DEVELOPMENT AND STUDY OF A HIGH-RESOLUTION TWO-DIMENSIONAL RANDOM VORTEX METHOD

A THESIS

submitted by

PRABHU RAMACHANDRAN

for the award of the degree

of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF AEROSPACE ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

JUNE 2004

THESIS CERTIFICATE

This is to certify that the thesis titled **DEVELOPMENT AND STUDY OF A HIGH-RESOLUTION TWO-DIMENSIONAL RANDOM VORTEX METHOD**, submitted by **Prabhu Ramachandran**, to the Indian Institute of Technology, Madras, for the award of the degree of **Doctor of Philosophy**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. M. Ramakrishna	**Prof. S. C. Rajan**
Research Guide	Research Guide
Professor	Assistant Professor
Dept. of Aerospace Engineering	Dept. of Aerospace Engineering
IIT-Madras, 600 036	IIT-Madras, 600 036

Place: Chennai
Date: 14th June, 2004

ACKNOWLEDGEMENTS

I am extremely fortunate to have Professors S. C. Rajan and M. Ramakrishna as my advisors. Their guidance, support, concern and encouragement in both technical and non-technical matters have moulded me and most of my work. Through their association I imbibed the love of learning and research. I also learnt from their example the importance of being a good human being. They were also responsible in part for kindling my interest in the field during my Bachelors. My heartfelt thanks to them.

I cannot thank my parents enough for their love and support through the entire course of my life. During the course of my PhD, they bore the brunt of my erratic schedule and other idiosyncrasies with their characteristic love and understanding. I am also grateful to my brother and sister for their concern, understanding and support.

The faculty of the Aerospace engineering department at IIT-Madras have always been very supportive and helpful. In particular I thank Prof. P. Sriram for being so kind and helpful.

I thank my doctoral committee members for their support and guidance. I am also grateful to all those who supported the conversion of my registration from a Masters to a PhD and especially to those who made it happen.

The departmental office staff (administrative and technical) have been both cheerful and understanding. They have always been very helpful. My thanks to them for making my life easier.

Most of the computations were made on the cluster in the Computers and Fluids Laboratory at the Department of Aerospace Engineering. My thanks to Prof. Ramakrishna for giving me a free hand in setting up the lab and machines. Some of the computations were made in the FPGA lab cluster. I thank Prof. Vinita

Vasudevan and Mrs. Sumathy of the Electrical Engineering Department who gave me access to the FPGA lab machines. I am grateful for their help.

All of my computational work required the use of a large number of open source software packages. I am thankful to all the developers of such excellent open software. In particular I thank the hundreds of developers and maintainers of the Debian GNU/Linux distribution. Much of this work would not be possible but for their efforts.

I am grateful to S. Shashidhar who gave me some of the important papers that he had collected and also discussed his research with me. It was his initial work on the RVM that motivated the present work.

Mr. Nikhil V. Raj and Dr. Jayanarayanan Sitaraman were kind enough to mail me, on short notice, several important papers that were unavailable at the IIT-M library. My sincere thanks to them. I am also grateful to Dr. Shankar Subramaniam for giving our group a copy of his doctoral and post-doctoral theses several years ago.

None of this work would be possible but for the grace of my Guru and the Lord. My life has a meaning and purpose thanks to my Guru's advice. I cannot thank Him enough.

ABSTRACT

KEYWORDS: Random vortex method; high-resolution; panel methods; fast multipole methods; fast algorithms; object-oriented design.

The hybrid random vortex method (RVM) involving vortex blobs and vortex sheets is implemented and explored in this work. The method is used to perform high-resolution simulations of two-dimensional, incompressible, Navier-Stokes fluid flows.

The development of an efficient vortex based solver involves several important algorithms. The Adaptive Fast Multipole Method (AFMM) (Carrier *et al.*, 1988) is implemented to accelerate the computation of the velocity field. The need to handle generic geometries requires the use of a panel method to satisfy the no-penetration boundary condition. A higher-order panel method that eliminates the edge-effect is developed in this work. However, the method is computationally intensive. In order to accelerate the computation, the AFMM is extended to be used with vortex panels. This necessitates a generalization of the AFMM to handle passive particles.

In the RVM, particles move randomly in the vicinity of complex geometries. A fast algorithm is developed to handle this.

Particles of the same sign of circulation are merged. Those of opposite sign are annihilated. Fast algorithms are used in their implementation. This significantly reduces the number of particles without adversely affecting the accuracy of the simulation.

All of these fast algorithms share similar components. An object-oriented design for vortex methods is developed. This design abstracts several of the key algorithms and allows for a significant amount of code re-use.

The developed random vortex based solver is applied to the flow past an impulsively started circular cylinder. The method is studied as the different computational parameters are varied. Recommendations on the optimal choices of these parameters are made.

The random vortex method is generally considered to be a low-accuracy scheme that is unsuitable for high-resolution simulations. To show that this is not the case, the method is used to simulate the flow past an impulsively started circular cylinder in the Reynolds number range 40–9500. A new variance reduction scheme is introduced that is simple and uses the inherent parallelism of the RVM. This enables high-resolution simulations at high Reynolds numbers. The results are compared exhaustively with available data from computations using deterministic diffusion schemes. It is demonstrated that the RVM produces results comparable to the best available data. The computational effort necessary for the simulations appears comparable to that necessary for most deterministic schemes. Thus, it is clearly demonstrated that the RVM can be used to perform high-resolution simulations.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **iii**

