



College of Engineering, Design and physical science

PhD Mechanical Engineering

Title:

Application of Immersed Boundary Method to Flexible Riser Problem

Author:

SeyedHossein MadaniKermani

Supervised by:

Prof. Hamid Bahai
Dr. Jan Wissink

Date:

March 2014

A thesis submitted in fulfilment of the requirements for the award of the degree of Doctor of Philosophy in Mechanical Engineering

Submitted for examination

Abstract

In the recent decades the Fluid-Structure Interaction (FSI) problem has been of great interest to many researchers and a variety of methods have been proposed for its numerical simulation. As FSI simulation is a multi-discipline and a multi-physics problem, its full simulation consists of many details and sub-procedures. On the other hand, reliable FSI simulations are required in various applications ranging from hemodynamics and structural engineering to aero-elasticity. In hemodynamics an incompressible fluid is coupled with a flexible structure with similar density (e.g. blood in arteries). In aero-elasticity a compressible fluid interacts with a stiff structure (e.g. aircraft wing) or an incompressible flow is coupled with a very light structure (e.g. Parachute or sail), whereas in some other engineering applications an incompressible flow interacts with a flexible structure with large displacement (e.g. oil risers in offshore industries). Therefore, various FSI models are employed to simulate a variety of different applications. An initial vital step to conduct an accurate FSI simulation is to perform a study of the physics of the problem which would be the main criterion on which the full FSI simulation procedure will then be based.

In this thesis, interaction of an incompressible fluid flow at low Reynolds number with a flexible circular cylinder in two dimensions has been studied in detail using some of the latest published methods in the literature. The elements of procedures have been chosen in a way to allow further development to simulate the interaction of an incompressible fluid flow with a flexible oil riser with large displacement in three dimensions in future.

To achieve this goal, a partitioned approach has been adopted to enable the use of existing structural codes together with an Immersed Boundary (IB) method which would allow the modelling of large displacements. A direct forcing approach, interpolation / reconstruction, type of IB is used to enforce the moving boundary condition and to create sharp interfaces with the possibility of modelling in three dimensions. This provides an advantage over the IB continuous forcing approach which creates a diffused boundary. And also is considered as a preferred method over the cut cell approach which is very complex in three dimensions with moving boundaries.

Different reconstruction methods from the literature have been compared with the newly proposed method. The fluid governing equation is solved only in the fluid domain using a Cartesian grid and an Eulerian approach while the structural analysis was performed using Lagrangian methods. This method avoids the creation of secondary fluid domains inside the solid boundary which occurs in some of the IB methods. In the IB methods forces from the Eulerian flow field are transferred onto the Lagrangian marker points on the solid boundary and the displacement and velocities of the moving boundary are interpolated in the flow domain to enforce no-slip boundary conditions. Various coupling methods from the literature were selected and improved to allow modelling the interface and to transfer the data between fluid and structure.

In addition, as an alternative method to simulate FSI for a single object in the fluid flow as suggested in the literature, the moving frame of reference method has been applied for the first time in this thesis to simulate Fluid-Structure interaction using an IB reconstruction approach.

The flow around a cylinder in two dimensions was selected as a benchmark to validate the simulation results as there are many experimental and analytical results presented in the literature for this specific case.

Some of the Contributions to knowledge

- Comprehensive and comparative study of the FSI methods considering facilities and physics of the problem to define an algorithm to be able to:
 - To simulate Large displacement/deformation
 - To integrated existing fluid flow and structural codes
 - To be extendable to three dimension
 - To be able to define sharp boundaries and resolve vortices in the flow field
 - To be fitted to the existing computational facilities
- Developing an FSI code based on the comprehensive study to simulate
 - Stationary cases
 - Forced Vibration with prescribe motion
 - Vortex Induced Vibration (VIV)
- Proposing a new interpolation procedure and comparing it with literature
- Characterising the domain parameters affecting Strouhal number, lift and drag coefficients
- Explaining some of discrepancies in the results of the lift and drag coefficient presented in the literature based on parametric study.
- Applying a moving reference frame along with an IB interpolation method to model FSI and VIV.

Acknowledgement

I would like to acknowledge the contributions of my supervisors Professor Hamid Bahai and Dr. Jan wissink who never hesitated to lead me throughout this thesis in every aspect.

Dedication

This thesis is dedicated to my father, who taught me that the large tasks can be accomplished by diligence and persistent work. It is also dedicated to my mother, who taught me to work and help without expectation. Finally, this thesis is dedicated to my wife who passionately encouraged and fully supported me throughout completion of this work.

Table of contents

Abstract	ii
Some of the Contributions to knowledge	iii
Acknowledgement	iv
Dedication	v
Table of contents	vi
List of figures	xi
List of tables	xvii
List of Abbreviations	xviii
List of Symbols	xix
Chapter 1. Introduction	1
1.1 Fluid-Structure interaction (FSI).....	2
1.2 Vortex shedding and Strouhal number	3
1.3 Vortex induced vibration and Lock-in phenomena.....	4
1.4 Fundamental parameter	8
1.5 Flow regimes and vortex formation.....	8
• Non-separation regime; $0 < Re < 4$ to 5.....	9
• Laminar steady regime; $4 < Re < 47$	9
• Laminar shedding regime; $45 < Re < 180$	9
• Wake transition regime; $180 < Re < 350$ to 400	11
• Shear layer transition regime; $350 < Re < 2 \times 10^5$	11
• Critical regime; $2 \times 10^5 < Re < 1 \times 10^6$	12
• Boundary layer transition regime; $Re > 1 \times 10^6$	12
1.6 Aims and objective.....	12
1.7 Summary	14
Chapter 2. Background and preliminary study	15
2.1 Main technical difficulties of a FSI simulation	15
2.2 Two fundamental computational approaches.....	16
2.2.1 Partitioned approach	16
2.2.2 Monolithic approach.....	17
2.3 Discretisation approach	18
2.3.1 Body conforming mesh methods-moving grid method.....	18

2.3.2	Non-conforming mesh methods-fixed grid method	19
2.4	Some FSI applications	20
2.4.1	Engineering application.....	20
2.4.2	Biomechanics applications	22
2.5	Summary and layout of thesis	23
Chapter 3.	Literature review.....	25
3.1	Immersed boundary methods (IB).....	25
3.1.1	Original immersed boundary method- applicable for elastic IB	27
3.1.2	Feedback forcing approach- applicable for rigid IB	30
3.1.3	Physical Virtual Model (PVM) approach	30
3.1.4	Immersed interface approach.....	30
3.1.5	Fictitious domain method	31
3.1.6	Ghost-Cell approach.....	32
3.1.7	Cut-Cell method – Cartesian method	32
3.1.8	Direct forcing approach	32
3.1.9	Interpolation or reconstruction method.....	33
3.2	Defining the interface cells	35
3.2.1	Point-in-polyhedron algorithm	35
3.2.2	Interfacial marker at the interface discontinuity algorithm	36
3.3	Boundary Reconstruction/Interpolation.....	38
3.3.1	Stepwise geometry -No interpolation.....	38
3.3.2	Weighted method	39
3.3.3	Linear interpolation method.....	39
3.3.4	Bilinear interpolation method	42
3.3.5	Revised interpolation method	43
3.3.6	Quadratic interpolation method.....	44
3.3.7	Higher order interpolation methods.....	45
3.4	Interface tracking methods.....	45
3.4.1	Second-order accuracy without sub-iteration (loosely coupled, weak solution) .	47
3.4.2	Fixed point FSI coupling algorithm with dynamic relaxation.....	49
3.4.3	Reduced-order modelling (ROM) and interface location prediction.....	50
3.5	Moving frame of reference.....	50
3.5.1	Moving frame Formulation	51
3.5.2	Moving frame reference boundary conditions	53

3.6	Freshly cleared nodes	53
3.7	Mass conservation and pressure treatment near IB	54
3.7.1	Fictitious adding mass effect.....	56
3.8	Calculation of force on immersed boundary	56
3.8.1	Integrating continuous force	57
3.8.2	Direct calculation of surface forces.....	57
3.8.3	Application of momentum conservation	58
3.8.4	Direct forcing method.....	58
3.9	Some related Bench mark studies	59
3.10	Discussion.....	60
Chapter 4.	Methodology.....	62
4.1	Governing equation	64
4.2	Non-dimensional governing equation	65
4.3	Discretization method.....	66
4.3.1	Staggered arrangement	66
4.3.2	Discretization of the momentum equation	67
4.3.3	Fractional step method.....	69
4.3.4	Calculation of pressure	70
4.3.5	Mesh generation.....	72
4.3.6	Location of velocities and pressure	73
4.4	Boundary conditions	74
4.4.1	Inlet	74
4.4.2	Outlet	75
4.4.3	Symmetry boundary condition	76
4.4.4	Solid boundary not conforming mesh (immersed boundary)	76
4.5	Solving procedure	79
4.6	Moving frame of reference.....	81
4.7	Evaluating forces and moment on an immersed boundary	82
4.8	Direct calculation of pressure over an IB.....	84
4.8.1	Calculation of pressure force without extrapolation.....	84
4.8.2	Extrapolating the pressure.....	85
4.8.3	Calculation of the shear forces around a cylinder	85
4.8.4	Application of momentum conservation to calculate force on IB.....	87
4.9	Lift and drag coefficient	89

4.10	Summary	90
Chapter 5.	Parametric study and validation	92
5.1	Parametric study	92
5.1.1	Parametric study - Mesh refinement effect.....	94
5.1.2	Parametric study – size of domain in front of cylinder	101
5.1.3	Parametric study – Blockage effect	104
5.1.4	Parametric study – Stretching factor	108
5.1.5	Parametric study – size of uniform area, x direction after cylinder	110
5.1.6	Parametric study- uniform size x direction before cylinder	113
5.1.7	Parametric study- uniform grid area in y direction.....	114
5.2	Validation	116
5.3	Summary	119
Chapter 6.	Comparative study of the interpolation methods - Stationary cylinder.....	121
6.1	Governing equation and computational domain.....	121
6.2	Interpolation method cases	122
6.2.1	Case A: No interpolation	123
6.2.2	Case B: Weighting method.....	123
6.2.3	Case C: linear interpolation method	124
6.2.4	Case D: Bilinear interpolation method.....	124
6.2.5	Case E: Proposed interpolation method	125
6.3	Results and discussion	125
6.4	Conclusion.....	131
Chapter 7.	Body cross flow oscillation	132
7.1	Forced Oscillation of a body in cross flow direction	132
7.2	Fluid-Structure interaction due to Vortex induced Vibration.....	133
7.3	First approach-moving frame of reference.....	134
7.3.1	Moving frame-governing equation	135
7.3.2	Moving frame-velocity boundary conditions.....	136
7.3.3	Moving frame-Neumann boundary for pressure Poisson equation.....	136
7.3.4	Moving frame of reference algorithm	137
7.4	Second approach, moving IB or fixed grid (inertial frame of reference)	138
7.4.1	Inertial frame-governing equation and boundary conditions	139
7.4.2	Inertial frame of reference algorithm.....	139
7.5	Calculation of the force on moving boundary	140

7.6	Parametric study	140
7.6.1	Parametric study- mesh size	141
7.6.2	Parametric study-size of domain before the cylinder.....	143
7.6.3	Parametric study- blockage effect	144
7.7	Results.....	145
7.7.1	Inertial effect - Froude-Krylov force.....	146
7.7.2	Moving frame verses inertial frame of references	148
7.7.3	Cross flow oscillation of circular cylinder – validation.....	150
7.7.4	Vortex induced vibration in cross flow direction	155
7.8	Summary	157
Chapter 8.	Conclusion and Future work	158
8.1	Simulation approaches.....	159
8.2	Validation of the results and feasibility of the method	161
8.3	Drawbacks verses advantages of the IB interpolation.....	163
8.4	Future work.....	164
	References	165
	Appendix A, Fortran Code, Construction method.....	174
	Appendix B, Conference paper 2011.....	235
	The IV International Conference on Computational Methods for Coupled Problems in Science and Engineering held in Kos, Greece 20-22 June 2011	
	Appendix C, Conference paper 2013.....	243
	The V International Conference on Computational Methods for Coupled Problems in Science and Engineering held in Ibiza, Spain 17-19 June 2013	

List of figures

Figure 1-1: Vortex shedding and pressure contour behind a cylinder at low Reynolds number. Dash lines and continuous lines are negative and positive pressure contours, respectively.....	3
Figure 1-2: experimental results of the cross flow response of flexibility mounted cylinder subject to a steady air stream. Originally presented by Feng 1968 and graphs reproduced by Sumer and Fredsøe 2006.	5
Figure 1-3: Types of vortex-Shedding as a function of oscillation amplitude in transverse direction, A/D , and oscillation wave length, λ/D , (Williamson & Roshko 1998). The critical curve marks the transition from the 2S to the 2P mode of vortex shedding. The dashed curves marked I and II, from Bishop & Hassan 1964, Indicate where the fluid forces acting on the cylinder underwent a sudden jump, for I decreasing λ/D and II increasing λ/D	6
Figure 1-4: Lock -in region as a function of amplitude, A/D , and frequency, f/f_s , of oscillation for the forced transverse vibrations of a circular cylinder. \square , \circ lock in vortex shedding border and Δ , $+$, unlocked vortex shedding area (Meneghini & Bearman 1995).	7
Figure 1-5 Flow regime at various Reynolds number (Sumer & Fredsoe 2006).....	10
Figure 2-1: a) Schematic Monolithic approach, b) Schematic Partitioned approach.....	16
Figure 2-2: Left, Schematic of body in a fluid flow with body conforming mesh. Right, Schematic of body in a fluid flow with body non-conforming mesh method. Ω_s is the solid domain, Ω_f is the fluid domain and Γ_s is the solid boundary	17
Figure 2-3: a few example of Fluid-Structure interaction (FSI) in different application....	21
Figure 3-1a) Transferring the boundary force F_k from each material point (Lagrangian coordinate) \mathbf{X}_s, \mathbf{t} to the fluid. Shaded area shows the area which force effect will be distributed in the fluid domain; b) various forcing function distribution (Mittal & Iaccarino 2005).....	28
Figure 3-2: A 2D Cartesian mesh with a solid boundary (circle). Interface points, that require interpolation, are identified by arrows. Points A1 to A8 are all neighbouring points of A. Note that A2 and A7 are inside the solid domain.....	34
Figure 3-3: a) background Cartesian grid with a polygon immersed boundary classified to fluid, solid and interface nodes, b) a ray casting test method for a polygon; point A is inside and point B is outside the polygon, c) a ray casting test method for a polyhedron (Borazjani et al 2008).	36
Figure 3-4 a) Definition of the immersed boundary topology by interfacial markers, arc length vector and normal vector; b) Identification of fluid nodes from solid nodes using a normal vector; c) Demonstration of interface nodes (o) and marker points (●), (Balaras 2004).	37
Figure 3-5 interpolation procedure sketch for u velocity in an staggered arrangement a) without interpolation b) weighted interpolation c) linear interpolation	39

Figure 3-6: calculating the interface cell velocity by linear interpolation; a) ambiguity in the direction of interpolation, Fadlun et al. 2000 model b) linear interpolation perpendicular to the IB, Balaras 2004 model.	40
Figure 3-7: interpolation scheme in direction perpendicular to the IB, Balaras 2004; three boundary options depends on the immersed boundary geometry and local grid size.	41
Figure 3-8: Schematic reconstruction of the IB unknown “b” with interpolation in the wall-normal direction. The triangle represents an unstructured element of the IB. The dash line is the intersection of the body with the Cartesian grid (Gilmanov et al. 2003).	42
Figure 3-9: Standard Reconstruction Method (SRM) for velocity in vertical (left) and horizontal (right) direction	42
Figure 3-10: Quadratic interpolation method for an interface velocity in the horizontal (left) and the vertical (right) directions. The middle pane shows that it is not always possible to use this type of quadratic interpolation.	44
Figure 3-11 : Coordinate transformation.....	52
Figure 3-12: Various methods for conservation of mass depending on the IB method a) mostly for the continuous forcing approach (standard approach) b) mostly for the cut cell approach with a reshaped control volume c) mostly for the reconstruction method, conservation of mass only in fluid domain (Kang et al. 2009).	55
Figure 4-1: Control volumes for a staggered grid: for mass conservation and scalar quantities (left), for x-momentum (centre) and for y-momentum (right).....	67
Figure 4-2: staggered arrangement used for discretization	68
Figure 4-3 : Unifrom staggered mesh coordinate	73
Figure 4-4: velocities and pressure positions in a staggered arrangement.....	74
Figure 4-5 Staggered arrangement – bold lines are cell boundaries which velocities are calculated, and pressure are calculated on intersection of light lines. Velocities in y direction need to be interpolated for inlet. Velocities in x direction are specified directly on the boundaries.....	75
Figure 4-6 Left, a part of domain with not conforming Cartesian mesh, regardless of solid existence. Right, A specific velocity with its 8 velocities around necessary for its calculation.....	77
Figure 4-7: A 2D Cartesian grid with staggered arrangement, left: u velocities needed to be interpolated near immersed boundary. Right: v velocities needed to be interpolated near the immersed boundary.	77
Figure 4-8: interpolation method for the velocity near the boundary in two different scenarios. $u_{i,j}$ has been interpolated between $u_1=0$ on the boundary and u_2	78
Figure 4-9: Right, shaded area shows the cells in which the pressure is updated in the CFD solver. Left, cells with at least one immersed boundary pressure points is shown.	79
Figure 4-10: flowchart of the flow solver used to apply interpolation method	80
Figure 4-11: calculation of lift and drag component of force due to pressure (left) and shear force (right)	83

Figure 4-12: left, pressure near the immersed boundary directly used as pressure on the boundary. Right, linear extrapolation method to calculate pressure on the immersed boundary.....	85
Figure 4-13: Calculating tangential velocity around the immersed boundary	86
Figure 4-14: Location of Immersed boundary (IB), control volume (C.V.), Control Surface (C.S.) to apply Conservation of momentum law	87
Figure 4-15: Surface normal vector n , velocity (u,v) and pressure on the control surfaces	88
Figure 5-1: flow pattern around a stationary cylinder at $Re=100$. High pressure area (Continuous line), low pressure area (dash line), blue and red counters are the vortices.	93
Figure 5-2: Background Cartesian mesh- parametric studies guide.	94
Figure 5-3: simulation accuracy of the immersed boundary based on the mesh size	95
Figure 5-4: Mesh refinement study, drag, drag due to pressure and shear stress for five different grid sizes from $dx=dy=0.1$ to 0.00625 around the circular cylinder	97
Figure 5-5: Drag coefficient due to pressure and shear stress verses the number of grid in each direction of the domain around a stationary cylinder at low Reynolds number, $Re=100$	98
Figure 5-6: Mesh refinement study for lift, lift due to pressure and friction for various grid size where computational domain in x and y is $[-15,15]$ and Stretching factor is 3.	98
Figure 5-7: Lift coefficient verses the number of grid points in each direction of the domain around a stationary cylinder at Low Reynolds number, $Re=100$	99
Figure 5-8: The Power Spectral density of lift coefficient for six different grids size in frequency domain, where computational domain in x and y is $[-15, 15]$ and Stretching factor is 3.	100
Figure 5-9: Strouhal number verses the number of grid point in each direction of the domain around a Stationary cylinder at low Reynolds number, $Re=100$	100
Figure 5-10: Effect of the Size of the fluid domain in front of the circular cylinder in x direction on the Drag coefficient.....	101
Figure 5-11: Effect of the Size of the fluid domain in front of the circular cylinder in x direction on the lift coefficient	102
Figure 5-12: The power spectral density of the Lift coefficient- Effect of the Size of the fluid domain in front of the cylinder in x direction on the lift coefficient.	103
Figure 5-13: Effect of the size of the computational domain in the y direction on the drag	104
Figure 5-14: Effect of the Size of the fluid domain in y direction on the Lift coefficient.	105
Figure 5-15: Power Spectral Density (PSD) of the Lift coefficient - Effect of Size of the domain in y direction on the lift coefficient.	106
Figure 5-16: Drag coefficient verse domain size in cross flow direction,	107
Figure 5-17: Lift coefficient verse the size of domain in perpendicular direction to the main stream velocity (cross flow direction).	107

Figure 5-18: Strouhal number verse the size of domain in perpendicular direction to the main stream velocity (cross flow direction).	108
Figure 5-19: Effect of the grid stretching factor on the Drag coefficient	108
Figure 5-20: Effect of the grid stretching factor on the Lift coefficient.	109
Figure 5-21: Effect of the grid stretching factor on Strouhal number.....	110
Figure 5-22: Effect of the uniform area after the circular cylinder in x direction on the Drag coefficient.	111
Figure 5-23: Effect of the uniform grid area after the circular cylinder in x direction on the lift coefficient.	112
Figure 5-24: Power Spectral density (PSD) of the Lift coefficient - Effect of the uniform Size of the fluid domain after the circular cylinder in x direction on the lift coefficient.	113
Figure 5-25: Effect of the uniform grid in front of the circular cylinder in the x direction on the drag coefficient.	114
Figure 5-26: Effect of the uniform grid in front of the circular cylinder in x direction on the Lift coefficient.	114
Figure 5-27: Effect of the vertical extend of the uniform area around the circular cylinder on the Drag coefficient.	115
Figure 5-28: Effect of the vertical extend of the uniform area around the circular cylinder on the Lift coefficient.	115
Figure 5-29: Schematic of the computational domain	116
Figure 5-30: Drag coefficient, Drag due to pressure and friction for a stationary cylinder at Re=100 versus non dimensional time.	117
Figure 5-31: Lift coefficient, the lift due to pressure and friction for a stationary cylinder at Re=100 verses non dimensional time.....	118
Figure 6-1: Fluid domain size and boundary conditions.....	122
Figure 6-2: Bilinear proposed interpolation in this study for the cells near the solid boundary in vertical (Left) and horizontal (right) velocity components.	124
Figure 6-3: Drag coefficient for the flow around a stationary cylinder at Re=100, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation; Case E, Suggested bilinear interpolation.....	127
Figure 6-4: Drag coefficient due to pressure (left) and due to shear stress (right) for the flow around a stationary cylinder at Re=100, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation1; Case E, proposed bilinear interpolation method	128
Figure 6-5: Lift coefficient for the flow around a stationary cylinder at Re=100, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation method; Case E, suggested bilinear interpolation method.....	128
Figure 6-6: Lift coefficient due to pressure (left) and shear stress (right) for the flow around a stationary cylinder at Re=100. Case A, without interpolation method; Case B:	

area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation method; Case E, suggested Bilinear interpolation	129
Figure 6-7: Power Spectral density of the lift coefficient; five different interpolation methods.....	130
Figure 7-1: Flow over a circular cylinder at two dimensions with vertical degree of freedom.....	133
Figure 7-2: mesh refinement study- Drag coefficient	142
Figure 7-3: mesh refinement study – Lift coefficient	143
Figure 7-4: Parametric study of the effect of size of domain before cylinder in x direction on the mean drag and maximum lift; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$	143
Figure 7-5: Parametric study of the effect of size of domain in y direction on drag coefficient; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$	144
Figure 7-6: Parametric study of the effect of the size of the domain in the y direction on the lift coefficient; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$	145
Figure 7-7: Parametric study of the effect of size of domain in y direction on the mean drag and maximum lift; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$	145
Figure 7-8: Using Froude-Krylov force (inertial force) to correct lift coefficient calculated in moving frame of reference	147
Figure 7-9: The drag (CD) due to pressure and shear stress, lift due to shear stress and pressure for cases A and B in the moving frame of reference.....	147
Figure 7-10: lift (lower curve) and drag (upper curve) due to pressure; dotted lines, inertia frame of reference(without smoothing); dash lines, moving frame of reference	148
Figure 7-11: lift (lower curve) and drag (upper curve) due to pressure; dotted lines, inertia frame of reference (with smoothing); dashed lines, moving frame of reference	149
Figure 7-12: lift (lower curve) and drag (upper curve) due to shear stress; dotted lines, inertia frame of reference; dash lines, moving frame of reference.....	149
Figure 7-13: Force coefficient and phase angle verses f_e/f_s . Left-Mean drag coefficient (CD), rms of drag and lift fluctuation coefficients (CD _{rms} and CL _{rms} respectively); Right-Phase angle between CL and the vertical position of the cylinder. -■-, present study; -▲-, Kim & Choi 2006.....	150
Figure 7-14: Drag (CD), Lift (CL) coefficient and y_c/D time history for $A/D=0.2$ and $Re=185$, (a) $f_e/f_s=0.8$, (b) $f_e/f_s=0.9$, (c) $f_e/f_s=1$, (d) $f_e/f_s=1.1$, (e) $f_e/f_s=1.12$, (f) $f_e/f_s=1.2$. CD: dash dot curve; CL: Continuous curve; y_c/D : dot curve.....	152
Figure 7-15: Drag (CD), Lift (CL) coefficient and y_c/D time history for $f_e/f_s=0.75$ and $Re=200$, (a) $A/D=0.25$, (b) $A/D=0.30$, (c) $A/D=0.45$, (d) $A/D=0.6$	153
Figure 7-16: Drag (CD), Lift (CL) and y_c/D over time for $f_e/f_s=0.90$, $A/D=0.15$, $Re=200$. Left figure present study, right figure shows results of Meneghini and Bearman 1995.	154

Figure 7-17: Drag (CD), Lift (CL) and y_c/D over time for $f_e/f_s=0.80$, $A/D=0.25$, $Re=200$.
 Left figure present study, right figure shows results of Meneghini and Bearman 1995.
 154

Figure 7-18: Drag (CD), Lift (CL) and y_c/D over time for $f_e/f_s=1.025$, $A/D=0.05$, $Re=200$;
 left figure) present study; Right figure) Meneghini and Bearman 1995. 155

List of tables

Table 5-1: Results of mesh refinement study around a stationary cylinder at $Re=100$	96
Table 5-2: parametric study of the Stretching factor, minimum grid size is $0.025D$, the domain size $[-15,15]$ in x and y direction, and the uniform domain size $[-2,4]$ in x and $[-2,2]$ in y direction.....	110
Table 5-3: Drag, lift and Strouhal number for present study and well known numerical and experimental studies for the flow around a circular cylinder at low $Re=100$	119
Table 6-1: Real computational time, 20 vortex shedding	126
Table 6-2: Strouhal number, lift and drag coefficient for the flow around a stationary cylinder and $Re=100$	130
Table 7-1: mesh refinement study of oscillating cylinder – Parameters and results.....	141
Table 7-2 : Amplitude of oscillation (y_{max}/D) at various reduced velocity at constant Reynolds number, $Re=150$, and low mass ratio, $m^*=2$	156
Table 7-3: Amplitude of oscillation (y_{max}/D) at various Reynolds number and reduced velocity at high mass ratio, $m^*=149.10$	156

List of Abbreviations

Abbreviation	Stands for
ALE	Arbitrary Lagrangian Eulerian
CFD	Computational Fluid Dynamic
CURVIB	Curvilinear Immersed Boundary
FD	Finite Difference
FE, FEA or FEM	Finite Element, Finite Element analysis or Finite Element Method
FV	Finite Volume
FSI	Fluid-Structure Interaction
HCIB	Hybrid Cartesian Immersed Boundary
IB	Immersed Boundary
IIM	Immersed Interface Method
LES	Large Eddy Simulation
NS	Navier Stokes
PVM	Physical Virtual Method
RLIM	Revised Linear Interpolation method
ROM	Reduced Order Method
SRM	Standard Reconstruction Method
Ured	Reduced Velocity
VIV	Vortex Induced Vibration

List of Symbols

Abbreviation	Stands for	units
A	Amplitude of oscillation	m
CD	Drag Coefficient	...
CL	Lift coefficient	...
CLrms	Root mean square of CL	...
CP	Pressure coefficient	...
D	Cylinder diameter	m
f	Frequency	Hz
f	Forcing function	N
fn	Natural frequency of the solid	Hz
fo	Oscillation frequency	Hz
fs	Strouhal frequency	Hz
fv	Vortex shedding frequency	Hz
FD	Drag force	N
FL	Lift force	N
Mred	Reduced mass of the system	...
Re	Reynolds number	...
St	Strouhal number	...
U_{∞}	Free stream velocity	m/s
t, t'	Time	s
Vr or Ur	Reduced velocity	...
X	Position (material point)	m
x	Position (Eulerian)	m
ρ	Density	Kg/m ³
τ	Shear stress	N/m ²
ν	Static viscosity	m ² /s
η	Dynamic viscosity	Pa.s
ξ	Damping ratio	...
ϕ	Phase angle	...

Chapter 1. Introduction

Fluid-Structure Interaction (FSI) analysis is widely used to study the physical phenomenon which occurs in many engineering applications in which a fluid flow interacts with a deformable or moving structure.

Many engineering structures are subjected to environmental fluid currents that can induce significant unsteady forces. For instance, wind can cause substantial forces to be imparted on chimney towers, cables, bridge decks and other common structures. Ocean currents can similarly affect offshore platforms, submerged pipelines and riser pipes¹. Without an accurate FSI analysis there will be a great uncertainty about the safety of these structures.

Moreover, FSI analysis has had significant impact in biomechanics research, making it possible to model the interaction between biological tissues (e.g. arteries) and biological fluid flow (e.g. blood).

Modelling many of these FSI problems with large displacement/deformation and complex geometry is quite challenging and despite extensive developments in the recent decades, there is still a demand for further work in this area of research. Also, FSI is a multi-physics and multi-disciplinary phenomenon therefore an accurate FSI modelling involves many detailed challenging procedures. The computation algorithms should be selected based on the physics of the problem and the availability of computational resources. In this study the main focus is on simulation of behaviour of offshore flexible risers and pipe lines. When a riser oscillates in the flow or is exposed to an oscillatory flow, the vortex shedding regime around the pipe can be changed dramatically. In a certain range of oscillation and amplitude, the oscillatory stream and vortex shedding can affect the structure's stability which could lead to structural failure. Sophisticated structural studies has been carried out for risers subjected to prescribed excitation forces, however the motivation here is to couple the structural analysis with a more realistic excitation by studying FSI in a real riser condition.

The ultimate aim of this research is to develop and test an in-house code which then be combined with an existing structural analysis program to simulate Fluid-Structure-

¹ - The risers pipes used in the offshore industry to convey fluids (oil and gas) from the seabed to the sea level and vice versa.

Interaction (FSI) of an offshore riser in realistic circumstances. In addition, the models and algorithm are employed in the code in such a way that the simulations can be run in a reasonable time using existing computational facilities. The final goal is to simulate vortex induced vibration of a riser in three dimensions, at high Reynolds number and with large displacements.

The first stage of this comprehensive program of work has been completed and is presented in this thesis. In this stage an in-house code is developed to simulate two dimensional flows around a flexible/deformable circular cylinder². To achieve this goal, first FSI methods are studied in detail; The FSI algorithm elements are selected in a way to facilitate further developments of the code to three dimensions, high Reynolds numbers and large displacements/deformations in future.

This thesis is divided in three parts. In the first part which includes chapters 1, 2 and 3 the motivation, introduction, background and literature review of the problem is presented and some examples of various FSI applications are provided. In the second part of the thesis which include chapters 4, 5 and 6 the FSI methodology based on the immersed boundary (IB) method with an interpolation/ reconstruction procedure is discussed and the proposed algorithm is presented. The results of a parametric study for a stationary case are validated with bench marks as well as the results from other IB interpolation methods in the literature. In the final part of the thesis, chapter 7, the simulation of the flow around a flexible cylinder is presented. The simulation results with a moving and an inertial reference frame are compared with one another and with some results from the literature. Also, case studies of Vortex Induced Vibration analysis of the proposed model are presented and validated against results from the literature.

In this introductory chapter, some fundamental topics and general parameters used in a full FSI analysis are briefly explained. At the beginning, the effect of the Reynolds number as a key parameter is discussed. This is followed by introducing some important terminology in the field. And finally a summary of the chapter is presented.

1.1 Fluid-Structure interaction (FSI)

Fluid-Structure Interaction (FSI) is a multiphase problem which involves Computational Fluid Dynamics (CFD) and Computational Structural analysis. The flow

2- The circular cross section is use for the whole study as in the offshore industry the riser's cross section are all circular due to the fact that not only these cross section has minimum stress concentration but also circular cross section can be manufactured and used easier in this application.

simulation is considered to be an FSI problem inside or around a deformable or moving boundary/structure when flow forces cause the structure to deform which, in turn, changes the boundary conditions of the fluid flow.

Numerical simulations of the fluid field interacting with moving boundaries are among the most challenging problems in computational mechanics. The reason for this is that the fluid domain changes with time and the location of boundaries depend on fluid flow forces inducing deformation/motion of the boundaries (Yang et al. 2008).

1.2 Vortex shedding and Strouhal number

When the fluid flow passes around an object, such as a circular cylinder, a boundary layer will form around the cylinder. Due to an adverse pressure gradient along the downstream half of the cylinder, the boundary layer separates at a specific angle behind the cylinder depending on the flow parameter. The separated boundary layer rolls up into vortices in the low pressure area behind the cylinder. After a period of growth these vortices are shed and washed downstream by the flow. These vortices create alternating pressure on either side of the cylinder and the body tends to move toward the low pressure zone. Therefore, the vortex shedding is the oscillating flow pattern that occurs periodically when a fluid (e.g. air or water) passes a bluff body at specific velocities, U , depending on the Reynolds number, size and shape of the body. The normalised vortex shedding frequency for a stationary body is known as the Strouhal number, $St (=f_v D/U)$; in which, the parameter D is the diameter of cylinder and f_v is vortex shedding frequency. The Von Karman vortex street is a famous vortex shedding patterns that forms behind a stationary cylinder. In Figure 1-1, the formation of the vortices is shown in the low pressure area (dash line contours) behind the cylinder.

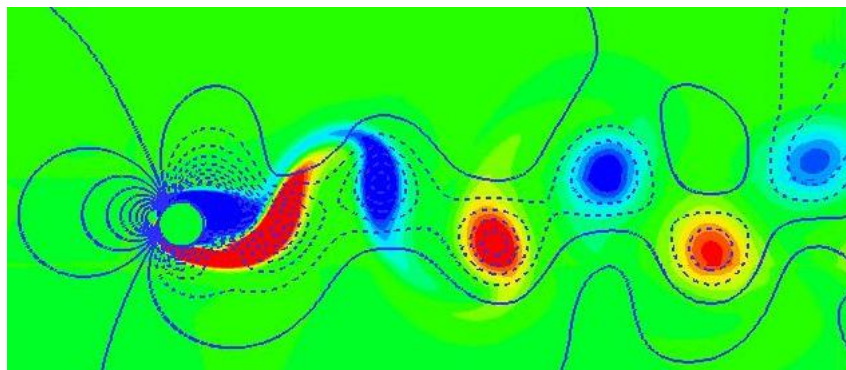


Figure 1-1: Vortex shedding and pressure contour behind a cylinder at low Reynolds number. Dash lines and continuous lines are negative and positive pressure contours, respectively.

1.3 Vortex induced vibration and Lock-in phenomena

The vortex shedding process and the shed vortices themselves induce periodic forces on the body. If the body is compliant or elastically supported then these forces can cause the body to vibrate. Such a vibration is called a Vortex Induced Vibration (VIV). The amplitude of vibration depends on many factors including the level of structural damping, the relative mass of the body to the fluid, the magnitude of the fluid forces and the proximity of the vortex shedding frequency to the natural frequencies of vibration of the body.

The fluid forces in both, the cross-stream (transverse) direction (lift), and the stream-wise (in-line) direction (drag) can induce VIV in their respective directions. The oscillatory component of the drag forces is normally far smaller than the oscillatory component of the lift force. Consequently in-line VIV is normally of lower amplitude than transverse VIV. The frequency of the in-line oscillatory force and the consequent vibration is normally twice that of the transverse oscillatory force and resulting motion.

Lock-in phenomena are defined where a body is vibrating in a fluid flow and the oscillation frequency and vortex shedding frequency become synchronized. According to the numerical and experimental results, lock-in only happens in body oscillation with amplitude above a specific threshold. And the range of oscillation frequencies (or reduced velocities) at which lock-in occurs will increase by increasing the oscillation amplitude.

The experiment of Feng 1968 addresses the VIV and Lock-in phenomena and related parameters. In this experiment a flexibility mounted cylinder with a transverse degree of freedom was exposed to various air velocity streams. For a flow velocity of U , the vortex shedding frequency, f_v , the vibration frequency, f_o ; the vibration amplitude, A , and the phase angle, ϕ , is measured. The phase angle is defined as the phase difference between the vortex shedding frequency and vibration frequency of the cylinder. The results are presented based on a normalised velocity known as reduced velocity V_r ($=U/D f_n$). In this formula, f_n is the natural frequency of the system. Figure 1-2 shows that there is no vibration at a reduced velocity lower than 4. For $4 < V_r < 5$ small vibrations occur at the natural frequencies of the system ($f/f_n=1$), while the vortex shedding frequency equals the cylinder Strouhal frequency. However according to Figure 1-2a, for $5 < V_r < 7$ the vortices will start to shed at the natural frequency of the system (i.e line $f/f_n=1$) (instead of the Strouhal frequency). In other words the vortex shedding frequency

locks in to the natural frequency of the system for a range of reduced velocities from $V_r=5$ to $V_r=7$.

In the lock-in range the system natural frequency, f_n , the vortex shedding frequency, f_v , and the vibration frequency, f , remain synchronised ; i.e. $f_n=f_v=f$. Therefore at this range of reduced velocities, the lift force contributes to the system's natural vibration which could lead to vibration with high amplitude (Figure 1-2).

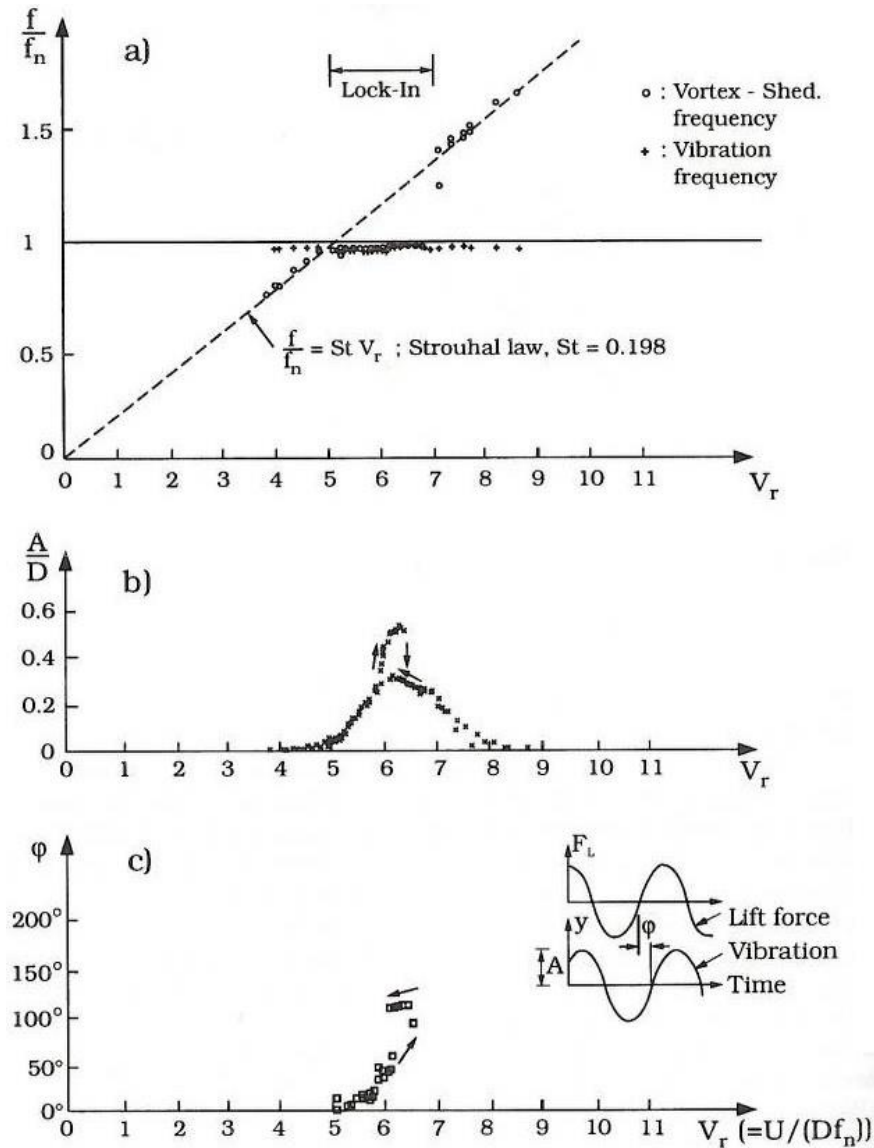


Figure 1-2: experimental results of the cross flow response of flexibility mounted cylinder subject to a steady air stream. Originally presented by Feng 1968 and graphs reproduced by Sumer and Fredsøe 2006.

For the higher reduced velocity, $V_r > 7$, the vortex shedding frequency unlocks from the natural frequency and jumps to the Strouhal natural frequency (Figure 1-2a). The range of lock-in period depends on the amplitude of vibration which itself depends on

structural damping. The lock-in range is larger for the higher amplitude of vibration (lower structural damping) as it may need higher V_r to unlock the shedding frequency from the natural frequency of the system (for more details see Sumer and Fredsøe 2006).

Moreover, according to Figure 1-2a, at higher reduced velocities (for instance $V_r=7.3$), while the system is vibrating at its natural frequency ($f/f_n=1$), the vortex shedding occurs at the Strouhal frequency. Since the forcing frequency (vortex-shedding frequency) is no longer in phase with the vibration of the cylinder, there is a reduction in amplitude of vibration. Also at higher reduced velocities, the vortex shedding frequency moves further away from the natural frequency of the system, which could lead to a greater reduction in the vibration amplitude. The experiment shows that at $V_r>8.5$ the vibration of the system completely disappears. Figure 1-2b and c, also show a hysteresis effect in amplitude and phase shift variation with respect to reduced velocity. i.e. the amplitude and phase will be slightly different at the same reduced velocity depending on the direction of the experiment; increasing the reduced velocity from low to high values in the course of experiment or vice-versa.

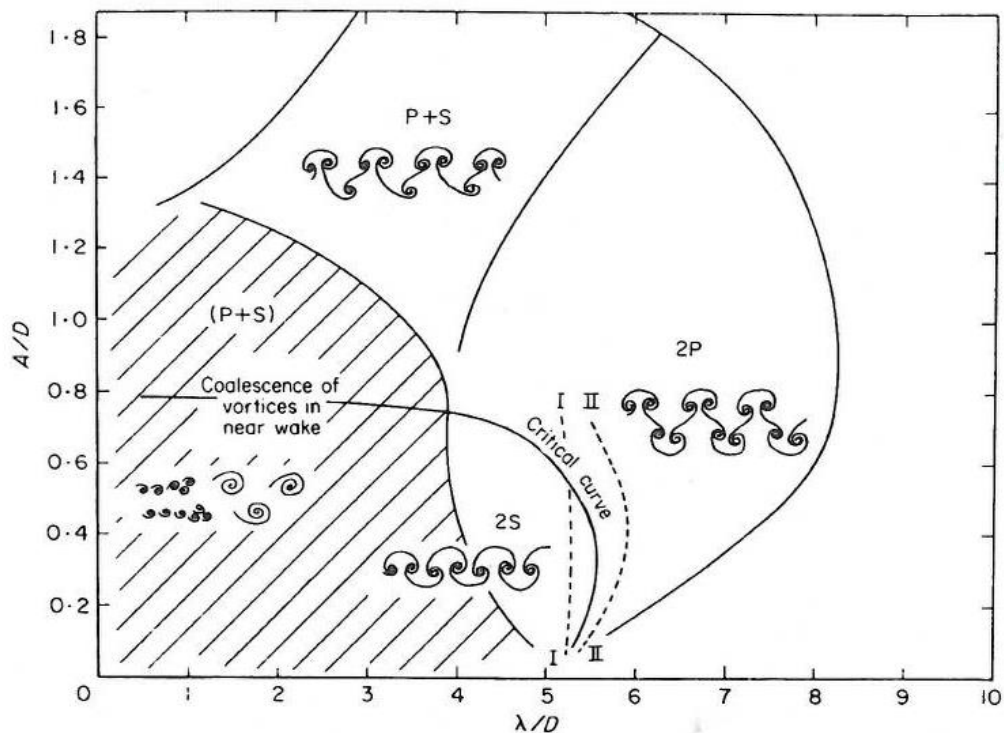


Figure 1-3: Types of vortex-Shedding as a function of oscillation amplitude in transverse direction, A/D , and oscillation wave length, λ/D , (Williamson & Roshko 1998). The critical curve marks the transition from the 2S to the 2P mode of vortex shedding. The dashed curves marked I and II, from Bishop & Hassan 1964, indicate where the fluid forces acting on the cylinder underwent a sudden jump, for I decreasing λ/D and II increasing λ/D .

In addition, the flow pattern of vortex shedding changes dramatically at different oscillation amplitudes (Li et al. 2002). Figure 1-3 shows the vortex pattern changes at different amplitudes of vibration, A/D , and different oscillation wave-lengths, λ/D ($=V_r$, reduced velocity). These experiments were conducted for $300 < Re < 1000$. Williamson & Roshko 1988 interpreted that the changes in the shedding mode that occurs across the critical curve is the reason of the jump in the phase angle and the lift coefficient that has been reported by Bishop & Hassan 1964 among others.

Meneghini and Bearman 1995 presented the lock in region for the range of amplitudes, A/D , varying from 0.025 to 0.6 and the range of oscillation frequencies, f/f_s , from 0.7 to 1.15 (Figure 1-4). This lock-in range is around the Strouhal vortex shedding frequency and two vortices of opposite circulation are shed per cycle (S type). When the frequency of the lift force was similar to the frequency of the oscillating flow lock-in was occurred.

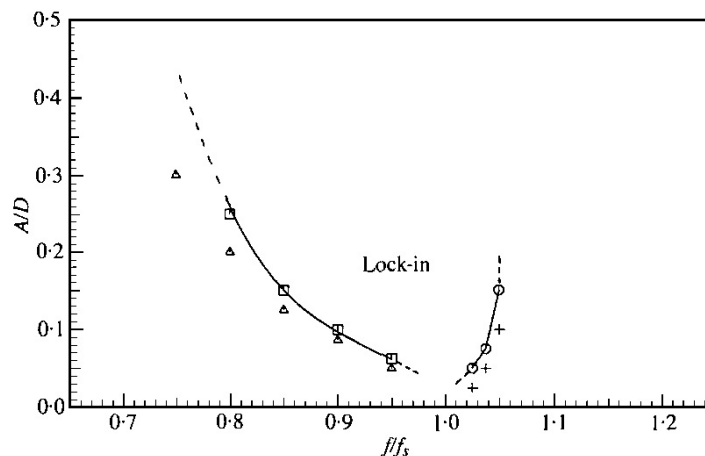


Figure 1-4: Lock -in region as a function of amplitude, A/D , and frequency, f/f_s , of oscillation for the forced transverse vibrations of a circular cylinder. □, ○ lock in vortex shedding border and △, +, unlocked vortex shedding area (Meneghini & Bearman 1995).

Vortex Induced Vibrations can have serious consequences as they provide a major source of fatigue and can cause bodies clashing in multiple body assemblies. The potential implications of VIV make predicting of its occurrence and its likely amplitude and frequency of response imperative when designing engineering structures that are exposed to flow. Recently, VIV has received a great deal of attention and various methods for its predictions have been developed rapidly.

1.4 Fundamental parameter

The two most important factors that determine the dynamics of the flow past a stationary bluff body³ are the body's cross-sectional shape and the Reynolds number.

The Reynolds number is a dimensionless number that gives a measure of the ratio of inertial forces to viscous forces and consequently, quantifies the relative importance of these two types of forces for a given flow condition. A laminar flow occurs at low Reynolds numbers, where the viscous forces are dominant, and is characterized by smooth, constant fluid motion. A turbulent flow, on the other hand, occurs at high Reynolds numbers and is dominated by inertial forces, which tend to produce random eddies, vortices and other flow fluctuations. A Reynolds number is meaningless without the selection of proper characteristic length and velocity scales. For the flow problem over a circular cylinder, the diameter of the cylinder is selected as the characteristic length scales which the free-stream velocity is chosen as the characteristic velocity scale. Practically, matching Reynolds numbers do not guarantee a similar flow, as very small changes in the parameters such as shape, roughness could result in very different flow regimes.

If the Reynolds number is large enough then the regions of recirculating flow can become detached from the body (separation). The re-circulating flow in these detached regions, which is referred to as an eddy or vortex, generally comprises of low speed (relative to the free stream flow speed) vortices. Once shed, or detached, the vortices are convected downstream of the body by the main flow. If the flow past the bluff body is fully developed then a wake instability mechanism causes vortices to be shed in a periodic fashion from alternating sides of the body. The body's resulting wake structure comprises a staggered array of vortices that trails downstream of the body. Such a wake is referred to as a von Karman vortex street.

1.5 Flow regimes and vortex formation

The dynamics of the flow past a stationary circular cylinder are dependent on many factors of which the Reynolds number is the most important. The effect of increasing the Reynolds number is firstly to initiate flow separation, then vortex shedding and at higher Reynolds numbers a gradual transition to turbulence, which starts in the far wake and moves upstream and eventually into the attached boundary layers with increasing

3- The engineering bodies those are non-streamlined, such as those that have square or circular cross-sections in the plane of the fluid flow are referred to as bluff bodies.

Reynolds number. Roshko 1954 was amongst the first to categorise the flows observed at different Reynolds numbers into various flow regimes

The knowledge of the flow regimes that exist for the flow past a stationary circular cylinder, and the Reynolds numbers at which these regimes begin and end, has undergone a continuous development since Roshko's initial categorisation of the regimes. This section discuss the current knowledge of these flow regimes in terms of the topology of the cylinder's wake, the state of the flow; laminar or turbulent and where appropriate the transition point, and certain key global parameters. The first of these key parameters is the Strouhal number, $St = f_s D / U_\infty$, which is a non-dimensional measure of the vortex shedding frequency. Roshko found that the Strouhal number behaves differently in each of the three regimes he identified. In the stable regime it rises rapidly, in the irregular regime it is approximately constant and in the transition regime he found it to be unstable, i.e. erratic.

- **Non-separation regime; $0 < Re < 4$ to 5**

At very low Reynolds numbers, $0 < Re < 4$ to 5 , the flow is laminar and is dominated by viscous effects. It remains fully attached to the cylinder, as sketched using streamlines in Figure 1-5, this regime is often referred to as creeping or Stokes flow.

- **Laminar steady regime; 4 to $5 < Re < 47$**

As the Reynolds number is increased beyond $Re \approx 5$ the boundary layers separate symmetrically from both sides of the cylinder. The separated shear layers roll up and form a pair of standing vortices (recirculation cells) in the cylinder's near wake, (see Figure 1-5) and the wake behind the cylinder is steady and symmetric about the wake centreline. Hence in this regime there are no fluctuating forces exerted on the cylinder and the Strouhal number is zero ($St=0$). As the Reynolds number increased through this regime the separation points, which are located towards the rear of the cylinder at $Re \approx 5$, move forward and the standing vortices grow in size. The length of the recirculation cells increases approximately linearly with Reynolds number.

- **Laminar shedding regime; $45 < Re < 180$**

At a Reynolds number of approximately 47, disturbances in the flow become amplified, resulting in cross-stream oscillation of the downstream end of the recirculation cells and a sinusoidal oscillation of the wake-trail downstream of the recirculation cells. As the Reynolds number is increased the recirculation cells shorten

and the amplitude of the oscillation increases until it is large enough to cause the vortices to roll up at its troughs and crests, resulting in a staggered array of laminar vortices.

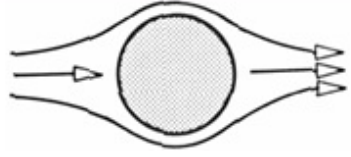
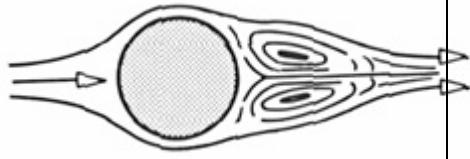
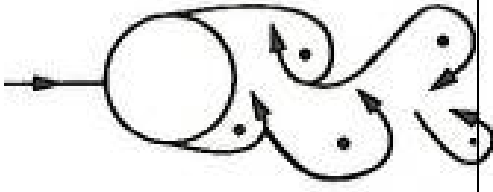

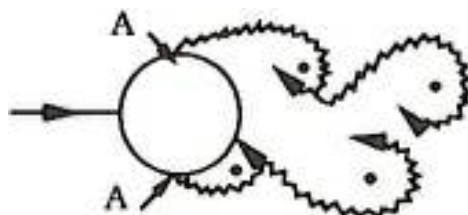
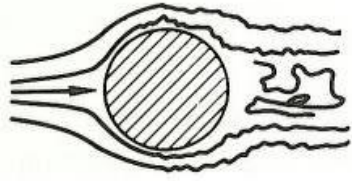
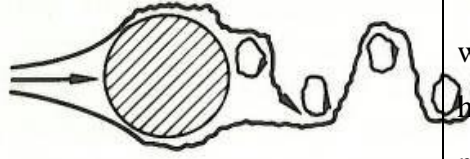
$0 < Re < 4$ to 5		Creeping or Stokes flow
$5 < Re < 45$		A pair of stable vortices
$45 < Re < 180$		Laminar vortex shedding
$200 < Re < 300$		Transition to turbulent in the wake
$300 < Re < 3 \times 10^5$		Up to the separation point the boundary remains laminar; however wake Completely turbulent.
$3 \times 10^5 < Re < 4 \times 10^6$		Turbulent Boundary layer separation, the boundary layer partly turbulent partly laminar
$Re > 4 \times 10^6$		Boundary layer and wake completely turbulent, however wake street is narrower than laminar flow.

Figure 1-5 Flow regime at various Reynolds number (Sumer & Fredsoe 2006).

As the Reynolds number is increased further the oscillating recirculation cells (vortices) detach themselves in a periodic fashion from the rear of the cylinder. The vortex street is now generated by a laminar instability of the near wake through a mutual instability of the two free shear layers. The flow throughout this laminar and unsteady shedding regime remains two-dimensional. As the Reynolds number is increased through this regime the Strouhal number increases, and the cylinder is subjected to fluctuating forces in both the streamwise and cross-flow directions. The rms lift coefficient and the base drag coefficient rise continuously with increasing Re.

- **Wake transition regime; $180 < \text{Re} < 350$ to 400**

This regime sees the development of large scale three-dimensionality in the cylinder's wake and is the regime that Roshko 1954 labelled as the transition regime. At $180 < \text{Re} < 194$ the wake develops three-dimensionality in the form of vortex loops and streamwise vortex pairs at a span wise wavelength of about 3 to 4 diameters, see Williamson(1996a). This change to three-dimensional shedding is hysteretic and is accompanied by a sudden fall in the Strouhal number and the base drag coefficient. At $230 < \text{Re} < 250$ there is a second more gradual change, which has finer scale streamwise vortices and a span-wise wavelength of about a diameter. This change is accompanied by a shift to a higher Strouhal number. The large scale three-dimensionality seen in this regime does not in itself imply that the wake is turbulent. The transition to turbulence first occurs in this regime in the far wake and gradually moves upstream with increasing Reynolds number.

- **Shear layer transition regime; 350 to $400 < \text{Re} < 2 \times 10^5$**

In this regime, which is also called the sub-critical regime and was labelled by Roshko 1954 as the irregular regime, the attached boundary layers remain laminar, transition occurs in the free shear layers and the wake is fully turbulent. The transition waves first appear in the free shear layers at $350 < \text{Re} < 400$. As the Reynolds number is increased the formation length (the length of vortex formation region) increases until at $1 \times 10^3 < \text{Re} < 2 \times 10^3$ chains of transition eddies are observed in the free shear layers. Further increasing the Reynolds number results in a decrease in the formation length and a movement of the transition points upstream towards the separation points. At $2 \times 10^3 < \text{Re} < 4 \times 10^3$ the transition eddies disappear, the formation length stops decreasing, and transition to turbulence occurs close to the cylinder. Throughout the remainder of the shear layer transition regime (up to $\text{Re} \approx 2 \times 10^5$) the transition points, the formation

length, the separation points ($\theta \approx 80^\circ$, where θ is measured from the front stagnation point), and the Strouhal number ($St \approx 0.2$) remain relatively constant. The forces experienced by the cylinder are closely related to the formation length. As the formation length increases ($350 < Re < 1 \times 10^3$) both $C_{L\ rms}$ and $-C_{Pb}$ decrease, whilst as the formation length decreases ($1 \times 10^3 < Re < 2 \times 10^4$) both $C_{L\ rms}$ and $-C_{Pb}$ increase, and whilst the formation length is relatively constant ($2 \times 10^4 < Re < 2 \times 10^5$) so are $C_{L\ rms}$ and $-C_{Pb}$. The dramatic fall and rise in $C_{L\ rms}$ for $0.5 \times 10^3 < Re < 5 \times 10^3$ is often referred as the ‘lift crises’, which should not be confused with the ‘drag crises’, of the critical regime, see below.

- **Critical regime; $2 \times 10^5 < Re < 1 \times 10^6$**

In this regime the initial flow separation is laminar; transition to turbulence occurs in the free shear layers which then reattach, resulting in the formation of thin separation-reattachment bubbles on either side of the cylinder. The turbulent boundary layer is able to withstand a higher adverse pressure gradient than a laminar boundary layer and final turbulent separation is delayed until $\theta = 140^\circ$. The postponement of final separation results in a much narrower wake than in the shear layer transition regime and a consequent reduction in the mean drag coefficient, \bar{C}_D , from $\bar{C}_D \approx 1.2$ at the end of the shear layer transition regime to $\bar{C}_D \approx 0.3$ in the critical regime. The dramatic fall in the drag coefficient is known as the ‘drag crisis’. Bearman (1969) found a regime, in which there is a separation-reattachment bubble on only one side of the cylinder, resulting in large mean lift forces, $\bar{C}_L \approx 1$.

- **Boundary layer transition regime; $Re > 1 \times 10^6$**

As the Reynolds number is increased further the separation-reattachment bubbles disappear as the transition point move further upstream ahead of the separation points and into the boundary layers. The turbulent boundary layers do not separate until $\theta = 120^\circ$, resulting in a narrow wake and a low drag. As the Reynolds number is increased the transition points and the separation points gradually move further upstream and the base suction coefficient increases.

1.6 Aims and objective

This research is mainly concerned with the numerical FSI prediction of the Vortex Induced Vibration (VIV) of elastically supported and flexible circular cylinders that are subjected to steady fluid currents, and in particular with perspective to simulate the

flexible deep-water marine riser pipes⁴ that are subjected to ocean currents. The objectives of this research can be summarised as follows:

- To study the physics of the FSI problems and the parameters that affects the VIV, especially in the context of circular cylinder and oil risers.
- To study numerical methods suitable for FSI simulation which, on the one hand, should be able to simulate the main physics related to the riser problem, and on the other, the methods should be suitable for applications in which there is limited computational power and limited simulation time.
- To apply and validate the selected methods from the previous stage for a two dimensional flow around a stationary cylinder at low Reynolds number. This initial stage includes the development and validation of an in-house code to solve this CFD problem.
- To apply and validate the FSI simulation for a two dimensional flexible cylinder using the selected methods from the second stage. This stage consists of the development and validation of an in-house FSI code to simulate forced and free vibrations of a flexible cylinder in a uniform flow based on the CFD code developed in the previous stage.
- To further develop the CFD and FSI codes to enable the analysis of the flow at realistic Reynolds numbers around stationary and flexible cylinders by addition of a turbulence modelling capability to the algorithm.
- To develop the in-house code to enable modelling of FSI for a long flexible riser in a flow field with a high Reynolds number by applying the “strip theory” features. In this stage the existing structural code will be coupled with the fluid flow to form a quasi-three dimensional FSI simulation.

To achieve these goals in this thesis, an immersed boundary method based on an interpolation/reconstruction procedure is developed; various interpolation methods

4- Riser pipes typically have axial lengths, L , of up to a few thousand meters and have outer diameters, D , of less than one metre, yielding length to diameter ratios, L/D , of $O(10^3)$. Risers are exposed to a variety of ocean currents with current speeds, U_∞ , of up to about 2 m/s and current profiles that can vary greatly with depth. The Reynolds numbers, $Re = U_\infty D / \nu$, where ν is kinematic viscosity of water, for these flows are typically of $O(10^5)$ to $O(10^6)$. As the offshore industry moves into increasingly deeper waters (>2000m depth), the riser pipes used have become longer and effectively more flexible, and are being excited into increasingly higher vibrational modes (>40th say).

which are presented in the literature are compared with a newly proposed method and the results are validated against some bench marks. In addition, a moving frame-of-reference methodology with an immersed boundary method is presented to simulate the flow around a flexible cylinder. Moreover, a moving cylinder with the newly developed interpolation method is modelled using an inertial frame of reference and the results are compared with the bench marks and the moving frame reference methodology. Finally, the Vortex-Induced-Vibration of a flexible cylinder using Aitken relaxation is modelled and the results are compared with those from the literature. Note that the last two objectives were not addressed in this thesis and will be the subject of future research.

1.7 Summary

In this chapter the physical aspects and fundamental concepts of the problem are explained and various parameters in an FSI simulation are discussed. The motivation and the goal of the research are outlined and the contribution made through this work to the knowledge is outlined. To simplify the simulation and to avoid high computational demands the models used in this thesis are limited to two dimensions and at low Reynolds numbers. However, the numerical algorithm adopted allows further development of the analysis to three dimensions and high Reynolds number flows in the future development of the work.

In the next chapter the background material and preliminary challenges common in FSI simulations are addressed. Also the application of the FSI simulation in engineering and scientific problems is explained. In addition the main approaches in the literature which are related to this research are reviewed. The main objective of the next chapter is to explain the advantages of a partitioned approach as compared to a monolithic approach. In addition, the advantages of an immersed boundary (IB) method in comparison with an Arbitrary-Eulerian-Lagrangian (ALE) methodology are briefly discussed.

Chapter 2. Background and preliminary study

Researchers have studied fluid dynamic for several centuries, numerous new ideas, tools and methods have been developed to solve a various fluid problems in a variety of engineering applications. Significant advances in the recent decades in computational power have enabled engineers to simulate very challenging fluid problems which were not deemed possible previously. Specifically the ability to accurately simulate complex fluid-solid interaction problems marks a revolution in the field of computational fluid mechanics. In this chapter some background information on state-of-the art research on FSI is presented and the reasoning behind the chosen methodology and algorithm for the riser problem is given whilst the principle approach and the main obstacles for a realistic FSI simulation are briefly addressed. Also some state-of-the-art Fluid-Structure interaction applications are introduced to demonstrate the importance of the research carried out in this field of science. In addition, at the end of the chapter the layout of the thesis is presented.

2.1 Main technical difficulties of a FSI simulation

In the recent years Fluid solid interaction has become an attractive area of research as it offers the potentials of simulating a physical phenomena as closely as possible to the that it actually occurs in nature which involves the interaction of fluid flow with a complex deformable body. As the fluid-structure interface moves in time, the spatial domain of the fluid flow will change, and the numerical simulation has to be able to handle this problem. In the conventional approach, the mesh needs to be updated in order to accurately track the interface and to represent the flow field near the boundary. Especially, in 3D problems with a complex geometry this process is quite complicated. Another challenge is to solve the fluid and structure equations simultaneously. There are some important factors that require attention when choosing the solution method for a coupled fluid-structure problem; including, a) how complex is the solid boundary?; b) how large is the structure-deformation?; c) how sensitive is the structure to a variation of

fluid dynamic of forces?; d) how accurate does the required solution need to be for FSI problems?; e) how much experienced is the researcher with FSI simulation?. In the following sections, the main approaches adopted followed to address these difficulties are discussed.

2.2 Two fundamental computational approaches

The numerical simulation of a FSI problem could be classified broadly in partitioned and monolithic approaches (see, Figure 2-1). Although, these expressions could be understood slightly differently in other fields of science, here the focus is mainly on the engineering applications.

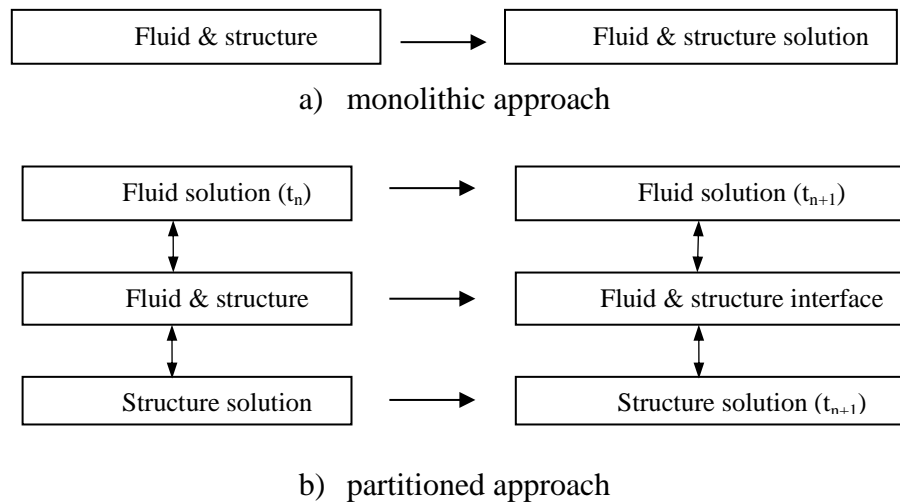


Figure 2-1: a) Schematic Monolithic approach, b) Schematic Partitioned approach.

2.2.1 Partitioned approach

In the partitioned or interaction approach (Hou et al. 2012) the fluid and structure are treated as separate entities which are solved separately with their own respective discretisation and algorithm. Interface conditions are used to communicate information between the fluid and structural solvers. The main advantages of this approach is that it allows the use of traditional solvers and advanced procedures for both the standard fluid flow and elasticity problems which simplifies the code development procedure by allowing the usage of existing simulation codes as a part of a FSI algorithm. As a result, the validation process of the code can remain limited to the validation of interface tracking. The main drawback in this approach is the implementation of the interaction of the fluid and structure and to find a converged solution; especially as the interface

location is not known and usually changes in time. In this approach, the interface location and its related parameters should be tracked and updated. This is a complicated process and may cause divergence errors in the simulation. Due to these issues, normally the partitioned approach tends to have a very slowly time converging time-step and is harder for parallel computing implementation.

The partitioned approach may be classified into weak and strong coupling approaches. In both of these approaches the fluid and structure are solved separately in time. In the weak coupling approach the parameters are not updated iteratively between fluid and structure to find a converged solution for the interface at each time step. In the strong coupling approach, however sub iterations at each time step force the fluid flow variables (velocity and pressure) to be coupled with structural parameters (deformation/displacement) and vice versa.

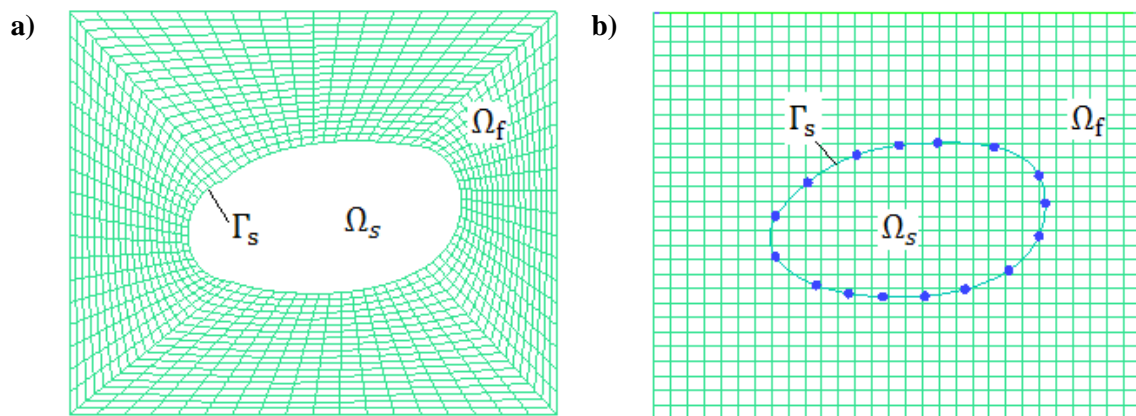


Figure 2-2: Left, Schematic of body in a fluid flow with body conforming mesh. Right, Schematic of body in a fluid flow with body non-conforming mesh method. Ω_s is the solid domain, Ω_f is the fluid domain and Γ_s is the solid boundary

2.2.2 Monolithic approach

In the monolithic approach (Hubner et al. 2004, Ryzhakov et al. 2010, 2012, 2013), both fluid and structure are treated in the same mathematical framework. In this approach, a unique formulation and algorithm is used to simulate the whole fluid and structure domain. This is a unified approach and the main advantage is that there is no need for further coupling and dealing with its associated interface tracking difficulties. Also, the method can be parallelized and can be solved using a unified space-time discretization method.

The main disadvantages of the monolithic methods are that they are typically hard to be treated numerically and it is not possible to use existing fluid and structural codes. It is also generally difficult to find a uniform formulation to solve complex problems.

2.3 Discretisation approach

Another general classification of the FSI solution procedure is based on discretisation and mesh methods which are broadly divided into the conforming mesh and non-conforming mesh methods.

2.3.1 Body conforming mesh methods-moving grid method

In the body conforming approach, the interface boundary corresponds to the physical boundary (Figure 2-2(a)). In this case, the interface location is part of the solution and the mesh needs to conform to the interface. Therefore, by advancing the solution in time, re-meshing is necessary due to the deformation/displacement of the structure (Borazjani et al. 2008 classified this method as a moving mesh method). In order to solve an FSI problem with a conforming mesh method on a structured grid using a finite difference approach, the differential form of fluid flow governing equations are transformed to curvilinear coordinates aligned with the grid lines (Ferziger and Peric 2002). Therefore, the solid boundary can be defined easily in the discretised governing equations as the grids conform to the structure geometry. For finite volume methods, the integral form of the fluid flow governing equations could be discretised for both structured and unstructured grids; and the geometrical information of the solid boundary can be used directly in the discretised equations. An important feature of this kind of FSI method is its interface tracking requirement. In this technique, the shape/position of the fluid domain is changed by the structure deformation/displacement. Therefore, the mesh moves/deforms to capture this new shape/position and to follow (track) the fluid-structure interface. The most famous example of this is the Arbitrary Lagrangian-Eulerian (ALE) interface tracking method which has gained a great deal of attention in the recent years. (Ohayon 2001, Wall 1999, Dettmer 2004, Dettmer and Peric 2006a and b, Bazilevs et al. 2006, Khurram & Masud 2006, Kuttler et al. 2006, Masud et al. 2007, Wall et al. 2007, Lohner et al. 2006, Wall et al. 2006, Bletzinger et al. 2006 among others).

2.3.2 Non-conforming mesh methods-fixed grid method

In Figure 2-2(b) a non-conforming grid is used for the flow domain. In this approach, the boundary location and interface conditions are imposed as constraints to the governing equation, and the fluid and structure equations can be solved separately on their own respective grids without any re-meshing procedure (Borazjani et al. 2008 classified this method as fixed grid method) . As the solid boundary cuts the Cartesian grid, to define the proper constraints (solid boundary) the fluid governing equations should be modified around the immersed boundary. These modifications of the governing equation are the subject of the immersed boundary method and will be reviewed in this thesis.

Clearly, in comparison with the body conforming mesh method, the main drawback of IB methods is the imposition of the boundary conditions on the solid-fluid interface (Mittal & Iaccarino 2005). In the conforming methods the solid boundaries are aligned with the grid lines. Therefore the boundary conditions (e.g. no-slip conditions) can be applied directly to the fluid governing equations. Also the grid size near the solid boundary can be chosen easier.

However, IB methods use a simple Cartesian grid to discretise the solution domain. Therefore, by using a Cartesian grid rather than a curvilinear system, the body conforming grids, can significantly reduce the number of computational processing operations due to coordinate transformations. Also, multi-grid techniques can be implemented easier when using Cartesian grids rather than curvilinear coordinate systems.

In addition, the primary advantage of the IB method is the ease of grid generation, which especially for complex geometries can be a cumbersome task in the case of the conforming mesh methods (Ferizeger and Peric 2002).

The main advantages of the IB method in comparison with the body conforming method is the ease with which moving boundaries (particularly in cases involving large displacements) are dealt with. The body conforming grid method requires the generation of a new mesh at each inner and outer time steps; also a procedure is required to map the solution from the previous grid to the new grid following the grid regeneration. As a result using a conforming mesh method could affect simplicity, accuracy and computational costs of the simulation (Tezduyar 2001).

2.4 Some FSI applications

In the recent decades Fluid-Structure Interaction (FSI) has become an important method of computational simulation. The main reason is that most of the engineering applications involve some sort of FSI problem and FSI algorithms have been used to successfully model various applications ranging from civil engineering to biomechanical, geophysical, and aero-dynamical applications. In the following section some of the main FSI applications will be introduced briefly to show the motivation and importance of this study.

2.4.1 Engineering application

Full scale wind turbine simulations (Figure 2-3a) are one of the FSI engineering applications which are performed to obtain accurate and reliable modelling as well as blade fraction prediction and design optimisation. Due to technical challenges only a few researchers (Gomez-Iradi et al. 2009, Hsu et al. 2013, Li et al. 2012) were able to recently perform a full scale wind turbine simulation. Bazilevs et al. 2013b used a partitioned approach along with the ALE-VMS finite element technique (Bazilevs et al. 2013a) for the aero-dynamical formulation and the Kirchhoff shell theory (Bazilevs et al. 2011, kiendl et al. 2009, Korobenko et al. 2013) for the blade in order to simulate a full scale wind turbine. Based on the numerical FSI analysis, they achieved a detailed structural model of the actual wind turbine with 32 different material zones, which was characterised by a distinct composite layout. With this special construction, they were able to design and built a blade with desirable natural frequencies. Also, they have validated their simulation results experimentally.

Another FSI application is the design of the cable-stayed bridges (suspension bridge) with highly nonlinear characteristics. In this simulation the deck is supported on several points by cables and the cables are connected to the support column. The Takoma Narrows Bridge (Figure 2-3b) is a famous example of the kind of structures that failed due to the resonance caused by a 64km/h wind condition on November 7, 1940 in US. Recently Hernández and Valdes 2013 used a partitioned approach to model Viaducto Zaragoza Bridge (Puebla, Mexico). For the fluid simulation, an incompressible Navier-Stokes eq. with Arbitrary Lagrangian Eulerian (ALE) formulation (Belytschko et al. 2000) is solved with the fractional step method proposed by Codina 2001. For the structural analysis a geometrically nonlinear model based on a finite element approach is

used. Also an Aitken scheme [Wüncher 2006] is used to facilitate the fluid structure interaction. Hernández and Valdes 2013 identified the resonance for some cables. To solve the problem, they suggested changing the operation conditions by adding frictional dampers at the cable connection points with the deck. Using this method, on the one hand due to the added mass the natural frequencies of cables were changed, whilst the dampers caused a reduction in the displacement amplitude which could potentially hinder the occurrence of resonance in the bridge.

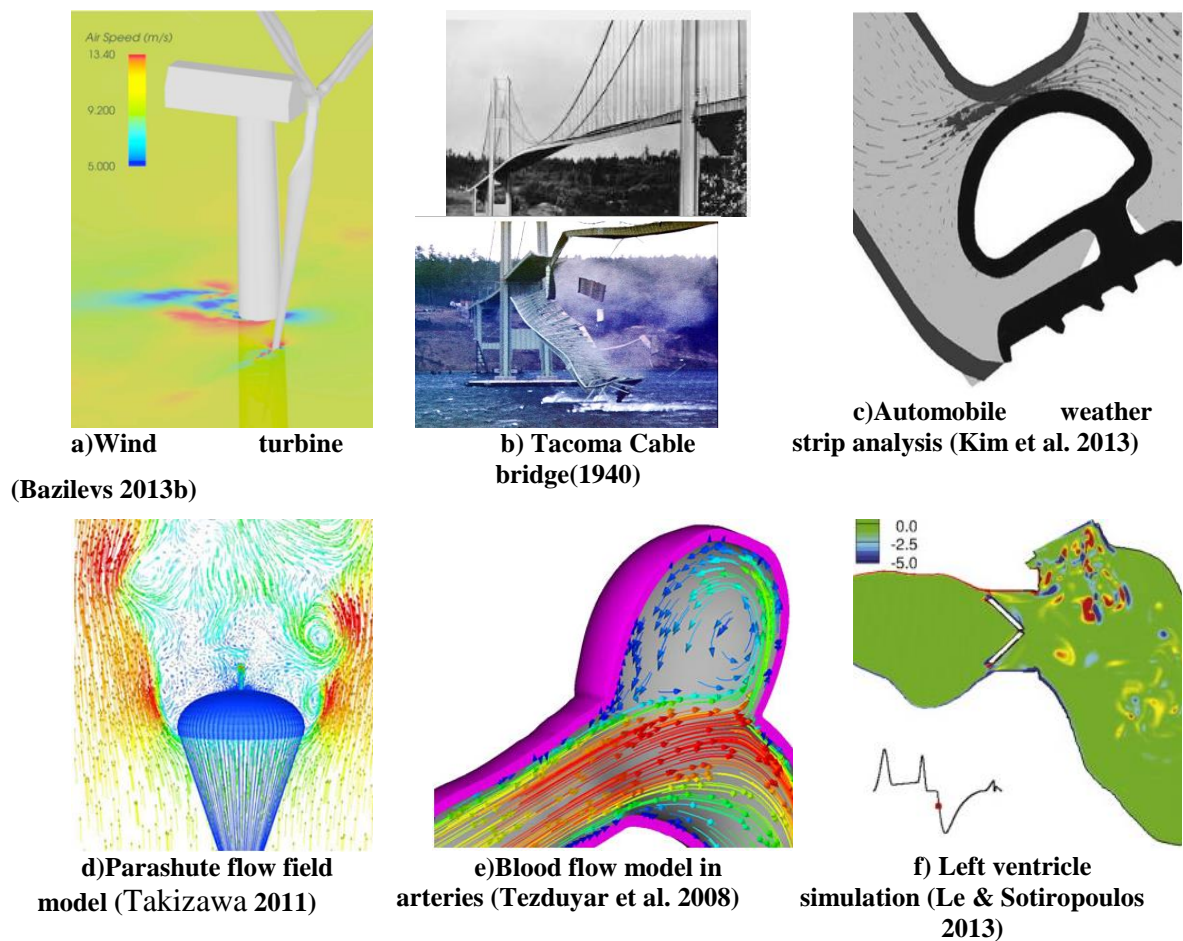


Figure 2-3: a few example of Fluid-Structure interaction (FSI) in different application

The FSI analysis is used to predict and improve the automotive weather-strip. The weather-strip (Figure 2-3c) is an important part that is employed in order to isolate the passenger compartment from water, dust and especially noise. There should be a large enough contact area of the weather-strip and the door and the body frame to minimize the wind noise level. Kim et al. 2013 implemented an FSI analysis to study the weather-strip deformation and the gap changes between the door and the frame body due to the external pressure drop that occurs when the vehicle is moving at high speed. They found

that the permanent deformation of the door weather-strip was the major factor responsible for the sound isolation performance.

Another famous and sophisticated study of the FSI technique is the comprehensive research carried out to develop the computation of spacecraft parachutes (especially for the Orion spacecraft, see Figure 2-3d) by the Tezduyar group (T_AFSM)⁵. Their preference is to use the Deforming-Spatial-Domain/Stabilized Space Time (DSD/SST) formulation (Tezduyar et al. 1992a,b,c) as interface-tracking technique, the quasi-direct FSI coupling method (Tezduyar et al. 2004 and 2006), and the stabilized space-time FSI technique (Tezduyar & Sathe 2007). Using a symmetrical FSI technique they managed to compute the parachute shapes and improve the parachute structural mechanics solutions.

2.4.2 Biomechanics applications

In spite of major developments in image processing techniques for hemodynamical studies (Hong et al. 2008, Lee et al. 2009 and Faludi et al. 2010), nowadays in vivo techniques only measure large scale blood flow characteristics. Understanding flow patterns, however, requires using very high resolutions to establish a link between heart disease and the patient's hemodynamics, an area of research which still attracts a great deal of attention (Kvitting et al. 2010). Very accurate numerical simulations could be the only option in order to better understand cardiac hemodynamics. Many researches focused on research in these areas. In the following part some of them are introduced.

Le and Sotiropoulos 2013 developed a novel model for simulating the left ventricle (LV) valve to study the FSI between the blood flow and a mechanical heart valve implant. They used a lumped type kinematic model along with Fitzhugh-Nagums framework (Fitzhugh 1961) to simulate the motion of LV wall in response to the heart pressure wave. For FSI modelling they used a curvilinear immersed boundary (CURVIB) method developed by Borazjani et al. 2008 with a domain decomposition approach. Their results were in good agreement with in vivo measurements.

Accurate FSI modelling between the deformable arteries walls and the blood flow is one of major challenges in the computational studies of cardiovascular fluid mechanics (Bazilevs et al. 2007 and Torii et al. 2007 among others). The coupled mathematical equations governing the blood flow and the structural blood arteries should be solved simultaneously to satisfy physical kinematic and kinetic conditions. Tezduyar et al. 2008

⁵ Team for Advanced Flow Simulation and Modeling (T_AFSM), Mechanical Engineering, Rice University — MS 321, 6100 Main Street, Houston, TX 77005, U.S.A.

presented arterial problems with the stabilized space-time FSI (SSTFSI) technique to increase the accuracy, robustness and efficiency of FSI modelling. They assumed that the arterial deformation during a heartbeat cycle is caused by blood pressure. As the arteries image geometries are based on time-averaged blood pressure value for patient-specific cases; they had to assume an estimated zero-pressure arterial for their further simulation. The arterial walls were modelled with geometrically nonlinear hyperplastic material (Figure 2-3e).

2.5 Summary and layout of thesis

An outline of motivation and wider possible applications of this study was provided in this chapter. The main objectives and difficulties of FSI simulations were discussed. The major classification of the FSI approaches were reviewed from different aspects. Finally, it was shown how FSI simulations are used to resolve real engineering and scientific problems by presenting a selection of research that was recently conducted.

It can be concluded that FSI problems occur in a very wide range of research ranging from the study of the behaviour of the suspension bridge, the performance and mechanics of Parachutes and wind turbines to diagnosing diseases and cardiovascular problems. Also, it is briefly explained why a specific FSI method is chosen among the other numerous versions of FSI methods which have been presented in the literature. The choice depends on the researcher's expertise, computational facilities and other features such as the required accuracy and type of the problem to be simulated. In the present study the motivation is to investigate the effect of VIV on the behaviour of flexible risers used in the offshore industry which requires a full FSI simulation. Considering the existing limitation on time and computational facilities, it was decided to study a 2D model of the riser which can easily be extended to a full three dimensional simulation, using a partitioned approach and an Immersed Boundary (IB) method. The main objective of this thesis is the implementation and validation of the IB approach using an interpolation approach in order to enforce non-grid conforming boundary condition. The future work comprises quasi 3D-simulations of long oil risers by applying the Strip theory and to add LES modelling to enable using the proposed approach in turbulent flows (higher Reynolds numbers). The layout of the rest of the chapters are as follows:

In chapter 3, a review of IB methods with a partitioned approach is presented. The Fluid-Structure Interaction (FSI) methods that are related to Immersed boundary (IB)

methods using interpolation / reconstruction are discussed in more detail. The focus will be mainly on methods for interpolation and interface tracking.

Chapter 4 discusses the methodology of the research and involves the following parts; First, the governing equations of the fluid flow and the structure are discussed briefly. This is followed by the presentation of the discretisation procedure used for the governing equations on the Cartesian grid. The IB interpolation procedure for the boundary conditions is shown in detail and also the FSI algorithm to model the problem is presented. In addition, the calculation of the lift and drag coefficients is explained using two different approaches. Finally, the coupling strategy between the fluid and structures is discussed in more details.

In Chapter 5, a parametric study and validation of the proposed algorithm is presented. In this chapter, the effects of the flow domain size in the transverse direction and behind the bluff body are presented. Also, the results of the mesh refinement study are discussed. In addition using a parametric study it will be discussed why the aspect ratio and stretching coefficient could affect the accuracy of the simulation results. Finally the influence of different mesh patterns around the solid boundary that are used in the simulation of the FSI methods is studied.

The proposed IB interpolation method is presented in chapter 6. The algorithm of this method is explained along with 4 other interpolation methods which are presented in the literature. The Strouhal number, lift and drag coefficient obtained by this method is compared with other interpolation method. The results show a good agreement with other second order accurate interpolation methods.

The results of a forced vibration and Vortex Induced Vibration (VIV) of a body in the transverse direction are presented in the chapter 7. In this chapter simulation results obtained in both a moving reference frame and an inertial frame are compared to each other and to the results presented in literature. Also a parametric study is conducted to show the appropriate mesh size.

Chapter 8 presents the conclusion and future research. In this chapter, the main results and achievements are summarised and discussed briefly and the future research is explained.

Chapter 3. Literature review

An accurate solution for Fluid-Structure Interaction (FSI) problems is of interest in many engineering and scientific applications. A FSI problem often involves simulating complex geometries with large displacement/deformation. Based on the mesh discretisation approach FSI methods can be classified into: boundary-conforming and non-boundary-conforming mesh methods (Hou et.al 2012). A well-known conforming mesh method is the Arbitrarily Lagrangian-Eulerian method (ALE). ALE methods use a grid that adapts and deforms with the moving boundary (section 2.3.1). Most of the industrial FSI applications typically have high Reynolds numbers, complex geometries and moving boundaries and need turbulence modelling and mesh deforming grid regenerating to solve the problem. Therefore, simulating FSI problems with moving grid methods (e.g. ALE method) requires significant computational power and a high storage capacity. A non-conforming mesh method (fixed grid method) is an alternative numerical approach which efficiently handles some of these complications. The Immersed Boundary (IB) method is an example of a non-conforming mesh method. This type of discretisation recently has received much attention in relation to solution of FSI problems. The non-conforming Immersed boundary (fixed grid) method is the subject of this review.

3.1 Immersed boundary methods (IB)

The immersed-boundary (IB) method is a technique for solving flow problems in regions with irregular boundaries using a simple and fixed structured grid solver. The term “immersed boundary” was initially used for a method developed by Peskin 1972 to simulate blood flow in a cardiovascular system. It was specifically designed to handle deforming (elastic) boundaries interacting with low Reynolds number flow. The simulation was carried out on a Cartesian grid. At locations where the boundary did not align with a grid line the solution algorithm was locally modified. The modifications were done in a way to enforce the desired boundary conditions on the flow domain. Enforcing the moving boundary on the governing equation is one of the most important

challenges in an IB algorithm. To do so; generally an additional forcing term is added to the governing equation ((3-1) to enforce the correct velocity boundary conditions. This term can be defined before and after discretisation of the governing equation (directly or indirectly, respectively). One of the main challenges is the definition of this forcing term which leads to various versions of IB methods. In the original immersed boundary (IB) method the effects of the moving boundaries on the flow field are applied through continuous functions, which cause diffusion of the boundary interface across a number of grid points. Due to this characteristic the method is known as the diffused method. Therefore, such IB methods require a high resolution mesh around the immersed interface to produce accurate results (Borazjani et al. 2008). Recently, numerous modifications and refinements have been proposed to enhance the accuracy, stability, and application range of the IB method (Mittal & Iaccarino 2005). For instance, a class of sharp-interface immersed boundary was introduced to remedy the diffusion of the boundary conditions at the interface. In some references “sharp interface methods” are classified as “Cartesian grids” which was originally designed for inviscid flows (Berger and Aftosmis 1998; Clarke et al. 1986 among others); In these methods the immersed boundary is modelled as a sharp interface and the effect of a moving boundary on the fluid is considered either locally by modifying the shape of the meshes to conform to the boundary (cut cell methods, Udaykumar et al. 1999); or by using a discrete delta function directly (instead of using a smooth function) into the system of discretised equations (immersed interface method, Le et al. 2006, Xu and Wang 2006 among others); or by reconstructing immersed conditions around the immersed boundary using an interpolation scheme (hybrid Cartesian/immersed boundary methods, HCIB, Gilmanov and Sotiropoulos 2005 among others); or even by combining the immersed boundary and a curvilinear body conforming mesh (Curvilinear- Hybrid Cartesian/Immersed boundary, CURVIB, Borazjani et al. 2008). In this thesis the term of “Immersed Boundary” (IB) is used to address all of the methods (including the Cartesian method). The common part in all of the methods is that the solution algorithm involves simulating viscous flows on a fix grid with (immersed or embedded) boundaries that do not conform to the grid lines.

As for the moving boundaries, also the solid boundaries do not necessarily conform to the grid lines. Fixed grid, non-confirming boundary methods can be generally classified by the way that the immersed boundary conditions are imposed on the solution domain or governing equations. In the traditional IB methods, the immersed boundaries

are imposed to the solution domain by introducing a source term (a forcing function, f) to the fluid governing equations (3-1) and (3-2).

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla P + \nu \nabla^2 \mathbf{u} + f \quad (3-1)$$

$f = 0$ in the fluid domain

$f \neq 0$ in the solid domain and at the immersed boundary

$$\nabla \cdot \mathbf{u} = 0 \text{ or } \frac{\partial u_i}{\partial x_i} = 0 \text{ in the fluid domain and the solid domain} \quad (3-2)$$

The forcing functions reproduce the effect of boundary condition on fluid solution domain. This source term or forcing function can be applied to the governing equations in two ways: the continuous forcing approach and discrete forcing approach. In the former, the forcing term is added to the governing equation before discretization of the whole physical domain and the forcing terms do not depend on the grid discretization method. In addition, the source term for the continuous forcing approach depends on the type of immersed boundary, which could be either an elastic or a rigid boundary. On the other hand; in the discrete forcing approach, the forcing term is implemented after the discretization and the source term highly depends on the discretization method. In this category (discrete forcing approach) the forcing term could be implemented either directly to the computational domain or indirectly to the governing equations by adding a discrete source term to the equations.

In the following section, some of the immersed boundary methods are briefly introduced and their advantages and disadvantages are discussed. The objective is to clarify the difference between these methods and the class of IB method that is presented in this thesis.

