H. Ganser. A rational approach to drag prediction of spherical and nonspherical particles. *Powder Technology*, 77:143–152, 1993.
- [27] F. Gao, D. M. Ingram, D. M. Causon, and C. G. Mingham. The development of a Cartesian cut cell method for incompressible viscous flows. *International Journal for Numerical Methods in Fluids*, 54:1033–1053, 2007.

- [28] R. Ghias, R. Mittal, and H. Dong. A sharp interface immersed boundary method for compressible viscous flows. *Journal of Computational Physics*, 225(1):528–553, July 2007.
- [29] A. Gilmanov and S. Acharya. A hybrid immersed boundary and material point method for simulating 3D fluid-structure interaction problems. *International Journal for Numerical Methods in Fluids*, 56(12):2151–2177, 2008.
- [30] 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(2):457–492, August 2005.
- [31] A. Gilmanov, F. Sotiropoulos, and E. Balaras. A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids. *Journal of Computational Physics*, 191(2):660–669, November 2003.
- [32] R. Glowinski, T. W. Pan, and T. I. Hesla. A distributed Lagrange multiplier/fictitious domain method for the simulation of flow around moving rigid bodies: application to particulate flow. *Computer Methods in Applied Mechanics and Engineering*, 184:241–267, 2000.
- [33] D. Goldstein, R. Handler, and L. Sirovich. Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics*, 105:354–366, 1993.
- [34] H. Goldstein, C. P. Poole, and J. Safko. *Classical mechanics*. 1980.
- [35] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.
- [36] 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(1):10–49, April 2007.
- [37] C. Gunther, D. Hartmann, and M. Meinke. A Level-set based cut-cell method for flows with complex moving boundaries. In *V European Conference on Computational Fluid Dynamics*, number June, 2010.
- [38] C. Gunther, L. Schneiders, M. Meinke, W. Schröder, and D. Hartmann. A Cartesian cut-cell method for sharp moving boundaries. In *20th AIAA Computational Fluid Dynamics Conference*, number June, pages 1–23, 2011.
- [39] S. Haeri and J. S. Shrimpton. 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, April 2012.

- [40] A. Haider and O. Levenspiel. Drag coefficient and terminal velocity of spherical and nonspherical particles. *Powder technology*, 58(1):63–70, 1989.
- [41] M. Hartman, O. Trnka, and K. Svoboda. Free Settling of Nonspherical Particles. *Industrial and Engineering Chemistry Research*, 33(8):1979–1983, August 1994.
- [42] D. Hartmann. A General Formulation of Boundary Conditions on Cartesian Cut-Cells for Compressible Viscous Flow. In *19th AIAA Computational Fluid Dynamics*, number June, 2009.
- [43] D. Hartmann, M. Meinke, and W. Schröder. An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods. *Computers and Fluids*, 37:1103–1125, 2008.
- [44] D. Hartmann, M. Meinke, and W. Schröder. A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids. *Computer Methods in Applied Mechanics and Engineering*, 200(9-12):1038–1052, February 2011.
- [45] J. E. Hilton, L. R. Mason, and P. W. Cleary. Dynamics of gassolid fluidised beds with non-spherical particle geometry. *Chemical Engineering Science*, 65(5):1584–1596, March 2010.
- [46] S. F. Hoerner. *Fluid-dynamic drag : practical information on aerodynamic drag and hydrodynamic resistance*. Hoerner, Brick Town, 1965.
- [47] A. Holzer and M. Sommerfeld. New simple correlation formula for the drag coefficient of non-spherical particles. *Powder Technology*, 184(3):361–365, June 2008.
- [48] A. Holzer and M. Sommerfeld. Lattice Boltzmann simulations to determine drag, lift and torque acting on non-spherical particles. *Computers and Fluids*, 38(3):572–589, March 2009.
- [49] H. H. Hu. Direct simulation of flows of solid-liquid mixtures. *International Journal of Multiphase Flow*, 22(2):335–352, 1996.
- [50] H. H. Hu, N. A. Patankar, and M. Y. Zhu. Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique. *Journal of Computational Physics*, 169(2):427–462, 2001.
- [51] X. Y. Hu, B. C. Khoo, N. A. Adams, and F. L. Huang. A conservative interface method for compressible flows. *Journal of Computational Physics*, 219(2):553–578, December 2006.
- [52] W. X. Huang and H. J. Sung. Improvement of mass source/sink for an immersed boundary method. . . . *journal for numerical methods in fluids*, (October 2006):1659–1671, 2007.
- [53] F. Ilincă and J.-F. F. Hétu. A finite element immersed boundary method for fluid flow around moving objects. *Computers and Fluids*, 39(9):1656–1671, October 2010.
- [54] G. B. Jeffery. The motion of ellipsoidal particles immersed in a viscous fluid. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 102(715):161–179, 1922.