LIST OF TABLES **xi**

LIST OF FIGURES **xx**

ABBREVIATIONS **xxi**

NOTATION **xxiii**

1 INTRODUCTION **1**

1.1 An overview of vortex methods 3

1.1.1 Advection 3

1.1.2 Diffusion 6

1.2 State of the RVM 9

1.3 Motivation 14

1.4 Objective of present work 15

1.5 Overview of thesis 17

2 PROBLEM FORMULATION **19**

2.1 Governing equations 19

2.2 Advection 22

2.2.1 Obtaining the velocity field from ω 22

2.2.2 Desingularized velocity kernels 25

2.2.3 Boundary conditions 28

2.2.4 Convective displacement of the vorticity 30

2.3 Diffusion 30

2.3.1 Random vortex method 31

2.4 Vortex sheets 32

3 NUMERICAL IMPLEMENTATION 36

3.1 Operator splitting 36

3.2 Discretization of vorticity 37

3.3 Advection 37

3.4 Diffusion 38

3.4.1 No-slip boundary condition 39

3.4.2 Sheet tagging 39

3.4.3 Conversion of sheets to blobs 40

3.4.4 Other techniques to satisfy no-slip 41

3.5 Merging and annihilation of vorticity 43

3.6 The panel method 44

3.6.1 Linear panels and the edge effect 46

3.6.2 Cubic panels 48

3.6.3 Solving the boundary value problem 50

3.6.4 Numerical results 52

3.7 Computational procedure 54

3.8 Ensemble averaging 55

3.9 The ensembled RVM 55

4 FAST SUMMATION TECHNIQUES 57

4.1 Introduction 57

4.2 AFMM for linear vortex panels 58

4.2.1 Multipole and local expansions 59

4.2.2 AFMM with passive particles 63

4.2.3 Numerical results 66

4.3 AFMM for higher order panels 69

4.3.1 Multipole and local expansions 69

4.3.2 Anderson's FMM without multipoles 78

4.3.3 Influence of cell size on accuracy 81

4.3.4 Numerical results 83
4.4 Summary 90

5 OTHER FAST ALGORITHMS 91

5.1 Particle motion and complex geometry 91
5.1.1 Domain decomposition and cell generation 93
5.1.2 Diffusion using the RVM 96
5.1.3 Numerical results 100
5.2 Computing the sheet velocity field 108
5.3 Conversion of sheets and blobs 109
5.4 Annihilation and merging of vorticity 110
5.4.1 A numerical study of annihilation and merging 110
5.4.2 Illustration of utility of annihilation 113
5.5 Summary 114

6 OBJECT-ORIENTED DESIGN 116

6.1 Introduction 116
6.2 Design overview 118
6.3 Notation for class diagrams 120
6.4 Design details 121
6.4.1 Geometry 121
6.4.2 Fluid particles 123
6.4.3 No-penetration BC: the panel method 126
6.4.4 Algorithms for hierarchically organizing particles 128
6.4.5 Adaptive Fast Multipole Method 131
6.4.6 Advection 133
6.4.7 Diffusion 133
6.4.8 Vortex methods 134
6.4.9 Unit testing 135
6.5 Summary 136

7 PARAMETRIC STUDY OF THE RVM 137

7.1 Computational method 137
7.2 Computational parameters 139
7.3 Determination of parameters 139
7.3.1 Scheme A 140
7.3.2 Scheme B 142
7.4 Primary diagnostics 143
7.5 Parametric study with scheme A 144
7.5.1 Variation of C 144
7.5.2 Variation of k 148
7.5.3 Variation of γ_{max} 153
7.5.4 Variation of R_a 157
7.6 Parametric study with scheme B 163
7.6.1 Variation of C 163
7.7 Parametric study using ensemble averaging 164
7.7.1 Ensemble averaging 166
7.7.2 Parametric study 167
7.8 Study of the ERVM 176
7.9 Summary 178

8 RESULTS AND DISCUSSION 180
8.1 Flow past a circular cylinder 180
8.1.1 $Re = 40$ 181
8.1.2 $Re = 550$ 187
8.1.3 $Re = 1000$ 195
8.1.4 $Re = 3000$ 200
8.1.5 $Re = 9500$ 210
8.2 Discussion 220

9 SUMMARY AND CONCLUSIONS 228
9.1 Contributions made 228
9.2 Conclusions 229
9.3 Suggestions for future work 231

A THE FAST MULTIPOLE METHOD 233
A.1 Background 233
A.2 Body-cell treecodes 235
A.2.1 Generating the tree 236
A.2.2 Multipole expansions 237
A.2.3 Shifting multipole expansions 239
A.2.4 The body-cell algorithm 240
A.3 Adaptive Fast Multipole Method (AFMM) 242
A.3.1 Local expansions 243
A.3.2 Shifting local expansions 244
A.3.3 General idea 244
A.3.4 Definitions and notation 245
A.3.5 The adaptive fast multipole algorithm 247
A.3.6 Optimizations 251
A.3.7 Numerical results 252
A.4 Summary 253

B COMPUTATION OF DIAGNOSTIC QUANTITIES 254
B.1 Computation of diagnostics 254
B.1.1 Forces and moments 254
B.1.2 Force computation: vortex momentum 255
B.1.3 Force computation: pressure and skin friction 264
B.1.4 Vorticity distribution 269
B.1.5 Optimal smoothing of vorticity 278
B.1.6 Streamlines, streaklines and path lines 285
B.1.7 Velocity vector plots 286
B.1.8 Surface vorticity distribution 286
B.1.9 Pressure distribution 287

LIST OF TABLES

2.1 Smoothing functions 27

4.1 CPU time for velocity of N cause particles on M effects. 66

4.2 Comparison of CPU times for cubic, flat, and fast panel methods. 66

5.1 Geometric complexity of the body. 102

5.2 Error, standard deviation and number of particles as the annihilation radius, R_a is varied. No merging is performed. 111

5.3 Error, standard deviation and number of particles as the merging radius, R_m is varied. No annihilation is performed. 112

5.4 Error, standard deviation and number of particles as the merging radius, R_m is varied and $R_a = 0.02$. 113

7.1 Parameters chosen as C is varied for scheme A. 145

7.2 Error in circulation, E_Γ and maximum circulation, Γ_{max} for different cases. 147

7.3 Error as compared to KL and noise level as Δt is varied for scheme A. 149

7.4 Number of panels, N and Δt for different k values given that $C \approx 0.4$. 152

7.5 Number of blobs and sheets at $T \approx 10$ for various values of k. Also shown is the error as compared to KL and the noise level in the curves. 152

7.6 Number of blobs and sheets at $T \approx 10$ for various values of γ_{max}. Also shown are the error and noise level in C_d versus T. 156

7.7 Number of blobs and sheets at $T = 6$ for various values of R_a. Also shown are the error and noise level in the curve for C_d. 159

7.8 Values of k_1 for scheme B, given corresponding k values from scheme A in Table 7.4. 163

7.9 Values of Δt and corresponding C value for scheme B. Also shown are the error and noise level in the curve for C_d. 164

7.10 The error, standard deviation and noise level for different number of trial runs. 166

7.11 Various combinations of parameters investigated. 168

7.12 The error, standard deviation, noise level and number of parameters for the different cases presented in Table 7.11. 169

7.13 The error, standard deviation and noise level for different number of trial runs for case D4. 171

7.14 The error, standard deviation, noise level and number of parameters for the variation of Δt for case C9 in Table 7.11. 171

7.15 The error, standard deviation, noise level and number of parameters for the variation of k_1 for case C9 in Table 7.11. 172

7.16 The error, standard deviation, noise level and number of parameters for the variation of γ_{max} for case C9 in Table 7.11. 173