3.1.1 Original immersed boundary method- applicable for elastic IB

Forcing approaches are normally categorised into continuous and discrete forcing approaches. In the continuous forcing method, a forcing function is applied to the Navier-Stokes equation (3-1) in order to enforce the correct boundary condition on the structure (e.g. enforcing a no-slip boundary condition on a stationary body). The most important issue in this method is the definition of the continuous forcing function. As the solid boundaries do not coincide with the grid lines, these functions need to enforce the correct boundary condition to the solution domain.

Several different functions have been developed by Peskin 1972, Saiki and Biringen 1996, Beyer and Leveque 1992, and Lai and Peskin 2000, among others. As illustrated in Figure 3-1, in all cases, a distributed function was used rather than a sharp function. The reason behind this is that firstly the solid boundaries do not coincide with the Cartesian grid and, secondly, in this way the Gibbs' oscillations phenomenon (Briscolini & Santangelo 1988) adjacent to the solid boundaries could be suppressed.

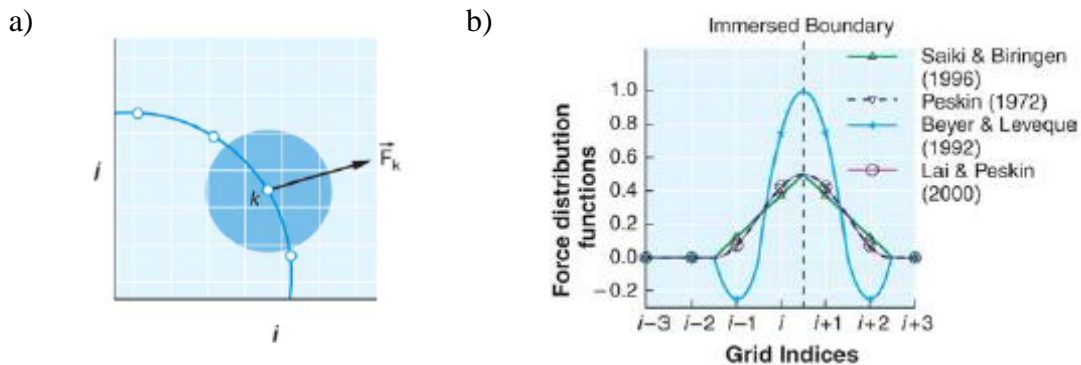


Figure 3-1a) Transferring the boundary force F_k from each material point (Lagrangian coordinate) $X(s, t)$ to the fluid. Shaded area shows the area which force effect will be distributed in the fluid domain; b) various forcing function distribution (Mittal & Iaccarino 2005).

Implementation of the boundary conditions with a continuous forcing function is attractive for elastic boundaries; as on the one hand, it has a physical interpretation for elastic boundaries and on the other, the force can be implemented easily. However, implementation of this method for rigid boundaries is relatively cumbersome due to the nature of the method as the definition of this force is based on elastic deformation of the boundary, in the linear elastic case, this is a direct application of Hook's law. When using a smooth forcing function, another problem is that the method cannot sharply represent the immersed boundaries and the effect of the boundary is distributed in the fluid domain (Figure 3-1a). As the boundary is not sharp (it is blurred) this method is not recommended for flows with a high Reynolds number (Mittal & Iaccarino 2005).

The source function, f , in equation (3-1) is defined by equation (3-3). Suppose a simple closed immersed boundary is defined parametrically by $\mathbf{X}(s, t)$, $0 \leq s \leq L_b$ and $\mathbf{X}(0, t) = \mathbf{X}(L_b, t)$ where s is a material point on the immersed boundary. $\mathbf{F}(s, t)$ is the boundary force at each segment ds of the material points. These boundary forces satisfy a generalised Hooke's law for an elastic boundary both in time, t , and space, $\mathbf{X}(0, t)$. According to equation (3-4), the force function, F , explicitly depends on the simulation

time. This definition resembles an active boundary like a muscle whose elasticity varies with time. For instance, in a two dimensional circular cylinder with several material points on its border, these forces at each material point try to preserve the circular shape of the boundary.

$$f(\mathbf{x}, t) = \int_0^{L_b} \mathbf{F}(s, t) \delta(\mathbf{x} - s) ds \quad (3-3)$$

$$\mathbf{F}(s, t) = \mathbf{S}(\mathbf{X}(0, t), t) \quad (3-4)$$

$$\frac{\partial \mathbf{X}(s, t)}{\partial t} = \mathbf{u}(\mathbf{X}(s, t), t) = \int_0^{L_b} \mathbf{u}(\mathbf{x}, t) \delta(\mathbf{x} - \mathbf{X}(s, t)) d\mathbf{x} \quad (3-5)$$

To apply this method, in the first place, the boundary force $\mathbf{F}(s, t)$ is calculated based on the displacement of material points on the boundary from the initial configuration $\mathbf{X}(0, t)$ according to equation (3-4). Then these forces are integrated over all material points to calculate the force from the immersed boundary on the fluid domain, equation (3-3).

The definition of $\delta(\mathbf{x} - s)$ characterises different versions of these methods. For instance, Lai and Peskin 2000 defined $\delta(\mathbf{x}) = d_h(x)d_h(y)$ in each coordinate direction in the vicinity of the material point on the immersed boundary, as shown in equation (3-6).

$$d_h(x) = \begin{cases} \frac{1}{8h} \left(3 - \frac{2|x|}{h} + \sqrt{1 + \frac{4|x|}{h} - 4\left(\frac{|x|}{h}\right)^2} \right), & \text{where } |x| \leq h; \\ \frac{1}{8h} \left(3 - \frac{2|x|}{h} + \sqrt{1 + \frac{4|x|}{h} - 4\left(\frac{|x|}{h}\right)^2} \right), & \text{where } h \leq |x| \leq 2h; \\ 0 & \text{otherwise.} \end{cases} \quad (3-6)$$

In the second step, the Navier-Stokes equations (3-1) and (3-2) are solved to find the updated velocity at the new time step. In these equations the force term, f , is the force from the boundary on the fluid domain described by equation (3-3) which has been calculated in the previous step. Finally, equation (3-5) is solved with new velocity to find the new configuration of the structure. The process will be repeated in time to eventually find the developed solution for the problem. The key point in this type of immersed boundary methods is that the structure should be elastic (not rigid solid) as the force at each material point is calculated from a ‘‘Hook’s law’’ equation. For rigid bodies the method described below is suggested by a number of researchers.

3.1.2 Feedback forcing approach- applicable for rigid IB

According to the studies by Goldstein et al 1993 and Saiki and Biringen 1996, an analytic expression for the force $f(x_s, t)$ acting on the boundary x_s at time t can be specified by the feedback forcing equation (3-7):

$$f(x_s, t) = \alpha_f \int_0^t [u(x_s, \hat{t}) - V(x_s, \hat{t})] d\hat{t} + \beta_f [u(x_s, \hat{t}) - V(x_s, \hat{t})] \quad (3-7)$$

Where $V(x_s, \hat{t})$ is the velocity of the moving boundary, $u(x_s, \hat{t})$ is the velocity of the fluid on the boundary, and α_f and β_f are constants. The above equation is a feedback based on the velocity difference $u(x_s, \hat{t}) - V(x_s, \hat{t})$ which imposes the flow velocity on the immersed boundary, u , to match the velocity of the immersed boundary, V , at the same point. The major drawback for the feedback forcing is that this method requires very small time steps $CFL = O(10^{-3} - 10^{-2})$. More details can be found on Fadlun et al. 2000.

3.1.3 Physical Virtual Model (PVM) approach

Introducing the boundary force, f , in equation (3-1) is the main challenge in an immersed boundary method. Lima E Silva et al. 2003 proposed a PVM approach to calculate the interfacial forces without an ad hoc constant that usually depends on domain and numerical model. In this method, the force is calculated over a sequence of Lagrangian points, representing the interface, using the updated velocity and pressure from the Navier-Stokes equation in the fluid domain. Silva implemented the conservation of momentum theorem in an arbitrary control volume included each Lagrangian points to calculate the interfacial force. One of the advantages of this method is that the forces due to friction and pressure is calculated separately, which are important factors in a vortex induced vibration context. This method is called the Physical Virtual Model as it is only based on the conservation laws. The simulation results for the flow around a stationary cylinder were found to match the numerical and experimental data in the literature.

3.1.4 Immersed interface approach

Using several grid nodes in the vicinity of the immersed boundary to spread the forcing function is an inherent feature of the original immersed boundary method. This issue complicates the extension of this method to high Reynolds number flows in

practical applications (Gilmanov and Sotiropoulos 2005). However, LeVeque and Li 1994 proposed a type of IB method, called the Immersed Interface Method (IIM) to overcome this issue. IIM only modifies the grid nodes in the immediate vicinity of the immersed boundary to enforce a set jump condition at the interface by adding the forcing function. This method maintains the interface sharpness for the immersed boundary and is second order accurate. In the method proposed by Lee and LeVeque in 2003 the boundary force is decomposed into a tangential and a normal component. The tangential forces were added to the momentum equations, while the normal component is applied to the pressure Poisson equation in terms of a pressure jump condition over the interface.

3.1.5 Fictitious domain method

Glowinski et al. 1999 proposed a different method by applying a fictitious domain method⁶. In this method the fluid governing equations were enforced inside of the rigid body as well as outside in the fluid domain. The fluid velocity inside the solid body is enforced by a Distributed Lagrange Multiplier (DLM) to behave like a rigid body (boundary) motion in the fluid domain. In fact, the multiplier creates additional body force inside the particle to maintain the rigid body motion for the solid body. Baaijens 2001 developed a DLM based on the Mortar Element⁷ (ME) method to impose the no slip boundary conditions as an equation for the Lagrange multiplier. He applied this method to simulate the behaviour of a two dimensional flexible slender body in a channel flow with fluctuating inlet velocities. Yu 2005, extended the fictitious domain method to three dimensional simulation and non-slender bodies. He used the continuum equations for the general material rather than Newton's equation for rigid body motions. Like the DLM in the rigid body motion, where a pseudo body force introduces the rigid body motion to the fluid domain, in his method the Lagrange multiplier forces the fictitious fluid (inside the solid) to move with the same velocity as the solid.

However, due to the need for an accurate representation of the boundary layer in high Reynolds number flow, the use of distributed, smooth forcing functions near the immersed boundary is not desirable. In these cases it is recommended to use a sharp

⁶ -fictitious domain methods, also known as domain-embedding methods, are one type of solution methods for partial differential equations. The main idea is to replace a simple but larger domain (the fictitious domain) in a problem with a complex time dependent geometry (see, Glowinski et al. 2000).

⁷ -Mortar methods are discretization methods for partial differential equations, which use separated discretisation, in non-conforming subdomains and the meshes in subdomains do not match at the interfaces, however, the equality of parameters on the interface is enforced by Lagrange multipliers to preserve the accuracy of the solution (Maday et al 1989).

interface with a higher local accuracy near the boundary. This goal can be achieved by imposing the boundary conditions directly on the immersed boundary. There are two well-known methods that fit into this category: the Ghost-Cell Finite-Difference Approach and the Cut-Cell Finite-Volume Approach.

3.1.6 Ghost-Cell approach

In the Ghost-Cell approach the immersed boundary is implemented by using ghost cells. Ghost cells are cells inside the solid boundary which have at least one neighbour on the fluid side. The parameters (imaginary velocity and pressure) in the ghost cell inside the solid are defined by an interpolation method which implicitly enforces the correct boundary condition for the immersed boundary. In this approach, there is a possibility of losing accuracy as this method is based on the mirrored velocity with respect to the solid body (as discussed by Kang 2008).

3.1.7 Cut-Cell method – Cartesian method

All of the immersed boundary methods discussed so far are not designed to consider the conservation laws near the solid boundary. However, the Cut-Cell method in combination with a Finite-Volume approach is designed in order to preserve the conservation of momentum and mass near the boundary. In this method, the cells which have been cut by the immersed boundary are reshaped or absorbed by neighbouring cells in order to form a new trapezoidal control volume cell shape. In this method, the governing equations are not modified. This method has been used by Mittal et al. 2003 & 2004 to simulate vortex-induced vibration around a stationary and a moving body and for free falling objects. Although considered to be consistent, this method suffers from slow convergence (due to small cells) and is regarded as being too complex which are its major disadvantages. Also, the extension of this method to 3D is not straightforward and needs complex polyhedral cells, which complicate the discretization of the Navier-Stokes equations (Ghias et al. 2007).

3.1.8 Direct forcing approach

The Navier-Stokes equations usually cannot be integrated analytically to define the forcing functions. Therefore, often, it is not possible to derive an analytical forcing function to enforce specific boundary conditions. To tackle this problem, a method has been suggested by Mohd-Yusof 1997 and Verzicco et al. 2000. In this method, which is

known as the direct forcing approach, the forcing functions are subtracted from the numerical solution after discretizing the Navier-Stokes equations. The important advantage of this method is that there is no need to define the forcing function parameters prior to solving the Navier-Stokes equations and there is no stability constraint due to the use of continuous forcing functions (Gibb's oscillation). However, it is still required to implement the distributed forcing functions which strongly depend on the discretization algorithm. Mohd-Yusof 1997 developed an expression for the forcing function, which does not have the time steps restriction. In this method, the discretized form of the Navier-Stokes equation is used directly to calculate the force expression by imposing the velocity of the immersed boundary (equation (3-9)).

$$\frac{u^{n+1} - u^n}{\Delta t} = RHS^{n+1/2} + f^{n+1/2} \quad (3-8)$$

In equations (3-8) and (3-9), the *RHS* comprises the convective, viscous and pressure terms of the Navier-Stokes equation. Therefore, the forcing term, $f^{n+1/2}$, is simply calculated to enforce the immersed boundary condition on the fluid domain and the governing equations by using equation (3-9).

$$f^{n+1/2} = -RHS^{n+1/2} + \frac{V^{n+1} - u^n}{\Delta t} \quad (3-9)$$

Another important issue in the direct forcing approach is the interpolation procedures. As the immersed boundary does not necessary coincide with the fluid parameters on the grid especially in a staggered arrangement, it is necessary to calculate and enforce the forcing function interpolation. Fadlun et al. 2000 have implemented three different interpolation schemes and compared their accuracy. As one of the main parts of this research relates to the interpolation procedures, various interpolation schemes have been studied in detail in the latter part of this review.

3.1.9 Interpolation or reconstruction method

In the interpolation method, the forcing function, f , equation (3-1) is not directly calculated to enforce boundary conditions. Instead, the flow velocity is interpolated at the interface cells and the forcing term is imposed indirectly to the discrete equations or directly to the computational domain. In other words, at the interface cells an interpolation formula replaces the Navier-Stokes equations. The interface points are defined as the points in the fluid domain near the solid boundary for which one of the neighbouring points in the discretized equations is inside the solid domain. Therefore,

the parameters related to these points cannot be updated through solving the governing equation. Any cells that contain one or more interface points are called interface cells. Figure 3-2 (left) shows the interface cells around a circular cylinder in which at least one of the points' parameters cannot be updated directly using the governing equations. For instance, in Figure 3-2(right), to update the velocity at point A using the governing equations its 8 neighbouring velocities are needed; however two of 8 velocity-components are inside the solid boundary. Therefore, the velocity at point A should be interpolated between the boundary points and other points inside the flow field.

Besides its simplicity, this method has a few advantages. There are no severe limitations on the time steps as the velocities on the boundary are implicitly or explicitly applied to the governing equation (fluid domain). In addition, the velocities in the fluid domain are separated from the non-physical velocity inside the solid boundary. As in the most of immersed boundary methods, due to its nature a secondary non-physical flow is created inside the solid boundary.

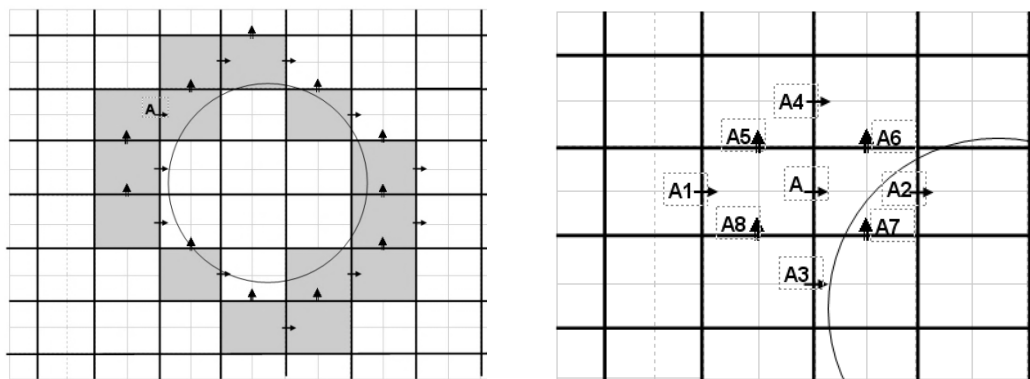


Figure 3-2: A 2D Cartesian mesh with a solid boundary (circle). Interface points, that require interpolation, are identified by arrows. Points A1 to A8 are all neighbouring points of A. Note that A2 and A7 are inside the solid domain.

One of the disadvantages of the interpolation or reconstruction method is the decoupling of pressure and conservation of mass at the interface. Iaccarino and Verzicco 2003 showed that a linear interpolation method is acceptable for those cases in which the first points of the interpolation in the fluid are inside the viscous sub layer. Several interpolation methods have been introduced by Ghias et al. 2004, Fadlun et al. 2000, Kang et al. 2009, Choi et al. 2007 among others. In the next section some of these interpolation methods are discussed in more detail.

3.2 Defining the interface cells

The general key feature for any sharp immersed boundary method is that the governing equations are solved on a grid that does not conform to the immersed solid boundary (moving or stationary). The governing equations are solved only on the fluid domain nodes in which all of the neighbouring points are located entirely in the fluid domain and the fluid nodes in immediate vicinity of the immersed boundary are updated by interpolation. In other words, the interface points are not updated inside of the governing equations. Instead, they are used as boundary conditions for the governing equations. Therefore one of the initial steps in applying this immersed boundary (IB) method is to classify the nodes in the background grid in three categories; the cells that are thoroughly within the fluid domain, the cells completely within the solid domain and the interface cells. The interface cells are the cells in which the immersed boundary crosses or in which the parameters cannot be updated using the governing equation. This classification of the grid cells can be performed in several ways. It is a straightforward procedure to identify them in a simple or analytically well described geometry. However, a complex computational geometry tool is required to identify the interface cells for a complex geometry (Iaccarino and Verzicco 2003). Gilmanov et al. 2003 presented an algorithm to identify interface nodes that is only applicable to simple convex bodies. Another algorithm, presented by Gilmanov & Sotiropoulos 2005, is applicable to identify the interface nodes for an arbitrary geometry. Borazjani et al. 2008 used the classical method of the point-in-polyhedron problem for their computational geometry. In the following part the methods of Borazjani et al. 2008, Gilmanov and Sotiropoulos 2005 are briefly discussed.

3.2.1 Point-in-polyhedron algorithm

Classifying the Cartesian grid into fluid and solid parts is a classical problem of the point-in-polyhedron procedure in computational geometry. A point and a polyhedron, whose geometry is introduced by its sides are defined in space, it is then required to establish whether the point is contained inside or outside of the polyhedron. In a two dimensional geometry the problem is downgraded to a point-in-polygon problem; with two major solution methods, the so called ray-casting method and the winding number method (Haines 1994). In the ray-casting method, a half infinite ray is drawn from a point in the domain and the number of intersections between the half infinite ray and the

polygon edges is counted. If the number is odd (point A in Figure 3-3b) than the point is inside the polygon (on the immersed solid), otherwise it is located outside of the polygon (in the fluid domain).

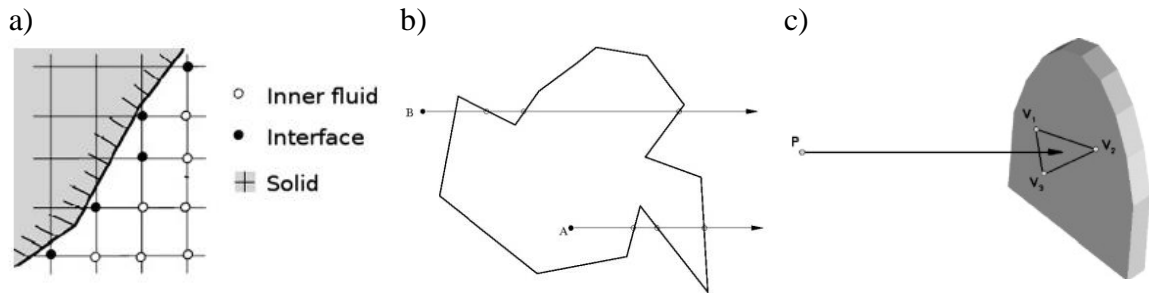


Figure 3-3: a) background Cartesian grid with a polygon immersed boundary classified to fluid, solid and interface nodes, b) a ray casting test method for a polygon; point A is inside and point B is outside the polygon, c) a ray casting test method for a polyhedron (Borazjani et al 2008).

Expanding the ray-caste-polyhedron method to 3D is straightforward and is briefly described as follows. Suppose that the surface of polyhedron is discretised with an unstructured triangle mesh and a point $p(x,y,z)$ is defined in space. A line is casted from the point p to the point $S(x,y,z)$ outside of polyhedron, the number of intersection of the ray with the triangle elements on the surface of polyhedron shows if the point p is outside (fluid node) or inside (solid node) of the polyhedron (Figure 3-3c). The core of the ray-triangle intersection algorithm is explained by Moller and Trumbore 1997 (for more details, see Borazjani et al. 2008).

3.2.2 Interfacial marker at the interface discontinuity algorithm

This methodology was initially proposed by Udaykumar et al. 1997. The fluid-structure interface is tracked as a discontinuity. The algorithm is very robust and is applied in a variety of problems, especially in FSI problems with a sharp immersed boundary method. The detailed algorithm is presented in the papers published by Udaykumar et al. 1997. Key features of this method are presented here to facilitate further discussion about the immersed boundary method with a sharp interface.

In this method an open or closed immersed boundary with an arbitrary shape is represented by interfacial markers which are defined by their arc length coordinates $X(S)$ in Figure 3-4. The markers are equally spaced with a spacing size of the same order of the background Cartesian grid. The start point is defined such that with increasing s , the fluid is always on the left hand side. By fitting quadratic polynomial with each point

through and its two immediate neighbours, the unknown coefficients in equation (3-10) are calculated to obtain a local parameterisation of the immersed boundary at each interfacial point.

$$x(s) = a_x s^2 + b_x s + c_x \text{ and } y(s) = a_y s^2 + b_y s + c_y \quad (3-10)$$

Using this calculating the normal vector to the immersed boundary is straightforward by employing the following equations.

$$n_x = \frac{-y_s}{\sqrt{(x_s^2 + y_s^2)}} \text{ and } n_y = \frac{-x_s}{\sqrt{(x_s^2 + y_s^2)}} \quad (3-11)$$

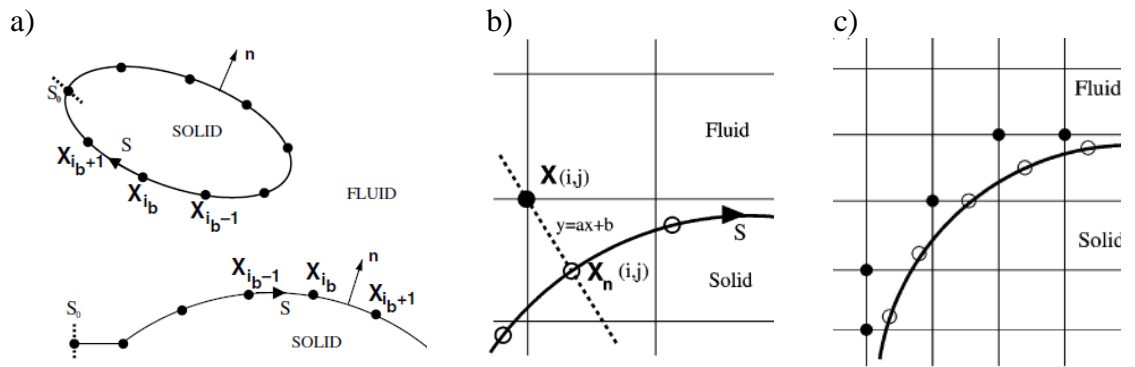


Figure 3-4 a) Definition of the immersed boundary topology by interfacial markers, arc length vector and normal vector; b) Identification of fluid nodes from solid nodes using a normal vector; c) Demonstration of interface nodes (o) and marker points (•), (Balaras 2004).

Having a local parameterization of the immersed boundary around each marker points, it is possible to identify the grid point closest to each marker point in the background Cartesian by using an iterative method, like the Newton-Raphson method. According to Figure 3-4b, a line (vector) is defined from each Cartesian point in the vicinity of the marker point perpendicular to the local approximation of the immersed boundary. The inner product of these vector and normal vector, equation (3-11), of immersed boundary at each marker point shows that if the grid point on the Cartesian background is on the fluid domain or in the solid domain. If the inner product is positive the point is outside (for closed solid boundaries) or on the left hand side (for the open boundaries) of the solid interface. (see Balaras 2004 for more details). Figure 3-4c shows the interface cells (black circle) which need special treatment to enforce the solid boundary condition in a sharp interface IB method.

3.3 Boundary Reconstruction/Interpolation

It is well known that the majority of immersed boundary approaches need some sort of interpolation procedure. For instance, the forcing method discussed earlier was based on the assumption that the unknown (velocity) positions on the grids exactly match with the immersed boundary location. In this case the boundary coincides with the grid lines especially with moving boundaries, which is not the case for complex geometries. In particular, for staggered arrangement, even if the grid lines and boundary location coincide together for one unknown (e.g. velocity in x direction) they will not coincide for the other unknowns. Therefore interpolation is needed in the IB solution procedure to enforce the immersed boundary in the presence of non-matching grid lines.

Due to the forcing method, the interpolation procedure would be different and can be categorised by two main approaches. In the first approach; the forcing function is spread in the vicinity of the immersed boundary, which in the original IB approach introduced by Peskin 1972 is achieved using a discrete Dirac delta function (section 3.1.1). The main drawback for this approach is that this spreading acts as extra dissipation close to the IB which could lead to an inaccurate prediction of the boundary layer. In the second approach a local solution of the unknown (velocity) is reconstructed to enforce the IB as a sharp interface with a relatively high degree of accuracy (depends on its procedure). This method of interpolation is widely used in the indirect forcing approaches. In other words, in the vicinity of the immersed boundary the flow governing equations are replaced by an interpolation equation. In this way, the unknown at the interface cells are determined and these values will be used as the boundary values for the governing equation. This process is repeated at each time step and the flow parameters in the interface cells are updated by direct interpolation and used as boundary conditions for the flow solver. Various interpolation methods have been developed to address this issue.

In this review, a number of interpolation procedures which could potentially be used in indirect discrete forcing approaches (interpolation or reconstruction) are categorized, explained and compared briefly in the following section.

3.3.1 Stepwise geometry -No interpolation

The simplest possible method is to identify edges of the interface cells as the solid boundaries to define the solid domain. In fact, in this way there is no interpolation

needed and the solid boundary will have a stepwise shape (Figure 3-5a). Also, the boundary itself will be somewhat diffused, as in the staggered methods the boundary conditions for the different velocity components are applied at different sides of an element. Fadlun et al. 2000 proposed this method for calculating and imposing forcing functions, respectively, from and to the velocities around immersed boundaries. As interpolation is not needed, this method will be less expensive while still giving acceptable results. The disadvantage of this method is that (especially on course meshes) the shape and size of the enforced boundary is different from the real solid boundary which could affect the lift and drag forces. Also, this method is only first order accurate in space.

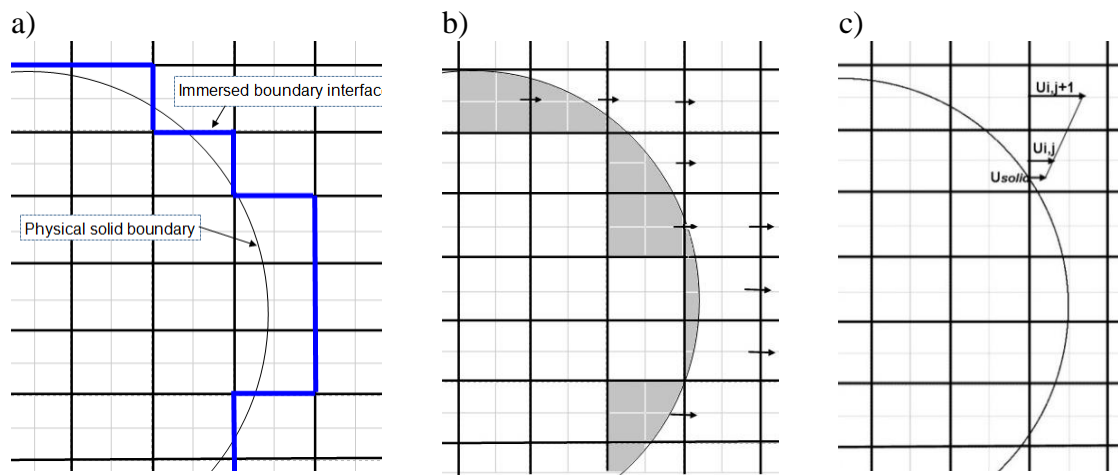


Figure 3-5 interpolation procedure sketch for u velocity in an staggered arrangement a) without interpolation b) weighted interpolation c) linear interpolation

3.3.2 Weighted method

This method is similar to the one discussed above. The major difference is that the boundary values (force term) for those cells that are crossed by the IB are corrected based on the volume/surface of cell that is occupied by the structure (Figure 3-5b). For each of the force and velocity components a coefficient is determined that corresponds to the ratio of the fluid part of the cell to the whole area of the cell, which is a first order accurate method in space. Fadlun et al. 2000 used this method to scale the forcing of the velocities closest to the boundaries.

3.3.3 Linear interpolation method

In this method, the velocities in the interface cells are calculated by a linear interpolation between the velocity at the solid boundary (applying the no slip condition) and in one point inside the fluid. Fadlun et al. 2000 suggested this interpolation method

to enforce the boundary condition to the fluid domain. Also, Kang et al. 2009 used this interpolation method for the immersed boundary but applied it in a direction parallel to the grid lines. In Figure 3-5c, the interpolation procedure for an IB cell velocity, $U_{i,j}$, in the vertical direction using the U_{solid} and $U_{i,j+1}$ (inside the fluid domain) is shown.

Application of this approach for a complex geometry could lead to a possible ambiguity in choosing the interpolation direction. Figure 3-6a, illustrates such ambiguities as the interface (IB cell) velocity, $U_{i,j}$ could be interpolated either in the horizontal or vertical direction (Kang et al. 2009). Balaras 2004 suggested using the linear interpolation scheme in a direction perpendicular to the boundary to overcome this problem. According to Figure 3-6b he suggested to calculate $U_{virtual}$ in the fluid domain at a location where $h_1=h_2$; therefore the interface velocity, $U_{i,j}$ is obtained from U_{solid} and $U_{virtual}$ using

$$U_{i,j} = \frac{h_2}{h}U_{solid} + \frac{h_1}{h}U_{virtual}, \quad \text{where } h = h_1 + h_2 \quad (3-12)$$

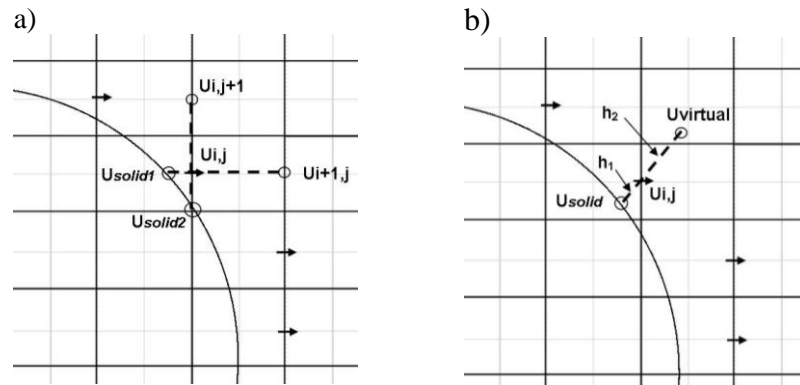


Figure 3-6: calculating the interface cell velocity by linear interpolation; a) ambiguity in the direction of interpolation, Fadlun et al. 2000 model b) linear interpolation perpendicular to the IB, Balaras 2004 model.

Balaras predicted three possibilities for calculating $U_{virtual}$ from U_1 to U_4 in fluid domain. According to Figure 3-7a) if none of the U_1 to U_4 is an interface velocity the $U_{virtual}$ can be calculated by equation (3-13); where α_i is the standard bilinear interpolation coefficient.

$$U_{i,j} = \sum_1^4 \alpha_i U_i \quad (3-13)$$

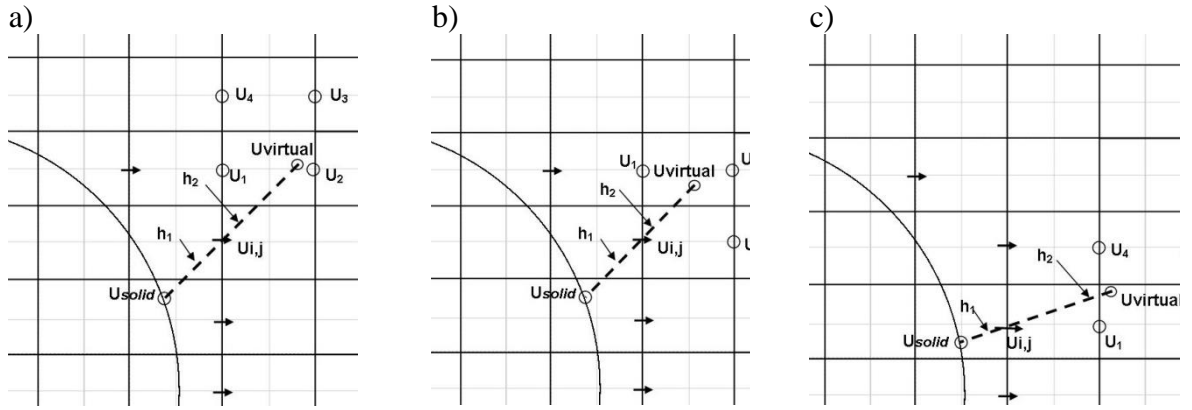


Figure 3-7: interpolation scheme in direction perpendicular to the IB, Balaras 2004; three boundary options depends on the immersed boundary geometry and local grid size.

Figure 3-7b) shows that if one of velocities around the U_{virtual} is the interface velocity, $U_{i,j}$, the U_{virtual} is interpolated from U_1 to U_3 . In addition, if more than one of the velocities around the U_{virtual} is an interface velocity (Figure 3-7c), in this case h_2 is chosen larger than h_1 in a way that at least three neighbouring velocities of U_{virtual} do not coincide with the interface velocities, $U_{i,j}$. Linear interpolation is a second order accurate scheme (for more detail see Balaras 2004, Kang et al. 2009).

Gilmanov et al. 2003 presented and applied the Balaras interpolation method to three dimensional problems. As explained earlier in the reconstruction/interpolation method, the entire fluid domain is solved using the boundary values specified at the interface cells, and the immersed bodies are excluded from the computation. Suppose at time step, n , all the velocities and pressures in the fluid domain (for example point α , β , δ and γ at Figure 3-8) are known and also suppose that the boundary conditions are known at all vertices of the unstructured grid at the same time step. To advance the flow governing equations to the next time step the values of the immersed cells (for example point b at Figure 3-8) are interpolated linearly between point a on the structure and point c inside fluid domain. Gilmanov et al. 2003 used another interpolation, equation (3-14), to calculate the value of the parameters (velocities) in Figure 3-8 (points c & a).

$$U_a = \left(\sum_{m=1,3} U_m / s_m \right) / \left(\sum_{m=1,3} 1 / s_m \right) \quad (3-14)$$

Where $m=1,3$ are the three vertices of the triangular element which include point a , and s_m are the distances between point a and each of the three vertices of the triangular element. The same method is used to calculate the boundary condition at point c by interpolating the values defined at α , β , δ and γ in Figure 3-8. Also, the pressure gradient

is calculated as a Dirichlet condition in a similar way at point b (see Gilmanov et al. 2003 for more detail).

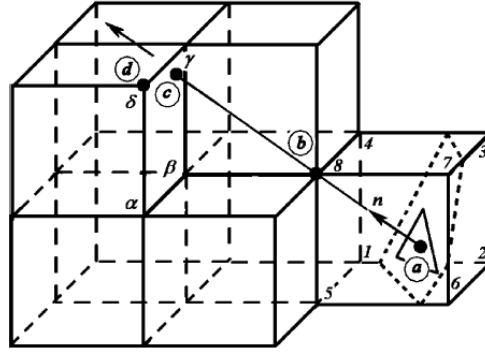


Figure 3-8: Schematic reconstruction of the IB unknown “b” with interpolation in the wall-normal direction. The triangle represents an unstructured element of the IB. The dash line is the intersection of the body with the Cartesian grid (Gilmanov et al. 2003).

3.3.4 Bilinear interpolation method

Kang et al. 2009 introduced this method as a linear interpolation method, (Standard Reconstruction Method, SRM). Two adjacent velocities in the horizontal and vertical directions and the velocity of the solid boundary are used to obtain the interpolated velocity at each interface point near the immersed boundary (Figure 3-9). Equation (3-15) is the interpolation formula for the velocity, where the coefficient ω represents the interpolation weight for each of the velocities.

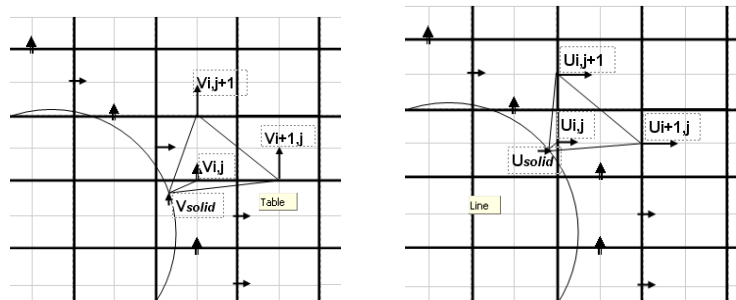


Figure 3-9: Standard Reconstruction Method (SRM) for velocity in vertical (left) and horizontal (right) direction

$$U_{i,j} = \omega_{i+1,j}U_{i+1,j} + \omega_{i,j+1}U_{i,j+1} + \omega_{solid}U_{solid} \quad (3-15)$$

To solve the governing equations in a fractional-step method, after each time step $U_{i,j}$ is calculated from the momentum interpolation, $\hat{U}_{i,j}$, followed by projection onto each divergence-free field. The intermediate velocity itself, $\hat{U}_{i,j}$ is calculated using an interpolation formula for the cells near the solid boundary.

3.3.5 Revised interpolation method

In spite of the various advantages in standard interpolation/Reconstruction methods (SRM) that have been discussed so far, there are several short-comings as well. An important issue is the decoupling between the pressure field and the local velocity near the immersed boundaries. Also, there is no explicit contribution of the velocity or pressure gradient at the previous time steps in the interpolation formula which could cause abnormal pressure gradients near the immersed boundary (Kang et al. 2009).

Kang et al. 2009 has revised the above interpolation methods to use the velocity field in the previous time step to obtain a more accurate interpolation for the next time step. To do so, the explicit difference between the velocities at two consecutive time steps is used to calculate the interface velocities at the new time step. In a fractional step strategy to solve the Navier-Stokes equations this difference could be defined as $\Delta U_{i,j} = \hat{U}_{i,j}^k - U_{i,j}^{k-1}$; where $\hat{U}_{i,j}^k$ is the intermediate velocity before the pressure (conservation of mass) projection step. In this case the interpolation formula based on the previous velocity is defined as,

$$\Delta U_{i,j} = \omega_{i+1,j} \Delta U_{i+1,j} + \omega_{i,j+1} \Delta U_{i,j+1} + \omega_{solid} \Delta U_{solid} \quad (3-16)$$

Or alternatively it can be expressed as:

$$\hat{U}_{i,j}^k = \omega_{i+1,j} \hat{U}_{i+1,j}^k + \omega_{i,j+1} \hat{U}_{i,j+1}^k + \omega_{solid} \hat{U}_{solid}^k + \eta (\hat{U}_{i,j}^{k-1} - \omega_{i+1,j} \hat{U}_{i+1,j}^{k-1} - \omega_{i,j+1} \hat{U}_{i,j+1}^{k-1} - \omega_{solid} \hat{U}_{solid}^{k-1}) \quad (3-17)$$

$$\text{Where } \eta = \sqrt{(\omega_{i+1,j} + \omega_{i,j+1}) / \omega_{solid}} \text{ and } (\eta \leq 1)$$

In addition, to compensate for the decoupling between the velocity and the local pressure, Kang et al. 2009 explicitly added the effect of the pressure gradient to the interpolation equation.

$$\begin{aligned} \hat{U}_{i,j}^k = & \omega_{i+1,j} \hat{U}_{i+1,j}^k + \omega_{i,j+1} \hat{U}_{i,j+1}^k + \omega_{solid} \hat{U}_{solid}^k \\ & + \eta (\hat{U}_{i,j}^{k-1} - \omega_{i+1,j} \hat{U}_{i+1,j}^{k-1} - \omega_{i,j+1} \hat{U}_{i,j+1}^{k-1} - \omega_{solid} \hat{U}_{solid}^{k-1}) \\ & - \delta_k \Delta t \left(\left. \frac{\partial p^{k-1}}{\partial x} \right|_{i,j} - \omega_{i+1,j} \left. \frac{\partial p^{k-1}}{\partial x} \right|_{i+1,j} - \omega_{i,j+1} \left. \frac{\partial p^{k-1}}{\partial x} \right|_{i,j+1} \right. \\ & \left. - \omega_{solid} \left. \frac{\partial p^{k-1}}{\partial x} \right|_{solid} \right) \end{aligned} \quad (3-18)$$

In equation (3-18), instead of the pressure gradient at the solid boundary, $\left. \frac{\partial p^{k-1}}{\partial x} \right|_{solid}$, the pressure gradient at the interface cell, $\left. \frac{\partial p^{k-1}}{\partial x} \right|_{i,j}$, is used, which is not affecting the second order accuracy of the formula. In the above formula the easiest choice is to select

$\delta_k = 1$, in which case the pressure gradients at the interface cell and the momentum equation are the same.

3.3.6 Quadratic interpolation method

In addition, Kang et al. 2009 introduced a quadratic interpolation formula to improve the Revised Linear Interpolation Method (RLIM). In this method the local pressure gradient is incorporated in the velocity interpolation to compensate the decoupling between the pressure and the velocity near the solid boundary, however there is no extra user defined parameter like the RLIM method in the interpolation formula. Figure 3-10a&c illustrate their interpolation method in the two-dimensional case in horizontal and vertical directions. Four adjacent velocities are used to enforce the momentum equation by a quadratic interpolation. For those interface points where the quadratic interpolation is not applicable (see, Figure 3-10b) due to geometry and curvature of the immersed boundary, it is replaced with the linear interpolation. Equation (3-19) is another version of the quadratic interpolation formula, where the origin of the local coordinate system is located at the interface velocity, $\hat{U}_{i,j}$. Though this interpolation formula has a third order accuracy, the overall accuracy of the flow solver is second order. Therefore, a quadratic interpolation only gives more degrees of freedom (more flexibility) to the velocities near the immersed boundary rather than higher order accuracy (more than two) to the problem.

$$\hat{U}_{i,j}^k = a_{i+1,j}^k x_{i+1}^2 + b_{i+1,j}^k x_{i+1} + a_{i,j+1}^k y_{j+1}^2 + b_{i,j+1}^k y_{j+1} + \hat{U}_{i,j} \quad (3-19)$$

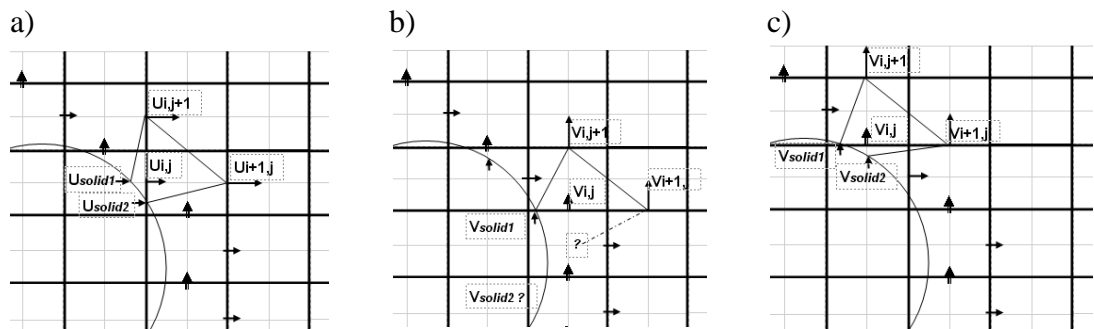


Figure 3-10: Quadratic interpolation method for an interface velocity in the horizontal (left) and the vertical (right) directions. The middle pane shows that it is not always possible to use this type of quadratic interpolation.

Moreover, in the quadratic interpolation there is no user defined parameter. Kang reported that this method could become unstable in some cases. Two remedies are

suggested; using a cubic interpolation instead of a quadratic one, which increases the number of coefficients and therefore the complexity of the method. Another suggestion is to use a linear interpolation when the two velocities on the immersed boundary (U_{solid1} and U_{solid2} or V_{solid1} and V_{solid2}) are nearer than some threshold to each other (more detail see Kang et al. 2009).

3.3.7 Higher order interpolation methods

Choi et al. 2007 also introduced a higher order interpolation method. It has been shown that a power law interpolation is better than a linear interpolation method, for higher Reynolds number. They introduced the concept of tangency correction by resolving the velocity into the normal and the tangential direction to the immersed boundary. The velocity profile in the tangential direction is written as a general power law in terms $\sim n^k$, rather than assuming a linear trend (n is the normal coordinate). Small value of power k , ($k=1/7$ or $1/9$) preserve the expected logarithmic distribution near the wall region which is necessary for application of a turbulent model. The normal velocity profile is defined in such a way that its second derivative is maintained at the immersed surface ($n=0$) to satisfy the Neumann boundary condition for the wall normal pressure gradient. Choi's numerical results shows that for Reynolds numbers less than 1000 a linear distribution of the velocity profile ($k=1$) is required, however, in problems with Reynolds numbers greater than 10,000 using the law power ($k=1/7$ or $1/9$) gives a more realistic flow separation result (more detail see Choi et al. 2007).

3.4 Interface tracking methods

An important challenge faced when using immersed boundary methods is to maintain stability in the FSI simulation, which may lead to very small time steps (Fauci and Fogelson 1993, Peskin 2002). It is possible to improve the numerical stability by calculating the boundary forces implicitly (strong coupling) in advance (time). Also, in the presence of very strong interaction between the fluid flow and structure (eg. blood flow in arteries), a strong coupling between the solvers is required to stabilise the simulation in a partitioned approach. This is due to the additional flow acceleration that is acting on the solid which is known as the added mass effect (Causin et al. 2005 and Idelsohn et al. 2009 among others).

Another challenge in the FSI modelling is the interface tracking between the fluid and the structure. Sub-iterations (strong coupling) between the fluid and the structure

solution increase the numerical stability of the interface tracking methods. Wood et al. 2010 explained that the FSI method becomes unstable in absence of sub-iteration steps between flow and structure solutions at each time step (weak coupling). They showed that one additional sub-iteration can reduce the numerical error by two orders of magnitude without adding a substantial overhead to the program; more sub-iterations could achieve an even better convergence.

In this context, the IB partitioned approaches are classified into strong (Farhat et al. 2006) and weak (Quarteroni et al. 2000) coupling methods. Weak coupling could produce acceptable results when the coupling (interaction between the fluid and the structure) is not strong like in aero-elasticity problems (Farhat et al. 2006). However, weak coupling may lead to instabilities when the density of the fluid and the structure are similar, for example in simulating blood flow in arteries.

Although many methods were developed to improve the treatment of the interface to gain a better accuracy, efficiency and stability for the FSI simulation (Tu and Peskin 1992, Mayo and Peskin 1993, Fauci and Fogelson 1993, LeVeque and Li 1997, Lee and LeVeque 2003, Mori and Peskin 2008, Newren et al. 2008, Hou and Shi 2008, Cenicerros et al. 2009 among others), it remains a challenge to produce a computationally efficient IB method (Hou et al. 2012).

Due to the coupling of the interface configuration and the boundary forces with the fluid flow simulation, solving a FSI problem implicitly requires solving a very large system of nonlinear equations. Finding a converged solution to such a large system is a very challenging problem. Due to these challenges, most of the simulations were originally based on explicit methods. Recently, however, implicit methods have been developed that exploit the improved computational power. Newren et al. 2008 presented an unconditionally stable procedure with a second order Crank-Nicolson formulation where inertia forces are neglected, assuming a linear and self-adjoint force at the interface. Mori and Peskin 2008 suggested a similar scheme and proposed a fully implicit method. Cenicerro et al. 2009 designed a cost-effective algorithm to solve the linear system arising from the implicit discretization. Also Wang 2006, 2007 and 2010 employed a fully implicit time integration algorithm along with a matrix free combination of Newton-Raphson and General Minimal RESidual (GMRES) solvers.

In addition, Badia et al. 2008 proposed a method to estimate the interface location and to replace Neumann and Dirichlet boundary conditions by a general Robin transmission method in a new FSI iteration. Having a better prediction of the fluid

structure interface, Farhat et al. 2006 proposed a FSI model with a second order accuracy in time. Another second order FSI method was developed by Zhang et al. 2007. In their model the CFD code was considered as a black box. A reduced-order method was introduced by Vierendeels et al. 2008 in an attempt to improve the computational efficiency. In the following section some of these methods are studied and compared in more detail.

3.4.1 Second-order accuracy without sub-iteration (loosely coupled, weak solution)

Li et al. 2002 proposed a loose coupling between the fluid flow and the structure for simulating an FSI problem in a relative frame of reference method. In the first step the force at t^{n+1} , F^{n+1} is calculated explicitly from the forces F^n and F^{n-1} using extrapolation or relaxation factors (equation ((3-20))). The parameter ϑ can be either a relaxation or extrapolation factor. $\vartheta = 3/2$ results in a second order extrapolation. In the second stage, the structural governing equation is solved to find the displacement, X^{n+1} , velocity, V^{n+1} , and acceleration, a^{n+1} at t^{n+1} by a second order, trapezoidal scheme ((3-21))). In the third step, the fluid governing equation is updated from time t^n to t^{n+1} with respect to the velocity, V^{n+1} , and the displacement, X^{n+1} , using a new boundary condition at time t^{n+1} . These steps are repeated for the whole range of FSI simulation problems in time domain.

$$F^{n+1} = \vartheta F^n + (1 - \vartheta)F^{n-1} \quad (3-20)$$

$$X^{n+1} = X^n + \frac{1}{2}\Delta t(V^n + V^{n+1}) \quad (3-21)$$

$$V^{n+1} = V^n + \frac{1}{2}\Delta t(a^n + a^{n+1}) \quad (3-22)$$

$$a^{n+1} + \frac{c}{m}V^{n+1} + \frac{K}{m}X^{n+1} = \frac{F^{n+1}}{m} \quad (3-23)$$

In addition, Li et al. 2002 suggested using an implicit method so that F^{n+1} is updated with the newly calculated flow field velocity, V^{n+1} , to fulfil a convergence criterion according to equation ((3-24)),

$$\|F_{j+1}^{n+1} - F_j^{n+1}/F_{j+1}^{n+1}\| < \varepsilon \quad (3-24)$$

In equation ((3-24)), j is the sub-iteration index and ε is a prescribed small constant. If the newly calculated F_{j+1}^{n+1} is converged then the program goes to the next time step.

Farhat et al. 2006 suggested two second-order temporal accuracy algorithms. These procedures are second-order accuracy for both the flow and the structures fields. Farhat

proposed a three-point backward difference method for solving the flow field and the structure in the following steps. a) Predicting the interface velocity based on the second-order accurate structural solution. b) Calculating the interface location based on the structural governing equation. c) Solving the fluid governing equation based on the new boundary conditions (interface location). d) Calculating force, pressure and shear stress, acting on the fluid-structure interface to be used to predict the interface velocity as described in for the first step.

The second method of Farhat was a half time step procedure; using the following steps: a) Predicting the interface velocity based on the governing structural equations in half of the time step; b) Calculating the interface position based on the velocity and governing structural equation in half of the time step; c) solving the fluid governing equation based on the velocity and location of the interface in half of the time step; d) Calculating the force from the fluid flow at half of the time step to find the velocity and location of the interface at the full time step by solving the governing structural equations.

Zhang et al. 2007 studied the accuracy, stability and efficiency of their two proposed FSI algorithms for an aero-elastic flutter benchmark. Their first algorithm solves structural dynamic equations under hydrodynamic forces. Those forces are calculated by a black box CFD simulation.

The structural equations are solved with a standard fourth order accurate Runge-Kutta method. The discretised equation uses the fluid pressure at $p(t)$ and $p(t + \frac{\Delta t}{2})$, in which the latter is predicted by a second order backward extrapolation procedure (equation ((3-25))).

$$p(t + \Delta t) \approx \frac{1}{8}(3p(t - 2\Delta t) - 10P(t - \Delta t) + 15p(t)) \quad (3-25)$$

In the next step, the structural equations are solved to find the new position of the boundary. Once the new boundary position is predicted, the CFD code solves the fluid governing equations to generate a new pressure distribution based on the new boundary location. The new pressure distribution is applied to the structural equations in the next time step.

The second algorithm of Zhang is based on a multi-step, implicit second order Adams Bashforth method to solve the structural equations, in which the predictor is an explicit second-order Adams scheme. The forces from the fluid flow in the predictor step

at time $n+1$ can be approximated by a second order accuracy (equation (3-26)) or a fourth order accuracy (equation (3-27)).

$$p(t + \Delta t) = 2p(t) - P(t - \Delta t) \quad (3-26)$$

$$p(t + \Delta t) = 4p(t) - 6P(t - \Delta t) + 4P(t - 2\Delta t) - pP(t - 3\Delta t) \quad (3-27)$$

Both Zhang algorithms require to call the CFD code only once per time step. They presented the result for a flutter benchmark. The simulation results confirm that their algorithms are superior to the conventional algorithm in which the fluid flow and structure equations are solved alternately.

3.4.2 Fixed point FSI coupling algorithm with dynamic relaxation

One of the most basic yet efficient approaches is the fixed point algorithm with dynamic relaxation which was suggested by Kuttler & Wall 2008, Mok & Wall 2001 and Wall 1999. This algorithm calculates the FSI interface within an incompressible fluid flow (of a body placed with a flexible structure). A Dirichlet-Neumann scheme is used to apply the algorithm to the FSI interface and to couple the nonlinear equation of flow to the structures. In this scheme, the flow becomes the Dirichlet part of the problem by the defining the flow velocity at the interface and the structure becomes the Neumann part of the scheme by describing the forces on the interface. This technique couples two black boxed field solvers (fluid and structure solvers) and predicts the FSI solution.

In the first place, a suitable location is predicted for the interface, y_{Γ}^{n+1} . Then, the interface velocity, $u_{\Gamma}^{n+1} = \frac{y_{\Gamma}^{n+1} - y_{\Gamma}^n}{\Delta t}$, is calculated for the flow domain based on the predicted location at the new time step, y_{Γ}^{n+1} , and the previous location, y_{Γ}^n . In the next step, the flow governing equation is solved based on this new velocity boundary condition (Dirichlet) to find the coupling forces on the interface. Finally, the governing structural equation is solved based on the calculated force (pressure) to obtain the structural displacement y_{Γ}^{n+1} . At this stage it is possible to define an iterative cycle to find a converged value of the structural displacement. A stopping criterion (equation ((3-28)) is introduced to check the convergence of the results.

$$r_{\Gamma,i+1}^{n+1} = y_{\Gamma,i+1}^{n+1} - y_{\Gamma,i}^{n+1} \quad (3-28)$$

In this equation, i , is the iteration index, and the residual, $r_{\Gamma,i+1}^{n+1}$, should be less than a certain value (Deparis 2004) to achieve convergence. To accelerate convergence, a relaxation coefficient is introduced.

$$y_{\Gamma,i+1}^{n+1} = y_{\Gamma,i}^{n+1} + \omega_i r_{\Gamma,i+1}^{n+1} = \omega_i r_{\Gamma,i+1}^{n+1} + (1 - \omega_i) d_{\Gamma,i}^{n+1} \quad (3-29)$$

As a result, the fixed-point algorithm to solve the FSI problems consists of a relaxed FSI cycle with appropriate relaxation factor and convergence criteria. The relaxation parameter should be small enough to guarantee the convergence of the FSI simulation, while avoiding unnecessary iterations. Also, it should be as large as possible to use as much as possible of the new solution for the next iteration. Kuttler & Wall 2008 suggested two methods to define the relaxation parameter; Aitken relaxation and steepest descent relaxation.

The main idea in the Aitken method (equation (3-30)) is to use the values from two previous iterations to calculate the current coefficient; therefore, there is no possibility to calculate the relaxation parameter after only one iteration.

$$\omega_{i+1} = -\omega \frac{r_{\Gamma,i+1}}{r_{\Gamma,i+2} - r_{\Gamma,i+1}} \quad (3-30)$$

3.4.3 Reduced-order modelling (ROM) and interface location prediction

Vierendeels et al. 2008 proposed a ROM procedure to solve the FSI problem for a heart valve as a bench mark. The heart valve was modelled with series of rigid links, connected by hinges along with a torsional stiffness. The sets of implicit FSI equations for the discretised fluid and structure are represented symbolically by equations (3-31) and (3-32) respectively.

$$G(x^{n+1}, p^{n+1}) = 0 \quad (3-31)$$

$$p^{n+1} = F(x^{n+1}) \quad (3-32)$$

A sub-iteration can be performed to find the interface at the new time step (x^{n+1}) as equation (3-33).

$$0 \approx G(x^{n+1,k}, p^{n+1,k}) \quad (3-33)$$

$$= G(x^{n+1,k-1}, p^{n+1,k-1}) \times \Delta x + \left. \frac{\partial G}{\partial p} \right|_{x^{n+1,k-1}, p^{n+1,k-1}} \times \Delta p$$

$$\text{Where } \Delta p \approx p(x^{n+1,k-1} + \Delta x) - p(x^{n+1,k-1})$$

3.5 Moving frame of reference

As it has been mentioned earlier, one of the main problems for simulating flow around a flexible structure is the moving boundaries. The two main techniques to tackle

this problem are classified as: deforming grid methods similar to the Arbitrary-Lagrangian-Eulerian (ALE) approach (Donea et al. 1982) and fixed grid methods, such as the Immersed Boundary (IB) method (Peskin 1972).

An alternative approach for solving the moving-boundary flow problem for a flexibly mounted, non-deforming (rigid) body is to attach the coordinate system to the body, and solve the Navier–Stokes equations in a moving frame of reference. The advantage of such an approach is that an optimized direct solver for the fluid can be efficiently applied. This is particularly important when considering the very long simulation time typically required to capture the instability of the fluid-structure interaction. Newman and Karniadakis 1988 applied a coordinate transformation to a flexible cable in the (x, y) direction; but did not include a rotational degree of freedom in their simulation, which is of great significance in some problems. Li et al. 2002 used a similar approach as Newman and Karniadakis for a single body undergoing both translation and rotation. They introduced a coordinate transformation attached to the transforming/rotating body. This formulation proved to be very flexible in handling every possible motion of a body in two dimensional plane. In the following section their method is briefly explained. In the Chapter 7 this method combined with the immersed boundary interpolation method will be used to simulate the flow around a flexible circular cylinder.

3.5.1 Moving frame Formulation

Assume that instantaneously the body translates by $d = (g(t), h(t))^T$ and rotates by an angle $\theta = \theta(t)$, in the absolute frame of reference (\acute{x}, \acute{y}) . Then a corresponding moving frame of reference (x, y) can be attached to the body using the transformation.

$$\begin{aligned}\acute{x} &= g(t) + x \cos\theta + y \sin\theta \\ \acute{y} &= h(t) - x \sin\theta + y \cos\theta\end{aligned}\tag{3-34}$$

Here, the prime denotes the absolute frame of reference, and the coordinates $\mathbf{x} = (x, y)^T$ denote the moving frame of reference whilst $d = (g(t), h(t))^T$ is the coordinate of the origin of the moving frame of reference in the absolute frame of reference. The rotational angle $\theta(t)$ is defined to be consistent with the aeronautical sign convention for the angle of attack, i.e., rotating the model clockwise in a flow from left to right increases the angle (Figure 3-11).

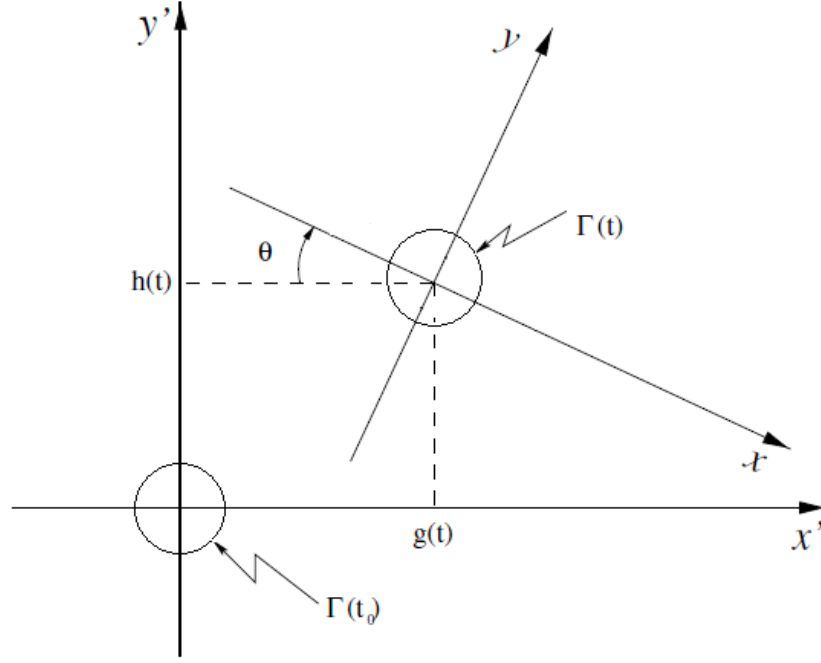


Figure 3-11 : Coordinate transformation

Li et al. 2002 have derived the Navier-Stokes equations in a moving reference for a two dimensional case using the above transformation, and obtained:

$$\nabla \cdot V = 0 \quad (3-35)$$

$$\frac{\partial V}{\partial t} + V \cdot \nabla V = -\nabla p + \vartheta \nabla^2 V + G(v, t) \quad (3-36)$$

$$G(v, t) = 2\dot{\theta} I_0 V + (\dot{\theta})^2 X + \ddot{\theta} I_0 X - A^T \ddot{d} \quad (3-37)$$

Where A is the rotation matrix:

$$A = \begin{pmatrix} \sin\theta & \cos\theta \\ -\cos\theta & \sin\theta \end{pmatrix}, \quad A^T = A^{-1} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad (3-38)$$

And the velocity V in the moving frame reference is defined by:

$$V = \dot{\theta} I_0 X + A^T (\dot{V}' - \dot{d}), \quad \text{where} \quad I_0 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad (3-39)$$

The term $2\dot{\theta} I_0 V$ is related to the deflecting or Coriolis force and $(\dot{\theta})^2 X$ is related to the centrifugal force. The terms $A^T \ddot{d}$ and $\ddot{\theta} I_0 X$ are related to the forces due to unsteady translation and rotation.

The pressure is kept unaltered. This not only simplifies the implementation of the pressure boundary condition for the splitting scheme in the moving frame of reference, but is also convenient when coupling the flow solver with a structural equation which is primarily driven by the pressure forces in the absolute frame of reference.

For a non-accelerating motion this formulation is effectively stating that the problem of a moving body in a uniform flow is equivalent to that of a stationary body in a moving flow. For example, when a body is fixed in a flow and has an angle θ with the free stream velocity, then according to the above formulation: $\dot{\theta} = 0, \theta \neq 0$. Therefore, the problem becomes flow around a body with not slip boundary conditions where the inlet velocities are $V = A \dot{V} = (\cos\theta; -\sin\theta)^T$.

3.5.2 Moving frame reference boundary conditions

The far field Dirichlet boundary condition for the transformed Navier–Stokes equations can be specified using equation (3-41), i.e.,

$$V = \dot{\theta} I_0 X + A^T (\dot{V} - \dot{d}); \quad (3-40)$$

\dot{V} is velocity in the far field in the absolute frame of reference.

For many numerical schemes, a far field Neumann boundary condition for the far field is typically defined in the absolute frame of reference, such that:

$$\nabla \dot{u} \cdot \dot{n} = g_N^{\dot{u}}, \quad \nabla \dot{v} \cdot \dot{n} = g_N^{\dot{v}} \quad (3-41)$$

Where \dot{n} is the outward normal to the boundary and $g_N^{\dot{u}}, g_N^{\dot{v}}$ are known functions. Therefore it is necessary to transform this condition into the moving frame of reference.

Li et al. 2002 derived the corresponding Neumann boundary conditions in the moving frame of reference as:

$$\nabla u \cdot n = g_N^{\dot{u}} - \dot{\theta} n_y, \quad \nabla v \cdot n = g_N^{\dot{v}} + \dot{\theta} n_x \quad (3-42)$$

3.6 Freshly cleared nodes

An important issue arises when the movement of an immersed interface (boundary) relative to the fixed background grid expose new nodes to the fluid domain that were originally in the solid body. The new fluid nodes need to be addressed carefully when using an FSI sharp interface method. Udaykumar et al. 2001 resolved this issue by introducing a cell-merging formulation along with a quadratic interpolation among neighbouring points in the fluid for the cut cell approach. Gilmanov and Sotiropoulos 2005 reported that in the direct forcing approach or reconstruction method, as long as the new grid point in the fluid is considered as an immersed cell, there should not be any problem as according to the definition of the immersed or interface cell, the values of the parameters at these points are interpolated before updating the fluid governing equation. In other words, this implies that the movement of the immersed boundary at each time

step should be less than a computational cell. To enforce the above conditions they have introduced the following restriction on the time step.

$$\Delta t \leq \frac{h}{\max_{m=1,M} (|U_m^n|, |V_m^n|, |W_m^n|)} \quad (3-43)$$

Where in equation (3-43) U, V and W are the Cartesian components of the velocity and h is the minimum grid size near the immersed boundary. Gilmanov and Sortirpolous explained that in the dual time step approach (with extra inner iterations to achieve a strong coupling at each time step) the above criterion is less restrictive in comparison with to the Courant condition for the stability of the simulation.

3.7 Mass conservation and pressure treatment near IB

Conservation of mass is a very important factor for the calculation of the pressure as pressure is a Lagrange multiplier for the continuity equation. There are few methods to conserve mass near an immersed boundary depending on the IB method. Figure 3-12(a) shows that in the continuous forcing approach the mass conservation is implemented for all the cells in the fluid and solid domains by assuming there is no IB. The primary advantage of this method is that there is no need to take extra measures in order to fulfil the mass conservation near the IB. However, few other issues need to be addressed. According to the equations (3-1) and (3-2) the gradient of the pressure in the fluid side of IB, $\Gamma_{IB-fluid}$, and at the immersed boundary Γ_{IB} (and/or solid domain) can be calculated by equations (3-44) and (3-45), respectively. Where in the equation (3-44), u^{fluid} is the flow velocity on the fluid side of domain; while in equation (3-45), the velocity could be either for the fluid or the solid domain around the IB. Practically, it has not been proven that the pressure gradients from these two equations are always the same at the IB; unless the f is zero or there is a discontinuity in the velocity near the immersed boundary (jump condition). Therefore, applying equations (3-1) and (3-2) to the whole domain (fluid and solid, Figure 3-12(a)) may not be sufficient to accurately predict the pressure around the immersed boundary (Kang et al. 2009).

$$\nabla P|_{\Gamma_{IB-fluid}} = \left[-\frac{\partial u^{fluid}}{\partial t} - u^{fluid} \cdot \nabla u^{fluid} + \vartheta \nabla^2 u^{fluid} \right]_{\Gamma_{IB-fluid}} \quad \text{when } \nabla \cdot u^{fluid} = 0 \quad (3-44)$$

$$\nabla P|_{\Gamma_{IB}} = \left[-\frac{\partial u}{\partial t} - u \cdot \nabla u + \vartheta \nabla^2 u + f \right]_{\Gamma_{IB}} \quad \text{when } \nabla \cdot u = 0 \quad (3-45)$$

Another example to show the need for an additional pressure treatment near Γ_{IB} is the case of very thin IB layer between two channels with a steady laminar flow in opposite directions. In this case, the pressure gradients at each side of the Γ_{IB} can be calculated by applying equation (3-44) which results in an independent (decoupled) solution across Γ_{IB} (more detail see Kang et al. 2009).

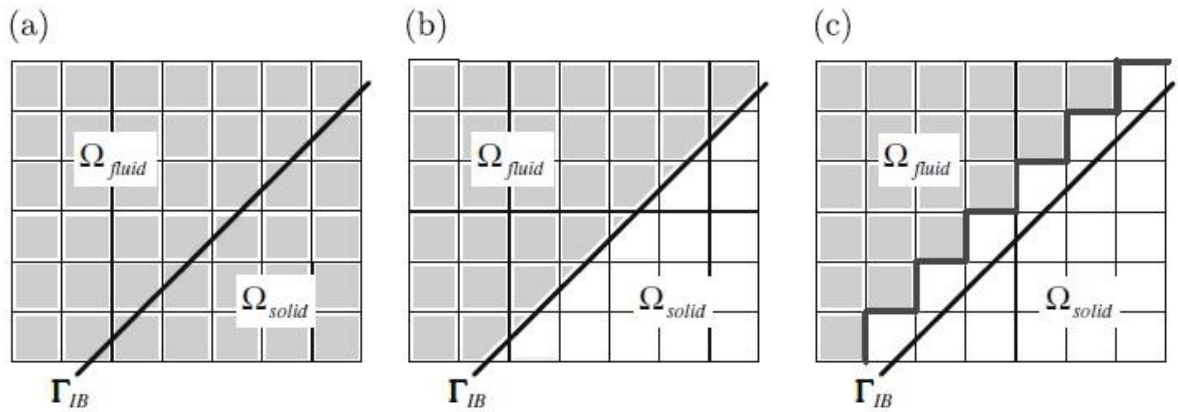


Figure 3-12: Various methods for conservation of mass depending on the IB method a) mostly for the continuous forcing approach (standard approach) b) mostly for the cut cell approach with a reshaped control volume c) mostly for the reconstruction method, conservation of mass only in fluid domain (Kang et al. 2009).

The decoupling of the solid domain from the fluid domain across the Γ_{IB} is similar to the immersed interface method that uses “jump conditions” at the immersed boundary (Lee and LeVeque 2003). Also Kim et al. 2001 suggested implementing a modified continuity equation in the solid domain or at Γ_{IB} in order to remove the unwanted coupling of the non-physical flow field in the solid domain to the actual flow field domain. The parameter ‘q’ in this equation is known as the mass forcing term.

$$\nabla \cdot u = q \quad \text{in solid domain and } \Gamma_{IB} \quad (3-46)$$

In addition, when the immersed boundary, Γ_{IB} , is forced inside the fluid domain either directly by using a Dirac delta function, f , in equation (3-1) or indirectly, by reconstructing the velocity at the interface nodes (the nodes on the fluid domain which have one neighbour in the solid domain) the forcing function and the reconstructed velocity should satisfy conservation of momentum.

Figure 3-12(b) shows another method to conserve mass around the immersed boundary, Γ_{IB} . In this method the cells which are crossed by the immersed boundary are divided into the fluid region which is solved purely by applying the Navier-Stokes equation, and the solid region which does not need any solution. In this case, Γ_{IB}

separates these two regions. Conservation of mass is satisfied in the reshaped control volumes similar to the Finite volume Method (FVM) for an unstructured grid. A Similar method was used by Udaykumar et al. 2001 among others in the cut cell approach. The advantages of this method are that the conservation of mass is automatically enforced at Γ_{IB} and the pressure gradient and velocity in the fluid region are independent of the parameters in the solid region. However, this method is very complicated for a moving complex geometry especially in three dimensions (Iaccarino and Verzicco 2003). Another shortcoming presented by Kirkpatrick et al. 2003 is that when the size of the reshape mesh is very small, the matrix condition number rises significantly.

In the final method, according to the Figure 3-12(c), the control volumes which are in the solid region and the ones crossed by the immersed boundary, Γ_{IB} , are excluded from the computational region and the mass conservation is only implemented in the fluid domain. This method does not create a pressure coupling problem between the fluid and solid region. Also, this method does not suffer from reshaping issues and other complexities of the cut cell method.

3.7.1 Fictitious adding mass effect

The added mass effect rises only when an immersed body in an oscillatory stream experiences an oscillatory hydrodynamic force in the direction of the stream. Morison et al. 1950 modelled this oscillatory force as being composed of an inertial and a drag force. The inertial force is in phase with the flow's acceleration whilst the drag force is in phase with the velocity.

Inertial forces consist of two parts: a 'buoyance force' which account for the pressure gradient required to accelerate the flow past the body and the 'added mass'. The added mass is the fictitious mass of the fluid that is considered to be attached to the structure, and if the structure is permitted to vibrate, it moves with the structure and therefore adds to its inertia. The contribution of the added mass force to the inertial forces acting on a vibrating structure is proportional to the relative acceleration of the fluid with respect to the structure.

3.8 Calculation of force on immersed boundary

Generally, the forces on the immersed boundary can be classified as drag and lift forces if the component of force is in line with the flow or in transverse direction to the flow, respectively. Also, the force on a body in a fluid flow arises from two parts; the

pressure distribution and the shear stress along the submerged body. Depending on the type of the immersed boundary method, the calculation of the force on the immersed boundary is performed in different ways. In the continuous forcing approach (original IB and feedback forcing method), the force on the IB is calculated directly with a continuous function to be added to the Navier-Stokes equation which is not subject of this thesis. In the sharp boundary approach, either using the cell deformation (cut cell method) or the reconstruction method, calculating the immersed boundary forces is a challenging task and despite a great number of publications this subject has rarely been explained properly (Balaras 2004). Lai and Peskin 2000 suggested three methods to calculate the drag force on an immersed boundary for the continuous forcing approach. Balaras 2004, based on Lai and Peskin method, suggested to employ conservation of momentum to calculate the immersed boundary forces in the sharp interface IB methods. In addition, Choi et al. 2007 used the method proposed by Balaras in their simulation. Moreover, there are other direct methods presented in literature to integrate the force due to the pressure as well as the force due to friction. For instance, Li et al. 2002 used a direct integration of the force on an immersed boundary in a moving reference frame method. In the section below some of these methods will be discussed in more detail.

3.8.1 Integrating continuous force

This method is only applicable in combination with the continuous forcing approach. The force, f , in the right hand side of equation (3-1) is integrated in the fluid domain or the force F (in equation (3-4)) is integrated over the material points at the immersed boundary (equation (3-47)). In this equation the negative sign can be explained by Newton's third law and L_b is the number of material points on the immersed boundary.

$$F = - \int_{fluid\ domain} f d\mathbf{x} = - \sum_{i=1}^{L_b} \mathbf{F}_i dS_i \quad (3-47)$$

3.8.2 Direct calculation of surface forces

The aerodynamical force exerted on a body by the flow is the integral of the local stress. Equation (3-48) expresses, σ , the local stress in terms of the pressure (normal stress) and τ , tangential stress (shear stress). The local stress can be integrated over the immersed boundary to calculate the forces from the fluid on the body (equation (3-49)).

$$\sigma = -p\mathbf{I} + \boldsymbol{\tau} \quad (3-48)$$

$$\mathbf{F} = \int_{\Gamma(t)} \boldsymbol{\sigma} \mathbf{n} ds = - \int_{\Gamma(t)} p \mathbf{n} ds + \int_{\Gamma(t)} \boldsymbol{\tau} \mathbf{n} ds = F_p + F_v \quad (3-49)$$

Where \mathbf{n} is the outward unit normal on the body, F_p refers to the pressure force and F_v refers to the viscous force. The above integration is defined in the absolute frame of reference.

3.8.3 Application of momentum conservation

When the under-lying Cartesian mesh for the flow, in an IB method is not aligned with the material points of the structure in a Lagrangian frame work; it is difficult to calculate interface forces in an immersed boundary method. To solve this problem, Balaras 2004 suggested a method based on the conservation of momentum for an optional control volume around the immersed body. Suppose Γ_0 is the boundary of a fixed control volume surrounding the immersed boundary, Γ_b . The conservation of momentum is applied to the bonded surface, $\Gamma = \Gamma_0 \cup \Gamma_b$. Using this, the force from the fluid on the immersed boundary is calculated by equation (3-50) in vector notation or by equation (3-51) in index notation for a two dimensional problems (for more details see Lai and Peskin 2000 and Balaras 2004).

$$\vec{F} = \frac{d}{dt} \int_{\text{boundaed area}} \rho \vec{u} dA - \int_{\Gamma_0} (\rho \vec{u} \vec{u} + p \mathbf{I} - \boldsymbol{\tau}) \cdot \mathbf{n} ds \quad (3-50)$$

$$F_i = \frac{d}{dt} \int_{\text{boundaed area}} \rho u_i dA - \int_{\Gamma_0} (\rho u_i u_i + p \delta_{ij} - \tau_{ij}) n_j ds \quad (3-51)$$

3.8.4 Direct forcing method

Another method to calculate the force from the fluid to the structure is the direct forcing approach which was introduced initially by Mohd-Yusof 1997. In this method, after discretization, the force is added to the Navier-Stokes equations. Equation (3-52) describes the semi-discretisation of equation (3-1). Equation (3-52) is explicitly rearranged to find the force, f , with respect to the other parameters. Finally, in this equation u^{n+1} is replaced by V_{solid}^{n+1} (equation (3-53)). In this equation, f represents the force of fluid on the immersed boundary (more details see Mohd-Yusof 1997 and Fadlun et al. 2000)

$$\frac{u^{n+1} - u^n}{\Delta t} + u \cdot \nabla u = -\frac{1}{\rho} \nabla P + \nu \nabla^2 u + f \quad (3-52)$$

$$f = \frac{V_{\text{solid}}^{n+1} - u^n}{\Delta t} + u \cdot \nabla u + \frac{1}{\rho} \nabla P - \partial \nabla^2 u \quad (3-53)$$

3.9 Some related Bench mark studies

The flow around a cylinder has been extensively studied both numerically and experimentally for several decades and several cases have been reviewed by Williamson 1996 and Williamson & Govardhan 2004 and 2008. The flow problem, is sufficiently simple to be analysed in great detail while, it is still retains the physics of more complex flows. Separation of the boundary layer from the surface makes the flow around a cylinder an interesting benchmark for immersed boundary method. In addition, as the main goal for this research is to simulate FSI for cylindrical oil risers, a study of the flow around a 2D cylinder is very relevant. In this section some of the numerical and experimental results which describe the flow field around a moving/stationary cylinder are presented.

Corbalan and de Souza 2010 suggested using an Eulerian method to predict the forcing term which is added to Navier-Stokes equations in the continuous force IB method. To validate and verify the method, four cases have been presented as benchmarks; flow over a stationary cylinder, flow over cylinder with a force oscillation in the transverse direction to the flow, flow over a cylinder forced to oscillate in line with the flow and flow over a cylinder with a forced rotational movement. In all cases the flow was laminar and the amplitude of the oscillation was 0.4 and 0.2 times the cylinder diameter. The frequency of the oscillation was selected to be 0.6 and 1.05 times of frequency of the vortex shedding around a stationary cylinder. The lift and drag forces for the above cases have been reported and compared with were compared the literature.

Choi et al. 2007 proposed a more general IB method that is valid for all Reynolds numbers and can be implemented for various grid topologies. The immersed boundary objects are represented by clouds of structured or unstructured nodes rendered as level sets in the computational domain which can be used to categorise the computational nodes as being in, near and outside of the flow domain. In addition, they have decomposed the velocity near the immersed boundary into a component normal to the IB and a tangential component. The tangential component near the boundary surface is calculated by using a power-law function of the wall normal distance. They also used general interpolation/reconstruction techniques to impose the immersed boundary. Five

different problems were simulated to verify their methods, including the flow over a stationary cylinder and over a cylinder oscillating in line with the flow direction.

3.10 Discussion

In a complete Fluid-Structure interaction simulation the main challenges are to address the complex boundary and large displacement of the immersed boundary. In the previous chapters the physics and importance of these kinds of study were presented. Also the general principles of the methodologies to tackle FSI problems are briefly described. In this chapter the main focus was to explain and compare Immersed boundary methods, their advantages and disadvantages. In additions, several technical issue related to this problem were addressed.

It was briefly discussed, that IB methods were originally based on adding an extra forcing term to the governing equations in order to enforce the boundary conditions. The way in which this source term was defined was the main difference between various versions of the IB methods.

As discussed earlier, in the discrete forcing approach the IB is imposed on the flow domain after the discretization of Navier-Stokes equations. this means that introducing the boundary conditions and forcing functions is not as straightforward as in the continuous forcing approach and depends on the discretization method and its implementation. Also, in the discrete forcing approach the definition of the pressure on the boundary is not as straightforward as in the continuous forcing approach and requires special treatment. The advantages of the discrete forcing approach are that the boundary conditions can be introduced sharply without any extra stability constraint, while the fluid and solid domains are clearly separated and the equations that describe the flow are only solved in the fluid domain.

Cut-cell methods for fluid-structure interaction problems with moving boundaries take significant amount of computational time (Udaykumar et al. 1999, 2001), while the Ghost-Cell approach will create non-physical results when solving the fluid equations in the solid domain.

Fadlun et al. 2000, studied the effect of three interpolation methods in the direct forcing approach for a few different problems. The simulation process has been repeated on various grids and the solution on the finest grid was assumed to be exact. It has been shown that in the “step geometry” (without interpolation) the error decreases slower than first order. Weighting the forcing by the fraction of volume occupied by the structure

better results with a nearly first order behaviour were obtained. The results obtained with a flat boundary showed that the weighting methods underestimate the velocities so the results are not entirely satisfactory. The linear interpolation method was the best among these three and showed a second order accuracy. In the linear interpolation, the velocity profile is assumed to vary linearly very close to the wall and this requires a sufficiently fine grid near the immersed boundary. This issue could be improved by using a local refinement with embedded grids (Kravchenko et al. 1996). However, the benefit and costs of this kind of improvement should be compared with the boundary conforming mesh method. Also Fadlun et al. 2000 claimed that interpolation methods have the same effect on both stationary and prescribed moving boundary problem simulations.

Methods for calculating the hydrodynamical forces from the fluid on the structure were explained as starting points to study the coupling between the fluid and the structure in an FSI simulation. Some of the coupling strategies introduced were used as part of solutions in the literature.

In the final section, some important concepts like fictitious mass and treatment of pressure at immersed boundaries are discussed briefly. Also some of the studies of flow around a cylinder are introduced. This problem will be used as bench mark later in the thesis.

In the next chapter the immersed boundary method based on the interpolation/reconstruction methodology is explained. The focus of the Chapter will be to explain the details of the procedure and the programming in order to address the key points that have been discussed thus far.

Chapter 4. Methodology

Simulating the flow around a moving boundary has been the subject of study during the recent decades. The moving boundary is one of the main issues that need to be solved in order to simulate the flow around a flexible structure. The two main techniques to tackle this problem are: moving grid methods such as the Arbitrary Lagrangian Eulerian (ALE) approach (Donea et al. 1982) and the fixed grid methods, such as the Immersed Boundary (IB) method (Peskin 1972).

ALE methods employ a grid that adapts to, moves and deforms with the moving boundary. Such methods have been applied to study the transient aero-elastic response of airfoils (Farhat et al. 1998), the FSI problem of a shock absorber [Le and Mouro 2001], the blood flow through compliant aortas (Fernandez & Moubachir 2005), etc. A significant limitation of the ALE approach, however, stems from the fact that the mesh conforms to the moving boundary and, as such, needs to be constantly displaced and deformed following the motion of the boundary. The mesh moving step could be quite challenging and expensive for complicated 3D problems. This situation is further exacerbated in problems involving large structural displacements for which frequent remeshing might be the only feasible approach to ensure a well-conditioned mesh at each time step of the simulation. Because of this inherent limitation, the ALE approach is only applicable to FSI problems involving relatively small structural displacements.

In fixed grid approaches, on the other hand, the entire computational domain including both the fluid and structure domains is discretized with a single, fixed, non-boundary conforming grid system. In this case most commonly a Cartesian mesh is used as the fixed background mesh. The effect of a moving immersed boundary is accounted for by adding forcing terms to the governing equations of fluid motion so that the presence of a no-slip boundary at the location of the interface can be felt by the surrounding flow. Because of the fixed grid arrangement, such methods are inherently applicable to FSI problems involving arbitrarily large structural displacements (Borzajani et al. 2008).

The governing equations for a conventional conforming structural grid are discretised in a curvilinear coordinate system to simulate the flow over a complex geometry. The main advantages of this approach are that imposition of boundary conditions is greatly simplified, and furthermore, the solver can be easily designed to maintain adequate accuracy and conservation property. However, depending on the geometrical complexity of the solid boundaries, grid generation and grid quality can be major issues. A multi-block approach may help to divide the complex geometry into simpler geometries. Furthermore, transformation of the governing equations to the curvilinear coordinate system results in a complex system of equations and this complexity can adversely impact the stability, convergence and operation of the solver.

Imposition of a non-grid-aligned solid boundary in a Cartesian grid method can be complicated. The main challenge is to construct a boundary treatment which does not adversely impact the accuracy and conservation properties of the underlying numerical solver. Especially, for viscous flows, an inadequate resolution of the boundary layers which form on the immersed boundaries can reduce the accuracy of the numerical solution (Ye et al. 1999). Immersed boundary methods have also been used successfully for viscous flow computations. However, in most cases (continuous forcing approach) the immersed boundary is distributed across a few cell-widths. This is mainly due to problems associated with representing a point force on a finite size mesh. Similarly, in the so-called volume-of-fluid (VOF) method (Scardovelli and Zaleski 1999), the process of interface reconstruction leads to a non-smooth interface. In contrast to these approaches, in (indirect forcing approach) Cartesian grid methods the boundary is represented by a sharp interface and this has advantages for high Reynolds number flows as well as flows with strong two-way coupling between the flow and the boundary motion.

In this chapter the implementation of an interpolation/reconstruction immersed boundary method (which is a Cartesian grid approach) to simulate flow around a flexible boundary is presented. It is supposed that the flow is two dimensional with low Reynolds number. A fractional step method is used to simplify the governing equations. A finite volume method with staggered variable arrangement in uniform Cartesian mesh has been used to discretize the Navier-Stokes equations.

The governing equations, discretisation, computational grid, interpolation procedure and algorithm of the code are explained in detail, together with the calculation of the lift and drag coefficients. In the immersed boundary method, the fluid grids in the vicinity of the structure's boundary which have at least one neighbour in the structural node

should be identified and this depends on the type of discretization used for the governing fluid equation. Therefore, in the following section a brief description of the derivation of the Navier-Stokes equations and its discretisation procedure is presented. Also, Navier-Stokes discretised equations are used later in the calculation of the pressure boundary condition for the pressure Poisson equation in the Chapter 7.

4.1 Governing equation

The derivation of the Navier–Stokes equations begins with an application of *Newton's second law* and conservation of momentum is enforced for an arbitrary portion of the fluid. In an *inertial frame of reference*, the general form of the equations of fluid motion is (Batchelor 1967):

$$\rho \left(\frac{\partial V}{\partial t} + V \cdot \nabla V \right) = -\nabla p + \nabla \cdot S + f \quad (4-1)$$

Where V is the flow velocity, ρ is the fluid density, p is the pressure, S is the stress tensor and f represents body forces enforced on the fluid to simulate boundary conditions. The above relation represents conservation of momentum in a fluid and is an application of Newton's second law to a continuum. In fact, this equation is applicable to any non-relativistic continuum and is known as the *Cauchy momentum equation* (Batchelor 1967).

The effect of *stress* in the fluid is represented by the ∇p and $\nabla \cdot S$ terms; these are gradients of surface forces, similar to the definition of stresses in a solid. ∇p is called the pressure gradient and arises from the isotropic part of the stress tensor. This part corresponds to the normal stress that is present in almost all situations. The anisotropic part of the stress tensor gives rise to $\nabla \cdot S$, which conventionally describes the viscous forces. For incompressible flows, there is only a shear effect and hence, T is the deviatoric stress tensor, so that the stress tensor σ is defined as (Batchelor 1967):

$$\sigma = -pl + S \quad (4-2)$$

The stress terms p and T are unknown, so the general form of the equations of motion is not applicable to solve problems. A force model is needed in the equations of motion to relate these stresses to the fluid motion (Feynman et al. 1963). few assumptions on the specific behaviour of a fluid are applied in order to specify the stresses in terms of other flow variables, such as velocity and density. Batchelor 1967

explained the assumptions on the deviatoric stress tensor S which is needed to obtain the Navier-Stokes equations.

Equation (4-3) presents the governing equation for an unsteady, incompressible fluid flow in vector form (Navier–Stokes equation).

$$\rho \left(\frac{\partial V}{\partial t} + V \cdot \nabla V \right) = -\nabla p + \mu \nabla^2 V + f \quad (4-3)$$

In the above equation $\frac{\partial V}{\partial t}$ is the unsteady acceleration, $V \cdot \nabla V$ is convection acceleration, $-\nabla p$ is the pressure gradient. $\mu \nabla^2 V$ implies that viscosity operates by *diffusion of momentum* of a Newtonian fluid, and f is a body force (force per unit volume), such as gravity or centrifugal force.

Note that only the convective terms are nonlinear for an incompressible Newtonian flow. The convective acceleration is an acceleration caused by a (possibly steady) change in velocity over *position*.