- [55] H. Ji, F.-S. Lien, and E. Yee. Numerical simulation of detonation using an adaptive Cartesian cut-cell method combined with a cell-merging technique. *Computers and Fluids*, 39(6):1041–1057, June 2010.
- [56] A. Johnson and T. E. Tezduyar. 3D Simulation of fluid-particle interactions with the number of particles reaching 100. *Computer Methods in Applied Mechanics and Engineering*, 145(3-4):301–321, June 1997.
- [57] A. Johnson and T. E. Tezduyar. Methods for 3D computation of fluid-object interactions in spatially periodic flows. *Computer methods in applied mechanics and Engineering*, 190:3201–3221, 2001.
- [58] M. W. Jones. 3D Distance from a Point to a Triangle. Technical Report February, Department of Computer Science, University of Wales, Swansea, 1995.
- [59] K. Khadra, P. Angot, S. Parneix, and J.-P. Caltagirone. Fictitious domain approach for numerical modelling of Navier Stokes equations. *International Journal for Numerical Methods in Fluids*, 34:651–684, 2000.
- [60] D. Kim and H. Choi. Immersed boundary method for flow around an arbitrarily moving body. *Journal of Computational Physics*, 212(2):662–680, March 2006.
- [61] J. Kim, D. Kim, and H. Choi. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, 171(1):132–150, 2001.
- [62] M. P. Kirkpatrick, S. W. Armfeld, and J. H. Kent. A representation of curved boundaries for the solution of the Navier-Stokes equations on a staggered three-dimensional Cartesian grid. *Journal of Computational Physics*, 184(2003):1–36, 2003.
- [63] P. K. Kundu and I. M. Cohen. *Fluid Mechanics*. Number v. 10 in Fluid mechanics. Academic Press, 2008.
- [64] D. Kunii and O. Levenspiel. *Fluidization engineering*, volume 2. Butterworth-Heinemann Boston, 1991.
- [65] R. Kurose and S. Komori. Drag and lift forces on a rotating sphere in a linear shear flow. *Journal of Fluid Mechanics*, 384:183–206, 1999.
- [66] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics*, 211:285–309, 1994.
- [67] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results. *Journal of Fluid Mechanics*, 271:311–339, 1994.

- [68] M.-C. Lai and C. S. Peskin. An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity. *Journal of Computational Physics*, 160(2):705–719, May 2000.
- [69] B. J. Lazaro and J. C. Lasheras. Particle dispersion in a turbulent, plane, free shear layer. *Physics of Fluids A: Fluid Dynamics*, 1(6):1035–1044, 1989.
- [70] J. Lee, J. Kim, H. Choi, and K.-S. Yang. Sources of spurious force oscillations from an immersed boundary method for moving-body problems. *Journal of Computational Physics*, 230(7):2677–2695, April 2011.
- [71] J. Lee and D. You. An implicit ghost-cell immersed boundary method for simulations of moving body problems with control of spurious force oscillations. *Journal of Computational Physics*, September 2012.
- [72] C. W. Li and L. L. Wang. An immersed boundary finite difference method for LES of flow around bluff shapes. *Int. J. Numer. Meth. Fluids*, pages 85–107, 2004.
- [73] C.-C. Liao, Y.-W. Chang, C.-A. Lin, and J. M. McDonough. Simulating flows with moving rigid boundary using immersed-boundary method. *Computers and Fluids*, 39(1):152–167, January 2010.
- [74] C.-C. Liao and C.-A. Lin. Simulations of natural and forced convection flows with moving embedded object using immersed boundary method. *Computer Methods in Applied Mechanics and Engineering*, 213-216:58–70, March 2012.
- [75] E. Loth. Drag of non-spherical solid particles of regular and irregular shape. *Powder Technology*, 182(3):342–353, March 2008.
- [76] H. Luo, H. Dai, P. J. S. A. Ferreira de Sousa, and B. Yin. On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries. *Computers and Fluids*, 56:61–76, March 2012.
- [77] S. Majumdar, G. Iaccarino, and P. Durbin. RANS solvers with adaptive structured boundary non-conforming grids. *Center for Turbulence Research Annual Research Briefs*, pages 353–366, 2001.
- [78] M. Mandø and L. Rosendahl. On the motion of non-spherical particles at high Reynolds number. *Powder Technology*, 202(1-3):1–13, 2010.
- [79] C. Marchioli, M. Fantoni, and A. Soldati. Orientation, distribution, and deposition of elongated, inertial fibers in turbulent channel flow. *Physics of Fluids*, 22(3):033301, 2010.
- [80] 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):1–31, November 2005.