7.17 The error, standard deviation, noise level and number of parameters for the variation of R_a for case C9 in Table 7.11. 173

7.18 The error, standard deviation, noise level and number of parameters for the different perturbations on case C9 in Table 7.11. The term RK4 refers to Runge-Kutta fourth order integration. 174

7.19 The error, standard deviation, noise level and number of parameters for combinations of parameters when using Strang discretization. 175

7.20 The error, standard deviation, noise level and average number of particles for different values of n_{sync} with n_{proc} fixed at 6. 178

7.21 The error, standard deviation, noise level and average number of particles for different values of n_{proc} with n_{sync} fixed at 5. 178

LIST OF FIGURES

2.1 Problem specification employing primitive variables. 20

2.2 Problem specification in vorticity-velocity form. 21

2.3 Velocity field due to different vortex blobs. 27

2.4 Illustration of a vortex sheet. 35

3.1 Reflection and conversion of vortex sheets and blobs. 41

3.2 A flat panel in the complex plane. 46

3.3 A flat panel in the $z' = (z - z_1)e^{-i\theta}$ plane. 46

3.4 A cubic panel having chord length λ in the z' plane. 48

3.5 Vorticity distribution for different types of panel attachments. . 51

3.6 Comparison of the error E versus distance from the surface of a circular cylinder obtained by using flat and cubic panels. 53

4.1 Hierarchical mesh for the flow past a circular cylinder with 400 panels and 10000 passive particles for case 1. 67

4.2 Hierarchical mesh for the flow past a circular cylinder with 400 panels and 10000 passive particles for case 2. 67

4.3 Comparison of the error E versus distance from the surface of a circular body obtained by using the fast panel method, the flat panel method, and the cubic panel method. 68

4.4 Higher order panel having a chord length λ in the z plane. . . . 70

4.5 Illustration of radius of convergence for panels. 74

4.6 Illustration of a panel in a cell and the location of maximum error. The cell C contains a panel at the corner. The cells $X1$ and $X2$ are cells where the X_b special local expansion is performed, $L1, L2, L3$ are cells where local expansions and multipole expansions due to C are computed. 82

4.7 Maximum relative error due to the multipole expansion, local expansion and X_b interaction versus change in β for the AFMM for higher order panels. Note that the curves for the multipole expansion and the local expansion coincide. 83

4.8 Plot of the maximum relative error due to various methods versus distance from the surface of the cylinder. 400 panels are used for the cylinder. The curves for the fast cubic method and Anderson's scheme coincide with the curve for the direct computation. . . . 84

4.9 Time taken versus number of particles in a square region. 400 panels are used for the cylinder. $\beta = 7.0$. Note that the curves for the fast cubic method (higher order panels) and the hybrid cubic/flat method almost coincide. 86

4.10 Time taken as n_e is varied. n_e is reduced by increasing A. 400 panels are used for the cylinder. $\beta = 7.0$. 87

4.11 Time taken as the number of panels is increased. 40000 points in a uniform grid are considered inside a square region. $\beta = 7.0$. . . 88

5.1 Discretization of the numerical layer into viscous boxes. 93

5.2 A viscous box passing through various cells. 94

5.3 Schematic of a line passing through various cells. The line joining z_1 and z_2 originates in the cell C and passes through the descendents of its colleague C_1. 96

5.4 Illustration for algorithm used to check intersections of a particle path with a panel. 98

5.5 Illustration for algorithm used to check intersections of blobs having a finite core radius, δ, with a panel. 99

5.6 Geometry chosen for study at level 1 and 2. 102

5.7 Simulation of diffusion outside the body at level 3 of its construction. 49152 vortex blobs are used in the simulation. The blobs are initially distributed on the outer surface of the body. The non-dimensional width of the body is 1. The figure is a plot at the end of 50 time steps. 103

5.8 Simulation of diffusion inside the body at level 3 of its construction. 49152 vortex blobs are used in the simulation. The blobs are initially distributed on the inner surface of the body. The non-dimensional width of the body is 1. The figure is a plot at the end of 50 time steps. 103

5.9 Variation of time taken for 50 time steps versus geometric complexity of the body. 49152 vortex blobs are used in the simulation and 768 equal sized panels are used for the body. 105

5.10 Variation of time taken for 50 time steps versus number of panels used. 49152 vortex blobs are used in the simulation and the complexity of the body is 43 (i.e. the body is at level 3 of the fractal construction). 105

5.11 Variation of time taken for 50 time steps versus number of blobs used in the simulation. 576 equal sized panels are used in the simulation and the complexity of the body is 43 (i.e. the body is at level 3 of the fractal construction). 106

5.12 Vorticity distribution for flow past the body at $T = 1.32$. 107

5.13 Vorticity distribution for flow past a complex shape at $T = 2.5$. 108

5.14 Flow past a circular cylinder at $Re = 3000$, with no annihilation at $T \approx 2$. 336340 vortex blobs and 65323 sheets are present. 114

5.15 Flow past a circular cylinder at $Re = 3000$, with sheet annihilation at $T \approx 2$. 43826 vortex blobs and 6897 sheets are present. . . . 115

5.16 Flow past a circular cylinder at $Re = 3000$, with sheet and blob annihilation at $T \approx 2$. 36552 vortex blobs and 6031 sheets are present. 115

6.1 Various entities involved in a vortex method. 118

6.2 A schematic of the vortex method algorithm. Red items require special techniques for computational efficiency. 119

6.3 Legend for the symbols used in class diagrams. 120

6.4 Geometry consisting of various bodies and parts. 122

6.5 Illustration of viscous boxes for curved geometric elements. . . . 122

6.6 Class diagram for the fluid elements. 124

6.7 Class diagram and containership relation for the fluid element managers. 125

6.8 Class diagram for the vortex panels and related classes. 126

6.9 Class diagram for the panel method and related classes. 127

6.10 Class diagrams for the different flavors of cells. 129

6.11 Relationship of fast algorithms and the cell managers. 134

7.1 C_d versus T for cases 1 to 4. 149

7.2 Zoomed view of C_d versus T for cases 1 to 4. 150

7.3 C_d versus T for cases 5 to 8. 150

7.4 Zoomed view of C_d versus T for cases 5 to 8. 151

7.5 Number of blobs versus $1/k$ at the end of $T = 10$. 153

7.6 C_d versus T as k varies between 1.5 to 4. 154

7.7 C_d versus T as k varies between 5 to 7. 154

7.8 Vorticity contours at $T \approx 5$ with $k = 1.5$. 155

7.9 Vorticity contours at $T \approx 5$ with $k = 3$. 155

7.10 Smoothed C_d versus T as γ_{max} is changed. 156

7.11 Smoothed C_d versus T as γ_{max} is changed. 157

7.12 Vorticity contours at $T \approx 5$ with $\gamma_{max} = 0.00625$. 158

7.13 Vorticity contours at $T \approx 5$ with $\gamma_{max} = 0.0125$. 158

7.14 Smoothed C_d versus T as R_a is varied between 0 to 0.025. . . . 160

7.15 Smoothed C_d versus T as R_a is varied between 0.05 and 0.4. . . 160

7.16 Vorticity contours at $T \approx 5$ with $R_a = 0$. 161

7.17 Vorticity contours at $T \approx 5$ with $R_a = 0.0125$. 161

7.18 Vorticity contours at $T \approx 5$ with $R_a = 0.05$. 162

7.19 Smoothed C_d versus non-dimensional time for different ensemble averages as compared to Koumoutsakos and Leonard (1995). . . 167