The incompressible Navier-stokes equations in a 2D Cartesian domain is defined as:

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) = -\frac{\partial p}{\partial x} + \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \rho g_x \quad (4-4)$$

$$\rho \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) = -\frac{\partial p}{\partial y} + \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + \rho g_y \quad (4-5)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (4-6)$$

4.2 Non-dimensional governing equation

For reasons of simplicity and easiness of generalization of the solution algorithm the non-dimensional form of the Navier-Stokes equation is used. When the Navier Stokes equation is presented using primitive variables, the following definitions are used to obtain the non-dimensional equations.

$$t^* = \frac{t V}{D}, \quad x^* = \frac{x}{D}, \quad y^* = \frac{y}{D}, \quad p^* = \frac{P}{\rho V^2}, \quad u^* = \frac{u}{V}, \quad v^* = \frac{v}{V}, \quad (4-7)$$

$$Re = \frac{VD}{\nu}, \quad g^* = \frac{gD}{\rho V^2}$$

The general form of the non-dimensional Navier-Stokes equation is given in equation (4-8). The external force used here is the gravity, g^* , though other volume forces might also be added.

$$\frac{\partial V^*}{\partial t^*} + V^* \cdot \nabla^* V^* = -\nabla^* p^* + \frac{1}{Re} \nabla^{*2} V^* + g^* \quad (4-8)$$

In this equation the ‘*’ identifies the non-dimensional variables. It is omitted from the equations later in the text.

4.3 Discretization method

To ensure the conservation of momentum by the discretization of convection, the convective term in the momentum equation is written in conservative form before discretizing. As shown in equations (4-9) and (4-10), this is equivalent to the non-conservative form.

$$\nabla \cdot VV = V \cdot \nabla V \quad (4-9)$$

$$\begin{aligned} \frac{\partial u^2}{\partial x} + \frac{\partial vu}{\partial y} &= 2u \frac{\partial u}{\partial x} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + u \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \\ &= u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \end{aligned} \quad (4-10)$$

Applying the same notation in the y direction, the non-dimensional Navier-Stokes equations become:

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial vu}{\partial y} = -\frac{\partial p}{\partial x} + 1/Re \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + g_x \quad (4-11)$$

$$\frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + 1/Re \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + g_y \quad (4-12)$$

4.3.1 Staggered arrangement

The first issue is to identify the points in the domain at which the values of the unknown dependent variables have to be computed. The obvious choice is to store all the variables at the centre of the control volumes; such an arrangement is called a collocated variable arrangement. Since many of the terms in each of the equations are essentially identical, the number of coefficients that must be computed and stored is minimized and the programming is simplified by this choice. However, there is no need for all the variables to share the same grid; a different arrangement may turn out to be advantageous. In Cartesian coordinate, the staggered arrangement introduced by Harlow and Welsh 1965 offers some advantages over the collocated arrangement. Several terms that require interpolation with the collocated arrangement, can be calculated (to a second-order approximation) without interpolation.

Typical staggered control volumes are shown in Figure 4-1. The control volume for the u_x and u_y are displaced with respect to the control volume for the continuity equation. Both the pressure and diffusion terms are naturally approximated by central difference approximations without interpolation, since the pressure nodes lie at CV face centres and the velocity derivatives needed for the diffusive terms are readily computed at the CV faces. In addition, the mass fluxes in the continuity equation at the faces of a pressure CV can be directly calculated.

The biggest advantage of the staggered arrangement is the strong coupling between the velocities and the pressure, which helps to avoid certain convergence problems and a decoupling of the pressure and velocity fields.

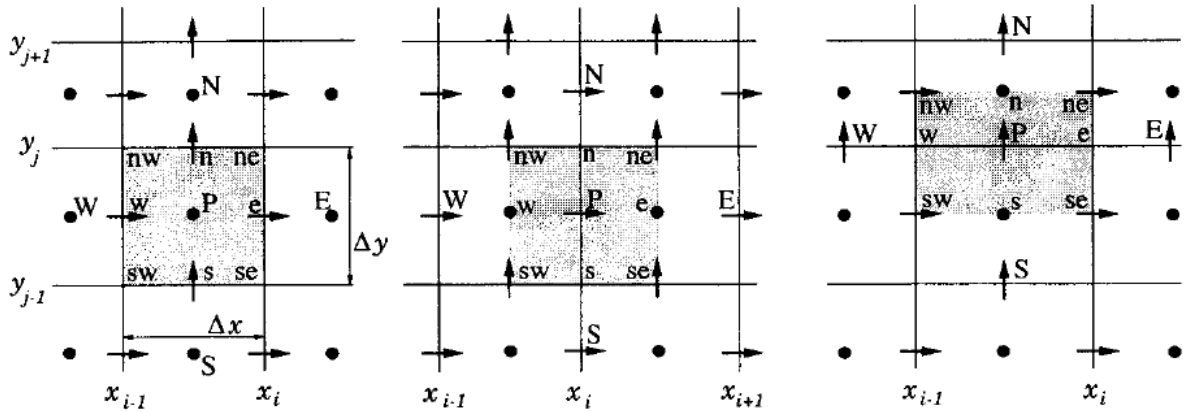


Figure 4-1: Control volumes for a staggered grid: for mass conservation and scalar quantities (left), for x-momentum (centre) and for y-momentum (right)

4.3.2 Discretization of the momentum equation

The cells are numbered using indices i and j which identify cell centre positions along the horizontal and vertical directions, respectively. Cell boundary positions are labelled with half-integer values for the indices. According to Figure 4-2 each parts of the x-momentum equation is discretized about the point $(i+1/2, j)$ as follows:

$$\frac{\partial u^2}{\partial x} = \frac{\left[\frac{1}{2} \left(u_{i+\frac{3}{2},j} + u_{i+\frac{1}{2},j} \right) \right]^2 - \left[\frac{1}{2} \left(u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j} \right) \right]^2}{(x_{i+1,j} - x_{i,j})} \quad \text{or} \quad (4-13)$$

$$\frac{\partial u^2}{\partial x} = \frac{(u_{i+1,j})^2 - (u_{i,j})^2}{2(x_{i+1,j} - x_{i,j})} \quad \text{where } u_{i,j} = \frac{1}{2} \left(u_{i-\frac{1}{2},j} + u_{i+\frac{1}{2},j} \right) \quad (4-14)$$

$$\frac{\partial uv}{\partial y} = \frac{\frac{1}{2} \left(u_{i+\frac{1}{2},j+1} + u_{i+\frac{1}{2},j} \right) \left[\frac{1}{2} \left(v_{i+1,j+\frac{1}{2}} + v_{i,j+\frac{1}{2}} \right) \right] - \frac{1}{2} \left(u_{i+\frac{1}{2},j-1} + u_{i+\frac{1}{2},j} \right) \left[\frac{1}{2} \left(v_{i+1,j-\frac{1}{2}} + v_{i,j-\frac{1}{2}} \right) \right]}{\left(y_{i+\frac{1}{2},j+\frac{1}{2}} - y_{i+\frac{1}{2},j-\frac{1}{2}} \right)} \quad (4-15)$$

$$\text{Or } \frac{\partial uv}{\partial y} = \frac{\left(u_{i+\frac{1}{2},j-\frac{1}{2}}\right)\left(v_{i+\frac{1}{2},j-\frac{1}{2}}\right) - \left(u_{i+\frac{1}{2},j+\frac{1}{2}}\right)\left(v_{i+\frac{1}{2},j+\frac{1}{2}}\right)}{\left(y_{i+\frac{1}{2},j+\frac{1}{2}} - y_{i+\frac{1}{2},j-\frac{1}{2}}\right)} \quad (4-16)$$

$$\text{Where } u_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}\left(u_{i+\frac{1}{2},j+1} + u_{i+\frac{1}{2},j}\right) \text{ and } v_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}\left(v_{i+1,j+\frac{1}{2}} + v_{i,j+\frac{1}{2}}\right) \quad (4-17)$$

$$\frac{\partial p}{\partial x} = \frac{p_{i+1,j} - p_{i,j}}{\left(x_{i+1,j} - x_{i,j}\right)} \quad (4-18)$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+\frac{3}{2},j} - 2u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}}{\left(x_{i+1,j} - x_{i,j}\right)^2} \quad (4-19)$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{u_{i+\frac{1}{2},j+1} - 2u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j-1}}{\left(y_{i+\frac{1}{2},j+1} - y_{i+\frac{1}{2},j}\right)^2} \quad (4-20)$$

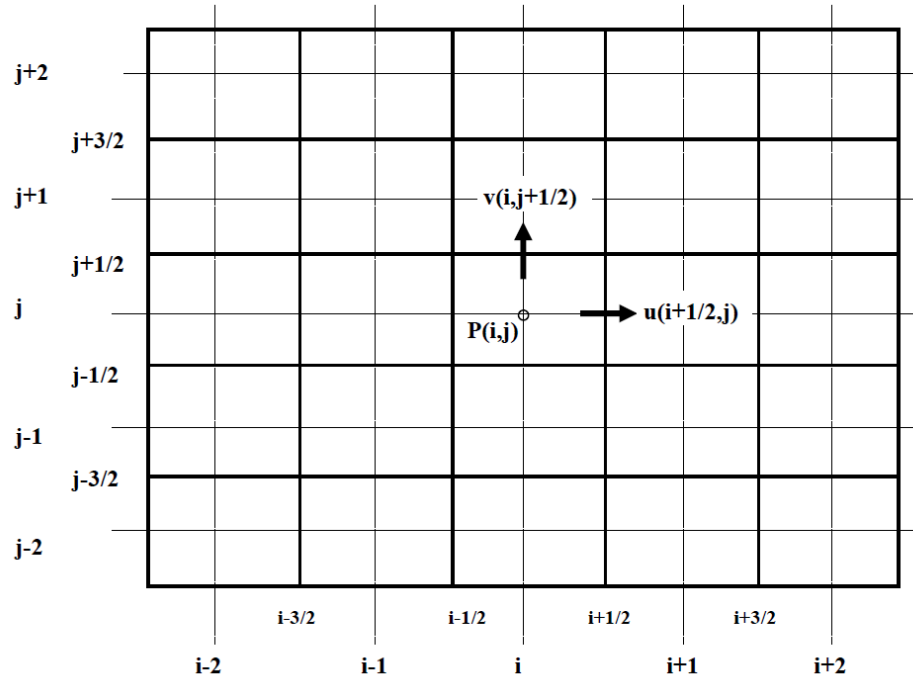


Figure 4-2: staggered arrangement used for discretization

In addition to the space index subscripts, there is a superscript for the number of time cycles. For instant $u_{i+\frac{1}{2},j}^{(n+1)}$ shows the horizontal velocity at the time $t = (n + 1)\delta t$, in which the δt is the time increment per cycle. When there is no superscript, it is correspond to the value of the parameter at time $t = n\delta t$.

$$\frac{\partial u}{\partial t} = \frac{u_{i+\frac{1}{2},j}^{n+1} - u_{i+\frac{1}{2},j}^n}{\delta t} \quad (4-21)$$

4.3.3 Fractional step method

To solve the Navier-Stokes equations a splitting method is used. In the first part of the solution an intermediate velocity is calculated by updating the velocity in time by taking into account only convection and diffusion terms. The results of this stage are updated by enforcing the Poisson equation for the pressure.

Equations (4-22) and ((4-24) show how the convective and diffusive term is used to calculated intermediate velocity \hat{u} and \hat{v} in x and y direction respectively.

$$\begin{aligned} \frac{\hat{u}_{i+\frac{1}{2},j}^{n+1} - u_{i+\frac{1}{2},j}^n}{\partial t} &= \frac{(\mathbf{u}_{i,j})^2 - (\mathbf{u}_{i+1,j})^2}{(x_{i,j} - x_{i+1,j})} & (4-22) \\ &+ \frac{\left(u_{i+\frac{1}{2},j-\frac{1}{2}}\right)\left(v_{i+\frac{1}{2},j-\frac{1}{2}}\right) - \left(u_{i+\frac{1}{2},j+\frac{1}{2}}\right)\left(v_{i+\frac{1}{2},j+\frac{1}{2}}\right)}{\left(y_{i+\frac{1}{2},j-\frac{1}{2}} - y_{i+\frac{1}{2},j+\frac{1}{2}}\right)} \\ &+ v \left(\frac{u_{i+\frac{3}{2},j} - 2u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}}{(x_{i+1,j} - x_{i,j})^2} \right. \\ &\left. + \frac{u_{i+\frac{1}{2},j+1} - 2u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j-1}}{(y_{i+\frac{1}{2},j+1} - y_{i+\frac{1}{2},j})^2} \right) + g_x \end{aligned}$$

$$\begin{aligned} \frac{\hat{v}_{i,j+\frac{1}{2}}^{n+1} - v_{i,j+\frac{1}{2}}^n}{\partial t} &= \frac{\left(u_{i-\frac{1}{2},j+\frac{1}{2}}\right)\left(v_{i-\frac{1}{2},j+\frac{1}{2}}\right) - \left(u_{i+\frac{1}{2},j+\frac{1}{2}}\right)\left(v_{i+\frac{1}{2},j+\frac{1}{2}}\right)}{\left(x_{i-\frac{1}{2},j+\frac{1}{2}} - x_{i+\frac{1}{2},j+\frac{1}{2}}\right)} & (4-23) \\ &+ \frac{(\mathbf{v}_{i,j})^2 - (\mathbf{v}_{i,j+1})^2}{(y_{i,j} - y_{i,j+1})} \\ &+ v \left(\frac{v_{i+1,j+\frac{1}{2}} - 2v_{i,j+\frac{1}{2}} + v_{i-1,j+\frac{1}{2}}}{(x_{i+1,j} - x_{i,j})^2} \right. \\ &\left. + \frac{v_{i,j+\frac{3}{2}} - 2v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}}}{(y_{i+\frac{1}{2},j+1} - y_{i+\frac{1}{2},j})^2} \right) + g_y \end{aligned}$$

For the velocities values which are not centred at points indicated in the mesh diagram, an average of adjacent values is applied. For example:

$$u_{i,j} = \frac{1}{2} \left(u_{i-\frac{1}{2},j} + u_{i+\frac{1}{2},j} \right) \quad \text{and} \quad v_{i,j} = \frac{1}{2} \left(v_{i,j-\frac{1}{2}} + v_{i,j+\frac{1}{2}} \right) \quad (4-24)$$

$$u_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2} \left(u_{i+\frac{1}{2},j+1} + u_{i+\frac{1}{2},j} \right) \quad \text{and} \quad v_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2} \left(v_{i+1,j+\frac{1}{2}} + v_{i,j+\frac{1}{2}} \right) \quad (4-25)$$

In this research equations (4-22) and (4-23) are solved by using 3rd order Range-Kutta method to calculate the intermediate velocities.

In the second stage of the fractional step method the intermediate velocities from Equations (4-22) and (4-23) are updated by adding the effect of pressure.

$$\frac{u_{i+\frac{1}{2},j}^{n+1} - \hat{u}_{i+\frac{1}{2},j}^n}{\partial t} = \frac{p_{i+1} - p_i}{x_{i+1} - x_i} \quad (4-26)$$

$$\frac{v_{i+\frac{1}{2},j}^{n+1} - \hat{v}_{i+\frac{1}{2},j}^n}{\partial t} = \frac{p_{j+1} - p_j}{x_{j+1} - x_j} \quad (4-27)$$

The calculation of pressure equation is discussed in the next part.

4.3.4 Calculation of pressure

The solution of the incompressible Navier-Stokes equations is complicated by the lack of an independent equation for the pressure, whose gradient contributes to momentum equations. One way to overcome this issue is to construct an equation for the pressure field to guarantee satisfaction of the continuity equation (Ferziger and Peric 2002).

The form of the continuity equation suggests that if the divergence of the momentum equation is taken, then the continuity equation could be used to simplify the resulting terms, which leads to a Poisson equation for the pressure. The procedure is as follow:

Taking the divergence from the general Cartesian form of Navier-Stokes (equation (4-3)) or from the non-dimensional form (equation (4-8)) will lead to:

$$\nabla \cdot \left(\frac{\partial V}{\partial t} + V \cdot \nabla V \right) = \nabla \cdot \left(-\nabla P + \frac{1}{Re} \nabla^2 V + g \right) \quad (4-28)$$

While in the indices form the equation will look like as:

$$\frac{\partial}{\partial x_i} \left(\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} \right) = \frac{\partial}{\partial x_i} \left(-\frac{\partial p}{\partial x_i} + 1/Re \frac{\partial^2 u_i}{\partial x_j \partial x_j} + g_{x_i} \right) \quad (4-29)$$

Transferring the divergence of the pressure gradient to the left side of the equation:

$$\frac{\partial}{\partial x_i} \left(\frac{\partial p}{\partial x_i} \right) = -\frac{\partial}{\partial x_i} \left(\frac{\partial u_i u_j}{\partial x_j} - 1/Re \frac{\partial^2 u_i}{\partial x_j \partial x_j} - g_{x_i} + \frac{\partial u_i}{\partial t} \right) \quad (4-30)$$

It is possible to simplify the above equation more, as the viscous and unsteady terms disappear by applying the continuity equation:

$$\frac{\partial}{\partial x_i} \left(\frac{\partial p}{\partial x_i} \right) = - \frac{\partial}{\partial x_i} \left(\frac{\partial u_i u_j}{\partial x_j} \right) \quad (4-31)$$

In some research even the temporal and viscous terms are not omitted depending on the accuracy of the results and in what order the continuity equation is forced in previous time steps. The above pressure equation (equation (4-31)) can be solved by one of the numerical methods for elliptic equations. In the pressure equation, the right hand side is a sum of the derivatives of terms in the momentum equations; these terms must be approximated in the same way as the momentum equation. To maintain the consistency among the approximation used, it is best to derive the pressure equation from the discretized momentum equation. From equations (4-28) and (4-29) one can obtain:

$$\frac{D_{i,j}^{n+1} - D_{i,j}^n}{\partial t} = -Q_{i,j} - \frac{p_{i+1,j} + p_{i-1,j} - 2p_{i,j}}{\partial x^2} - \frac{p_{i,j+1} + p_{i,j-1} - 2p_{i,j}}{\partial y^2} \quad (4-32)$$

$$+ 1/Re \left(\frac{D_{i+1,j} + D_{i-1,j} - 2D_{i,j}}{\partial x^2} - \frac{D_{i,j+1} + D_{i,j-1} - 2D_{i,j}}{\partial y^2} \right)$$

$$D_{i,j} = \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\partial x} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\partial y} \quad (4-33)$$

$$Q_{i,j} = \frac{(u_{i+1,j})^2 + (u_{i-1,j})^2 - 2(u_{i,j})^2}{\partial x^2} - \frac{(v_{i+1,j})^2 + (v_{i-1,j})^2 - 2(v_{i,j})^2}{\partial y^2} \quad (4-34)$$

$$+ \frac{2}{\partial x \partial y} \left[\left(u_{i+\frac{1}{2},j+\frac{1}{2}} \right) \left(v_{i+\frac{1}{2},j+\frac{1}{2}} \right) + \left(u_{i-\frac{1}{2},j-\frac{1}{2}} \right) \left(v_{i-\frac{1}{2},j-\frac{1}{2}} \right) \right]$$

$$- \left(u_{i+\frac{1}{2},j-\frac{1}{2}} \right) \left(v_{i+\frac{1}{2},j-\frac{1}{2}} \right) - \left(u_{i-\frac{1}{2},j+\frac{1}{2}} \right) \left(v_{i-\frac{1}{2},j+\frac{1}{2}} \right) \Big]$$

The procedure for determining the pressure is based on the requirement that $D_{i,j}^{n+1}$ vanishes for every cell at the end of the time cycle. This assumption leads to the equation for the pressure:

$$\frac{p_{i+1,j} + p_{i-1,j} - 2p_{i,j}}{\partial x^2} - \frac{p_{i,j+1} + p_{i,j-1} - 2p_{i,j}}{\partial y^2} = -R_{i,j} \quad (4-35)$$

Where $R_{i,j}$ will be:

$$R_{i,j} = Q_{i,j} - \frac{D_{i,j}}{\partial t} - \nu \left(\frac{D_{i+1,j} - D_{i-1,j} - 2D_{i,j}}{\partial x^2} - \frac{D_{i,j+1} - D_{i,j-1} - 2D_{i,j}}{\partial y^2} \right) \quad (4-36)$$

In principal, it is possible to use $Q_{i,j}$ instead of $R_{i,j}$ in the equation (4-35), since $D_{i,j}$ should be zero in previous time steps. However, in practice the use of $R_{i,j}$ is desirable so that equation (4-35) does not have to be solved extremely accurately in order to keep the accumulative error in the divergence to a sufficiently low level. Harlow and Welch 1965 reported that with a very stringent convergence requirement, the cumulative results of calculation are independent of using $D_{i,j}$ or $R_{i,j}$ in equation (4-35).

Equations equations (4-22) to (4-27), (4-35) and (4-36) are the main equations used in performing the calculation of the flow parameters. $R_{i,j}$ is computed for every cell, using the velocities available at the beginning of the cycle using equations (4-26) and (4-27). Secondly $p_{i,j}$ is calculated using equation (4-35). Finally the intermediate velocities that are calculated from equations (4-22) and (4-23) are updated by inserting the new pressure in equations (4-26) and (4-27). The process will be continued in time.

4.3.5 Mesh generation

One of the advantages of using immersed boundary methods is the use of simple Cartesian mesh generation. In this approach regardless of the location of boundary a structured grid is created to cover the entire computational domain, including possible solid objects inserted in the flow domain. In this research staggered grid arrangement is used. In Figure 4-3, the computational grid is shown by the black lines with coordinates $x_{coord}(i)$ and $y_{coord}(j)$ in x and y direction, respectively. The blue lines are passing through the centre of the computational cells. These coordinates are stored in the $x_{crd}(i)$ and $y_{crd}(j)$ arrays in x and y direction respectively. Also, as the staggered variable arrangement is used, in order to define the boundary conditions it was necessary to define the blue line beyond the computational grid, effectively introducing “ghost” or virtual grid points. The calculation however is just limited to the main area. Later in Chapter 5, a special mesh is used which is finer around the solid boundary which becomes coarser towards the outside in order to limit the number of mesh points.

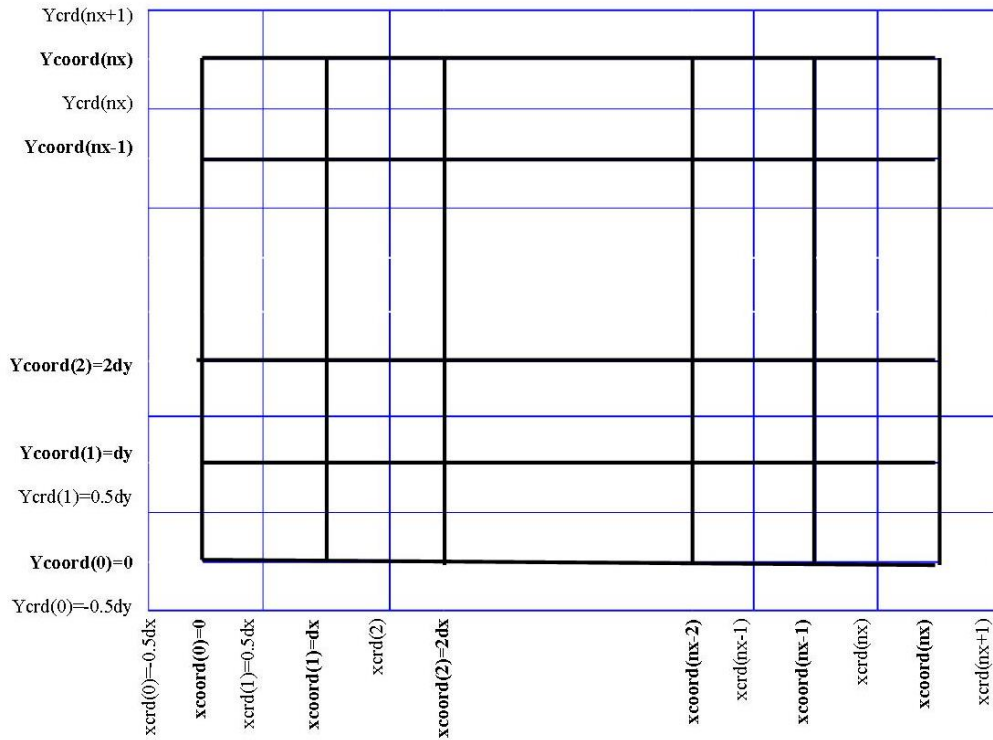


Figure 4-3 : Uniform staggered mesh coordinate

4.3.6 Location of velocities and pressure

In the collocated arrangement the pressure and velocities are defined at the centre of the grid cells. However, in the staggered arrangement the pressures and the velocities are not defined at the same locations. According to the Figure 4-4 the pressures are defined at the cell centres where the lines $x_{crd}(i)$ and $y_{crd}(j)$ are intersecting. On the other hand the velocity in x direction, $u(i,j)$ is introduced at the intersections of the $x_{coord}(i)$ and $y_{crd}(j)$ lines and the velocity in y directions is defined at the intersections of $y_{coord}(j)$ and $x_{crd}(i)$.

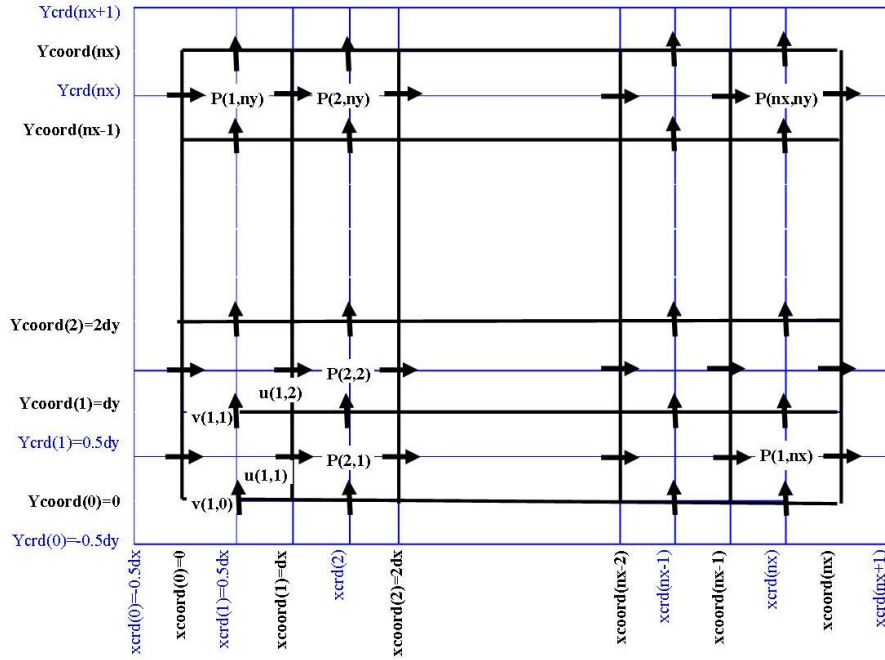


Figure 4-4: velocities and pressure positions in a staggered arrangement

4.4 Boundary conditions

As the staggered arrangement has been used for the discretization of the governing equations, the definition of the boundary condition should match this arrangement. For the inlet, outlet, top and bottom, uniform velocity, convective outflow and symmetry boundary conditions for the velocity have been used respectively. According to the IB procedure when the solid boundary is not aligned with the background grid the definition of the no-slip condition at the immersed boundary (here the cylinder wall) becomes cumbersome. The definition of the boundary conditions are detailed in the following sections.

4.4.1 Inlet

At the inlet, it is straight forward to introduce the velocity in the x direction, u , as this velocity is defined on the cell boundary, $xcoord(0)$, which is the first line of the computational grid. According to the Figure 4-5, $u(0,1)$, $u(0,2)$... and $u(0,ny)$ (or generally $u(0,j)$ $j=1,2,...ny$) has been defined as the inlet velocity in x direction (green arrows in Figure 4-5). However the inlet velocity in the y direction, v , cannot be defined directly due to the staggered arrangement of the variables. To resolve the issue, the velocities $v(0,j)$ and $v(1,j)$ are defined in a way that the average of these two velocities, corresponds to the actual v -velocity at the inlet. $(v(0,0)+v(1,0))/2$ or $(v(0,1)+v(1,1))/2$

.... And $(v(0,ny)+v(1,ny))/2$. For the special case of the zero inlet v-velocity $v(0,j) = -v(1,j)$ is defined (the red arrows in Figure 4-5).

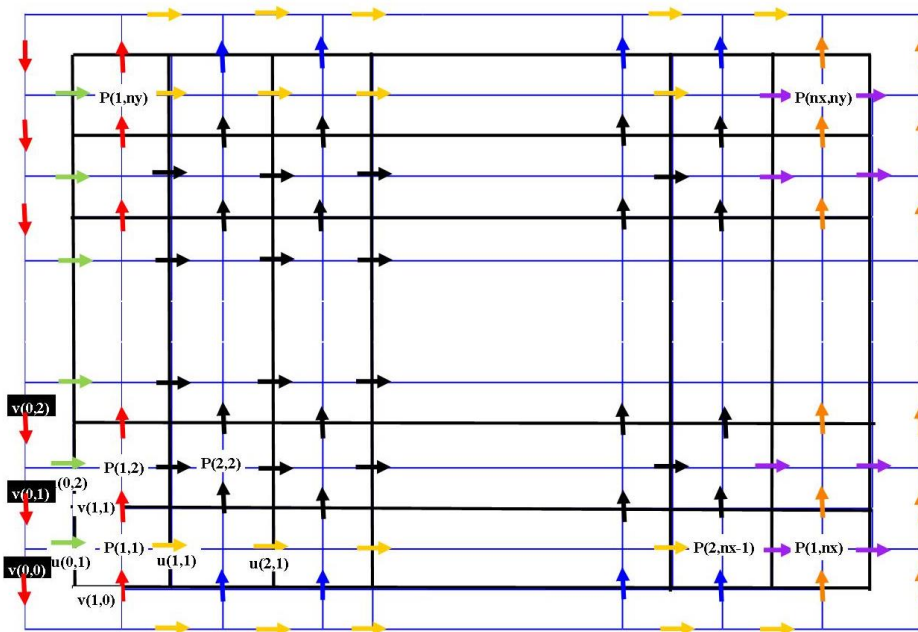


Figure 4-5 Staggered arrangement – bold lines are cell boundaries which velocities are calculated, and pressure are calculated on intersection of light lines. Velocities in y direction need to be interpolated for inlet. Velocities in x direction are specified directly on the boundaries.

4.4.2 Outlet

As the velocities and the pressure are not known at the exit and the computational domain must be finite, according to the Orlanski 1976 the convective outflow boundary conditions are applied for each velocity flux component. The location of the outflow boundary must be sufficiently downstream of the immersed object and the recirculation from the IB should not be present and the streamlines should be parallel. Also at the outflow boundary:

$$\frac{\partial u}{\partial y} = \frac{\partial v}{\partial y} = 0 \quad \text{and} \quad \frac{\partial p}{\partial n} = 0 \quad (4-37)$$

Equations (4-38) and (4-39) present a simplified version of the unsteady convective boundary conditions in the staggered arrangement. In these equations, U_{con}^* is the convection velocity at the outlet and it is assumed to be a constant value. In Figure 4-5 the purple arrows and orange arrows are the u and v velocities that are used to implement the outflow boundary conditions in x and y directions respectively.

$$\frac{(u_{(nx,j)}^{n+1} - u_{(nx,j)}^n)}{dt} = U_{con}^* \frac{(u_{(nx,j)}^n - u_{(nx-1,j)}^n)}{dx} \quad (4-38)$$

$$\frac{(v_{(nx+1,j)}^{n+1} - v_{(nx+1,j)}^n)}{dt} = U_{con}^* \frac{(v_{(nx+1,j)}^n - v_{(nx,j)}^n)}{dx} \quad (4-39)$$

4.4.3 Symmetry boundary condition

For the top and the bottom boundaries the symmetry condition has been used. ie. no flows passes across the boundaries. This implies that the normal velocities are set to zero and the normal gradient of the u velocity is assumed to be zero as well. In Figure 4-5, the blue arrows represent the velocities in the y direction which are set to zero at the boundary (Dirichelt boundary conditions) and the yellow arrows define the location of the Neumann boundary condition for the velocity in x direction. Equations define the symmetry boundary conditions that are applied for the staggered arrangement.

$$v_{i,ny} = 0 \quad u_{i,ny+1} = u_{i,ny} \quad (4-40)$$

$$v_{i,0} = 0 \quad u_{i,0} = u_{i,1} \quad (4-41)$$

4.4.4 Solid boundary not conforming mesh (immersed boundary)

It has been mentioned earlier that the use of Cartesian coordinates may result in a mesh that is not aligned with the solid boundaries. Solid boundaries could cut the grid cells which complicates the implementation of the boundary conditions. For instance, it is not always possible to apply no slip boundary conditions directly at the walls of a solid. To resolve this issue different methods are used to introduce a solid boundary to the fluid flow. This notion is the main subject of the immersed boundary methods and has been addressed in the chapter 3.

In this part, the procedure to define the boundary conditions around the non-conforming solid boundaries is briefly discussed. Firstly, a Cartesian mesh is defined for the whole of the fluid domain regardless of the location of an immersed solid, see Figure 4-6 left.

The presence of the solid boundary is introduced to the flow solver by using an interpolation immersed boundary method. As shown in Figure 4-6 right, to update each velocity component in the CFD solver, 8 neighbouring velocities located around that specific velocity are needed for the discretization of the Navier-Stokes equations on a staggered grid.

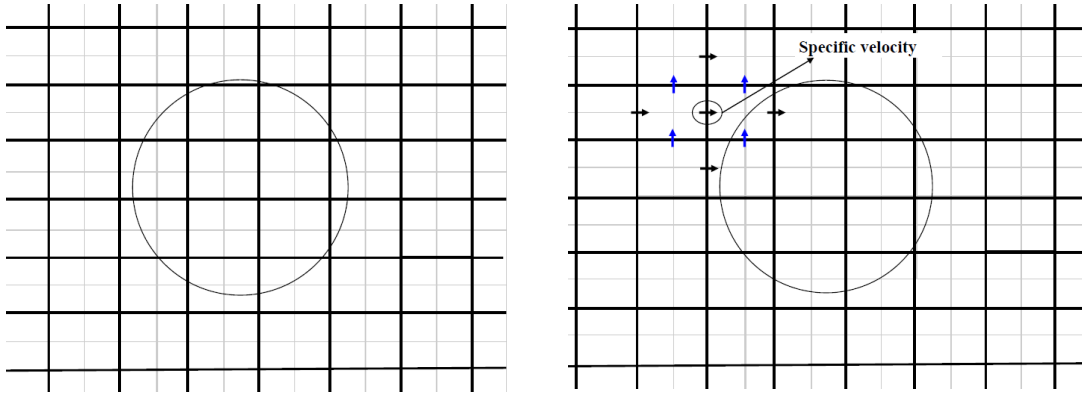


Figure 4-6 Left, a part of domain with not conforming Cartesian mesh, regardless of solid existence. Right, A specific velocity with its 8 velocities around necessary for its calculation.

In this figure, it is clearly shown that two out of eight neighbouring velocities are located inside the solid. The flow solver cannot update this specific velocity automatically and specific treatment is needed. In these cases the governing equations are replaced with interpolation equations that use velocities at the wall of the solid and neighbouring velocities located in the flow field.

Figure 4-7, shows all the velocities in the x and y directions (u and v components) which need to be interpolated inside the fluid domain. All of these velocities cannot be calculated automatically by the governing equation as at least one out of the eight neighbouring velocities components are located inside the solid.

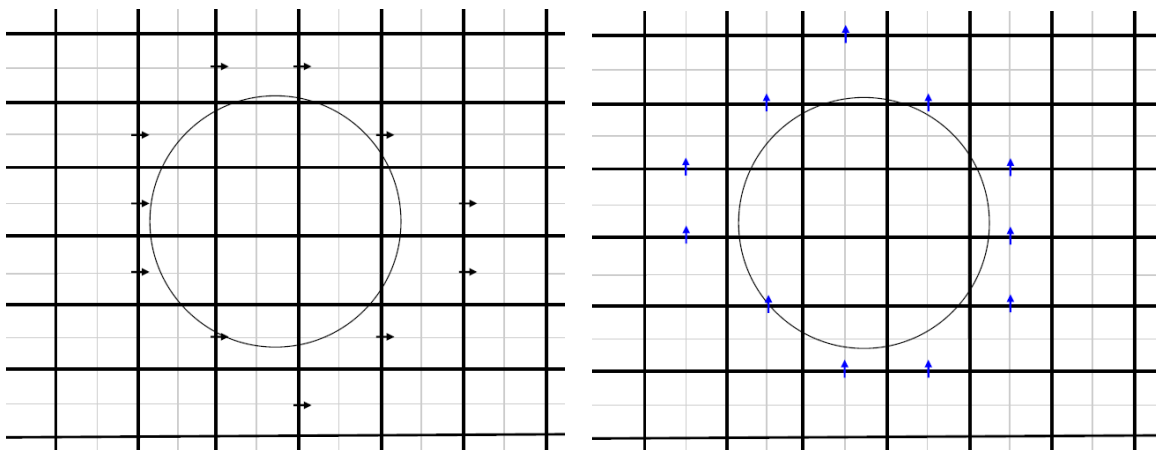


Figure 4-7: A 2D Cartesian grid with staggered arrangement, left: u velocities needed to be interpolated near immersed boundary. Right: v velocities needed to be interpolated near the immersed boundary.

Interpolation equations are formulated for all “boundary” velocities in the flow domain that require interpolation.

Interpolation is implemented in the direction perpendicular to the solid boundary, unlike some of interpolation methods presented in the literature. To simplify the interpolation procedure, the solid boundary is locally assumed to be a circular cylinder. Perpendicular lines cut the cylinder on one side and grid lines on the other side. Figure 4-8, shows two possible interpolation scenarios to interpolate $u_{i,j}$. In Figure 4-8, the velocity, u_2 , is interpolated using $u_{i-1,j}$ and $u_{i-1,j+1}$. Then $u_{i,j}$ is interpolated using u_1 (on the cylinder wall) and u_2 . This procedure will be repeated for all the u and v boundary velocities that are presented on Figure 4-7.

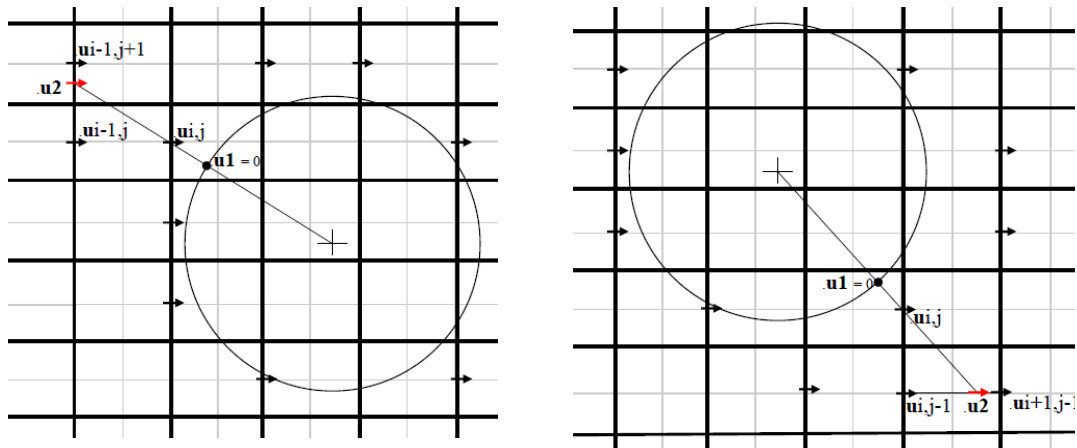


Figure 4-8: interpolation method for the velocity near the boundary in two different scenarios. $u_{i,j}$ has been interpolated between $u_1=0$ on the boundary and u_2 .

It is assumed that the normal pressure gradient is nearly zero ($\frac{\partial p}{\partial n} \approx 0$) near the stationary (or moving with constant velocity) immersed boundary (if the IB has acceleration, $\frac{\partial p}{\partial n} \neq 0$) therefore pressure is not extrapolated to the immersed boundary. On the other hand, the pressure of 4 locations is used in the staggered arrangement to update the pressure inside the fluid governing equation in CFD solver (Pressure Poisson equation). As shown in Figure 4-9 left, the value of $P_{i,j}$ depend on $P_{i,j+1}$, $P_{i,j-1}$, $P_{i+1,j}$ and $P_{i-1,j}$. If any of these four points were inside of the solid boundary, they are assumed to be the same value as $P_{i,j}$ ($\frac{\partial p}{\partial n} = 0$). In the case of the moving IB (with acceleration) the pressure gradient is calculated by projecting the differential form of the momentum equation perpendicular to the boundary (see chapter 7).

In Figure 4-9 right, shaded cells are the cells in which the pressures are updated using governing equations. On the Figure 4-9 left, although pressures on the shaded areas are updated using governing equations, they have a neighbour of which their pressure value is not explicitly updated.

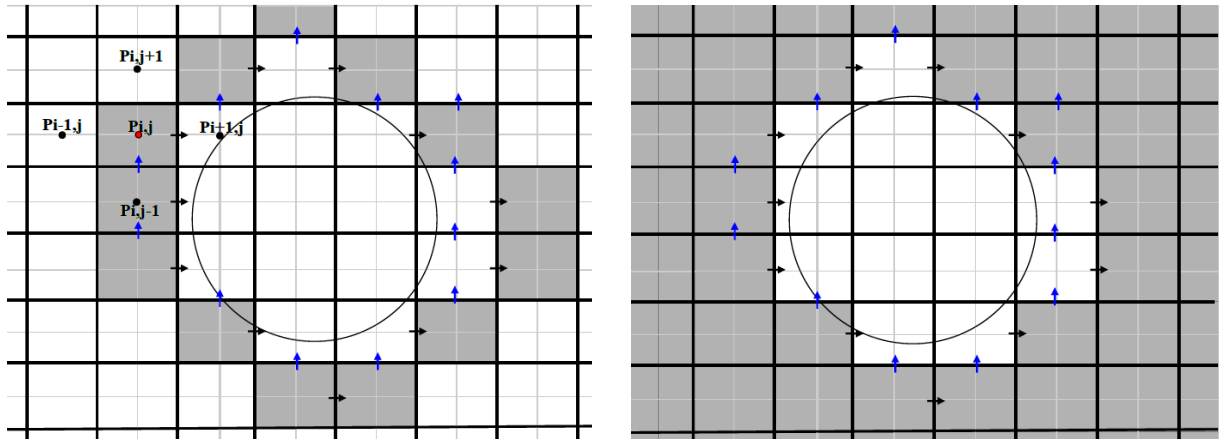


Figure 4-9: Right, shaded area shows the cells in which the pressure is updated in the CFD solver. Left, cells with at least one immersed boundary pressure points is shown.

In the next section the algorithm of the code is explained briefly and the flow chart of the program is presented.

4.5 Solving procedure

The solution algorithm consists of four main parts.

- At the beginning a simple Cartesian grid is created as a computational domain. Three attributes, *umask*, *vmask* and *pmask* are defined for *u* and *v* velocities and pressure respectively at entire domain. These attributes are zero for the cells of the domain that are covered by the immersed solid (they are not directly updated by governing equations). The interface cells which were not updated by governing equations are categorised and the interpolation coefficients for the velocity component are calculated (in the “Ingrid” part of the following flowchart). Also, the initial condition and constant parameters are defined at this stage (in the “init” part of the algorithm). The boundary conditions are implemented in the “bounds” algorithm. The interpolation formulas are applied to the governing equations as boundary conditions. In addition the discretised equations matrix is decomposed using an incomplete LU decomposition algorithm in the “inisol” section of the program.

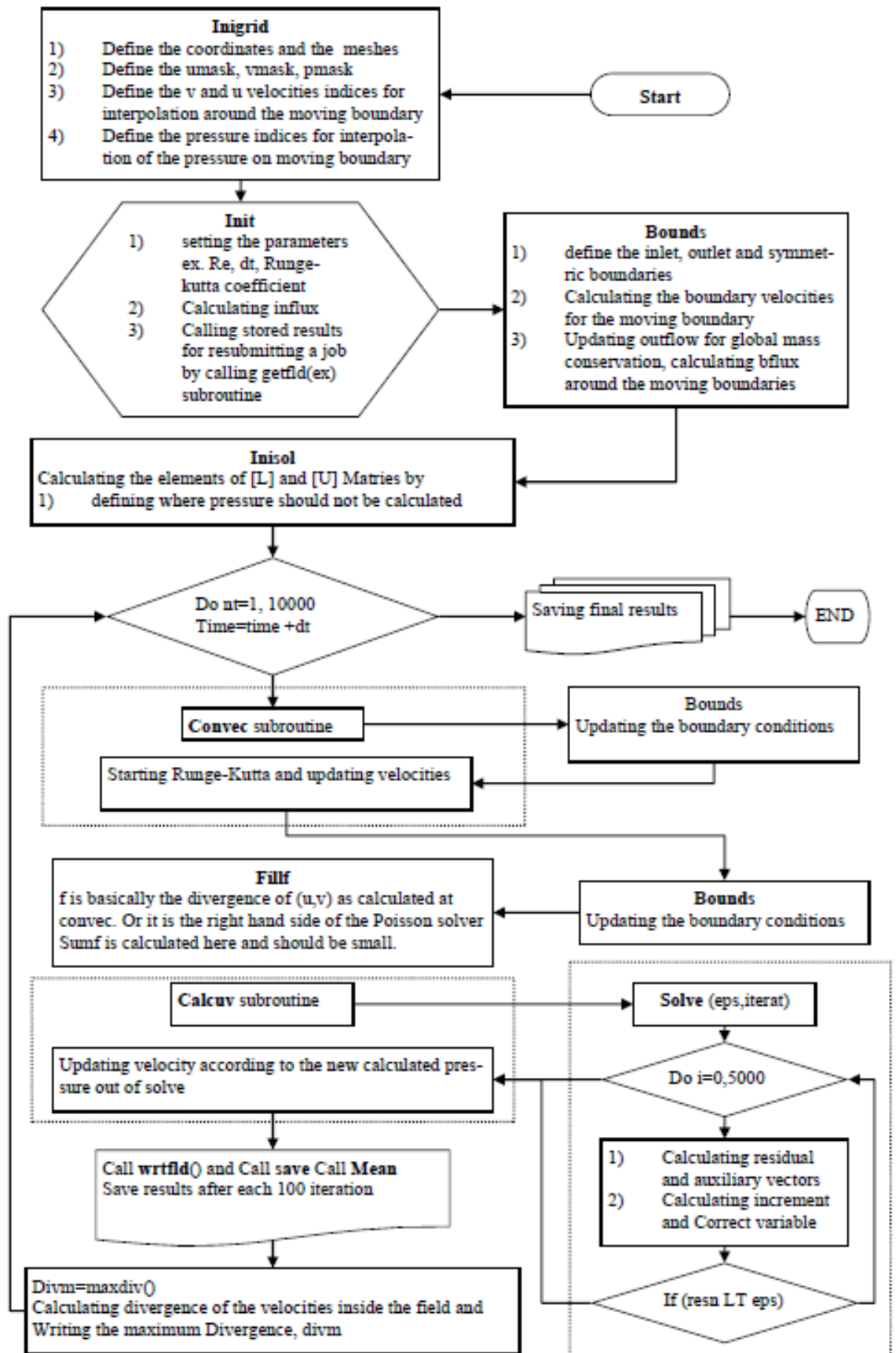


Figure 4-10: flowchart of the flow solver used to apply interpolation method

- In the second stage, a Runge-Kutta algorithm is used to calculate the intermediate velocity components by implementing the convective and diffusive part of the Navier-Stokes equations and updated boundary conditions.
- In the third stage, using the intermediate velocities the pressure Poisson equation is solved in the 'solve' subroutine. This algorithm is the most time consuming part of the codes and is repeated to find a converged solution for the pressure (to a user define range) at each time step (or at least 5000 times).
- In the final stage the velocity is updated using the new pressure from the previous stage which effectively projects the intermediate velocities on a divergence-free velocity field. This part is performed in the 'calcu' subroutine.

The program is marched in time from the second stage to reach a developed solution. At each time step the minimum and maximum divergence of the velocities are calculated. And the results are saved at each time step. The above algorithm is presented at Figure 4-10. In this flow chart the moving immersed boundary is not included. In the next section the solution of the Navier –Stokes equations in moving frame of reference is discussed and the related algorithm is explained in Chapter 7.

4.6 Moving frame of reference

Moving frame of reference has been widely used to solve the Fluid-Structure interaction for the problems in which a rigid body is displacing/rotating in a steady flow field (for instance Li et al. 2002). This method which is presented in Section 3.5 is capable of handling large displacement/rotation of a body in two dimensions. However, there are two main differences in the simulation used here and the one that Li et al 2002 has introduced. First of all, Li et al. used a spectral/hp spatial discretization, while, here an FVM with a staggered variable arrangement is used for the discretization. Secondly, here an immersed boundary with interpolation method is used to force the solid boundary, while in Li et al. the solid boundary was resolved with the unformatted mesh, so the no-slip boundary conditions would be directly enforced. For simplification, the cylinder was only allowed to move in transverse direction so that equation (3-37) could be simplified to incorporate only the acceleration of the solid boundary in the transverse direction. The moving frame of reference method is used to simulate the flow around an oscillating cylinder in the cross flow direction in chapter 7 in more detail. The

simulation results using this method are compared with the literature and results of simulation in an inertial frame of reference.

To evaluate the simulation and also to couple the fluid governing equation to the structural solver (for the fluid structure interaction), the forces and moment acting on a body in the moving frame of reference should be calculated. In the next sections, the calculation of force and moments acting on an immersed boundary with both an inertial frame of reference and a moving frame of references is addressed.

4.7 Evaluating forces and moment on an immersed boundary

To simulate Fluid-Structure-Interaction (FSI) and Vortex Induced Vibration (VIV) using immersed boundaries and the interpolation method, it is necessary to calculate the body forces explicitly. Here, the method used to calculate the lift and drag force due to pressure and shear stress is discussed. It is assumed that drag and lift forces are positive in the x and y direction, respectively. Figure 4-11(left) and equations (4-5) to (4-7) illustrate the calculation method for the lift and drag force due to the pressure on an immersed body.

The forces will be resolved into components parallel and perpendicular to the free stream velocity.

The hydrodynamic force exerted on a body by the flow can be obtained by the integration of local stress:

$$\sigma = -pI + \tau \quad (4-42)$$

$$F = \int_{\Gamma(t)} \sigma \hat{\mathbf{n}} d\mathbf{s} = - \int_{\Gamma(t)} p \hat{\mathbf{n}} d\mathbf{s} + \int_{\Gamma(t)} \tau \hat{\mathbf{n}} d\mathbf{s} = \hat{F}_p + \hat{F}_v \quad (4-43)$$

Where $\hat{\mathbf{n}}$ is the outward unit normal on the body, \hat{F}_p refers to the pressure force and \hat{F}_v refers to the viscous force. Note that the above integration is defined in the absolute frame of reference.

The total force, however, can be evaluated in the transformed plane and then mapped back onto the absolute frame of reference since:

$$F = \hat{F}_p + \hat{F}_v = A(F_p + F_v) \quad (4-44)$$

Where F_p, F_v are the forces calculated in the transformed plane.

$$dF = PdA \quad (4-45)$$

$$dF_{Liftp} = PdA(-\sin\theta) \quad (4-46)$$

$$dF_{Dragp} = PdA(-\cos\theta) \quad (4-47)$$

P is the pressure on the immersed boundary, dF_{Liftp} and dF_{Dragp} are the component of lift and drag due to pressure. dA is the area between two consecutive locations on the immersed boundary in which the pressure was used to calculate lift and drag forces. In IB methods, especially when using interpolation, the pressure on the immersed boundary is not known directly, however for the stationary cases it is assumed that the gradient of pressure is zero near the immersed boundary; hence the nearest pressure on the fluid domain is taken as the pressure on the immersed boundary (Figure 4-12). In the following part, the pressure calculation method is discussed in more details.

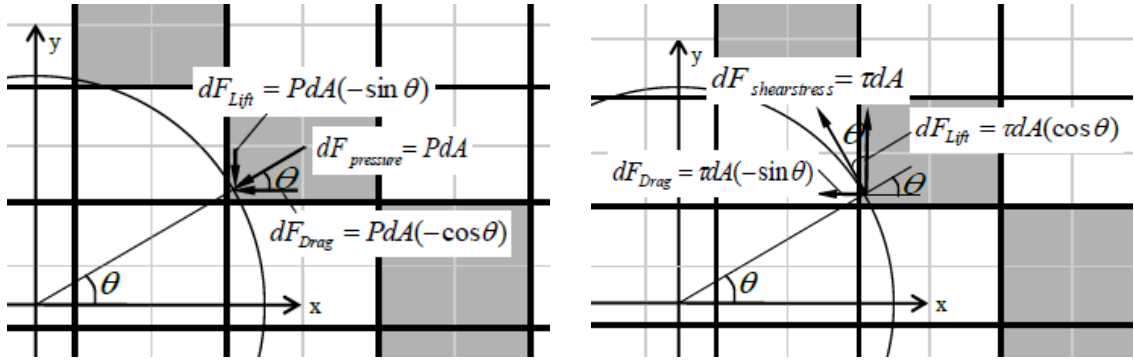


Figure 4-11: calculation of lift and drag component of force due to pressure (left) and shear force (right)

Lift and drag due to shear stress are calculated as illustrated in Figure 4-11 right, equations (4-48) and (4-49).

$$dF_{Shear} = \tau_{wall} dA \quad (4-48)$$

$$dF_y = dF_{lifts} = (\tau_{wall} dA) \cos \theta \quad (4-49)$$

$$dF_x = dF_{drags} = (\tau_{wall} dA) (-\sin \theta) \quad (4-50)$$

τ_{wall} is the shear stress on the immersed boundary. dF_{drags} and dF_{lifts} are the components of drag and lift due to the shear forces on the IB. The Shear stress calculation method is presented in the next part. To simulate the solid body with a rotational degree of freedom, calculation of the angular momentum is necessary. The momentum due to shear force can be calculated using the equation (4-51), in this equation, R is the radius of the immersed boundary (circular cylinder).

$$dM_{Shear} = R \tau_{wall} dA \quad (4-51)$$

Generally, in the inertial frame, the moment of the forces on a surface $\Gamma(t)$ of a body about an origin O (for instance the centre of the body) is given by:

$$\dot{M} = \int_{\Gamma(t)} \dot{r} \times (\sigma \cdot \hat{\mathbf{n}}) d\mathbf{s} = \oint p(\dot{r} \times \hat{\mathbf{n}}) d\mathbf{s} + \oint \dot{r} \times (\tau \cdot \hat{\mathbf{n}}) d\mathbf{s} = \dot{M}_p + \dot{M}_v, \quad (4-52)$$

where \dot{r} is a vector from the origin to the element of the surface $\Gamma(t)$. The origin is an optional point due to the definition of momentum; moment is a free vector. Therefore the resultant moment for both moving frame and inertial frame of reference is the same. Which simply states that \dot{r} is the rotated vector from the origin in the moving frame of reference. Therefore for a two-dimensional problem:

$$\dot{M} = M = - \oint p(\mathbf{x} \times \mathbf{n}) d\mathbf{s} + \oint \mathbf{x} \times (\tau \cdot \mathbf{n}) d\mathbf{s} \quad (4-53)$$

In the above equations, \dot{M} and M are the moments of the interaction forces in the inertial and moving frames of reference, respectively. Also \mathbf{x} is the location of the element on the surface $\Gamma(t)$ in the moving frame. The moments calculated either in the moving frame of reference or in the inertial inertia frame of reference will be the same.

4.8 Direct calculation of pressure over an IB

Finding the pressure around the immersed boundary is an important issue when calculating lift and drag forces. After finding the pressure on the immersed boundary, the body force due to the pressure on the immersed body can be calculated by integrating the pressure over its boundary. The vertical and horizontal components of the force will be Lift and Drag forces due to the pressure, respectively. The pressure on the immersed body can be calculated either directly or by extrapolation.

4.8.1 Calculation of pressure force without extrapolation

For a stationary immersed boundary or a boundary with constant velocity, one can assume that the gradient of the pressure in the perpendicular direction to the surface is zero close to the boundary. Therefore, the pressure on the immersed boundary will be the same as the pressure in the nearest cell when looking outward in the radial direction. These pressures are located in the flow domain and updated by the governing equations of the fluid flow. Figure 4-12 left, illustrates how, the pressure near the cylinder was used as the pressure on top of the cylinder.

If the immersed body undergoes acceleration, the gradient of the pressure near the IB is not negligible and the gradient of the pressure can be calculated by projecting the momentum equation in the direction perpendicular to the immersed boundary. The subject will be address in the Chapter 7.

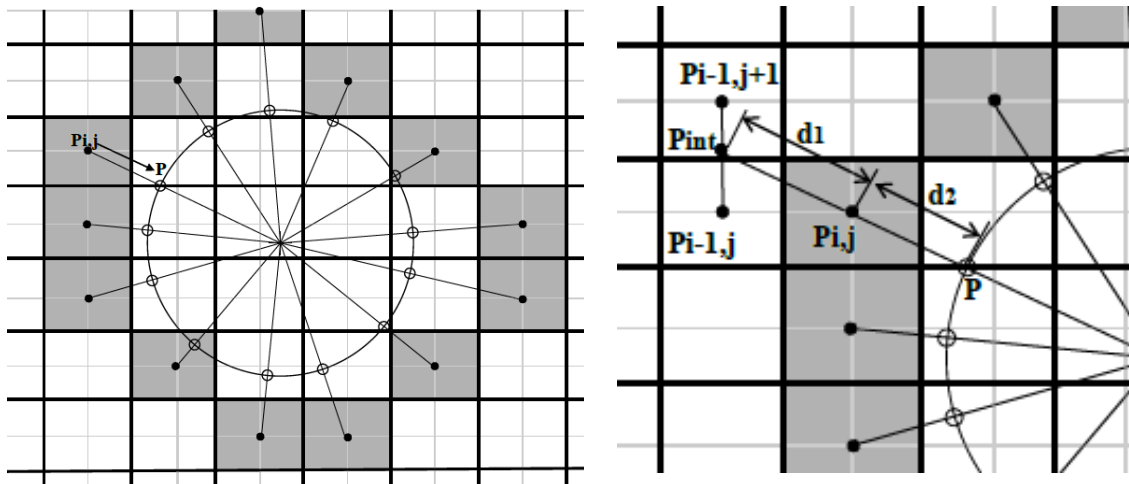


Figure 4-12: left, pressure near the immersed boundary directly used as pressure on the boundary. Right, linear extrapolation method to calculate pressure on the immersed boundary.

4.8.2 Extrapolating the pressure

This method could be either linear or nonlinear (second order, exponential...). In this research only the linear extrapolation of the pressure to the cylinder wall is studied. For the linear extrapolation of the pressure on the cylinder, two consecutive pressure values in the perpendicular direction to the immersed boundary are needed for each point. Figure 4-12 shows a schematic of the extrapolation method. A line perpendicular to the immersed boundary is used to find two pressure values at 2 locations. The first pressure, $P_{i,j}$ is used directly, however, the second pressure, P_{int} , is interpolated using two other pressure points. Following the calculation of $P_{i,j}$ and P_{int} , the pressure on the cylinder can be found by linear extrapolation.

4.8.3 Calculation of the shear forces around a cylinder

As it has been mentioned earlier, to calculate shear forces around the immersed boundary it is necessary to find the gradient of the velocities around the boundary. The gradient of velocity parallel to the cylinder is assumed to be linear at each point around the cylinder. As the velocity of the cylinder is known (from the structural analysis), the first step is to find the velocities in the centre of the boundary cells around the cylinder. In the staggered arrangement, the u and v velocities in the centre of the cell are calculated by averaging their values from the cell edges. The location of tangential velocity around the cylinder is shown in Figure 4-13 left. In the second step, the tangential velocity is calculated by projecting the velocities vector on the local tangent to

the solid boundary. It is supposed that the tangential velocity is positive in the counter-clock wise direction in order to obtain a unique formula for the calculation of the tangential velocities around the boundary. Equation and Figure 4-13 right depict the calculation of the tangential velocity for a specific point around the cylinder. In the final step, shear stress and shear force on the boundary are calculated using equation (4-56). In addition, the lift and drag forces are calculated by projecting the shear forces in x and y direction respectively. It is worth mentioning that taking the counter clock wise direction as the positive direction is optional and this does not change the generality of the method. However, it should be noted that in the calculation of both the lift and drag forces the same assumption is made.

Using the above, the total lift and drag forces around the cylinder can be calculated by integrating their partial values around the immersed boundary.

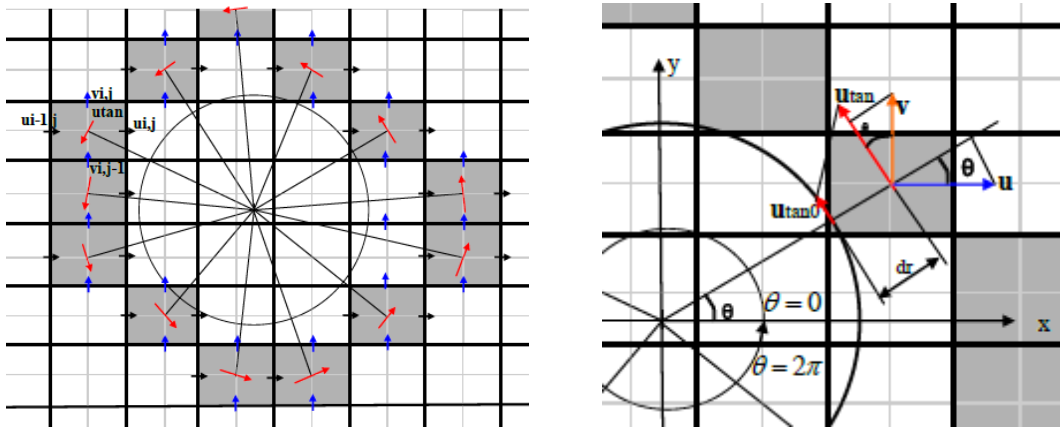


Figure 4-13: Calculating tangential velocity around the immersed boundary

$$U_{tan} = -U \sin \theta + V \cos \theta \quad (4-54)$$

$$U_{tan0} = -U_{solid} \sin \theta + V_{solid} \cos \theta \quad (4-55)$$

In the above equations, U_{tan0} and U_{tan} are the tangential velocities on the immersed boundary and the boundary cell, respectively. The shear stress on the boundary can be calculated by equation (4-56). In this equation dr is the distance between U_{tan} and U_{tan0} in the radial direction and μ is the dynamic viscosity of the fluid.

$$\tau_{wall} = \mu \frac{du}{dy} = \mu \frac{U_{tan} - U_{tan0}}{dr} \quad (4-56)$$

In general, shear stress is defined by $\tau = \mu \frac{du}{dy}$, however if the two side of this equation are divided by density, then $\tau/\rho = \mu/\rho \frac{du}{dy}$ and $\vartheta = \mu/\rho$. Also, in the numerical

simulation of fluid flow for simplicity, it is assumed that $u_\infty = 1$ and $D = 1$, so that $Re = \frac{\rho u_\infty D}{\mu} = \frac{1}{\vartheta}$ or $\vartheta = 1/Re$ and therefore $\tau/\rho = \vartheta \frac{du}{dy}$. Using the shear stress over the density on the immersed boundary can be calculated by $\tau/\rho = 1/Re \frac{du}{dy}$.

The above idea can be explained using non-dimensional parameters as well. According to equation (4-7), the non-dimensional form of the shear stress is given by equation (4-57). In equation (4-58) it is simply shown that $1/Re \frac{du}{dy}$ is a non-dimensional value as long as the velocity and displacement are non-dimensional.

$$\tau^* = \frac{\tau}{\rho u_\infty^2} \quad (4-57)$$

$$\tau^* = 1/Re \frac{du^*}{dy^*} = \frac{1}{\frac{\rho u_\infty D}{\mu}} \times \frac{\frac{du}{dy}}{\frac{u_\infty}{D}} = \frac{1}{\rho u_\infty^2} \times \mu \times \frac{du}{dy} = \frac{\tau}{\rho u_\infty^2} \quad (4-58)$$

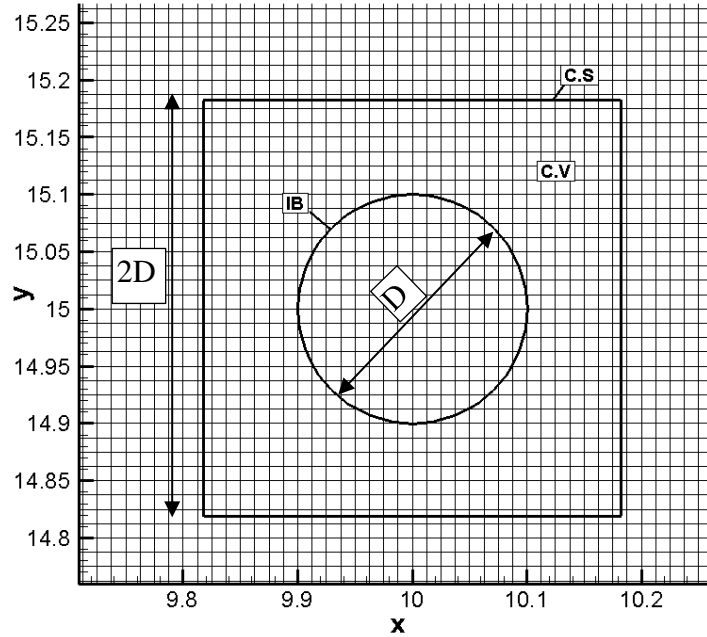


Figure 4-14: Location of Immersed boundary (IB), control volume (C.V.), Control Surface (C.S.) to apply Conservation of momentum law

4.8.4 Application of momentum conservation to calculate force on IB

The accurate calculation of lift and drag forces on an IB is a challenging task, especially when an interpolation/reconstruction IB is used. The reason is that on the one hand the forces on the IB surface strongly depend on the formation of vortices and on the other the way boundary conditions are forced affect vortices when using a non-conforming mesh method. Despite the existence of extensive literature about the FSI methods, the calculation of body forces have received much less attention (Balaras

2004). In this section, as mentioned earlier in section 3.8.3, conservation of momentum is applied to calculate the forces around an immersed boundary.

As illustrated in the Figure 4-14, conservation of momentum (equation (3-51)) is applied to the control volume limited between the IB and C.S in horizontal (x) direction and vertical (y) direction to calculate drag and lift forces respectively.

$$F_{Drag} = \frac{d}{dt} \int_{C.V} \rho u dA - \int_{\Gamma_0=C.S} (\rho u u_j + p \delta_{1j} - \tau_{1j}) n_j ds \quad (4-59)$$

$$F_{Lift} = \frac{d}{dt} \int_{C.V} \rho v dA - \int_{\Gamma_0=C.S} (\rho v u_j + p \delta_{2j} - \tau_{2j}) n_j ds \quad (4-60)$$

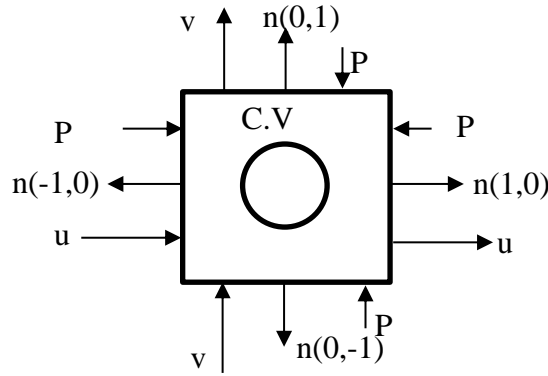


Figure 4-15: Surface normal vector n , velocity (u,v) and pressure on the control surfaces

According to the Figure 4-15, the last integrals (the control surface integral) in equations (4-59) and (4-60) are expanded to enable calculating the lift and drag forces on the immersed boundary. Using this, the control surface integral in equation (4-58) becomes:

$$\begin{aligned} & \int_{\Gamma_0=C.S} (\rho u u_j + p \delta_{1j} - \tau_{1j}) n_j ds \quad (4-61) \\ &= \int_{C.S.west} (\rho u(-u) + (-p) \\ & \quad - (-\tau_{11})) ds \\ &+ \int_{C.S.north} (\rho u(v) + (0) - (\tau_{12})) ds \\ &+ \int_{C.S.east} (\rho u(u) + (p) - (\tau_{11})) ds \\ &+ \int_{C.S.south} (\rho u(-v) - (0) - (-\tau_{12})) ds \end{aligned}$$

The first integral in (4-59)

resent the temporal changes of the momentum in the control volume.

$$\begin{aligned}
 \int_{\Gamma_0=C.S} (\rho v u_j + p \delta_{2j} - \tau_{2j}) n_j ds & \quad (4-62) \\
 &= \int_{C.S.west} (\rho v(-u) + (0) \\
 &\quad - (-\tau_{21})) ds \\
 &+ \int_{C.S.north} (\rho v(v) + (p) - (\tau_{21})) ds \\
 &+ \int_{C.S.east} (\rho v(u) + (0) - (\tau_{22})) ds \\
 &+ \int_{C.S.south} (\rho v(-v) - (p) - (-\tau_{22})) ds
 \end{aligned}$$

Where in the above equations, the stresses τ_{11} , τ_{22} , τ_{12} are defined by:

$$\tau_{11} = -\frac{2}{3}\mu\nabla.V + 2\mu\frac{\partial u}{\partial x} \quad (4-63)$$

$$\tau_{22} = -\frac{2}{3}\mu\nabla.V + 2\mu\frac{\partial v}{\partial y} \quad (4-64)$$

$$\tau_{12} = \tau_{21} = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \quad (4-65)$$

The first term on the left hand side of equations (4-59) and (4-60) become near zero in the steady state condition and can be neglected, however, in this study in order to be able to present the results in the transient conditions, they are integrated in the control volume/surface (surface bounded between the IB boundary and the control surface (C.S.)).

4.9 Lift and drag coefficient

The dimensionless drag, C_D , and lift, C_L , coefficients are defined by:

$$C_D = \frac{F_{Drag}}{\frac{1}{2}\rho u_{\infty}^2 D} \quad (4-66)$$

$$C_L = \frac{F_{Lift}}{\frac{1}{2}\rho u_{\infty}^2 D} \quad (4-67)$$

where F_{Drag} , F_{Lift} , ρ , u_{∞} , D are drag force, lift force, fluid density, free stream velocity and the cylinder diameter, respectively. In these equations the value of the drag and lift forces are dimensional. On the other hand, the values of the drag and lift forces which are calculated in the equations (4-59) and (4-60) are non-dimensional because all parameters on the right hand side of these equations are non-dimensional. Therefore, to calculate the lift and drag coefficients from the drag and lift forces (equations (4-59) and (4-60)), equations (4-66) and (4-67) become:

$$C_D = 2 \times F_{Drag} \quad (4-68)$$

$$C_L = 2 \times F_{Lift} \quad (4-69)$$

In addition, as discussed earlier, the drag and lift forces on a cylinder submerged in a flow arise from two sources, the shear stress and the pressure distribution over the body. Therefore:

$$C_D = 2 \times F_{Dragp} + 2 \times F_{Drags} \quad (4-70)$$

$$C_L = 2 \times F_{Liftp} + 2 \times F_{Lifts} \quad (4-71)$$

4.10 Summary

In this chapter the main body of the algorithm that is developed to simulate flow around a solid boundary is outlined. Most of the details are explained in a way to support the interpolation/reconstruction immersed boundary method. At the beginning, the governing equations and their discretisation procedures are discussed. It is explained that a fractional step method is used to update the velocities at each new time step. Then the background Cartesian grid was introduced using a staggered arrangement of velocities. The boundary conditions at the inlet, outlet, symmetry and immersed boundaries are introduced in detail in Section 4.4. In chapter 7 it will be shown that the boundary conditions play a vital role in the definition of the moving frame of reference.

One of the contributions of this research is the implementation of the immersed boundary in a Cartesian grid using the interpolation method presented in Section 4.4.4. In that part, the way in which velocities are interpolated near the immersed boundary is explained. In Section 4.5 the solution algorithm is briefly explained. In this procedure, one of the main challenges is the calculation of the pressure and the shear forces at the cylinder due to the fact that the immersed boundary is not aligned with the grid. This problem is addressed in sections 4.7 and 4.8. In these sections the methods used to

calculate lift and drag forces on the immersed boundary are presented and briefly compared. The accurate calculation of lift and drag forces is necessary to be able to implement the fluid structure interaction for a flexible solid body. This issue is discussed in more details in chapter 7.

In the next chapter, the algorithm outlined here is validated by comparing the results with a bench mark. As flow around a circular cylinder has been studied extensively, it was decided to use this as a bench mark; this case has similarities to the simulation of the oil riser pipe, the study of which is the ultimate aim of this research.

In addition, to clarify the role of the computational grid on the results, a comprehensive parametric study is performed for the two dimensional flow around a circular cylinder in next chapter.

Chapter 5. Parametric study and validation

In the previous chapter the algorithm which was developed to simulate the flow around an immersed boundary was presented. An immersed boundary interpolation method was used to apply the solid boundary conditions. In this chapter, the fluid flow around a stationary cylinder in two dimensions at a low Reynolds number is selected as a bench mark to validate the code written in FORTRAN. This bench mark has been used by many researchers to validate their methods resulting in several experimental and numerical simulations which are available in the literature for comparison purposes.

At first a parametric study is performed. Here, six parameters related to the size of computational domain which might affect the simulation results are investigated. In addition, the results of the lift and drag coefficients at low Reynolds number, $Re=100$ are compared with those in the literature to assess the accuracy of the method. In this study, the hydrodynamic forces are calculated by two methods: 1) by application of the conservation of momentum and 2) by a direct integration of pressure and shear force on the immersed boundary.

5.1 Parametric study

The ultimate goal of this research is to apply the strip theory to simulate the interaction of the fluid flow and oil risers. In this theory, the flow around the cross section of the riser is simulated at several levels along the pipe. The hydrodynamics forces that are calculated at each level are linked through the structural model to update the location/shape of the riser. This process is repeated several times to obtain a converged solution at every time step. This simulation requires very high computational power. Therefore, identifying methods that allow minimizing the computational demand needed to solve the Fluid-Structure interaction (FSI) problems and in particular the riser problem is of paramount importance. In this chapter the parameters that might affect the simulation of flow around a cross section of the riser are investigated. The criterion was to select the parameters in such a way as to minimized computational power while still

providing acceptable results. To achieve this goal, the two-dimensional flow around a stationary cylinder was taken as a bench mark.

In the first place a grid refinement study is performed to investigate the dependency of the interpolation method on the size of the mesh near the immersed boundary. Also, the size of the computational domain might be very important. On the one hand, the domain boundaries should be far enough away (large computational domain) from the cylinder to be able to neglect the effect of the boundaries on the accuracy of simulation and, on the other, the domain should be small enough to limit the computational demand. The overall effect of the domain in the y direction is addressed by studying the blockage effect in the literature. The effect of the domain size upstream of the cylinder is referred to as the entrance effect and is addressed for the first time in this thesis as far as the author is aware. This part is labelled 'c' in Figure 5-2.

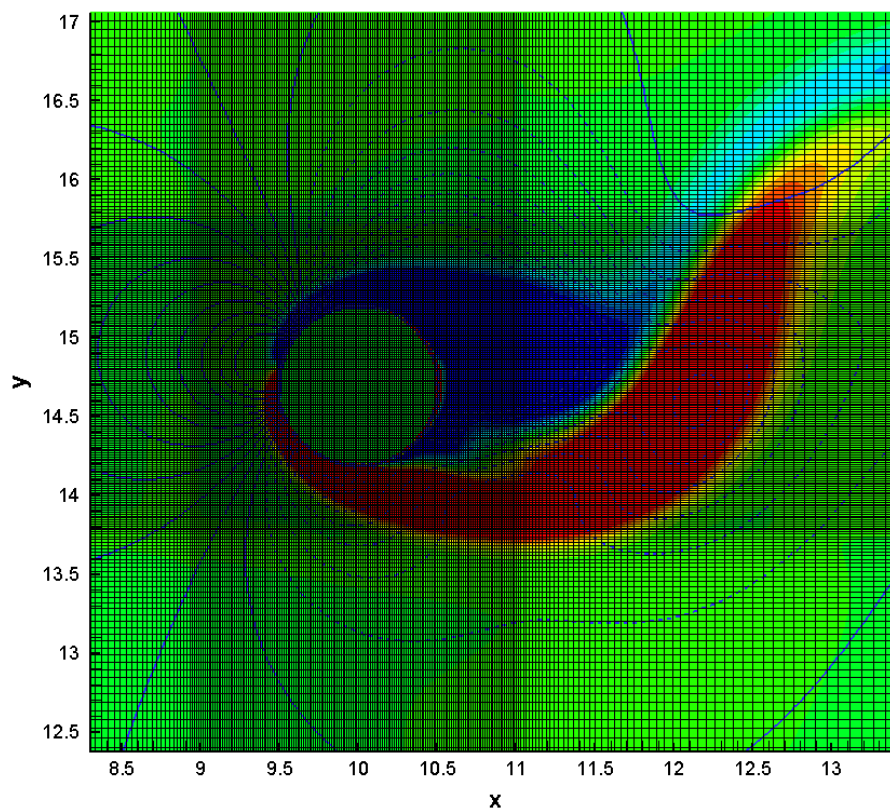


Figure 5-1: flow pattern around a stationary cylinder at $Re=100$. High pressure area (Continuous line), low pressure area (dash line), blue and red counters are the vortices.

In general, the places where the variables exhibit large gradients are the most sensitive regions with regard to the grid size. According to Figure 5-1, the areas around the cylinder with very high pressure and velocities gradients coincide. Therefore a very

dense mesh is necessary around the cylinder. On the other hand, there are hardly any gradients far from the cylinder; therefore, a coarse mesh can be used in these locations. To address this issue, a uniform mesh around the cylinder is used. However the size of this uniform grid area might be important as well. To investigate this effect, the size of uniform grid before and after the cylinder (x direction) and also the size of uniform area in the y direction are studied separately; these lengths are identified by ‘e’, ‘f’ and ‘b’ in the Figure 5-2.

In addition, using a stretching factor is necessary to maintain a coarse grid far from the cylinder (area with low gradients) and to have a fine grid near the cylinder. The effect of the stretching factor is studied as well. To fulfil these criteria a comprehensive investigation is presented in this chapter to show the effect of the domain and grid sizes on the simulation results.

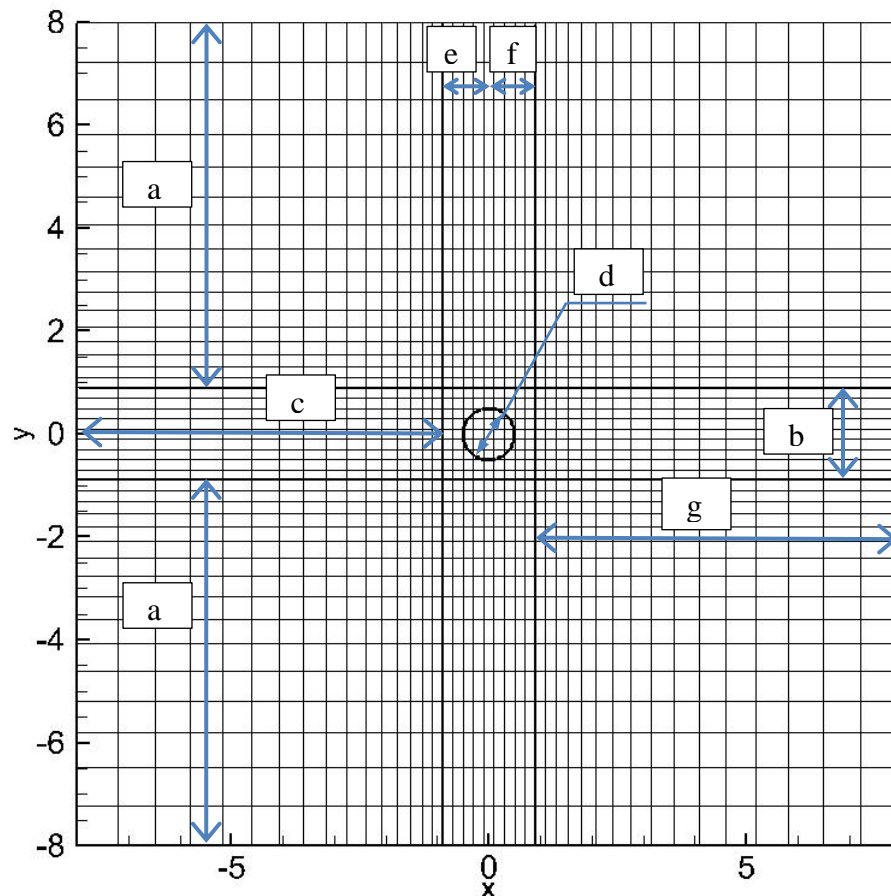


Figure 5-2: Background Cartesian mesh- parametric studies guide.

5.1.1 Parametric study - Mesh refinement effect

The size of the mesh near the Immersed Boundary (IB) plays a significant role both in the accuracy of the results and in the computational expenses. To find the proper

mesh size for the numerical simulation and also maintain the second-order accuracy of the model, a mesh refinement study is performed. In this study, the centre of the cylinder is located at centre of the Cartesian coordinate and the size of computational domain in both the x and y directions is taken as $[-15D,15D]$, and the uniform grid around the cylinder in both the x and y directions covers the regions $[-1D,1D] \times [-1D,1D]$ (2 times of the cylinder diameter in each direction). In this uniform area around the cylinder 6 grid sizes ranging from $0.2D$ to $0.00625D$ are used for the simulation (see Table 5-1). A stretching factor of 3 is used to extend the grid from uniform area to the computational boundary in all 6 cases and the Strouhal number, the drag and the lift coefficients for the flow problem are compared. Table 5-1 shows the details of the grids and their results. The stretching factor helps to reduce the actual number of nodes in the grid. For instance, in a $30D \times 30D$ domain using a grid size of $0.2D$ (without stretching) the number of points in each direction becomes 150. This number reduces to 57 grid points when using a stretching factor of 3. The effect of stretching on hydrodynamic forces will be discussed later in this chapter.

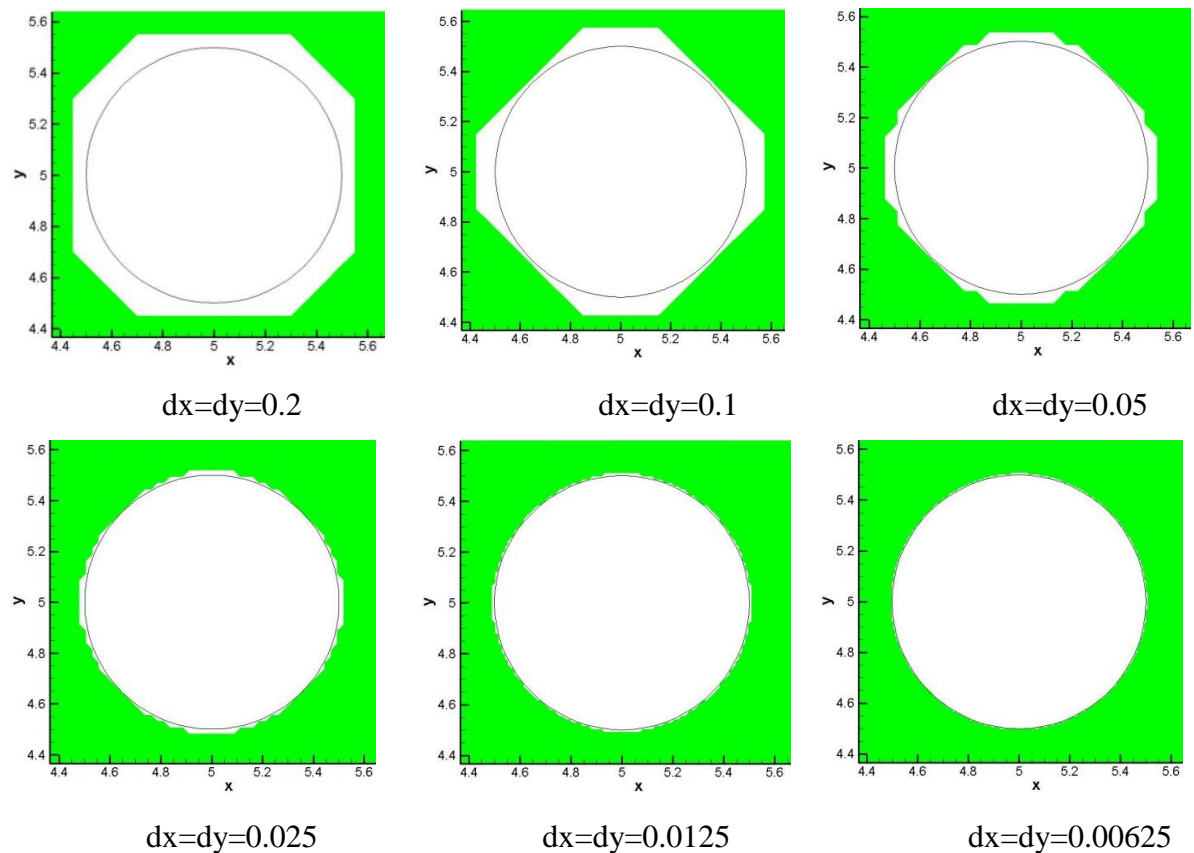


Figure 5-3: simulation accuracy of the immersed boundary based on the mesh size

In non-conforming boundary approaches, when the grid get finner near the immersed boundary the shape of the cylinder (IB) is approximated more accurately. Figure 5-3 compares the effect of the grid size close to the cylinder on the approximation of the cylinder boundary. Clearly, the finer grids lead to a better approximation and are likely to produce more accurate results.

Table 5-1: Results of mesh refinement study around a stationary cylinder at Re=100.

$\Delta x = \Delta y$	Number of grid at each direction	Total no. of grid points	Strouhal No.	Mean drag coefficient	Max lift coefficient
0.2	57*57	3'249	0.147	1.467	0.285
0.1	109*109	11'881	0.154	1.315	0.225
0.05	213*213	45'369	0.159	1.334	0.305
0.025	423*423	178'929	0.1637	1.329	0.314
0.0125	837*837	700'569	0.174	1.327	0.315
0.00625	1669*1669	2'785'561	0.1743	1.328	0.316

Figure 5-4 presents the drag coefficient, drag due to pressure and drag due to the shear stress for 5 different grid sizes from $dx=dy=0.1D$ to $0.00625D$. The results for $dx=dy=0.2D$ are not shown as it is out of the range compared to the other results. The results show that the components of the drag (drag due to pressure and shear stress) are more affected by the grid size than the drag coefficient. For instance, the mean drag coefficient due to pressure reduces from 1.15 to 1.1, which is about 4.5%, when the grids become finer from $0.1D$ to $0.05D$; while the mean drag increased from 1.315 to 1.335, which is about 1.5%.

In addition, Figure 5-4 and Figure 5-6 show that by increasing the number of grid points the mean drag due to pressure reduces and converges to the value of 1.05. This trend, however, is reversed for the drag due to the shear stress. The results show that the mean drag due to shear stress increases and converges to a value of 0.33 by increasing the number of the grid points in each direction from 50 to 1600. Therefore, the drag coefficients for sufficiently fine grids (approximately finer than $0.025D$) are less dependent on the grid size due to the fact that the errors in the calculation of the drag due to the shear stress and pressure tend to cancel one another.

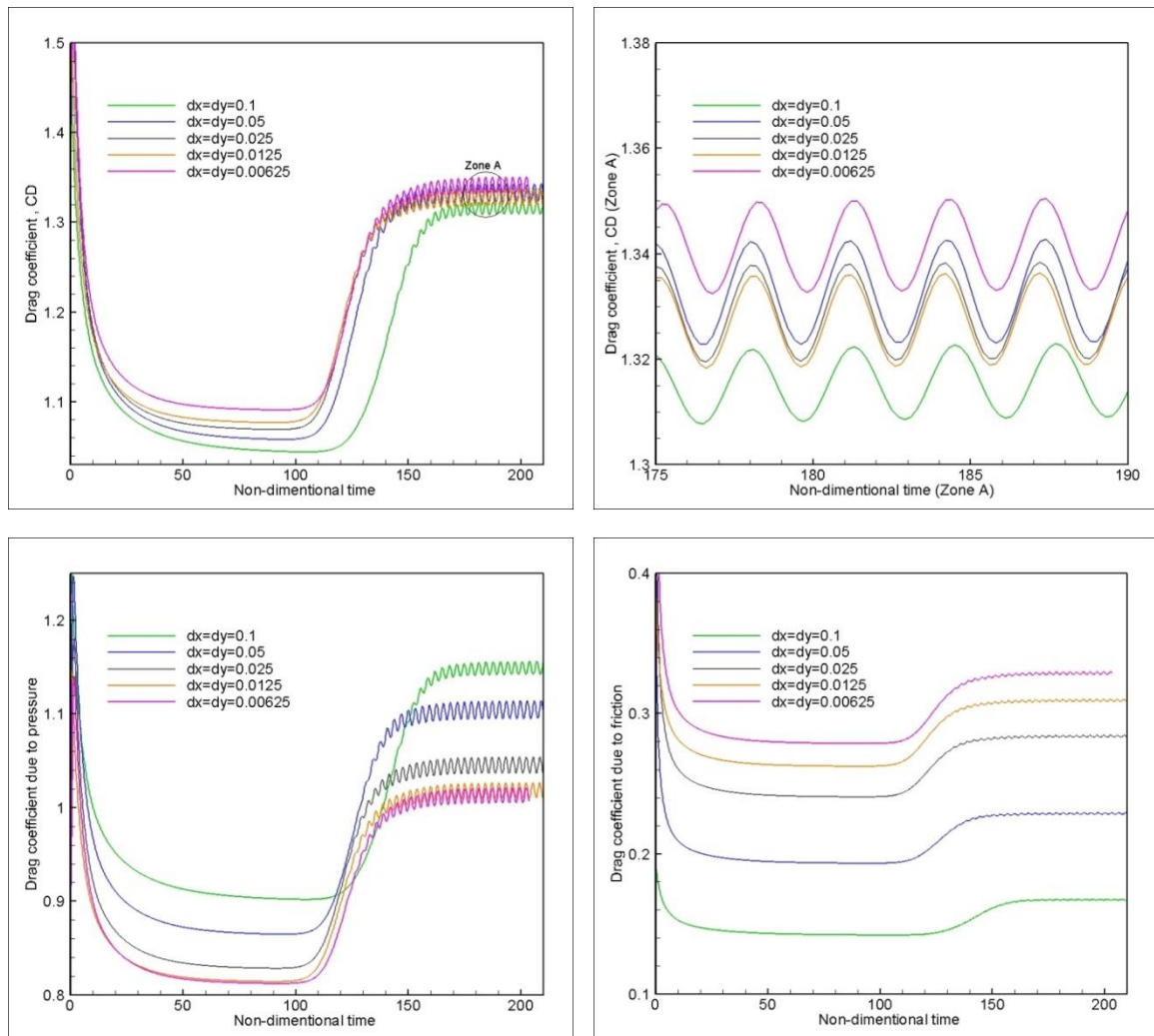


Figure 5-4: Mesh refinement study, drag, drag due to pressure and shear stress for five different grid sizes from $dx=dy=0.1$ to 0.00625 around the circular cylinder

The lift coefficient, the lift due to pressure and shear stress are compared in Figure 5-6 for five different grid sizes from $dx=dy=0.1$ to 0.00625 . The results for the grid with $dx=dy=0.2$ (coarsest grid) is not shown as it is out of range in compare to the other cases. The numerical results show that (unlike the drag coefficient components) the lift coefficient, lift due to the pressure and the shear stress have similar trends. For instance, if the grid sizes are reduced from $0.1D$ to $0.05D$ the total lift, lift due to pressure and lift due to shear stress increase from 0.22 to 0.3 , from 0.21 to 0.28 and from 0.01 to 0.02 respectively. Also, Figure 5-5 shows that the drag due to pressure and friction are converging for the grid size smaller than 0.025 (see Table 5-1).