- [81] A. Mark, R. Rundqvist, and F. Edelvik. Comparison Between Different Immersed Boundary Conditions for Simulation of Complex Fluid Flows. In *ICMF*, pages 1–9, 2010.
- [82] A. Mark and B. G. M. van Wachem. Derivation and validation of a novel implicit second-order accurate immersed boundary method. *Journal of Computational Physics*, 227:6660–6680, 2008.
- [83] M. R. Maxey and J.R. Riley. Equation of motion for a small rigid sphere in a nonuniform flow. *Physics of Fluids*, 26(4):883, 1983.
- [84] M. Meyer, a. Devesa, S. Hickel, X. Y. Hu, and N .A. Adams. A conservative immersed interface method for Large-Eddy Simulation of incompressible flows. *Journal of Computational Physics*, 229(18):6300–6317, September 2010.
- [85] R. Mittal, H. Dong, M. Bozkurtas, 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(10):4825–4852, January 2008.
- [86] R. Mittal and G. Iaccarino. Immersed Boundary Methods. *Annual Review of Fluid Mechanics*, 37(1):239–261, January 2005.
- [87] J. Mohd-Yusof. Combined immersed boundary/B-spline methods for simulation of flow in complex geometries. *Center for Turbulence Research Annual Research Briefs1*, pages 317–328, 1997.
- [88] P. H. Mortensen, H. I. Andersson, J. J. J. Gillissen, and B. J. Boersma. Dynamics of prolate ellipsoidal particles in a turbulent channel flow. *Physics of Fluids*, 20(9):093302, 2008.
- [89] H. Niazmand and M. Renksizbulut. Surface effects on transient three-dimensional flows around rotating spheres at moderate Reynolds numbers. *Computers and Fluids*, 32(10):1405–1433, December 2003.
- [90] J. E. S. Oliveira, A. Mark, and B. G. M. van Wachem. A Fully Coupled and Implicit Immersed Boundary Method for Multiphase Flows. In *ICMF*, number 2003, pages 1–9, 2007.
- [91] J. O’Rourke. *Computational geometry in C*. 1994.
- [92] D. Pan and T.-T. Shen. Computation of incompressible flows with immersed bodies by a simple ghost cell method. *Int. J. Numer. Meth. Fluids*, pages 1378–1401, 2009.
- [93] N. A. Patankar, P. Singh, and D. D. Joseph. A new formulation of the distributed Lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 26:1509–1524, 2000.
- [94] N. Peller, A. L. Duc, F. Tremblay, and M. Manhart. High-order stable interpolations for immersed boundary methods. *International Journal for Numerical Methods in Fluids*, 52:1175–1193, 2006.