8.1 Impulsively started flow past a circular cylinder. 180

8.2 Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 40$. Solid line is the total drag force. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 182

8.3 Body vorticity for impulsively translated cylinder at $Re = 40$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 183

8.4 Vorticity flux for impulsively translated cylinder at $Re = 40$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 184

8.5 Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 40$. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 185

8.6 Iso-vorticity contours for an impulsively translated cylinder at $Re = 40$. 186

8.7 Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 550$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995), Shankar (1996) and Ploumhans and Winckelmans (2000). 188

8.8 Pressure and friction drag versus T for impulsively translated cylinder at $Re = 550$. Solid line: pressure drag, dashed line: friction drag. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 188

8.9 Radial velocity along the axis of symmetry on the rear side of an impulsively translated cylinder at $Re = 550$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996). 189

8.10 Body vorticity for impulsively translated cylinder at $Re = 550$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 190

8.11 Vorticity flux for translated cylinder at $Re = 550$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 191

8.12 Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 550$. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000). 192

8.13 Iso-vorticity contours for an impulsively translated cylinder at $Re = 550$. 193

8.14 Streamlines for an impulsively translated cylinder at $Re = 550$. 194

8.15 Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 1000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Smith and Stansby (1988, 1989). 195

8.16 Pressure and friction drag versus T for impulsively translated cylinder at $Re = 1000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 197

8.17 Body vorticity for impulsively translated cylinder at $Re = 1000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 198

8.18 Vorticity flux for impulsively translated cylinder at $Re = 1000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 199

8.19 Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 1000$. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 200

8.20 Iso-vorticity contours for an impulsively translated cylinder at $Re = 1000$. 201

8.21 Streamlines for an impulsively translated cylinder at $Re = 1000$. 202

8.22 Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 3000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995); Shankar (1996); Ploumhans and Winckelmans (2000). 203

8.23 Pressure and friction drag versus T for impulsively translated cylinder at $Re = 3000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 203

8.24 Radial velocity along the axis of symmetry on the rear side of an impulsively translated cylinder at $Re = 3000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996). 204

8.25 Body vorticity for impulsively translated cylinder at $Re = 3000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 205

8.26 Vorticity flux for impulsively translated cylinder at $Re = 3000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 206

8.27 Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 3000$. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000). 207

8.28 Iso-vorticity contours for an impulsively translated cylinder at $Re = 3000$. 208

8.29 Streamlines for an impulsively translated cylinder at $Re = 3000$. 209

8.30 Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 9500$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995); Shankar (1996). 210

8.31 Pressure and friction drag versus T for impulsively translated cylinder at $Re = 9500$. Solid line and dashed lines correspond to the solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 211

8.32 Radial velocity along the axis of symmetry on the rear side of an impulsively translated cylinder at $Re = 9500$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996). 212

8.33 Tangential velocity component along radial lines at different angles for an impulsively translated cylinder at $Re = 9500$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996). 213

8.34 Body vorticity for impulsively translated cylinder at $Re = 9500$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 214

8.35 Vorticity flux for impulsively translated cylinder at $Re = 9500$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995). 215

8.36 Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 9500$. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000). 216

8.37 Iso-vorticity contours for an impulsively translated cylinder at $Re = 9500$. 217

8.38 Streamlines for an impulsively translated cylinder at $Re = 9500$. 219

8.39 Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 3000$. $n_{proc} = 2$, $n_{sync} = 5$ and $\gamma_{max} = 0.05$. 226

A.1 A cluster of m point vortices considered for the multipole expansion. 238

A.2 Colleagues and associates of cell b. Cells marked c are colleagues (and therefore associates) and those marked a are only associate cells (and not colleagues). 246

A.3 The five different lists for a cell b. The cells marked ‘u’ belong to the U_b list, ‘v’ to the V_b list and similarly for the other lists. . . 246

A.4 The U_b, V_b, and Z_b lists for cell b. The cells marked ’u’ belong to the U_b list, ‘v’ to the V_b list and ’z’ to the Z_b list. 248

A.5 Comparison of CPU time taken as the number of vortices is varied for the direct method and the AFMM. The dotted line plotting the AFMM results is for a case where the vortices are distributed on an ellipse. The solid line plots the variation when the particles are distributed randomly inside a unit square using uniform deviates. 253

B.1 Vortex momentum versus time for flow past an impulsively started cylinder at $Re = 3000$. The solid line plots I_x and the dashed line plots I_y. 256

B.2 Vortex momentum versus time for flow past an impulsively started cylinder at $Re = 3000$. This is a zoomed view of the I_x curve shown in figure B.1. 257

B.3 Force along x axis for flow past an impulsively started cylinder at $Re = 3000$. The force was obtained by taking central differences directly on the data. 257

B.4 Force along x axis obtained by using a sliding window average. 258

B.5 Force along x axis. The solid curve is obtained using a least squares fit with a 7th order polynomial for each 0.5 second interval of data with 50% overlap. The dashed curve is obtained using a sliding average (window size = 11, applied 5 times). 259

B.6 A test function used to obtain the best scheme for removing noise from data. 260

B.7 Error (E_1) for the sliding window average with different window sizes applied various number of times. 261

B.8 Error (E_2) for the sliding window average with different window sizes applied various number of times. 261

B.9 Error (E_1) for piecewise polynomial least squares fit of the data as different order polynomials are used and as the number of points in the interval are varied. 262

B.10 Error (E_2) for piecewise polynomial least squares fit of the data as different order polynomials are used and as the number of points in the interval are varied. 263

B.11 Error (E_2) versus different number of points per piece as the number of sample points on the curve is varied. Piecewise polynomials are fit to the data using a least squares fit. 264