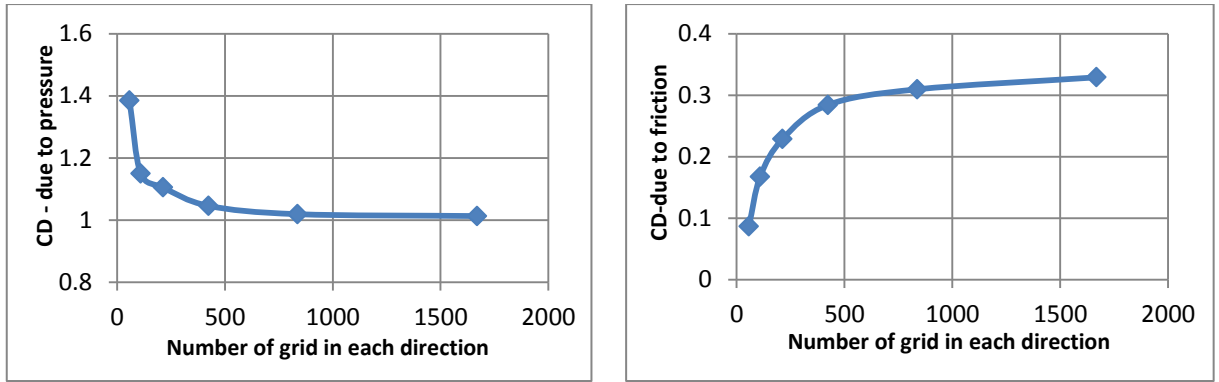


Figure 5-5: Drag coefficient due to pressure and shear stress versus the number of grid in each direction of the domain around a stationary cylinder at low Reynolds number, $Re=100$

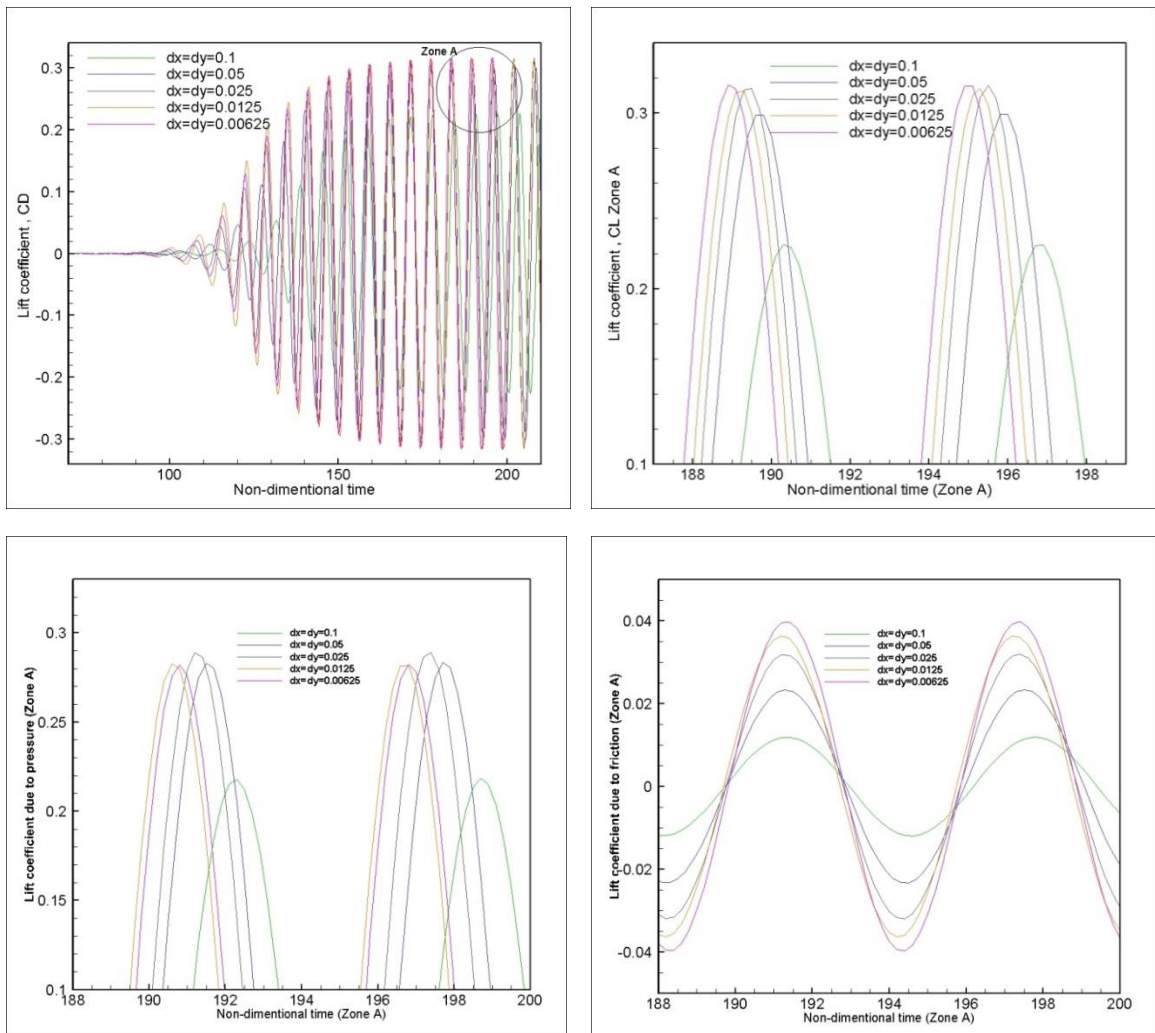


Figure 5-6: Mesh refinement study for lift, lift due to pressure and friction for various grid size where computational domain in x and y is $[-15,15]$ and Stretching factor is 3.

In addition, Figure 5-7 shows that the lift highly depends on the grid size in the coarse grid range. For instance, the lift coefficient increases significantly from 0.225 to 0.305 by decreasing the grid size from 0.1D to 0.05D which is about a 26% rise; while for the relatively fine grids (finer than 0.025D), the lift coefficient is less dependent on the grid size (at low Reynolds number).

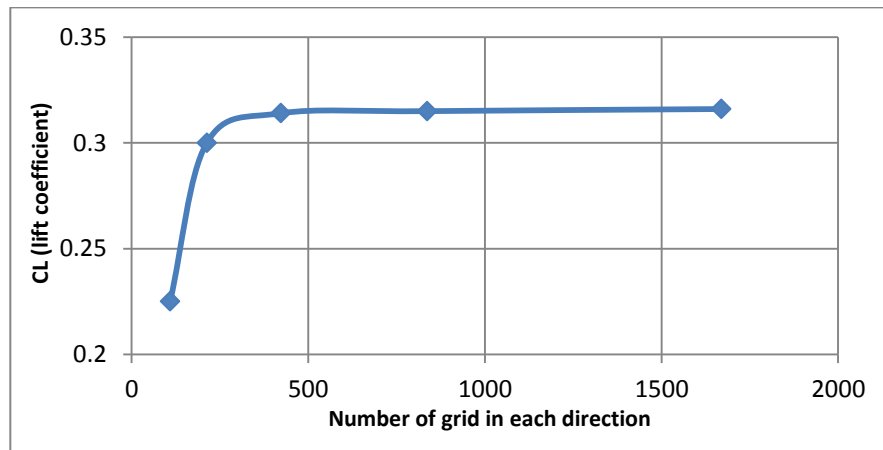


Figure 5-7: Lift coefficient versus the number of grid points in each direction of the domain around a stationary cylinder at Low Reynolds number, $Re=100$

A comparison between the results obtained for the lift and drag coefficients shows that the lift coefficient, more than the drag coefficient, depends on the grid size for the coarse meshes. For the fine meshes both of them are relatively independent of the grid size. For instance, by decreasing the grid size from 0.1D to 0.05D the drag and lift coefficients change by 1.5% and 26% respectively. A further decrease in grid size from 0.025D to 0.0125D leads only to a lift and drag coefficient change of about 0.15% and 0.3% respectively. It should be noted that for any grid size the drag coefficient is not as grid dependent as the lift coefficient. This is due to the fact that the errors in the drag due to the pressure and shear stress cancel each other out. For the cases $dx=dy=0.025D$ and $0.0125D$, the difference in the drag due to pressure is about 2.5% and the drag due to shear stress changes by about 8%. However, the change in the drag coefficient is just about 0.15%.

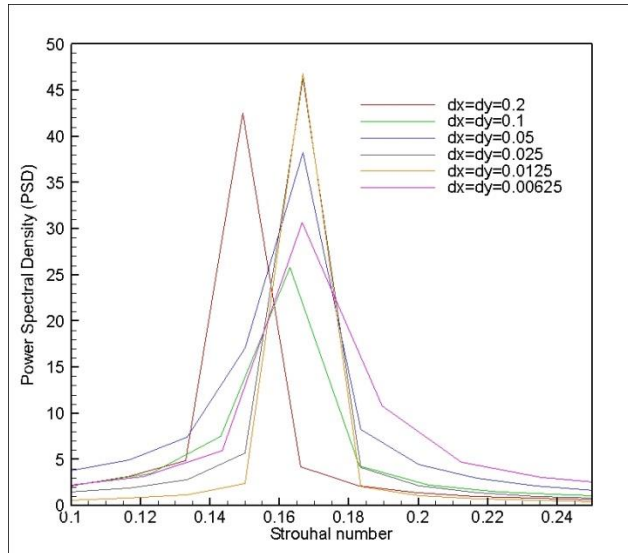


Figure 5-8: The Power Spectral density of lift coefficient for six different grids size in frequency domain, where computational domain in x and y is [-15, 15] and Stretching factor is 3.

The numerical results also show that the Strouhal frequencies are affected most by the coarse grids. For instance, for grid size $dx=dy=0.2D$, the Strouhal frequency is 0.147, which it is 4.5% lower than the Strouhal frequency for the grid size 0.1D. Figure 5-8 shows the power spectral density (PSD) of the lift coefficient for six different grid sizes ranging from 0.2D to 0.00625D at low Reynolds flow, $Re=100$. For fine grids the Strouhal number is much less dependent of the grid size and converges to the value $f_s=0.164$ (Figure 5-9).

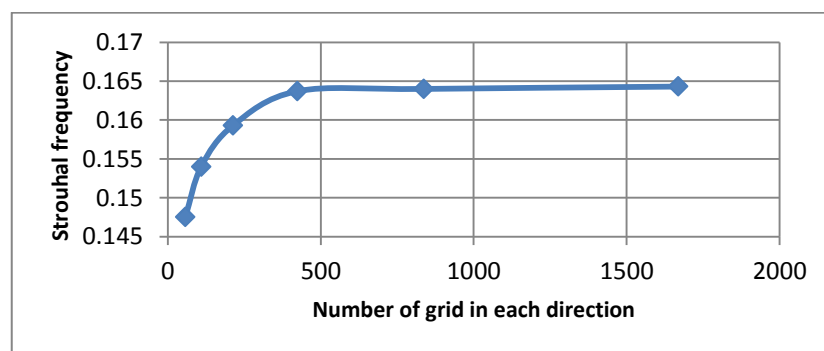


Figure 5-9: Strouhal number verses the number of grid point in each direction of the domain around a Stationary cylinder at low Reynolds number, $Re=100$

5.1.2 Parametric study – size of domain in front of cylinder

The size of the computational domain in front of the cylinder is an important parameter in the study of the flow over a circular cylinder at low Reynolds number. To study this effect the flow over a stationary cylinder at $Re=100$ is simulated. Four different flow domain sizes ranging from 5D to 20D upstream of the cylinder are compared, whilst other domain parameters are kept constant. The size of the domain in the transverse direction is 30 D; the grid size in the uniform area around the cylinder is $dx=dy=0.025$, and the sizes of the uniform grid area is 1D and 5D in front of and after the cylinder in the x direction and 3D above and below the cylinder in the y direction (Figure 5-2). The grid stretching factor for the mesh from the uniform area to the border of the computational grid is 3.

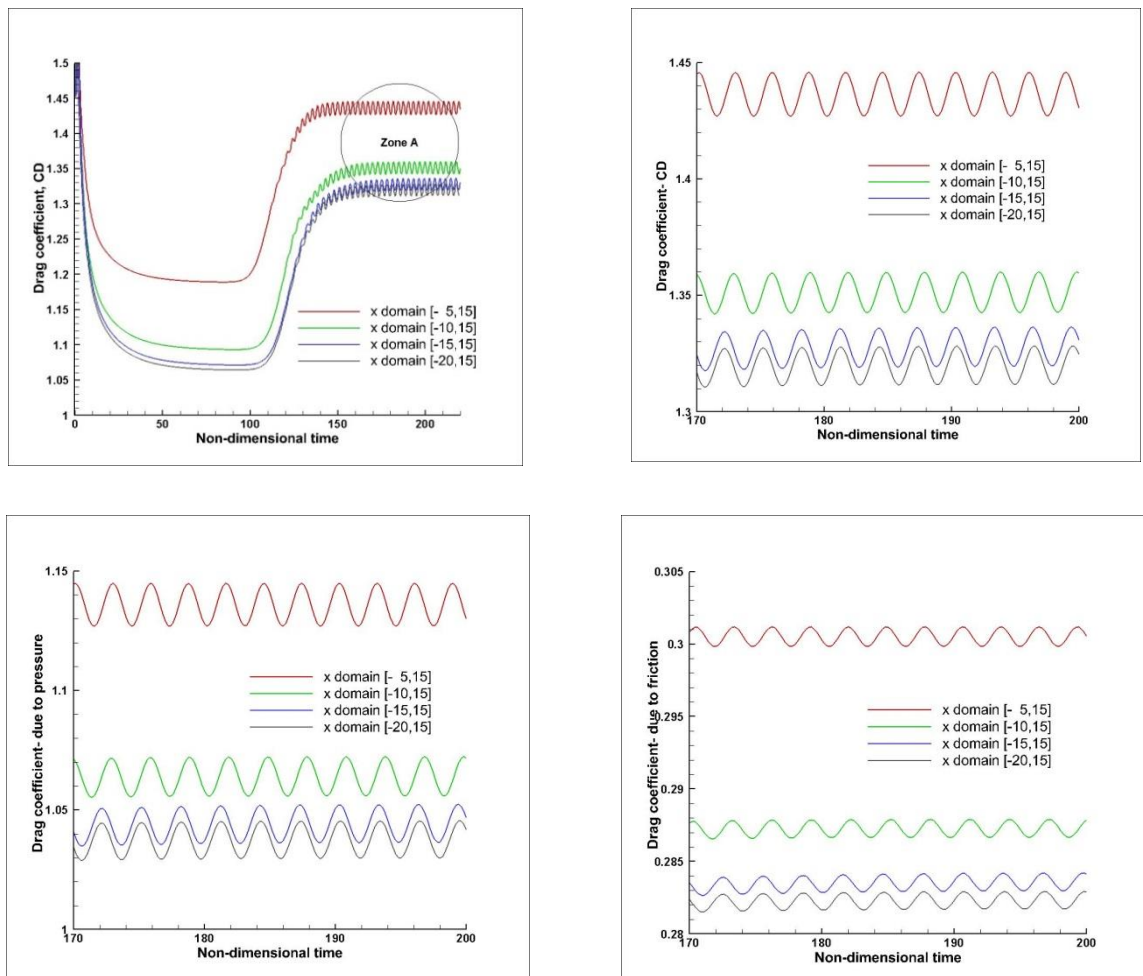


Figure 5-10: Effect of the Size of the fluid domain in front of the circular cylinder in x direction on the Drag coefficient

It can be seen that the size of computational domain significantly affects the results (Figure 5-10). The drag coefficient changes by 10% (from 1.44 to the 1.32) when the domain size changes from 5D to 20D behind the cylinder. This value is decreased by 0.6% when the size of domain in front of the cylinder is increased from 15D to 20D (from 1.328 to 1.32). The simulation results show that this trend is similar for the drag coefficients due to the pressure and friction. It can be concluded that the size of 15D behind the cylinder gives sufficiently accurate results at relatively low computational cost.

By increasing the size of the domain in front of the cylinder from 5D to 20D the lift coefficient is affected in a similar way as with the drag coefficient (Figure 5-11). In this case, the lift coefficient decreases from 0.337 to 0.3 (about 12%). This change becomes less than 3%, when the sized of the domain in front of the cylinder increases from 15D to 20D.

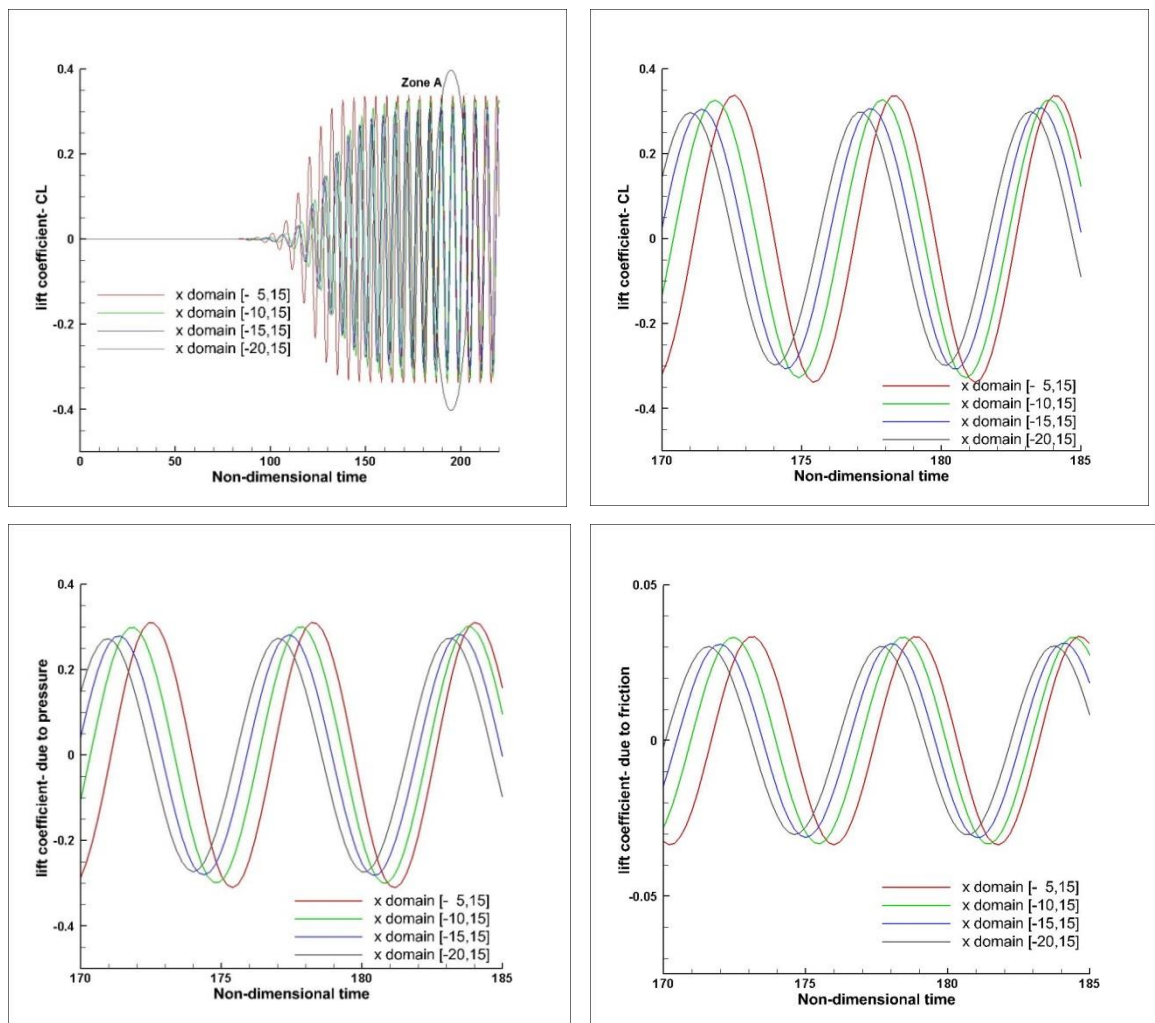


Figure 5-11: Effect of the Size of the fluid domain in front of the circular cylinder in x direction on the lift coefficient

By using larger computational domain in front of the cylinder the simulation results show that the lift due to pressure is affected slightly more than the lift due to friction. For instance the lift coefficient is changed by about 12% while the computational domain in front of the cylinder is changed from 5D to 15D. In this case, the lift force due to pressure is changed slightly more than 13% and the lift due to the friction is changed by less than 10%.

Another important parameter which is affected by the size of the domain in front of the cylinder is the Strouhal number. By increasing the size of domain in front of the cylinder from 5D to 20D this parameter is decreased from 0.173 to 0.164 (about 5.5%). However, for a sufficiently large domain in front of the cylinder (15D and above) there is hardly any difference in the Strouhal number results (Figure 5-12).

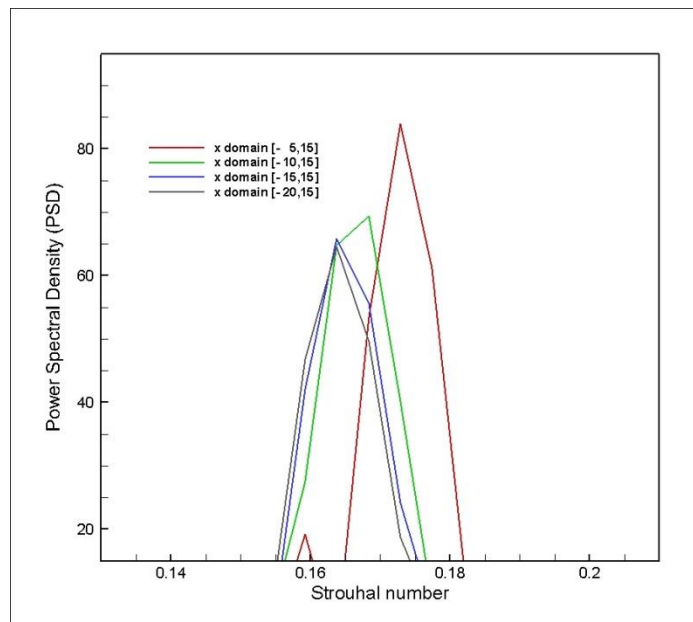


Figure 5-12: The power spectral density of the Lift coefficient- Effect of the Size of the fluid domain in front of the cylinder in x direction on the lift coefficient.

It is worth mentioning that some of the differences in the results reported by different researcher could be explained by this parameter. For example Choi et al. 2007, used a grid with the dimension of the $80D \times 80D$ for their simulation and obtained 1.34, 0.315 and 0.164 for Drag, lift coefficient and Strouhal number, while Lai and Peskin 2000 used a computational domain with 6D in front of the cylinder and reported a higher value for the drag coefficient (see Table 5-3).

5.1.3 Parametric study – Blockage effect

In the case of flow passing a bluff body, the blockage effect was reported by Karniadakis and Triantafyllou 1992, who showed that the simulation results will be affected by the size of the computational domain in the cross flow direction. On the other hand, solving the flow governing equations in a very large domain is very expensive and might not improve the results noticeably. In this section a parametric study is carried out to determine the minimum domain size for which the blockage effect is negligible. To achieve this goal, the flow (low Reynolds number, $Re=100$) over a circular cylinder with diameter D , is simulated for 5 different domain sizes from $10D$ ($[-5,5]$) to $50D$ ($[-25,25]$) in y direction (perpendicular to the flow, x direction). In this problem the cylinder is located at $(x,y)=(0,0)$ and has equal distance to the domain's upper and lower boundaries.

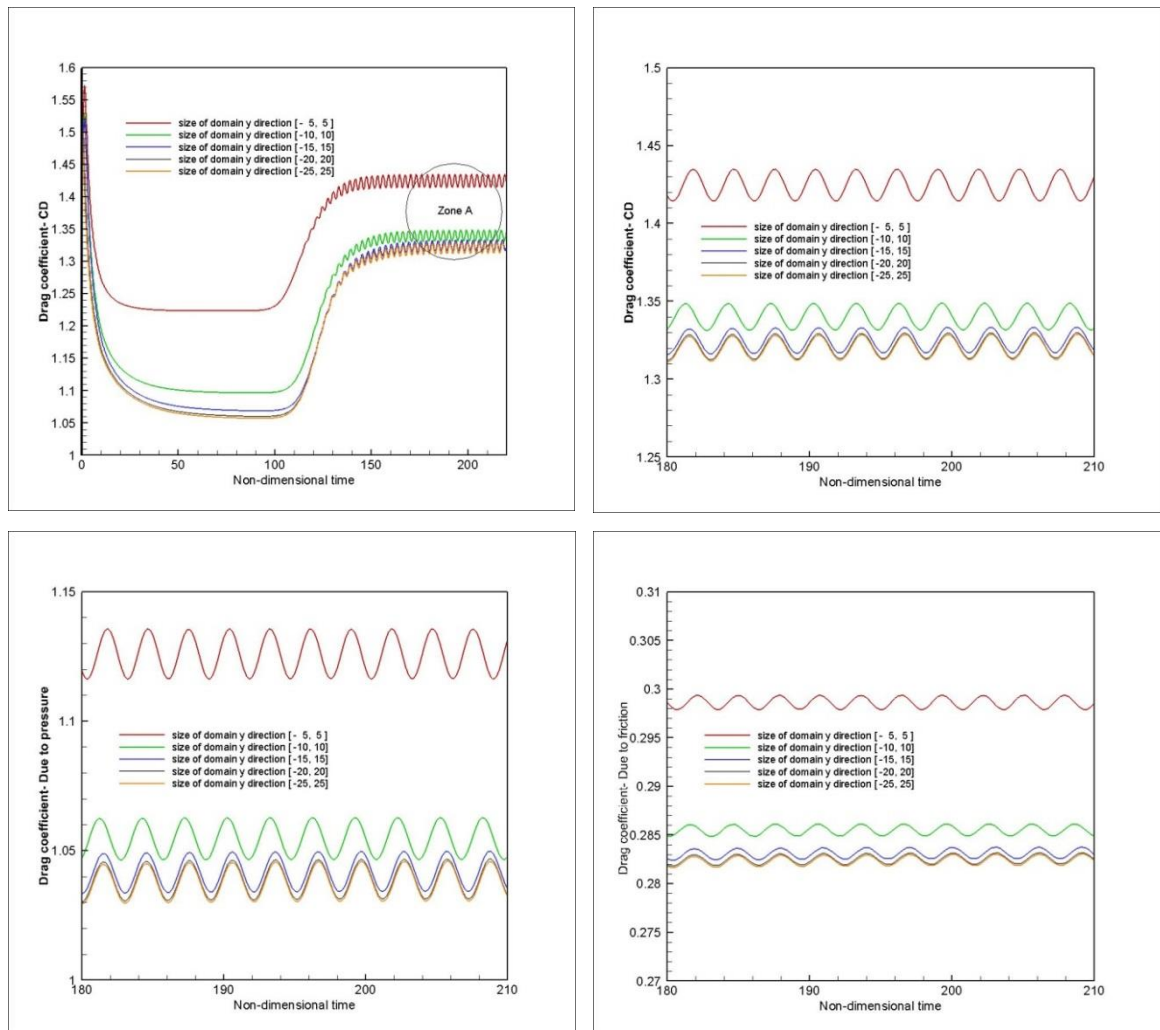


Figure 5-13: Effect of the size of the computational domain in the y direction on the drag

The Hydrodynamic quantities (lift and drag) and Strouhal number for these cases are compared. Other computational domains parameters remain constant during the simulation; the size of the domain in the x direction is $[-15D,15D]$; the size of uniform grid area around the cylinder is $[-2D,4D] \times [-2D,2D]$ in the x and y directions respectively where $dx=dy=0.025$. The grid is stretched by a factor of 3 from the uniform area around the cylinder to the domain boundaries. The numerical results show that the blockage effect significantly affects the hydro-dynamical quantities in the small domain. For instance, the mean drag and maximum lift coefficients for the domain with $y \in [-5D,5D]$ are 1.43 and 0.322, respectively while for the domain $y \in [-10,10]$ these quantities are 1.34 and 0.285, respectively. These results show that at $Re=100$ if the size of domain is doubled from 10D to 20D in the cross flow direction the lift and drag coefficients decrease by about 7 % and 12% respectively.

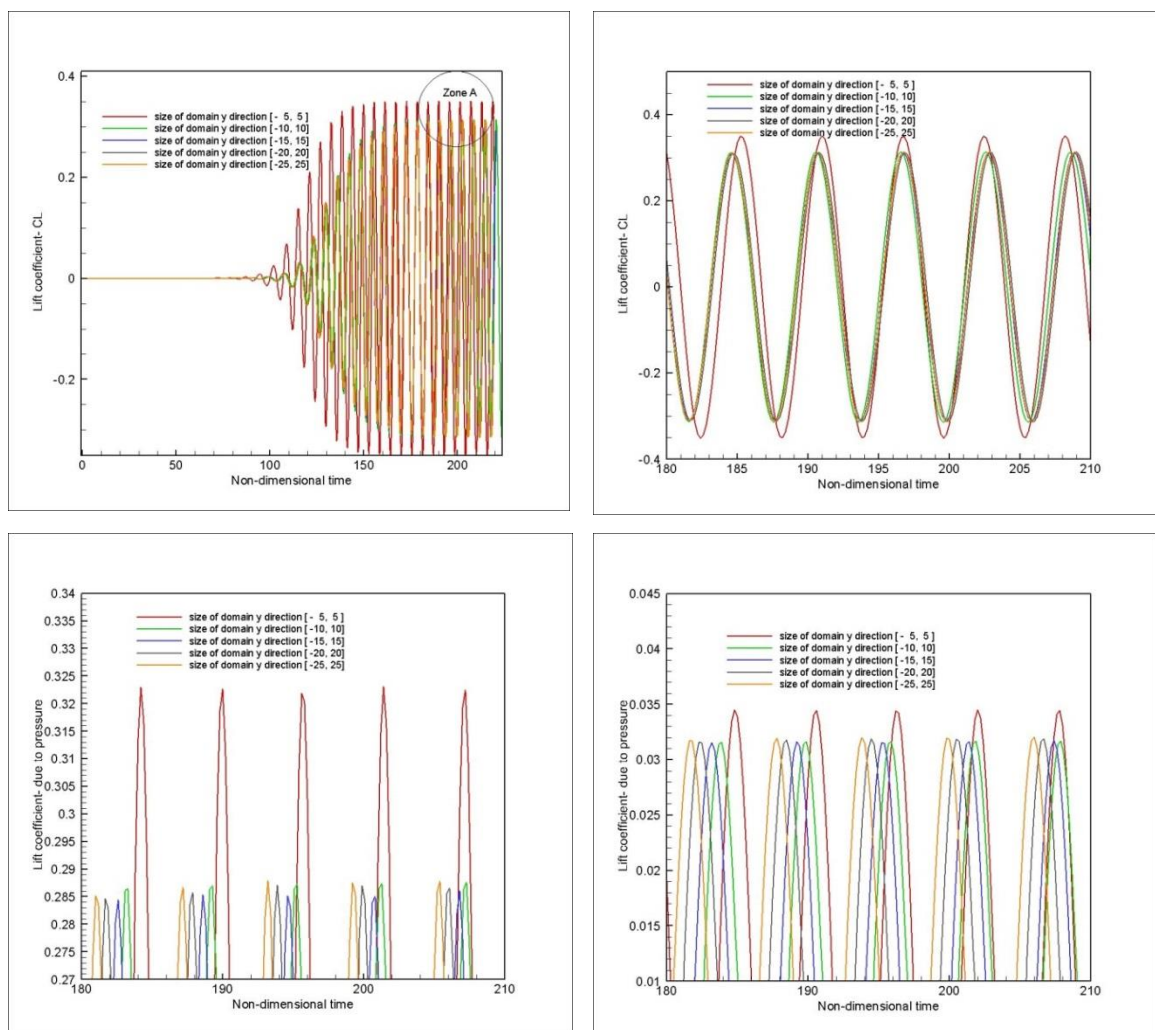


Figure 5-14: Effect of the Size of the fluid domain in y direction on the Lift coefficient.

However, as shown in Figure 5-13, for the domain that is larger than $[-15D, 15D]$ the blockage effect on the lift and drag coefficient is very limited. For instance, by enlarging the vertical direction of the domain from $30D$ to $50D$, the drag coefficient only decreases by less than 1%, and the change in the lift coefficient is negligible. Also, Figure 5-13 shows that the blockage has similar effect on the drag coefficient, the drag due to the pressure and the shear stress.

Figure 5-14 shows the blockage effect on the lift coefficient at low Reynolds number. The numerical simulation shows that enlarging the domain in the cross flow direction by more than $20D$ will not affect the lift coefficient.

In addition, the numerical results (Figure 5-14) show that the lift coefficient due to the pressure is more affected by the size of the domain in the y direction than the lift coefficient due to the shear stress. For instance, by doubling the size of domain in the cross flow direction from $10D$ to $20D$, the lift coefficient due to pressure and shear stress decrease 12.3%, and about 11%, respectively.

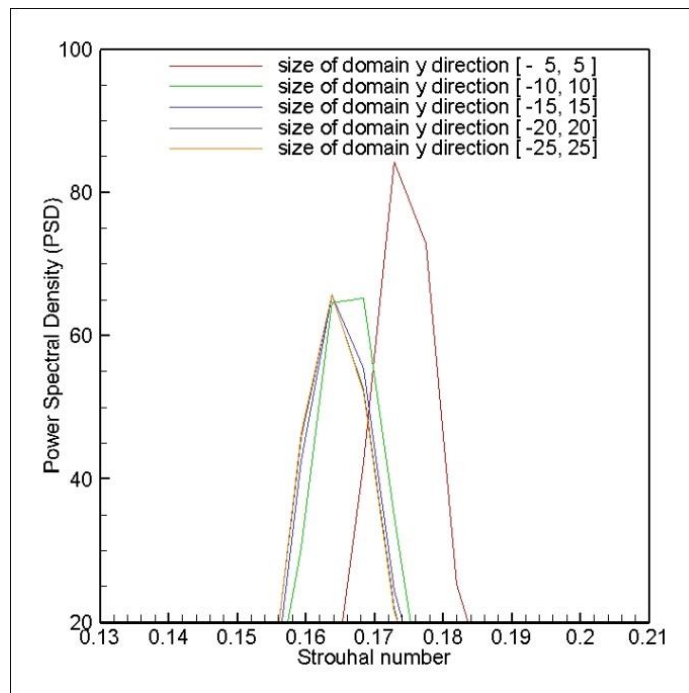


Figure 5-15: Power Spectral Density (PSD) of the Lift coefficient - Effect of Size of the domain in y direction on the lift coefficient.

The Strouhal number is affected by the blockage effect as well as lifts and drag coefficient. Figure 5-15 shows the power spectral density (PSD) of the lift coefficient for the flow around a cylinder at $Re=100$ for five different domain sizes in the y

direction from 10D to 50D. The numerical results show that for small domains the blockage effect is more severe. For instance, by increasing the domain from 10D (the case [-5,5] in Figure 5-15) to 20D (the case [-10,10] in Figure 5-15) the Strouhal number changes from 0.175 to 0.168. However, for the domain that is larger than 30D (the case [-15,15] in Figure 5-15) in the cross flow direction, the Strouhal number remains about 0.164 and does not change when further enlarging in the size of the domain in the y direction.

Figure 5-16 to Figure 5-18 show that at low Reynolds numbers, if the size of domain is more than 30D in the cross flow direction, the blockage effect on the lift, drag and Strouhal number is negligible. The drag coefficient for a cylinder in the cross flow direction at $Re=100$, was reported to be 1.44 (by Corbalan & de Souza 2010) and 1.33 (by Kim et al. 2001); the blocking effect might be one of the reasons for this difference.

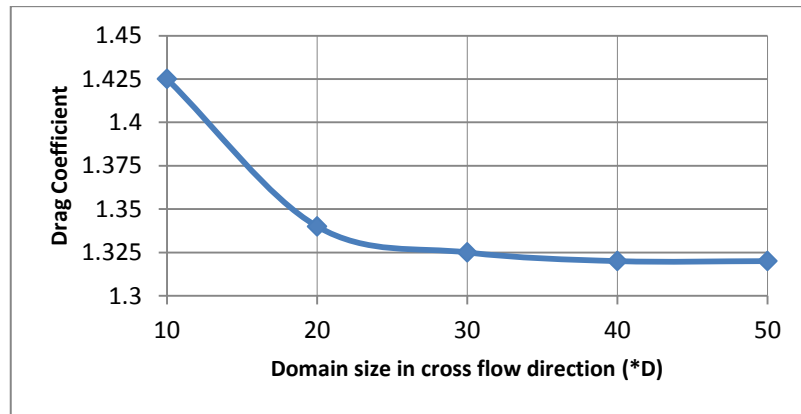


Figure 5-16: Drag coefficient verse domain size in cross flow direction,

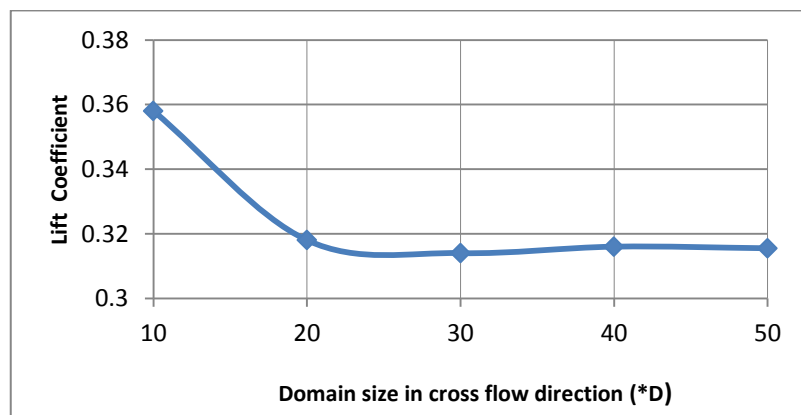


Figure 5-17: Lift coefficient verse the size of domain in perpendicular direction to the main stream velocity (cross flow direction).

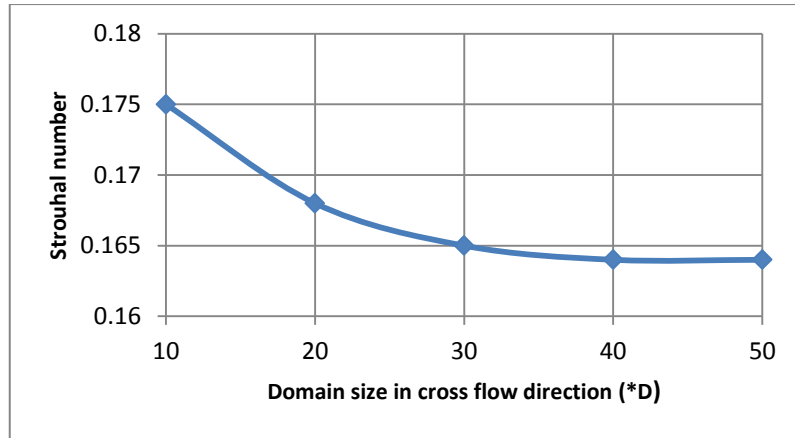


Figure 5-18: Strouhal number verse the size of domain in perpendicular direction to the main stream velocity (cross flow direction).

5.1.4 Parametric study – Stretching factor

According to the mesh refinement study for the flow around a bluff body (Section 5.1.1) in order to obtain accurate hydro-dynamical forces, a fine grid around the immersed boundary is recommended. On the other hand, to minimize the blockage and entrance length effects (Section 5.1.2 and 5.1.3) on the simulation results a relatively large computational domain is needed. These issues lead to high computational costs. A stretching technique allows refining the grid near the IB, while using a coarse mesh in the outer region to ensure that the computational domain is sufficiently large. In this way the number of grid points and the computational resources needed are minimised without compromising the accuracy of the simulation. In this section the effect of the stretching factor on the lift, drag and Strouhal number is presented.

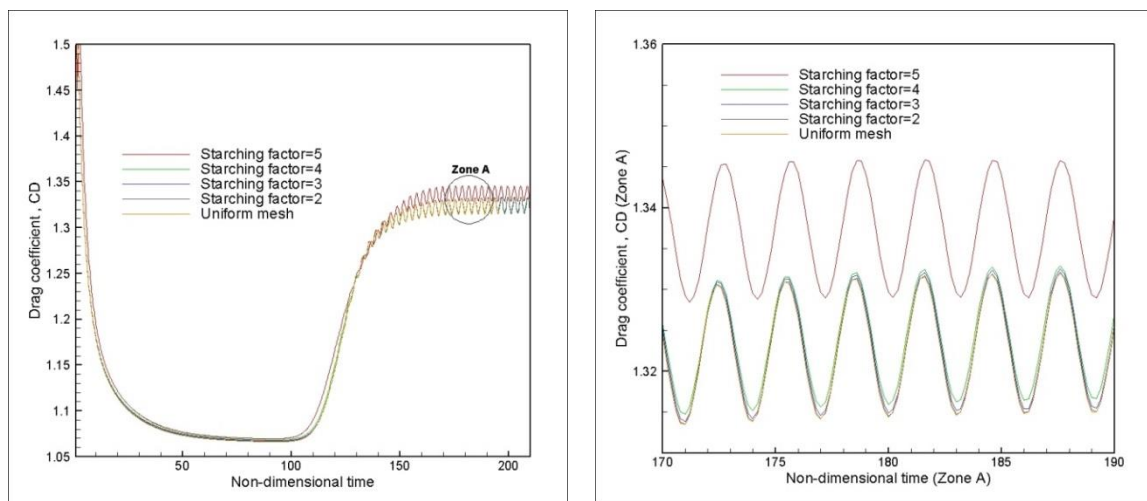


Figure 5-19: Effect of the grid stretching factor on the Drag coefficient

To study the effect of the stretching factor, the flow around a stationary cylinder at low Reynolds number, $R=100$ is simulated on a uniform grid using four grids with stretching factors ranging from 2 to 5. Other domain parameters remain constant; the size of domain in both the x and y directions is $30D$, the size of uniform grid area around the cylinder is $6D$ ($2D$ in front and $4D$ after) in the x direction and $4D$ ($2D$ on each side) in y the direction. The size of the grid in the uniform grid area around the immersed boundary is $dx=dy=0.025D$.

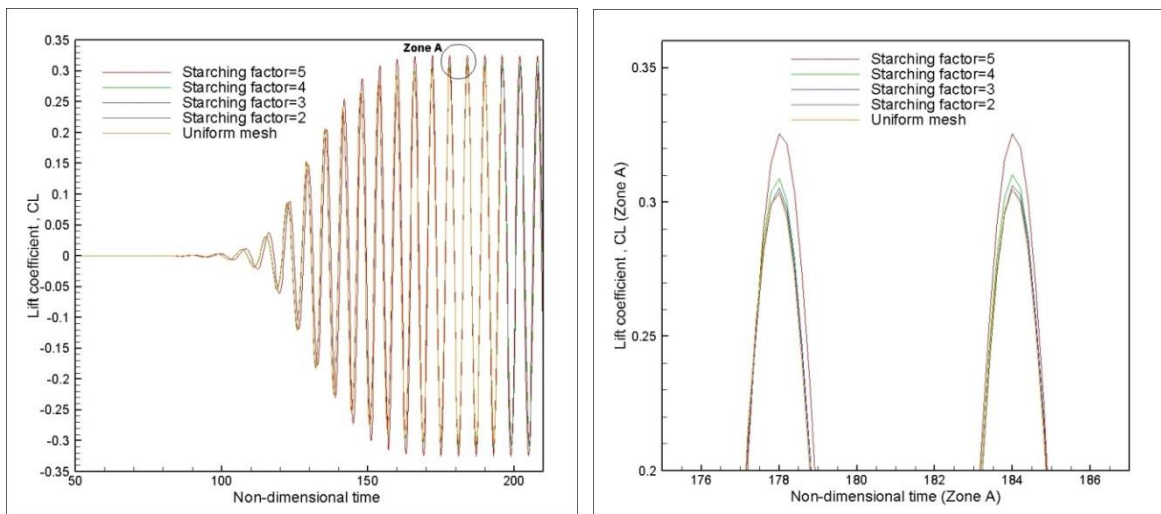


Figure 5-20: Effect of the grid stretching factor on the Lift coefficient.

Numerical results show that the high stretching factor could slightly affect the drag and lift coefficients and Strouhal number. Figure 5-19 and Figure 5-20 show that a stretching factor of 5 leads to a mean drag of 1.3375 and max lift of 0.327 which is about 1% higher than the values obtained when using a stretching factor of 4, where the mean drag coefficient is about 1.3225 and maximum lift is about 0.31. In the case where the stretching factor is less than 4, the drag and lift coefficients are hardly affected by the stretching factor and the simulation results are matching well with the results on the uniform grid.

According to the Table 5-2, at $Re=100$ the grid-stretching could significantly reduce the number of nodes and hence the computational expense while the lift, drag and Strouhal number have a small effect of only one percent.

Table 5-2: parametric study of the Stretching factor, minimum grid size is 0.025D, the domain size [-15,15] in x and y direction, and the uniform domain size [-2,4]in x and [-2,2]in y direction.

Stretching factor	No. grid (x direction)	No. grid (y direction)	Total No.	CD(mean)	CL(Max)	Strouhal number
5	310	235	72'850	1.3375	0.327	0.166
4	385	317	122'045	1.3225	0.31	0.164
3	531	475	252'225	1.3215	0.305	0.164
2	771	735	566'685	1.3215	0.305	0.164
Uniform grid	1200	1200	1'440'000	1.3215	0.305	0.164

According to the Figure 5-21, the stretching factor hardly affects the Strouhal number (the frequency of vortex shedding around a cylinder) at low Reynolds numbers. It can be seen in this figure that the Strouhal number changes from 0.166 to 0.164 when the stretching factor changes from 5 to 4, however there is hardly any variation in the Strouhal number when the stretching factor becomes less than 4.

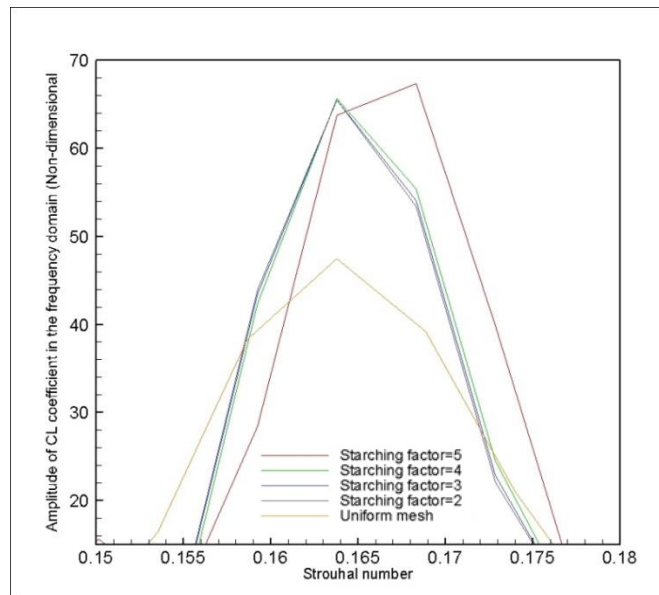


Figure 5-21: Effect of the grid stretching factor on Strouhal number.

5.1.5 Parametric study – size of uniform area, x direction after cylinder

In the simulation of free flow passing a circular cylinder at low Reynolds number, the gradient of the velocity and pressure are relatively high near and in the wake of the

cylinder. Therefore, a higher resolution grid (dense mesh) is necessary in these areas in order to obtain an accurate simulation. A coarser grid should be sufficient to resolve the far field where the flow parameters do not vary very much. The size of uniform grid around the cylinder should be large enough to guarantee the accuracy of the results and it should be small enough to minimise the computational costs. In this section the optimum size of this area downstream of the cylinder (dimension “f” in the Figure 5-2) is investigated. The size of the uniform mesh before the cylinder in the x direction (dimension “e” in the Figure 5-2) and the size of the symmetric uniform mesh in the y direction (dimension “g” in the Figure 5-2) are presented in Sections 5.1.6 and 5.1.7 respectively. The flow around a cylinder at $Re=100$ is simulated in two dimensions. The overall size of the computational domain in the x and y directions is taken as $[-15,15]$. The size of the grid in the uniform area is $dx=dy=0.025$.

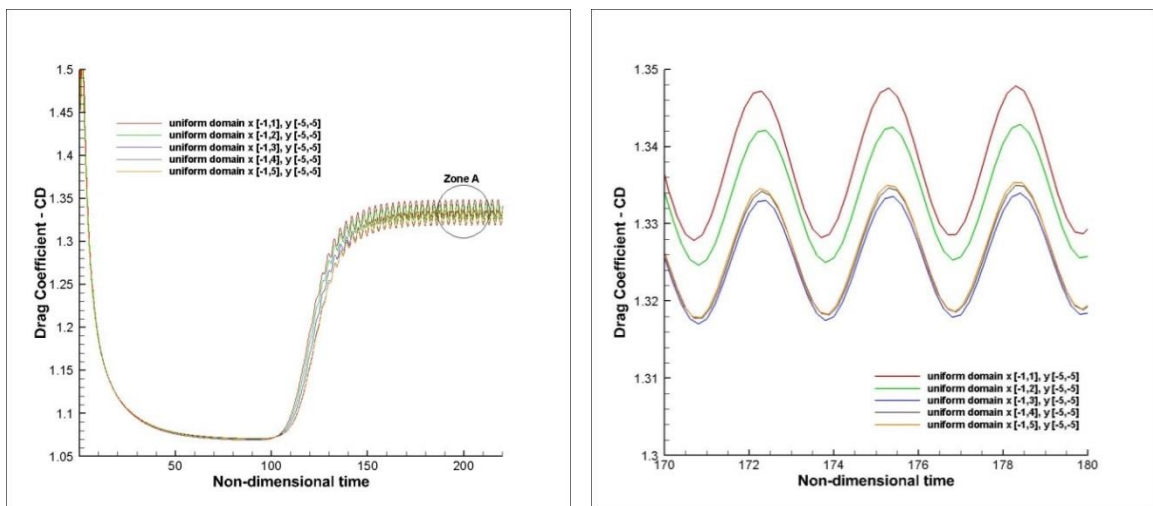


Figure 5-22: Effect of the uniform area after the circular cylinder in x direction on the Drag coefficient.

The size of the uniform area after the cylinder is changed from 1D to 5D while the rest of the domain parameters remain constant. In this study the uniform domains $[-1,1]$, $[-1,2]$, $[-1,3]$, $[-1,4]$, $[-1,5]$ in the x direction are compared. The uniform grid area around the cylinder in the y direction is maintained at $[-5,5]$.

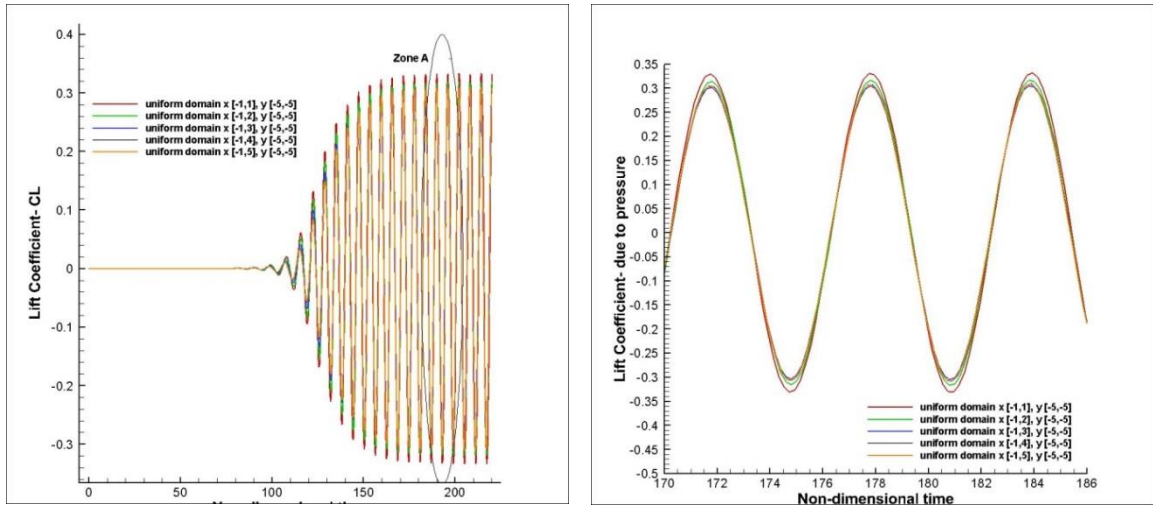


Figure 5-23: Effect of the uniform grid area after the circular cylinder in x direction on the lift coefficient.

Figure 5-22 shows the effect of the length of the uniform mesh after the cylinder on the overall drag coefficient. The simulation results show about one percent increase in the drag coefficient when the uniform area after the cylinder is changed from 5D $[-1,5]$ to 1D $[-1,1]$. However, when this size is changed from $[-1,3]$ to $[-1,5]$ the changes in the drag coefficient were found to be negligible. Both the drag due to pressure and friction were found to behave in a similar way.

Figure 5-23 shows the results of varying the length of the uniform grid area behind the cylinder on the lift coefficient. The maximum lift coefficient was found to reduce from 0.333 to 0.307 when comparing case $[-1,1]$ to case $[-1,5]$ respectively, which is about 8 percent. However, the difference just changed by less than one percent when the uniform grid size ahead of the cylinder changes from 3D to 5D. A similar trend is observed for the lift coefficients due to pressure and shear stress. In other words, the uniform size $[-1,3]$ in the x direction is a good choice to obtain accurate results for both the lift and drag coefficient.

Figure 5-24 shows the power spectral density (PSD) of the lift coefficient in the frequency domain for different uniform sub-grid length after the cylinder in the x direction. The results show that this parameter does not affect the Strouhal number as all cases show the same frequency for the lift coefficient.

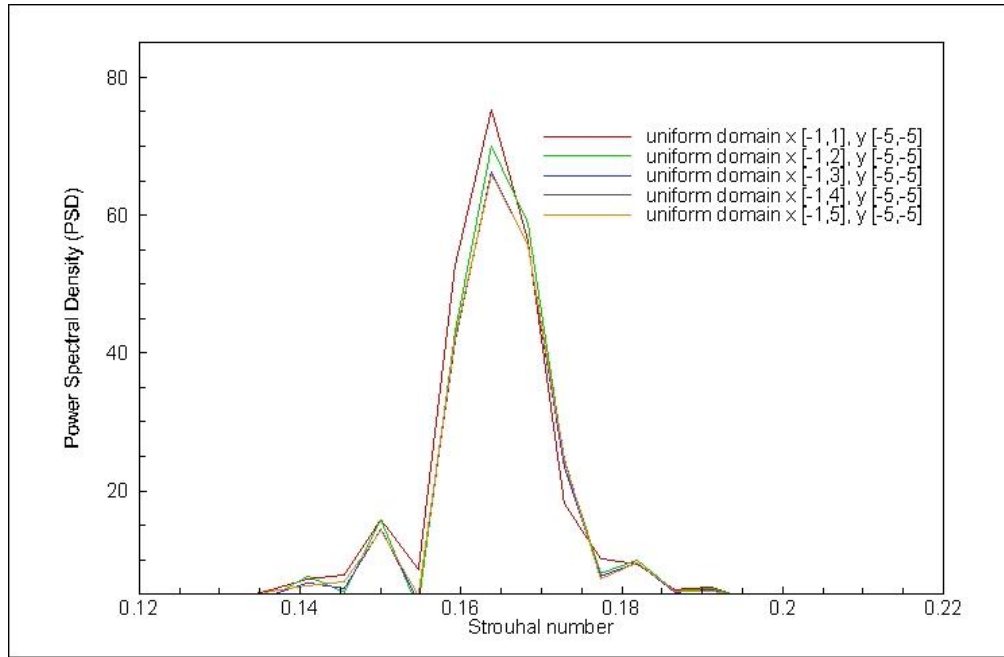


Figure 5-24: Power Spectral density (PSD) of the Lift coefficient - Effect of the uniform Size of the fluid domain after the circular cylinder in x direction on the Strouhal number.

5.1.6 Parametric study- uniform size x direction before cylinder

In this section the effect of the uniform grid size in front of the cylinder in x direction (dimension “e” in the Figure 5-2) on the hydrodynamic forces is investigated. The size of the uniform grid length in front of the cylinder is changed from 1D to 5D while the rest of the domain parameters remain unchanged. The size of the uniform domain in the y direction is [-5,5]. Figure 5-25 shows that the mean drag coefficient decreases from 1.3285 to 1.325 when the uniform area in front of the cylinder increases from 1D to 2D, respectively, which is about 0.3%. However, if the size of the uniform grid area in front of cylinder is longer than 2D, the effect of this parameter on the mean drag coefficient is absolutely negligible. The simulation results show a similar effect on the drag due to pressure and due to shear stress. Therefore, if the size of the uniform grid in front of the cylinder is taken to be longer than 2D, the effect of this parameters can be neglected at Re=100.

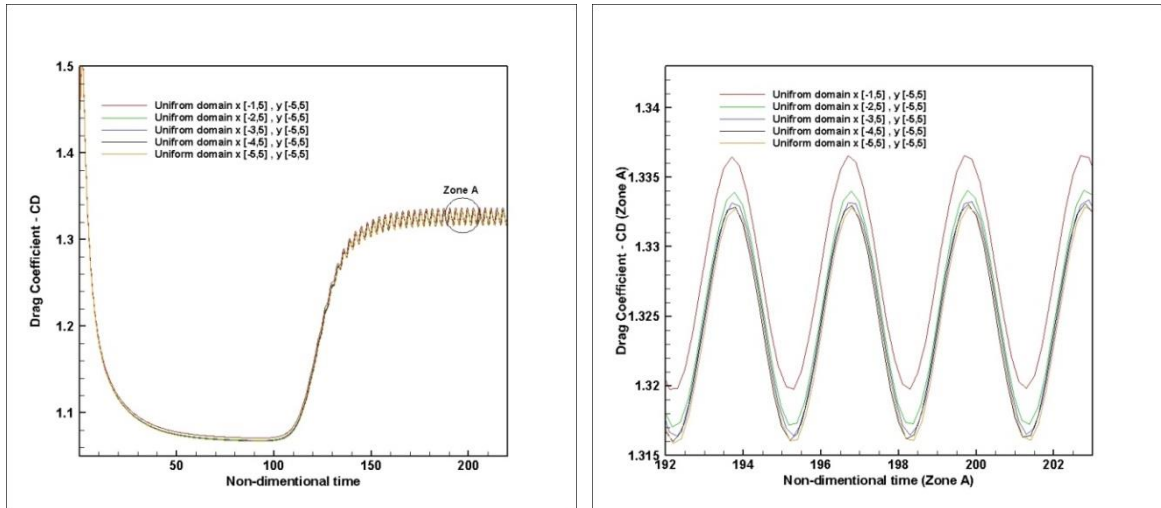


Figure 5-25: Effect of the uniform grid in front of the circular cylinder in the x direction on the drag coefficient.

Figure 5-26 hardly shows any changes in the lift coefficient due to changing the uniform grid in front of the cylinder from 1D to 5D. Also the simulation results hardly show any changes in the lift due to pressure and shear stress by changing this parameter. In addition, the power spectral density of the lift coefficient results shows that the Strouhal number is not affected by this parameter either.

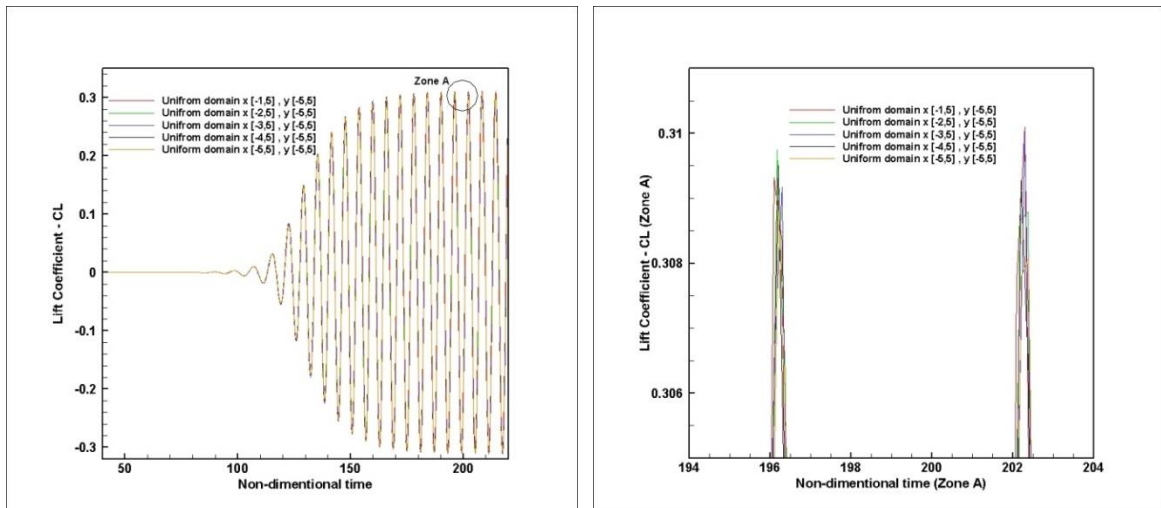


Figure 5-26: Effect of the uniform grid in front of the circular cylinder in x direction on the Lift coefficient.

5.1.7 Parametric study- uniform grid area in y direction

In this section the effect of the size of the uniform grid area around the cylinder in the y direction (dimension “g” in the Figure 5-2) on the lift and drag coefficient is presented. Here, this parameter is changed from 2D ([-1,1]) to 10D ([-5,5]), while the rest of the domain parameters remains constant. The total size of the grid in both the x

and y directions is 30D, the uniform grid size in the x direction around the cylinder is [-1D,5D] and the size of the grid cells around the cylinder is $dx=dy=0.025D$.

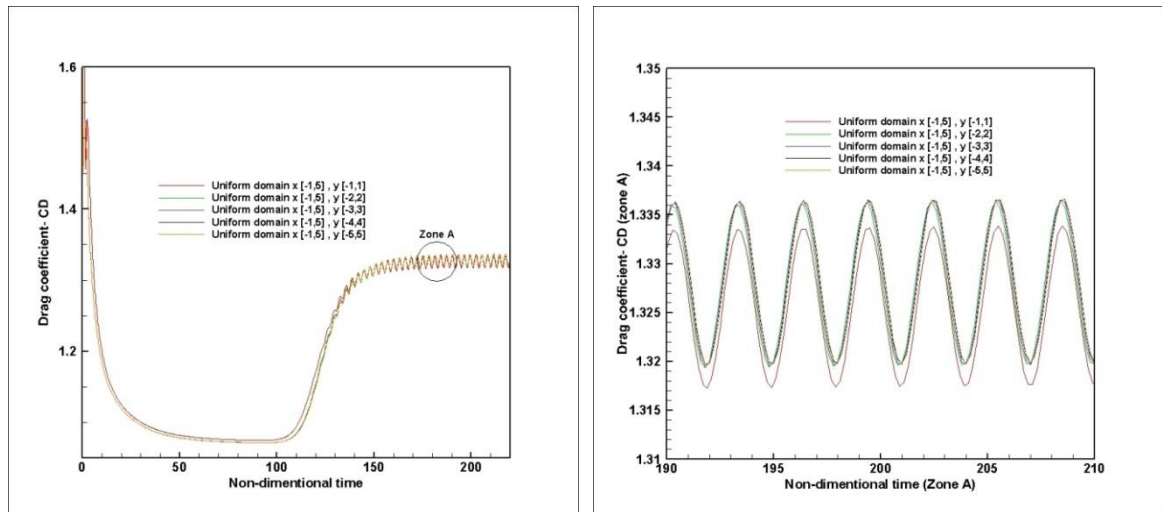


Figure 5-27: Effect of the vertical extend of the uniform area around the circular cylinder on the Drag coefficient.

Figure 5-27 and Figure 5-28 show that the drag and lift coefficients are hardly affected by increasing the size of the uniform area around the cylinder in the y direction beyond [-2,2].

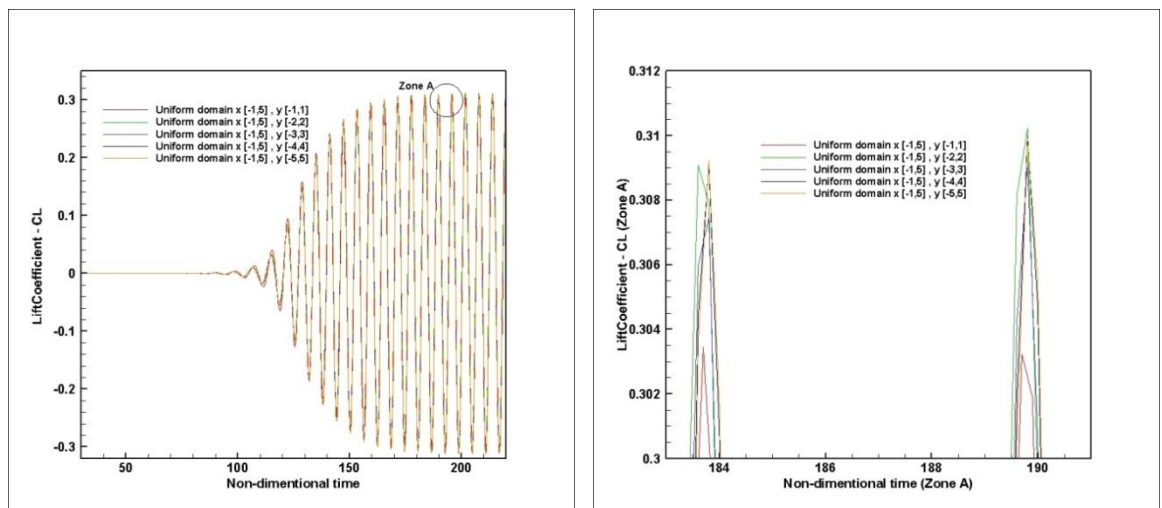


Figure 5-28: Effect of the vertical extends of the uniform area around the circular cylinder on the Lift coefficient.

According to Figure 5-27, by increasing size of the uniform grid area from [-1,1] to [-2,2] in the y direction the mean drag coefficient increases from 1.325 to 1.33

respectively, which is less than 0.3 percent. Also, according to Figure 5-28, the maximum lift coefficient changes from 0.303 to 0.309 when the size of the uniform grid in the y direction is changed from [-1,1] to [-2,2] which is less than 0.6%. The simulation results show a similar trend for the lift and drag due to pressure and shear stress. Moreover, the power spectral density (PSD) of the lift coefficient shows that the Strouhal frequency is not affected at all by changing the size of uniform grid area in the y direction.

5.2 Validation

In this section, the numerical code and the IB interpolation method are validated by comparison with other numerical and experimental results presented in the literature. The flow around a stationary circular cylinder at a low Reynolds number of $Re=100$ is chosen as a bench mark. According to the parametric study presented, the domain sizes in the x and y directions are selected as $[-15D, 15D]$ while the uniform grid area in the x and y directions around the cylinder is $[-2,4] \times [-2,2]$ (see Figure 5-29). The grid size in the uniform area is $0.025D$ and the grid is stretched towards the computational boundaries by a stretching factor of 3. The numerical results for the lift and drag coefficients and the Strouhal number are compared with those given the literature.

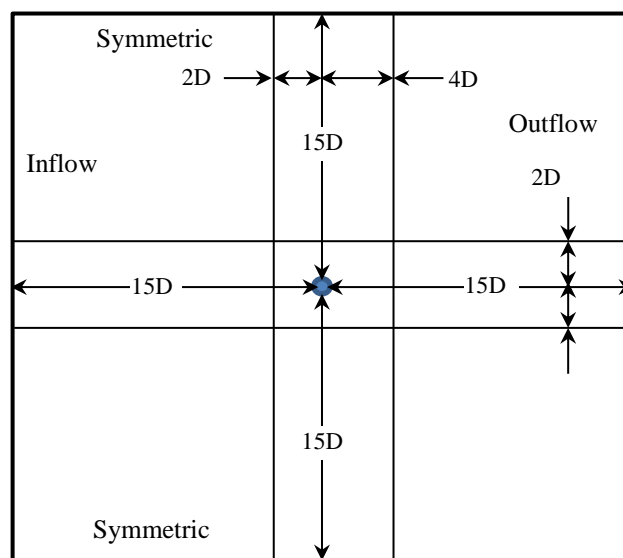


Figure 5-29: Schematic of the computational domain

A Dirichlet boundary conditions ($U_{\infty}=1$ and $V=0$) is applied at the inflow; to model the far field a symmetric boundary ($U(i,ny)=U(i,ny-1)$ and $V(i,ny)=0$) is used while a convective boundary condition is applied at the outflow.

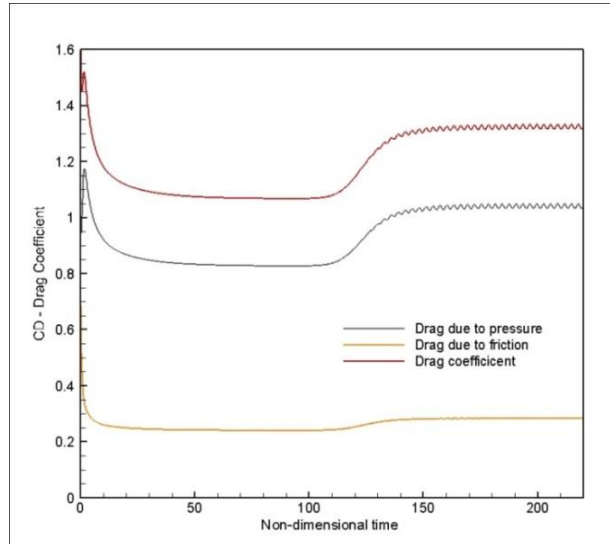


Figure 5-30: Drag coefficient, Drag due to pressure and friction for a stationary cylinder at $Re=100$ versus non dimensional time.

Figure 5-30 shows the drag coefficient plotted against the non-dimensional simulation time. The results took about 150 non-dimensional time units to reach the steady state solution without adding an external perturbation to trigger the vortex shedding. The drag due to the shear stress and the pressure are integrated around the cylinder to obtain the total drag coefficient.

At $Re=100$ the mean drag coefficient, the drag due to pressure and shear stress are 1.325, 1.05, 0.275 respectively. Also, according to the Table 5-3 the results match the experimental and other numerical results very well.

Figure 5-31 presents the numerical results for the lift coefficient at $Re=100$. The results show that the amplitude of the lift due to the pressure dominates the lift coefficient. The simulation results show that the amplitude of lift coefficient, lift due to pressure and shear stress are about 0.31, 0.282 and 0.03, respectively. The pattern of the lift and drag coefficient and also the Strouhal number ($St=0.164$) are matching very well with the literature.

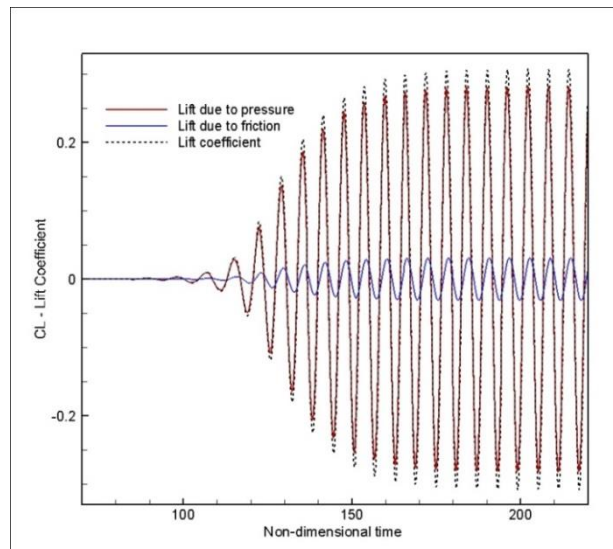


Figure 5-31: Lift coefficient, the lift due to pressure and friction for a stationary cylinder at $Re=100$ versus non dimensional time.

The flow around a cylinder is a well-known test case which has received a great deal of attention in the literature. Table 5-3 shows the results of a few of these studies (more details are presented on chapter 6). Interestingly, the reported results show differences of about 10% and 7% between the reported drag and lift coefficients respectively. At first glance it looks like these differences are due to the various algorithms and approaches that have been applied to solve the problem. Some of the differences are caused by the use of various computational domain sizes. Apart from the mesh sizes that were employed, also the size of the computational domain maybe one of the causes of these differences. The size of domain could affect the results in three ways, either the domain is not high enough to prevent the blockage effect, or the length of the domain before the cylinder is not large enough to prevent an inflow effect or the size of the domain after the cylinder is not large enough to be able to neglect the outflow affecting the simulation results. For instance, the size of the domain in front of the cylinder in the case of Corbalan & de Souza 2010 is $6.5D$ and in the case of Lai & Peskin (2000) is about $6D$ ($1.85/0.3$), the higher mean drag that is predicted in comparison to the other cases (for instance Kim et al. 2001) was due to the use of the relatively small inflow domain used by former researcher in front of the cylinder.

In addition, in the simulation of Lima E Silva et al. 2003, the inflow length is sufficiently long but the blockage effect due to the limited vertical extend of the domain causes the drag coefficient increases to 1.39.

According to the parametric study conducted in this chapter for the flow around the cylinder at $Re=100$, the size of the domain before the cylinder should be more than $15D$ and the size of the domain in the cross flow direction should be more than $30D$ in order to ensure that domain size does not influence the drag coefficient. In all cases, the size of domain after the cylinder was long enough to prevent an additional effect on the drag coefficient from the outflow boundary condition. According to Table 5-3, the size of the computational domain has a similar effect on both lift and Strouhal number as well as on the drag coefficient (more results are presented on Table 6-2).

Table 5-3: Drag, lift and Strouhal number for present study and well known numerical and experimental studies for the flow around a circular cylinder at low $Re=100$.

	Domain before cylinder	Domain size in y direction	Domain size in x cylinder	D	CD	CL	St.
Corbalan & de souza 2010 (IB- Force)	6.5D	15D	19.2D	1	1.44	± 0.31	-----
Lima E Silva et al. 2003 (IB- PVM)	16D	15D	30D	1	1.39	-----	-----
Lai & Peskin 2000 (IB- Force)	1.85	8	8	0.3	1.447	± 0.329	0.165
Kim et al. 2001 (IB- Force+ mass source)	-----	100D	70D	1	1.33	± 0.32	0.165
Roshko 1954 (experiment)	-----	-----	-----	-----	-----	-----	0.164
Williamson 1988 (experiment)	-----	-----	-----	-----	-----	-----	0.166
Present study (IB – Interpolation)	15D	30D	30D	1	1.33	± 0.31	0.164

5.3 Summary

Fluid Structure Interaction (FSI) has received a great deal attention in the recent decades and many approaches have been adopted to solve this problem. In this thesis, the focus is on the Immersed Boundary approach with interpolation/reconstruction methodology. On the one hand, the IB method makes it possible to model FSI problems with complex boundary and large structural displacement, on the other the IB method needs special care and a high mesh resolution near the immersed boundary. In general FSI problems, and in particular IB approach, are relatively expensive and therefore a selection of optimum parameters to model this problem is important. In this chapter the developed methodology and code is validated and a comprehensive parametric study is

conducted for flow around a stationary cylinder at low Reynolds number, $Re=100$. This particular case is a well-known benchmark and several experimental and computational results are reported in the literature. It is shown that the size of the domain significantly affects the flow parameters and this could be a potential reason for some of the discrepancies in results reported in the literature. Numerical simulation results show that if the size of the domain is increased from $5D$ to $10D$ before the cylinder, the lift and drag coefficients decrease by about 10%. However, a further enlargement of the domain does not change these values any more. In addition, the size of the domain in the cross flow direction is important. When the size of the domain in the y direction is increased from $10D$ to $25D$, the lift and drag coefficients decrease by about 10%.

However, numerical results show that the sizes of the uniform grid patch around the cylinder and the grid stretching factor only have a limited effect on the lift and drag coefficients. The results show that any size of the uniform grid area in the y direction larger than $[-2,2]$ does not affect the results. Also, the uniform size in the x direction is proposed to be $[-2,4]$ to limit affecting the flow parameters. Also, stretching factor less than 4, have very limited influence on the lift and drag coefficients.

In addition, the Strouhal number and lift and drag coefficients for the optimum domain sizes were compared with the reported values in the literature. All the parameters were found to be in very good agreement with the numerical and experimental results reported elsewhere.

In the next chapter, the present IB approach is compared with alternative interpolation/reconstruction methods reported in the literature.

Chapter 6. Comparative study of the interpolation methods - Stationary cylinder

Peskin 1972 introduced the Immersed boundary (IB) approach. In this method, using a Cartesian grid the solid boundary is imposed on the flow by adding a forcing function to the flow equations. Since 1972, Peskin's method has been developed further by many researchers (see Chapter 3 for more details) and most of the modern immersed boundary approaches use an interpolation procedure to enforce the non-grid conforming boundaries. In the direct forcing approaches, the interpolation is used to implement the forcing functions at the interface cells in order to enforce the immersed boundary on the governing equations. In the IB interpolation method, the forcing function, f , which is needed to enforce the boundary conditions is not calculated directly; but instead, the flow velocity is interpolated at the interface cells and the solid boundary is imposed indirectly on the discrete equations. The interface points are identified as those points in the fluid domain whose at least one of its neighbouring points is inside the solid domain. Therefore, the flow parameters (i.e. velocities and pressures) related to these points cannot be updated directly by the governing equation (Figure 3-2right). Any cells that contain one or more interface points are called the interface cells. In the indirect forcing approach (interpolations approach), at every time step the flow parameters in the interface cells are updated by direct interpolation formulas and the results are used as the boundary condition in the flow solver. In this chapter, the flow around a circular cylinder at low Reynolds number is selected as a bench mark and four IB interpolation/reconstruction methods which have been introduced previously in the literature review chapter are compared with the proposed interpolation method in this research.

6.1 Governing equation and computational domain

The unsteady, incompressible Navier-Stokes equations (4-3) are used as the governing equations. A staggered variable arrangement, as introduced by Harlow and Welch 1965, is used to discretize the governing equations on a Cartesian grid (equations (4-11) to (4-20)). The continuity equation is enforced by taking the divergence of the momentum equations to form a Poisson equation for the pressure (equations (4-32) to

(4-36)). The governing equations were solved by a two steps fractional method (equations (4-22) and (4-23)).

Based on the parametric study conducted in chapter 5, the size of the computational domain is selected in a way to ensure that the boundaries have limited effect on the simulation results (Figure 6-1). Therefore, the size of the domain in y and x directions are taken to be $20D$ and $15D$, respectively and the grid size is chosen to be $dx=dy=0.05D$ which is the coarsest grid that gives acceptable results (according to the mesh refinement study presented in section 5.1.1).

Since the entire domain is meshed using a uniform Cartesian grid, the implementation of the grid-conforming inlet, outlet and side boundary conditions was straightforward and the boundary conditions along the circular cylinder are implemented using five different immersed boundary interpolation methods.

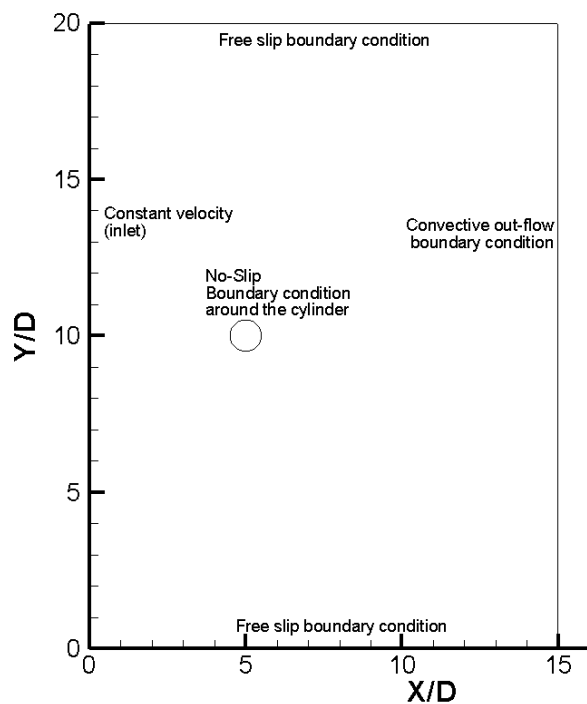


Figure 6-1: Fluid domain size and boundary conditions.

6.2 Interpolation method cases

Several methods are used in the literature to interpolate/reconstruct the velocity in the boundary cells near the immersed boundary (section 3.3). Four interpolation methods plus the interpolation method introduced in this thesis are compared to one another. To do so, the first step is to define the interface cells in the specific geometry, which could be complicated for geometries with unknown analytical functions (Iaccarino & Verzicco

2003). Here, the flow around a stationary circular cylinder at low Reynolds number, $Re=100$ is selected as a bench mark. The next step is to determine the interpolation formulas for each individual interpolation method. These formulas will be used to update the flow parameters (velocities and pressure) in the interface cells adjacent to the solid boundaries. The flow solver uses these values as the boundary conditions for the rest of the flow domain. In the following part, these interpolation methods are explained briefly.

6.2.1 Case A: No interpolation

The simplest possible method is to add the interface cells to the solid domain. In this case an interpolation is not needed and the solid boundary assumes a stepwise shape (Figure 3-5a). The immersed boundary is diffused, in the staggered variable arrangement as the velocity components are defined at different sides of an element. Fadlun et al. 2000 proposed a similar method for imposing forcing functions for the immersed boundaries. As no interpolation is conducted in this method, it is expected to be relatively faster while still giving acceptable results. In the case of a moving body (displacement/ deformation) this method is potentially more efficient as the interpolation formulas do not need to be updated in the course of the displacement/deformation. In the simulation of the flow around a complex geometry with curved boundaries, this method could lead to inaccurate results for the lift and drag coefficient when using relatively coarse grids. On a fine grid this method could give more accurate results, but this would compromise the advantage of the method which is the lower computational demand.

6.2.2 Case B: Weighting method

This method is similar to the one discussed above as Case A. The major difference is that the values for the velocities in the boundary cells are associated with the area of the cell which is covered by the fluid over the total cell area. In this method the area of cells which are common between the fluid and structures are used to calculate this weighting coefficients. Figure 3-5 (right) shows the location of these weighted boundary velocities in the cells that are part fluid and part solid. For each of the velocity components a coefficient is determined that corresponds to the ratio of the fluid part of the two adjacent cells to the whole area of the two cells.

6.2.3 Case C: linear interpolation method

The third method is a linear interpolation method where the velocities in the interface cells are calculated by interpolating between the velocity at the solid boundary applying the no slip condition and one point in the fluid domain. Fadlun et al. 2000 suggested using this interpolation method to enforce the boundary condition to the fluid domain in the forcing IB approaches. The linear interpolation is ideal for the problems in which the immersed boundaries are parallel to the Cartesian grids lines. The advantage of this method is that the interpolation formula is simple as the interpolation points coincide with grid nodes on the Cartesian coordinates where the velocities are defined in the discretised governing equations; however for the inclined and curved immersed boundaries the interpolation direction (either x or y direction in two dimensional simulation) might slightly affect the simulation results.

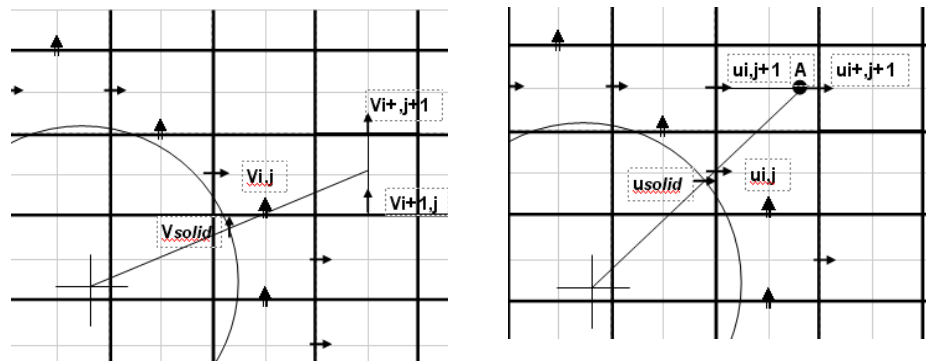


Figure 6-2: Bilinear proposed interpolation in this study for the cells near the solid boundary in vertical (Left) and horizontal (right) velocity components.

6.2.4 Case D: Bilinear interpolation method

Kang et al. 2009 presented various interpolation methods for the immersed boundary method in two dimensions considering the effect of the pressure near the boundary as well as velocity in the previous time step. In this comparison study his interpolation schemes where only involve a pure velocity interpolation were selected. In the Standard Reconstruction method (SRM), Kang et al. 2009, used the two neighbouring velocities in the horizontal and vertical directions that were located closest to the immersed boundary to interpolate velocities at the interface points (Figure 3-9). The resulting interpolation formula for the velocity in the horizontal direction is presented by equation (3-15), where the coefficients represent the interpolation weights. This method is similar to the linear methods (Case C), however, interpolations are

performed in both x and y directions in order to find boundary velocities. For some points, due to the curvature of the immersed boundary, the interpolation is only possible in one of the directions; therefore this method reduces to a linear interpolation method at those points (Figure 3-10).

6.2.5 Case E: Proposed interpolation method

The bilinear interpolation method proposed in this paper is based on interpolating the boundary velocity values in the direction perpendicular to the immersed boundary. In this method, perpendicular lines from the boundary surface are drawn which intersect the locations of the boundary velocities and cut the line between the first two known velocities in the fluid domain (Point A, Figure 6-2 right). The velocity is interpolated between two known velocities at the intersection point A. Then, the boundary cell velocity values will be interpolated using the solid boundary velocity (for a stationary cylinder with no-slip conditions this velocity is zero) and the velocity at point A. Figure 6-2 (left) shows this interpolation for velocities in the y direction and Figure 6-2 (right) shows the interpolation for the velocity in the x direction.

There are some alternative interpolation methods presented in the literature that interpolate the interface cell velocities in the perpendicular direction to the immersed boundary (Balaras 2004, Gilmanov et al. 2003 among others); but in these methods the procedure to find the interpolation points is very time consuming (see section 3.3.3). For the stationary cases, the interpolation formulas are calculated only once, prior to the simulation, and at each time step the values of the boundary cells are updated using the same formulas. However, as for the problems with moving immersed boundaries, the interpolation formulas should be recalculated at each time step, the interpolation method should not be too time consuming to execute.

6.3 Results and discussion

The flow around a stationary circular cylinder at low Reynolds number, $Re=100$, is taken as a bench mark. Five different interpolation treatments are implemented separately to represent the immersed boundary (the circular cylinder). The Strouhal number (St), drag (CD) and lift (CL) coefficients for various cases are compared.

For any solid body both the pressure distribution and the friction along the solid surface may contribute to the lift and drag forces. In this chapter, the pressure at the surface is obtained by taking the wall-nearest pressure values in the flow domain on the

outside of the solid body, thereby assuming that the wall normal gradient of the pressure near the surface is negligibly small. The component of the drag and lift forces due to pressure distribution is calculated by integrating the pressure along the solid boundary. On the other hand, the shear-force component of the lift and drag forces is calculated from near the surface of the solid. The tangential velocity near the solid surface is obtained at the wall-nearest point outside of the body and is subsequently used to calculate the wall-shear stress at the cylinder surface (see chapter 4 for more details).

Table 6-1: Real computational time, 20 vortex shedding

	Case A	Case B	Case C	Case D	Case E
Real time (s)	3231	3225	3379	4441	3383

The simulation times for 20 complete vortex shedding periods are measured for these five different interpolation cases. Once vortex shedding commenced all simulations were found to run at virtually the same speed (Table 6-1) showing that the computational effort needed for the interpolation was negligible as most of the computational time (more than 70%) is taken by the Pressure Poisson solver. However, for a non-stationary cylinder, it is expected that updating the interpolation formulas may lead to an increment in the execution time for the linear and bilinear methods.

Figure 6-3 (left) shows that Case C (linear interpolation) is the quickest method to develop vortex shedding, which indicates that the implementation of boundary conditions with linear interpolation causes significant numerical noise. In Case E (proposed bilinear method), on the other hand, the vortex-shedding instability kicks in much later evidencing that the level of numerical noise introduced by this type of interpolation is very small.