- [95] C. S. Peskin. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics*, 10(2):252–271, 1972.
- [96] C. S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, July 2003.
- [97] A. Pinelli, I. Z. Naqavi, U. Piomelli, and J. Favier. Immersed-boundary methods for general finite-difference and finite-volume Navier-Stokes solvers. *Journal of Computational Physics*, 229(24):9073–9091, December 2010.
- [98] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532, November 1983.
- [99] A. M. Roma, C. S. Peskin, and M. J. Berger. An Adaptive Version of the Immersed Boundary Method. *Journal of Computational Physics*, 153(2):509–534, August 1999.
- [100] L. Rosendahl. Using a multi-parameter particle shape description to predict the motion of non-spherical particle shapes in swirling flow. *Applied Mathematical Modelling*, 24(1):11–25, January 2000.
- [101] S. I. Rubinow and J. B. Keller. The transverse force on a spinning sphere moving in a viscous fluid. *Journal of Fluid Mechanics*, 11(03):447–459, 1961.
- [102] L. Schiller and A. Naumann. Über die grundlegenden Berechnungen bei der Schwerkraftaufbereitung. *Ver. Deut. Ing.*, 77:318–320, 1933.
- [103] J. H. Seo and R. Mittal. A Sharp-Interface Immersed Boundary Method with Improved Mass Conservation and Reduced Spurious Pressure Oscillations. *Journal of computational physics*, 230(19):7347–7363, August 2011.
- [104] S. Shin, S. Y. Bae, I. C. Kim, Y. J. Kim, and J. S. Goo. Computations of flow over a flexible plate using the hybrid Cartesian/immersed boundary method. *International Journal for Numerical Methods in Fluids*, 55(3):263–282, 2007.
- [105] M. Sommerfeld, B. G. M. van Wachem, and R. Oliemans. ERCOFTAC Special Interest Group on ”Dispersed Turbulent Multi-Phase Flow”, Best Practice Guidelines. Technical report, 2007.
- [106] J. M. Stockie and B. R. Wetton. Analysis of Stiffness in the Immersed Boundary Method and Implications for Time-Stepping Schemes. *Journal of Computational Physics*, 154(1):41–64, September 1999.
- [107] S. Takiguchi, T. Kajishima, and Y. Miyake. Numerical scheme to resolve the interaction between solid particles and fluid turbulence. *JSME international journal. Series B, fluids and thermal engineering*, 42(3):411–418, 1999.

- [108] 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(11):4012, 2002.
- [109] O. Tropp, A. Tal, and I. Shimshoni. A fast triangle to triangle intersection test for collision detection. *Computer Animation and Virtual Worlds*, 17(5):527–535, 2006.
- [110] Y.-H. Tseng and J. H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, 192(2):593–623, December 2003.
- [111] 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(1):345–380, November 2001.
- [112] M. Uhlmann. First experiments with the simulation of particulate flows. Technical Report No. 1020. Technical report, CIEMAT, Madrid, Spain, 2003.
- [113] M. Uhlmann. An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, 209(2):448–476, November 2005.
- [114] B. G. M. van Wachem, J. C. Schouten, C. M. van den Bleek, R. Krishna, and J. L. Sinclair. Comparative Analysis of CFD Models of Dense Gas Solid Systems. *AIChE Journal*, 47(5), 2001.
- [115] H. Wadell. The coefficient of resistance as a function of Reynolds number for solids of various shapes. *Journal of the Franklin Institute*, 1934.
- [116] D. Wan and S. Turek. An efficient multigrid-FEM method for the simulation of solidliquid two phase flows. *Journal of Computational and Applied Mathematics*, 203(2):561–580, June 2007.
- [117] C. Wen and Y. Yu. Mechanics of fluidization. *Chem. Eng. Prog. Symp. Ser.*, 62(62):100–111, 1965.
- [118] J. Yang and E. Balaras. An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. *Journal of Computational Physics*, 215(1):12–40, June 2006.
- [119] J. Yang and F. Stern. A simple and efficient direct forcing immersed boundary framework for fluidstructure interactions. *Journal of Computational Physics*, 231(15):5029–5061, June 2012.
- [120] T. Ye, R. Mittal, and H. S. Udaykumar. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, 156(2):209–240, December 1999.
- [121] C. Yin, L. Rosendahl, S. Knudsen Kaer, and H. Sorensen. Modelling the motion of cylindrical particles in a nonuniform flow. *Chemical engineering science*, 58(15):3489–3498, 2003.

- [122] M. Zastawny, G. Mallouppas, F. Zhao, and B. G. M. van Wachem. Derivation of drag and lift force and torque coefficients for non-spherical particles in flows. *International Journal of Multiphase Flow*, 39:227–239, October 2012.
- [123] M. Zastawny, B. G. M. van Wachem, and J. E. S. Oliveira. An Immersed Boundary Method for Interacting Particles. In *7th International Conference on Multiphase Flow, ICMF 2010, Tampa, FL, May 30 June 4*, number 4, pages 1–8, 2010.
- [124] N. Zhang and Z.C. Zheng. An improved direct-forcing immersed-boundary method for finite difference applications. *Journal of Computational Physics*, 221(1):250–268, January 2007.