B.12 Illustration of sheet inside a viscous box. 266

B.13 ω_{wall} versus time for Stokes' first problem. The exact and computed results are plotted for different kinematic viscosities, ν, when $N = 500$. 268

B.14 ω_{wall} versus time for Stokes' first problem. The exact and computed results are plotted for different numbers of sheets used, N, when $\nu = 0.001$. 268

B.15 Grid in relation to the interpolation schemes used. 272

B.16 Illustration of various parameters used for interpolation. 273

B.17 Interpolation for a point P using the XY distribution. Temporary particles are first created along the $y =$ const. grid lines (along the dashed lines) and these particles are then distributed to the grid points along the $x =$ const. lines. 274

B.18 Smoothed function as N varies, with $\nu = 0.1$ and $\sigma = 1$ at $t = 10$. 279

B.19 Smoothed function as N varies, with $\nu = 0.001$ and $\sigma = 0.1$ at $t = 10$. 279

B.20 Approximate function obtained using linear interpolation as N varies, with $\nu = 0.001$ and $\sigma = 0.1$ at $t = 10$. 281

B.21 Approximate function obtained using smoothing and second order interpolation using twice the optimal values of ϵ and h. $\nu = 0.001$, $\sigma = 0.1$, $t = 10$ and $N = 2500$. 281

B.22 Approximate function obtained using second order interpolation with different grid spacing. $\nu = 0.001$, $\sigma = 0.1$, $t = 10$ and $N = 2500$. 282

B.23 Approximate function obtained using second order interpolation along with Laplace smoothing for different grid spacings h. $\nu = 0.001$, $\sigma = 0.1$, $t = 10$, $N = 2500$ and $h_{opt} = 0.072$. 283

B.24 Vorticity along the x axis for a two-dimensional diffusion problem along with Laplace smoothing for different grid spacings h. Third order interpolation is used. $\nu = 0.001$, $t = 10$, $N = 2500$ and $h_{opt} = 0.0592$. 284

ABBREVIATIONS

AFMM	Adaptive Fast Multipole Method
CCSVM	Corrected Core-Spreading Vortex Method
CFD	Computational Fluid Dynamics
ERVM	Ensembled Random Vortex Method
FMM	Fast Multipole Method
OOD	Object-Oriented Design
PSE	Particle Strength Exchange
RVM	Random Vortex Method
VRT	Vorticity Redistribution Technique

NOTATION

$\hat{e}_n$	Unit vector normal to boundary
$\hat{e}_s$	Unit vector tangential to boundary
$\vec{F}$	Force of fluid on body, $kg\ m/s^2$
f_δ	Smoothing function
h	Grid spacing
h_{num}	Numerical layer height
$\vec{I}$	Hydrodynamic impulse or vortex momentum, $kg\ m/s$
$\hat{k}$	Unit vector perpendicular to the $x-y$ plane
N	Number of particles
n	Co-ordinate along normal direction to body surface
n_{proc}	Number of processors used
n_{sync}	Number of time steps after which the data from processors are synchronized
p	Pressure of the fluid, Pa
$\vec{r}$	Position vector
R_a	Non-dimensional radius of circle inside which particles are considered for annihilation
R_m	Non-dimensional radius of circle inside which particles are considered for merging
Re	Reynolds number
s	Co-ordinate along tangential direction to body surface
Δt	Time step, s
t	Time in seconds, s
u	Component of the velocity vector along x direction, m/s
U_e	Velocity component along tangential direction at the edge of the numerical layer, m/s
$\vec{V}$	Velocity of the fluid, m/s

$\vec{V}_0$	Initial velocity field, m/s
$\vec{V}_B$	Velocity of the boundary, m/s
v	Component of the velocity vector along y direction, m/s
$\vec{x}$	Position vector
x, y	Cartesian co-ordinate directions
z	Complex co-ordinate
δ	Core radius of vortex blob
$\delta(\vec{x})$	Dirac distribution (delta function)
Φ	Complex potential
ϕ	Velocity potential
Γ	Circulation of a vortex element (anti-clockwise is positive), m^2/s
γ	Intensity of vortex sheet, m/s
γ_{max}	Maximum strength of an individual vortex sheet element, m/s
λ	Length of released sheet element, m
μ	Viscosity of fluid, $kg/m/s$
ν	Kinematic viscosity of fluid, m^2/s
ρ	Density of the fluid, kg/m^3
$\vec{\omega}$	Vorticity of the fluid, $1/s$
ω	Magnitude of fluid vorticity along the $\hat{k}$ direction, $1/s$
ω_0	Initial vorticity field, $1/s$
ψ	Stream function

CHAPTER 1

INTRODUCTION

A large class of fluid flows can be modeled by the Navier-Stokes (NS) equations. These are second order, non-linear, partial differential equations. Analytical solutions are available only for a few simple cases. Consequently, solutions for general fluid flow problems must be obtained using a numerical method. Due to the non-linear nature of the NS equations and the variety of imposed boundary and initial conditions, obtaining numerical solutions is a challenging task. Computational fluid dynamics (CFD) is a field dedicated to the numerical simulation of fluid flow problems. Traditional CFD involves partitioning the underlying physical space into a fixed computational grid (structured or unstructured). The NS equations are discretized on this grid. Generating the grid can involve considerable effort for complex geometries. For unsteady problems the grid may have to be adaptive in order to capture the features accurately. This can be problematic. The computational grid can also introduce numerical difficulties in the problem.

Vortex methods avoid some of the difficulties with grid-based methods and offer an interesting alternative approach. A vorticity-velocity formulation is employed. The vorticity field is discretized into particles of vorticity (rather than specifying it on a fixed grid). The velocity field is obtained from the vorticity field and the particles are tracked in a Lagrangian fashion based on the NS equations. The approach eliminates the need to discretize the convection terms in the NS equations. Since the vorticity is tracked in terms of displacements of individual vortex particles, there is no need for a fixed grid on which the governing differential equations and the unknowns are discretized. The key advantages of vortex methods are listed below.

- Vorticity is discretized only in regions where it is non-zero.
- A Lagrangian tracking of vorticity is used. Traditional grid generation is not necessary.

- Numerical diffusion due to the discretization of the convective terms on a grid is eliminated.
- The method is self adaptive. Vortex particles are created where needed (typically on the boundary) and as the particles of vorticity move, the natural evolution of the vorticity field is captured automatically.
- The scheme is time-accurate and therefore ideally suited for unsteady flows.
- Boundary conditions at infinity are automatically taken care of.
- The solution procedure has a strong physical appeal.
- Speziale (1987) has shown that vortex methods can be easily used to solve the NS equations in non-inertial reference frames.
- The method has been successfully applied to a large number of problems. Some of these will be discussed later in this chapter.