Figure 6-3 (right), shows a comparison of the drag coefficients obtained in calculations of flow over a stationary cylinder at $Re=100$ using various interpolation methods. It can be seen that in the cases C, D and E, (linear and Bilinear interpolation methods) the results are converging to a value of $C_D = 1.43$. However, Case A (without interpolation) leads to a higher drag coefficient, $C_D=1.46$ and Case B (weighting method) leads to a lower drag coefficient $C_D=1.42$. In the literature for this bench mark (at $Re=100$), the drag coefficient is reported in the range from 1.33 to 1.47 (Table 6-2).

Therefore, all the interpolation methods are predicting acceptable values for the drag coefficient.

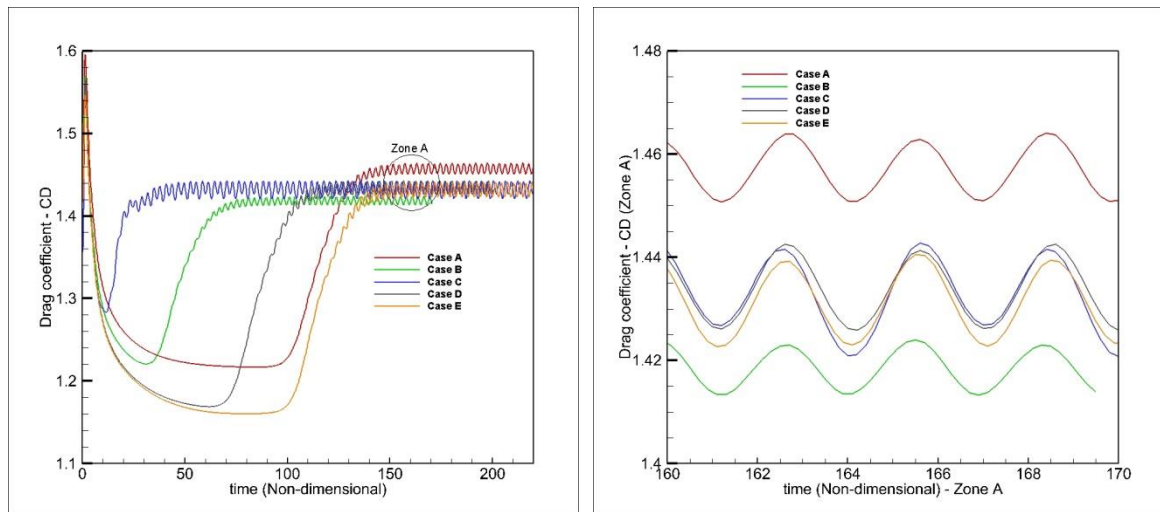


Figure 6-3: Drag coefficient for the flow around a stationary cylinder at $Re=100$, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation; Case E, Suggested bilinear interpolation.

The drag coefficient due to pressure and shear stress show a slightly different behaviour (Figure 6-4). The drag coefficient due to the pressure for linear (case C) and bilinear (cases D and E) methods are about 1.18, however for Case A (without interpolation) and case B (weighting method) these values are 1.11 and 1.06 respectively. The results show that both case A and B predict smaller values (by about 8%) pressure drag coefficient in comparison to the other linear and bilinear interpolation methods.

On the other hand, in the case A and B, the mean drag coefficient due to shear stress (Figure 6-4 right) are predicted to be 0.345 and 0.355 respectively, which is about 40% higher than the values predicted by linear and bilinear methods (case C, D and E) which are about 0.245. The numerical results show that the two cases A and B are predicting a lower value for the drag coefficient due to the pressure (about 8%) and a higher value for the drag due to the shear stress (about 40%) in comparison to the linear and bilinear cases. But the drag coefficient for cases A and B differ only about 2% from those obtained in the other interpolation methods. This can be explained by the fact that accumulated errors are cancelling out. Therefore, it is important to notice that the difference among the methods should not be judged only by the drag coefficient and the

drag components should be investigated as well. In addition, according to the Figure 6-3 right and Figure 6-4, in the linear and bilinear interpolation (case C, D and E) methods, the drag coefficient, the drag due to pressure and drag due to the shear stress are converging nearly to the same values. The numerical results confirm that the suggested bilinear interpolation method (case E) has the same accuracy as the linear (case C) and bilinear (case D) methods.

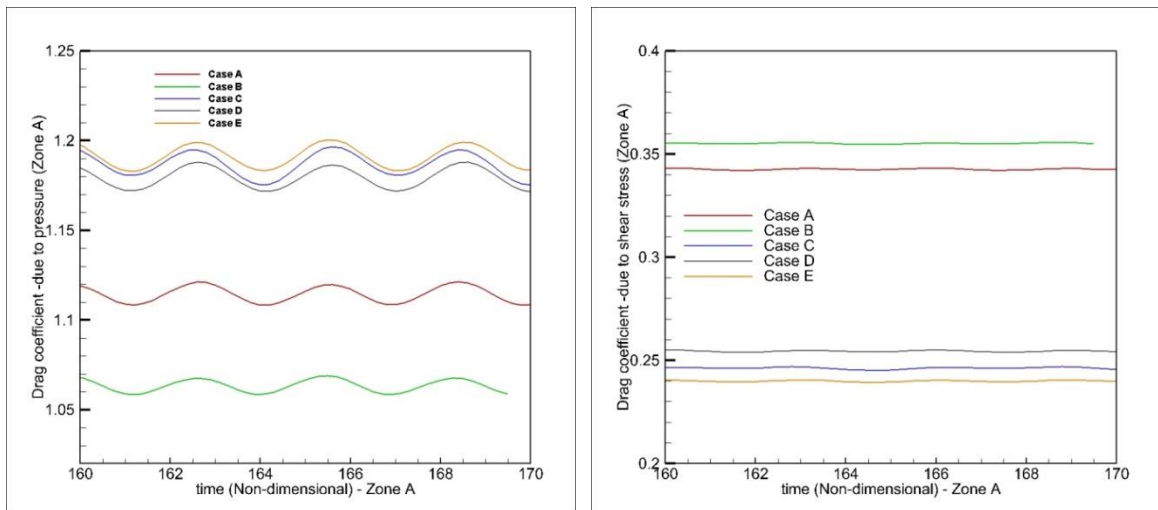


Figure 6-4: Drag coefficient due to pressure (left) and due to shear stress (right) for the flow around a stationary cylinder at $Re=100$, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation1; Case E, proposed bilinear interpolation method

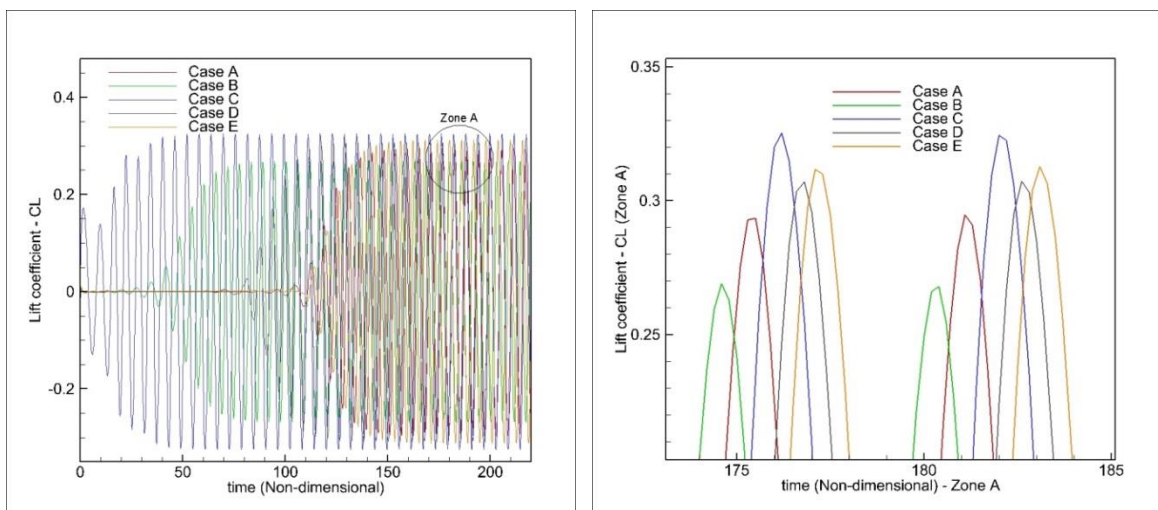


Figure 6-5: Lift coefficient for the flow around a stationary cylinder at $Re=100$, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation method; Case E, suggested bilinear interpolation method

Figure 6-5 and Figure 6-6 show the comparison of the lift coefficients for the various interpolation cases. It can be seen that, like the drag coefficients, also the lift coefficients for the bilinear cases are nearly the same (Case D and E) with $CL \approx \pm 0.31$. Case B (weighting method) shows the lowest value for the lift ($CL = \pm 0.27$) and Case C (linear interpolation), shows the highest value for the lift coefficient ($CL = \pm 0.325$).

The simulation results show that the lift coefficient predicted by the linear (case C) and bilinear (case D and E) methods are matching well with the results reported in the literature (Table 6-2). However Case A (without interpolation) and case B (weighting method) show a lift coefficient that is slightly lower than the values reported in the literature.

The Power Spectral density (PSD) of the lift coefficient is presented in the Figure 6-7. The PSD graph illustrates that interpolation methods could affect the frequency of the vortex shedding (Strouhal number) for the stationary cylinder. The numerical results show that, apart from cases A (without interpolation) and B (weighting method) that predict a higher Strouhal number (0.174 and 0.176), the other interpolation methods do not affect severely the Strouhal number. The Strouhal number for the linear interpolation method (Case C), for the bilinear interpolation method (Case D) and for the proposed bilinear interpolation method (Case E) is predicted about 0.169.

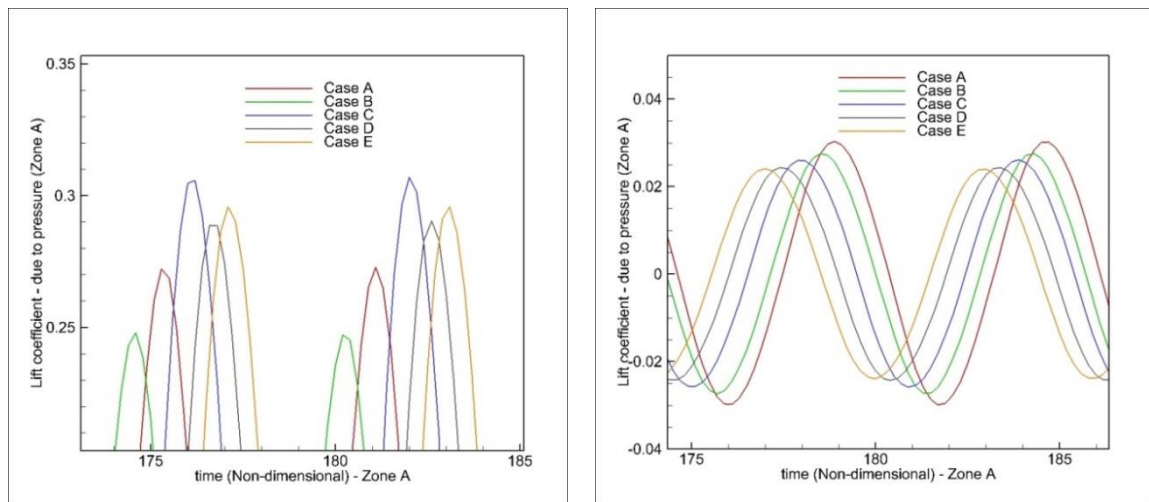


Figure 6-6: Lift coefficient due to pressure (left) and shear stress (right) for the flow around a stationary cylinder at $Re=100$. Case A, without interpolation method; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation method; Case E, suggested Bilinear interpolation

In the numerical literature, the Strouhal number for the flow around a circular cylinder at $Re=100$ is reported in the range from 0.164 to 0.175. However, most of the

experimental results reported show a Strouhal number in the range from 0.164 to 0.167. According to the parametric study (Chapter 6), the reason that these numerical simulations present higher values for the Strouhal number is related to the entrance length (before the cylinder), which it is taken to be 5D in this chapter.

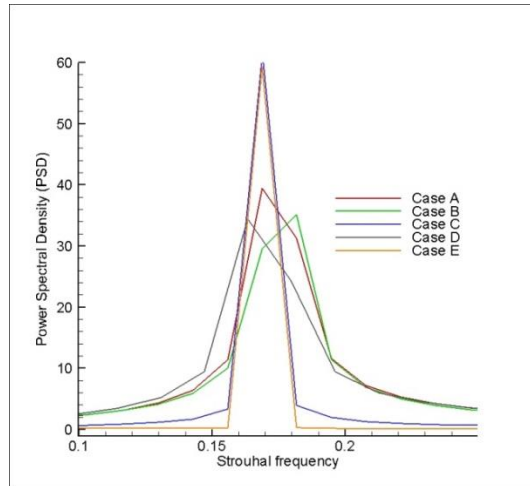


Figure 6-7: Power Spectral density of the lift coefficient; five different interpolation methods

Table 6-2: Strouhal number, lift and drag coefficient for the flow around a stationary cylinder and Re=100.

simulation methods	Strouhal Number	Drag Coefficient	Lift coefficient
Case A	0.174	1.46	0.29
Case B	0.175	1.42	0.27
Case C	0.169	1.432	0.325
Case D	0.169	1.434	0.305
Case E	0.168	1.432	0.312
Park 1998, fitted method	0.165	1.33	0.33
Williamson 1988(exp.)	0.166
Kim et al. 2001	0.165	1.33	0.32
Roshko 1954(exp.)	0.164
Lai and Peskin 2000	0.165	1.4473	0.3299
Choi et al. 2007	1.351	0.315
Corbalan & de Souza 2010	1.44	0.31

Table 6-2 shows a comparison of the Strouhal number, lift and drag coefficient for the flow around a stationary cylinder at $Re=100$ using various methods; ranging from the experimental methods (Roshko 1954 and Willamsion 1988) to the body fitted mesh (Park et al. 1998) and immersed boundary methods (Kim et al. 2001, Lai & Peskin 2000, Choi et al. 2007 and Corbalan & de Souza 2010). It can be seen that the Strouhal number varies between 0.16 and 0.18; the drag coefficient between 1.33 and 1.4473 and the lift coefficient between 0.31 and 0.33.

6.4 Conclusion

The objective of the present study was to compare the accuracy and computational efficiency of various IB interpolation methods and select the most accurate and least expensive method for future use in the simulations of flow around a deformable cylinder. The fractional step method and a staggered variable arrangement on a Cartesian grid have been used to solve the governing equations. In the proposed IB method the velocities near non-conforming boundaries were interpolated in the normal direction to the walls, thereby considering the curvature of the geometry. The Strouhal number, drag and lift coefficient for 5 different IB interpolation methods are compared. The overall results show a good agreement with the literature for most of the interpolation methods for the stationary cylinder at a low Reynolds number, $Re=100$. The drag coefficient results for the five different interpolation methods differ by no more than 2%, while the drag due to shear stress shows differences of up to 40% due to the accumulated errors, however simulation results only show a 2% difference in drag coefficients. The Strouhal numbers for five different interpolation methods differ only by a maximum of 3%. The simulation results show a difference of about 15% on the lift coefficient between the interpolation methods. However the lift coefficients calculated by linear and bilinear interpolation methods were formed to match well with literature.

In addition, the bilinear interpolation method took about 2% more computational time per vortex shedding cycle compared to the other methods. In the next chapter the proposed interpolation method is used to simulate body cross flow oscillation of a circular cylinder.

Chapter 7. Body cross flow oscillation

Having studying the flow over a stationary bluff body in previous chapters, the focus of this chapter is on the flow over a moving body with a degree of freedom in the cross flow direction. This chapter briefly presents the theory and governing equations necessary to simulate a moving body in a uniform stream. Also, it is explained how the Navier-Stokes equations with IB interpolation are modified to allow modelling of a moving boundary in the presence of either forced oscillations or with prescribed motion and Vortex-Induced Vibrations (VIV) in the cross flow. In this model the IB interpolation technique is used to represent the immersed boundary on a Cartesian grid. To simulate the FSI problem, two approaches are followed; an inertial frame of reference and a moving (non-inertial) frame of reference. In the latter case, the frame of reference is attached to the body and the governing equations are solved in a relative frame of reference.

7.1 Forced Oscillation of a body in cross flow direction

In a forced excitation of a body, the body oscillates at the forcing frequency with a prescribed motion in the cross flow direction. At some specific range of oscillation the frequency of vortex shedding around the body becomes similar to the oscillation frequency. From the literature it is known that the frequency of vortex shedding can be controlled for a limited range of reduced velocities, where the vortex shedding frequency and the body oscillation frequency become synchronized. This phenomenon is usually known as 'lock-in'. Simulation results show that the lock-in occurs only in a frequency range close to the system's natural frequency, above a threshold of oscillation amplitude. The lock-in range increases with increasing the amplitude (Figure 1-4). Moreover, a dramatic change might occur in the flow patterns and lift and drag forces by increasing the oscillation amplitude in the lock-in region. Another important issue in a cross flow oscillation is the phase change between the vortex shedding and the forced oscillation. In some cases the amplitude of the lift coefficient for the vibrating cylinder is lower than a stationary case, due to the fact that the inertial part of the lift force dominates in this range of oscillations and has a different phase than the lift due to the vortex shedding. This issue in low amplitude vibration could lead to a lock-in and beating pattern. The body's motion in the y direction is defined as a sinusoidal motion as,

$$y_c(t) = A_{0y} \sin(\omega t) = A_{0y} \sin(2\pi f t) \quad (7-1)$$

Where $y_c(t)$ is the location of the centre of the cylinder and A_{0y} , ω and f is the amplitude, the frequency in rad/s and the frequency in Hz of the prescribe oscillation, respectively.

7.2 Fluid-Structure interaction due to Vortex induced Vibration

When a flow passes a bluff body, Fluid-Structure Interaction (FSI) and vortex shedding phenomena may incur the bluff body to oscillate. This oscillation is known as Vortex-Induced Vibration (VIV) in the literature. If the frequencies of the VIV and the natural frequency of the structure become similar, the flow may induce resonance in the structure. The governing equation of a structure (Figure 7-1) that is flexible (one degree of freedom) in the cross flow direction is given by:

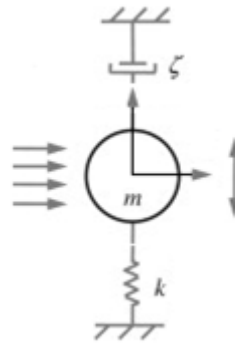


Figure 7-1: Flow over a circular cylinder at two dimensions with vertical degree of freedom

$$m \frac{d^2 y}{dt^2} + C \frac{dy}{dt} + ky = F_L(t) \quad (7-2)$$

Where m , C and K are mass, damping and stiffness of the structure, respectively, while y corresponds to the transverse displacement of the centre of the body. F_L is the hydrodynamic force in the cross flow direction. The same non-dimensional scaling as in the flow governing equation is applied here to non-dimensionalize the structural governing equation.

$$\frac{d^2 y^*}{dt^{*2}} + 2 \times \xi \times \left(\frac{2\pi}{V_r}\right) \frac{dy^*}{dt^*} + \left(\frac{2\pi}{V_r}\right)^2 y^* = \frac{2 \times C_L(t)}{\pi m^*} \quad (7-3)$$

Where, the non-dimensional parameters are labelled by a ‘*’. In the reminder of this thesis this sign is dropped for simplicity. $V_r = \frac{U_\infty}{f_N D}$ is the reduced velocity where

$f_N = \left(\frac{1}{2\pi}\right)\sqrt{\frac{K}{m}}$ is the natural frequency of the undamped structural system (mass and spring), U_∞ and D are the free stream velocity and cylinder diameter, respectively (the same reference scales as used to non-dimensionalized the Navier-Stokes equations) . $C_L = \frac{F_L}{\frac{1}{2}\rho U_\infty^2 D L_c}$ is the lift coefficient where ρ is the fluid density and L_c is the length of the cylinder (This length is assumed to be $L_c=1$ in the two dimensional simulation). $m^* = \frac{m}{m_f} = \frac{m}{\rho_f \pi \left(\frac{D^2}{4}\right) L_c}$ is the mass ratio, i.e. the mass of the structure (cylinder) m , over the mass of the fluid replaced by the structure m_f . $y^* = \frac{y}{D}$ is the non-dimensionalized vertical displacement. $t^* = t \times \frac{D}{U_\infty}$ is the non-dimensional time and $\xi = \frac{c}{c_c}$ is the structural damping ratio where $c_c = 2\sqrt{km}$ is the critical damping.

In an FSI simulation, at every time step the hydrodynamic forces are calculated by solving the flow governing equations and the displacement of the structure based on these forces is predicted. In the same time step the flow governing equations for the new configuration of the structure is solved to predict the new hydrodynamic forces. This process is continued iteratively to obtain a converged solution with the convergance criteria being a constant position of the structure before going to the next time step i.e strong coupling.

The free vibration (VIV) and forced vibration of a structure can be presented in either a moving frame of reference or an inertial frame of reference. In the following sections these two approaches are briefly presented. Also, the simulation results based on these two approaches will be compared.

7.3 First approach-moving frame of reference

In this approach the reference frame is fixed to the moving body and the boundary conditions are defined in a way to resemble the same problem for an observer moving with the body. This can be explained due the fact that the flow about a circular cylinder forced to oscillate in the transverse direction to a free stream is kinematically the same as the flow about a fixed cylinder in a free stream with a superimposed oscillatory cross flow. It should be noted that these two flows differ dynamically due to the inertial effects. This effect is known as the Froude-Krylov force in the literature (Meneghini and Bearman1995).

$$C_L = (C_L)_{relative\ frame} + \frac{\pi D}{2U^2} \frac{d^2 y_{solid}}{dt^2} \quad (7-4)$$

In this equation C_L is the lift coefficient in the inertial frame of reference, $(C_L)_{relative\ frame}$ is the lift coefficient which is calculated in the moving frame of reference and $\frac{\pi D}{2U^2} \frac{d^2 y_{solid}}{dt^2}$ is the non-dimensional inertial term for a circular cylinder. D is the cylinder diameter and $\frac{d^2 y_{solid}}{dt^2}$ is the acceleration of the cylinder in the inertial frame of reference.

In the discussion of the methodology (Chapter 4) it was explained that regardless of the simulation approach (conforming grid, e.g. ALE or non-conforming grid, e.g. IB), it is possible to simulate moving boundaries in a non-inertial frame of reference. The combination of the conforming grid approach with a non-inertial frame of reference could be the best algorithm to simulate FSI for a single two dimensional rigid body motion in fluid flow. On the other hand, the relative reference frame could improve non-conforming grid approaches significantly as the IB formulation does not need to be updated because relative displacement of the body and the background computational grid is zero. In this approach, the governing equation of the flow is solved in a moving reference frame which is attached to the cylinder. To solve the governing equation in the relative frame two fundamental changes are necessary. First of all, the governing equation should be derived in the relative frame of reference. This subject has been addressed in section 3.5 for a general case. The Navier-Stokes equation in the relative frame of reference has additional terms to compensate for the effect of the moving frame in the calculation. Also, the boundary conditions should be introduced in the relative reference frame as well. Here, only the movement in the transverse direction is considered. The updating of the governing equations and boundary conditions is described below.

7.3.1 Moving frame-governing equation

Equations (3-35) to (3-39) govern the flow in the moving frame of reference given a general movement in the two dimensional case. For the movement of the body in the cross flow direction the governing equations can be written as:

$$\nabla \cdot \mathbf{V} = 0 \quad (7-5)$$

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} = -\nabla p + \vartheta \nabla^2 \mathbf{V} - \dot{\mathbf{d}} \quad (7-6)$$

In these equations the velocities are all relative. The $G(v, t)$ term in equation (3-37) is simplified to only \ddot{d} which is the transverse acceleration of the moving body in the inertial frame of reference. The other terms in equation (3-37) cancel due to the fact that the moving frame is not rotating, hence $\theta = \dot{\theta} = \ddot{\theta} = 0$ and the rotation matrix $A = A^T = I$. For instance, in the case of a transverse sinusoidal oscillation (equation (7-1)) of the cylinder, \ddot{d} reads:

$$\ddot{d} = \dot{y}_c(t) = -A_{0y}\omega^2 \text{Sin}(\omega t) \quad (7-7)$$

7.3.2 Moving frame-velocity boundary conditions

The boundary conditions should be applied in the relative frame of reference. Equation (3-39) shows the velocity in the relative frame of reference. For the transverse oscillation of the body the frame does not have an angular velocity, i.e. $\dot{\theta} = 0$, hence

$$V = \dot{V} - \dot{d} = \dot{V} - \dot{y}_c(t) = \dot{V} - A_{0y}\omega \text{Cos}(\omega t) \quad (7-8)$$

\dot{V} is the velocity in the absolute frame of reference, in this frame a symmetric boundary condition is applied in the top and bottom of the computational domain hence in the absolute frame of reference the velocities normal to these boundaries are zero, $\dot{V} = 0$. Therefore equation (7-8) can be simplified for the above case (movement of body in the transverse direction) to give:

$$V = -\dot{d} = -\dot{y}_c(t) = -A_{0y}\omega \text{Cos}(\omega t) \quad (7-9)$$

7.3.3 Moving frame-Neumann boundary for pressure Poisson equation

Finding a proper pressure boundary condition for the elliptic pressure Poisson equation (PPE), equation (7-10), has been the subject of some controversy (Gresho & Sani 1987 and Sani et al. 2006). First of all, as a necessary condition for the existence of a solution to a problem with a Neumann boundary condition (equation (7-11)), the boundary condition should be well-posed i.e. the source and the boundary data should satisfy the compatibility condition (equation (7-13)).

$$-\nabla^2 p = f \quad (7-10)$$

$$\frac{\partial p}{\partial n} = g \quad (7-11)$$

The compatibility condition is obtained by applying the divergence theorem (also known as Gauss' theorem) to the integration of the Poisson equation over the domain

(equation (7-12)). To do so, the Laplace operator is written as the divergence of the gradient vector.

$$-\int_{\Omega} \nabla^2 p = -\int_{\Omega} \nabla \cdot \nabla p = -\int_{\Omega} \frac{\partial p}{\partial n} = \int_{\Omega} f \quad (7-12)$$

$$\int_{\partial\Omega} g + \int_{\Omega} f = 0 \quad (7-13)$$

More precisely, equation (7-13) states that the outward flux of the vector field (gradient of pressure on the boundaries) is equal to the volume integral (here surface) of the divergence (of the pressure gradient) over the region inside the boundaries. In other words, it states that the sum of all sources minus the sum of all sinks gives the net flow out of a region.

A natural method to define the Neumann boundary for the pressure is by using the normal component of the momentum equation at the boundaries (Blackburn and Henderson 1999). By taking the dot product of the domain outward normal unit vector, n , with the momentum equation (7-6), the Neumann pressure boundary condition is obtained as

$$n \cdot \nabla p = \frac{\partial p}{\partial n} = -n \cdot \left[\frac{\partial V}{\partial t} + V \cdot \nabla V + \vartheta(\nabla \times \nabla \times V) + A\ddot{d} \right] \quad (7-14)$$

In the above equation, according to the suggestion of the Orsag et al. 1986, the viscous term is presented by using the vectors identify:

$$\nabla^2 V = \nabla(\nabla \cdot V) - \nabla \times \nabla \times V \quad (7-15)$$

Also, Blackburn and Henderson 1999 suggested writing the non-linear term (convection term) as a skew symmetric form (equation (7-16)).

$$V \cdot \nabla V = (V \cdot \nabla V + \nabla \cdot VV)/2 \quad (7-16)$$

7.3.4 Moving frame of reference algorithm

Using a non-inertial reference frame allows to simulate FSI problems with moving boundaries in a fixed Cartesian grid (as compared to an ALE approach with moving/deforming grid) while the interpolation coefficient maintains unchanged (in comparison to an IB approach in an inertial frame). Therefore using a moving frame of reference would be potentially an efficient approach; however this method is limited to a single moving object or synchronised moving objects. The algorithm for the simulation of a forced vibration of a rigid body using a moving frame of reference is as follows.

1. The flow boundary condition (location and velocities) are updated according to the prescribed motion of the cylinder in time.
2. The flow velocity is updated at the new time step using an explicit Rung-Kutta method.
3. The pressure Poisson equation with Neumann boundary conditions is solved by an iterative method.
4. The velocity vectors are updated by using the pressure term from the previous step.

The above algorithm is repeated until a steady state solution is reached.

7.4 Second approach, moving IB or fixed grid (inertial frame of reference)

When simulating flow around a stationary cylinder, the interpolation formulas are calculated once and the interface velocities (around the cylinder) are updated using interpolation formulae for every iteration. Therefore, the interpolation formulae at the boundary cells remain unchanged. However, in a moving cylinder, the position of the cylinder is changing, and therefore the boundary cells and interpolation formulae could potentially change. In other words, at each time step, if the position of the cylinder is changed, the interpolation formulae should be updated as well. To do this, before updating the interpolation formulae each time step, the position of centre of cylinder is updated automatically to the new position using the prescribe motion (equation (7-1)).

One of the important issues is the relation between the time steps of the fluid flow and the time steps of prescribed motion of the structure. Choosing the time step of the structure and the flow depends on the CFL number in the fluid flow and the prescribed motion of the structure. It is important that the time step in the fluid should not lead to instability. However, choosing a very small time step will be expensive. Firstly, because the interpolation formulae and also the LU decomposition matrices should be recalculated each time step and secondly, the boundary conditions of the flow will change at each time step which leads to a higher number of inner iteration for the flow (Poisson solver) to resolve these perturbations.

Choosing different time step for the flow and the structure is not recommended as it may cause a spurious phase between the lift coefficient and the displacement of the

cylinder. An approach, like artificial incompressibility that uses dual time stepping is a potential remedy for this problem (Gilmanov and Sotiropoulos 2005).

7.4.1 Inertial frame-governing equation and boundary conditions

Equation (4-8) presents the momentum equation in the non-dimensional form. This equation and the continuity equation are solved by the fractional step (Chorin projection approach) method as explained in section 4.3.3. In this context, the vector form of the governing equation is as follows:

$$\frac{\partial V}{\partial t} = -V \cdot \nabla V + \vartheta \nabla^2 V \quad (7-17)$$

$$\frac{\partial V}{\partial t} = -\nabla p \quad (7-18)$$

In equation (7-17) (by ignoring the pressure in the momentum equation), an intermediate velocity that does not satisfy the incompressibility constraint is calculated. The intermediate velocity will be projected to a solenoidal space (divergence-free velocity field) using equation (7-18). In this equation, the pressure field is calculated by solving the pressure Poisson equation (PPE). (PPE is formed by forcing the mass conservation to the divergence of the momentum equation).

The boundary conditions for the domain remain unchanged compared to the stationary case. However, the boundary around the moving object should be updated in time according to the prescribed motion of the cylinder. Also the Neumann condition for the pressure Poisson equation should be updated according to the following equation as explained in the previous section:

$$\frac{\partial p}{\partial n} = -n \cdot \left[\frac{\partial V}{\partial t} + V \cdot \nabla V + \vartheta (\nabla \times \nabla \times V) \right] \quad (7-19)$$

7.4.2 Inertial frame of reference algorithm

The main advantage when using an immersed boundary approach is the ability to simulate the Fluid-Structure-Interaction (FSI) for a moving object on a fixed grid. In this approach, unlike the Arbitrary-Lagrangian-Eulerian (ALE) approach the computational grid is not deforming or displacing, even though at each time step the interpolation formula needs to be updated. To simulate a cylinder moving with a prescribe oscillation in the cross flow direction the following algorithm is used.

1. From the prescribed motion and the simulation time, the position of the cylinder is known and is used to calculate the interpolation formulae and LU matrices.

2. The velocity around the cylinder at each new position is updated with new interpolation formula.
3. The pressure Poisson equation with Neumann boundary conditions is used to enforce the continuity equation.
4. The velocity field is updated using the new pressure gradient as calculated in step 3.

The above algorithm is continued in time to reach a fully developed solution.

7.5 Calculation of the force on moving boundary

In the non-conforming boundary approach, the computation of local forces on a moving boundary is not a trivial problem (Yang & Balaras 2006). Lai and Peskin 2000 compared three methods of force calculation to their own approach (immersed boundary-continuous forcing approach). In section 4.8 a direct method is presented to calculate the local force on the stationary (or moving with constant velocity) immersed boundary. In this section the method is developed for the moving boundary case as well. The simulation results show that for low amplitudes of oscillation (i.e. small acceleration) the same procedures are acceptable. However, for oscillations with higher acceleration, corresponding to higher amplitude and/or frequency of oscillation, some special treatment (extrapolation of the pressure near the boundary) could improve the simulation accuracy (Gilmanov and Sotiropoulos 2005).

7.6 Parametric study

In this section various parameters which could potentially affect hydrodynamic forces from the uniform free stream on an oscillating cylinder are briefly addressed. According to the parametric study for a stationary cylinder the mesh size, domain size up stream of the cylinder and the domain size in the transverse direction to the flow are the most influential factors. Here, these effects are studied for cylinder oscillating in cross flow direction with an amplitude of $A/D=0.2$, while the frequency of excitation is $f_e=1.05 \times f_s$. The parametric study is performed at $Re=100$, based on the free stream velocity and cylinder diameter. So that the Strouhal frequency is $f_s=0.167$. The effects of different prescribe motions (amplitude and frequency) on the lift and drag coefficient are presented later in the results section.

7.6.1 Parametric study- mesh size

The size of the grid around the immersed boundary is an important parameter in the study of the flow around the bluff body. The boundary conditions can be applied more precisely while there are fine grids around the IB; however, using a very fine mesh near the IB is very expensive and might slow down the simulation process significantly. The results of the mesh refinement study for the flow around a circular cylinder is presented in this section to show the optimum grid size for this problem.

Table 7-1: mesh refinement study of oscillating cylinder – Parameters and results

$\Delta x = \Delta y$	Number of grid at each direction	Total no. of grid points	Actual computational time (s) (for 100 time-units)	Mean-Drag coefficient	Max-Lift coefficient
0.1	122×97	11'834	2'500 (45 minutes)	1.33	0.45-0.71
0.05	240×191	45'840	5'800 (1.6 hours)	1.58	0.575
0.025	468×375	175'500	27'800 (7.2 hours)	1.59	0.55
0.0125	942×742	698'964	237'000 (2.74 days)	1.59	0.545
0.00625	1880×1489	2'799'320	2'206'000 (25.7days)	1.60	0.55

According to the Figure 5-2, the centre of the cylinder is located at the origin of the computational grid and the size of domain the in x, y directions is $[-15D, 30D] \times [-20D, 20D]$. The size of the embedded uniform grid area is $[-2D, 4D] \times [-2D, 2D]$ and the stretching factor is 4. The cylinder is forced to oscillate in the cross flow direction with an amplitude of 0.2D and a frequency of $F=f_o/f_s=1.05$.

In this study, the mesh size of the embedded uniform mesh is changed from 0.1D to 0.00625D. The numerical results show that for the coarse mesh ($dx=dy=0.1D$) the lift and drag coefficient are highly affected by the size of the mesh, however for the fine meshes, this effect is negligible. For instance, if the size of the mesh changes from 0.1D to 0.05D, the mean drag coefficient increases by about 16%, while a decreases in the grid size from 0.025D to 0.0125D results changes is negligible in lift and drag coefficients (less than 1%).

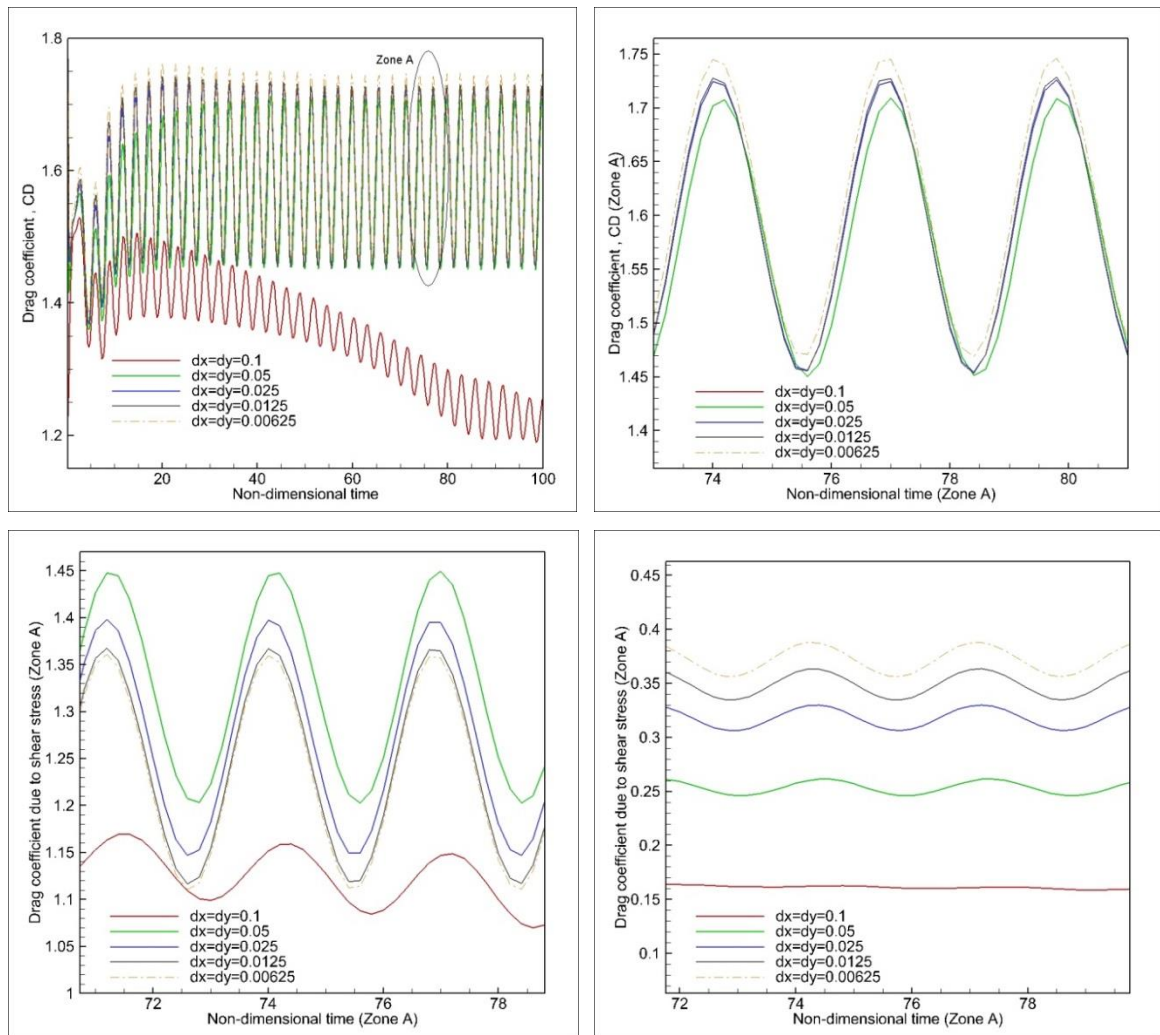


Figure 7-2: mesh refinement study- Drag coefficient

In Table 7-1 the accuracy of the numerical results and the computational time needed to achieve the accuracy is presented for five different mesh sizes. It is shown that the computational time to simulate hundred non-dimensional time step increases from 1.6 hours to 25.7 days (385.5 times increase) when the mesh is refined from a size of 0.05D to 0.00625D (8 times decrease), respectively.

Figure 7-2 and Figure 7-3 show the time history of the lift and drag coefficients for the five different mesh sizes listed in Table 7-1. The graphs show that the drag coefficient, the drag due to the pressure and due to shear stress are more sensitive to the mesh size than the lift coefficient. It can be seen that the lift and drag coefficient converge to the steady solution after about 50 non-dimensional simulation time.

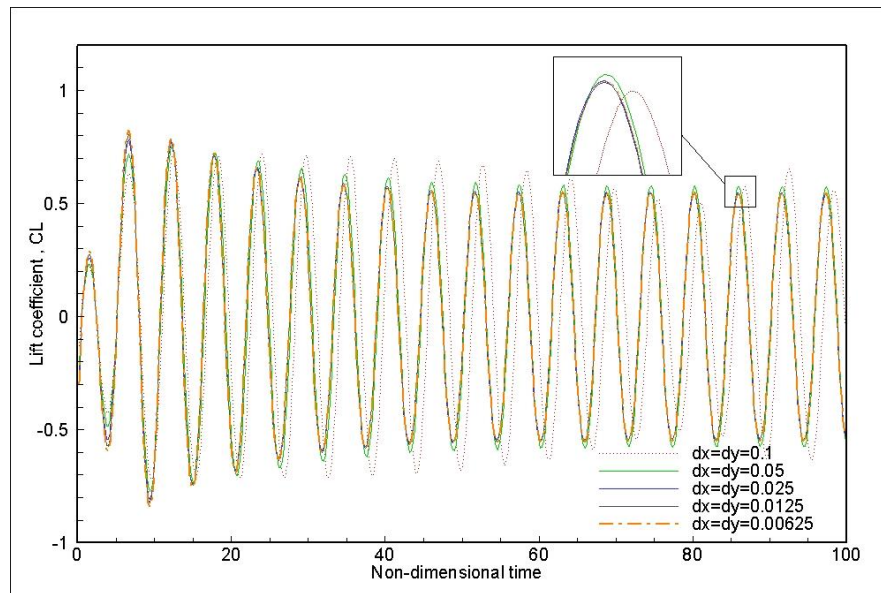


Figure 7-3: mesh refinement study – Lift coefficient

7.6.2 Parametric study-size of domain before the cylinder

One of the important parameters which highly affects the hydrodynamic forces is the length of the computational domain upstream of the solid body which received less attention in the literature. The size of domain upstream of the cylinder is changed from 5D (five times of the cylinder diameter) to 30D. The size of the domain in the y direction for this study remains 40D. The simulation results (Figure 7-4) show that mean drag coefficient decreases by 4.4% and the maximum lift coefficient increases by 23.6%, respectively, by increasing the size of the domain upstream of the cylinder from 5D to 15D. However, if the size of the domain upstream of the cylinder is further increased from 20D to 30D, the mean drag and maximum lift coefficients only change by -0.2% and 1.5% respectively.

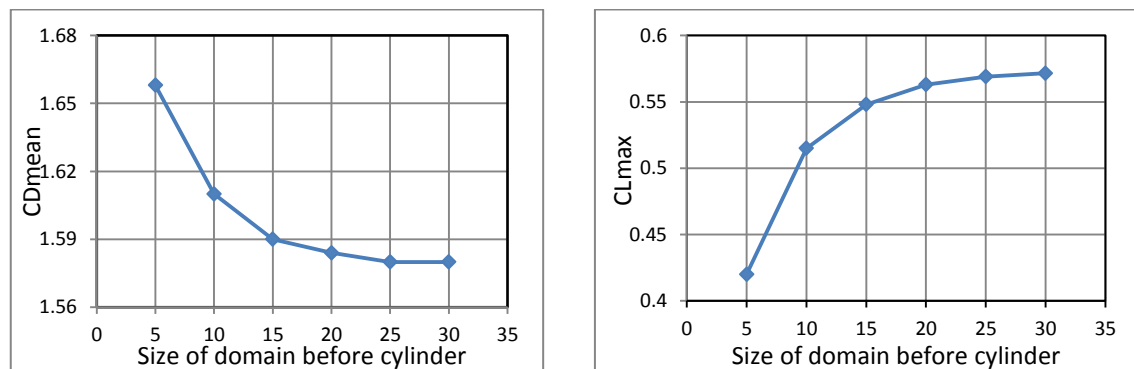


Figure 7-4: Parametric study of the effect of size of domain before cylinder in x direction on the mean drag and maximum lift; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$

7.6.3 Parametric study- blockage effect

Another important parameter affecting hydrodynamic forces is the size of the domain in the transverse direction which is addressed as the blockage effect in the literature. In this simulations the size of the domain in the transverse direction is increased from 10D to 100D; while the rest of the parameters is kept constant; in this case, according to the parametric study in the chapter 5, the size of the domain in the x direction is [-15D,30D].

The simulation results (Figure 7-5 and Figure 7-6) show that if the size of the domain in the transverse direction changes from 10D to 40D the mean drag and maximum lift coefficients are decreased and increased by 4% and 24.6% respectively.

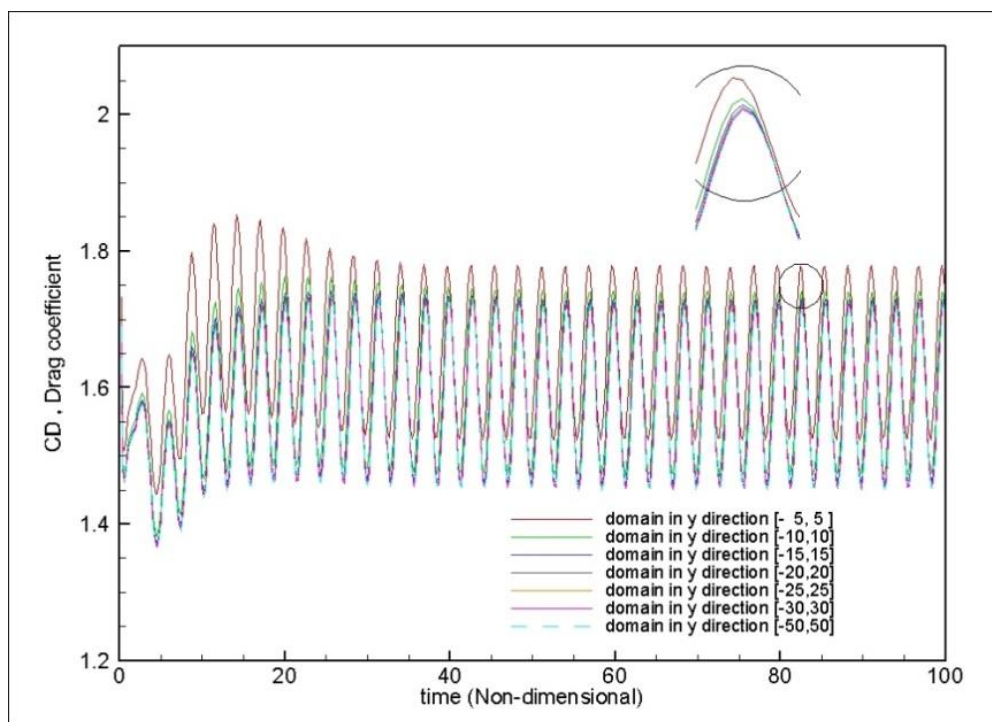


Figure 7-5: Parametric study of the effect of size of domain in y direction on drag coefficient; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$

However if the size of the domain in the y direction is further increased from 50D to 100D the mean drag and maximum lift coefficients change by about -0.1% and 1.6%, respectively. Figure 7-7 show the drag and lift coefficients based on the oscillation time for various domain sizes in the y direction.

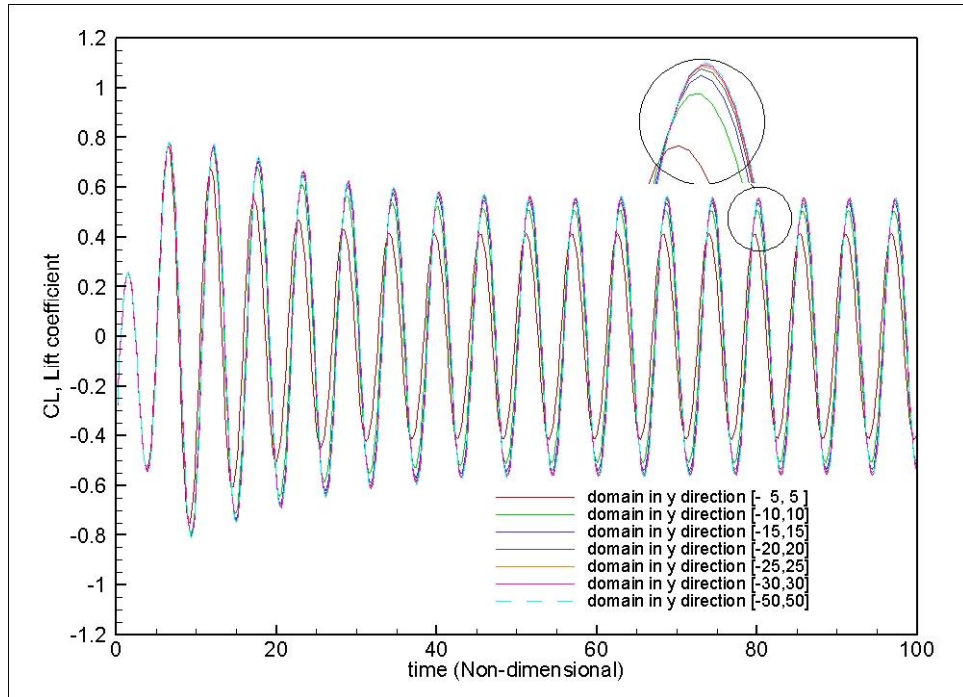


Figure 7-6: Parametric study of the effect of the size of the domain in the y direction on the lift coefficient; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$

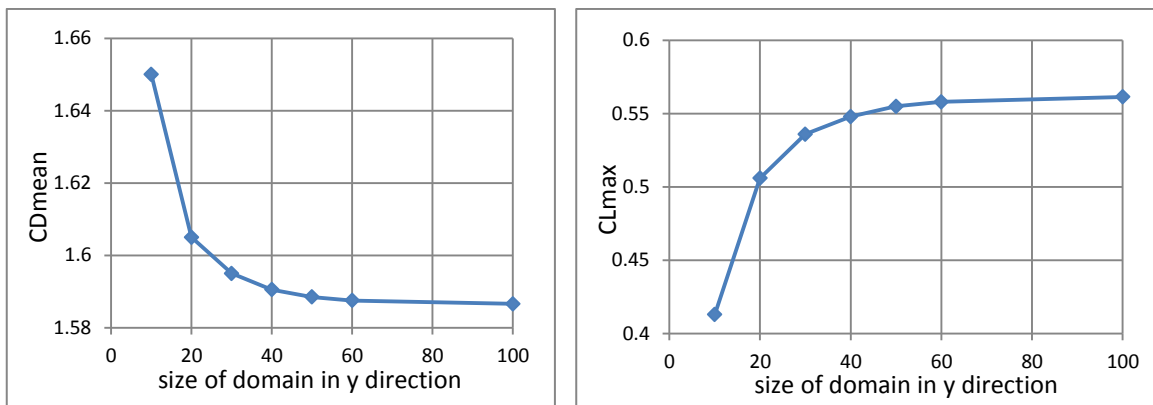


Figure 7-7: Parametric study of the effect of size of domain in y direction on the mean drag and maximum lift; cross flow oscillation with $A/D=0.2$ and $f_e/f_s=1.05$ at $Re=100$

7.7 Results

In this section, to validate the FSI algorithm presented in the previous sections, several cases with various amplitudes and frequencies of oscillation were selected as a bench mark. The simulations were repeated for various Reynolds numbers and the simulation results were compared with the literature and with inertial reference frame simulation results.

7.7.1 Inertial effect - Froude-Krylov force

As explained in section 7.3, it is possible to solve the flow governing equations in the moving frame of reference (equation (7-6)) as well as in the inertial frame of reference (equation (7-17)). This can be explained by the fact that the flow about a circular cylinder forced to oscillate in the cross flow direction is kinematically similar to the flow about a fixed cylinder in a free stream with a superimposed oscillatory cross flow (Meneghini and Bearman 1995). It should be noted that these two flows differ dynamically due to inertia effects (Froude-Krylov force). However, if the flow governing equations are fully derived in the moving reference frame (equation (4-6)) the inertia term has already been added to the equations, and the hydrodynamic forces comprise the inertia effect too. Therefore, the inertia effects should be added to the relative hydrodynamic forces if the flow governing equation (equation (4-17)) is solved with the relative velocities instead of absolute one without deriving the equation in a moving frame. To demonstrate the effect of inertial forces, the flow around a cylinder that is forced to vibrate in the cross flow direction is solved in the moving frame of reference using the following two methods. In the first case (Case A), equation (7-17) is used and in the second case (Case B), equation (7-6) is used. In both cases, $Re=150$, $F = f_e/f_s=0.9$ and the cylinder is forced to oscillate in the cross flow direction ((7-20)

equation (7-20)).

$$y_c(t) = A_{0y}\sin(\omega t) = A_{0y}\sin(2\pi \times F \times f_s t) \quad (7-20)$$
$$= 0.15 \sin(2\pi \times 0.9 \times 0.196 \times t)$$

In addition, in both cases the reference frame is attached to the cylinder and the relative velocities are defined at the inlet and far-field boundaries (top and bottom). At the outflow the convective boundary condition is used.

The simulation results show that in both cases (Case A and B) the pressure, the lift due to shear, the drag coefficient due to pressure and the shear stress are the same (Figure 7-9). However the inertial force shows a difference in the lift coefficients due to pressure (equation (7-4)) between cases A and B. In Figure 7-8, the red line and the green line show the lift coefficient (due to pressure) for cases A and B, respectively. In this figure, if the lift in case A (red line) and the inertial effect (orange line) are added together (back dots), the results are similar to the lift coefficient obtained in the case B (Green line).

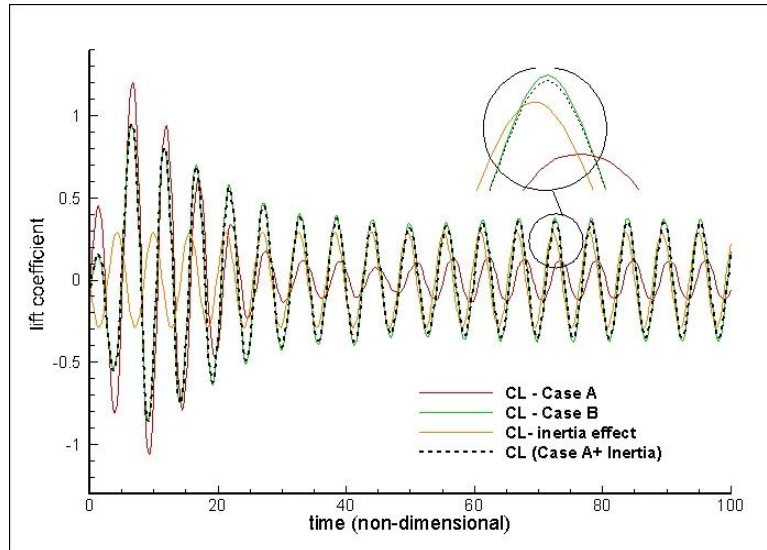


Figure 7-8: Using Froude-Krylov force (inertial force) to correct lift coefficient calculated in moving frame of reference

According to the equation (7-4), the effect of inertial force on the lift component for the above case is calculated as follows:

$$CL_{inertia} = \frac{\rho \pi \frac{D^2}{4} \times 1 \times \frac{d^2 y_{solid}}{dt^2}}{\frac{1}{2} \rho U^2 D \times 1} = \frac{\pi D}{2U^2} \frac{d^2 y_{solid}}{dt^2} \quad (7-21)$$

Where ρ is the density of the fluid flow; $\rho \pi \frac{D^2}{4} \times 1$, is the mass of the displaced flow by the cylinder and $\frac{d^2 y_{solid}}{dt^2}$ is the acceleration of the oscillating cylinder in the cross flow direction (referred to as \ddot{d} in equation (7-6)).

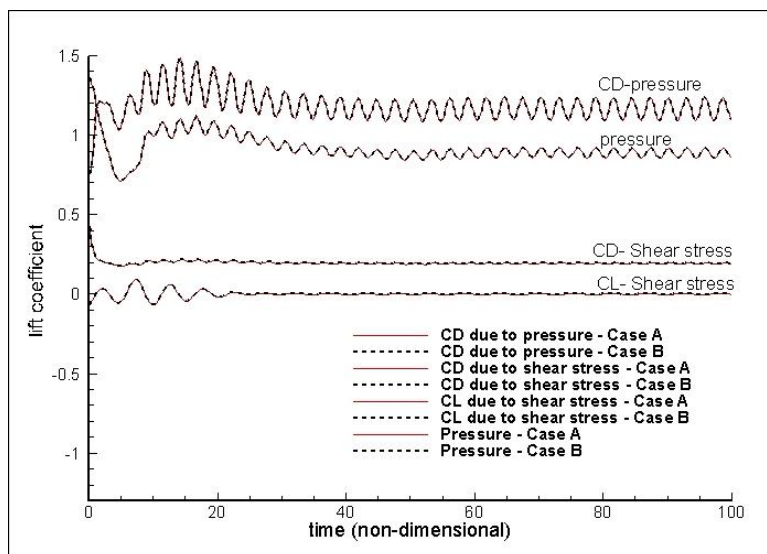


Figure 7-9: The drag (CD) due to pressure and shear stress, lift due to shear stress and pressure for cases A and B in the moving frame of reference

7.7.2 Moving frame verses inertial frame of references

The governing equations are solved in both the moving frame of reference (Section 7.3) and the inertial frame of reference (Section 7.4). In both approaches an IB interpolation method is used to enforce the immersed boundary. In the moving frame of reference, however, the interpolation formulas are not updated so that the simulation is less time consuming and the results are much smoother. In this section, a comparison between these two approaches is presented. In both cases a cylinder is forced to oscillate in the cross flow direction, the Reynolds number is 100 ($Re=100$), the amplitude of the oscillation is $0.2D$ and the frequency of the oscillation is 1.05 times the vortex shedding frequency (0.167). The Reynolds number is based on the free stream velocity and the cylinder diameter, D .

The lift and drag due to pressure for both approaches (moving and fixed frame of reference) are shown in the Figure 7-10. The results from the inertial frame of reference simulations show noise in the lift and drag signal due to pressure (dotted line). The reason for this is that the interpolation formulas are updated at each time step.

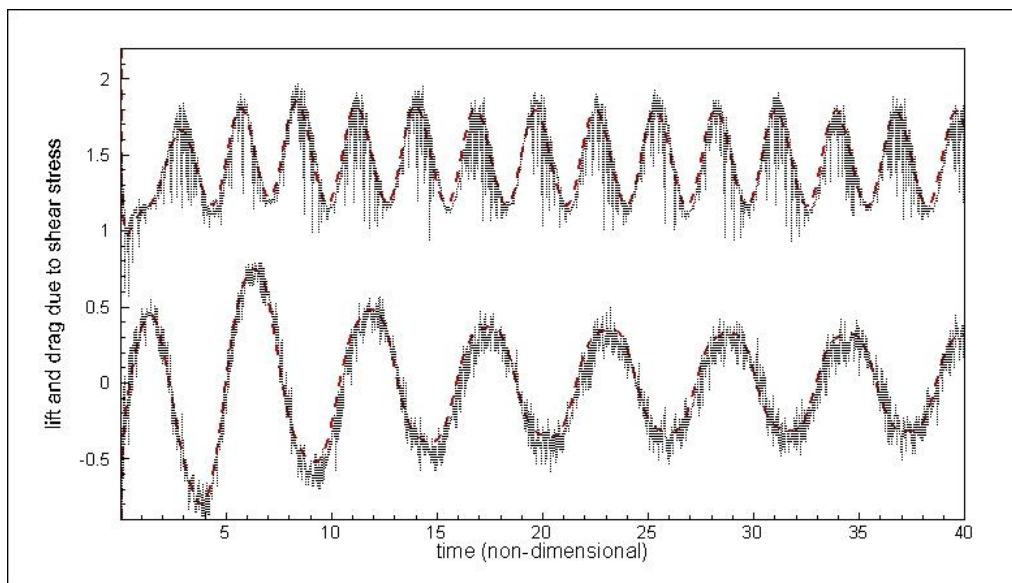


Figure 7-10: lift (lower curve) and drag (upper curve) due to pressure; dotted lines, inertia frame of reference(without smoothing); dash lines, moving frame of reference

Despite the noise that the inertial frame produces for the lift and drag due to pressure, both frames of reference calculate nearly the same values for the lift and drag coefficient after smoothing the graph of the inertial frame of reference results by omitting the noise (Figure 7-11).

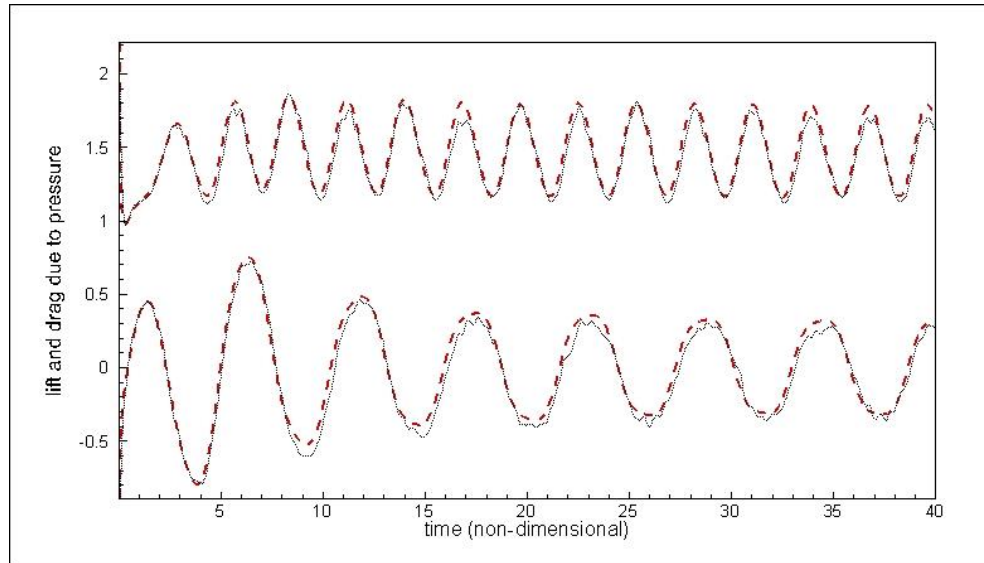


Figure 7-11: lift (lower curve) and drag (upper curve) due to pressure; dotted lines, inertia frame of reference (with smoothing); dashed lines, moving frame of reference

Figure 7-12 shows the lift and drag due to shear stress for both non-inertial and inertial frame of reference simulations. The simulation results do not show any noise in the lift and drag due to shear stress for both approaches. It can be concluded that the noise in the lift and drag coefficient are due to the calculation of the pressure. Also it can explain why the inertial frame of reference approach is so time consuming. Not only updating the interpolation formulas is taking extra simulation time but also the Pressure Poisson equation (as the most expensive part of the code) needs more iterations to converge due to the noise in the pressure.

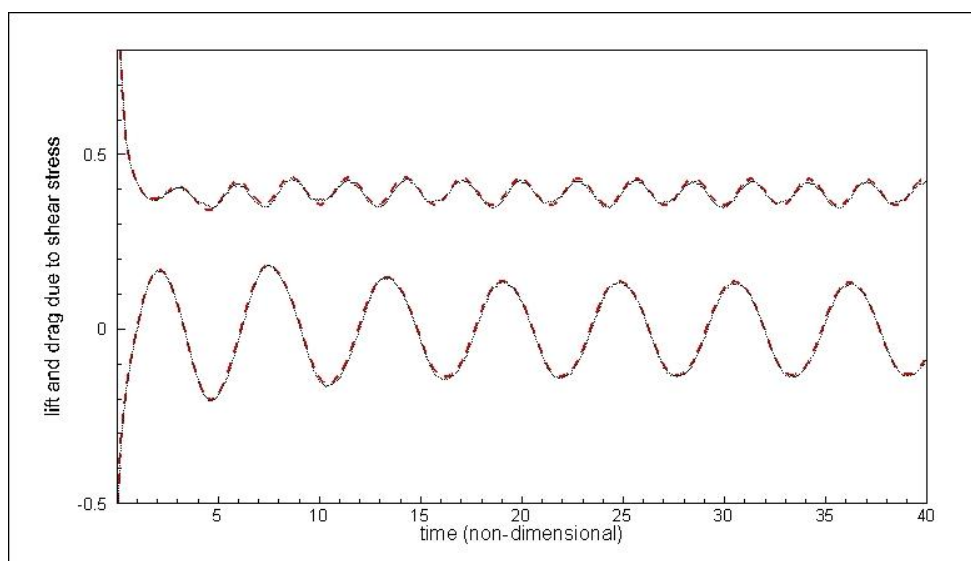


Figure 7-12: lift (lower curve) and drag (upper curve) due to shear stress; dotted lines, inertia frame of reference; dash lines, moving frame of reference

7.7.3 Cross flow oscillation of circular cylinder – validation

In this section, the simulation results induced by a transverse oscillation of a circular cylinder in a steady free stream are compared with those in the literature. The Reynolds number is calculated based on the cylinder diameter and the free stream velocity. The cylinder is forced to oscillate in the cross flow direction according to:

$$y_c(t) = A_{0y}\text{Sin}(\omega t) = A_{0y}\text{Sin}(2\pi f_e t) \quad (7-22)$$

Where, $y_c(t)$ is the location of circular cylinder that changes in time, A_{0y} is the amplitude of the transverse oscillation and f_e is the excitation frequency. The simulation is performed at low Reynolds numbers, $Re=185$ and $R=200$, $0.05 \leq A_{0y}/D \leq 0.6$ and $0.8 \leq f_e/f_s \leq 1.2$ in order to carry out a comparison with the results presented by Kim and Choi 2006 and Meneghini and Bearman 1995. f_s is the frequency of the vortex shedding for a stationary cylinder (Strouhal number). To calculate the Strouhal frequency at each Reynolds number, the flow around the stationary cylinder is simulated separately. The grid is distributed similarly to what is shown in Figure 5-1 and Figure 5-2. The number of the grid points in x (stream wise) and y (cross flow) direction are 531×478 , respectively. Around the cylinder a uniform grid with $dx=dy=0.025$ is used. The size of computational domain is $[-15D \text{ to } 15D]$ in both x and y directions.

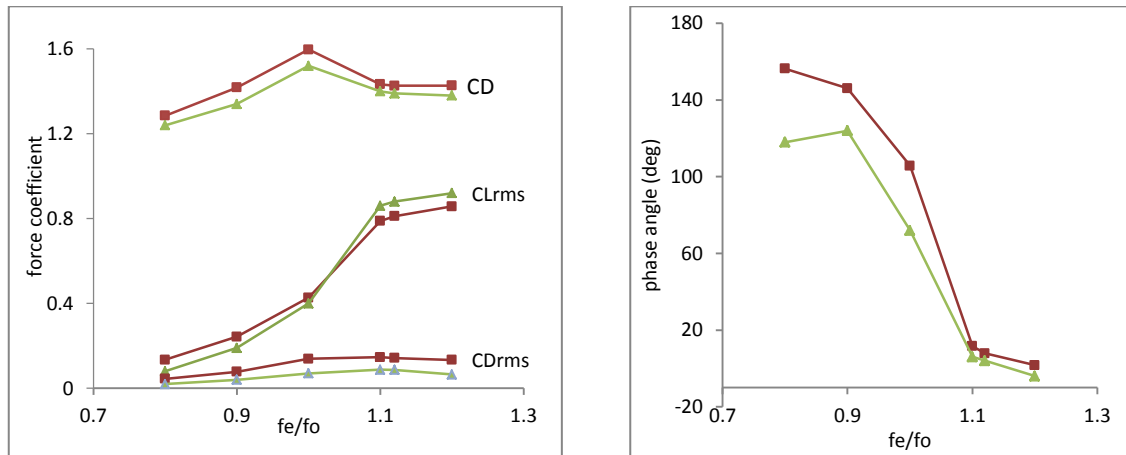


Figure 7-13: Force coefficient and phase angle versus f_e/f_s . Left-Mean drag coefficient (CD), rms of drag and lift fluctuation coefficients (CDrms and CLrms respectively); Right-Phase angle between CL and the vertical position of the cylinder. -■-, present study; -▲-, Kim & Choi 2006.

The flow governing equations are solved in the moving frame of references, in which the origin corresponds with the centre of the circular cylinder and a dirichlet

boundary condition ($u/u_\infty=1$ and $v=-v_{\text{cylinder}}$) is defined at the inlet; at the top and bottom (farfield) of the computational domain a Neumann boundary condition ($\frac{\partial u}{\partial y} = 0$) and a Dirichlet boundary condition ($v=-v_{\text{cylinder}}$) is used; and the convective boundary condition ($\frac{\partial u_i}{\partial t} + c \frac{\partial u_i}{\partial x} = 0$) is conducted at the outflow, where c is the space-averaged streamwise velocity.

The hydrodynamic forces (the force due to pressure and shear stress) are resolved in the x and y directions in the physical domain yielding F_x and F_y . These two forces are non-dimensionalized according to equations (4-66) and (4-67). In these equations u_∞ is the free stream velocity.

In the first place, the effect of frequency of excitation on the hydrodynamic forces at a constant amplitude of oscillation, $A_{0y}/D = 0.2$ is investigated. The simulation is performed at $f_e/f_s=0.8, 0.9, 1, 1.1, 1.12$ and 1.2 . In this simulation the Reynolds number is $Re=180$.

Figure 7-13-left shows the mean drag coefficient (CD) and the root mean square (rms) of the drag and the lift fluctuations (CDrms and CLrms, respectively); and Figure 7-13-right shows the phase angle between the lift coefficient and the location of cylinder. The simulation results are in good agreement with the results presented by Kim and Choi 2006. However, in the present study, the rms of the lift coefficient in the excitation frequencies below the Strouhal frequency are predicted to be smaller, while for the excitation frequency above the Strouhal number, these values are calculated to be higher than Kim and Choi's prediction. The Drag coefficient and the rms of the drag at all frequencies of excitation are predicted to be slightly higher than the results presented by Kim and Choi. In other words, the fluctuations in the drag coefficient are predicted to be higher in this research. This can be explained by the definition of the lift and drag coefficients to be either perpendicular to the free stream or perpendicular to the relative velocity.

Figure 7-14 shows how the frequency of excitation might affect the amplitude and the pattern of the hydrodynamic forces. In this figure the amplitude of oscillation is $0.2D$ and the Reynolds number is $Re=180$. For the frequencies of excitation lower than or equal to the Strouhal frequency the lift and drag coefficients reach a steady solution (synchronization) however at frequencies of excitation higher than the Strouhal frequency a beating phenomenon is observed. It can be concluded that for excitation frequencies above the Strouhal frequency, the boundary where lock-in occurs is much

closer to $f_e/f_s=1$ than for excitation frequencies below the Strouhal frequency. This is in complete agreement with the numerical results presented by Meneghini and Bearman 1995 and experimental results reported by Bearman and Curie 1979 where lock-in was observed only below the Strouhal frequency. However, at higher Reynolds numbers, experimental results reported by Koopmann 1967 show an almost symmetrical boundary of lock-in around the Strouhal frequency.

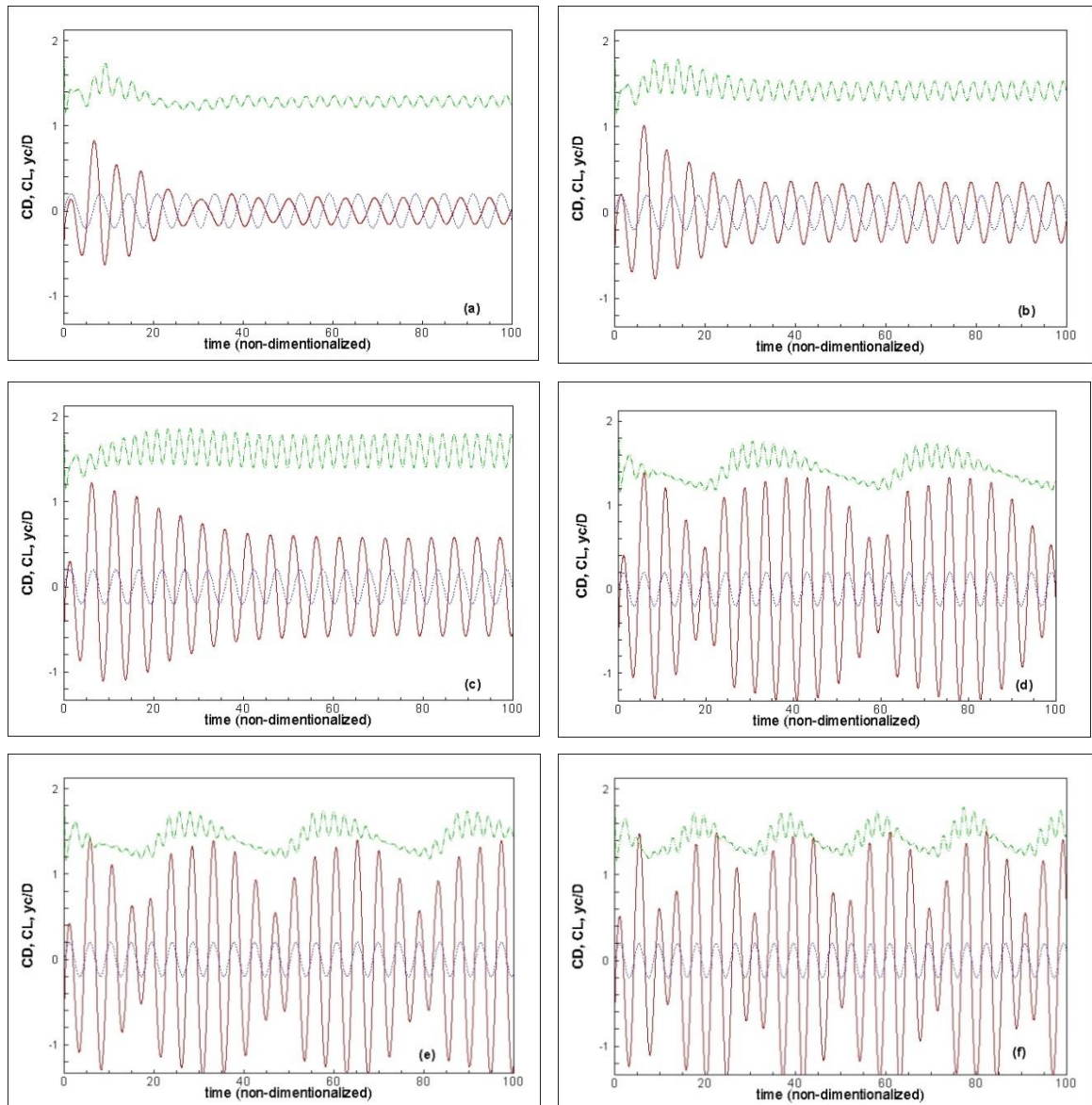


Figure 7-14: Drag (CD), Lift (CL) coefficient and y_c/D time history for $A/D=0.2$ and $Re=185$, (a) $f/f_s=0.8$, (b) $f/f_s=0.9$, (c) $f/f_s=1$, (d) $f/f_s=1.1$, (e) $f/f_s=1.12$, (f) $f/f_s=1.2$. CD: dash dot curve; CL: Continuous curve; y_c/D : dot curve

Figure 7-15 shows time histories of hydrodynamic forces at an excitation frequency of $f_e/f_s=0.75$, and a Reynolds number of $Re=200$ for four different amplitudes of

excitation, $A/D=0.25, 0.3, 0.45, 0.6$. The lock-in does not occur for $f_e/f_s < 0.70$. For the case $f_e/f_s=0.75$ and $A/D=0.25$ a very light beating phenomena is observed (albeit not very clear). By increasing the amplitude of excitation, a synchronization between the forcing excitation and the vortex shedding frequency occurs. It seems that at this range of frequencies ($f_e/f_s=0.75$) and for $A/D < 0.3$, the flow cannot decide whether to shed at the frequency of vortex shedding or at the frequency of excitation. Meneghini and Bearman 1995 got similar results, however, they observed synchronization above $A/D=0.5$.

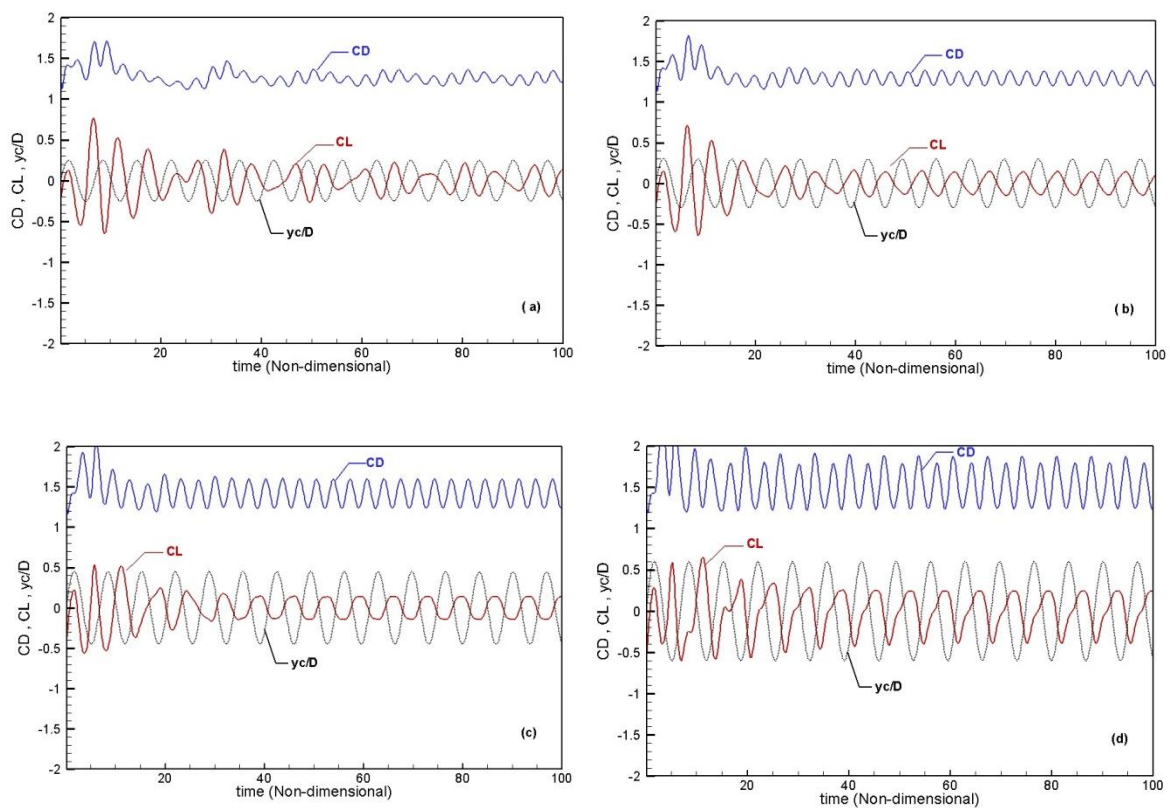


Figure 7-15: Drag (CD), Lift (CL) coefficient and y_c/D time history for $f_e/f_s=0.75$ and $Re=200$, (a) $A/D=0.25$, (b) $A/D=0.30$, (c) $A/D=0.45$, (d) $A/D=0.6$.

In Figure 7-16 to Figure 7-18 the lift and drag coefficient for three cases with $A/D=0.15$ and $f_e/f_s=0.9$, $A/D=0.25$ and $f_e/f_s=0.8$ and $A/D=0.05$ and $f_e/f_s=1.025$ are presented. The Results show excellent agreement with the results presented by Meneghini and Bearman 1995. Meneghini and Bearman used a mesh conforming method with a moving reference frame, but they did not directly include the effect of a moving frame inside the governing equations. Instead, they used the Froude-Krylov force to add the inertial effect to the hydrodynamic forces. In the first case (Figure

7-16), the phase difference between the excitation frequency and the lift coefficient is about 175° so that this case is inside the lock-in range (Figure 1-4). After starting the numerical simulation the frequency of the vortex-shedding gradually changes to the excitation frequency.

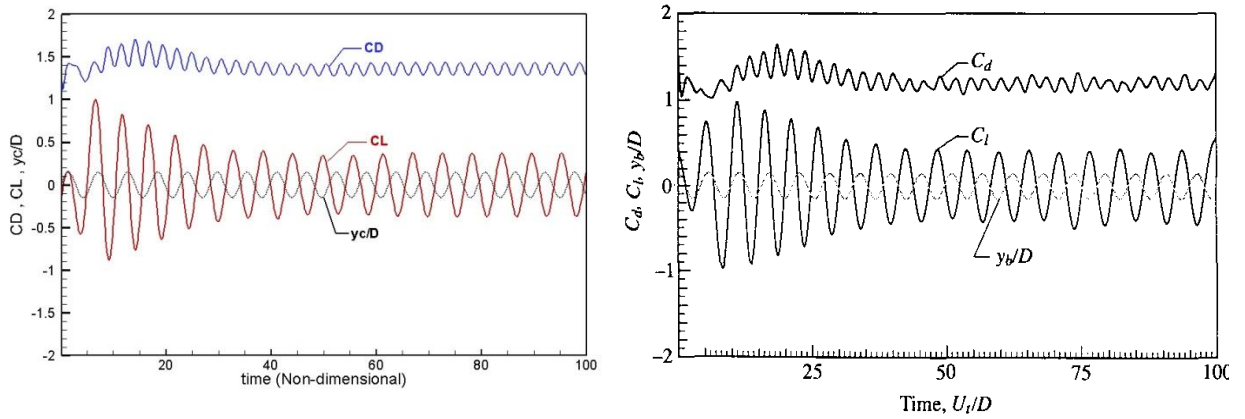


Figure 7-16: Drag (CD), Lift (CL) and y_c/D over time for $f_e/f_s=0.90$, $A/D=0.15$, $Re=200$. Left figure present study, right figure shows results of Meneghini and Bearman 1995.

By decreasing the excitation frequency, the amplitude of the excitation should increase to remain in the lock-in region (Figure 1-4). In Figure 7-17, the results of the case $A/D=0.25$ and $f_e/f_s=0.8$ is presented. In this case, the phase difference between the lift coefficient and the cylinder displacement is nearly 180° and the amplitude of the lift coefficient is lower than in the case of the stationary cylinder. Meneghini and Bearman 1995 explained that this could be due to the fact that inertial part of the lift force (due to the cross flow oscillation of cylinder) cancels out the lift due to vortex shedding which is dominant in stationary cases.

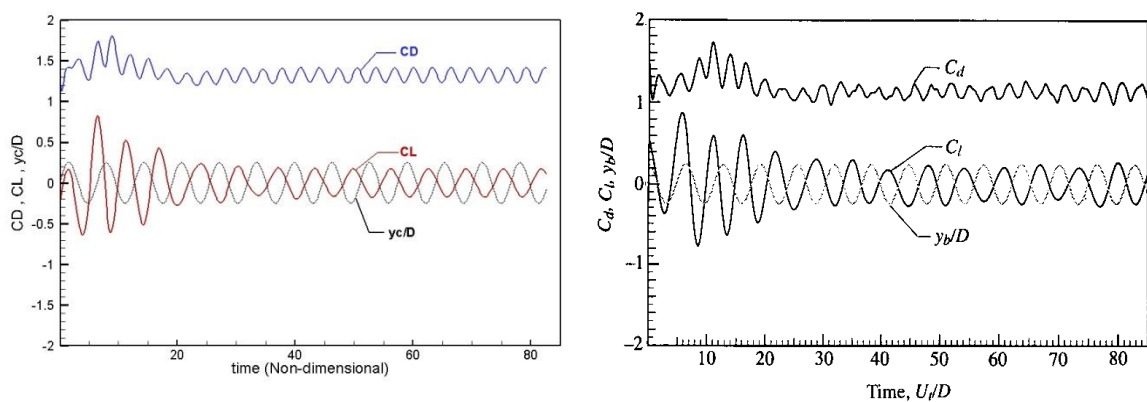


Figure 7-17: Drag (CD), Lift (CL) and y_c/D over time for $f_e/f_s=0.80$, $A/D=0.25$, $Re=200$. Left figure present study, right figure shows results of Meneghini and Bearman 1995.

A comparison for the case $A/D=0.05$ and $f_e/f_s=1.025$ is shown in Figure 7-18. The phase difference in this case between the frequency of the excitation and the cylinder displacement is about 15° . The numerical results show that the above $f_e/f_s=1.075$ lock-in does not occur for any amplitude of excitation.

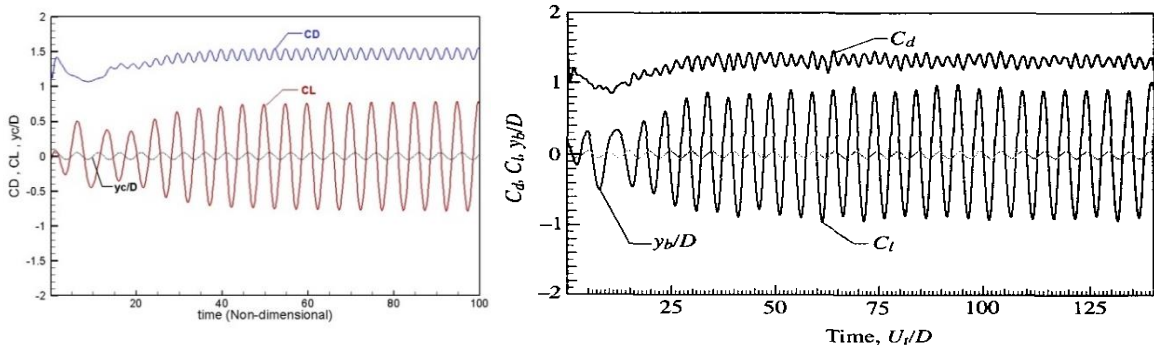


Figure 7-18: Drag (CD), Lift (CL) and yc/D over time for $f_e/f_s=1.025$, $A/D=0.05$, $Re=200$; left figure) present study; Right figure) Meneghini and Bearman 1995.

7.7.4 Vortex induced vibration in cross flow direction

In this section, to demonstrate the accuracy of the formulation provided in section 7.2.2, the flow around a circular cylinder in two dimensions with one degree of freedom in the cross flow direction is simulated. Several runs are performed at high and low mass ratios and the results are validated in compare to experimental and numerical results presented in the literature. Also the lock-in region is investigated.

In the first stage, the simulation results for the low mass ratio are compared to the results provided by Borazjani et al. 2008 and Ahn and kallinderis 2006 who employed IB method and ALE approach, respectively. In this case the Reynolds number, mass ratio and damping ratio are fixed at 150, 2, 0, respectively and the stiffness of the structural system is changed by varying the reduced velocity from 3 to 8. The size of computational domain is $[-15D, 15D]$ both in x and y direction and the cylinder is in the centre of the domain. Also, there is a uniform grid around the cylinder in the area $[-2D, 2D]$ in x and y direction, the uniform grid in this area is $dx=dy=0.025D$ and the non-dimensional time step is $dt=0.001$. Simulation of the flow around a stationary cylinder shows that the vortex shedding frequency or Strouhal number is $St=0.2$ at $Re=150$, therefore the lock-in phenomenon should occur around this frequency and hence reduce frequency of $Vr=5$.

The simulation results show that the applied IB reconstruction method accurately predicted the lock-in range, however, the maximum amplitude is predicted lower than

the one predicted by Borazjani et al. 2008 and Ahn & Kallinderis 2006. This might be due to the definition of the pressure boundary for the pressure Poisson equation.

Table 7-2 : Amplitude of oscillation (y_{max}/D) at various reduced velocity at constant Reynolds number, $Re=150$, and low mass ratio, $m^*=2$.

Reduced Velcotiy	$V_r=3$	$V_r=4$	$V_r=5$	$V_r=6$	$V_r=7$	$V_r=8$	$V_r=25$
Recent Research	0.04	0.42	0.38	0.30	0.20	0.06	0.03
Borazjani et al. 2008	0.06	0.52	0.48	0.43	0.38	0.08	-----
Ahn & Kallinderis 2006	0.06	0.56	0.52	0.42	0.37	0.08	-----

In the second stage, for the high mass ratio, the results presented by Anagnostopoulos and Bearman 1992 are used for validation; these results have been used for validation by several researcher (see for example, Yang et al. 2008, Li et al. 2002 among others). Therefore, to be able to compare the results, a mass ratio, $m^* = 149.10$ and a damping ration, $\xi = 0.0012$, is selected. The Reynolds number changes in the range of 90 to 140 which is euivalent to the reduced velocity of 5.02 to 7.81.

Table 7-3: Amplitude of oscillation (y_{max}/D) at various Reynolds number and reduced velocity at high mass ratio, $m^*=149.10$.

V_r and Re	$V_r=5.02$	$V_r=5.30$	$V_r=5.58$	$V_r=5.8$	$V_r=6.41$	$V_r=7.81$
	$Re=90$	$Re=95$	$Re=100$	$Re=105$	$Re=115$	$Re=140$
Anagnostopoulos and Bearman 1992 (exp.)	-----	0	0	0.54	0.5	0
Yang et al. 2008 (Nu.)	0	0.42	0.41	0.36	0.22	0
Schulz and Kallinderis 1998 (Neu.)	0	0.5	0.48	0.45	0.43	0
Present computation	0	0.1	0.24	0.36	0.22	0.0012

Simulation results (Table 7-3) show that the applied IB model in this study has a good agreement with the experimental results presented by Anagnostopoulos and Bearman 1992 in terms of predicting the range of reduced velocities which VIV occurs. For instance, the amplitude of oscillations reported by Yang et al. 2008 and Schulz & Kallinderis1998 at reduced velocity of $V_r=5.30$ ($Re=95$) are 0.42 and 0.5, respectively. However, at present study the amplitude of oscillation at $V_r=5.30$ is 0.1 which shows better agreement with the experimental results which shows zero amplitude at this reduce velocity.

Generally numerical results presented in the literature predict lower amplitude of oscillation in comparison to the experimental results. In the present study the same trend is observed. The reason behind this might be that although the reduced velocity is the same for both experiment and numerical simulation, the numerical simulation is normally performed at low Reynolds number in which vortices can be assumed two dimensional. Therefore the two dimensional numerical simulations cannot model three dimensional aspects of the vorticities which occur at higher Reynolds numbers.

7.8 Summary

In this chapter, the forced vibration and the vortex induced vibration of a bluff body in a uniform flow are discussed and the simulation results are compared and validated using well-established experimental and numerical bench marks. It was shown that the immersed boundary interpolation approach used for the stationary cylinder in chapters 5 and 6 could be applied for the moving immersed boundary as well. A comprehensive parametric study is performed to show how the computational domain parameters could affect the hydrodynamic forces and computational costs. Based on a parametric study, for low Reynolds numbers simulation a domain size of $[-15D, 30D] \times [-20D, -20D]$ in x and y direction respectively and a mesh size of $dx=dy=0.025$ around the immersed boundary are recommended.