The vortex method was originally developed for incompressible, single-phase flows. However, the method has been extended to handle compressible flows (Anderson, 1985; Krishnan and Ghoniem, 1992; Eldredge *et al.*, 2002*a*). Two-phase particulate flows along with two-way coupling are simulated by Chen and Marshall (1999) using vortex methods.

Vortex methods do have some limitations and practical difficulties in their implementation. Some of these are listed below.

- Non-trivial to implement for general flows.
- Handling large number of particles requires special and fairly involved techniques to improve computational efficiency.
- Handling generic boundaries efficiently and accurately is challenging.
- Higher order accuracy for long times requires the use of specialized techniques.
- Three-dimensional flows are not simple extensions of the two-dimensional case.

The present work is concerned with the vortex method applied to two-dimensional, incompressible, Navier-Stokes fluids.

1.1 An overview of vortex methods

In a vortex method the vorticity field is discretized into particles of vorticity. The particulate vorticity is tracked as per the governing differential equations. Vortex methods usually employ fractional steps or an operator splitting technique. During each time step, the governing differential equation is solved in two sub-steps — advection and diffusion. The advection step involves the convection of the vorticity based on the velocity field, which is obtained from the vorticity field. The convective displacements of the particles are determined from the known velocity field. The vorticity is then diffused in the second sub-step. The order of the two sub-steps may be varied. The boundary conditions imposed are that the fluid velocity on the surface of solid walls should equal that of the local surface. At infinity the velocity due to compactly supported vorticity is naturally zero, fulfilling the required boundary condition. In the following, more details on advection and diffusion are provided along with a brief survey of relevant literature.

1.1.1 Advection

The velocity field is required in order to advect the vortex elements. Given a vorticity field, the velocity field is obtained using the Biot-Savart law applied to the vorticity. Obtaining an accurate velocity field efficiently is the primary prerequisite for advection.

The earliest vortex methods used point vortices to discretize the vorticity resulting in the point vortex method. This method is second order (Goodman *et al.*, 1990) and converges to solutions of the Euler equations. However, the point vortex has a singular vorticity and velocity field. This causes numerical problems[1] since poor velocity approximations are obtained in the vicinity of the point vortices (Beale and Majda, 1985).

The difficulty created by the singularity in the velocity field can be mitigated

---

[1] Historically, research in this area has been driven by the evolution and roll-up of a vortex sheet. When simulating this problem with the point-vortex model, researchers obtained poor results (see Krasny (1987, p. 124)).

by desingularizing the point vortex. This can be done by using a scheme such as the cloud-in-cell (CIC) algorithm[2] to compute the velocity as done by Christiansen (1973), Tryggvason (1988, 1989) and others. However, the method requires the use of a grid.

Chorin and Bernard (1973) proposed that the point vortex be desingularized into vortex blobs in order to obtain bounded velocities and better approximations to the vorticity field. The resulting method is called the vortex blob method. This approach is grid-free and widely used. The method has been successfully used to model a variety of flows (discussed subsequently) including the evolution and roll-up of vortex sheets (Krasny, 1986, 1987).

While the point vortex method is second order (Goodman *et al.*, 1990), higher order methods can be constructed if vortex blobs are used. However, higher order accuracy for long times also requires special attention. Perlman (1985) found that a large amount of smoothing is necessary. Others use a re-zoning (also known as remeshing or regridding) technique (Beale and Majda, 1985; Nordmark, 1991, 1996; Koumoutsakos, 1997). This method uses a grid to re-initialize the vortex particles occasionally. Ghoniem *et al.* (1988) and Krishnan and Ghoniem (1992) introduce new particles along the principal strain direction while Meiburg (1989) adaptively modifies the smoothing to ensure a well organized distribution of particles. Remeshing and excessive smoothing can be avoided by using a better scheme for the velocity quadrature (Russo and Strain, 1994; Strain, 1996, 1997).

The method of contour dynamics (Zabusky *et al.*, 1979) is an alternative to the vortex blob method applicable to the Euler equations with piecewise constant vorticity patches. In this approach, the motion of the iso-contours of vorticity can be used to determine the evolution of the vorticity patches. The method therefore tracks the boundary of the vorticity contours.

In the present work, the vortex blob method is used and no remeshing procedure is employed.

---

[2] The method uses a grid to interpolate the vorticity and rapidly solves the Poisson equation on this grid. This eliminates the singular velocity field.

Fast summation schemes

In vortex blob methods, the velocity induced on a vortex particle, P, can be computed by finding the influence of all the particles on P. For N particles, this is an $O(N^2)$ process. This can be prohibitively expensive since N is large when high accuracy is desired. The $O(N^2)$ computation can be accelerated using a variety of fast summation techniques (Anderson, 1986; Greengard and Rokhlin, 1987; Carrier *et al.*, 1988; Anderson, 1992; Draghicescu and Draghicescu, 1995). The adaptive fast multipole method (AFMM) due to Carrier *et al.* (1988) reduces the operation count to $O(N)$. The present work uses this algorithm. These fast summation algorithms work by hierarchically identifying clusters of particles and evaluating particle-cluster and cluster-cluster interactions in addition to particle-particle interactions. These techniques are discussed in considerable detail in appendix A and chapter 4.

Boundary conditions

The velocity field must satisfy a no-penetration boundary condition on solid walls (also called the no-flow boundary condition). Different techniques are used to satisfy this condition, the most common being the method of images. For simple geometries, the computational domain can be transformed to the plane of a circle or a half plane. Image vortices are introduced in the transformed plane to satisfy the boundary condition. This technique is not suitable for arbitrary geometries. Some researchers (Sethian, 1984) have used grid-based fast Poisson solvers to satisfy the boundary condition. However, this approach introduces a grid in an otherwise grid-free simulation. A panel method (Katz and Plotkin, 1991) or a boundary element method is an alternative requiring a grid only on the boundary of the domain. The method is applicable to complex geometries. In this technique, singularities are distributed on the body surface in order to satisfy the boundary condition. If high accuracy is desired, it is necessary to use a higher order panel method. In the present work a panel method that uses cubic geometry elements is developed. This is discussed in section 3.6. It is also possible to improve the computational efficiency of the panel method by adapting the fast multipole method suitably. These techniques are discussed in chapter 4.

1.1.2 Diffusion

Diffusion of vorticity occurs as a result of the viscosity of the fluid. This viscosity has two significant effects on a vortex based simulation.

- The creation of new vorticity to satisfy the no-slip boundary condition.
- The diffusion of existing vorticity in the fluid.