To simulate moving boundaries two approaches were followed, using either a moving (non-inertial) frame or fixed (inertial) frame of references. Compared to the inertial frame of reference, the moving frame of reference results were much smoother and the computational time was lower. However, the moving frame approach is limited to simulations of single or synchronized moving bluff bodies in the fluid flow.

Also, it is shown by deriving the governing equations in the moving frame of reference that the Froude-Krylov force should not be added to the hydrodynamic forces to compensate for the inertial effect.

In addition it is shown that the noise in the results from the inertial frame of reference simulation is due to the calculation of the pressure which maybe improved by using a dual time step formulation or by using an accurate interpolation of the pressure at the immersed boundary. Moreover, the VIV simulations show that the results are in good agreement with the literature.

Chapter 8. Conclusion and Future work

The simulation of Fluid Structure Interaction (FSI) is a multi-disciplinary and a multi-physics problem and a full FSI simulation has to address many issues. The main goal in this research was to develop an FSI code to simulate Vortex-Induced Vibration (VIV) in the flexible riser application. The riser problem involves simulation of a flexible, slender structure with large displacement and bending in an unsteady fluid flow. A full simulation of this problem with the current knowledge and computational power is not feasible at the moment due to the multi-physics nature of the problem. Many research groups have worked in the past to model this problem and suggested various models and due to recent developments in computational power, CFD and Structural algorithms, a continuous progress in the research in this area is being made.

A partitioned strategy has been used to link the CFD and structural codes to be able to model the riser problem in a quasi-three dimensional using the strip theory. In the strip theory, the flow is computed in a number of two dimensional planes that are positioned at intervals along the pipes. The flow in each plane of the strip theory model is solved using a two dimensional Navier-Stokes solver. The response of the pipe to the flow loading is computed using various beam theories through a structural code. At this stage, a loose or strong coupling strategy will be used to alternatively pass the load from the flow to the structure and pass the new location of the structure to the flow solvers.

In an FSI problem, an initial and vital step for a feasible and accurate simulation is to study the physics of the problem. In this PhD thesis the main focus was to simulate the flow around a flexible body in the two dimensional plain. The outcome of this research will be used for a future modelling of the riser problem in the frame work of the strip theory. Using the strip theory for the riser problem, the problem was reduced to a well-documented simulation of the flexible circular cylinder in two dimensions. However, due to the fact that this two dimensional simulation will be used as a part of a bigger model special attention was needed. The first issue was that the two dimensional flow solver should be able to handle large displacements/deformations of the structure. Secondly, the flow solver should be computationally efficient. Thirdly, it was needed to integrate the flow solver with a structural code. Finally, the algorithm has to be

expandable to three dimensions to be able to model turbulent high Reynolds number flow in the future. Therefore, considering the physics of the problem and the restriction on the computational facilities, a comprehensive study of the available FSI approaches was conducted to find an appropriate algorithm that fulfils the set criteria.

8.1 Simulation approaches

There are two main simulation approaches for FSI problems: monolithic and partitioned approaches. In the monolithic approach both fluid and structure are formulated in the same mathematical framework and a unique algorithm is used to solve the entire fluid and structure domain. However, in order to link the CFD code with a structural code using the strip theory in future, a partitioned approach was preferred. Within this approach the fluid and the structure were treated as two separate computational entities and to be solved with their own respective discretisation and solution algorithm. Interface conditions were used to communicate information between the flow and structural solvers.

Another important feature for the FSI code is that the code should be able to model large displacements. There are two main discretisation method; the conforming method and the non-conforming method. In the former, the interface boundary condition is identical to the physical boundary condition making the interface location part of the solution requiring the grid to conform to the interface. By advancing in time, re-meshing might become necessary due to deformation/ displacement of structure. Therefore, this approach is expensive due to the regular re-meshing in every time step. In addition this method is good for low displacement due to inherit limitation in mesh deformation. However, in the non-conforming approach, the boundary location and interface conditions were imposed as constraints on the governing equations defined on a background Cartesian grid, and no re-meshing procedure is needed. As the solid boundary cuts the Cartesian grid, to define the proper boundary condition the flow governing equations need to be modified near the immersed boundary. The modifications of the governing equations near the structure are the subject of the immersed boundary method which were addressed and evaluated in this thesis.

Immersed boundary methods comprise various ways of enforcing boundary condition. By adopting the indirect forcing approach, interpolation/ reconstruction was used to enforce the moving boundary. In this approach however unlike the continuous forcing approach in which a diffused boundary is created, sharp interfaces are created.

The method also allows the possibility of modelling in three dimensions which is not easily possible in the cut cell approach due to its very complex application procedure in three dimensions.

In this PhD thesis a new IB interpolation/reconstruction method is proposed. In this method the interpolation is performed in a direction perpendicular to the IB boundary, similar to that proposed by Gilmanov & Sotiropoulos 2005. However, in this model a different logic and a direct approach is used to select the interpolation points without trial and error. The simulation results were compared with other interpolation methods proposed in the literature and the results of lift and drag coefficient showed a very good agreement between the methods.

The definition and calculation of the lift and the drag forces in an FSI problem using an IB approach is not a trivial problem. In this thesis two methods were conducted which were found to match well with one another; the direct integration of the pressure and shear forces on the immersed boundary and the application of the conservation of momentum in integral form. The lift and drag coefficient results were used to validate the methodology and the code for both a stationary circular cylinder and a flexible cylinder oscillating in the cross flow direction.

A circular cylinder oscillating in the cross flow direction was modelled in two dimensions as an initial stage in the study of the riser problem. At this stage two methods were presented, an inertial and a non-inertial frame of reference method. In the former, the Navier-Stokes equations were solved in an inertial frame of reference and the movement of the structure was modelled using an IB method. Due to the fact that at each time step the interpolation formulas were updated, the algorithm was relatively slow (time consuming). In the second method, the frame of reference was fixed to the cylinder and the fluid flow was solved using an observer point of view on the circular cylinder, therefore, the flow governing equation (Navier-Stokes equations) were defined and solved in a moving frame of reference. Although, this method was more efficient, it is only really suitable for a single object oscillating in the flow, for instance a single riser.

To solve the pressure Poisson equation, the normal gradient of the pressure at the immersed boundaries (Neumann boundary condition) was assumed to be zero in the case of a stationary cylinder in a uniform flow. However, the definition of the correct pressure boundary conditions for the FSI problem was a challenging issue because the structure undergoes acceleration relative to the flow. In this case, the gradient of the pressure in the perpendicular direction to the immersed boundary was calculated by

projection of the differential form of the momentum equation in that direction. This boundary was defined carefully to maintain the well-posed conditions for the pressure Poisson equation.

In the final stage of this thesis, some VIV simulations of a flexible cylinder in the cross flow direction were presented. To maintain the two dimensionality of the flow, the simulation was carried out at a low Reynolds number. The vortex shedding from the cylinder was creating oscillating force on the cylinder. These forces were used to solve the structural governing equations. In this thesis the equation of motion of an elastically supported cylinder is used. The force from the Eulerian flow field was transferred to the Lagrangian marker points on the solid boundary and the displacement and velocities of the moving boundary were interpolated to the flow domain to enforce no-slip boundary conditions. In the case of a rigid cylinder the force is transferred to the centre of the mass of the cylinder.

8.2 Validation of the results and feasibility of the method

The flow around a circular cylinder in two dimensions was taken as a benchmark due to its similarity to the physics of the riser. Also, the flow around a circular cylinder is a famous benchmark that has been used extensively to validate many FSI methodologies. Many experimental and analytical results are presented in the literature for this specific case. In addition, the choice of the overall size of the domain and the size of the grids near the immersed boundary were found to be important when accurate simulation results were desired from an FSI simulation in general and particularly when the IB approach is used. On the one hand, the parameters were selected in a way to give accurate, reliable and repeatable results whilst on the other, the methodology and the solution were found to be computationally inexpensive. Generally, it is important to determine the optimum parameters for an FSI problem in order to control the size of the problem. However, for a riser problem in which several two dimensional simulations and a structural code will be executed simultaneously using the optimum parameters for the simulation is vital. To achieve this objective, a comprehensive parametric study was performed to find the optimum range of the parameters for the domain which gives good results with minimum computational cost. This study was able to address some of the discrepancies found in the literature in respect of the reported Strouhal number, lift and drag coefficients.

For the flow around a circular cylinder at a low Reynolds number seven parameters were studied. Of these parameters the grid size around the IB, the entrance length before cylinder and the size of the domain in cross flow direction (blockage effect) were found to be the most important. The numerical results as well as results published in the literature showed that these parameters could significantly affect the results. For instance in the literature the drag coefficient for the steady flow around a stationary cylinder at $Re=100$ was reported in the range of 1.447 to 1.32 showing a 9% difference in the reported results. Also, the reported Strouhal numbers for the same cases varied from 0.182 to 0.164 showing discrepancies of up to 10% in the results. By relating the simulation results to the simulation parameters it was possible to explain these discrepancies. Some of these differences stemmed from the size of domain in the numerical calculation rather than the methodology of the solution. The results of the parametric study at $Re=100$ showed that if the entrance length increased from 5D to 10D the Strouhal number, lift and drag coefficient tends to decrease by about 10%. A further enlargement of the domain behind the cylinder had negligible effect on the Strouhal number, lift and drag coefficient. Therefore, for this specific problem an inflow length of 10D before the cylinder was found to be optimum. Similarly, the size of the domain in the cross flow direction (blockage effect) was also found to be important.

The mesh refinement study for the drag coefficient showed an interesting behaviour between the drag coefficient's components (pressure drag and friction drag). As far as the author is aware, this issue has not been reported before in the literature. The numerical results showed that the drag coefficient was less affected (about 3%) than the lift coefficient (about 12%) when changing the size of the mesh from 0.1 to 0.00625 (4 times) in the mesh refinement study. This issue can be explained by the fact that the components of the drag coefficient were reversely responding to the grid size. i.e. by further refinement of the grid, the drag due to the pressure converged to a lower value while the drag due to friction converged to a higher value. This shows that the drag is less sensitive to the size of the grid.

Additionally, a comparison was presented of IB Interpolation / Reconstruction methods. Four different interpolation methods were compared with the proposed interpolation method in this thesis. The numerical results showed that the proposed interpolation method was stable and gave accurate results compared to other linear and bi-linear methods. Also this method does not suffer from the problem associated with the

bi-linear methods in finding interpolation points in the orthogonal directions when modelling high curvature IB.

To fulfil the objective of this research, in the final step the numerical simulation of the flow around a cylinder oscillating in the cross flow direction was presented. This problem was presented in both fixed and moving frames of reference and the results were found to match well. The simulation of the flow around a cylinder is a well-known problem and has been used to validate FSI methodology by many researchers. To the best knowledge of author this was the first time that this problem was modelled using a sharp immersed boundary Interpolation/ Reconstruction technique along with a moving frame of reference.

In the next part, the main draw backs of the applied methodology will be discussed and also some works will be proposed to address these issues as future research.

8.3 Drawbacks verses advantages of the IB interpolation

The Immersed Boundary with an Interpolation/reconstruction approach was used in this thesis to enable modelling moving boundaries with large displacements. As any other FSI method, this method has also some drawbacks. The most important of which, in comparison to an ALE approach is that it is not straight forward to apply the boundary conditions on the moving boundary, especially for curved boundaries. This is common with all IB approaches and becomes more complicated because a staggered grid arrangement is used in the discretisation of the governing equations.

Another important issue was the calculation of the hydrodynamic forces at the immersed boundaries. Calculating the lift and the drag forces on the IB immersed boundary was not a trivial problem, especially, when an Interpolation method was applied to the FSI problem. However, using momentum principle could help to address this problem.

Despite these shortcomings, however, it is concluded from the experience gained from this research work that the IB the interpolation/Reconstruction method, is considered as an appropriate method to apply to the flexible riser problem with large displacement/deformation using the strip theory approach. Firstly, this method could handle large displacements where a conforming method like ALE would be computationally more expensive. Also as the IB method adopted here was a sharp interface method, unlike the IB forcing approach it does not create diffuse boundaries near solid boundaries. This method can be simply developed to three dimensions, where

the IB cut cell method would become very expensive and complicated. Finally, this method does not create a secondary flow inside the solid boundary, unlike the ghost cell methods which create non-physical flow inside the solid boundaries.

8.4 Future work

This PhD research was part of a larger research project which aims to model the VIV for a slender oil riser and publication of some journal paper are planned during the completion of project in near future. In this study, the methodology to solve the flow around a flexible circular cylinder in two dimensions was addressed. This will be used as part of strip theory to model FSI for whole flexible risers used in the offshore industries. In this section, the suggestions for future work are all directly related to this PhD thesis.

- All the simulations in this thesis, including the parametric study, were limited to a low Reynolds number, $Re=100$. The parametric study to show the effect of the Reynolds number on the FSI parameters is recommended for further low Reynolds numbers $40 < Re < 200$.
- In a real riser problem, the Reynolds number is of order of $O(10^4)$, therefore adding a suitable algorithm to model the turbulence is necessary.
- The Neumann boundary conditions for the pressure, $\frac{\partial p}{\partial n}$, do not noticeably affect the lift and drag forces. A proper parametric study will help to understand the range of oscillations that the ‘standard’ boundary condition $\frac{\partial p}{\partial n} = 0$ is sufficiently accurate.
- The moving frame of reference presents promising results for the cylinder oscillation in the cross flow direction. It is suggested to further develop this simulation for inline oscillations using the IB and interpolation approaches.
- Finally, to improve the results for the moving cylinder in the inertial frame of reference, it is suggested to use a dual time integration to reduce the fluctuation of the response. Also, this method is very slow in comparison to the moving frame of reference approach. It is suggested to use a parallel processing capability to improve this method. For simulations with more than one cylinder oscillating in the flow domain, this method offers the only solution, as the moving frame of reference method cannot be used in multi-cylinders simulations.

References

- Ahn, H.T. and Kallinderis, Y. 2006, strongly coupled flow/structure interactions with geometrically conservative ALE scheme on general hybrid meshes, *Journal of Computational Physics*, 219(2), pp. 671-696.
- Anagnostopoulos, P. and Bearman, P. 1992, Response characteristics of a vortex-excited cylinder at low Reynolds numbers, *Journal of Fluids and Structures*, 6(1), pp. 39-50.
- Baaijens, F.P., 2001, A fictitious domain/mortar element method for fluid-structure interaction, *International Journal for Numerical Methods in Fluids*, 35(7), pp. 743-761
- Badia, S., Nobile, F., Vergara, C. 2008, Fluid-Structure partitioned procedures based on robin transmission conditions, *journal of computational physics*, vol. 227, pp. 7027-75051.
- Balaras, E. 2004, Modelling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations, *J. computers and fluids*, vol. 33 pp. 375-404.
- Batchelor, G. K. 1967, *An Introduction to Fluid Dynamics*, Cambridge University Press, ISBN 0-521-66396-2
- Bazilevs, Y., Calo V.M., Zhang, Y., Hughes, T.J.R. 2006 Iso geometric fluid-structure interaction analysis with applications to arterial blood flow, *Computational Mechanics*, vol. 38 pp. 310-322
- Bazilevs, Y., Calo, V.M., Tezduyar, T.E., Hughes, T.J.R. 2007, γ discontinuity-capturing for advection-dominated processes with application to arterial drug delivery, *International Journal for Numerical Methods in Fluids*, vol. 54 pp. 593-608.
- Bazilevs, Y., Hsu, M.C., Kiendl, J., Uehner, R. W., Bletzinger K.U. 2011, 3D simulation of wind turbine rotors at full scale. Part II: Fluid-structure interaction modelling with composite blades, *International Journal for Numerical Methods in Fluids*, vol. 65 pp. 236-253.
- Bazilevs, Y., Takizawa, K., Tezduyar, T.E. 2013a, *Computational Fluid-Structure Interaction: Methods and Applications*, Wiley, Chichester.
- Bazilevs, Y., Korobenko, A., Deng, X., Tippmann, J., Hsu, M.C. 2013b, Wind turbine simulation: structural mechanics, FSI and computational steering, V *International Conference on Computational Methods for Coupled Problems in Science and Engineering COUPLED PROBLEMS*, pp. 229-240.
- Bearman, P.W. 1969, On vortex shedding from a circular cylinder in the critical Reynolds number regime, *Journal of Fluid mechanics*, vol. 37, no. 03, pp. 577-585
- Bearman, P.W., Curie, I.G. 1979, Pressure fluctuation measurements on an oscillating circular cylinder, *Journal of fluid mechanics*, vol. 91 pp. 661-677.
- Belytschko, W., Liu, W., Moran, B. 2000, *Nonlinear Finite elements for Continua and Structures*, ed. Wiley, Chichester.
- Berger, M. and Aftosmis, M. 1998, Aspects (and aspect ratios) of Cartesian mesh methods, *Sixteenth International Conference on Numerical Methods in Fluid Dynamics*. Springer, pp. 1-12
- Beyer, R.P., LeVeque, R.J. 1992, Analysis of a one dimensional model for the immersed boundary method, *SIAM Journal Number Annual* vol. 29 pp. 332-364
- Bishop, R.E.D., Hassan, A.Y. 1964, the lift and drag forces on a circular cylinder oscillating in a flowing fluid. *Proc. R. Soc. London Ser*, vol. 277, pp. 51-75.
- Blackburn, H.M. and Henderson, R.D. 1999, A study of two-dimensional flow past an oscillating cylinder, *Journal of Fluid Mechanics*, 385, pp. 255-286.

- Bletzinger, K.U, Wuchner, R., Kupzok, A. 2006, Algorithmic treatment of shells and free form-membranes in FSI, In Fluid–Structure Interaction, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, pp. 336–355.
- Briscolini, M., Santangelo, P. 1988, Development of the mask method for incompressible unsteady flows, *J. Computational Physics*, pp. 84-57.
- Borzajani, I., Ge, L., Sotiropoulos, F. 2008, Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies, *Journal of Computational Physics* vol. 227 pp. 7587–7620, Available online at www.sciencedirect.com
- Bungartz, H.J, Schafer, M. 2006 (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, pp. 82–100.
- Causin, P., Gerbeau, J.F., Nobile, F. 2005, Added-mass effect in the design of partitioned algorithms for fluid–structure problems, *Comput. Methods Appl. Mech. Eng.*, vol. 194, pp. 4506–4527.
- Ceniceros, H.D., Fisher, J.E., Roma, A.M. 2009, efficient solutions to robust, semi-implicit discretization of the immersed boundary method, *Journal of computational physics*, Vol. 228, pp. 7137-7158.
- Choi, J.I. Oberio, R.C., Edwards, J.R., Rosati, J.A. 2007, An immersed boundary method for complex incompressible flows, *Journal of computational physics* vol. 224, pp. 757-784.
- Clarke, D., Salas, M., Hassan, H. 1986, Euler calculations for multi-element airfoils using Cartesian grids, *AIAA J.* vol. 24 pp. 1128–1135.
- Codina, R. (2001) 'Pressure stability in fractional step finite element methods for incompressible flows', *Journal of Computational Physics*, 170(1), pp. 112-140.
- Corbalan Gois E.R., de Souza L.F. 2010, An Eulerian Immersed Boundary Method for flow simulations over stationary and moving rigid bodies. *J. Braz. Soc. Mech. Sci. & Eng.* [online]. vol.32, pp. 477-484, Available online: <<http://www.scielo.br/scielo>>.
- Deparis, S., 2004, Numerical analysis of axisymmetric flows and methods for fluid-structure interaction arising in blood flow simulation, EPFL.
- Dettmer, W., 2004, Finite element modeling of fluid flow with moving free surfaces and interfaces including fluid–solid interaction. Ph.D. Thesis, University of Wales Swansea.
- Dettmer, W., Peric, D., 2006a, A computational framework for fluid–rigid body interaction: finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering*, vol. 195 pp. 1633–1666.
- Dettmer, W., Peric D.A. 2006b, computational framework for fluid–structure interaction: finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering* (Available Online).
- Donea, J., Giuliani, S., Halleux, J. 1982, An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid–structure interactions, *Computer Methods in Applied Mechanics and Engineering* vol. 33, pp. 689–723.
- Fadlun, E.A., Verzicco, R., Orlandi, p., Mohd-Yusof, j. 2000, Combined immersed boundary finite difference methods for three-dimensional complex flow simulations, *Journal of computational physics* vol. 161, pp. 35-60.
- Faludi, R., Szulik, M., D'hooge, J., Herijgers, P., Rademakers, F., Pedrizzetti, G., Voigt, J.U. 2010. Left ventricular flow patterns in healthy subjects and patients with prosthetic mitral valves: an in vivo study using echocardiographic particle image velocimetry, *Journal of Thoracic and Cardiovascular Surgery*, vol. 139, no. 6, pp. 1501–1510

- Farhat, C., Lesoinne, M., LeTallec, P. 1998, Load and motion transfer algorithms for fluid/structure interaction problems with nonmatching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity, *Computer Methods in Applied Mechanics and Engineering* vol. 157, no.1, pp. 95–114.
- Farhat, C., van der Zee, K. G., Geuzaine, P. 2006, Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity, *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, pp. 1973–2001.
- Fauci L.J., Fogelson A.L. 1993, Truncated newton methods and the modelling of complex immersed elastic structures, *Communication on pure and applied mathematics*, vol. 46, no. 6, pp. 787-818.
- Feng, C.C. 1968, The measurements of vortex-induced effects in flow past a stationary and oscillating circular and D-section cylinders. Master's thesis, University of British Columbia, Vancouver, Canada.
- Fernandez, M., Moubachir, M. 2005, A Newton method using exact jacobians for solving fluid–structure coupling, *Computers and Structures*, vol. 83 no.1-3, pp. 127–142.
- Ferziger, J.H., Peric M. ed. 2002, computational methods for fluid dynamics, third edition, ISBN 3-540-42074-6 Springer-Verlag Berlin Heidelberg NewYork.
- Feynman, R. P., Leighton, R. B., Sands, M. 1963, *The Feynman Lectures on Physics*, Reading, Mass.: Addison-Wesley, ISBN 0-201-02116-1, vol. 1, no. 9–4 and no.12–1
- FitzHugh, R. 1961, Impulses and physiological states in theoretical models of nerve membrane. *Biophysical J.* Vol. 1, pp.445–466.
- Förster, C., Wall W.A., Ramm E. 2007, Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows, *Comput. Methods Appl. Mech. Eng.*, 196 (7), pp. 1278–1293.
- Ghias, R., Mittal, R., Lund, TS. 2004, A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries. *AIAA* vol. 80
- Ghias, R., Mittal, R., Dong, H. 2007, A sharp interface immersed boundary method for compressible viscous flows, *journal of computational physics* vol. 225, pp. 528-553.
- Gilmanov, A., Sotiropoulos, F. 2005, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, *Journal of Computational physics*, vol. 207, pp. 457-492.
- Gilmanov, A., Sotiropoulos, F., Balaras, E. 2003, A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids, *J. Computational physics* vol.191, pp. 660-669.
- Glowinski, R., Pan, T.W., Hesla, T.I., Joseph, D.D. 1999, A distributed Lagrange multiplier/fictitious domain method for particulate flows. *Int. J. Multiphase Flow* vol. 25, pp. 755-794.
- Glowinski, R., Pan, T.W., Hesla, T.I., Joseph, D.D., Periaux, J. 2000, A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. *J. Comput. Phys.* Vol. 169, pp. 363-426.
- Goldstein, D., Handler, R., Sirovich, L. 1993, Modeling a no-slip flow boundary with an external force field, *journal of computational physics* vol. 105, no. 2, pp. 354-366.
- Gomez-Iradi, S.G, Steijl, R., Barakos, G.N. 2009, Development and validation of a CFD technique for the aerodynamic analysis of HAWT. *Journal of Solar Energy Engineering*, vol. 131, pp. 031009–1–13.
- Gresho, P.M., Sani R.L. 1987, On pressure boundary conditions for the incompressible Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, vol. 7, pp. 1111–1145.

- Haines, E., 1994, Point in polygon strategies, *Graphics gems IV*, 994, pp. 24-26.
- Harlow, F.H, Welch, J.E. 1965, Numerical calculation of time dependent viscous incompressible flow of fluid with free surface, *Phys. Fluids*, vol. 8, pp. 2182-2189.
- Hernández, A., Valdés, J.G. 2013, Determination of 3d wind induced vibration of cables for cable-stayed bridges, V International Conference on Computational Methods for Coupled Problems in Science and Engineering COUPLED PROBLEMS, pp. 458-466.
- Hong, G.R., Pedrizzetti G., Tonti G., Li P., Wei Z., Kim J.K., Baweja A., Liu S., Chung N., Houle H., Narula J., Vannan M.A. 2008, Characterization and quantification of vortex flow in the human left ventricle by contrast echocardiography using vector particle image velocimetry, *Journal of the American College of Cardiology Imaging*, 1 (6), pp. 705-717
- Hou, T.Y., Shi, Z. 2008, An efficient semi-implicit immersed boundary method for the Navier-Stokes equation, *journal of computational physics*, Vol. 227, pp. 8968-8991.
- Hou, G., Wang, J., Layton, A. 2012, Numerical Methods for Fluid-Structure Interaction: A Review ARTICLE, *Commun. Comput. Phys.* doi: 10.4208/cicp.291210.290411s, Vol. 12, No. 2, pp. 337-377.
- Hubner, B., Walhorn, E., Dinkler, D. 2004, A monolithic approach to fluid-structure interaction using space-time finite elements, *Journal of computer methods in applied mechanics and engineering*, vol. 193, no. 23-26, 2004, pp. 2087-2104.
- Hsu, M.C., Akkerman, I., Bazilevs Y. 2013, Finite element simulation of wind turbine aerodynamics: Validation study using NREL Phase VI experiment. *Wind Energy*, 2013. Accepted.
- Iaccarino, G., Verzicco, R. 2003, Immersed boundary technique for turbulent flow simulations. *Appl. Mech. Rev.* vol. 56 pp. 331-47.
- Idelsohn, S.R., Pin, F.D., Rossi, R., Oñate, E. 2009, Fluid-structure interaction problems with strong added-mass effect, *Int. J. Numer. Methods Eng.*, vol. 80, pp. 1261-1294.
- Kang, S. 2008, An improved immersed boundary method for computational of turbulent flows with heat transfer, P.h.D thesis, Stanford University.
- Kang, S., Iaccarino, G., Moin, P. 2009, Accurate immersed boundary reconstructions for viscous flow simulations, *AIAA Journal*, vol. 47, no. 7.
- Karniadakis, G.E. and Triantafyllou, G.S. 1992, Three-dimensional dynamics and transition to turbulence in the wake of bluff objects, *Journal of Fluid Mechanics*, 238, pp. 1-30.
- Khurram, R.A., Masud, A. 2006, A multi-scale/stabilized formulation of the incompressible Navier-Stokes equations for moving boundary flows and fluid-structure interaction. *Computational Mechanics*, vol. 38, pp. 403-416.
- Kiendl, J., Bletzinger, K.U., Linhard, J., uchner, R.W. 2009, Isogeometric shell analysis with Kirchhoff-Love elements, *Computer Methods in Applied Mechanics and Engineering*, vol. 198 pp. 3902-3914.
- Kim, D., Choi, H. 2006, Immersed boundary method for flow around an arbitrarily moving body, *Journal of computational physics* vol. 212, pp. 662-680.
- Kim, J., Kim, D., Choi, H. 2001, An immersed boundary finite volume method for simulation of flow in complex geometries, *Journal of computational physics* vol.171, pp.132-140.
- Kim, H.Y., Kim, H.J., Kim, T.H. 2013, Evaluation of automotive weather-strip by coupled analysis of fluid-structure-noise interaction, V International Conference on Computational Methods for Coupled Problems in Science and Engineering, COUPLED PROBLEMS 2013, pp. 1365-1372.

- Kirkpatrick, C., Unger, R., Krump-Konvalinkova, V., Peters, K., Schmidt, H. and Kamp, G., 2003, Experimental approaches to study vascularization in tissue engineering and biomaterial applications, *Journal of Materials Science: Materials in Medicine*, 14(8), pp. 677-681.
- Koopmann, G. 1967, The vortex wakes of vibrating cylinders at low Reynolds numbers, *Journal of Fluid Mechanics*, 28(03), pp. 501-512.
- Korobenko, A., Hsu, M.C., Akkerman, I., Tippmann, J., Bazilevs, Y. 2013, Structural mechanics modelling and FSI simulation of wind turbines, *Mathematical Models and Methods in Applied Science*, vol. 23, pp. 249–272, DOI: 10.1142/S0218202513400034.
- Kravchenko, A., Moin, P. and Moser, R., 1996, Zonal embedded grids for numerical simulations of wall-bounded turbulent flows, *Journal of Computational Physics*, 127(2), pp. 412-423.
- Kuttler, U., Forster, C., Wall, W.A. 2006, A solution for the incompressibility dilemma in partitioned fluid–structure interaction with pure Dirichlet fluid domains. *Computational Mechanics*, vol. 38, pp. 417–429.
- Kuttler, u., Wall, W.A. 2008, Fixed-point fluid-structure interaction solvers with dynamic relaxation, *Journal of computational mechanics*, vol. 43, pp. 61-72.
- Kvitting, J.P.E., Dyverfeldt P., Sigfridsson A., Franzén S., Wigström L., Bolger A.F., Ebbers T. 2010, In vitro assessment of flow patterns and turbulence intensity in prosthetic heart valves using generalized phase-contrast MRI, *Journal of Magnetic Resonance Imaging*, Vol. 31, No. 5, pp. 1075–1080
- Lai, M.C., Peskin C.S. 2000, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, *Journal of Computational. Phys*, vol. 160, pp. 705–19.
- Le, T.P., Mouro, J. 2001, Fluid structure interaction with large structural displacements, *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 24, pp. 3039–3067.
- Le, D.V., Khoo, B.C., Peraire, J. 2006, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, *Journal of Computational Physics*, vol. 220, pp. 109-138
- Le, T.B., Sotiropoulos, F. 2013, fluid structure interaction of an aortic heart valve prosthesis driven by an animated anatomic left ventricle, *J. of computational physics*, vol. 244, pp. 41-62.
- Lee, J., Niederer, S., Nordsletten, D., Le, Grice, I., Smail, B., Kay, D., Smith, N. 2009, Coupling contraction, excitation, ventricular and coronary blood flow across scale and physics in the heart, *Philosophical Transactions of the Royal Society A*, vol. 367 (1896), pp. 2311–2331.
- Lee, L., LeVeque, R.J. 2003, An immersed interface method for incompressible Navier–Stokes equations, *SIAM J. Sci. Comput*, vol. 25 no. 3, pp. 832–856.
- LeVeque, R.J., and Li, Z. 1994, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM Journal on Numerical Analysis*, vol. 31, pp. 1091-1044.
- LeVeque, R.J. and Li, Z. (1997) 'Immersed interface methods for Stokes flow with elastic boundaries or surface tension', *SIAM Journal on Scientific Computing*, 18(3), pp. 709-735.

- Li, L., Sherwin, S.J., Bearman, P.W. 2002, A moving frame of reference algorithm for fluid/structure interaction of rotating and translating bodies, *Int. J. Numer. Meth. Fluids*, vol. 38, pp. 187-206.
- Li, Y., Carrica, P.M., Paik, K.J., Xing, T. 2012, Dynamic overset CFD simulations of wind turbine aerodynamics, *Renewable Energy*, vol. 37 pp. 285–298.
- Lima E Silva, A.L.F., Silveira-Neto, A., Damasceno, J.J.R. 2003, Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method, *Journal of Computational Physics* vol. 189, no. 2, pp. 351–370.
- Lohner, R., Cebal, JR., Yang, C., Baum, JD., Mestreau, EL., Soto, O. 2006, Extending the range of applicability of the loose coupling approach for FSI simulations. In *Fluid–Structure Interaction, lecture note in computational science and engineering volume*, Springer: Berlin, vol. 53, pp. 82-100.
- Maday, Y., Mavriplis, C., Patera, A.T. 1989, Nonconforming mortar element methods: application to spectral discretizations, in *Domain decomposition methods* (Los Angeles, CA, 1988), SIAM, Philadelphia, PA, pp. 392-418.
- Masud, A., Bhanabhagvanwala, M., Khurram. R.A. 2007, An adaptive mesh rezoning scheme for moving boundary flows and fluid–structure interaction. *Computers and Fluids*, vol. 36, pp. 77–91.
- Mayo, A.A., Peskin, C.S. 1993, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries. *J. of Fluid Dynamics in Biology* (Cheer AY and van Dam DP, eds.), Oxford University Press, 1993, paper No.11., *Contemporary Mathematics* vol. 141, pp. 261-277,
- Meneghini, J.R., Bearman, P.W. 1995, Numerical simulation of high amplitude oscillatory flow about a circular cylinder, *J. of fluids and structures*, vol. 9, pp. 435-455.
- Mittal, R., Bonilla, C., Udaykumar, H.S. 2003, Cartesian grid methods for simulating flows with moving boundaries. In *Computational Methods and Experimental Measurements* ed. XI, Brebbia C.A., Carlomagno G.M., Anagnostopoulos p.
- Mittal, R., Seshadri, V., Udaykumar, H.S. 2004, Flutter, tumble and vortex induced autorotation, *Theor. Comput. Fluid Dyn.*, vol. 17 no. 3, pp. 165–70.
- Mittal, R., Iaccarino, G. 2005, Immersed Boundary Methods, *Annual Review of Fluid Mechanics*, vol. 37, pp. 239–261. doi:10.1146/annurev.fluid.37.061903.175743
- Mohd-Yusof J. 1997, Combined immersed boundaries/B-Spline Methods for simulations of flows in complex geometries, *CTR Annual Research Briefs*, NASA AMES/Stanford University.
- Mok, D.P., wall, W.A. 2001, Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures, In: Wall, W.A., Bletzinger, k.U., Schweitzerhof, K., (eds) *Trends in computational structural mechanics*.
- Moller, T., Trumbore B. 1997, Fast, minimum storage ray-triangle intersection, *Journal of Graphics Tools*, vol. 2 no.1, pp. 21-28.
- Mori, Y., Peskin, C.S. 2008, Implicit second order immersed boundary methods with boundary mass, *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 25-28, pp. 2049-2067.
- Morison, J., Johnson, J. and Schaaf, S., 1950, The force exerted by surface waves on piles, *Journal of Petroleum Technology*, 2(05), pp. 149-154.
- Newren E., Fogelson A., Guy R., Kirby M. 2008, A comparison of implicit solvers for the immersed boundary equations, *computer methods in applied mechanics and engineering*, vol. 197, pp. 2290-2304.

- Newman, D., Karniadakis, G.E. 1988, A direct numerical simulation study of flow past a freely vibrating cable. *Journal of Fluid Mechanics*, vol. 344, pp. 95–136.
- Ohayon, R. 2001, Reduced symmetric models for modal analysis of internal structural-acoustic and hydroelastic-sloshing systems. *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 3009–3019.
- Orlanski, I. 1976, A simple boundary condition for unbounded hyperbolic flows. *J. Comput. Phys.*, vol. 21 pp. 251–69.
- Peskin, C.S. 1972, Flow patterns around heart valves: a numerical method, *Journal of Computational Physics*, vol. 10, pp. 252–271.
- Peskin, C.S. 2002, The immersed boundary method, *Acta Numerica*, pp. 479-517.
- Quarteroni, A., Tuveri, M., Veneziani, A. 2000, Computational vascular fluid dynamics: problems, models and methods, in *Computing and Visualization in Science*, vol. 2, no. 4, pp. 163-197.
- Roshko, A. 1954, On the drag and shedding frequency of bluff cylinders, NACA TN, 3169.
- Ryzhakov, P., Rossi, R., Idelsohn, S.R., Oñate, E. 2010, a monolithic Lagrangian approach for fluid-structure interaction problems, *J. Computational Mechanics*, vol. 46 no. 6, pp. 883-899.
- Ryzhakov, P., Oñate, E., Rossi, R., Idelsohn, S.R. 2012, Improving mass conservation in simulation of incompressible flows, *International Journal for Numerical Methods in Engineering*, vol. 90 no.12, pp. 1435-548.
- Ryzhakov, P., Rossi, R., Vina, A., Oñate, E. 2013, Modelling and simulation of the sea-landing of aerial vehicles s using the particle finite element method, *J. of Ocean engineering* vol. 66 pp. 92-100.
- Saiki, EM, Biringeren, S. 1996, Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method. *Journal of Computational Physics*, vol. 123, pp. 450–65.
- Sani, R.L., Shen, J., Pironneau, O., Gresho, P.M. 2006, Pressure boundary condition for the time-dependent incompressible Navier-Stokes equations, *Internat. J. Numer. Methods Fluids*, vol. 50, pp. 673–682.
- Scardovelli, R., Zaleski, S. 1999, Direct numerical simulation of free surface and interfacial flow, *Ann. Rev. Fluid Mech.*, vol. 31, pp. 567.
- Schulz, K.W. and Kallinderis, Y. 1998, Unsteady flow structure interaction for incompressible flows using deformable hybrid grids, *Journal of Computational Physics*, 143(2), pp. 569-597.
- Sumer, B.M., Fredsøe, J. 2006, *Hydrodynamics around cylindrical Structures*, Revised Edition, Advanced series on ocean engineering- vol. 26, World Scientific publishing Co.Pte.Ltd.
- Takizawa, K., Wright, S., Moorman, C. and Tezduyar, T.E. 2011, Fluid–structure interaction modeling of parachute clusters, *International Journal for Numerical Methods in Fluids*, 65(1-3), pp. 286-307.
- Tezduyar, T.E., Sathe, S., Keedy, R., Stein, K. 2004, Space-Time Techniques for Finite Element Computation of Flows with Moving Boundaries and Interfaces, *Proceedings of the III International Congress on Numerical Methods in Engineering and Applied Sciences*, Monterrey, Mexico.
- Tezduyar, T.E., Sathe, S., Keedy, R., Stein, K. 2006, Space-Time Finite Element Techniques for Computation of Fluid-Structure Interactions, *Computer Methods in Applied Mechanics and Engineering*, vol. 195, pp. 2002-2027, doi: 10.1016/j.cma.2004.09.014.

- Tezduyar, T.E., Sathe, S. 2007, Modeling of Fluid-Structure Interactions with the Space-Time Finite Elements: Solution Techniques, *International Journal for Numerical Methods in Fluids*, vol. 54, pp. 855-900, doi: 10.1002/flid.1430.
- Tezduyar, T.E. 1992a, Stabilized Finite Element Formulations for Incompressible Flow Computations, *Advances in Applied Mechanics*, vol. 28, pp. 1-44, doi: 10.1016/S0065-2156(08)70153-4.
- Tezduyar, T.E., Behr, M., Liou, J. 1992b, A New Strategy for Finite Element Computations Involving Moving Boundaries and Interfaces, *The Deforming-Spatial-Domain/Space-Time Procedure: I. The Concept and the Preliminary Numerical Tests*, *Computer Methods in Applied Mechanics and Engineering*, vol. 94, pp. 339-351, doi: 10.1016/0045-7825(92)90059-S.
- Tezduyar T.E., Behr M., Mittal, S., Liou, J. 1992c, A New Strategy for Finite Element Computations Involving Moving Boundaries and Interfaces -- The Deforming-Spatial-Domain/Space-Time Procedure: II. Computation of Free-surface Flows, Two-liquid Flows, and Flows with Drifting Cylinders, *Computer Methods in Applied Mechanics and Engineering*, vol. 94, pp. 353-371, doi: 10.1016/0045-7825(92)90060-W.
- Tezduyar, T.E., 2001, Finite element methods for flow problems with moving boundaries and interfaces, *Archives of Computational Methods in Engineering*, 8(2), pp. 83-130.
- Tezduyar, T.E. 2003, Computation of Moving Boundaries and Interfaces and Stabilization Parameters, *International Journal for Numerical Methods in Fluids*, vol. 43, pp. 555-575, doi: 10.1002/flid.505.
- Tezduyar, T.E., Sathe, S., Schwaab, M., Conklin, B.S. 2008, Arterial Fluid Mechanics Modelling with the Stabilized Space-Time Fluid-Structure Interaction Technique, *International Journal for Numerical Methods in Fluids*, vol. 57, pp. 601-629.
- Torii, R., Oshima, M., Kobayashi, T., Takagi, K., Tezduyar, T.E. 2007, Numerical investigation of the effect of hypertensive blood pressure on cerebral aneurysm—dependence of the effect on the aneurysm shape. *International Journal for Numerical Methods in Fluids*, vol. 54, pp.995–1009.
- Tu, C., Peskin, C.S. 1992, Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods. *SIAM Journal on Scientific and Statistical Computing* vol. 13, pp. 1361-1376.
- Udaykumar, H.S., Kan, H.C., Shyy, W., Tran-Son-Tay R. 1997, Multiphase dynamics in arbitrary geometries on fixed Cartesian grids, *J. Computational Physics*, vol. 137, pp. 366.
- Udaykumar, H.S., Mittal, R., Shyy, W. 1999, Computation of solid-liquid phase fronts in the sharp interface limit on fixed grids, *J. Comput. Phys.*, vol. 153, pp. 534–74.
- Udaykumar, H.S., Mittal, R., Rampunggoon, P., Khanna, A. 2001, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, *J. Comput. Phys.*, vol. 174 pp. 345–80.
- Verzicco, R.m, Mohd-Yusof, J., Orlandi, P., Haworth, D. 2000, Large eddy simulation in complex geometric configurations using boundary body forces, *AIAA J.* vol. 38 no. 3, pp.427.
- Vierendeels, J., Dumont, K., Verdonck, P.R. 2008, A partitioned strongly coupled fluid-structure interaction methods to model heart valve dynamics *journal of computational and applied mathematics*, Vol. 215, pp. 602,609.
- Wall, W.A. 1999, Fluid–structure interaction with stabilized finite elements. Ph.D. Thesis, University of Stuttgart.
- Wall, W.A., Gerstenberger, A., Gammitzer, P., Forster, C., Ramm, E. 2006, Large deformation fluid–structure interaction—advances in ALE methods and new fixed grid

- approaches. In *Fluid–Structure Interaction, Lecture Notes in Computational Science and Engineering*. Springer: Berlin, pp. 195–232.
- Wall, W.A., Genkinger, S., Ramm, E. 2007, A strong coupling partitioned approach for fluid–structure interaction with surfaces, *Computers and Fluids*, vol. 36, pp. 169–183.
- Wang, X.S. 2006, from immersed boundary method to immersed continuum method, *international journal for multi-scale Computational engineering*, vol 4. no.1, pp. 127-145.
- Wang, X.S. 2007, An interactive matrix-free method in implicit immersed boundary/continuum method, *computers and structures*, vol. 85, pp. 739-748.
- Wang, X.S. (eds.) 2010, Immersed boundary/continuum methods, in *computational modelling in Biomechanics*, De S. et al., Springer, pp.3-48.
- Williamson, C.H. 1988, Defining a universal and continuous Strouhal–Reynolds number relationship for the laminar vortex shedding of a circular cylinder, *Physics of Fluids (1958-1988)*, 31(10), pp. 2742-2744.
- Williamson C.H.K. 1996a, Vortex dynamics in the cylinder wake, *annual review of fluid mechanics*, annu. Rev. Fluid Mechanics, vol. 28, pp. 477-526.
- Williamson, C.H.K. 1996, Three-dimensional vortex dynamic in wakes, invited Review in special edition on Jets, wakes and shear layers of experimental, *Thermal and Fluid science*, vol. 12, pp. 150-168.
- Williamson, C.H.K., Govadhan, R. 2004, Vortex-induced vibration, *Annu. Rev. Fluid Mech.* vol. 36, pp. 413-55.
- Williamson, C.H.K., Govardhan, R. 2008, A brief review of recent results in vortex-induced vibrations, *journal of wind Engineering and industrial aerodynamics* vol. 96, pp. 713-735.
- Williamson, C.H.K., Roshko, A. 1988, Vortex formation in the wake of an oscillating cylinder. *J. Fluids Structure*, vol. 2, pp. 355-381.
- Wood, C., Gil, A.J., Hassan, O., Bonet, J. 2010, Partitioned block-gauss-seidel coupling for dynamic fluid-structure interaction, *Computers and Structures*, vol. 88, pp. 1367-1382.
- Wüncher, R. 2006, *Mechanics and Numerics of form finding and fluid-structure interaction of Membrane structures*. Doctoral Thesis, Technical University of Munich.
- Xu, S., Wang, Z.J. 2006, An immersed interface method for simulating the interaction of a fluid with moving boundaries, *Journal of Computational Physics*, vol. 216, pp. 454-493.
- Yang, J., Balaras, E. 2006, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, *Journal of computational physics*, vol. 215, pp. 12-40.
- Yang, J., Preidikman, S. & Balaras, E. 2008, A strongly coupled, embedded-boundary method for fluid–structure interactions of elastically mounted rigid bodies, *Journal of Fluids and Structures*, vol. 24, no. 2, pp. 167-182.
- Ye, T., Mittal, R., Udaykumar H.S., Shyy, W. 1999, An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries, *Journal of Computational Physics*, vol. 156, pp. 209–240,
- Yu, Z. 2005, A DLM/FD method for fluid/flexible-body interactions, *Journal of computational physics*, 207(1), pp. 1-27.
- Zhang, W., Jiang Y., Ye, Z. 2007, two better loosely coupled solution algorithms of CFD based aeroelastic simulation, *engineering applications of computational fluid mechanics*, vol. 1, no. 4, pp. 253-262.

Appendix A, Fortran Code, FSI by Reconstruction method

```
c
PROGRAM DISCO
c
implicit none
c
integer nnx, nny, MxSurf, Mxy      ! grid dimension in x and y direction
common /cylsize/ acyl, bcyl, Rcyl
double precision acyl, bcyl, Rcyl ! cylinder center point x and y
direction plus its radius
parameter (nnx=600, nny=850, MxSurf=50)
parameter (Mxy=2*nnx+2*nny)
cc  parameter (acyl=5.00000, bcyl=10.00000, Rcyl=0.5000000000)
c
common /veloxx/ u(0:nnx,0:nny+1), v(0:nnx+1,0:nny),
&              p(0:nnx+1,0:nny+1), a(0:nnx, nny, 4), b(nnx, 0:nny, 4)
double precision u, v, p, a, b
c
common / dimenx/ nx, ny, Re, RRe, dt, time, dts, nx2, ny2, nn
integer nx, ny, nx2, ny2, nn
double precision Re, RRe, dt, time, dts
double precision maxdiv, divm
c
common / bndvinfi/ ip1v(Mxy), jplv(Mxy), ip2v(Mxy), jp2v(Mxy),
&              iinterp(Mxy), jinterp(Mxy), nbndv
integer ip1v, jplv, ip2v, jp2v, iinterp, jinterp, nbndv
c
common / bnduinfi/ ip1u(Mxy), jplu(Mxy), ip2u(Mxy), jp2u(Mxy),
&              iinterpu(Mxy), jinterpu(Mxy), nbndu
integer ip1u, jplu, ip2u, jp2u, iinterpu, jinterpu, nbndu
c
common / bndvinfR/ wv1(Mxy), wv2(Mxy)
double precision wv1, wv2
c
common / bnduinfR/ wu1(Mxy), wu2(Mxy)
double precision wu1, wu2
c
integer nt, i, j, k, t, ksub
logical EX
c
inquire(file='movie.dat', exist=EX)
if (EX) go to 15
c
open(12, file = 'movie.dat', position='append',
&      form='formatted')
write(12, '(A)') 'variables="x","y","u","v","p"'
close(12)
15 continue
c
c
c
time=0.D0
call inigrid()
call init()
call interpolate()
call bounds()
call inisol()
c
do nt = 1, 1000000
time=time+dt
write(*,*) 'time = ', time
ksub=0
call structuremain
call structure(ksub)
c
c
cc  call convec()
```

```

cc      call fillf()
cc      call calcuv()
c      ***** FSI Part *****
c      if ((mod(nt,100) .EQ. 1) .AND.
c &      ( nt .NE. 1)) then
c      call forc vib
c      call structure()
c      call structuretwo()
c      end if
c      call convergence(ksub)
c      ***** end of FSI *****
c      STOP
c      if (mod(nt,1000) .EQ. 0) then
c        write(*,*) 'Saving field.dat...'
c        call wrtflld()
c        call savfld()
c      end if
c      if (mod(nt,10).EQ.0) then
c        call mean()
c      end if
c      call bounds() ! this line is added in test7
c      divm=maxdiv()
c      write(*,*) 'Maximum Divergence = ',divm
c    end do

c
c    call wrtflld()
c    call savfld()
c    call etimetest
c  end

cc
c

      subroutine inigrid()
c
c ** initialize the grid, blocking, extrapolation of velocities at
c ** boundaries
c
c    integer nnx, nny, MxSurf, Mxy
c    parameter (nnx=600, nny=850, MxSurf=50)
c    parameter (Mxy=2*nnx+2*nny)
cc    parameter (acyl=5.000000, bcyl=10.000000, Rcyl=0.500000)
c
c    common /veloxx/ u(0:nnx,0:nny+1), v(0:nnx+1,0:nny),
c &    p(0:nnx+1,0:nny+1), a(0:nnx, nny, 4), b(nnx, 0:nny, 4)
c    double precision u, v, p, a, b
c
c    common / griddd/ xcrd(0:nnx+1), ycrd(0:nny+1), scalar(nnx, nny),
c &    xfree(Mxy, MxSurf), yfree(Mxy, MxSurf)
c    double precision xcrd, ycrd, scalar, xfree, yfree
c
c    common / griddx/ xcoord(0:nnx), ycoord(0:nny)
c    double precision xcoord, ycoord
c
c    common /maskdiv/ idiv(Mxy), jdiv(Mxy), ndiv
c    integer idiv, jdiv, ndiv
c
c    common /minsx/ pins(0:nnx+2,0:nny+2), uins(0:nnx+2,0:nny+2),
c &    vins(0:nnx+2,0:nny+2)
c    double precision pins, uins, vins
c
c    common /bniinf/ jyu(nnx), jyv(nnx)
c    integer jyu, jyv
c
c    common /dimenx/ nx, ny, Re, RRe, dt, time, dts, nx2, ny2, nn
c    integer nx, ny, nx2, ny2, nn, nx3, nn1
c    double precision Re, RRe, dt, time, dts
c    integer t, k, nxl, nyl
c

```

```

double precision yj,y0,y1,delta,tanh0,tanh1,coef1
c
common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
c
double precision dx,dy,fact,offset,alpha,xtg,ytg,dnorm,dyl
double precision gradient,intercept, xint,yint,weight,a1,c1,PI
logical EX
c
common /homeadd/ home
character*40 home
c
RRe=1D0/Re
c   write(*,*)'ingrid,home',home
c   STOP
c
c   *****Reading an arbitrary grid from a file if exist otherwise
c   making a uniform grid *****
c
cc  inquire(file='grid.bin',exist=EX)
c
cc  if (EX) then
cc    open(unit=12,file='grid.bin',form='UNFORMATTED')
cc    rewind(12)
c
cc    read(12) nx,ny
cc    read(12) ( xcrd(i),i=0,nx+1)
cc    read(12) ( ycrd(j),j=0,ny+1)
c    read(12) ( txu(i),i=1,nx )
c    read(12) ( txv(i),i=1,nx )
c    read(12) ( tyu(i),i=1,nx )
c    read(12) ( tyv(i),i=1,nx )
c    read(12) ( vnoru(i),i=1,nx )
c    read(12) ( vnorv(i),i=1,nx )
c    read(12) ( fyu(i),i=1,nx )
c    read(12) ( fyv(i),i=1,nx )
c    read(12) ( jyu(i),i=1,nx )
c    read(12) ( jyv(i),i=1,nx )
cc    close(12)
c
cc    do i=0,nx
cc      xcoord(i)=0.5D0*(xcrd(i)+xcrd(i+1))
cc    end do
c
cc    do j=0,ny
cc      ycoord(j)=0.5D0*(ycrd(j)+ycrd(j+1))
cc    end do
c
cc    do i=1,nx
cc      do j=1,ny
cc        pmask(i,j)=1.D0
cc        umask(i,j)=1.D0
cc        vmask(i,j)=1.D0
cc      end do
cc    end do
c
cc    do i=1,nx
cc      do j=1,jyv(i)
cc        pmask(i,j)=0.D0
cc      end do
cc    end do
cc    write(*,*) (i,jyv(i),pmask(i,jyv(i))),i=1,nx)
c
c    do i=1,nx
c      do j=1,ny
c        if (pmask(i,j) .EQ. 0.D0) then
cc          amask(i,j)=0.D0
c          if (i .GT. 1) amask(i-1,j)=0.D0

```

```

c          end if
c          end do
c          end do
c
cc         return
cc         end if
c
c ** Grid does not exist, produce a mesh which is uniform in x and y
c ** direction, later we need to increase the density of mesh near
c ** the cylidner to capture the vorticity
c
c*****          ! This part is for making fine mesh around cylinder x
direction
cc         i=0
cc         dx=0.1D0
cc         xcrd(0)=-0.5D0*dx
cc         xcrd(1)=0.5D0*dx
cc         xcoord(0)=0.D0
cc100      i=i+1
cc         xcoord(i)=xcoord(i-1)+dx
c         xcrd(i+1)=xcrd(i)+dx
cc         xcrd(i)=(xcoord(i)+xcoord(i-1))/2
cc         If (xcoord(i) .LT. 6.0) then
cc             dt=0.1D0
cc         else If ((xcoord(i) .LE.9.05) .AND. (xcoord(i) .GE. 6.0)) then
cc             dx=dx-0.00125D0
c         dx=dx-0.00126
cc         else if ((xcoord(i) .GT. 11) .AND. (xcoord(i) .LT. 13.9)) then
cc             dx=dx+0.00125D0
c         dx=dx+0.00126
cc         else if ((xcoord(i) .GT. 9.05) .AND. (xcoord(i) .LE. 11)) then
cc             dx=0.05D0
c         dx=0.025D0
c         dx=0.0125
c         dx=0.00675
cc         else
cc             dx=0.1D0
cc         end if
c         write(*,*)'i,dx,xcoord(i)', i,dx, xcoord(i)
c         write(*,*)'i,xcoord,xcrd',i,xcoord(i),xcrd(i)
cc         if (xcoord(i) .LE. 25) go to 100
cc         xcrd(i+1)=xcoord(i)+0.5D0*dx
cc         nx=i
c *****          ! end of making fine mesh around cylinder x direction
          delta=5.D0 !3.D0
c
          nx2 =37 !78
          nx3 =37 !66
          dx  =0.0250D0!0.025D0
          nn  =4.0D0/dx
          nx=nx2+nx3+nn
          xcoord(0) = -15.0D0
          xcoord(nx2)= -2.0D0
c
          do i=1,nx2-1
          yj=1.D0*i
          y0=delta/2.D0*yj/nx2
          y1=delta/2.D0
          tanh0 = (exp(y0)-exp(-y0))/(exp(y0)+exp(-y0))
          tanh1 = (exp(y1)-exp(-y1))/(exp(y1)+exp(-y1))
          coef1 = tanh0/tanh1
          xcoord(i) = (1.D0-coef1)*xcoord(0)+coef1*xcoord(nx2)
          xcrd(i)=(xcoord(i)+xcoord(i-1))/2.0D0
          end do
c
          write(*,*)'nx2, dx',nx2,xcoord(nx2)-xcoord(nx2-1)
c
          do k=1,nn-1

```



```

        xcoord(nx2+k)=xcoord(nx2)+k*dx
        xcrd(nx2+k)  =(xcoord(nx2+k)+xcoord(nx2+k-1))/2.0D0
    end do
c
        xcoord(nx2+nn)  = 2.0D0
        xcoord(nx)      =15.0D0
c

do i=1,nx3-1
    yj=1.D0*(nx3-i)
    y0=delta/2.D0*yj/nx3
    y1=delta/2.D0
    tanh0 = (exp(y0)-exp(-y0))/(exp(y0)+exp(-y0))
    tanh1 = (exp(y1)-exp(-y1))/(exp(y1)+exp(-y1))
    coef1 = tanh0/tanh1
    xcoord(nx2+nn+i) = coef1*xcoord(nx2+nn)+
&      (1.0D0-coef1)*xcoord(nx
xcrd(nx2+nn+i)=(xcoord(nx2+nn+i)+xcoord(nx2+nn+i-1))/2
end do
    write(*,*)'nx2+nn,dx',nx2+nn,xcoord(nx2+nn+1)-xcoord(nx2+nn)
c

    xcrd(nx)  =(xcoord(nx)+xcoord(nx-1))/2.0D0
    xcrd(nx+1)=2*xcoord(nx)-xcrd(nx)
    xcrd(0)= xcoord(0)-(xcrd(1)-xcoord(0))
    xcrd(nx2)=(xcoord(nx2)+xcoord(nx2-1))/2.0D0
    xcrd(nx2+nn)=(xcoord(nx2+nn)+xcoord(nx2+nn-1))/2.0D0

c
    k=0
c
    nx=2*nx2+nn
c
    do i=nx2+nn+1,nx
c
        k =k+1
c
        xcoord(i)= -xcoord(nx2-k)
c
        xcrd(i)=(xcoord(i)+xcoord(i-1))/2
c
    end do
c
    xcrd(nx+1)=xcoord(nx)+(xcoord(nx)-xcrd(nx-1))
c
    xcrd(0)= xcoord(0)-(xcrd(1)-xcoord(0))
c
    xcrd(nx2)=(xcoord(nx2)+xcoord(nx2-1))/2
c
    do i=0,nx+1
        write(*,*)'i,dx,xcoord,xcrd',
&      i,2*(xcoord(i)-xcrd(i)), xcoord(i),xcrd(i)
    end do

c
cc
    open(unit=12,file='trial.out')
cc
    write(12,*)'variables="x","y"'
cc
    write(12,*)
cc
&    'ZONE T="scalar field",I=',2*ny2+nn,'J =',2*ny2+nn,'F=BLOCK'
c
cc
    write(12,'(5E16.8)')((y(i),i=1,2*ny2+nn),j=1,2*ny2+nn)
cc
    write(12,'(5E16.8)')((y(j),i=1,2*ny2+nn),j=1,2*ny2+nn)
c
cc
    close(12)

cccc
c
! making uniform mesh xdirection
cc
nx=250
cc
ny=300 ! was 400 for 20D in y direction
cc
dx = 25D0/nx
cc
xcrd(0)=-0.5D0*dx
cc
do i=0,nx
! this part has change to make a bit new andices
cc
    xcoord(i)=i*dx
cc
    xcrd(i+1)=xcrd(i)+dx
cc
end do

c ***** ! end of uniform mesh x direction

c
alpha = 1.02 ! stretching of 2 per cent.
c
dy = 1D0/ny ! this is the mean dy

```

```

c
c   ycoord(ny)=1D0
c   do j=ny-1,0,-1
c       ycoord(j)=ycoord(j+1)-dy
c       dy=alpha*dy
c   end do
c   offset=ycoord(0)
c   fact=ycoord(ny)-offset
c
c   do j=0,ny
c       ycoord(j)=(ycoord(j)-offset)/fact
c   end do
c
c   ycrd(0)=-0.5D0*ycoord(1)
c
c ***** ! creating fine mesh around the cylinder in y direction
c
cc       j=0
cc       dy=0.1D0
cc       ycrd(0)=-0.5D0*dy
cc       ycrd(1)=0.5D0*dy
cc       ycoord(0)=0.D0
cc
cc110    j=j+1
cc       ycoord(j)=ycoord(j-1)+dy
c       ycrd(j+1)=ycrd(j)+dy
cc       ycrd(j)=(ycoord(j)+ycoord(j-1))/2
cc       if ((ycoord(j) .LE. 11.0) .OR. (ycoord(j) .GE.18.9)) then
cc           dy=0.1D0
cc       else if ((ycoord(j) .GT. 11.0) .AND. (ycoord(j) .LE. 14.05)) then
cc           dy=dy-0.00125D0
c       dy=dy-0.001262
cc       else if ((ycoord(j) .GE. 16) .AND. (ycoord(j) .LT. 18.9)) then
cc           dy=dy+0.00125D0
c       dy=dy+0.001262
cc       else
cc           dy=0.05D0
c       dy=0.025D0
c       dy=0.0125
c       dy=0.00535
cc       end if
c       write(*,*)'j,dy,ycoord(j)', j,dy, ycoord(j)
cc       if (ycoord(j) .LE. 30) go to 110
cc       ycrd(j+1)=ycrd(j)+0.5D0*dy
cc       ny=j
cc       write(*,*)'ny',ny
c ***** end of creating fine mesh around the cylidner in y direction
c
c
c   ny1=119
c   ny2=37
c   delta=5.D0 !3.D0
c   nn = 160!160
c
c   ny2 = (ny1+1)/2
c   ycoord(0) = -15.0D0
c   ycoord(ny2)= -2.0D0
c
c   do j=1,ny2-1
c       yj=1.D0*j
c       y0=delta/2.D0*yj/ny2
c       y1=delta/2.D0
c       tanh0 = (exp(y0)-exp(-y0))/(exp(y0)+exp(-y0))
c       tanh1 = (exp(y1)-exp(-y1))/(exp(y1)+exp(-y1))
c       coef1 = tanh0/tanh1
c       ycoord(j) = (1.D0-coef1)*ycoord(0)+coef1*ycoord(ny2)
c       ycrd(j)=(ycoord(j)+ycoord(j-1))/2
c   write(*,*)'j,dy,ycoord(j)',j,dy, ycoord(j)

```

```

end do
c
nn1    = INT(4/ABS(ycoord(ny2)-ycoord(ny2-1)))
dy1    = 4.0D0 /nn1
c
dy     = 4.0D0/nn
write(*,*) 'dy,nn =', dy,nn,dy1,nn1
c
do k=1,nn-1
    ycoord(ny2+k)=ycoord(ny2)+k*dy
    ycrd(ny2+k)=(ycoord(ny2+k)+ycoord(ny2+k-1))/2
c
    write(*,*) 'j,dy,ycoord(j)',ny2+k,dy, ycoord(ny2+k)
end do
c
    k=0
    ny=2*ny2+nn
do j=ny2+nn,ny
c
    k =k+1
    ycoord(j)= -ycoord(ny2-k)
    ycrd(j)=(ycoord(j)+ycoord(j-1))/2
    k =k+1
c
    write(*,*) 'j,dy,ycoord(j)', j,dy, ycoord(j)
end do
    ycrd(ny+1)=ycoord(ny)+(ycoord(ny)-ycrd(ny))
    ycrd(0)= ycoord(0)-(ycrd(1)-ycoord(0))
    ycrd(ny2)=(ycoord(ny2)+ycoord(ny2-1))/2
do j=0,ny+1
    write(*,*) 'j,dy,ycoord(j)',j,ycoord(j)-ycoord(j-1),
& ycoord(j),ycrd(j)
end do

c
c ***** creating uniform mesh
c
cc dy= 30D0/ny !was 20 in y direction
cc ycrd(0)=-0.5D0*dy
cc ycoord(0)=-dy
cc do j=0,ny ! this part has change to make a bit new
andices
cc ycoord(j)=j*dy
cc ycrd(j+1)=ycrd(j)+dy
cc end do
c
c ycrd(ny+1)=1D0+0.5D0*(ycoord(ny)-ycoord(ny-1))
c
open(unit=12,file='grid.dat')

write(12,*) 'variables="x","y"'
write(12,*)
& 'ZONE T="scalar field",I = ',nx,' J = ',ny,' F=BLOCK'
write(12,'(5E16.8)') ((xcoord(i),i=1,nx),j=1,ny)
write(12,'(5E16.8)') ((ycoord(j),i=1,nx),j=1,ny)
close(12)
c
STOP
open(unit=12,file='grid2.dat')
write(12,*) 'variables="x","y"'
write(12,*)
& 'ZONE T="scalar field",I = ',nx,' J = ',ny,' F=BLOCK'
write(12,'(5E16.8)') ((xcrd(i),i=1,nx),j=1,ny)
write(12,'(5E16.8)') ((ycrd(j),i=1,nx),j=1,ny)
close(12)
c
STOP
c *****define u and v and p absolute inside of the
cylinder*****
cc uins=1
cc vins=1
cc pins=1
cc

```

```

cc      do i=2,nx-1
cc      do j=2,ny-1
cc      if (sqrt((xcrd(i)-acyl)**2+(ycoord(j)-bcyl)**2) .LT. Rcyl) then
cc          vins(i,j)=vsolid
cc      end if
cc      if (sqrt((xcoord(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) then
cc          uins(i,j)=usolid
cc      end if
cc      if (sqrt((xcrd(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) then
cc          pins(i,j)=0
cc      end if
cc      end do
cc      end do
c *****end of part *****
c
c      do i=1,nx
c          xtg = xcrd(i)
c          ytg = 0.269D0*sqrt(0.1D0 * xtg)
c          fxdm = MAX(1.D0-2.D0*MAX(xtg-4.D0,0.D0),0.D0)
c          txv(i)=1D0
c          tyv(i)=0.05D0*0.269D0/sqrt(0.1D0 * xtg)
c          dnorm = SQRT(txv(i)**2+tyv(i)**2)
c          txv(i)=txv(i)/dnorm
c          tyv(i)=tyv(i)/dnorm
c          vnorv(i)=fxdm*0.081/sqrt(0.1D0 * xtg) ! scaled by inlet vel.
c          vnorv(i)=0.D0
c          do j=1,ny
c              jyv(i)=j
c              pmask(i,j)=0.D0
c              if (ycoord(j)-ytg .GT. 0D0) GoTo 20
c          end do
c          STOP 'ERROR 002'
c 20      continue
c          jyu(i)=jyv(i)+1
c
c          fyv(i)=ytg
c      end do
c
c      open(unit=12,file='checkv.dat')
c      write(12,*) '*** v: i, j, ycoord, ytg: '
c      do i=1,400
c          write(12,*) i,jyv(i),ycoord(jyv(i)),fyv(i)
c      end do
c      close(12)
c
c      STOP 'check it'
c      do i=1,nx
c      do j=1,ny
c          if (pmask(i,j) .EQ. 0.D0) then
c              amask(i,j)=0.D0
c              if (i .GT. 1) amask(i-1,j)=0.D0
c          end if
c      end do
c      end do
c
c      do i=1,nx
c      do j=2,ny
c          if ((amask(i,j) .EQ. 1.D0) .AND. (amask(i,j-1) .EQ. 0.D0)) then
c              jyu(i)=j
c          end if
c      end do
c      end do
c
cc      open(unit=12,file='grid.bin',form='UNFORMATTED')
cc      rewind(12)
c
cc      write(12) nx,ny
cc      write(12) ( xcrd(i),i=0,nx+1)

```

```

cc      write(12) ( ycrd(j),j=0,ny+1)
cc      write(12) ( txu(i),i=1,nx )
cc      write(12) ( txv(i),i=1,nx )
cc      write(12) ( tyu(i),i=1,nx )
cc      write(12) ( tyv(i),i=1,nx )
cc      write(12) (vnoru(i),i=1,nx )
cc      write(12) (vnorv(i),i=1,nx )
cc      write(12) ( fyu(i),i=1,nx )
cc      write(12) ( fyv(i),i=1,nx )
cc      write(12) ( jyu(i),i=1,nx )
cc      write(12) ( jyv(i),i=1,nx )
cc      close(12)
c
      return
      end
cc

      subroutine interpolate()
c
c ** initialize the grid, blocking, extrapolation of velocities at
c ** boundaries
c
      integer nnx, nny, MxSurf, Mxy
      parameter (nnx=600, nny=850, MxSurf=50)
      parameter (Mxy=2*nnx+2*nny)
c      parameter (acyl=5.000000, bcyl=10.000000, Rcyl=0.500000)
c
      common /cylzise/ acyl, bcyl, Rcyl
      double precision acyl, bcyl, Rcyl
      common /veloxx/ u(0:nnx,0:nny+1), v(0:nnx+1,0:nny),
&                  p(0:nnx+1,0:nny+1), a(0:nnx, nny, 4), b(nnx, 0:nny, 4)
      double precision u, v, p, a, b
c
      common / griddd/ xcrd(0:nnx+1), ycrd(0:nny+1), scalar(nnx, nny),
&                  xfree(Mxy, MxSurf), yfree(Mxy, MxSurf)
      double precision xcrd, ycrd, scalar, xfree, yfree
c
      common / griddx/ xcoord(0:nnx), ycoord(0:nny)
      double precision xcoord, ycoord
c
      common / bndvinfi/ ip1v(Mxy), jp1v(Mxy), ip2v(Mxy), jp2v(Mxy),
&                  iinterpv(Mxy), jinterpv(Mxy), nbndv
      integer ip1v, jp1v, ip2v, jp2v, iinterpv, jinterpv, nbndv
c
      common / bnduinfi/ ip1u(Mxy), jp1u(Mxy), ip2u(Mxy), jp2u(Mxy),
&                  iinterpu(Mxy), jinterpu(Mxy), nbndu
      integer ip1u, jp1u, ip2u, jp2u, iinterpu, jinterpu, nbndu

      common / bndvinfR/ wv1(Mxy), wv2(mxy)
      double precision wv1, wv2
c
      common / bnduinfR/ wu1(Mxy), wu2(Mxy)
      double precision wu1, wu2
c
      common / bndpinfi/ ip1p(0:Mxy), jp1p(0:Mxy), ip2p(0:Mxy), jp2p(0:Mxy),
&                  ip3p(0:Mxy), jp3p(0:Mxy), iinterpp(0:Mxy), jinterpp(0:Mxy),
&                  nbndp
      integer ip1p, jp1p, ip2p, jp2p, ip3p, jp3p, iinterpp, jinterpp, nbndp
c
      common / bndpinfR/ teta(0:Mxy), unitvi(0:Mxy), unitvj(0:Mxy),
&                  wp1(0:Mxy), wp2(0:Mxy), delta1(0:Mxy)
      double precision teta, unitvi, unitvj, wp1, wp2, delta1
c
      common /maskdiv/ idiv(Mxy), jdiv(Mxy), ndiv
      integer idiv, jdiv, ndiv

      common /masksx/ pmask(0:nnx,0:nny), umask(0:nnx,0:nny),

```

```

&                                vmask(0:nnx,0:nny)
double precision pmask,umask,vmask

C
common / bndinf/ txu(nnx),txv(nnx),tyu(nnx),tyv(nnx),
&                vnoru(nnx),vnorv(nnx),fyu(nnx),fyv(nnx),
&                vup(0:nnx+1),uup(0:nnx),influx
double precision txu,txv,tyu,tyv,vnoru,vnorv,fyu,fyv,vup,uup,
&                influx

C
common /minsx/ pins(0:nnx+2,0:nny+2),uins(0:nnx+2,0:nny+2),
&                vins(0:nnx+2,0:nny+2)
double precision pins,uins,vins

C
common /bniinf/ jyu(nnx),jyv(nnx)
integer jyu,jyv

C
common /dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,nx2,ny2,nn
double precision Re,RRe,dt,time,dts
integer t,k

C
common /sbody/ vsolid,ysolid,usolid,xsolid,aysolid,axsolid
double precision vsolid,ysolid,usolid,xsolid,aysolid,axsolid

C
double precision dx,dy,fact,offset,alpha,xtg,ytg,dnorm
double precision gradient,intercept,xint,yint,weight,a1,c1,PI
logical EX

C
common /homeadd/ home
character*40 home

C

xsolid=0.D0
RRe=1.D0/Re
bcyl=0.D0!+ysolid
acyl=0.D0!+xsolid
Rcyl=0.5D0

CC
C *****define u and v and p absolute inside of the
cylinder*****
CC    uins=1
CC    vins=1
CC    pins=1
CC
CC    do i=2,nx-1
CC    do j=2,ny-1
CC        if (sqrt((xcrd(i)-acyl)**2+(ycoord(j)-bcyl)**2) .LT. Rcyl) then
CC            vins(i,j)=vsolid
CC        end if
CC        if (sqrt((xcoord(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) then
CC            uins(i,j)=usolid
CC        end if
CC        if (sqrt((xcrd(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) then
CC            pins(i,j)=0
CC        end if
CC    end do
CC    end do
C *****end of part *****
    umask=0.D0
    vmask=0.D0
    do i=1,nx-1
    do j=1,ny
        umask(i,j)=1.D0
    end do
    end do

C
do i=1,nx

```

```

do j=1,ny-1
  vmask(i,j)=1.D0
end do
end do
c
c *** to define where vmask(i,j)=0 and umask(i,j)=0 and pmask(i,j)=0
do i=2,nx-1
  do j=2,ny-1
    if
& ((sqrt((xcrd(i+1)-acyl)**2+(ycoord(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcrd(i-1)-acyl)**2+(ycoord(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcrd(i)-acyl)**2+(ycoord(j+1)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcrd(i)-acyl)**2+(ycoord(j-1)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i)-acyl)**2+(ycrd(j+1)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i-1)-acyl)**2+(ycrd(j+1)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i-1)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcrd(i)-acyl)**2+(ycoord(j)-bcyl)**2) .LE. Rcyl))
& then
  vmask(i,j)=0
  end if
  if
& ((sqrt((xcoord(i+1)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i-1)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i)-acyl)**2+(ycrd(j+1)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i)-acyl)**2+(ycrd(j-1)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcrd(i+1)-acyl)**2+(ycoord(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcrd(i)-acyl)**2+(ycoord(j)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcrd(i)-acyl)**2+(ycoord(j-1)-bcyl)**2) .LT. Rcyl) .OR.
& (sqrt((xcoord(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LE. Rcyl) .OR.
& (sqrt((xcrd(i+1)-acyl)**2+(ycoord(j-1)-bcyl)**2) .LT. Rcyl))
& then
  umask(i,j)=0
  end if
end do
end do
c
c c ***** definition of pmask

do i=1,nx
  do j=1,ny
    if ((umask(i-1,j)+umask(i,j)+vmask(i,j-1)+vmask(i,j)).EQ.1) then
      umask(i ,j )=0
      umask(i-1,j )=0
      vmask(i ,j )=0
      vmask(i ,j-1)=0
    end if
  end do
end do
c
c ***** end of definition of pmask

pmask=0.D0
do i=1,nx
  do j=1,ny
    if ((umask(i-1,j)+umask(i,j)+vmask(i,j-1)+vmask(i,j)).GE.1) then
      pmask(i,j)=1.D0
    end if
  end do
end do

c
c ***** v-velocities *****
c *****interpolation to find the boundary value of vmask(i,j)
k=0
nbndv=0
do j=2,ny-1
  do i=2,nx-1

```

```

C
      deltal=sqrt(((xcrd(i)-acyl)**2)+(ycoord(j)-bcyl)**2)
      If ((vmask(i,j) .EQ. 0.D0) .AND. (deltal .GE. Rcyl)) then
          k=k+1
          gradient=((bcyl-ycoord(j))/((acyl-xcrd(i))))
          intercept=bcyl-acyl*gradient
C      write(*,*)'gradient, intercept', gradient,intercept
C
C *****  third quarter of circle *****
C
          if ((ycoord(j) .LE. bcyl) .AND. (xcrd(i) .LE. acyl))
      &         then
          xint=(ycoord(j-1)-intercept)/gradient
C
          If ((acyl .EQ. xcrd(i)) .OR.
      &         ((xint .GE. xcrd(i-1)) .AND.
      &         (xint .LE. xcrd(i)))) then
          yint=ycoord(j-1)
          weight=((xint-xcrd(i-1))/(xcrd(i)-xcrd(i-1))) ! weight
of grater I indices of v on interpolation point
          If (acyl .EQ. xcrd(i)) then
              weight=1
              xint=xcrd(i)
          End if
          iplv(k)=i
          jplv(k)=j-1
          ip2v(k)=i-1
          jp2v(k)=j-1
C      write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycrd
C      & third1',k,i,j,weight,xcrd(i-1),xint,xcrd(i),gradient,intercept
          Else
              xint=xcrd(i-1)
              yint=gradient*xint+intercept
              weight=((yint-ycoord(j-1))/(ycoord(j)-ycoord(j-1)))
              If (bcyl .EQ. ycoord(j)) then !new 20/5/13
                  weight=1 !new 20/5/13
                  yint=ycoord(j) !new 20/5/13
              End if !new 20/5/13
              iplv(k)=i-1
              jplv(k)=j
              ip2v(k)=i-1
              jp2v(k)=j-1
C      write(*,'(A,3I4,6F8.2)')'i,ik,jk,wet,yco
C      & third2',k,i,j,weight,ycoord(j-1),yint,ycoord(j),intercept,gradient
          End if
      End if
C
C
C      If ((ycoord(j) .GT. bcyl) .AND. (xcrd(i) .LE. acyl))
!*****  second quater of circle
      &         then
          xint=(ycoord(j+1)-intercept)/gradient
C
          If ((xint .GE. xcrd(i-1)) .AND. (xint .LE. xcrd(i))
      &         .OR. (acyl .EQ. xcrd(i))) Then
          yint=ycoord(j+1)
          weight=((xint-xcrd(i-1))/(xcrd(i)-xcrd(i-1))) ! weight of
grater I indices of v on interpolation point
          If (acyl .EQ. xcrd(i)) then
              weight=1
              xint=xcrd(i)
          End if
          iplv(k)=i
          jplv(k)=j+1
          ip2v(k)=i-1
          jp2v(k)=j+1
C      write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycrd
C      &scnd1',k,i,j,weight,xcrd(i-1),xint,xcrd(i),gradient,intercept

```



```

        Else
            xint=xcrd(i-1)
            yint=gradient*xint+intercept
            weight=((yint-ycoord(j))/(ycoord(j+1)-ycoord(j)))
                iplv(k)=i-1
                jplv(k)=j+1
                ip2v(k)=i-1
                jp2v(k)=j
c            write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycoord
c            &scnd2',k,i,j,weight,ycoord(j-1),yint,ycoord(j),gradient,intercept

            End if
            End if

c
c
                If ((ycoord(j) .GT. bcyl) .AND. (xcrd(i) .GT. acyl))
!***** first quater of circle
                &
                    then
                        xint=(ycoord(j+1)-intercept)/gradient
c
                    If ((xint .GE. xcrd(i)) .AND. (xint .LE. xcrd(i+1)))
                &
                    then
                        yint=ycoord(j+1)
                        weight=((xint-xcrd(i))/(xcrd(i+1)-xcrd(i))) ! weight of
grater I indices of v on interpolation point
                        iplv(k)=i+1
                        jplv(k)=j+1
                        ip2v(k)=i
                        jp2v(k)=j+1
c
c            write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycoord
c            & first1',k,i,j,weight,xcrd(i),xint,xcrd(i+1),gradient,intercept

                    Else
                        xint=xcrd(i+1)
                        yint=gradient*xint+intercept
                        weight=((yint-ycoord(j))/(ycoord(j+1)-ycoord(j)))
                            iplv(k)=i+1
                            jplv(k)=j+1
                            ip2v(k)=i+1
                            jp2v(k)=j
c            write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycoord
c            & firt2',k,i,j,weight,ycoord(j),yint,ycoord(j+1),gradient,intercept

                    End if
                    End if

c
c
                If (((ycoord(j) .LE. bcyl) .AND. (xcrd(i) .GT. acyl)))
!***** fourth quater of circle
                &
                    then
                        xint=(ycoord(j-1)-intercept)/gradient
c
                    If ((xint .GE. xcrd(i)) .AND. (xint .LE. xcrd(i+1)))
                &
                    then
                        yint=ycoord(j-1)
                        weight=((xint-xcrd(i))/(xcrd(i+1)-xcrd(i))) ! weight of
grater I indices of v on interpolation point
                        iplv(k)=i+1
                        jplv(k)=j-1
                        ip2v(k)=i
                        jp2v(k)=j-1
c            write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycrd
c            & four1',k,i,j,weight,xcrd(i),xint,xcrd(i+1),gradient,intercept

                    Else
                        xint=xcrd(i+1)

```

```

        yint=gradient*xint+intercept
        weight=((yint-ycoord(j-1))/(ycoord(j)-ycoord(j-1)))
        if (bcyl .EQ. ycoord(j)) then      ! 21/5/13
            weight=1                      ! 21/5/13
            yint= ycoord(j)                ! 21/5/13
        endif                               ! 21/5/13
            ip1v(k)=i+1
            jplv(k)=j
            ip2v(k)=i+1
            jp2v(k)=j-1
c         write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycoord
c     & four2',k,i,j,weight,ycoord(j-1),yint,ycoord(j),gradient,intercept
            End if
        End if
c
        iinterpv(k)=i
        jinterpv(k)=j
        a1=sqrt((acyl-xcrd(i))**2+(bcyl-ycoord(j))**2)-Rcyl ! distance of
boundary point to the boundary of circle
        c1=(sqrt((acyl-xint)**2+(bcyl-yint)**2))-Rcyl      ! distance of
interpolation point to the boundary of circle
cc         wv1(k)=(a1/c1)*weight
cc         wv2(k)=(a1/c1)*(1-weight)
        wv1(k)=weight
        wv2(k)=(a1/c1)
        if (a1 .EQ. 0) then                ! point is on the solid c1 to a1
            wv1(k)=0
            wv2(k)=0
        end if
c         write(*,'(A,3I5,7F10.2)')'i,ik,jk,weight,wv1,wv2,
c     & final',k,i,j,weight,wv1(k),wv2(k),a1,c1,xint,yint
        End if
c
        end do
        end do
        nbndv=k
        write(*,*) 'k,nbndv=',k,nbndv
c ** ***** u-velocities
*****:
c *****interpolation to find the bounadry value of umask(i,j)
        k=0
        nbndu=0
c
        do j=2,ny-1
        do i=2,nx-1
            deltal= sqrt((xcoord(i)-acyl)**2+(ycrd(j)-bcyl)**2)
            If ((umask(i,j) .EQ. 0) .AND.(deltal .GE. Rcyl)) then
                k=k+1
                gradient=(bcyl-ycrd(j))/(acyl-xcoord(i))
                intercept=bcyl-acyl*gradient
c
c
c         If ((ycrd(j) .LE. bcyl) .AND. (xcoord(i) .LE. acyl))      !
***** third quater of circle *****
            &         then
c                 yint=gradient * xcoord(i-1)+intercept
c
c                 If ((bcyl .EQ. ycrd(j)) .OR. ! 21/5/13
&                 ((yint .GE. ycrd(j-1)) .AND.
&                 (yint .LE. ycrd(j)))) then
                    xint=xcoord(i-1)
                    weight=(yint-ycrd(j-1))/(ycrd(j)-ycrd(j-1)) ! weight
of grater I indices of v on interpolation point
                    if (bcyl .EQ. ycrd(j)) then !21/5/13

```

```

                                weight = 1           !21/5/13
                                yint =ycrd(j)        !21/5/13
                                endif                !21/5/13
                                iplu(k)=i-1
                                jplu(k)=j
                                ip2u(k)=i-1
                                jp2u(k)=j-1
c                               write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycrd
c                               & third1',k,i,j,weight,ycrd(j-1),yint,ycrd(j),gradient,intercept

                                Else
                                    yint=ycrd(j-1)
                                    xint=(yint-intercept)/gradient
                                    weight=(xint-xcoord(i-1))/(xcoord(i)-xcoord(i-1))
                                    IF (acyl .EQ. xcoord(i)) then
                                        weight=1
                                        xint=xcoord(i)
                                    End if
                                    iplu(k)=i
                                    jplu(k)=j-1
                                    ip2u(k)=i-1
                                    jp2u(k)=j-1
c                               write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,xcoord
c                               &third2',k,i,j,weight,xcoord(i-1),xint,xcoord(i),gradient,intercept
                                    End if
                                End if

c
                                If ((ycrd(j) .GT. bcyl) .AND. (xcoord(i) .LE. acyl)) !
***** second quarter of circle *****
                                &
                                    then
                                        yint=gradient * xcoord(i-1)+intercept
c
                                    If ((yint .GE. ycrd(j)) .AND.
                                        &
                                            (yint .LE. ycrd(j+1))) then
                                                xint=xcoord(i-1)
                                                weight=(yint-ycrd(j))/(ycrd(j+1)-ycrd(j)) ! weight of
grater I indices of v on interpolation point
                                                iplu(k)=i-1
                                                jplu(k)=j+1
                                                ip2u(k)=i-1
                                                jp2u(k)=j
c                               write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,ycrd
c                               &scnd1',k,i,j,weight,ycrd(j),yint,ycrd(j+1),gradient,intercept
                                    Else
                                        yint=ycrd(j+1)
                                        xint=(yint-intercept)/gradient
                                        weight=(xint-xcoord(i-1))/(xcoord(i)-xcoord(i-1))
                                        If (acyl .EQ. xcoord(i)) then
                                            weight=1
                                            xint=xcoord(i)
                                        End if
                                        iplu(k)=i
                                        jplu(k)=j+1
                                        ip2u(k)=i-1
                                        jp2u(k)=j+1
c                               write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,xcoord
c                               &scnd2',k,i,j,weight,xcoord(i-1),xint,xcoord(i),gradient,intercept
                                    End if
                                End if

c
                                If ((ycrd(j) .GT. bcyl) .AND. (xcoord(i) .GT. acyl)) !
***** first quarter of circle *****
                                &
                                    then
                                        yint=gradient * xcoord(i+1)+intercept
c
                                    If ((yint .GE. ycrd(j)) .AND.
                                        &
                                            (yint .LE. ycrd(j+1))) then
                                                xint=xcoord(i+1)

```

```

                                weight=((yint-ycrd(j))/(ycrd(j+1)-ycrd(j))) ! weight of
grater I indices of v on interpolation point
                                iplu(k)=i+1
                                jplu(k)=j+1
                                ip2u(k)=i+1
                                jp2u(k)=j
c                                write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,xcoord
c                                &first2',k,i,j,weight,ycrd(j),yint,ycrd(j+1),gradient,intercept

                                Else
                                yint=ycrd(j+1)
                                xint=(yint-intercept)/gradient
                                weight=((xint-xcoord(i))/(xcoord(i+1)-xcoord(i)))
                                iplu(k)=i+1
                                jplu(k)=j+1
                                ip2u(k)=i
                                jp2u(k)=j+1
c                                write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,xcoord
c                                &first2',k,i,j,weight,xcoord(i),xint,xcoord(i+1),gradient,intercept
                                End if
                                End if

c
c                                If ((ycrd(j) .LE. bcyl) .AND. (xcoord(i) .GT. acyl))
***** fourth quater of circle ***** !
                                &                                then
c                                yint=gradient * xcoord(i+1)+intercept

                                If ((bcyl .EQ. ycrd(j)) .OR.
                                &                                ((yint .GE. ycrd(j-1)) .AND.
                                &                                (yint .LE. ycrd(j)))) then
                                xint=xcoord(i+1)
                                weight=((yint-ycrd(j-1))/(ycrd(j)-ycrd(j-1))) ! weight
of grater I indices of v on interpolation point
                                if (bcyl .EQ. ycrd(j)) then
                                        weight = 1
                                        yint = ycrd(i)
                                end if
                                iplu(k)=i+1
                                jplu(k)=j
                                ip2u(k)=i+1
                                jp2u(k)=j-1
c                                write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,xcoord
c                                &four1',k,i,j,weight,ycrd(j-1),yint,ycrd(j),gradient,intercept
                                Else
                                yint=ycrd(j-1)
                                xint=(yint-intercept)/gradient
                                weight=((xint-xcoord(i))/(xcoord(i+1)-xcoord(i)))
                                iplu(k)=i+1
                                jplu(k)=j-1
                                ip2u(k)=i
                                jp2u(k)=j-1
c                                write(*,'(A,3I5,6F10.2)')'i,ik,jk,weight,xcoord
c                                & four2',k,i,j,weight,xcoord(i),xint,xcoord(i+1),gradient,intercept
                                End if
                                End if

c
c
                                iinterpu(k)=i
                                jinterpu(k)=j
                                a1=sqrt((acyl-xcoord(i))**2+(bcyl-ycrd(j))**2)-Rcyl ! distance of
boundary point to the bondary of circle
                                c1=sqrt((acyl-xint)**2+(bcyl-yint)**2)-Rcyl ! distance of
interpolation point to the boundary of circle
cc                                wu1(k)=(a1/c1)*weight
cc                                wu2(k)=(a1/c1)*(1-weight)
                                wu1(k)=weight ! 21/5/13
                                wu2(k)=(a1/c1) !21/5/13

```

```

                if (a1 .EQ. 0) then ! point on the solid boudnary
                    wu1(k)=0          !21/5/13
                    wu2(k)=0          !21/5/13
                endif
            End if
c
        end do
    end do
    nbndu=k
    write(*,*) 'k,nbndu',k,nbndu
c
c
c ***** pressure interpolation indices on
moving boundary*****
    k=0
    nbndp = 0
    PI=4.*ATAN(1.)
    do j=8,ny-8
    do i=8,nx-8
        if ((pmask(i,j) .EQ. 1) .AND.
& ((pmask(i+1,j) .EQ. 0) .OR. (pmask(i-1,j) .EQ. 0) .OR.
& (pmask(i,j+1) .EQ. 0) .OR. (pmask(i,j-1) .EQ. 0))) then
c
            deltal= sqrt((xcrd(i)-acyl)**2+(ycrd(j)-bcyl)**2)
            gradient=(bcyl-ycrd(j))/(acyl-xcrd(i))
            intercept=bcyl-acyl*gradient
            k=k+1
            teta(k)=ATAN2((ycrd(j)-bcyl),(xcrd(i)-acyl))
            if (teta(k) .LE. 0) then
                teta(k)=teta(k)+ 2*PI
            end if
c
            teta(k)=ATAN2((bcyl-ycrd(j)),(acyl-xcrd(i)))
            unitvi(k)=(xcrd(i)-acyl)/deltal
            unitvj(k)=(ycrd(j)-bcyl)/deltal
            ip1p(k)=i
            jp1p(k)=j
            If ((ycrd(j) .LE. bcyl) .AND. (xcrd(i) .LE. acyl)) then      !
***** third quater of circle *****
                yint=ycrd(j-1)
                xint= (yint- intercept)/gradient      !update 21/5/13
c
                teta(k)=(2*PI/3)-(ATAN(gradient))
c
                if (acyl .EQ. xcrd(i)) teta(k)=2*PI/3
                if ((xint .GE. xcrd(i-1) .AND. (xint .LE. xcrd(i))) .OR.
& (acyl .EQ. xcrd(i))) then
                    wp1(k)=(xint-xcrd(i-1))/(xcrd(i)-xcrd(i-1))
c
                    if (acyl .EQ. xcrd(i)) then
                        wp1(k)=1
                        xint=xcrd(i)
                    end if
c
                    ip2p(k)=i
                    jp2p(k)=j-1
                    ip3p(k)=i-1
                    jp3p(k)=j-1
                else
                    xint=xcrd(i-1)
                    yint=gradient * xint+intercept
                    wp1(k)=(yint-ycrd(j-1))/(ycrd(j)-ycrd(j-1))
                    if (bcyl .EQ. ycrd(j)) then !21/5/21
                        wp1(k)=1          !21/5/21
                        yint=ycrd(j)      !21/5/21
                    endif
                    !21/5/21
                    ip2p(k)=i-1
                    jp2p(k)=j
                    ip3p(k)=i-1
                    jp3p(k)=j-1
                end if
            end if
        end do
    end do

```

```

        end if
    end if
c
        If ((ycrd(j) .GT. bcyl) .AND. (xcrd(i) .LE. acyl)) then      !
***** second quarter of circle *****
        yint=ycrd(j+1)
        xint= (yint- intercept)/gradient !update 21/5/13
c
        if ((xint .GE. xcrd(i-1) .and. (xint .LE. xcrd(i))) .OR.
&         (acyl .EQ. xcrd(i))) then
        wp1(k)=(xint-xcrd(i-1))/(xcrd(i)-xcrd(i-1))
c
                If (acyl .EQ. xcrd(i)) then
                wp1(k)=1
                xint=xcrd(i)
                End if
c
        ip2p(k)=i
        jp2p(k)=j+1
        ip3p(k)=i-1
        jp3p(k)=j+1
    else
        xint=xcrd(i-1)
        yint=gradient * xint+intercept
        wp1(k)=(yint-ycrd(j))/(ycrd(j+1)-ycrd(j))
        ip2p(k)=i-1
        jp2p(k)=j+1
        ip3p(k)=i-1
        jp3p(k)=j
    end if
end if
c
    If ((ycrd(j) .GT. bcyl) .AND. (xcrd(i) .GT. acyl)) then ! upadte
21/5/13 ***** first quarter of circle *****
        yint=ycrd(j+1)
        xint= (yint - intercept)/gradient !update 21/5/13
c
        if ((xint .GE. xcrd(i)) .AND. (xint .LE. xcrd(i+1))) then
        wp1(k)=(xint-xcrd(i))/(xcrd(i+1)-xcrd(i))
        ip2p(k)=i+1
        jp2p(k)=j+1
        ip3p(k)=i
        jp3p(k)=j+1
    else
        xint=xcrd(i+1)
        yint=gradient * xint+intercept
        wp1(k)=(yint-ycrd(j))/(ycrd(j+1)-ycrd(j))
        ip2p(k)=i+1
        jp2p(k)=j+1
        ip3p(k)=i+1
        jp3p(k)=j
    end if
end if
c
    If ((ycrd(j) .LE. bcyl) .AND. (xcrd(i) .GT. acyl)) then ! update
21/5/13 ***** fourth quater of circle *****
        yint=ycrd(j-1)
        xint= (yint- intercept)/gradient
c
        teta(k)=2*PI*ATAN(abs(aradiant))
c
        if (acyl .EQ. xcrd(i)) teta(k)=3*PI/2
c
        if ((xint .GE. xcrd(i)) .AND. (xint .LE. xcrd(i+1))) then
        wp1(k)=(xint-xcrd(i))/(xcrd(i+1)-xcrd(i))
        ip2p(k)=i+1
        jp2p(k)=j-1
        ip3p(k)=i
        jp3p(k)=j-1

```

```

else
  xint=xcrd(i+1)
  yint=gradient * xint+intercept
  wp1(k)=(yint-ycrd(j-1))/(ycrd(j)-ycrd(j-1))
  if (bcyl .EQ. ycrd(j)) then
    wp1(k)=1
    yint=ycrd(j)
  endif
  ip2p(k)=i+1
  jp2p(k)=j
  ip3p(k)=i+1
  jp3p(k)=j-1
end if
c
end if
c
iinterpp(k)=i
jinterpp(k)=j
a1=sqrt((acyl-xcrd(i))**2+(bcyl-ycrd(j))**2)-Rcyl ! distance of
first pressure point to the boundary of circle
c1=sqrt((acyl-xint)**2+(bcyl-yint)**2)-Rcyl ! distance of
interpolation point to the boundary of circle
wp2(k)=(a1/c1)
delta1(k)=a1
end if
end do
end do
nbndp=k
write (*,*)'nbndp',nbndp
do k=1,nbndp
  do j=k+1,nbndp
    if (teta(j) .LT. teta(k)) then
      temp1=teta(j)
      temp2=iinterpp(j)
      temp3=jinterpp(j)
      temp4=ip1p(j)
      temp5=jp1p(j)
      temp6=ip2p(j)
      temp7=jp2p(j)
      temp8=ip3p(j)
      temp9=jp3p(j)
      temp10=wp1(j)
      temp11=wp2(j)
      temp12=unitvi(j)
      temp13=unitvj(j)
      temp14=delta1(j)
c
      teta(j)=teta(k)
      iinterpp(j)=iinterpp(k)
      jinterpp(j)=jinterpp(k)
      ip1p(j)=ip1p(k)
      jp1p(j)=jp1p(k)
      ip2p(j)=ip2p(k)
      jp2p(j)=jp2p(k)
      ip3p(j)=ip3p(k)
      jp3p(j)=jp3p(k)
      wp1(j)=wp1(k)
      wp2(j)=wp2(k)
      unitvi(j)=unitvi(k)
      unitvj(j)=unitvj(k)
      delta1(j)=delta1(k)
c
      teta(k)=temp1
      iinterpp(k)=temp2
      jinterpp(k)=temp3
      ip1p(k)=temp4
      jp1p(k)=temp5
      ip2p(k)=temp6

```

```

        jp2p(k)=temp7
        ip3p(k)=temp8
        jp3p(k)=temp9
        wp1(k)=temp10
        wp2(k)=temp11
        unitvi(k)=temp12
        unitvj(k)=temp13
        delta1(k)=temp14
c
        end if
    end do
end do
    teta(nbndp+1)=teta(1)+2 * 4. * ATAN (1.)
    iinterpp(nbndp+1)=iinterpp(1)
    jinterpp(nbndp+1)=jinterpp(1)
    ip1p(nbndp+1)=ip1p(1)
    jp1p(nbndp+1)=jp1p(1)
    ip2p(nbndp+1)=ip2p(1)
    jp2p(nbndp+1)=jp2p(1)
    ip3p(nbndp+1)=ip3p(1)
    jp3p(nbndp+1)=jp3p(1)
    wp1(nbndp+1)=wp1(1)
    wp2(nbndp+1)=wp2(1)
    unitvi(nbndp+1)=unitvi(1)
    unitvj(nbndp+1)=unitvj(1)
    delta1(nbndp+1)=delta1(1)
c
    teta(0)=teta(nbndp)-2 * 4. * ATAN (1.)
    iinterpp(0)=iinterpp(nbndp)
    jinterpp(0)=jinterpp(nbndp)
    ip1p(0)=ip1p(nbndp)
    jp1p(0)=jp1p(nbndp)
    ip2p(0)=ip2p(nbndp)
    jp2p(0)=jp2p(nbndp)
    ip3p(0)=ip3p(nbndp)
    jp3p(0)=jp3p(nbndp)
    wp1(0)=wp1(nbndp)
    wp2(0)=wp2(nbndp)
    unitvi(0)=unitvi(nbndp)
    unitvj(0)=unitvj(nbndp)
    delta1(0)=delta1(nbndp)

c****end of pressure indices interpolation *****
    return
end
cc
    subroutine bounds()
c
    integer nnx, nny, MxSurf, Mxy
    parameter (nnx=600, nny=850, MxSurf=50)
    parameter (Mxy=2*nnx+2*ny)
c
c
    common /cylzise/ acyl, bcyl, Rcyl
    double precision acyl, bcyl, Rcyl
c
    common /veloxx/ u(0:nnx,0:ny+1), v(0:nnx+1,0:ny),
&                p(0:nnx+1,0:ny+1), a(0:nnx,ny,4), b(nnx,0:ny,4)
    double precision u, v, p, a, b
c
    common /velotemp/ utemp(0:nnx,0:ny+1), vtemp(0:nnx+1,0:ny)
    double precision utemp, vtemp
c
    common / gridd/ xcrd(0:nnx+1), ycrd(0:ny+1), scalar(nnx,ny),
&                xfree(Mxy, MxSurf), yfree(Mxy, MxSurf)
    double precision xcrd, ycrd, scalar, xfree, yfree

```



```

c
common / griddx/ xcoord(0:nxx), ycoord(0:nny)
double precision xcoord, ycoord
c
common / bndvinfi/ ip1v(Mxy), jplv(Mxy), ip2v(Mxy), jp2v(Mxy),
& iinterp(Mxy), jinterp(Mxy), nbndv
integer ip1v, jplv, ip2v, jp2v, iinterp, jinterp, nbndv
c
common / bnduinfi/ ip1u(Mxy), jplu(Mxy), ip2u(Mxy), jp2u(Mxy),
& iinterpu(Mxy), jinterpu(Mxy), nbndu
integer ip1u, jplu, ip2u, jp2u, iinterpu, jinterpu, nbndu

common / bndvinfR/ wv1(Mxy), wv2(Mxy)
double precision wv1, wv2
c
common / bnduinfR/ wu1(Mxy), wu2(Mxy)
double precision wu1, wu2
c
common / masksx/ pmask(0:nxx, 0:nny), umask(0:nxx, 0:nny),
& vmask(0:nxx, 0:nny)
double precision pmask, umask, vmask
c
common / bndinf/ txu(nxx), txv(nxx), tyu(nxx), tyv(nxx),
& vnoru(nxx), vnorv(nxx), fyu(nxx), fyv(nxx),
& vup(0:nxx+1), uup(0:nxx), influx
double precision txu, txv, tyu, tyv, vnoru, vnorv, fyu, fyv, vup, uup,
& influx
c
common / sbody/ vsolid, ysolid, usolid, xsolid, aysolid, axsolid
double precision vsolid, ysolid, usolid, xsolid, aysolid, axsolid
c
common / bniinf/ jyu(nxx), jyv(nxx)
integer jyu, jyv
c
common / dimenx/ nx, ny, Re, RRe, dt, time, dts, nx2, ny2, nn
integer nx, ny, nx2, ny2, nn
double precision Re, RRe, dt, time, dts
c
double precision vi, vip, uip, uip2, xtg, ytg, dnorm, ubound, vbound, vtan,
& flux, bflux, fact, dux, ycl, uint, vint,
& vsolidRelative, usolidRelative
c
common / homeadd/ home
character*40 home
c
logical EX
c
write(*,*) '***** at the beginning of bounds*****'
c
c ** inlet boundary at the left grid-line
c
do j=1,ny
u(0,j)=1.D0
end do
c
do j=0,ny
c v(0,j)=-v(1,j)
v(0,j)=-vsolid
end do
c
c ** symmetry boundary at the upper and lower side
this is not fullfilled (except for 1<x<2)
c
do i=0,nx
c v(i,ny)=0
c u(i,ny+1)=u(i,ny) ! indices should be check to see if ny
is correct or ny+1 !