Viscosity ensures that the fluid on the wall adheres to it. Vortex methods either introduce new vorticity along the boundary or introduce a vorticity flux on the boundary in order to satisfy this boundary condition. In some cases a thin sheet of vorticity having appropriate strength at the boundary is created. This sheet of vorticity is discretized into a layer of vortex blobs as done by (Chorin, 1973; Clarke and Tutty, 1994; Lin *et al.*, 1997) and various others. In order to model the boundary layer accurately, Chorin (1978) proposed a "vortex sheet" model where vorticity near the boundary is represented as thin vortex sheet elements. This approach is used by (Chorin, 1978, 1980; Ghoniem *et al.*, 1982; Cheer, 1989; Choi *et al.*, 1988) and various others. Other variants of anisotropic vortex sheet elements (Teng, 1982; Bernard, 1995; Huyer and Grant, 1996; Marshall and Grant, 1996) have also been developed and used. Koumoutsakos *et al.* (1994) instead diffuse the vorticity from the sheet to nearby vortex particles such that the boundary condition is satisfied. This approach has been used by (Koumoutsakos and Leonard, 1995; Cottet *et al.*, 2000; Ploumhans and Winckelmans, 2000) and others who use the particle strength exchange (Degond and Mas-Gallic, 1989) diffusion scheme. A recently developed technique uses impulse elements (Summers, 2000) in the form of vortex monopoles or vortex dipoles generated at the boundary to satisfy the boundary condition.

There are several techniques to simulate diffusion of vorticity in a manner suitable to particle based methods. The earliest of these is the method based on random walks due to Chorin (1973). In this method, each vortex particle performs a random walk. The steps of the random walk are drawn from a Gaussian distribution having zero mean and a variance dependent on both the kinematic viscosity, ν, and time step. The resulting vortex method is called the random vortex method (RVM). The method produces results with statistical noise. It is known to have a low rate of convergence (Roberts, 1985), $O(\sqrt{\nu/N})$, where N is the number of particles. It is, however, very easy to implement for simple geometries. The method is also completely grid-free. One (rarely mentioned) computational difficulty with the method is in the efficient handling of the random motion of the particles in the presence of arbitrary complex geometries. The present work describes a scheme to address this in section 5.1. A detailed account of simulating diffusion with random walk methods for various applications is presented in Ghoniem and Sherman (1985). Chang (1988) explores the connection between the RVM and stochastic differential equations. He shows how the method can be extended to be used in Runge-Kutta time stepping schemes. Heemink and Blokland (1995) use the connection to stochastic differential equations to show how spatially varying viscosity can be handled using the random walk algorithm. Fogelson and Dillon (1993) show that particle methods using random walks converge when smoothed carefully. However, a large number of particles is necessary to obtain accurate results.

Deterministic diffusion schemes seek to eliminate the statistical noise in the RVM and improve its low rate of convergence. The core spreading technique was proposed by Leonard (1980). This technique involves increasing the size of the vortex blob using the heat kernel. The method is grid-free. However, Greengard (1985) proved that the core-spreading model does not converge when used in the context of the NS equations. Rossi (1996) showed that if the blobs were split into smaller blobs once they grew beyond a particular size, then the core-spreading model does converge. Shiels (1998) successfully used the core spreading model along with the modifications suggested by Rossi (1996, 1997) to obtain high resolution results. Kida (1998) developed a modified version of the core-spreading model and compared it with the original core spreading model and the random vortex method.

Clarke and Tutty (1994) employ a diffusion scheme called the “diffusion velocity method”. The scheme was originally proposed by Degond and Mustieles (1990) and also by Ogami and Akamatsu (1991). In this scheme, diffusion is simulated by adding a diffusive velocity to the convective velocity and then displacing the particles. The method is grid-free. In practice (Clarke and Tutty, 1994) the method does not diffuse vorticity correctly when the vortices are spaced far apart.

The particle strength exchange (PSE) technique of Degond and Mas-Gallic (1989) is a deterministic diffusion scheme that has been well studied and applied. The method works by replacing the Laplacian in the diffusion equation by an integral operator. The method requires a reasonable amount of overlap between the vortex particles in order to be stable and accurate. This necessitates the use of a remeshing procedure where the vorticity is interpolated onto a grid and the vortex particles are regenerated. The method has been successfully used to produce high resolution simulations (Koumoutsakos and Leonard, 1993, 1995; Shiels, 1998; Ploumhans *et al.*, 1999; Ploumhans and Winckelmans, 2000). The PSE originally required that the vortex blobs have the same core-radius but recent work due to Cottet *et al.* (2000) and Ploumhans and Winckelmans (2000) enable the use of spatially varying cores. Eldredge *et al.* (2002*b*) have extended the idea of using integral operators to approximate arbitrary derivatives of fluid properties defined on a collection of particles. This idea is used in their development of the compressible vortex method (Eldredge *et al.*, 2002*a*).

Fishelov (1990) also proposed a deterministic vortex method that is similar to the PSE. The spatial derivatives in the viscous term are applied on the smoothing function. This differentiated smoothing function is convolved with the vorticity to obtain the new vorticity distribution. This method also requires that the cores of the vortex blobs overlap and therefore requires remeshing. Bernard (1995) used this scheme to develop a deterministic vortex sheet algorithm for boundary layer flows.

Shankar and van Dommelen (1996*a*); Shankar (1996) have developed a grid-free scheme called the vorticity redistribution technique (VRT). The method is similar to the PSE in that the strength (circulation) of the vortex particles are distributed to neighboring particles. However, a system of equations is solved to obtain the correct fractions of the circulation to be distributed. The method is not dependent on the smoothing function used by the vortex blobs. Further, it imposes no requirement on the spatial ordering of the blobs and requires no remeshing. The method has been used for high resolution simulations (Shankar, 1996; Shankar and van Dommelen, 1996*b*; Takeda *et al.*, 1999). The method also appears to require fewer particles than equivalent PSE computations to obtain results of similar accuracy. The most attractive feature of this method is its grid-free nature.

Apart from purely Lagrangian vortex method implementations there are hybrid schemes that also make use of a fixed grid during the computation. Cottet (1990) uses the PSE technique to simulate diffusion. A fixed grid is used to correct the errors due to particle distortion. The method of Qian and Vezza (2001) uses a cell-centered finite volume method to solve the integral equations for the vorticity. Only the cells with non-zero vorticity are used in this scheme. The velocity field used in the flux term evaluation is computed using the Biot-Savart law. This computation of the velocity is accelerated using a fast summation technique (Van Dommelen and Rundensteiner, 1989). The results compare well with the available high-resolution vortex method simulations.