```

```

c      v(i,0 )=0
c      u(i,0 )=u(i,1 )
cc     v(i,ny)=vup(i)
c      end do
c *** relative velocity at the upper and lower side
      do i=0,nx
          v(i,ny)=-vsolid
          v(i,0)=-vsolid
c      u(i,0)= u(i,n+1) !period boundary for the u
          u(i,0 )=u(i,1 )
          u(i,ny+1)=u(i,ny)
      end do

c *****solid boundary around the cylinder (immersed
boudnary) *****
c
c      write(*,*)'nbndu,nbndv', nbndu,nbndv
          vsolidRelative=0
          usolidRelative=0
          do i=1,nbndv          !nbndv
              ik=iinterpv(i)
              jk=jinterpv(i)
              v(ik,jk)=(1-wv2(i))*vsolidRelative +
&                  wv2(i)* wv1(i) *vtemp(ip1v(i),jplv(i))+
&                  wv2(i)*(1-wv1(i)) *vtemp(ip2v(i),jp2v(i))
c      write(*,'(A,3I5,5F16.8)') 'i,ik,jk,x(ik),y(jk),v(ik,jk),wv1,wv2='
c      & ,i,ik,jk,xcrd(ik), ycrd(jk),v(ik,jk),wv1(i),wv2(i)
          end do
c
      do i=1,nbndu          !nbndu
          ik=iinterpu(i)
          jk=jinterpu(i)
          u(ik,jk)=(1-wu2(i))*usolidRelative+
&                  wu2(i) * wul(i) *u(ip1u(i),jplu(i))+
&                  wu2(i) *(1-wul(i))*u(ip2u(i),jp2u(i))
c      if ((u(ik,jk) .GE. 1) .OR. (u(ik,jk) .LE. -1)) then
c      write(*,*) 'i,ik,jk,u(ik,jk)=' ,i,ik,jk,u(ik,jk)
c      end if
      end do
c
cccc***** 22/5/13 defining velocity inside the solid*****
c
      do i=2,nx-1
          do j=2,ny-1
              if (sqrt((xcrd(i)-acyl)**2+(ycoord(j)-bcyl)**2) .LT. Rcyl) then
                  v(i,j)=vsolidRelative
              end if
cc      if (sqrt((xcoord(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) then
cc      uins(i,j)=usolid
cc      end if
cc      if (sqrt((xcrd(i)-acyl)**2+(ycrd(j)-bcyl)**2) .LT. Rcyl) then
cc      pins(i,j)=0
cc      end if
          end do
      end do
c *****end of part *****
c

c ***** part to improve divergence around the cylinder
c
ccc     do i=1,nx
ccc     do j=1,ny
ccc     if (umask(i,j)+umask(i-1,j)+vmask(i,j)+vmask(i,j-1) .EQ. 1) then
ccc     if (umask(i,j) .NE.1) then

```

```

ccc      u(i,j)=u(i-1,j)-
ccc      &          ((xcoord(i)-xcoord(i-1))/(ycoord(j)-ycoord(j-1)))*
ccc      &          (v(i,j)-v(i,j-1))
ccc      end if
c
ccc      if (umask(i-1,j) .NE. 1) then
ccc      u(i-1,j)=u(i,j)+
ccc      &          ((xcoord(i)-xcoord(i-1))/(ycoord(j)-ycoord(j-1)))*
ccc      &          (v(i,j)-v(i,j-1))
ccc      end if
c
ccc      if (vmask(i,j) .NE. 1) then
ccc      v(i,j)=v(i,j-1)-
ccc      &          ((ycoord(j)-ycoord(j-1))/(xcoord(i)-xcoord(i-1)))*
ccc      &          (u(i,j)-u(i-1,j))
ccc      end if
c
ccc      if (vmask(i,j-1) .NE. 1) then
ccc      v(i,j-1)=v(i,j)+
ccc      &          ((ycoord(j)-ycoord(j-1))/(xcoord(i)-xcoord(i-1)))*
ccc      &          (u(i,j)-u(i-1,j))
ccc      end if
c
ccc      end if
ccc      end do
ccc      end do

c      STOP
c
cc      do i=0,nx
cc      u(i,ny+1)=2*uup(i)-u(i,ny)
cc      end do
c
c

c      do i=2,nx-1
cc      do j=1,ny-1
cc      if (amask(i,j) .NE. amask(i-1,j)) then
cc      bflux=bflux-u(i,j)*(ycoord(j)-ycoord(j-1))
cc      end if
cc      end do
cc      end do
c
c
c ** Exit boundary conditions
c
flux=0.D0
do j=1,ny
u(nx ,j)=u(nx ,j)-umask(nx-1,j)*dt*(u(nx,j)-u(nx-1,j))/
&          (xcoord(nx)-xcoord(nx-1))
&          v(nx+1,j)=v(nx+1,j)-vmask(nx-1,j)*dt*(v(nx+1,j)-v(nx,j))/
&          (xcrd(nx+1)-xcrd(nx))
flux=flux+umask(nx-1,j)*u(nx,j)*(ycoord(j)-ycoord(j-1))
end do
c
if (flux .LT. 1D-6) then
flux=0.D0
do j=1,ny
u(nx,j)=umask(nx-1,j)
flux=flux+umask(nx-1,j)*u(nx,j)*(ycoord(j)-ycoord(j-1))
end do
end if
c
c ** Udata outflow for global mass conservation
c
cJW WARNING CHANGE THIS BACK LATER
bflux=0.D0

```

```

c***** this part took out to check the convergence problem
c
  do nbnd=1,nbndu
    i=iinterpu(nbnd)
    j=jinterpu(nbnd)
    if (pmask(i+1,j)+pmask(i,j) .EQ. 1) then
      if (pmask(i+1,j) .EQ. 1.D0) then
        bflux=bflux+u(i,j)*(ycoord(j)-ycoord(j-1))
      else if (pmask(i,j) .EQ. 1.D0) then
        bflux=bflux-u(i,j)*(ycoord(j)-ycoord(j-1))
      end if
    end if
  end do
c
  do nbnd=1,nbndv
    i=iinterpv(nbnd)
    j=jinterpv(nbnd)
    if (pmask(i,j+1)+pmask(i,j) .EQ. 1) then
      if (pmask(i,j+1) .EQ. 1.D0) then
        bflux=bflux+v(i,j)*(xcoord(i)-xcoord(i-1))
      else if (pmask(i,j) .EQ. 1.D0) then
        bflux=bflux-v(i,j)*(xcoord(i)-xcoord(i-1))
      end if
    end if
  end do
c
c ***** this part has been added to improve the divergence problem
ccc  do i=2, nx-1
ccc  do j=2, ny-1
ccc    if (pmask(i+1,j)+pmask(i,j) .EQ. 1) then
ccc      if (pmask(i+1,j) .EQ. 1.D0) then
ccc        bflux=bflux+u(i,j)*(ycoord(j)-ycoord(j-1))
ccc      else if (pmask(i,j) .EQ. 1.D0) then
ccc        bflux=bflux-u(i,j)*(ycoord(j)-ycoord(j-1))
ccc      end if
ccc    end if
ccc    if (pmask(i,j+1)+pmask(i,j) .EQ. 1) then
ccc      if (pmask(i,j+1) .EQ. 1.D0) then
ccc        bflux=bflux+v(i,j)*(xcoord(i)-xcoord(i-1))
ccc      else if (pmask(i,j) .EQ. 1.D0) then
ccc        bflux=bflux-v(i,j)*(xcoord(i)-xcoord(i-1))
ccc      end if
ccc    end if
ccc  end do
ccc  end do

c    write(*,*) 'outflux,bflux = ',flux,bflux
c
c    fact=(influx+bflux)/flux
c    write(*,*) 'influx, BFLUX, fact = ',influx,bflux,influx-bflux,fact
c    do j=1,ny
c      u(nx,j)=fact*umask(nx-1,j)*u(nx,j)
c    end do
c    call fillf()
c    STOP
c
c    write(*,*)' ***** at the end of bounds*****'
c    return
c    end

cc
cc
c    subroutine init()
c
c    integer nnx, nny, MxSurf, Mxy
c    parameter (nnx=600, nny=850, MxSurf=50)
c    parameter (Mxy=2*nnx+2*nny)
c
c    common /veloxx/ u(0:nnx,0:nny+1),v(0:nnx+1,0:nny),

```

```

&          p(0:nnx+1,0:nnx+1), a(0:nnx,nnx,4), b(nnx,0:nnx,4)
double precision u,v,p,a,b
c
common / gridd/ xcrd(0:nnx+1), ycrd(0:nnx+1), scalar(nnx,nnx),
&          xfree(Mxy,MxSurf), yfree(Mxy,MxSurf)
double precision xcrd, ycrd, scalar, xfree, yfree
c
common / griddx/ xcoord(0:nnx), ycoord(0:nnx)
double precision xcoord, ycoord
c
common / bndinf/ txu(nnx), txv(nnx), tyu(nnx), tyv(nnx),
&          vnoru(nnx), vnorv(nnx), fyu(nnx), fyv(nnx),
&          vup(0:nnx+1), uup(0:nnx), influx
double precision txu, txv, tyu, tyv, vnoru, vnorv, fyu, fyv, vup, uup,
&          influx
c
common / bndvinfi/ ip1v(Mxy), jp1v(Mxy), ip2v(Mxy), jp2v(Mxy),
&          iinterp(Mxy), jinterp(Mxy), nbndv
integer ip1v, jp1v, ip2v, jp2v, iinterp, jinterp, nbndv
c
common / bnduinfi/ ip1u(Mxy), jp1u(Mxy), ip2u(Mxy), jp2u(Mxy),
&          iinterpu(Mxy), jinterpu(Mxy), nbndu
integer ip1u, jp1u, ip2u, jp2u, iinterpu, jinterpu, nbndu

common / bndvinfR/ wv1(Mxy), wv2(Mxy)
double precision wv1, wv2
c
common / bnduinfR/ wu1(Mxy), wu2(Mxy)
double precision wu1, wu2
c
common / masksx/ pmask(0:nnx,0:nnx), umask(0:nnx,0:nnx),
&          vmask(0:nnx,0:nnx)
double precision pmask, umask, vmask
c
common / bniinf/ jyu(nnx), jyv(nnx)
integer jyu, jyv
c
common / dimenx/ nx, ny, Re, RRe, dt, time, dts, nx2, ny2, nn
integer nx, ny, nx2, ny2, nn
double precision Re, RRe, dt, time, dts
c
common / rkcom / urk(0:nnx,0:nnx+1), vrk(0:nnx+1,0:nnx),
&          ar(10,10), br(10), nrk
double precision urk, vrk, ar, br
integer nrk
c
double precision vi, vip, uip, uip2, xtg, ytg, dnorm, ubound, vbound,
&          vtan, flux, fact, dx
double precision cdx(100), udx(100), vdx(100)
integer nil, one
logical EX
c
common / sbody/ vsolid, ysolid, usolid, xsolid, aysolid, axsolid
double precision vsolid, ysolid, usolid, xsolid, aysolid, axsolid
c
c
common / epsili/ epstemp
double precision epstemp
c
common / homeadd/ home
character*40 home
c
epstemp=5.0D-7
c
write(*,*)' *****beginning of init*****'
Re = 100.D0

```

```

RRe = 1D0/Re
dt = 0.001D0 ! it was 0.001D0
time= 0.D0
vsolid =0.D0
ysolid =0.D0
usolid =0.D0
xsolid =0.D0

C
C ** Set the initial velocities and pressure
C
u=1.D0
urk=1.D0
do i=1,nx-1
u(i,0 )=1.D0
u(i,ny+1)=1.D0
end do
do j=0,ny ! it was ny+1
do i=1,nx-1 ! it was nx
u(i,j)=1.D0*umask(i,j)
urk(i,j)=1.D0 !*umask(i,j)
end do
end do

C
do j=0,ny ! it was ny
do i=0,nx+1 ! it was nx+1
v(i,j)=0.D0
vrk(i,j)=0.D0
end do
end do

C
do j=10,ny-10
do i=10,nx-10
u(i,j)=1.D0 *umask(i,j)
urk(i,j)=1.D0*umask(i,j)
v(i,j)=1.D0*vmask(i,j)
vrk(i,j)=1.D0*vmask(i,j)
end do
end do

C
do j=1,10
br(j) = 0.0D0
do i=1,10
ar(i,j) = 0.0D0
end do
end do

C
nrk = 3
ar(2,1) = 2.0D0/3.0D0
ar(3,2) = 2.0D0/3.0D0
br(1) = 0.250D0
br(2) = 0.375D0
br(3) = 0.375D0

C
do l=1,4
do j=1,ny
do i=0,nx
a(i,j,l)=0.D0
end do
end do
end do

C
do l=1,4
do j=0,ny
do i=1,nx
b(i,j,l)=0.D0
end do
end do
end do

```

```

c
  do j=0,ny+1
  do i=0,nx+1
    p(i,j)=0.D0
  end do
end do

c ** Determine influx

c
  influx=0.D0
  further study
  do j=1,ny
    influx=influx+u(0,j)*(ycoord(j)-ycoord(j-1))
  end do

c
  write(*,*) 'influx = ',influx

c
  call getfld(ex)
  if (ex) then
    write(*,*) 'data has been read from file'
    write(*,*) 'time = ',time
  end if

c
  write(*,*) '*****end of init*****'
  return
end

cc
cc
  subroutine convec

c
  integer nnx, nny, MxSurf, Mxy
  parameter (nnx=600, nny=850, MxSurf=50)
  parameter (Mxy=2*nnx+2*ny)

c
  common /veloxx/ u(0:nnx,0:ny+1), v(0:nnx+1,0:ny),
&                p(0:nnx+1,0:ny+1), a(0:nnx,ny,4), b(nnx,0:ny,4)
  double precision u, v, p, a, b

c
  common /gridd/ xcrd(0:nnx+1), ycrd(0:ny+1), scalar(nnx,ny),
&              xfree(Mxy, MxSurf), yfree(Mxy, MxSurf)
  double precision xcrd, ycrd, scalar, xfree, yfree

c
  common /griddx/ xcoord(0:nnx), ycoord(0:ny)
  double precision xcoord, ycoord

c
  common /SORpar/ aim(nnx,ny), aip(nnx,ny), ajm(nnx,ny),
&              ajp(nnx,ny), diag(nnx,ny), f(nnx,ny)
  double precision aim, aip, ajm, ajp, diag, f

c
  common /bndvinfi/ ip1v(Mxy), jplv(Mxy), ip2v(Mxy), jp2v(Mxy),
&                iinterp(Mxy), jinterp(Mxy), nbndv
  integer ip1v, jplv, ip2v, jp2v, iinterp, jinterp, nbndv

c
  common /bnduinfi/ ip1u(Mxy), jplu(Mxy), ip2u(Mxy), jp2u(Mxy),
&                iinterpu(Mxy), jinterpu(Mxy), nbndu
  integer ip1u, jplu, ip2u, jp2u, iinterpu, jinterpu, nbndu

  common /bndvinfR/ wv1(Mxy), wv2(Mxy)
  double precision wv1, wv2

c
  common /bnduinfR/ wu1(Mxy), wu2(Mxy)
  double precision wu1, wu2

c
  common /masksx/ pmask(0:nnx,0:ny), umask(0:nnx,0:ny),
&              vmask(0:nnx,0:ny)
  double precision pmask, umask, vmask

c

```

```

common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,nx2,ny2,nn
double precision Re, RRe,dt,time,dts
c
common /rkcom / urk(0:nnx,0:nnny+1), vrk(0:nnx+1,0:nnny),
& ar(10,10), br(10), nrk
double precision urk, vrk, ar, br
integer nrk
c
common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
c
common /homeadd/ home
character*40 home
cc
integer i,j,k
c
write(*,*)' ***** at the beginning of convec *****'
call bounds()
c
c ** Save velocity at old time
c
do j=0,ny+1
do i=0,nx
urk(i,j) = u(i,j)
end do
end do
c
do j=0,ny
do i=0,nx+1
vrk(i,j) = v(i,j)
end do
end do
c
c ** Start doing RK substeps
c
do k1 = 1, nrk
C
do j=1,ny
do i=1,nx-1
u(i,j) = urk(i,j)
if (k1 .GT. 1) then
do j1=1,k1-1
u(i,j)=u(i,j)+dt*umask(i,j)*ar(k1,j1)*a(i,j,j1)
end do
end if
end do
end do
c
do j=1,ny-1
do i=1,nx
v(i,j) = vrk(i,j)
if (k1 .GT. 1) then
do j1=1,k1-1
v(i,j)=v(i,j)+dt*vmask(i,j)*ar(k1,j1)*b(i,j,j1)
ccc v(i,j)=v(i,j)+dt*ar(k1,j1)*b(i,j,j1)
end do
end if
end do
end do
c
call bounds
c
do j=1,ny
do i=1,nx-1
c
a(i,j,k1)=-0.25D0*umask(i,j)*(
& ((u(i,j)+u(i+1,j))*(u(i,j)+u(i+1,j)))-

```



```

&      (u(i,j)+u(i-1,j))*(u(i,j)+u(i-1,j)))/(xcrd(i+1)-xcrd(i))+
&      ((u(i,j)+u(i,j+1))*(v(i,j)+v(i+1,j))-
&      (u(i,j)+u(i,j-1))*(v(i,j-1)+v(i+1,j-1)))/
&      (ycoord(j)-ycoord(j-1))+ umask(i,j)*
& RRe*((u(i+1,j)-2*u(i,j)+u(i-1,j)))/(xcrd(i+1)-xcrd(i))**2)+
&      (u(i,j+1)-2*u(i,j)+u(i,j-1))/(ycoord(j)-ycoord(j-1))**2))-
&      axsolid
c
      end do
      end do
c
      do i=1,nx
      do j=1,ny-1
c
          b(i,j,k1)=-0.25D0*vmask(i,j)*(
ccc
          b(i,j,k1)=-0.25D0*(
&      ((v(i,j)+v(i,j+1))*(v(i,j)+v(i,j+1))-
&      (v(i,j)+v(i,j-1))*(v(i,j)+v(i,j-1)))/(ycrd(j+1)-ycrd(j))+
&      ((v(i,j)+v(i+1,j))*(u(i,j)+u(i,j+1))-
&      (v(i,j)+v(i-1,j))*(u(i-1,j)+u(i-1,j+1)))/
&      (xcoord(i)-xcoord(i-1))+ vmask(i,j)*
ccc
          &      (xcoord(i)-xcoord(i-1))+
& RRe*((v(i,j+1)-2*v(i,j)+v(i,j-1)))/(ycrd(j+1)-ycrd(j))**2)+
&      (v(i+1,j)-2*v(i,j)+v(i-1,j))/(xcoord(i)-xcoord(i-1))**2))-
&      aysolid
c
          end do
          end do
c
      end do
c
      do j=1,ny
      do i=1,nx
          u(i,j) = urk(i,j)
          v(i,j) = vrk(i,j)
          do j1=1,nrk
              u(i,j)=u(i,j)+dt*umask(i,j)*br(j1)*a(i,j,j1)
              v(i,j)=v(i,j)+dt*vmask(i,j)*br(j1)*b(i,j,j1)
ccc
              v(i,j)=v(i,j)+dt*br(j1)*b(i,j,j1)
          end do
          end do
          end do
c
      call bounds()
c
      write(*,*)'***** at the end of convec *****'
      return
      end
cc
cc
      subroutine calcuv
c
      integer nnx, nny, MxSurf, Mxy
      parameter (nnx=600, nny=850, MxSurf=50)
      parameter (Mxy=2*nnx+2*nny)
c
      common /veloxx/ u(0:nnx,0:nny+1),v(0:nnx+1,0:nny),
&      p(0:nnx+1,0:nny+1),a(0:nnx,nny,4),b(nnx,0:nny,4)
      double precision u,v,p,a,b
c
      common /gridd/ xcrd(0:nnx+1), ycrd(0:nny+1), scalar(nnx,nny),
&      xfree(Mxy,MxSurf), yfree(Mxy,MxSurf)
      double precision xcrd, ycrd, scalar, xfree, yfree
c
      common /griddx/ xcoord(0:nnx), ycoord(0:nny)
      double precision xcoord, ycoord

      common /dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn

```

```

integer nx,ny,nx2,ny2,nn
double precision Re, RRe,dt,time,dts
c
common /SORpar/ aim(nnx,ny),aip(nnx,ny),ajm(nnx,ny),
&               ajp(nnx,ny),diag(nnx,ny),f(nnx,ny)
double precision aim,aip,ajm,ajp,diag,f
c
common / bndvinfi/ ip1v(Mxy),jplv(Mxy),ip2v(Mxy),jp2v(Mxy),
&               iinterp(Mxy),jinterp(Mxy),nbndv
integer ip1v,jplv,ip2v,jp2v,iinterp,jinterp,nbndv
c
common / bnduinfi/ ip1u(Mxy),jplu(Mxy),ip2u(Mxy),jp2u(Mxy),
&               iinterpu(Mxy),jinterpu(Mxy),nbndu
integer ip1u,jplu,ip2u,jp2u,iinterpu,jinterpu,nbndu

common / bndvinfR/ wv1(Mxy),wv2(Mxy)
double precision wv1,wv2
c
common / bnduinfR/ wu1(Mxy),wu2(Mxy)
double precision wu1,wu2
c
common /masksx/ pmask(0:nnx,0:ny),umask(0:nnx,0:ny),
&               vmask(0:nnx,0:ny)
double precision pmask,umask,vmask
c
common /rkcom / urk(0:nnx,0:ny+1),vrk(0:nnx+1,0:ny),
&               ar(10,10),br(10),nrk
double precision urk,vrk,ar,br
integer nrk
common /epsili/epstemp
double precision epstemp,eps
c
integer i,j,k
c
common /homeadd/ home
character*40 home
c
c
eps = epstemp
c
c
call solve(eps,iterat)
c
do j=1,ny
do i=1,nx-1
u(i,j) = u(i,j) - dt*umask(i,j)*
&       (p(i+1,j)-p(i,j))/(xcrd(i+1)-xcrd(i))
end do
end do
c
do j=1,ny-1
do i=1,nx
c
v(i,j) = v(i,j) - dt*pmask(i,j)*pmask(i,j+1)*
v(i,j) = v(i,j) - dt*vmask(i,j)*
&       (p(i,j+1)-p(i,j))/(ycrd(j+1)-ycrd(j))
end do
end do
c
return
end
cc
cc
subroutine mean()
c
integer nnx,ny,MxSurf,Mxy
parameter (nnx=600,ny=850,MxSurf=50)
parameter (Mxy=2*nnx+2*ny)
c

```

```

common /veloxx/ u(0:nnx,0:nnny+1),v(0:nnx+1,0:nnny),
&                p(0:nnx+1,0:nnny+1),a(0:nnx,nnny,4),b(nnx,0:nnny,4)
double precision u,v,p,a,b
c
common /velmen/ um(nnx,nnny), vm(nnx,nnny), pm(nnx,nnny),
&                uu(nnx,nnny), vv(nnx,nnny), uv(nnx,nnny)
double precision um,vm,pm,uu,vv,uv
c
common /parmen/ nmean
integer nmean

common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,nx2,ny2,nn
double precision Re, RRe,dt,time,dts
c
double precision fac
c
integer i,j,k
c
common /homeadd/ home
character*40 home
c
if (nmean .EQ. 0) then
do j=1,ny
do i=1,nx
um(i,j)=0.0D0
vm(i,j)=0.0D0
pm(i,j)=0.0D0
uu(i,j)=0.0D0
vv(i,j)=0.0D0
uv(i,j)=0.0D0
end do
end do
end if
c
nmean=nmean+1
fac=1.D0/nmean
do j=1,ny
do i=1,nx
um(i,j)=(1.D0-fac)*um(i,j)+0.50D0*fac*(u(i-1,j)+u(i,j))
vm(i,j)=(1.D0-fac)*vm(i,j)+0.50D0*fac*(v(i,j-1)+v(i,j))
pm(i,j)=(1.D0-fac)*pm(i,j)+ fac* p(i,j)
uu(i,j)=(1.D0-fac)*uu(i,j)+0.25D0*fac*(u(i-1,j)+u(i,j))**2
vv(i,j)=(1.D0-fac)*vv(i,j)+0.25D0*fac*(v(i,j-1)+v(i,j))**2
uv(i,j)=(1.D0-fac)*uv(i,j)+0.25D0*fac*(u(i-1,j)+u(i,j))*
&                (v(i,j-1)+v(i,j))
end do
end do
c
return
end
cc
cc
subroutine inisol()
c
parameter (nnx=600,nnny=850,nnxy=nnx*nnny,MxSurf=50)
parameter (Mxy=2*nnx+2*nnny)
c
common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,ny2,nn,nx3,nn1
double precision Re, RRe,dt,time,dts

common / gridd/ xcrd(0:nnx+1), ycrd(0:nnny+1), scalar(nnx,nnny),
&                xfree(Mxy,MxSurf), yfree(Mxy,MxSurf)
double precision xcrd, ycrd, scalar, xfree, yfree
c
common / griddx/ xcoord(0:nnx), ycoord(0:nnny)

```

```

double precision xcoord, ycoord
c
common / bndvinfi/ ip1v(Mxy),jp1v(Mxy),ip2v(Mxy),jp2v(Mxy),
& iinterpv(Mxy),jinterpv(Mxy),nbndv
integer ip1v,jp1v,ip2v,jp2v,iinterpv,jinterpv,nbndv
c
common / bnduinfi/ ip1u(Mxy),jp1u(Mxy),ip2u(Mxy),jp2u(Mxy),
& iinterpu(Mxy),jinterpu(Mxy),nbndu
integer ip1u,jp1u,ip2u,jp2u,iinterpu,jinterpu,nbndu

common / bndvinfR/ wv1(Mxy),wv2(mxy)
double precision wv1,wv2
c
common / bnduinfR/ wu1(Mxy),wu2(Mxy)
double precision wu1,wu2

c
common /masksx/ pmask(0:nnx,0:nny),umask(0:nnx,0:nny),
& vmask(0:nnx,0:nny)
double precision pmask,umask,vmask
c
common /bniinf/ jyu(nnx),jyv(nnx)
integer jyu,jyv
c
common /indi / li(nnx),maxit
integer li
c
common /coefs / ae(nnxy),aw(nnxy),an(nnxy),as(nnxy),ap(nnxy),
& fp(nnxy),alfa
double precision ae,aw,an,as,ap,fp,alfa
c
common /ludeco/ un(-nny:nnxy),ue(-nny:nnxy),lw(nnxy),
& ls(nnxy),lpr(nnxy)
double precision un,ue,lw,ls,lpr
c
double precision p1,p2
c
common /epsili/epstemp
double precision epstemp
c
common /homeadd/ home
character*40 home
c
write(*,*)' ***** at the beginning of inisol*****'
maxit = 5000
alfa = 0.92D0
do i=-nny,nnxy
    ue(i)=0.D0
    un(i)=0.D0
end do
c
nxy=nx*ny
do i=1,nx
do j=1,ny
    li(i)=(i-1)*ny
END DO
c
do i=1,nx
do j=1,ny
c
    ij=li(i)+j
    ij=(i-1)*ny+J
    ae(ij)=(ycoord(j)-ycoord(j-1))/(xcrd(i+1)-xcrd(i))
    an(ij)=(xcoord(i)-xcoord(i-1))/(ycrd(j+1)-ycrd(j))
    aw(ij)=(ycoord(j)-ycoord(j-1))/(xcrd(i)-xcrd(i-1))
    as(ij)=(xcoord(i)-xcoord(i-1))/(ycrd(j)-ycrd(j-1))
end do
end do
c
! solid boundary
do i=1,nx-1
do j=1,ny-1

```

```

      ij=(i-1)*ny+j
      If (pmask(i,j) .NE. 0) then
        if (pmask(i+1,j).EQ.0) then
          ae(ij)=0.D0
        end if
        if (pmask(i-1,j).EQ.0) then
          aw(ij)=0.D0
        end if
        if (pmask(i,j+1).EQ.0) then
          an(ij)=0.D0
        end if
        if (pmask(i,j-1).EQ.0) then
          as(ij)=0.D0
        end if
      end if
    end do
  end do

c                                     ! west and east boundary

  do j=1,ny
    i=1
    aw((i-1)*ny+j)=0.D0
    i=nx
    ae((i-1)*ny+j)=0.D0
  end do

c

  do i=1,nx                                     ! north and south boundary
    ij=(i-1)*ny+1
    as(ij)=0.D0
    ij=(i-1)*ny+ny
    an(ij)=0.D0
  end do

c
c
c  do i=1,nx
c    as(li(i)+jyv(i))=0.D0 ! this is for the immersed boundary
c    as(li(i)+      1)=0.D0
c    an(li(i)+      ny)=0.D0
c    do j=1,ny
c      ij=li(i)+j
c      if (amask(i,j).EQ.0.D0) then
c        ae(ij)=0.D0
c      end if
c      if (i .GT. 1) then
c        if (amask(i-1,j).EQ.0.D0) aw(ij)=0.D0
c      end if
c      if (j .GT. 1) then
c        if (pmask(i,j-1).EQ.0.D0) as(ij)=0.D0
c      end if
c    end do
c  end do
c
c
c  do j=1,ny
c    aw(li( 1)+j)=0.D0
c    ae(li(nx)+j)=0.D0
c  end do
c
c  do i=1,nx
c  do j=1,ny
c    ij=(i-1)*ny +j
c    ap(ij)=-(ae(ij)+aw(ij)+an(ij)+as(ij))
c  end do
c  end do

C
C.....CALCULATE ELEMENTS OF [L] AND [U] MATRICES
c

  do i=1,nx
  do ij=(i-1)*ny+1,(i-1)*ny+ny
    lw(ij)=aw(ij)/(1.D0+alfa*un(ij-ny))
  end do
  end do

```

```

        ls(ij)=as(ij)/(1.D0+alfa*ue(ij- 1))
        p1=alfa*lw(ij)*un(ij-ny)
        p2=alfa*ls(ij)*ue(ij- 1)
        lpr(ij)=1.D0/(ap(ij)+p1+p2-lw(ij)*ue(ij-ny)-ls(ij)*un(ij-1))
        un(ij)=(an(ij)-p1)*lpr(ij)
        ue(ij)=(ae(ij)-p2)*lpr(ij)
    end do
end do

c
c
c
c    write(*,*)'***** at the end of inisol*****'
    return
end
cc

subroutine fillf()
c
    integer nnx, nny, MxSurf, Mxy
    parameter (nnx=600, nny=850, nnxy=nnx*nyy, MxSurf=50)
    parameter (Mxy=2*nnx+2*nyy)
c
    common /veloxx/ u(0:nnx,0:nyy+1), v(0:nnx+1,0:nyy),
&                p(0:nnx+1,0:nyy+1), a(0:nnx,nyy,4), b(nnx,0:nyy,4)
    double precision u, v, p, a, b

    common /rkcom / urk(0:nnx,0:nyy+1), vrk(0:nnx+1,0:nyy),
&                ar(10,10), br(10), nrk
    double precision urk, vrk, ar, br
    integer nrk
c
    common / bndinf/ txu(nnx), txv(nnx), tyu(nnx), tyv(nnx),
&                vnoru(nnx), vnorv(nnx), fyu(nnx), fyv(nnx),
&                vup(0:nnx+1), uup(0:nnx), influx
    double precision txu, txv, tyu, tyv, vnoru, vnorv, fyu, fyv, vup, uup,
&                influx
c
    common / bniinf/ jyu(nnx), jyv(nnx)
    integer jyu, jyv
c
    common / gridd/ xcrd(0:nnx+1), ycrd(0:nyy+1), scalar(nnx,nyy),
&                xfree(Mxy, MxSurf), yfree(Mxy, MxSurf)
    double precision xcrd, ycrd, scalar, xfree, yfree
c
    common / griddx/ xcoord(0:nnx), ycoord(0:nyy)
    double precision xcoord, ycoord
c
    common / dimenx/ nx, ny, Re, RRe, dt, time, dts, nx2, ny2, nn
    integer nx, ny, nx2, ny2, nn
    double precision Re, RRe, dt, time, dts
c
    common /coefs/ ae(nnxy), aw(nnxy), an(nnxy), as(nnxy), ap(nnxy),
&                fp(nnxy), alfa
    double precision ae, aw, an, as, ap, fp, alfa
c
    common /pcorterm/ pctw(nnx,nyy), pcte(nnx,nyy), !pressure correction
&                pctn(nnx,nyy), pcts(nnx,nyy),
&                pctIBn(nnx,nyy), pctIBs(nnx,nyy),
&                pctIBe(nnx,nyy), pctIBw(nnx,nyy)
    double precision pctw, pcte, pctn, pcts, pctIBn, pctIBs, pctIBe, pctIBw
c
    common / bndvinfi/ ip1v(Mxy), jp1v(Mxy), ip2v(Mxy), jp2v(Mxy),
&                iinterp(Mxy), jinterp(Mxy), nbndv
    integer ip1v, jp1v, ip2v, jp2v, iinterp, jinterp, nbndv
c
    common / bnduinfi/ ip1u(Mxy), jp1u(Mxy), ip2u(Mxy), jp2u(Mxy),
&                iinterpu(Mxy), jinterpu(Mxy), nbndu
    integer ip1u, jp1u, ip2u, jp2u, iinterpu, jinterpu, nbndu

```

```

common / bndvinfR/ wv1(Mxy),wv2(mxy)
double precision wv1,wv2
C
common / bnduinfR/ wu1(Mxy),wu2(Mxy)
double precision wu1,wu2
C
common /masksx/ pmask(0:nnx,0:nny),umask(0:nnx,0:nny),
&
vmask(0:nnx,0:nny)
double precision pmask,umask,vmask
C
common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
C
common /indi / li(nnx),maxit
integer li, maxit
C
double precision sumf,flux,pctIBsc,pctIBsd,pctIBnc,pctIBnd
integer i,j,k
C
common /homeadd/ home
character*40 home
C
C ** f is basically the divergence of (u,v) as calculated in convec
C
C write(*,*) 'dt = ',dt
C ***** calculating correction term neumann boundary*****
C ***** poisson pressure equation *****
C
pctw=0.D0
pcte=0.D0
pctn=0.D0
pcts=0.D0
pctIBn=0.D0
pctIBs=0.D0
pctIBe=0.D0
pctIBw=0.D0
pctIBnc=0.D0
pctIBnd=0.0D0
pctIBsc=0.D0
pctIBsd=0.D0
pctIBec=0.D0
pctIBed=0.0D0
pctIBwc=0.D0
pctIBwd=0.D0

C
C *****west and east boundary
C do j=1, ny
C i=1
C pctw(1,j)=((ycoord(j)-ycoord(j-1))/(xcrd(i)-xcrd(i-1)))*!aw(ij)
C & ((u(1,j)**2-u(0,j)**2)/(xcrd(1)-xcrd(0))-
C & RRe*(u(2,j)-2*u(1,j)+u(0,j))/(xcrd(i+1)-xcrd(i))**2)*
C & (xcrd(1)-xcrd(0))
C i=nnx
C pcte(nnx,j)=((ycoord(j)-ycoord(j-1))/(xcrd(i+1)-xcrd(i)))*!ae(ij)
C & (-1*(u(nnx,j)**2-u(nnx-1,j)**2)/(xcrd(i)-xcrd(i-1)))+
C & RRe*((u(nnx-2,j)-2*u(nnx-1,j)+u(nnx,j))/(xcrd(i)-xcrd(i-1))**2+
C & (u(nnx,j-1)-2*u(nnx,j)+u(nnx,j+1))/(ycrd(j+1)-ycrd(j))**2))*
C & (xcrd(nnx+1)-xcrd(nnx))
C
C end do
C *****north and south boundary
C do i=1,nnx
C j=ny
C pctn(i,ny)=((xcoord(i)-xcoord(i-1))/(ycrd(j+1)-ycrd(j)))* !an(ij)
C & ((v(i,ny)**2-v(i,ny-1)**2)/(ycoord(j)-ycoord(j-1)))+
C & ((u(i,ny)*v(i,ny)-

```

```

&          u(i-1,ny)*v(i,ny))/
&          (xcoord(i)-xcoord(i-1)))+
&          (v(i,j+1)-vrk(i,j+1))/dt)*! this might not be ture
&          (ycrd(j+1)-ycrd(j))*0.D0

      j=1
pcts(i,j)=((xcoord(i)-xcoord(i-1))/(ycrd(j)-ycrd(j-1)))*!as((ij)
&          ((v(i,j)**2-v(i,j-1)**2)/(ycoord(j)-ycoord(j-1)))+
&          (u(i,0)*v(i,0)-
&          u(i-1,0)*v(i,0)))/
&          (xcoord(i)-xcoord(i-1))+
&          (v(i,j-1)-vrk(i,j-1))/dt)*! this might not be true
&          (ycrd(j)-ycrd(j-1))*0.D0
end do
c   write(*,*)'i=202,pcts,pctn,pt,pb',pcts(202,1),pctn(202,ny),
c   &          p(202,286),p(202,198)
c   call bounds
c
do nbnd=1,nbndv
  i=iinterpv(nbnd)
  j=jinterpv(nbnd)
  if (pmask(i,j+1)+pmask(i,j) .EQ. 1) then
    if (pmask(i,j+1) .EQ. 1.D0) then !top half
      if ((pmask(i+1,j)+pmask(i+2,j)) .EQ. 0) then!left half
pctIBsc=((v(i,j+1)**2-v(i,j)**2)/(ycoord(j+1)-ycoord(j))+
&        (u(i,j+1)*0.5*(v(i,j)+v(i,j+1))-
&        u(i-1,j+1)*0.5*(v(i,j)+v(i,j+1)))/
&        (xcoord(i)-xcoord(i-1)))*0.D0
pctIBsd=RRe*(-1*(v(i,j)-2*v(i-1,j)+v(i-2,j)))/
&          (xcrd(i)-xcrd(i-1))**2+
&          ((u(i,j+2)-u(i-1,j+2))/(xcoord(i)-xcoord(i-1))-
&          (u(i,j+1)-u(i-1,j+1))/(xcoord(i)-xcoord(i-1)))/
&          (ycrd(j+2)-ycrd(j+1)))*0.D0
Fycon=fycon+v(i,j)*v(i,j)
c
      else! right half
pctIBsc=((v(i,j+1)**2-v(i,j)**2)/(ycoord(j+1)-ycoord(j))+
&        (u(i,j+1)*0.5*(v(i,j)+v(i,j+1))-
&        u(i-1,j+1)*0.5*(v(i,j)+v(i,j+1)))/
&        (xcoord(i)-xcoord(i-1)))*0.D0
pctIBsd= RRe*(-1*(v(i,j)-2*v(i+1,j)+v(i+2,j)))/
&          (xcrd(i+1)-xcrd(i))**2+
&          ((u(i,j+2)-u(i-1,j+2))/(xcoord(i)-xcoord(i-1))-
&          (u(i,j+1)-u(i-1,j+1))/(xcoord(i)-xcoord(i-1)))/
&          (ycrd(j+2)-ycrd(j+1)))*0.D0
      end if
pctIBs(i,j+1)=((xcoord(i)-xcoord(i-1))/(ycrd(j)-ycrd(j-1)))*
&          (pctIBsc+pctIBsd+(v(i,j)-vrk(i,j))/dt+aysolid)*
&          (ycrd(j+1)-ycrd(j))
c   write(*,*)'i,j,pctIBs',i,j,pctIBsc,pctIBsd,pctIBs(i,j+1),v(i,j),
c   &vrk(i,j)
c
      else if (pmask(i,j) .EQ. 1.D0) then!bottom
        if ((pmask(i+1,j)+pmask(i+2,j)) .EQ. 0) then!left half
pctIBnc=((v(i,j)**2-v(i,j-1)**2)/(ycoord(j)-ycoord(j-1)))+
&        (u(i,j)*0.5*(v(i,j)+v(i,j-1))-
&        u(i-1,j)*0.5*(v(i,j)+v(i,j-1)))/
&        (xcoord(i)-xcoord(i-1)))*0.D0
c
pctIBnd=RRe*(-1*(v(i,j)-2*v(i-1,j)+v(i-2,j)))/
&          (xcrd(i)-xcrd(i-1))**2+
&          ((u(i,j)-u(i,j-1))/(ycrd(j)-ycrd(j-1))-
&          (u(i-1,j)-u(i-1,j-1))/(ycrd(j)-ycrd(j-1)))/
&          (xcoord(i)-xcoord(i-1)))*0.D0
c
      else !right half
pctIBnc=((v(i,j)**2-v(i,j-1)**2)/(ycoord(j)-ycoord(j-1)))+

```



```

&          (u(i,j)*0.5*(v(i,j)+v(i,j-1))-
&          u(i-1,j)*0.5*(v(i,j)+v(i,j-1)))/
&          (xcoord(i)-xcoord(i-1))!*0.D0
pctIBnd= RRe*(-1*(v(i,j)-2*v(i+1,j)+v(i+2,j))/
&          (xcrd(i+1)-xcrd(i))**2+
&          ((u(i,j)-u(i,j-1))/(ycrd(j)-ycrd(j-1))-
&          (u(i-1,j)-u(i-1,j-1))/(ycrd(j)-ycrd(j-1))))/
&          (xcoord(i)-xcoord(i-1))!*0.D0
c
      end if
      pctIBn(i,j)=(xcoord(i)-xcoord(i-1))/(ycrd(j+1)-ycrd(j))*
&          (pctIBnc+pctIBnd+(v(i,j)-vrk(i,j))/dt+aysolid)*
&          (ycrd(j-1)-ycrd(j-2))
c      write(*,*)'i,j,pctIBn',i,j,pctIBnc,pctIBnd,pctIBn(i,j),v(i,j)
c      &,vrk(i,j)
      end if
    end if
  end do
c
do nbnd=1,nbndu
  i=iinterpu(nbnd)
  j=jinterpu(nbnd)
  if (pmask(i+1,j)+pmask(i,j) .EQ. 1) then
    if (pmask(i+1,j) .EQ. 0.D0) then !left half
      if ((pmask(i+1,j-1)+pmask(i+1,j-2)) .EQ. 0) then!top half
pctIBec=((u(i,j)**2-u(i-1,j)**2)/(xcoord(i)-xcoord(i-1))+
&          (0.5*(u(i-1,j)+u(i,j))*(v(i,j)-v(i,j-1)))/
&          (ycoord(j)-ycoord(j-1)))*!0.D0
pctIBed=RRe*(-1*(u(i,j)-2*u(i,j+1)+u(i,j+2))/
&          (ycrd(j+1)-ycrd(j))**2+
&          ((v(i,j)-v(i,j-1))/(ycoord(j)-ycoord(j-1))-
&          (v(i-1,j)-v(i-1,j-1))/(ycoord(j)-ycoord(j-1)))/
&          (xcrd(i)-xcrd(i-1)))*!0.D0
c
      else! bottom half
pctIBec=((u(i,j)**2-u(i-1,j)**2)/(xcoord(i)-xcoord(i-1))+
&          (0.5*(u(i-1,j)+u(i,j))*(v(i,j)-v(i,j-1)))/
&          (ycoord(j)-ycoord(j-1)))*!0.D0
pctIBed=RRe*(-1*(u(i,j)-2*u(i,j-1)+u(i,j-2))/
&          (ycrd(j)-ycrd(j-1))**2+
&          ((v(i,j)-v(i,j-1))/(ycoord(j)-ycoord(j-1))-
&          (v(i-1,j)-v(i-1,j-1))/(ycoord(j)-ycoord(j-1)))/
&          (xcrd(i)-xcrd(i-1)))*!0.D0
      end if
      pctIBe(i,j)=(ycoord(j)-ycoord(j-1))/(xcrd(i+1)-xcrd(i))*
&          (pctIBec+pctIBed+(u(i,j)-urk(i,j))/dt)*
&          (xcrd(i+1)-xcrd(i))
c      write(*,*)'i,j,pctIBe',i,j,pctIBec,pctIBed,pctIBe(i,j),u(i,j)
c      &urk(i,j)
c
      else if (pmask(i+1,j) .EQ. 1.D0) then!Right half
        if ((pmask(i,j-1)+pmask(i,j-2)) .EQ. 0) then!top half
pctIBwc=((u(i+1,j)**2-u(i,j)**2)/(xcoord(i+1)-xcoord(i))+
&          (0.5*(u(i+1,j)+u(i,j))*(v(i+1,j)-v(i+1,j-1)))/
&          (ycoord(j)-ycoord(j-1)))*!0.D0
pctIBwd=RRe*(-1*(u(i,j)-2*u(i,j+1)+u(i,j+2))/
&          (ycrd(j+1)-ycrd(j))**2+
&          ((v(i+1,j)-v(i+1,j-1))/(ycoord(j)-ycoord(j-1))-
&          (v(i+2,j)-v(i+2,j-1))/(ycoord(j)-ycoord(j-1)))/
&          (xcrd(i+2)-xcrd(i+1)))*!0.D0
        else !bottom half
pctIBwc=((u(i+1,j)**2-u(i,j)**2)/(xcoord(i+1)-xcoord(i))+
&          (0.5*(u(i+1,j)+u(i,j))*(v(i+1,j)-v(i+1,j-1)))/
&          (ycoord(j)-ycoord(j-1)))*!0.D0

```

```

    pctIBwd=RRe*(-1*(u(i,j)-2*u(i,j-1)+u(i,j-2)) /
&                (ycrd(j)-ycrd(j-1))**2+
&                ((v(i+1,j)-v(i+1,j-1))/(ycoord(j)-ycoord(j-1))-
&                (v(i+2,j)-v(i+2,j-1))/(ycoord(j)-ycoord(j-1)))/
&                (xcrd(i+2)-xcrd(i+1)))*0.D0
c
        end if
        pctIBw(i+1,j)=(ycoord(j)-ycoord(j-1))/(xcrd(i)-xcrd(i-1))*
&                (pctIBwc+pctIBwd+(u(i,j)-urk(i,j))/dt)*
&                (xcrd(i+1)-xcrd(i))
c        write(*,*)'i,j,pctIBw',i,j,pctIBwc,pctIBwd,pctIBw(i+1,j),u(i,j)
c    &,urk(i,j)
        end if
        end if
    end do
c
    sumf = 0D0
    do i=1,nx
    do j=1,ny
        ij=(i-1)*ny+j
        fp(ij)=pmask(i,j)*
&                ((u(i,j)-u(i-1,j))*(ycoord(j)-ycoord(j-1))+
&                (v(i,j)-v(i,j-1))*(xcoord(i)-xcoord(i-1)))/dt)-
&                pctw(i,j)-pcte(i,j)+pctn(i,j)-pcts(i,j)+
&                pctIBn(i,j)-pctIBs(i,j)-pctIBe(i,j)+pctIBw(i,j)
        sumf=sumf+fp(ij)
    end do
    end do
c
    write(*,*) 'GLOBAL: ',sumf
c
    close(12)
c    STOP 'in fillf'
c
    return
    end
cc
double precision function maxdiv()
c
integer nnx, nny, MxSurf, Mxy
parameter (nnx=600, nny=850, MxSurf=50)
parameter (Mxy=2*nnx+2*ny)
c
common /veloxx/ u(0:nnx,0:ny+1),v(0:nnx+1,0:ny),
&                p(0:nnx+1,0:ny+1),a(0:nnx,ny,4),b(nnx,0:ny,4)
double precision u,v,p,a,b
c
common / bndinf/ txu(nnx),txv(nnx),tyu(nnx),tyv(nnx),
&                vnoru(nnx),vnorv(nnx),fyu(nnx),fyv(nnx),
&                vup(0:nnx+1),uup(0:nnx),influx
double precision txu,txv,tyu,tyv,vnoru,vnorv,fyu,fyv,vup,uup,
&                influx
c
common / bniinf/ jyu(nnx),jyv(nnx)
integer jyu,jyv
c
common / gridd/ xcrd(0:nnx+1),ycrd(0:ny+1),scalar(nnx,ny),
&                xfree(Mxy,MxSurf),yfree(Mxy,MxSurf)
double precision xcrd,ycrd,scalar,xfree,yfree
c
common / griddx/ xcoord(0:nnx),ycoord(0:ny)
double precision xcoord,ycoord
c
common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,nx2,ny2,nn
double precision Re,RRe,dt,time,dts
c
common /SORpar/ aim(nnx,ny),aip(nnx,ny),ajm(nnx,ny),

```

```

&          ajp(nnx, nny), diag(nnx, nny), f(nnx, nny)
double precision aim, aip, ajm, ajp, diag, f
c
common / bndvinfi/ ip1v(Mxy), jplv(Mxy), ip2v(Mxy), jp2v(Mxy),
&          iinterp(Mxy), jinterp(Mxy), nbndv
integer ip1v, jplv, ip2v, jp2v, iinterp, jinterp, nbndv
c
common / bnduinfi/ ip1u(Mxy), jplu(Mxy), ip2u(Mxy), jp2u(Mxy),
&          iinterpu(Mxy), jinterpu(Mxy), nbndu
integer ip1u, jplu, ip2u, jp2u, iinterpu, jinterpu, nbndu

common / bndvinfR/ wv1(Mxy), wv2(Mxy)
double precision wv1, wv2
c
common / bnduinfR/ wu1(Mxy), wu2(Mxy)
double precision wu1, wu2
c
common /masksx/ pmask(0:nnx, 0:nny), umask(0:nnx, 0:nny),
&          vmask(0:nnx, 0:nny)
double precision pmask, umask, vmask
c
double precision div(nnx, nny), divmx, divmn
c
integer i, j, k
c
common /homeadd/ home
character*40 home
c
c ** f is basically the divergence of (u,v) as calculated in convec
c
divmx = 0.D0
divmn = 100000.D0
imx=0
jmx=0
do i=1, nx
do j=1, ny
div(i, j)=pmask(i, j)*
&          ((u(i, j)-u(i-1, j))/(xcoord(i)-xcoord(i-1))+
&          (v(i, j)-v(i, j-1))/(ycoord(j)-ycoord(j-1)))
c      div(i, j)=pmask(i, j)*
c      &          ((u(i, j)-u(i-1, j))/(xcoord(i)-xcoord(i-1))+
c      &          (v(i, j)-v(i, j-1))/(ycoord(j)-ycoord(j-1)))
      if (ABS(div(i, j)) .GT. divmx) then
          imx=i
          jmx=j
          divmx=ABS(div(i, j))
      end if
      if ((pmask(i, j)).GT.0.5D0) then
          divmn=MIN(divmn, ABS(div(i, j)))
      end if
end do
end do
c
write(*, *) 'Minimum divergence = ', divmn
write(*, '(A, 2I4, 4E12.4)') 'Max. divergence reached at (x,y)= ',
&          imx, jmx,          !xcrd(imx), ycrd(jmx),
&          u(imx, jmx), u(imx-1, jmx), v(imx, jmx), v(imx, jmx-1)
maxdiv=divmx
return
end
cc
cc
subroutine solve(eps, iterat)
c
double precision eps
integer iterat
c
parameter (nnx=600, nny=850, nnxy=nnx*nyy, nnyy=nnx*nyy+nyy)

```

```

c
common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,nx2,ny2,nn
double precision Re, RRe,dt,time,dts

c
common /veloxx/ u(0:nnx,0:nnny+1),v(0:nnx+1,0:nnny),
& p(0:nnx+1,0:nnny+1),a(0:nnx,nnny,4),b(nnx,0:nnny,4)
double precision u,v,p,a,b

c
common /pretemp/ptemp(0:nnx+1,0:nnny+1)
double precision ptemp

c
c
common / griddx/ xcoord(0:nnx), ycoord(0:nnny)
double precision xcoord, ycoord

c
common /indi / li(nnx),maxit

c
common /masksx/ pmask(0:nnx,0:nnny),umask(0:nnx,0:nnny),
& vmask(0:nnx,0:nnny)
double precision pmask,umask,vmask

integer li

c
common /coefs / ae(nnxy),aw(nnxy),an(nnxy),as(nnxy),ap(nnxy),
& fp(nnxy),alfa
double precision ae,aw,an,as,ap,fp,alfa

c
common /ludeco/ un(-nnny:nnxy),ue(-nnny:nnxy),lw(nnxy),
& ls(nnxy),lpr(nnxy)
double precision un,ue,lw,ls,lpr

c
double precision res(-nnny:nnny),res0,resn,rsm

c
common /homeadd/ home
character*40 home
double precision temp1,temp2

c
temp1=0.D0
temp2=0.D0
res = 0.D0

c
C.....CALCULATE RESIDUAL AND AUXILLIARY VECTORS; INNER ITERATION LOOP
C
do n=1,maxit
c
resn=0.D0
do i=1,nx
do j=1,ny
ij=(i-1)*ny+j
res(ij)=pmask(i,j)*(fp(ij)-ap(ij)*p(i,j)-an(ij)*p(i,j+1)-
& as(ij)*p(i,j-1)-ae(ij)*p(i+1,j)-aw(ij)*p(i-1,j))
resn=MAX(res(ij),resn)
res(ij)=(res(ij)-ls(ij)*res(ij-1)-lw(ij)*res(ij-ny))*lpr(ij)
end do
end do

c
c
c
open(unit=12,file='gridp.dat')
c
write(12,*) 'variables="x","y","u","v","p",
c & "ap","an","as","aw","ae"
c write(12,*)
c 'ZONE T="scalar field",I = ',nx,' J = ',ny,' F=BLOCK'
c write(12,'(5E16.8)') ((xcoord(i),i=1,nx),j=1,ny)
c write(12,'(5E16.8)') ((ycoord(j),i=1,nx),j=1,ny)
c write(12,'(5E16.8)') ((p(i,j),i=1,nx),j=1,ny)
c write(12,'(5E16.8)') ((u(i,j),i=1,nx),j=1,ny)

```

```

c      write(12,'(5E16.8)') ((v(i,j),i=1,nx),j=1,ny)
c      write(12,'(5E16.8)') ((ap((i-1)*ny+j),i=1,nx),j=1,ny)
c      write(12,'(5E16.8)') ((an((i-1)*ny+j),i=1,nx),j=1,ny)
c      write(12,'(5E16.8)') ((as((i-1)*ny+j),i=1,nx),j=1,ny)
c      write(12,'(5E16.8)') ((aw((i-1)*ny+j),i=1,nx),j=1,ny)
c      write(12,'(5E16.8)') ((ae((i-1)*ny+j),i=1,nx),j=1,ny)
c      write(12,'(5E16.8)') ((res((i-1)*ny+j),i=1,nx),j=1,ny)

c      close(12)
c      if (n .EQ. 1) res0=resn
c
c.....CALCULATE INCREMENT AND CORRECT VARIABLE
c
c      do i=nx,1,-1
c      do j=ny,1,-1
c          ij=(i-1)*ny+j
c          res(ij)=res(ij)-un(ij)*res(ij+1)-ue(ij)*res(ij+ny)
c          p(i,j)=p(i,j)+pmask(i,j)*res(ij)
c      end do
c      end do

c
c.....CONVERGENCE CHECK
c
c      rsm=resn/(res0+1.D-20)
c      if (mod(n,20) .EQ. 0) then
c          temp2=temp1
c          temp1=resn
c          write(*,*) n,' sweep, res = ',resn
c      end if

c
c      if ((resn .LT. eps) .OR.
c      &      ((abs(temp1-temp2) .LT. 1.0D-8) .AND.
c      &      (n .GT. 500) .AND.
c      &      ((resn .LT. 1.0D-5) .AND. (resn .GT. 1.0D-6))) .OR.
c      &      ((abs(temp1-temp2) .LT. 1.0D-9) .AND.
c      &      (n .GT. 300) .AND.
c      &      (resn .LT. 1.0D-6))) then
c
c          GoTo 100
c      end if

c
c      end do
100 continue

c      pmean=0.D0
c      nn=0
c      do i=1,nx
c      do j=1,ny
c          if (pmask(i,j) .GT. 0.5) then
c              nn=nn+1
c              pmean=pmean+p(i,j)
c          end if
c      end do
c      end do

c
c      pmean=pmean/(1.D0*nn)
c      do i=1,nx
c      do j=1,ny
c          p(i,j)=pmask(i,j)*(p(i,j)-pmean)
c          ptemp(i,j)=p(i,j)
c      end do
c      end do

c
c      return
c      end
cc

subroutine force()

```

```

c
integer nnx, nny, MxSurf, Mxy
parameter (nnx=600, nny=850, MxSurf=50)
parameter (Mxy=2*nnx+2*nny)
c
common /cylzise/ acyl, bcyl, Rcyl
double precision acyl, bcyl, Rcyl
c
double precision w
parameter (w=0.5D0)
c
common /veloxx/ u(0:nnx, 0:nny+1), v(0:nnx+1, 0:nny),
& p(0:nnx+1, 0:nny+1), a(0:nnx, nny, 4), b(nnx, 0:nny, 4)
double precision u, v, p, a, b
c
common /rkcom / urk(0:nnx, 0:nny+1), vrk(0:nnx+1, 0:nny),
& ar(10, 10), br(10), nrk
double precision urk, vrk, ar, br
integer nrk
c
common /masksx/ pmask(0:nnx, 0:nny), umask(0:nnx, 0:nny),
& vmask(0:nnx, 0:nny)
double precision pmask, umask, vmask
common /gridd/ xcrd(0:nnx+1), ycrd(0:nny+1), scalar(nnx, nny),
& xfree(Mxy, MxSurf), yfree(Mxy, MxSurf)
double precision xcrd, ycrd, scalar, xfree, yfree
c
common /griddx/ xcoord(0:nnx), ycoord(0:nny)
double precision xcoord, ycoord
c
common /bndvinfi/ ip1v(Mxy), jp1v(Mxy), ip2v(Mxy), jp2v(Mxy),
& iinterpv(Mxy), jinterpv(Mxy), nbndv
integer ip1v, jp1v, ip2v, jp2v, iinterpv, jinterpv, nbndv
c
common /bnduinfi/ ip1u(Mxy), jp1u(Mxy), ip2u(Mxy), jp2u(Mxy),
& iinterpu(Mxy), jinterpu(Mxy), nbndu
integer ip1u, jp1u, ip2u, jp2u, iinterpu, jinterpu, nbndu

common /bndvinfR/ wv1(Mxy), wv2(Mxy)
double precision wv1, wv2
c
common /bnduinfR/ wu1(Mxy), wu2(Mxy)
double precision wu1, wu2
c
common /bndpinfi/ ip1p(0:Mxy), jp1p(0:Mxy), ip2p(0:Mxy), jp2p(0:Mxy),
& ip3p(0:Mxy), jp3p(0:Mxy), iinterpp(0:Mxy), jinterpp(0:Mxy),
& nbndp
integer ip1p, jp1p, ip2p, jp2p, ip3p, jp3p, iinterpp, jinterpp, nbndp
c
common /bndpinfR/ teta(0:Mxy), unitvi(0:Mxy), unitvj(0:Mxy),
& wp1(0:Mxy), wp2(0:Mxy), delta1(0:Mxy)
double precision teta, unitvi, unitvj, wp1, wp2, delta1
c
common /dimenx/ nx, ny, Re, RRe, dt, time, dts, nx2, ny2, nn
integer nx, ny, nx2, ny2, nn, nx3, nn1
double precision Re, RRe, dt, time, dts
c
common /FCDC/ FL, FD, liftf, dragf, liftp, dragp, pres, FLift, Fdrag, Fa
& FL2, FD2
double precision FL, FD, liftf, dragf, liftp, dragp, pres, FLift, Fdrag, Fa
& FL2, FD2
c
integer i, j, k
c
common /sbody/ vsolid, ysolid, usolid, xsolid, aysolid, axsolid
double precision vsolid, ysolid, usolid, xsolid, aysolid, axsolid

double precision dliftf, dliftp, ddragf, ddragp, dpres,
& dliftptemp, ddragptemp

```

```

    double precision dliftf2,dliftp2,ddragf2,ddragp2,beta,
& liftf2,liftp2,dragf2,dragp2
c
c
    common/FaVYold/FLn,FLold,aysolidn,vsolidn,ysolidn
    double precision FLn,FLold,aysolidn,vsolidn,ysolidn
c
    double precision utan,utan0,unorm,lift,drag,psurface
    double precision Fy1,Fy2,Fy3,Fy4,Fy5,Fx1,Fx2,Fx3,Fx4,Fx5,Fbflux
c
    common /homeadd/ home
    character*40 home
c
    write(*,*)'force subroutine'
    ddragf=0.D0
    dliftf=0.D0
    ddragp=0.D0
    dliftp=0.D0
c
    ddragf2=0.D0
    dliftf2=0.D0
    ddragp2=0.D0
    dliftp2=0.D0
    liftf2=0.D0
    dragf2=0.D0
    liftp2=0.D0
    dragp2=0.D0
c
    dpres=0.D0
    liftf=0.D0
    dragf=0.D0
    liftp=0.D0
    dragp=0.D0
    pres=0.D0
    Fy1=0.D0
    Fy2=0.D0
    Fy3=0.D0
    Fy4=0.D0
    Fy5=0.D0
    FLift=0.D0
    Fx1=0.D0
    Fx2=0.D0
    Fx3=0.D0
    Fx4=0.D0
    Fx5=0.D0
    Fdrag=0.D0
    Fa=0.D0
    Fbflux=0.D0
    FLold=FL
c
C*****new method of calculation of force*****

    nx1=37
    nx12=197
    ny1=37
    ny12=197
    do i=nx1,nx12
    do j=ny1,ny12
    if ((sqrt((xcrd(i)-acyl)**2+(ycoord(j)-bcyl)**2) .GE. Rcyl) .AND.
& (sqrt((xcrd(i+1)-acyl)**2+(ycoord(j)-bcyl)**2) .GE. Rcyl)) then
    if ((j .LT. ny12) .AND. (i .LT. nx12)) then
    Fy1=Fy1+0.5*(v(i,j)+v(i+1,j)-vrk(i,j)-vrk(i+1,j))*
& (xcrd(i+1)-xcrd(i))*(ycrd(j+1)-ycrd(j))/dt
    Fa=Fa+aysolid*(xcrd(i+1)-xcrd(i))*(ycrd(j+1)-ycrd(j))
    end if
    if ((i .EQ. nx1) .AND. (j .LT. ny12)) then
    Fy2=Fy2+(-1*v(i,j)*0.25*(u(i,j)+u(i-1,j)+u(i-1,j+1)+u(i,j+1)))+ !v(-u)

```

```

&      RRe*((v(i+1,j)-v(i-1,j))/(xcrd(i+1)-xcrd(i-1)))+0.5*!t21
&(((u(i-1,j+1)+u(i,j+1)-u(i-1,j)-u(i,j))/(ycrd(j+1)-ycrd(j))))*!t12
&      (ycrd(j+1)-ycrd(j))          !deltay
      end if
      if ((i .EQ. nx12) .AND. (j .LT. ny12)) then
      Fy3=Fy3+(v(i,j)*0.25*(u(i,j)+u(i-1,j)+u(i-1,j+1)+u(i,j+1))-
&      RRe*((v(i+1,j)-v(i-1,j))/(xcrd(i+1)-xcrd(i-1)))+0.5*!-t21
&(((u(i-1,j+1)+u(i,j+1)-u(i-1,j)-u(i,j))/(ycrd(j+1)-ycrd(j))))*!t12
&      (ycrd(j+1)-ycrd(j))
      end if
      if ((j .EQ. ny12) .AND. (i .LT. nx12)) then
      Fy4=Fy4+((0.25*(v(i,j)+v(i+1,j)+v(i+1,j-1)+v(i,j-1)))**2+ !vv
&      0.5*(p(i,j)+P(i+1,j))-
c      &((( -2/3)*RRe*((u(i+1,j)-u(i-1,j))/(xcoord(i+1)-xcoord(i-1)))+0.5*
c      &(v(i,j)+v(i+1,j)-v(i-1,j)-v(i-1,j+1))/(ycoord(j)-ycoord(j-1))))+
&      2*RRe*0.5*(v(i,j)+v(i+1,j)-v(i,j-1)-v(i,j-1))/
&      (ycoord(j)-ycoord(j-1))) *
&      (xcrd(i+1)-xcrd(i))
      end if
      if ((j .EQ. ny1) .AND. (i .LT. nx12)) then
      Fy5=Fy5+(-1*(0.25*(v(i,j)+v(i+1,j)+v(i+1,j-1)+v(i,j-1)))**2- !-vv
&      0.5*(p(i,j)+P(i+1,j))+
c      &((( -2/3)*RRe*((u(i+1,j)-u(i-1,j))/(xcoord(i+1)-xcoord(i-1)))+0.5*
c      &(v(i,j)+v(i+1,j)-v(i-1,j)-v(i-1,j+1))/(ycoord(j)-ycoord(j-1))))+
&      2*RRe*0.5*(v(i,j)+v(i+1,j)-v(i,j-1)-v(i+1,j-1))/
&      (ycoord(j)-ycoord(j-1))) *
&      (xcrd(i+1)-xcrd(i))
      end if
      end if
      end do
      end do

c
c      do nbnd=1,nbndv
c          i=iinterpv(nbnd)
c          j=jinterpv(nbnd)
c          if (pmask(i,j+1)+pmask(i,j) .EQ. 1) then
c              if (pmask(i,j+1) .EQ. 1.D0) then!top of the cylinder
c                  Fbflux=Fbflux-v(i,j)*v(i,j)*(xcoord(i)-xcoord(i-1))
c                  write(*,*)'i,j,top,-vv,Fbflux',i,j,
c          & -1*v(i,j)*v(i,j)*(xcoord(i)-xcoord(i-1)),Fbflux
c                  else if (pmask(i,j) .EQ. 1.D0) then!bottom of the cylinder
c                  Fbflux=Fbflux+v(i,j)*v(i,j)*(xcoord(i)-xcoord(i-1))
c                  write(*,*)'i,j,bot,vv,Fbflux',i,j,
c          & v(i,j)*v(i,j)*(xcoord(i)-xcoord(i-1)),Fbflux
c                  end if
c              end if
c          end do

cc
c      do nbnd=1,nbndu
c          i=iinterpu(nbnd)
c          j=jinterpu(nbnd)
c          if (pmask(i+1,j)+pmask(i,j) .EQ. 1) then
c              if (pmask(i+1,j) .EQ. 1.D0) then !right
c                  Fbflux=Fbflux-u(i,j)*(0.5*(v(i+1,j)+v(i+1,j-1)))*
c          &      (ycoord(j)-ycoord(j-1))
c                  write(*,*)'i,j,right,uv,Fbflux',i,j,
c          & u(i,j)*(-0.5*(v(i,j)+v(i,j)))*(ycoord(j)-ycoord(j-1)),Fbflux
c                  else if (pmask(i,j) .EQ. 1.D0) then !left
c                  Fbflux=Fbflux-u(i,j)*(0.5*(v(i,j)+v(i,j-1)))*
c          &      (ycoord(j)-ycoord(j-1))
c                  write(*,*)'i,j,left,uv,Fbflux',i,j,
c          & u(i,j)*(-0.5*(v(i,j)*v(i,j-1)))*(ycoord(j)-ycoord(j-1)),Fbflux
c                  end if
c              end if
c          end do

cc
      FLift=Fy1+Fy2+Fy3+Fy4+Fy5+Fa!+Fbflux

```



```

open(unit=12,file='FLlift.dat',
&           position='append')
write(12,'(8F15.6)')time,FLlift,Fy1,Fy2,Fy3,Fy4,Fy5,Fa!,Fbflux
close(12)
*****
do i=nx1,nx12
do j=ny1,ny12
if ((sqrt((xcoord(i)-acyl)**2+(ycrd(j)-bcyl)**2) .GE. Rcyl) .AND.
& (sqrt((xcoord(i)-acyl)**2+(ycrd(j+1)-bcyl)**2) .GE. Rcyl)) then
if ((j .LT. ny12) .AND. (i.LT. nx12)) then
Fxl=Fxl+0.5*(u(i,j)+u(i,j+1)-urk(i,j)-urk(i,j+1))*
& (xcrd(i+1)-xcrd(i))*(ycrd(j+1)-ycrd(j))/dt
end if
if ((i .EQ. nx1) .AND. (j .LT. ny12)) then
Fxl=Fxl+(-1*(0.25*(u(i,j)+u(i-1,j)+u(i-1,j+1)+u(i,j+1)))*2- !u(-u)
& 0.5*(p(i,j)+p(i,j+1))+
& 2*RRe*0.5*(u(i,j+1)+u(i,j)-u(i-1,j)-u(i-1,j+1)))/
& (xcrd(i+1)-xcrd(i)))*!t12
& (ycrd(j+1)-ycrd(j)) !deltay
end if
if ((i .EQ. nx12) .AND. (j .LT. ny12)) then
Fxl=Fxl+((0.25*(u(i,j)+u(i-1,j)+u(i-1,j+1)+u(i,j+1)))*2+
& 0.5*(p(i,j)+p(i,j+1))-
& 2*RRe*0.5*(u(i,j+1)+u(i,j)-u(i-1,j)-u(i-1,j+1)))/
& (xcrd(i+1)-xcrd(i)))*!t12
& (ycrd(j+1)-ycrd(j))
end if
if ((j .EQ. ny12) .AND. (i .LT. nx12)) then
Fxl=Fxl+(u(i,j)*(0.25*(v(i,j)+v(i+1,j)+v(i+1,j-1)+v(i,j-1)))- !uv
& (RRe*(0.5*(v(i+1,j)+v(i+1,j-1)-v(i,j)-v(i,j-1)))/
& (xcrd(i+1)-xcrd(i))+
& (u(i,j+1)-u(i,j-1))/(ycrd(j+1)-ycrd(j-1)))))*
& (xcrd(i+1)-xcrd(i))
end if
if ((j .EQ. ny1) .AND. (i .LT. nx12)) then
Fxl=Fxl+((-1*u(i,j)*0.25*(v(i,j)+v(i+1,j)+v(i+1,j-1)+v(i,j-1)))+ !-uv
& RRe*((0.5*(v(i+1,j)+v(i+1,j-1)-v(i,j)-v(i,j-1)))/
& (xcrd(i+1)-xcrd(i))+
& (u(i,j+1)-u(i,j-1))/(ycrd(j+1)-ycrd(j-1)))))*
& (xcrd(i+1)-xcrd(i))
end if
end if
end do
end do
Fdrag=Fxl+Fxl2+Fxl3+Fxl4+Fxl5
open(unit=12,file='Fdrag.dat',
&           position='append')
write(12,'(9F15.6)')time,Fdrag, Fxl,Fxl2,Fxl3,Fxl4,Fxl5
close(12)
*****
beta=ATAN(vsolid)
write(*,*)'beta,vsolid',beta,vsolid
ccc open(unit=12,file='degree.dat',
ccc &           position='append')
c rewind(12)
do k=1,nbndp
c
psurface=(1-wp2(k))*p(ip1p(k),jp1p(k))+
& wp2(k)*(wp1(k)*p(ip2p(k),jp2p(k))+(1-wp1(k))*p(ip3p(k),jp3p(k)))
c
utan=-((u(ip1p(k),jp1p(k))+u(ip1p(k)-1,jp1p(k)))/2)*sin(teta(k))
& ((v(ip1p(k),jp1p(k))+v(ip1p(k),jp1p(k)-1))/2)*cos(teta(k))
c utan0=-usolid*sin(teta(k))+vsolid*cos(teta(k)) !added on 11/5/13
utan0=0 ! as the reference frame is on the cylinder
c
dliftf=2*RRe*((utan-utan0)/deltal(k))*(cos(teta(k)))*0.5*
& (0.5*ABS(teta(k-1)-teta(k+1)))
ddragf=2*RRe*((utan-utan0)/deltal(k))*(-1*sin(teta(k)))*0.5*

```

```

&      (0.5*ABS(teta(k-1)-teta(k+1)))
dliftp=2*psurface*(-1*sin(teta(k)))*0.5*
&      (0.5*ABS(teta(k-1)-teta(k+1)))
ddragp=2*psurface*(-1*cos(teta(k)))*0.5*
&      (0.5*ABS(teta(k-1)-teta(k+1)))
dpres=p(ip1p(k),jp1p(k))*0.5*(0.5*ABS(teta(k-1)-teta(k+1)))
c
dliftf2=2*RRe*((utan-utan0)/delta1(k))*(cos(teta(k)+beta))*0.5*
&      (0.5*ABS(teta(k-1)-teta(k+1)))
ddragf2=2*RRe*((utan-utan0)/delta1(k))*(-1*sin(teta(k)+beta))*
&      0.5*(0.5*ABS(teta(k-1)-teta(k+1)))
dliftp2=2*p(ip1p(k),jp1p(k))*(-1*sin(teta(k)+beta))*0.5*
&      (0.5*ABS(teta(k-1)-teta(k+1)))
ddragp2=2*p(ip1p(k),jp1p(k))*(-1*cos(teta(k)+beta))*0.5*
&      (0.5*ABS(teta(k-1)-teta(k+1)))
c      write(*,*) 'dLp,dLp2,dLf,dLf2',dliftp,dliftp2,dliftf,dliftf2,beta
c
      liftf2=liftf2+dliftf2
      dragf2=dragf2+ddragf2
      liftp2=liftp2+dliftp2
      dragp2=dragp2+ddragp2
c
      liftf=liftf+dliftf
      dragf=dragf+ddragf
      liftp=liftp+dliftp
      dragp=dragp+ddragp
      pres=pres-dpres
c      write(*,*) 'nbndp',nbndp
ccc      write(12,'(8F15.6)')teta(k),p(ip1p(k),jp1p(k)),dpres,
ccc &      2*RRe*((utan-utan0)/delta1(k))*(cos(teta(k))),
ccc &      2*RRe*((utan-utan0)/delta1(k))*(-1*sin(teta(k))),
ccc &      2*RRe*((utan-utan0)/delta1(k)),
ccc &      2*p(ip1p(k),jp1p(k))*(-1*sin(teta(k))),
ccc &      2*p(ip1p(k),jp1p(k))*(-1*cos(teta(k)))
end do
ccc      close(12)
      FL2=0.5*(liftf2+liftp2)
      FD2=0.5*(dragf2+dragp2)
      FL=0.5*(liftf+liftp)
      FD=0.5*(dragf+dragp)
      write(*,*) 'FL,FL2,lf,lf2,lp,lp2',FL,FL2,liftf,dliftf2,liftp,dliftp2
c      STOP
      return
      end
c
cc
      subroutine forc vib()

      integer nnx, nny, MxSurf, Mxy
      parameter (nnx=600, nny=850, MxSurf=50)
      parameter (Mxy=2*nnx+2*ny)
c
      double precision w
      parameter (w=0.5D0)
c
      common /veloxx/ u(0:nnx,0:ny+1),v(0:nnx+1,0:ny),
&      p(0:nnx+1,0:ny+1),a(0:nnx,ny,4),b(nnx,0:ny,4)
      double precision u,v,p,a,b
c
      common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
      integer nx,ny,nx2,ny2,nn
      double precision Re, RRe,dt,time,dts
c
      common /rkcom / urk(0:nnx,0:ny+1), vrk(0:nnx+1,0:ny),
&      ar(10,10), br(10), nrk
      double precision urk, vrk, ar, br
      integer nrk

```

```

c      common /masksx/ pmask(0:nnx,0:nny),umask(0:nnx,0:nny),
&      vmask(0:nnx,0:nny)
      double precision pmask,umask,vmask
      double precision, Dimension(0:nnx,0:nny):: umaskt,vmaskt
c
      common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
      double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
      double precision vsolidtemp, xsolidtemp, usolidtemp, ysolidtemp
&      sstiff,smass,sdamping, fst,Fco,omega

      common /FCDC/ FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
c      &      ,FL2,FD2
      double precision FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
c      &      ,FL2,FD2
c      integer i,j,k
c
      common /epsili/epstemp
      double precision epstemp
c
      common /homeadd/ home
      character*40 home
c      *****
cc
      fst= 0.167 ! strouhal number
      Fco=1.05D0
      omega= 2*(4*Atan(1.D0))*Fco* fst
      ysolid= 0.2 * sin(omega*time)
      vsolid= 0.2 * omega * cos(omega*time)
      aysolid=-0.2 * omega * omega * sin(omega*time)
c      vmaskt=vmask
c      umaskt=umask
c      call interpolate()
c      call inisol()
c      vmaskt=vmask-vmaskt
c      umaskt=umask-umaskt
c      if ((sum(umaskt) .EQ. 0 ) .AND.
c      &      (sum(vmaskt) .EQ. 0 ))
c      &      goto 18
c      call inisol()
c      epstemp=5.0D-7
c      do i=1,10
c      u=urk
c      v=vrk
c      call convec()
c      call fillf()
c      call calcuv()
c      end do
c      write(*,*)'sum(vmask-vmaskt)', sum (vmaskt)
c      STOP

c18      continue
      epstemp =5.0D-7

cc      open(unit=12,file='vysolid.dat',
cc      &      position='append')
cc      write(12,'(3E16.8)') time, vsolid, ysolid
cc      close(12)
      return
      end

      subroutine structuremain
c
      integer nnx, nny, MxSurf, Mxy
      parameter (nnx=600, nny=550, MxSurf=50)
      parameter (Mxy=2*nnx+2*nny)

```

```

C
C   integer, intent(inout)::ksub
C   double precision w
C   parameter      (w=0.5D0)
C
C   common /veloxx/ u(0:nnx,0:nny+1),v(0:nnx+1,0:nny),
&                p(0:nnx+1,0:nny+1),a(0:nnx,nny,4),b(nnx,0:nny,4)
C   double precision u,v,p,a,b
C
C   common /velotemp/ utemp(0:nnx,0:nny+1),vtemp(0:nnx+1,0:nny)
C   double precision utemp,vtemp
C
C   common /pretemp/ptemp(0:nnx+1,0:nny+1)
C   double precision ptemp
C
C   common / dimenx/ nx,ny,Re,RRe,dt,time,dts
C   integer nx,ny
C   double precision Re, RRe,dt,time,dts
C
C   common /rkcom / urk(0:nnx,0:nny+1), vrk(0:nnx+1,0:nny),
&                ar(10,10), br(10), nrk
C   double precision urk, vrk, ar, br
C   integer nrk
C
C   common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
C   double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
C   double precision xsolidn, usolidn,
&                sstiff, smass, sdamping
C   double precision vsolidtemp0, ysolidtemp0,
&                vs05ns, ys05ns, vs05nss, ys05nss, vslns, yslns
&                fn, fn05s, fn05ss, fnlns
C   double precision eps1, eps2, eps
C   common /FCDCL/ FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
C   double precision FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
C   double precision eta, mratio,Vr,PI,CLift
C
C   common/FaVYold/FLn,FLold,aysolidn,vsolidn,ysolidn
C   double precision FLn,FLold,aysolidn,vsolidn,ysolidn
C   double precision coeff0, coeff1
C   double precision vksub,dvksub,dvksub0,yksub,dyksub,dyksub0
C   double precision alfa,landa,landav,small
C   double precision ytemp,vstemp,ytemp0,vtemp0,aytemp
C   integer i,j,k,l
C
C   common /homeadd/ home
C   character*40 home
C   *****
C   ***** non-dimensional format of structure*****
C   eps=0.000001 ! convergence cirterion
C   small=1e-20
C   PI=4.D0*ATAN(1.D0)
C   eta=0.0012D0 ! damping ration, eta=C/Cc=C/(2(km)^0.5) 20/4/14
C   mratio=149.10 !(4/PI)*2 ! mass ratio=msolid/mfluid 23/4/14
C   Vr=5.58 !at Re=100 radius velocity=U/(Fn.D) 20/4/14
C
C   ysolidn=ysolid !to record inital value at time n
C   vsolidn=vsolid
C   write(*,*)'yn1,vn,ysolid,vsolid', ysolidn,vsolidn,ysolid,vsolid
C   FLn=FL
C   *****
C   Start outter iteration
C   *****
C
C   l=1
101 continue
C***** start flow updating *****

```

```

        call convec
        call fillf
        call calcuv
        call force
        write(*,*)'time,l,FLn,FL',time,l,FLn,FL
c***** end flow updating *****
        k=1
113      continue
        dyksub0=dyksub
        dvksub0=dvksub !14/04/14
        yksub=ysolid ! to record inital value at k
        vksub=vsolid
c
c***** start solving the structure *****
c
        ysolid=ysolidn+0.5*dt*(vsolid+vsolidn)
        aysolid=-2*eta*(2*PI/Vr)*vsolid-
&          ((2*PI/Vr)**2)*ysolid+
&          2*(2*FL)/(PI*mratio) ! Clift=2*FL
        vsolid=vsolidn+dt*aysolid
c
        write(*,'(A,I5,5E16.4)')'k1,time,ysolidn,ysolid,vsolidn,vsolid',
&k,time,ysolidn,ysolid,vsolidn,vsolid
c
        if (((k .LE. 5) .OR.
&          (abs(ysolid-yksub) .GT. eps)) .AND.
&          (k .LT. 15)) then
            k=k+1
            Go to 113
        end if
c        write(*,*)'ytemp,vstemp,aytemp',time,k,ytemp,vstemp,aytemp
c
c***** end of solving structure equation*****
c
c        if (k .EQ. 1) then
c            landa=0.3
c            landav=0.3
c        else
c
c            dvksub=vksub -vsolid !14/04/14
c            dyksub=yksub -ysolid
c            landav= landav+(landav-1)*
&              (dvksub0-dvksub)*dvksub/((dvksub0-dvksub)**2+small)
c
c            landa= landa+(landa-1)*
&              (dyksub0-dyksub)*dyksub/((dyksub0-dyksub)**2+small)
c        write(*,('A,I5,6E15.6'))'k,time,dyksub,yksub,dyksub0,ysolid',
c        &k,time,dyksub,yksub,dyksub0,ysolid,landa
c        end if
c        ysolid=landa*ysolidn+(1-landa)*ysolid
c        vsolid=landa*vsolidn+(1-landa)*vsolidn
c        write(*,*)'landa,ysolid,vsolid',time,k,landa,landav,
c        & ysolid,vsolid
c
c        write(*,*)'ysolidn,vsn,aysn',time,k,ysolidn,vsolidn,aysolidn
c
c        write(*,*)'ytemp,vstemp,aytemp',time,k,ytemp,vstemp,aytemp
c
c        if (((abs(ysolid-yksub) .GT. eps).OR.!23/4/14
&          (k .LT. 5)) .AND.
&          (k .LT. 15)) then
c
c        write(*,*)' *****No. of outter-iteration,ksub=',l
c        l=l+1
cc        call interpolate ()
cc        call inisol ()

```

```

c
do j=0,ny+1
do i=0,nx
  utemp(i,j)=u(i,j)! to use in bounds to update boundaries
  u(i,j) = urk(i,j) !to start fluid solver from time n
end do
end do

c
do j=0,ny
do i=0,nx+1
  vtemp(i,j)=v(i,j) !to use in bounds to update boundaries
  v(i,j) = vrk(i,j) !to start fluid solver from time n
end do
end do

c
      goto 101
    end if
c*****
c      end of outer iteration
c *****
c
      open(unit=12,file='pld1.dat',
& position='append')
& write(12,'(10E15.6)')time,pres,liftp,liftf,dragp,dragf,FL,FD,
& vsolid,ysolid
      close(12)

c
      return
    end

c
      subroutine solidsolver()

c
      integer nnx,ny,MxSurf,Mxy
      parameter (nnx=600,ny=550,MxSurf=50)
      parameter (Mxy=2*nnx+2*ny)

c
      integer, intent(inout)::ksub
      double precision w
      parameter (w=0.5D0)

c
      common /veloxx/ u(0:nnx,0:ny+1),v(0:nnx+1,0:ny),
& p(0:nnx+1,0:ny+1),a(0:nnx,ny,4),b(nnx,0:ny,4)
      double precision u,v,p,a,b

c
      common /pretemp/ptemp(0:nnx+1,0:ny+1)
      double precision ptemp

c
      common / dimenx/ nx,ny,Re,RRe,dt,time,dts
      integer nx,ny
      double precision Re, RRe,dt,time,dts

c
      common /rkcom / urk(0:nnx,0:ny+1), vrk(0:nnx+1,0:ny),
& ar(10,10), br(10), nrk
      double precision urk, vrk, ar, br
      integer nrk

c
      common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
      double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
      double precision xsolidn, usolidn
      double precision sstiff,smass,sdamping
      double precision vsolidtemp0, ysolidtemp0,FLtemp0
& vs05ns,ys05ns,vs05nss,ys05nss,vs1ns,ys1ns
& fn,fn05s,fn05ss,fn1ns
      double precision eps1,eps2
      common /FCDC/ FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
      double precision FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
      double precision eta, mratio,Vr,PI,CLift

```

```

C
common/FaVYold/FLn,FLold,aysolidn,vsolidn,ysolidn
double precision FLn,FLold,aysolidn,vsolidn,ysolidn
double precision ytemp,vtemp,ytemp0,vtemp0,small,aytemp
integer i,j,k

C
common /homeadd/ home
character*40 home
C
*****
C
small=1e-20
PI=4.D0*ATAN(1.D0)
eta=0.0D0 ! damping ration, eta=C/Cc=C/(2(km)^0.5) 20/4/14
mratio=(4/PI)*2 ! mass ratio=msolid/mfluid 23/4/14
Vr=8 !at Re=100 radious velocity=U/(Fn.D) 20/4/14

C
Call force()

C
C
write(*,*)'solid solver,vtemp',vtemp
aysolidn=-2*eta*(2*PI/Vr)*vsolidn-
& ((2*PI/Vr)**2)*ysolidn+
& 2*(2*FLn)/(PI*mratio) ! Cliftn=2*FLn

C
ytemp=ysolidn+dt * vsolidn
vtemp=vsolidn+dt * aysolidn

C
aytemp=-2*eta*(2*PI/Vr)*vsolidn-
& ((2*PI/Vr)**2)*ysolidn+
& 2*(2*FL)/(PI*mratio) ! Clift=2*FL

C
ysolid=ysolidn+0.5*dt*(vsolidn+vtemp)
write(*,*)'here',ysolid,ysolidn,vsolidn,vtemp,dt
vsolid=vsolidn+0.5*dt*(aysolidn+aytemp)
C
write(*,*)'ysolidn,vsolidn,aysolidn',time,k,ysolidn,vsolidn,aysolidn
C
write(*,*)'ytemp,vtemp,aytemp',time,k,ytemp,vtemp,aytemp
C
return
end

C
subroutine structure(ksub)
C
integer nnx,ny,MxSurf,Mxy
parameter (nnx=600,ny=550,MxSurf=50)
parameter (Mxy=2*nnx+2*ny)
C
integer, intent(inout)::ksub
double precision w
parameter (w=0.5D0)
C
common /veloxx/ u(0:nnx,0:ny+1),v(0:nnx+1,0:ny),
& p(0:nnx+1,0:ny+1),a(0:nnx,ny,4),b(nnx,0:ny,4)
double precision u,v,p,a,b
C
common /pretemp/ptemp(0:nnx+1,0:ny+1)
double precision ptemp
C
common /dimenx/ nx,ny,Re,RRe,dt,time,dts
integer nx,ny
double precision Re, RRe,dt,time,dts
C
common /rkcom / urk(0:nnx,0:ny+1), vrk(0:nnx+1,0:ny),
& ar(10,10), br(10), nrk
double precision urk, vrk, ar, br
integer nrk

```

```

c
common /sbody/ vsolid, ysolid, usolid, xsolid, aysolid, axsolid
double precision vsolid, ysolid, usolid, xsolid, aysolid, axsolid
double precision xsolidn, usolidn,
&          sstiff, smass, sdamping
double precision vsolidtemp0, ysolidtemp0,
&          vs05ns, ys05ns, vs05nss, ys05nss, vs1ns, ys1ns
&          fn, fn05s, fn05ss, fn1ns
double precision eps1, eps2
common /FCDC/ FL, FD, liftf, dragf, liftp, dragp, pres, FLift, Fdrag, Fa
double precision FL, FD, liftf, dragf, liftp, dragp, pres, FLift, Fdrag, Fa
double precision eta, mratio, Vr, PI, CLift

c
common /FaVYold/ FLn, FLold, aysolidn, vsolidn, ysolidn
double precision FLn, FLold, aysolidn, vsolidn, ysolidn
double precision coeff0, coeff1
integer i, j, k

c
common /homeadd/ home
character*40 home

c
*****
c ***** non-dimensional format of structure*****
eta=0.D0 ! damping ration, eta=C/Cc=C/(2(km)^0.5) 6/2/14
mratio=2 ! mass ratio=msolid/mfluid 6/2/14
Vr=3 !at Re=100 radious velocity=U/(Fn.D) 6/2/14
PI=4.D0*ATAN(1.D0)
c CLift=2*FL ! lift coefficient

cc sstiff = 6.05 !1.1 ! strouhal number=0.167 then f=2*3.14*sqrt
cc smass = 5.0D0 ! f= (1/2*3.14)*sqrt (k/m)
cc sdamping=5.5 ! damping ratio=0.5
c critical damping=2*sqrt(km)
cc
c state space for the solid,
c  $d^2x/dt^2 + (c/m) dx/dt + (k/m)x = CL * (1/2) * density * v^2$ 
c  $v = dx/dt$ 
c  $dv/dt = CL * (1/2) * density * v^2 - (c/m)v - (k/m)x$ 
c
c
c if (ksub .EQ. 0) then
coeff0=3/2
FL=coeff0*FLn+(1-coeff0)*FLold
CLift=2*FL ! 6/2/2014
end if

c
cc coeff1=1+(sdamping/smass)*0.5*dt+(sstiff/smass)*0.25*dt*dt
cc aysolid=
cc & (FL /smass)-
cc & (sdamping/smass)*(vsolidn+0.5*dt*aysolidn)/coeff1 -
cc & (sstiff/smass)*(ysolidn+dt*vsolidn+0.25*dt*dt*aysolidn)/coeff1
aysolid=
& 2*CLift/(PI*mratio)-
& 2*eta*(2*PI/Vr)*(vsolidn+0.25*dt*aysolidn)-
& ((2*PI/Vr)**2)*(ysolidn+dt*vsolidn+0.25*dt*dt*aysolidn)

c
vsolid=vsolidn+0.5*dt*(aysolidn+aysolid)
ysolid=ysolidn+0.5*dt*(vsolidn+vsolid)

c
call interpolate()
call inisol()

c
return
end

c
c
c subroutine convergence(ksub)
c
integer nnx, nny, MxSurf, Mxy

```



```

parameter (nnx=600, nny=550, MxSurf=50)
parameter (Mxy=2*nnx+2*nny)
c
integer, intent(inout)::ksub
double precision w
parameter (w=0.5D0)
c
common /veloxx/ u(0:nnx,0:nny+1),v(0:nnx+1,0:nny),
& p(0:nnx+1,0:nny+1),a(0:nnx,nny,4),b(nnx,0:nny,4)
double precision u,v,p,a,b
c
common /pretemp/ptemp(0:nnx+1,0:nny+1)
double precision ptemp
c
common /dimenx/ nx,ny,Re,RRe,dt,time,dts
integer nx,ny
double precision Re, RRe,dt,time,dts
c
common /rkcom / urk(0:nnx,0:nny+1), vrk(0:nnx+1,0:nny),
& ar(10,10), br(10), nrk
double precision urk, vrk, ar, br
integer nrk
c
common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
double precision xsolidn, usolidn
double precision sstiff,smass,sdamping
double precision vsolidtemp0, ysolidtemp0,FLtemp0
& vs05ns,ys05ns,vs05nss,ys05nss,vslns,yslns
& fn,fn05s,fn05ss,fnlns
double precision eps1,eps2
common /FCDC/ FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
double precision FL,FD,liftf,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
c
common/FaVYold/FLn,FLold,aysolidn,vsolidn,ysolidn
double precision FLn,FLold,aysolidn,vsolidn,ysolidn
integer i,j,k
c
common /homeadd/ home
character*40 home
c
*****
eps1=0.005 ! should relate to mesh size
eps2=0.0001
c
c vsolidn=vsolid
c ysolidn=ysolid
c
c vsolidtemp0=vsolid
c ysolidtemp0=ysolid
c***** forth order Rung-Kutta calculation of structure****
cc fn=(CL/smash-(sdamping/smash)*vsolidn-(sstiff/smash)*ysolidn)
cc vs05ns=vsolidn+0.5*dts*fn ! above CL is total lift force
(0.5*CL*1*1**2
cc ys05ns=ysolidn+0.5*dts*vsolidn
cc
cc fn05s=(CL/smash-(sdamping/smash)*vs05ns-(sstiff/smash)*ys05ns)
cc vs05nss=vsolidn+0.5*dts*fn05s ! above CL is total lift force
(0.5*CL*1*1**2
cc ys05nss=ysolidn+0.5*dts*vs05ns
cc
cc fn05ss=(CL/smash-(sdamping/smash)*vs05nss-(sstiff/smash)*ys05nss)
cc vslns=vsolidn+dts*fn05ss
cc yslns=ysolidn+dts*vs05nss
cc
cc fnlns=(CL/smash-(sdamping/smash)*vslns-(sstiff/smash)*yslns)
cc vsolid=vsolidn+(1.0D0/6)*dts*(fn+2*fn05s+2*fn05ss+fnlns)
cc ysolid=ysolidn+(1.0D0/6)*dts*(vs05ns+2*vs05nss+2*vslns+vsolid)

```

```

cc      write(*,*) CL
c      write(*,*) fn, fn05ns, fn05nss,fnlns
c      write(*,*) vs05ns, vs05nss, vslns
c      write(*,*) ysn05ns, ys05nss,yslns
cc

17      FLtemp0=FL
        Call force()
c
        write(*,*)'convergence subroutine ksub=',ksub
        write(*,*)'FL,FLtemp0', FL, FLtemp0,abs((FL-FLtemp0)/FL)
        if ((ksub .LE. 10) .AND.
&          (abs((FL-FLtemp0)/FL) .GT. eps1)) then
c
        ksub=ksub+1
        write(*,*)' *****No. of sub-iteration, Ksub=',ksub
        write(*,*)'abs((FLn+1 -FLn+1old)/FLn+1)',
&          abs((FL-FLtemp0)/FL)
c
c *****Starting outer iteration for creating strong
c *****coupleing between the structure and fluid
c***** at the same time step with the same initial
c ***** velocity, but with the new position and velicity of
c ***** structure
        do j=0,ny+1
        do i=0,nx
            u(i,j) = urk(i,j)
        end do
        end do
c
        do j=0,ny
        do i=0,nx+1
            v(i,j) = vrk(i,j)
        end do
        end do
c
        call structure(ksub)
c
        call convec
        call Fillf
        call calcuv
        go to 17
c
        else
            aysolidn=aysolid
            vsolidn =vsolid
            ysolidn =ysolid
            FLold   =FLn
            FLn     =FL
        end if
c
cc      open(unit=12,file='pld1.dat',
cc      &          position='append')
cc      write(12,'(10E15.6)')time,pres,liftp,liftp,dragp,dragf,FL,FD,
cc      & vsolid,ysolid
cc      close(12)
c
        return
    end

c
c
cc      subroutine wrtfld()
c
        integer nnx,nyy,MxSurf,Mxy
        parameter (nnx=600,nyy=850,MxSurf=50)

```

```

parameter (Mxy=2*nnx+2*ny)
c
double precision w
parameter      (w=0.5D0)
c
common /veloxx/ u(0:nnx,0:ny+1),v(0:nnx+1,0:ny),
&              p(0:nnx+1,0:ny+1),a(0:nnx,ny,4),b(nnx,0:ny,4)
double precision u,v,p,a,b
c
common / bndinf/ txu(nnx),txv(nnx),tyu(nnx),tyv(nnx),
&              vnoru(nnx),vnorv(nnx),fyu(nnx),fyv(nnx),
&              vup(0:nnx+1),uup(0:nnx),influx
double precision txu,txv,tyu,tyv,vnoru,vnorv,fyu,fyv,vup,uup,
&              influx
c
common / bniinf/ jyu(nnx),jyv(nnx)
integer jyu,jyv
c
common / griddx/ xcoord(0:nnx), ycoord(0:ny)
double precision xcoord, ycoord
c
common / gridd/ xcrd(0:nnx+1), ycrd(0:ny+1), scalar(nnx,ny),
&              xfree(Mxy,MxSurf),yfree(Mxy,MxSurf)
double precision xcrd, ycrd, scalar, xfree, yfree
c
common /SORpar/ aim(nnx,ny),aip(nnx,ny),ajm(nnx,ny),
&              ajp(nnx,ny),diag(nnx,ny),f(nnx,ny)
double precision aim,aip,ajm,ajp,diag,f
c
common / bndvinfi/ ip1v(Mxy),jp1v(Mxy),ip2v(Mxy),jp2v(Mxy),
&                 iinterp(Mxy),jinterp(Mxy),nbndv
integer ip1v,jp1v,ip2v,jp2v,iinterp,jinterp,nbndv
c
common / bnduinfi/ ip1u(Mxy),jp1u(Mxy),ip2u(Mxy),jp2u(Mxy),
&                 iinterpu(Mxy),jinterpu(Mxy),nbndu
integer ip1u,jp1u,ip2u,jp2u,iinterpu,jinterpu,nbndu

common / bndvinfR/ wv1(Mxy),wv2(Mxy)
double precision wv1,wv2
c
common / bnduinfR/ wu1(Mxy),wu2(Mxy)
double precision wu1,wu2
c
common / bndpinfi/ip1p(0:Mxy),jp1p(0:Mxy),ip2p(0:Mxy),jp2p(0:Mxy),
&              ip3p(0:Mxy),jp3p(0:Mxy),iinterpp(0:Mxy),jinterpp(0:Mxy),
&              nbndp
integer ip1p,jp1p,ip2p,jp2p,ip3p,jp3p,iinterpp,jinterpp,nbndp
c
common / bndpinfR/ teta(0:Mxy),unitvi(0:Mxy),unitvj(0:Mxy),
&              wp1(0:Mxy),wp2(0:Mxy),delta1(0:Mxy)
double precision teta,unitvi,unitvj,wp1,wp2,delta1
c
common /masksx/ pmask(0:nnx,0:ny),umask(0:nnx,0:ny),
&              vmask(0:nnx,0:ny)
double precision pmask,umask,vmask
c
common /minsx/ pins(0:nnx+2,0:ny+2),uins(0:nnx+2,0:ny+2),
&              vins(0:nnx+2,0:ny+2)
double precision pins,uins,vins
c
common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,nx2,ny2,nn
double precision Re, RRe,dt,time,dts
c
common /velmen/ um(nnx,ny), vm(nnx,ny), pm(nnx,ny),
&              uu(nnx,ny), vv(nnx,ny), uv(nnx,ny)
double precision um,vm,pm,uu,vv,uv
c

```

```

common /parmen/ nmean
integer nmean

c
integer i,j,k

c

common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid

c
double precision dliftp,dliftp,ddragf,ddragp,dpres

c
common /FCDCL/ FL,FD,liftp,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
c & FL2,FD2
double precision FL,FD,liftp,dragf,liftp,dragp,pres,FLift,Fdrag,Fa
c & FL2,FD2

c

common /homeadd/ home
character*40 home

c
c
cc call force()
cc open(unit=12,file='pld1.dat',
cc & position='append')
cc write(12,'(15E15.6)')time,pres,liftp,liftp,dragp,dragf,FL,FD,
cc & vsolid,ysolid,FLift,Fdrag,Fa,FL2,FD2
c write(12,'(6E16.8)') time,pres,liftp,drag,liftp,dragl
cc close(12)
c
cc open(unit=12,file='degree.dat',
cc & position='append')
cc write(12,'(6E16.8)') degree,dpres,dliftp,ddragp,dliftp,ddragf
cc close(12)
c
c *****velocity out put for test of the divergenc
cc open(unit=12,file='velocity.dat')
cc rewind(12)
cc do i=89,111
cc do j=89,111
cc write(12,'(A,2I4,5E16.8)')i,j,u,umask,v,vmask,pmask',i,j,u(i,j),
cc & umask(i,j),v(i,j),vmask(i,j),pmask(i,j)
cc end do
cc end do
cc close(12)

c***** output by results on the Coordinate line

cc open(unit=12,file='fieldcoord.dat')
cc rewind(12)
c
cc write(12,*) 'variables="x","y","u","v","p","umask",
cc & "vmask","pmask"'
cc write(12,*)
cc & 'ZONE T="t=",time," I = ',nx,'J = ',ny,'F=BLOCK'
cc write(12,'(5E16.8)') ((xcoord(i),i=1,nx),j=1,ny)
cc write(12,'(5E16.8)') ((ycoord(j),i=1,nx),j=1,ny)
cc write(12,'(5E16.8)') ((u(i,j),i=1,nx),j=1,ny)
cc write(12,'(5E16.8)') ((v(i,j),i=1,nx),j=1,ny)
cc write(12,'(5E16.8)') ((p(i,j),i=1,nx),j=1,ny)
cc write(12,'(5E16.8)') ((umask(i,j),i=1,nx),j=1,ny)
cc write(12,'(5E16.8)') ((vmask(i,j),i=1,nx),j=1,ny)
cc write(12,'(5E16.8)') ((pmask(i,j),i=1,nx),j=1,ny)
cc close(12)

c
open(unit=12,file='field.dat')

```

```

rewind(12)
c
write(12,*) 'variables="x","y","u","v","p","umask",
& "vmask","pmask"'
write(12,*)
& 'ZONE T="t=",time," I = ',nx,'J = ',ny,'F=BLOCK'
write(12,' (5E16.8)') ((xcrd(i),i=1,nx),j=1,ny)
write(12,' (5E16.8)') ((ycrd(j),i=1,nx),j=1,ny)
write(12,' (5E16.8)')
& ((0.5D0*pmask(i,j)*(u(i-1,j)+u(i,j)),i=1,nx),j=1,ny)
write(12,' (5E16.8)')
cc22/5/13 & ((0.5D0*pmask(i,j)*(v(i,j-1)+v(i,j)),i=1,nx),j=1,ny)
& ((0.5D0*(v(i,j-1)+v(i,j)),i=1,nx),j=1,ny)
write(12,' (5E16.8)') ((p(i,j),i=1,nx),j=1,ny)
write(12,' (5E16.8)') ((umask(i,j),i=1,nx),j=1,ny)
write(12,' (5E16.8)') ((vmask(i,j),i=1,nx),j=1,ny)
write(12,' (5E16.8)') ((pmask(i,j),i=1,nx),j=1,ny)
close(12)

cc open(unit=12,file='fieldu.dat')
cc rewind(12)
c
cc write(12,*) 'variables="x","y","u","umask","vmask"'
cc write(12,*)
cc & 'ZONE T="t=",time," I = ',nx,'J = ',ny,'F=BLOCK'
cc write(12,' (5E16.8)') ((xcoord(i),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((ycrd(j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((u(i,j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((umask(i,j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((vmask(i,j),i=1,nx),j=1,ny)
cc close(12)
c
c
cc open(unit=12,file='fieldv.dat')
cc rewind(12)
c
cc write(12,*) 'variables="x","y","v","pmask"'
cc write(12,*)
cc & 'ZONE T="t=",time," I = ',nx,'J = ',ny,'F=BLOCK'
cc write(12,' (5E16.8)') ((xcrd(i),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((ycoord(j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((v(i,j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((pmask(i,j),i=1,nx),j=1,ny)
cc close(12)
c
cc open(unit=12,file='f.dat')
cc rewind(12)
c
cc write(12,*) 'variables="x","y","f","umask","vmask","pmask"'
cc write(12,*)
cc & 'ZONE T="t=",time," I = ',nx,'J = ',ny,'F=BLOCK'
cc write(12,' (5E16.8)') ((xcrd(i),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((ycrd(j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((f(i,j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((umask(i,j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((vmask(i,j),i=1,nx),j=1,ny)
cc write(12,' (5E16.8)') ((pmask(i,j),i=1,nx),j=1,ny)
cc close(12)
c
c
return
end
cc
cc
subroutine savfld()
c
integer nnx, nny, MxSurf, Mxy

```

```

parameter (nnx=600, nny=850, MxSurf=50)
parameter (Mxy=2*nnx+2*nny)
c
double precision w
parameter (w=0.5D0)
c
common /veloxx/ u(0:nnx,0:nny+1), v(0:nnx+1,0:nny),
& p(0:nnx+1,0:nny+1), a(0:nnx,nny,4), b(nnx,0:nny,4)
double precision u, v, p, a, b
c
common / bndinf/ txu(nnx), txv(nnx), tyu(nnx), tyv(nnx),
& vnoru(nnx), vnorv(nnx), fyu(nnx), fyv(nnx),
& vup(0:nnx+1), uup(0:nnx), influx
double precision txu, txv, tyu, tyv, vnoru, vnorv, fyu, fyv, vup, uup,
& influx
c
common / bniinf/ jyu(nnx), jyv(nnx)
integer jyu, jyv
c
common / griddx/ xcoord(0:nnx), ycoord(0:nny)
double precision xcoord, ycoord
c
common /masksx/ pmask(0:nnx,0:nny), umask(0:nnx,0:nny),
& vmask(0:nnx,0:nny)
double precision pmask, umask, vmask
common / gridd/ xcrd(0:nnx+1), ycrd(0:nny+1), scalar(nnx,nny),
& xfree(Mxy,MxSurf), yfree(Mxy,MxSurf)
double precision xcrd, ycrd, scalar, xfree, yfree
c
common /SORpar/ aim(nnx,nny), aip(nnx,nny), ajm(nnx,nny),
& ajp(nnx,nny), diag(nnx,nny), f(nnx,nny)
double precision aim, aip, ajm, ajp, diag, f
c
common / dimenx/ nx, ny, Re, RRe, dt, time, dts, nx2, ny2, nn
integer nx, ny, nx2, ny2, nn
double precision Re, RRe, dt, time, dts
c
common /velmen/ um(nnx,nny), vm(nnx,nny), pm(nnx,nny),
& uu(nnx,nny), vv(nnx,nny), uv(nnx,nny)
double precision um, vm, pm, uu, vv, uv
c
common /rkcom / urk(0:nnx,0:nny+1), vrk(0:nnx+1,0:nny),
& ar(10,10), br(10), nrk
double precision urk, vrk, ar, br
integer nrk
c
common /parmen/ nmean
integer nmean
c
common /sbody/ vsolid, ysolid, usolid, xsolid, aysolid, axsolid
double precision vsolid, ysolid, usolid, xsolid, aysolid, axsolid
c
common /FaVYold/ FLn, FLold, aysolidn, vsolidn, ysolidn
double precision FLn, FLold, aysolidn, vsolidn, ysolidn
c
integer i, j, k
c
common /homeadd/ home
character*40 home
c
open(unit=12, file='field.bin',
&form='UNFORMATTED')
rewind(12)
c
write(12) time, dt, Re, vsolid, ysolid, aysolid, usolid, xsolid,
&vsolidn, ysolidn, aysolidn, FLn, FLold
write(12) ((u(i, j), i=0, nx), j=0, ny+1)
write(12) ((v(i, j), i=0, nx+1), j=0, ny)

```

```

write(12) ((p(i,j),i=0,nx+1),j=0,ny+1)
write(12) ((urk(i,j),i=0,nx),j=0,ny+1)
write(12) ((vrk(i,j),i=0,nx+1),j=0,ny)

c
close(12)

c
open(unit=12,file='means.bin',
&form='UNFORMATTED')
rewind(12)

c
write(12) nmean
write(12) ((um(i,j),i=1,nx),j=1,ny)
write(12) ((vm(i,j),i=1,nx),j=1,ny)
write(12) ((pm(i,j),i=1,nx),j=1,ny)
write(12) ((uu(i,j),i=1,nx),j=1,ny)
write(12) ((vv(i,j),i=1,nx),j=1,ny)
write(12) ((uv(i,j),i=1,nx),j=1,ny)

c
close(12)

c
open(12, file = 'movie.dat',position='append',
& form='formatted')

c
write(12,*)
c
write(12,*)
c
& 'ZONE T="t=",time," I = ',nx,' J = ',ny,' F=BLOCK'

write(12,*) 'ZONE T="t=",time," I=',nx,' J=',ny,' F=BLOCK'
write(12,'(A,f16.6)') 'SOLUTIONTIME=',time
c
write(12) nx,ny,time
write(12,'(5E16.8)') ((xcrd(i)+xsolid,i=1,nx),j=1,ny) ! test4
write(12,'(5E16.8)') ((ycrd(j)+ysolid,i=1,nx),j=1,ny) ! test4
ccc write(12,'(5E16.8)') ((xcrd(i),i=1,nx),j=1,ny) ! test2 and test3
ccc write(12,'(5E16.8)') ((ycrd(j),i=1,nx),j=1,ny) ! test2 and test3
ccc write(12,'(5E16.8)')
c
ccc & ((0.5D0*(u(i-1,j)+u(i,j))),i=1,nx),j=1,ny)
ccc write(12,'(5E16.8)')
ccc & ((0.5D0*(v(i,j-1)+v(i,j))),i=1,nx),j=1,ny)
ccc write(12,'(5E16.8)') ((p(i,j),i=1,nx),j=1,ny)
c
write(12,'(5E16.8)')
& ((pmask(i,j)*0.5D0*(u(i-1,j)+u(i,j))+usolid),i=1,nx),j=1,ny)
write(12,'(5E16.8)')
& ((pmask(i,j)*0.5D0*(v(i,j-1)+v(i,j))+vsolid),i=1,nx),j=1,ny)
write(12,'(5E16.8)') ((pmask(i,j)*p(i,j),i=1,nx),j=1,ny)

close(12)

c
return
end

cc
cc
subroutine getfld(ex)

c
logical ex

c
integer nnx, nny, MxSurf, Mxy
parameter (nnx=600, nny=850, MxSurf=50)
parameter (Mxy=2*nnx+2*ny)

c
double precision w
parameter (w=0.5D0)

c
common /veloxx/ u(0:nnx,0:ny+1),v(0:nnx+1,0:ny),
& p(0:nnx+1,0:ny+1),a(0:nnx,ny,4),b(nnx,0:ny,4)

```

```

double precision u,v,p,a,b
c
common / bndinf/ txu(nnx),txv(nnx),tyu(nnx),tyv(nnx),
&                vnoru(nnx),vnorv(nnx),fyu(nnx),fyv(nnx),
&                vup(0:nnx+1),uup(0:nnx),influx
double precision txu,txv,tyu,tyv,vnoru,vnorv,fyu,fyv,vup,uup,
&                influx
c
common / bniinf/ jyu(nnx),jyv(nnx)
integer jyu,jyv
c
common / griddx/ xcoord(0:nnx), ycoord(0:nyy)
double precision xcoord, ycoord
c
common / gridd/ xcrd(0:nnx+1), ycrd(0:nyy+1), scalar(nnx,nyy),
&              xfree(Mxy,MxSurf), yfree(Mxy,MxSurf)
double precision xcrd, ycrd, scalar, xfree, yfree
c
common /SORpar/ aim(nnx,nyy),aip(nnx,nyy),ajm(nnx,nyy),
&              ajp(nnx,nyy),diag(nnx,nyy),f(nnx,nyy)
double precision aim,aip,ajm,ajp,diag,f
c
common / bndvinfi/ ip1v(Mxy),jp1v(Mxy),ip2v(Mxy),jp2v(Mxy),
&                iinterp(Mxy),jinterp(Mxy),nbndv
integer ip1v,jp1v,ip2v,jp2v,iinterp,jinterp,nbndv
c
common / bnduinfi/ ip1u(Mxy),jp1u(Mxy),ip2u(Mxy),jp2u(Mxy),
&                iinterpu(Mxy),jinterpu(Mxy),nbndu
integer ip1u,jp1u,ip2u,jp2u,iinterpu,jinterpu,nbndu

common / bndvinfR/ wv1(Mxy),wv2(mxy)
double precision wv1,wv2
c
common / bnduinfR/ wu1(Mxy),wu2(Mxy)
double precision wu1,wu2
c
common /masksx/ pmask(0:nnx,0:nyy),umask(0:nnx,0:nyy),
&              vmask(0:nnx,0:nyy)
double precision pmask,umask,vmask
c
common / dimenx/ nx,ny,Re,RRe,dt,time,dts,nx2,ny2,nn
integer nx,ny,nx2,ny2,nn
double precision Re, RRe,dt,time,dts
c
common /velmen/ um(nnx,nyy), vm(nnx,nyy), pm(nnx,nyy),
&              uu(nnx,nyy), vv(nnx,nyy), uv(nnx,nyy)
double precision um,vm,pm,uu,vv,uv
c
common /parmen/ nmean
integer nmean
c
common /rkcom / urk(0:nnx,0:nyy+1), vrk(0:nnx+1,0:nyy),
&              ar(10,10), br(10), nrk
double precision urk, vrk, ar, br
integer nrk
c
common /sbody/ vsolid, ysolid, usolid, xsolid,aysolid,axsolid
double precision vsolid, ysolid, usolid, xsolid,aysolid,axsolid
c
common/FaVYold/FLn,FLold,aysolidn,vsolidn,ysolidn
double precision FLn,FLold,aysolidn,vsolidn,ysolidn
c
integer i,j,k
double precision dt1
c
common /homeadd/ home
character*40 home
c

```



```

    inquire(file='field.bin',EXIST=ex)
    if (.not. ex) return
c
    open(unit=12,file='field.bin',
&form='UNFORMATTED')
    rewind(12)
c
    write(*,*) 'Reading from field.bin '
c
    read(12) time,dt,Re,vsolid,ysolid,aysolid,usolid,xsolid,
&vsolidn,ysolidn,aysolidn,FLn,FLold
    RRe=1D0/Re
    read(12) ((u(i,j),i=0,nx),j=0,ny+1)
    read(12) ((v(i,j),i=0,nx+1),j=0,ny)
    read(12) ((p(i,j),i=0,nx+1),j=0,ny+1)
    read(12) ((urk(i,j),i=0,nx),j=0,ny+1)
    read(12) ((vrk(i,j),i=0,nx+1),j=0,ny)

c
    close(12)
c
c
    return
    end

c--
c etime.f: Demonstrate measurement of elapsed time
c--
    subroutine etimetest
    real etime      ! Declare the type of etime()
    real elapsed(2) ! For receiving user and system time
    real total     ! For receiving total time
    integer i, j
    print *, 'Start'
    total=etime(elapsed)
    open(unit=12,file='ptime')
    rewind(12)

    write(12,*)'End:*total=', total, ' user=', elapsed(1),
&      'system=', elapsed(2)
    close(12)
c
    Stop
    Return
    end

```

INVESTIGATING THE EFFECT OF ROTATIONAL DEGREE OF FREEDOM ON A CIRCULAR CYLINDER AT LOW REYNOLDS NUMBER IN CROSS FLOW

SEYED HOSSEIN MADANI*, JAN WISSINK, HAMID BAHAI

School of Engineering and Design - Brunel University West London

*Islamic Azad University – South Tehran Branch
email: Hossein.Madani@brunel.ac.uk

Key words: circular cylinder; Vortex-Induced Vibrations, rotational d.o.f.

Summary. Numerical simulations of Vortex-Induced Vibrations (VIV) of a circular cylinder in cross flow with a rotational degree of freedom about its axis have been carried out by means of a finite-volume method. The study is performed in two dimensions at a Reynolds number of $Re_D = 100$, based on the free stream velocity and the diameter, D , of the cylinder. The effect of the rotational degree of freedom on the cylinder's lift and drag forces are compared with the baseline simulation results of flow around a stationary cylinder. The introduction of a rotational degree of freedom (d.o.f) is observed to cause the lift and drag forces to change. Also, the pattern of vortex shedding behind the cylinder is found to drastically change when the cylinder is allowed to rotate.

1. INTRODUCTION

The study of flows around cylinders has a long history [1, 2]. The early studies were focussed on the flow around a stationary cylinder at various Reynolds numbers. Subsequently, investigations have been carried out of flow around a cylinder with a prescribed rotational velocity [2]. Although the study of flows around rotating circular cylinders is not new, most of the previous works consider the rotational speed as a parameter that can be used to decrease the effect of vortex shedding on the cylinder. In other words, angular velocity is viewed as a way to reduce the root mean square of the lift force. This is the reason why rotation of the cylinder is used in feedback control of wakes [3]. Forced oscillatory rotation of a circular cylinder has also been investigated numerically as well as experimentally [4]. All of these studies focused on imposed oscillatory angular velocities. Etienne and Fontaine [5] studied the effect of vortex shedding on a two dimensional cylinder with two spatial d.o.f. They observed that the cylinder was mainly oscillating transversely and slightly in line with the flow. When they added a rotational degree of freedom, for an arbitrary rotational moment of inertia, the transverse amplitude of oscillation was found to be reduced by a factor of two, while the mean in-line deflection was also found to decrease by a factor between 1.5 to 2. In their case, the Magnus effect was found to be negligible as the maximum angular velocity was only on the order of 5% of the free-stream velocity U [5].

In this study, we evaluate the effect of introducing a rotational degree of freedom, on the flow around a circular cylinder. To achieve this we introduce the rotational angle of the cylinder as an unknown that is affected by friction-induced torque.

Below, we present some of the results obtained by performing a parametric study in which the moment of inertia (I) and the rotational spring rigidity (k) are varied. The spring rigidity is used to control the rotational degree of freedom (d.o.f) and both k and I determine the natural frequency (f) of the system. An important parameter in this context is the so-called reduced velocity, $U_r = U/D$, where U is the free-stream velocity and D is diameter of the cylinder. It should be noted that both k and I are defined for a unit-length cylinder.

2. GENERAL SPECIFICATION

The cylinder was allowed to rotate about its axis. The rotation was controlled by adding a torsional spring with stiffness K . Without the presence of the spring the cylinder was observed to rotate rigidly in one direction.

Because of the simplicity of the problem and the low Reynolds number, we were able to model the set up as a two-dimensional flow problem. The computational domain is shown in figure 2. At the inlet, the flow is assumed to be uniform with $u=U_0$ and $v=0$, where u and v are the velocities of the flow in x -direction and y -direction respectively. A free-slip boundary condition is applied along the upper and lower boundaries while a convective outflow boundary condition is applied at the outlet. At the surface of the cylinder, finally, a no-slip boundary condition is prescribed.



Figure 1 : Cylinder with Rotational degree of freedom, a Torsional spring- stiffness of K and the moment of Inersial of I

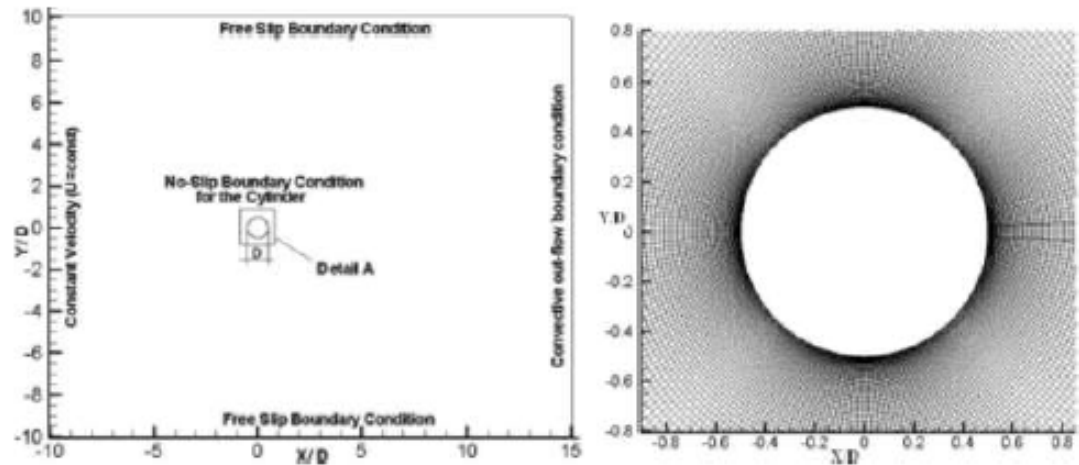


Figure 2: Left: The computational domain showing the boundary conditions. Right: Zoomed view of the O-mesh close to the cylinder corresponding to "Detail A" at the left

This problem is an example of a rotational harmonic oscillator that can oscillate about the axis of the cylinder. This behaviour is analogous to linear spring-mass oscillators. The general equation of motion is given by:

$$I \frac{d^2\theta}{dt^2} + C \frac{d\theta}{dt} + \kappa\theta = \tau(t) \quad (1)$$

If the damping is small, $C \ll \sqrt{\kappa I}$, as is the case in this study, the frequency of vibration is very close to the natural resonance frequency of the system:

$$f_n = \frac{\omega_n}{2\pi} = \frac{1}{2\pi} \sqrt{\kappa/I} \quad (2)$$

In the absence of a driving force ($\tau = 0$), the general solution of the resulting homogeneous problem is given by:

$$\theta = A e^{-\alpha t} \cos(\omega t + \phi) \quad (3)$$

Where:

$$\omega = \sqrt{\omega_n^2 - \alpha^2} = \sqrt{\kappa/I - (C/2I)^2} \quad (4)$$

Table 1 : Definition of terms in the equations

Definition of terms		
Term	Unit	Definition
θ	Radians	Angle of deflection from rest position
I	kg m ²	Moment of inertia
C	kg m ² s ⁻¹ rad ⁻¹	Rotational friction (damping)
κ	N m rad ⁻¹	Coefficient of torsion spring
\mathcal{T}	N m	Drive torque
f_n	Hz	Undamped (or natural) resonance frequency
ω_n	rads ⁻¹	Undamped resonance frequency in radians
f	Hz	Damped resonance frequency
ω	rads ⁻¹	Damped resonance frequency in radians
α	s ⁻¹	Reciprocal of damping time constant
ϕ	Rad	Phase angle of oscillation
L	M	Distance from axis to where force is applied

The angular velocity of the cylinder is determined by a numerical approximation of eq. (1) with C=0, using an Euler scheme for the integration of time. The torque $\tau(t)$ is calculated every time step by integrating the tangential frictional forces of the flow on the cylinder.

3. NUMERICAL RESULTS

For this simulation the LESOCC flow solver has been used. LESOCC has been developed at the Institute of Hydromechanics at Karlsruhe, Institute of Technology, Germany. In Wissink and Rodi [6] it has been extensively tested for the simulation of flow around a cylinder at $Re=3200$. LESOCC uses a second-order accurate discretization of the convection and diffusion, combined with a three-stage Runge-Kutta method for the time-integration. It uses a collocated variable arrangement combined with momentum interpolation to avoid a decoupling of the pressure and velocity fields.

For the present study, a mesh independency test was carried out and, as a result, a mesh with (360×126) points in the circumferential and radial direction respectively was chosen. Numerous runs have been carried out on the computing cluster at Brunel University. To simulate each case using 8 processors it takes nearly 2000 hours for the results to converge. Figure 3 shows how the results converged for one specific case. The Reynolds number was kept constant at $Re=100$ for all cases. To initiate the vortex shedding, we applied a random perturbation to the flow. Figure 4 shows that the results are not dependent on the initial perturbation.

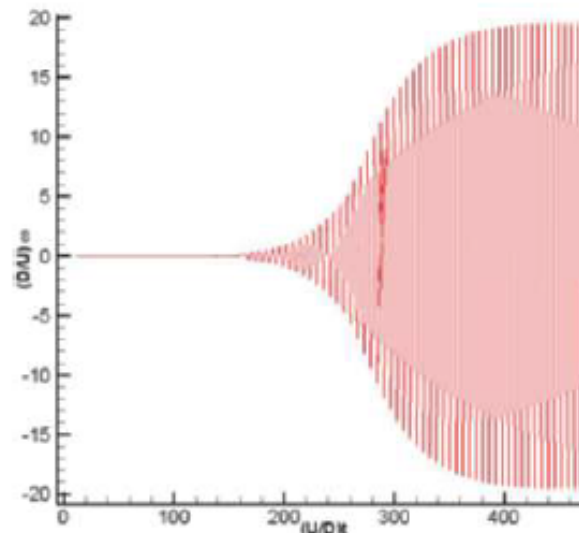


Figure 3: The results of the simulation Converges- $Re=100$, $I=0.333$, $K=0.3648$ $Ur=6$

In this study, the effect of a rotational degree of freedom on vortex shedding and lock-in phenomena was investigated and the results were compared with flow around a stationary cylinder. Initially an attempt was made to perform a simulation with a rotational d.o.f without any restoring force ($K=0$). As a result, the cylinder was observed to rotate in only one direction. It was therefore decided to add a restoring force by the introduction of a rotational spring ($k>0$). To establish which moment of inertia, I , would be relevant to our problem, we assumed a solid cylinder with the same density of water (1000kg/m^3). The diameter of the cylinder was chosen to be 20 cm; as a result $I = (1/8)mD^2 = (1/32)\pi\rho D^4 = 0.157\text{ kg/m}^2$. (For the

case with $D=0.2\text{m}$ and a density of water equal to 1000 kg/m^3 , the equivalent moment of inertia for the cylinder becomes $I=0.157\text{ kg.m}^2$) and the corresponding non-dimensional moment of inertia becomes $I=0.5$. Figure 5 on the left shows the effect of inertia of the cylinder on the frequency of the vortex shedding of the cylinder (with a constant rotational stiffness $k=0.05$). For high (low) amounts of inertia the frequency decreases (increases) dramatically. Figure 5, right, depicts the relation between K/I and the natural frequency of the system and the frequency of the rotational velocity (ω). This graph clearly proves equation (2). The power of the K/I is 0.5045, which is almost the same as predicted by theory (0.5) and the coefficient is 0.1598 which is very close to the coefficient $1/2\pi=0.1591$ in eq. (2).

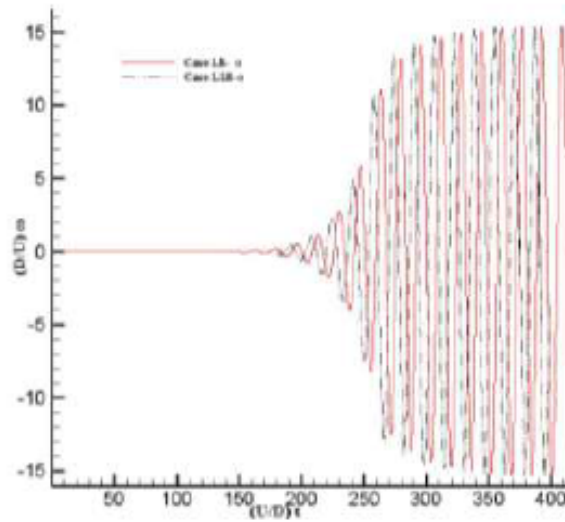


Figure 4: The effect of changes in random perturbation on the convergence of the results for two similar cases- $Re=100, I=0.333, K=0.05, U_r=16$

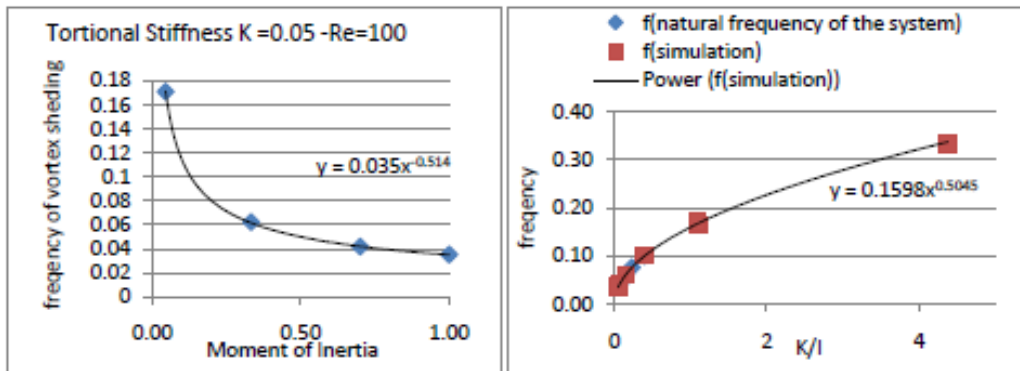


Figure 5 –left: Natural frequency verses inertia when $K=0.05$ constant. Right: natural frequency and vortex shedding frequency verses (K/I) , the parameters are non-dimensional.

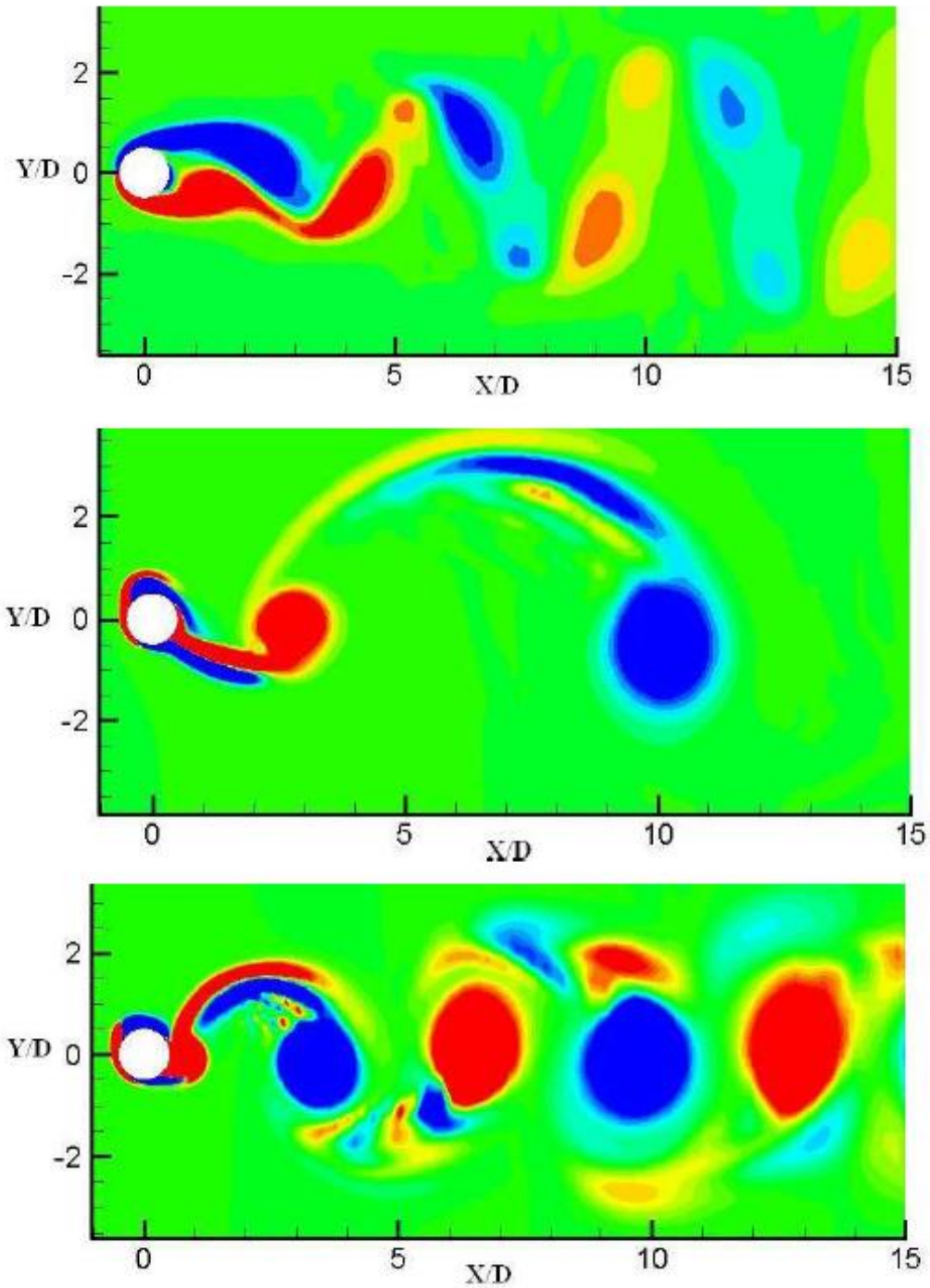


Figure 6 : Vortex Shedding $Re=100$ - first case: stationary cylinder, $St=0.167$, Second case $I=0.333$ and $K=0.05$, $Ur=16$, $St=0.0625$, third case: $I=0.333$, $K=0.365$, $Ur=6$, $St=0.167$

Figure 6 shows vortex shedding for 3 different cases in a two dimensional flow at $Re=100$. The first one is the stationary cylinder; the frequency of the vortex shedding for this case is 0.17. These result exactly match the results of Roshko [7], who measured the frequencies using a hot-wire velocity probe. For the low Reynolds number laminar region Roshko condensed his results to an equation of the form $St = 0.212 (1 - 21.2 / Re)$ where St is the Strouhal number $S=FD/U$ [7]. The second and third pictures in figure 6 show the vortex shedding from a cylinder with a rotational degree of freedom. For the cases $I=0.333$ combined with $k=0.05$ and $k=0.365$, the frequencies of vortex shedding become $f=0.0625$ and $f=0.167$ respectively. In figure 7, the effect of rotational d.o.f was compared with the stationary cylinder. Etienne and Fontaine [5] observed that the introduction of a rotational degree of freedom causes a reduction in the vortex-induced vibration in the transverse direction with the flow [5]. It implies that we should expect a lower lift when we have a rotational d.o.f in combination with spatial degrees of freedom. In the absence of a spatial degree of freedom, our results show a completely different behaviour and predict a significant increase in unsteady lift forces acting on the cylinder due to the Magnus effect.

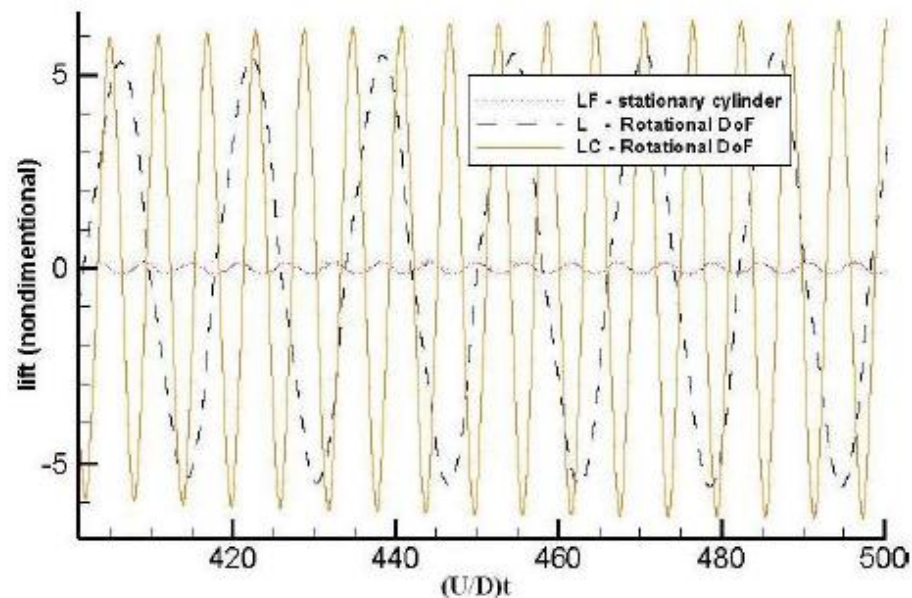


Figure 7- lift for three cases. first case : stationary cylinder, second case: Rotational d.o.f $I=0.333$ and $K=0.05$, third case: Rotational degree of freedom $I=0.333$, $K=0.365$

4. Conclusion

The introduction of a rotational degree of freedom which allows the cylinder to rotate about its axis, has a significant effect on the pattern of vortex shedding at low Reynolds numbers. In all cases considered, the vortex shedding locks-in to the natural frequency of the inertial/spring system. Compared to the baseline simulation of flow around a stationary cylinder, the addition of a rotational degree of freedom to the cylinder was observed to

significantly increase unsteady lift forces because of the Magnus effect, while also the drag forces were not diminished. In the near future, we aim to complete the present parametric study of the effects of inertia/spring stiffness on the flow pattern and the lift and drag forces.

References

- [1] C.H.K. Williamson and R. Govardhan, "Vortex- Induced Vibration", *Annu. Rev. Fluid Mech.*(2004). 36:413-55.
- [2] T.Sarpkaya, "A critical review of the intrinsic nature of vortex-induced vibrations", *Journal of fluids and structure* 19 (2004) 389-447.
- [3] Lee S B, Baek S J, Sung H J (2009). Feedback control of a circular cylinder wake with rotation oscillation, *Fluid Dyn. Res.*, **41**, 1-24.
- [4] Lam K M (2009). Vortex shedding flow behind a slowly rotating circular cylinder. *Journal of Fluids and Structures*, **25**, 245–262,
- [5] S. Etienne, E. Fontaine, " Effect of Rotational degree of freedom on Vortex-Induced Vibrations of a circular cylinder in Cross-Flow", *Proceedings of the international offshore and polar engineering conference*, 3, pp. 1089-1093, 2010.
- [6] J.G. Wissink, W. Rodi, "Numerical study of the near wake of a circular cylinder", *International journal of heat and fluid flow*, 29(2008) 1060-1070.
- [7] Roshko, Anatol (1954). On the drag and shedding Frequency of two dimensional bluff bodies, NACA technical note 3169.

A COMPARATIVE STUDY OF IMMERSED-BOUNDARY INTERPOLATION METHODS FOR A FLOW AROUND A STATIONARY CYLINDER AT LOW REYNOLDS NUMBER

SEYED HOSSEIN MADANI*, JAN WISSINK, HAMID BAHAI

School of Engineering and Design - Brunel University West London

*Islamic Azad University – South Tehran Branch

email: Hossein.Madani@brunel.ac.uk

Key words: Immersed-Boundary, Reconstructive method, Interpolation methods, stationary cylinder, Vortex shedding

ABSTRACT

The accuracy and computational efficiency of various interpolation methods for the implementation of non grid-confirming boundaries is assessed. The aim of the research is to select an interpolation method that is both efficient and sufficiently accurate to be used in the simulation of vortex induced vibration of the flow around a deformable cylinder. Results are presented of an immersed boundary implementation in which the velocities near non-confirming boundaries were interpolated in the normal direction to the walls. The flow field is solved on a Cartesian grid using a finite volume method with a staggered variable arrangement. The Strouhal number and Drag coefficient for various cases are reported. The results show a good agreement with the literature. Also, the drag coefficient and Strouhal number results for five different interpolation methods were compared it was shown that for a stationary cylinder at low Reynolds number, the interpolation method could affect the drag coefficient by a maximum 2% and the Strouhal number by maximum of 3%. In addition, the bi-linear interpolation method took about 2% more computational time per vortex shedding cycle in companion to the other methods.

INTRODUCTION

Obtaining accurate solutions for Fluid-Structure Interaction (FSI) problems is of interest in many engineering and scientific applications. A broad classification of FSI methods is based on the type of mesh employed in the discretisation, where we can differentiate between boundary-conforming and non-boundary-conforming mesh methods [1]. A well-known conforming mesh method is the Arbitrarily Lagrangian-Eulerian method (ALE). For non-conforming mesh methods, usually an immersed boundary method is used and most recent developments in FSI methods are based on this approach. The latter is the subject of this review.

The immersed-boundary (IB) method is a technique for solving flow problems in regions with irregular boundaries using a simple structured grid solver. The term “immersed boundary method” was initially used for a method developed by Peskin [2] which was used to simulate blood flow in a cardiovascular system. It was specifically designed to handle deforming (elastic) boundaries interacting with low Reynolds number flow. The simulation was carried

out on a Cartesian grid and at those locations where the boundary did not align with a mesh line the solution algorithm was locally modified to enforce the desired boundary conditions on the flow. More recently, numerous modifications and refinements have been proposed to enhance the accuracy, stability, and application range of the IB method [3].

Depending on the way that the boundary conditions are imposed on the immersed boundary, the IB methods can be generally categorized into continuous and discrete forcing approaches. In the continuous forcing method, a forcing function is applied to the Navier-Stokes equation in order to maintain the boundary condition on the structure (e.g. enforcing a no-slip boundary condition on a stationary body). The most important issue in this method is the definition of the continuous forcing function needed to enforce the correct boundary condition. Several different functions have been developed by Peskin [2], Saki and Birnigen [4], Beyer and Leveque [5], and Lai and Peskin [6], among others. In all cases, a distributed function was used rather than a sharp function because firstly the solid boundaries do not coincide with the Cartesian mesh and, secondly, this way the Gibbs' oscillations phenomenon [7] adjacent to the solid boundaries could be suppressed. Applying a continuous force to enforce the boundary conditions is attractive for elastic boundaries as it has a physical meaning and its implementation is relatively easy. However, the implementation of this method to enforce rigid boundaries is relatively cumbersome due to its definition. Another problem is that by using a smooth function the method cannot sharply represent the immersed boundaries which is not recommended for high Reynolds number flows [3].

Because the Navier-Stokes equations usually cannot be integrated analytically to find the forcing functions, it is usually not possible to derive an analytical forcing function to enforce specific boundary conditions. To tackle this problem, a method has been suggested by Mohd-Yusof [8] and Verzicco [9]. In this method, which is known as Indirect Discrete forcing approach, forcing functions are subtracted from the numerical solution after discretizing the Navier-Stokes equations. The important advantage of this method is that there is no need to define the forcing function parameters prior to solving the Navier-Stokes equations and there is no stability constraint due to using continuous forcing functions (Gibb's oscillation). However, it is still needed to implement the distributed forcing functions which strongly depend on the discretization algorithm. Another division of the discrete forcing approach is Direct Discrete Forcing.

Due to the need for an accurate representation of the boundary layer in high Reynolds number flow, the use of distributed, smooth forcing functions near the immersed boundary is not desirable. In these cases it is recommended to use a sharp interface with a higher local accuracy near the boundary. This goal can be achieved by imposing the boundary conditions directly on the immersed boundary. There are two well-known methods that fit into this category: the Ghost-Cell Finite-Difference Approach and the Cut-Cell Finite-Volume Approach.

In the Ghost-Cell approach the immersed boundary is implemented by the use of ghost cells. Ghost cells are cells inside the solid boundary which have at least one neighbour on the fluid side. The parameters (imaginary velocity and pressure) in the ghost cell (inside the solid) are defined by an interpolation method which implicitly enforces the correct boundary condition for the immersed boundary. Iaccarino and Verzicco [10] showed that a linear interpolation method is acceptable for those cases in which the first points of the interpolation

in the fluid are inside the viscous sub layer. Other interpolation methods have been introduced by Ghias [11].

The entire immersed boundary methods discussed so far are not designed to consider the conservation laws near the solid boundary. However, the Cut-Cell method used in combination with a Finite-Volume approach is designed in order to preserve the conservation of momentum and mass near the boundary. In this method the cells, which have been cut by the immersed boundary, are reshaped or absorbed by neighbouring cells in order to form a new trapezoidal control volume cell shape. This method has been used by Mittal [12, 13] to simulate vortex-induced vibration around a stationary and a moving body and for free falling objects. Extending this method to 3D is not straightforward and needs complex polyhedral cells, which complicate the discretization of the Navier-Stokes equations.

As discussed earlier in the discrete forcing approach, the IB is imposed on the domain after the discretization of Navier-Stokes equations. This means that introducing the boundary conditions and forcing functions is not as straightforward as in the continuous forcing approach and depends on the discretization method and its implementation. Also, in discrete forcing approach the definition of the pressure on the boundary is not as straightforward as in the continuous forcing approach and requires special treatment. Advantages of the discrete forcing approach are that the boundary conditions can be introduced sharply without any extra stability constraint, while the fluid and solid domains are clearly separated and the equations that describe the flow are only solved in the fluid domain.

In this paper, we focus on the indirect discrete forcing approach, where the forcing function is not calculated directly and added to the Navier-Stokes equation. We are not intent to use any Cut-Cell or Ghost Cell methods as the applying the Cut-Cell method for fluid-structure interaction problems with moving boundaries takes lots of computational time [14, 15], while the Ghost-Cell approach will create non-physical results when solving the fluid equations in the solid domain.

In the next section, the formulation of the fluid dynamical problem which has been used in the simulation is introduced.

FORMULATION AND NUMERICAL METHODS

The governing equation for an unsteady, incompressible fluid flow in vector form is given by the Navier-Stokes equation which reads:

$$\rho \left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} \right) = -\nabla p + \mu \nabla^2 \mathbf{V} + \mathbf{f}, \quad (1)$$

$$\nabla \cdot \mathbf{V} = 0 \quad (2)$$

where \mathbf{f} is the external force on the fluid domain which is used to implement the boundary condition on non-conforming solid boundaries. In this paper this force is not applied directly to the governing equations. Instead, the non-conforming boundary conditions are introduced by interpolating velocities close to the solid boundary.

The incompressible Navier-stokes equations in a 2D Cartesian domain are given by:

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{\text{Re}} \left(\frac{\partial^2 u_i}{\partial x_j^2} \right) \quad (3)$$

$$\frac{\partial u_i}{\partial x_i} = 0 \quad (4)$$

where p is the generalised pressure which is defined by the static pressure divided by the density. Hence, to obtain the correct the static pressure we need to multiply p by the density.

A staggered variable arrangement, as introduced by Harlow and Welch [16], is used to discretize the governing equations on a Cartesian mesh. The continuity equation is enforced by taking the divergence of the momentum equation and using the continuity equation to simplify the results to form a Poisson equation for the pressure field. This equation is solved by strongly implicit procedure (SIP), Stone's method, at every time step [17].

To maintain a consistent implementation, the pressure equation is discretized in a similar way as the momentum equation.

The extent of the computational domain was selected to be relatively large to ensure that the location of the boundaries does not affect the simulation. For this reason the size of the y direction was taken to be $20D$ and the size of the x direction was taken to be $15D$ which was deemed to be sufficient to capture the vortex shedding behind the cylinder. The mesh size was chosen was $dx=dy=0.05D$ which was checked in a mesh refinement study.

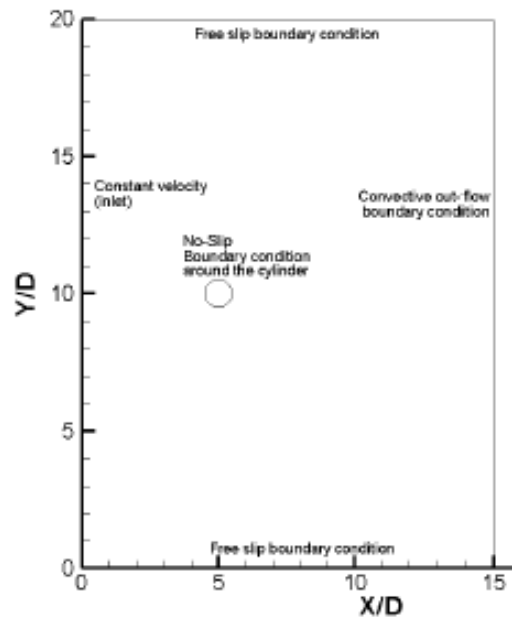


Figure 1: Fluid domain size and boundary conditions.

As the entire domain was meshed using a Cartesian grid, the implementation of the grid-conforming inlet, outlet and side boundary conditions was straightforward, while the boundary conditions along the cylinder were implemented using immersed boundary methods.

In the next section a number of velocity-interpolation methods, both from the literature as well as a novel second-order interpolation method for the implementation of the non-conforming boundary conditions in the fluid domain are introduced and compared. It will be shown that the results show a good agreement with the literature.

INTERPOLATION METHODS

In this section interpolation or reconstruction methods are compared. To enforce boundary conditions using the interpolation method, the forcing function f is not calculated directly. Instead, the flow velocity is interpolated at the interface cells and the forcing term is imposed indirectly to the discrete equations. The interface points are defined as the points in the fluid domain near the solid boundary for which one of the neighbouring points in the discretized equations is inside the solid domain. Therefore, the parameters related to these points cannot be updated through solving the governing equation (Figure 2). Any cells that contain one or more interface points are called interface cells. It is well known that most immersed boundary approaches need some sort of interpolation procedure. In the direct forcing approaches, interpolation is used in order to determine the forcing functions at the interface cells which enforce the correct boundary conditions to the governing equation. In the indirect forcing approach (interpolations approach), each time step the flow parameters in the interface cells are updated by direct interpolation and used as boundary condition for the flow solver. In this review, a number of interpolation procedures which could potentially be used in indirect discrete forcing approaches (interpolation or reconstruction approaches) will be compared.

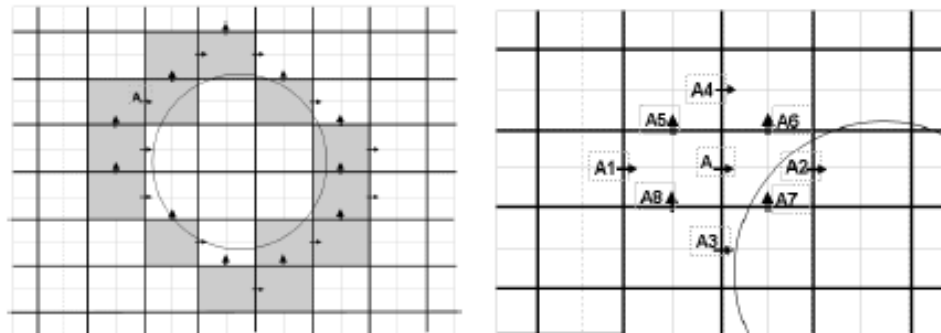


Figure 2: A 2D Cartesian mesh with a solid boundary (circle). Interface points, that require interpolation, are identified by arrows. Points A1 to A8 are all neighbouring points of A. Note that A2 and A7 are inside the solid domain.

Below it will be shown in detail how these interface cells have been treated and how possible problems that may occur near boundaries, like the decoupling of pressure and conservation of mass, may be overcome. To do so, first step is to define the interface cells for the specific geometry, which could be complicated for geometries with unknown functions [10]. Subsequently, it will be ensured that the flow governing equations are not solved inside any interface and solid cells. The most important step in the interpolation methods is to determine the flow parameters in the interface cells adjacent to the solid boundaries which will be used as boundary conditions for the rest of the flow domain that will subsequently be updated by the flow solver. Various interpolation methods have been developed to tackle this problem. In the following part, these methods are categorized and explained in more detail.

Case A: No interpolation

The simplest possible method is to select the interface cells at the solid boundaries and define the solid domain inside those cells. In fact, in this case there is no interpolation and the

solid boundary will have a stepwise shape (Figure 3). Also, the boundary itself is somewhat diffused, as in the staggered methods the boundary conditions for the different velocity components are applied at different sides of an element. Fadun [18] proposed a similar method for imposing forcing functions for immersed boundaries. Here, however, the method is applied directly to define the solid boundaries, while the governing equation will be subsequently solved in the remainder of the computational domain assuming no-slip boundary conditions for the solid boundaries. As interpolation is not needed, this method will be relatively fast while still giving acceptable results. The disadvantage of this method - when used in the calculation of flow around a circular cylinder - is that on coarse meshes shape and size differences between the cylinder and the solid boundary could affect lift and drag forces.

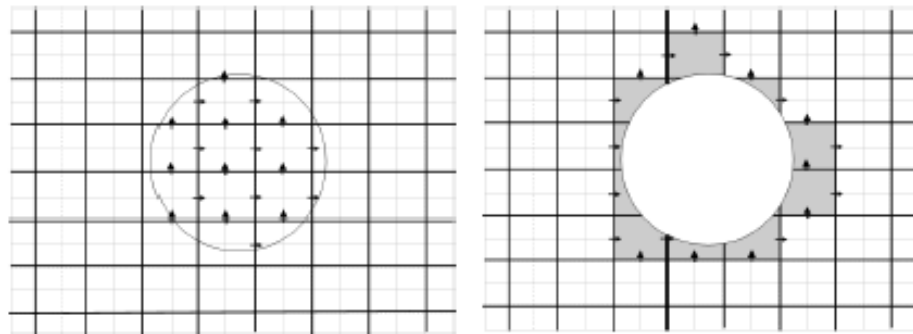


Figure 3: left, the arrows shows velocity inside solid body (solid velocity), assumed zero for a stationary cylinder. Right, hatched cells used to define weighting coefficient.

Case B: Weighting method

This method is similar to the one discussed above. The major difference is that the boundary values for the velocity in those cells that are part fluid and part solid are weighted accordingly. Figure 3 (right) shows the location of these weighted boundary velocities in the cells that are part fluid and part solid. For each of the velocity components a coefficient is determined that corresponds to the ratio of the fluid part of the two adjacent cells to the whole area of the two cells.



Figure 4: linear interpolation method: interpolating V_j for a stationary solid in vertical direction (left), interpolating V_j for a moving solid (V_{solid}) in vertical direction (middle); interpolating U_i for a moving solid (U_{solid}) in horizontal direction.

solid boundary will have a stepwise shape (Figure 3). Also, the boundary itself is somewhat diffused, as in the staggered methods the boundary conditions for the different velocity components are applied at different sides of an element. Fadhun [18] proposed a similar method for imposing forcing functions for immersed boundaries. Here, however, the method is applied directly to define the solid boundaries, while the governing equation will be subsequently solved in the remainder of the computational domain assuming no-slip boundary conditions for the solid boundaries. As interpolation is not needed, this method will be relatively fast while still giving acceptable results. The disadvantage of this method - when used in the calculation of flow around a circular cylinder - is that on course meshes shape and size differences between the cylinder and the solid boundary could affect lift and drag forces.

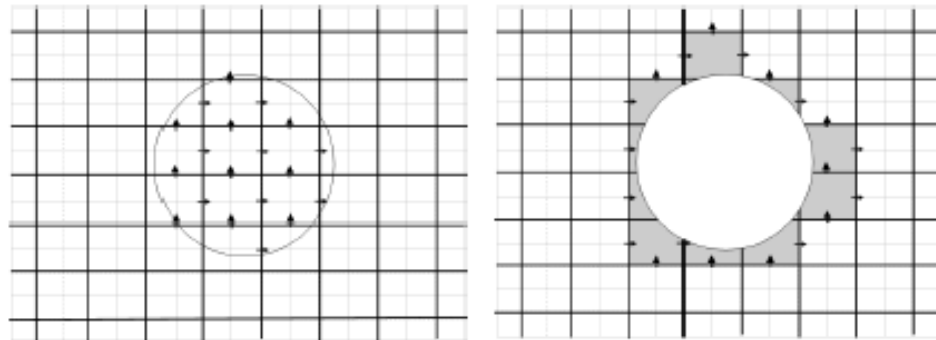


Figure 3: left, the arrows shows velocity inside solid body (solid velocity), assumed zero for a stationary cylinder. Right, hatched cells used to define weighting coefficient.

Case B: Weighting method

This method is similar to the one discussed above. The major difference is that the boundary values for the velocity in those cells that are part fluid and part solid are weighted accordingly. Figure 3 (right) shows the location of these weighted boundary velocities in the cells that are part fluid and part solid. For each of the velocity components a coefficient is determined that corresponds to the ratio of the fluid part of the two adjacent cells to the whole area of the two cells.



Figure 4: linear interpolation method: interpolating V_j for a stationary solid in vertical direction (left), interpolating V_j for a moving solid (V_{solid}) in vertical direction (middle); interpolating U_{ij} for a moving solid (U_{solid}) in horizontal direction.

pressure and the velocity near the solid boundary. Figure 6 illustrates their interpolation method that uses four adjacent velocities and enforces the momentum equation by a quadratic interpolation in the two-dimensional case. As we only focus on comparing velocity interpolation methods Kang's method is excluded from the comparison.

Case E: Proposed interpolation method

The bilinear interpolation method proposed in this paper is based on interpolating the boundary velocity values in the direction perpendicular to the solid boundary. In this method, a line from the boundary velocity position is drawn perpendicular to the boundary surface and extended to cut the line between the first two known velocities in the fluid domain (Point A, Figure 7 right). The velocity will be interpolated at the intersection point A. Then, the boundary cell velocity values will be interpolated using the solid boundary velocity (for a stationary cylinder with no-slip conditions this velocity is zero) and the velocity at point A. Figure 7 (left) shows this interpolation for velocities in y direction and Figure 7 (right) shows the interpolation for the velocity in the x direction.

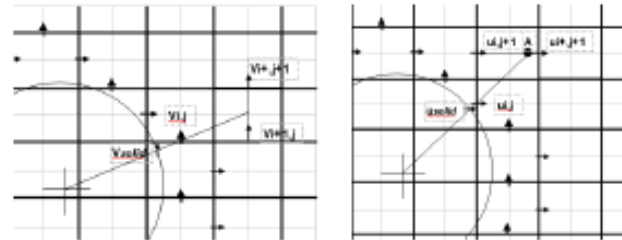


Figure 7: Bilinear proposed interpolation in this study for the cells near the solid boundary in vertical (Left) and horizontal (right) velocity components.

RESULTS AND DISCUSSION

The flow around a stationary cylinder at $Re=100$ has been simulated with different interpolation treatments to represent the immersed boundary. The Strouhal number, drag and lift coefficients for various cases are compared.

The Strouhal number is the non-dimensional frequency of the vortex shedding around the body and is defined by:

$$St = \frac{f_s D}{U_\infty},$$

where f_s is the frequency of the vortex shedding, D is the cylinder diameter and U_∞ is the far-field velocity.

The drag coefficient on a body in a fluid flow includes both the shear stress and the pressure drag on the solid surface. The dimensionless drag coefficient is defined by:

$$C_D = \frac{F_D}{\frac{1}{2}\rho u_\infty^2 D}$$

The lift force on the cylinder is generated when the vortex shedding starts around the structure. The dimensionless lift coefficient is defined by

$$C_L = \frac{F_L}{\frac{1}{2}\rho u_\infty^2 D}.$$

For any solid body both the pressure distribution and the friction along the solid surface may contribute to the lift and drag forces. In the present study, the pressure at the surface is obtained by taking the wall-nearest pressure values in the flow domain on the outside of the solid body, thereby assuming that the wall normal gradient of the pressure near the surface is negligibly small. The component of the drag and lift forces due to pressure distribution is calculated by integrating the pressure along the solid boundary. On the other hand, the shear-force component of the lift and drag forces is calculated from near the surface of the solid. The tangential velocity near the solid surface is obtained at the wall-nearest point outside of the body and is subsequently used to calculate the wall-shear stress at the cylinder surface.

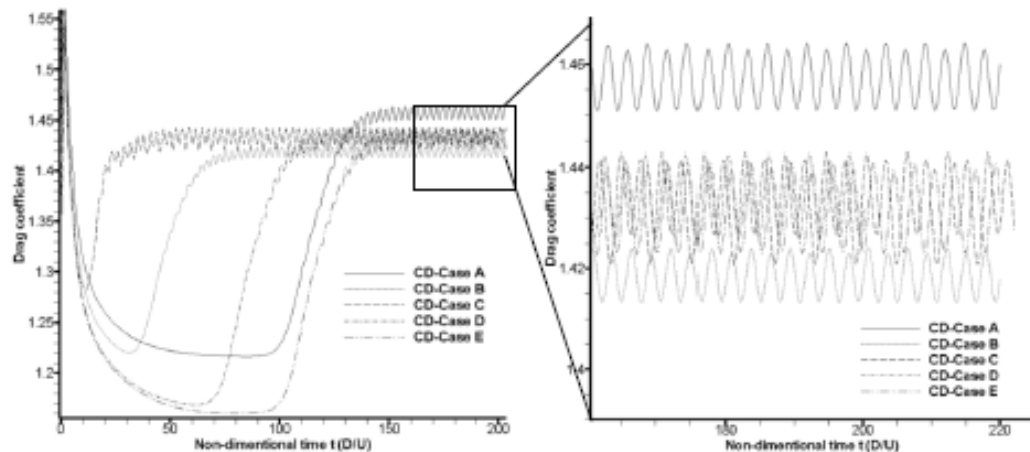


Figure 8: Drag coefficient for the flow around a stationary cylinder at $Re=100$, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation1; Case E, Bilinear interpolation2

Figure 8, shows a comparison of the drag coefficients obtained in calculations of flow over a stationary cylinder at $Re=100$ using various interpolation methods. It can be seen that in the cases C, D and E, (linear and Bilinear interpolation methods) the results were converging to a value of $C_D = 1.43$. However, Case A (without interpolation) leads to a higher drag coefficient, $C_D=1.46$ and Case B (weighting method) leads to a lower drag coefficient $C_D=1.42$. Once vortex shedding commenced all simulations were found to run at virtually the same speed (Table 1) showing that the computational effort needed for the interpolation was negligible. However, for a non-stationary cylinder, it is expected that the required repeated calculation of interpolation coefficients may lead to a reduction in execution speed.

Table 1: Real computational time, 20 vortex shedding

	Case A	Case B	Case C	Case D	Case E
Real time (s)	3231	3225	3379	4441	3383

Figure 8 (left) shows that Case C (linear interpolation) is the quickest method to develop vortex shedding, which indicates that if the implementation of boundary conditions with linear interpolation causes significant numerical noise. In Case E (proposed bilinear method),

on the other hand, the vortex-shedding instability kicks in much later evidencing that the level of numerical noise introduced by this type of interpolation is very small.

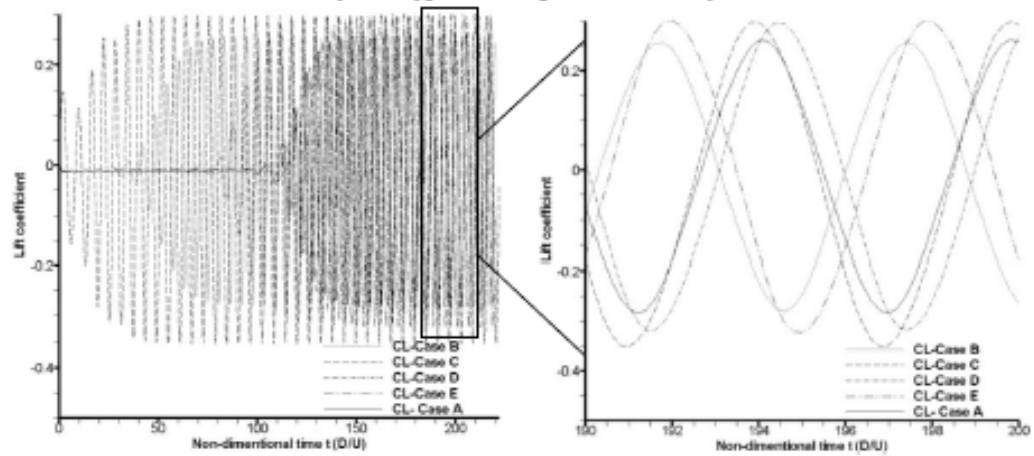


Figure 9: Lift coefficient for the flow around a stationary cylinder at $Re=100$, Case A, without interpolation; Case B: area weighting method; Case C, Linear interpolation method; Case D, Bilinear interpolation1; Case E, Bilinear interpolation2

Figure 9, shows the comparison of lift coefficients for the various interpolation cases. It can be seen from the figure that like the drag coefficient, the lift coefficient for the linear and bilinear cases are nearly the same (Case C, D and E) $CL=0.33$. However, the Case B (related to weighting area method) shows lower values for the lift ($CL=0.27$). In Case A (without interpolation), the lift due to shear stress is out of range, but the lift due to pressure is acceptable. The reason for the unacceptable results for the lift due to shear is that the velocity for case A was selected out of the boundary layer (the aim was to choose similar conditions for all cases).

Table 2: Strouhal number, lift and Drag coefficient for the flow around a stationary cylinder and $Re=100$.

simulation methods	Strouhal Number	Drag Coefficient	Lift coefficient
Case A	0.174	1.46	0.26
Case B	0.175	1.42	0.27
Case C	0.169	1.432	0.325
Case D	0.169	1.434	0.305
Case E	0.168	1.432	0.31
Park [22] fitted method	0.165	1.33	0.33
Williamson(exp.)[20]	0.166
Kim [22]	0.165	1.33	0.32
Roshko (exp.)[20]	0.164
Lai and Peskin [6]	0.165	1.4473	0.3299
Choi [7]	1.351	0.315
Corbalan & Souza [24]	1.44	0.31

Table 2 shows the comparison of the Strouhal number, lift and drag coefficient for various methods; from the experimental methods (Roshko and Williamson reported by [20]) to the body fitted mesh [21] and immersed boundary methods [22, 6, 23, 24], for the flow around a stationary cylinder at $Re=100$. It can be seen that the Strouhal number varies between 0.16 and 0.18; the Drag coefficient between 1.33 and 1.4473 and the lift coefficient between 0.31 and 0.33.

CONCLUSION

The objective of the present study is to compare the accuracy and expenses of different IB interpolation methods and select the most accurate and least expensive method for future use in simulations of flow around a deformable cylinder. A finite-volume method on a Cartesian grid with a staggered variable arrangement has been used. In this IB implementation the velocities near non-conforming boundaries were interpolated in the normal direction to walls, thereby considering the curvature of the geometry. The Strouhal number and Drag coefficient for different cases are reported. The results show a good agreement with the literature for most of the interpolation methods for the stationary cylinder. The drag coefficient and Strouhal number results for five different interpolation methods were compared it was shown that for a stationary cylinder at low Reynolds number, the interpolation method could affect the drag coefficient by a maximum 2% and the Strouhal number by maximum of 3%. In addition, the bi-linear interpolation method took about 2% more computational time per vortex shedding cycle in companion to the other methods.

REFERENCES

- [1] Hou G, Wang J. and Layton A, Numerical Methods for Fluid-Structure Interaction – A Review, *Commun. Comput. Phys* doi:10.4208/, Vol12,No. 2, pp. 337-377(2012).
- [2] Peskin C.S., Flow patterns around heart valves: A numerical method, *Journal Comput. Phys.* 10,252 (1972).
- [3] Mittal, R., and Iaccarino, G., "Immersed Boundary Methods," *Annual Review of Fluid Mechanics*, Vol 37, 2005, pp. 239–261. doi:10.1146/annurev.fluid.37.061903.175743
- [4] Saiki EM, Biringen S.. Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method. *Journal of Computational Physics* 123:450–65(1996).
- [5] Beyer RP, Leveque RJ., Analysis of a one dimensional model for the immersed boundary method. *SIAM Journal Numerical Analysis* 29:332–64(1992).
- [6] Lai MC, Peskin CS., An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, *Journal of Computational Phys.* 160:705–19 (2000).
- [7] Briscolini M. and Santangelo P., Development of the mask method for incompressible unsteady flows, *J. Computational Physics* 84-57,1988.
- [8] Mohd-Yusof J., Combined immersed boundaries/B-spline Methods for simulations of flows in complex geometries, *CTR Annual Research Briefs*, NASA AMES/Stanford University, 1997.
- [9] Verzicco R, Mohd-Yusof J., Orlandi P. & Haworth D., Large eddy simulation in complex geometric configurations using boundary body forces, *AIAA J.* 38(3), 427 (2000).

- [10] Iaccarino G, Verzicco R. Immersed boundary technique for turbulent flow simulations. *Appl Mech Rev.* 56:331–47(2003).
- [11] Ghas R, Mittal R, Lund TS. A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries. *AIAA Pap.* 2004-0080 (2004).
- [12] Mittal R, Bonilla C, Udaykumar HS. Cartesian grid methods for simulating flows with moving boundaries. In *Computational Methods and Experimental Measurements- XI*, ed. CA Brebbia, GM Carlomagno, P Anagnostopoulos(2003).
- [13] Mittal R, Seshadri V, Udaykumar HS. Flutter, tumble and vortex induced autorotation. *Theor. Comput. Fluid Dyn.* 17(3):165– 70(2004).
- [14] Udaykumar HS, Mittal R, StryW. Computation of solid-liquid phase fronts in the sharp interface limit on fixed grids. *J. Comput. Phys.* 153:534–74(1999).
- [15] Udaykumar HS, Mittal R, Rampungoon P, Khanna A. A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. *J. Comput. Phys.* 174:345–80(2001).
- [16] Harlow F.H. and Welsh J.E., Numerical Calculation of time-dependent viscous incompressible Flow of Fluid with Free surface, *The Physics of Fluids*, Vol 8, No.12, 1965
- [17] Ferziger J.H and Peric M, *Computational methods for fluid dynamics*, 3rd edition, Springer, 2002.
- [18] Fadun, E. A., Verzicco, R., Orlandi, P., and Mohd-Yusof, J., "Combined Immersed-Boundary Finite Difference Methods for Three- Dimensional Complex Flow Simulations," *Journal of Computational Physics*, Vol 161, No. 1, pp. 35–60 (2000).
- [19] Kang S., Iaccarino G., Moin P., Accurate immersed-boundary reconstructions for viscous flow simulations, *AIAA J.* 47 (7) (2009) 1750–1760.
- [20] Liu C., Zheng X., Sung C.H., Preconditioned multigrid methods for unsteady incompressible flows, *journal of computational physics* 139,359(1998).
- [21] Park J., Kwon K., and Choi H., Numerical solutions of flow past a circular cylinder at Reynolds number up to 160, *KSME international Journal*, vol 12, No. 6, 1998, pp.1200-1205.
- [22] Kim J., Kim D., Choi H., and immersed boundary finite volume method for simulations of flow in complex geometries, *journal of computational physics*, 171 (2001) 132.
- [23] Choi J.I, Oberoi R.C., Edwards J.R., Rosati J. A., An immersed boundary method for complex incompressible flows, *Journal of Computational Physics*, Volume 224, Issue 2, 10 June 2007, Pages 757-784, ISSN 0021-9991, 10.1016/j.jcp.2006.10.032.
- [24] Corbalan Gois E.R. and De Souza L.F., An Eulerian immersed boundary method for flow simulations over stationary and moving rigid bodies, *Journal of the Braz. Soc. Of Mech. Sci & Eng.* special issue 2010, vol XXXII, No. 5/477.