Leonard (1980) provides an early review of vortex methods. Puckett (1991) also reviews vortex methods and discusses the RVM and its implementation in detail. Cottet and Koumoutsakos (2000) present details of vortex methods with an emphasis on some of the more recent developments.

The focus of the present work is the hybrid random vortex method (RVM) using both vortex blobs and vortex sheets.

1.2 State of the RVM

In this section the various results and applications of the RVM are reviewed.

The RVM as devised by Chorin (1973, 1978) has been used to simulate a variety of fluid flows. Chorin (1980) used the method to study boundary layer instability in two and three dimensions.

Porthouse and Lewis (1981) independently devised the random vortex method to investigate boundary layer flows and also to simulate the wake of an impulsively started cylinder. Their solutions were of a low resolution but illustrated the usefulness of vortex methods.

Ghoniem *et al.* (1982) modeled turbulent flow in a combustion tunnel using the RVM. The method of images along with conformal transformation was used to satisfy the no-flow boundary condition. The velocity field was obtained using the direct $O(N^2)$ method. The simulations were in good qualitative agreement with experimental results.

Cheer (1983) used the hybrid vortex sheet/blob method to simulate the flow past a circular cylinder and a Joukowski airfoil at Reynolds numbers of 1000 and 2000. The velocity field was obtained using the direct method. The method of images was used to satisfy the solid-wall boundary condition. Conformal transformation was used in the case of the airfoil problem. The computations were fairly coarse and the results compared well with known experimental results.

Sethian (1984) simulated turbulent combustion inside a rectangular chamber using the random vortex method. A fast Poisson solver was used to impose the solid wall boundary condition. The direct method was used to compute the velocity. Both vortex sheets and blobs were used.

Smith and Stansby (1985) simulated wave-induced bed flows in spatially periodic domains using the RVM. The vortex-in-cell technique was used to compute the velocity field. A boundary integral method was used to compute the strength of the vortex sheet to be shed at the bed wall. The flow due to different imposed wave motions and the flow over a rippled bed were studied. The results for the averaged flow field agreed well with existing theoretical and experimental data.

Ghoniem and Gagnon (1987) studied the recirculating flow for the case of a backwards facing step at low Reynolds number. Sethian and Ghoniem (1988) also studied the flow past a backward facing step for a range of Reynolds numbers for different values of the relevant computational parameters. The direct method was used for the velocity. Both vortex sheets and blobs were used. The results were compared with experimental data and the agreement was found to be good.

Ghoniem and Ng (1987) studied the dynamics of a forced shear layer inside a channel. The forced shear layer was generated in the center of the channel and since there were no solid walls of interest in the immediate vicinity of the shear layer, no vortex sheets were released along the walls of the channel.

Smith and Stansby (1988) simulated the flow past an impulsively started circular cylinder. They used a vortex-in-cell technique to compute the velocity field. No vortex sheets were used. Their results were of a low resolution. However, they obtained good agreement with various experimental and theoretical results.

Choi *et al.* (1988) simulated the initial transient flow inside a lid-driven square cavity using the RVM for Reynolds numbers of 2000, 5000 and 10000. The no-penetration condition was satisfied using a velocity field which was obtained in terms of elliptic integrals. The results were in good agreement with those of a finite difference code. Notably, they also discussed the available techniques to remove "parasitic elements"[3] from the computation.

Cheer (1989) simulated the flow past an impulsively started circular cylinder for various Reynolds numbers. She used the method of images to satisfy the no-penetration boundary condition. The direct method was used to compute the velocities. Results for various Reynolds numbers were compared with experimental and computational data. Good agreement was obtained. However, the computations were of a low resolution.

Smith and Stansby (1989) introduced a scheme where particles that strike a solid wall are deleted (absorbed by the body) instead of being reflected. Both reflection and absorption were studied. It was seen that absorption produced better results with fewer particles. They simulated the flow past an impulsively started cylinder at various Reynolds numbers. Heat transfer was modeled by

---

[3] In the RVM, it is commonly found that negatively signed sheets are created in regions where only positive vorticity is physically valid. Similarly, positive vorticity is sometimes released in regions of negative vorticity. These oppositely signed vortices are called "parasitic elements".

the generation of “temperature particles” that are similar to vortex particles. The results compared well with results from a finite difference simulation and experiments.

Summers (1989) simulated incompressible boundary layer flow with an external velocity given by $U(x) = U_0 x^m$, where m is real. The time-averaged steady state solutions from these simulations were compared with the similarity solutions of the Falkner-Skan equations. Good agreement was obtained excepting for the cases where m is negative (adverse pressure gradient) and close to the case where full separation is expected.

Puckett (1989) studied the vortex sheet method in considerable detail. A new sheet creation approach was proposed and the use of higher order sheets was also proposed and studied. He proved theoretically that one time step of the sheet creation process and random walk consistently approximates the exact solution of the heat equation. He performed an extensive numerical study and compared his results with the Blasius profile. He found that obtaining optimal results required that the parameters be carefully chosen. Recommendations regarding the choice of sheet creation scheme, integration scheme and sheet tagging were also made.

Baden and Puckett (1990) computed the flow inside a rectangular domain with a stationary vortex at its center. They used Anderson’s method of local corrections (Anderson, 1986) to accelerate the computation of the velocity and the random vortex method to simulate diffusion. They also accelerated the computation of the velocity of the vortex sheets by organizing the sheets into “bins” based on their spatial location.

Kim and Flynn (1995) simulated the flow past multiple bodies using the RVM. A panel method was used to enforce the no-penetration boundary condition. The adaptive fast multipole method (Carrier *et al.*, 1988) was used for the computation of the velocity of the vortex blobs. Vortex sheets were also used. They were interested more in the flow patterns rather than quantitative measures. Hence, their results were of a low resolution.

Huyer and Grant (1996) used the RVM in the context of anisotropic vorticity elements. They used thin rectangular vortex elements with constant vorticity. They also employed a fast multipole method to accelerate the computations.

Mortazavi *et al.* (1996) study the convergence of the RVM as some of the critical computational parameters are varied. The flow of an internal jet in a confined region expanding into a chamber is studied. They find that the parameters must be carefully chosen to avoid numerical accuracy and stability related problems. However, no concrete guidelines are provided on how the parameters are to be chosen optimally.

The RVM has also been used recently without vortex sheets by researchers to simulate the flow past a circular cylinder (Clarke and Tutty, 1994), pitching airfoil (Lin *et al.*, 1997), the flow past square and rectangular cylinders (Taylor and Vezza, 1999*b*,*a*). Of these, Clarke and Tutty (1994) used the $O(N \log\ N)$ method due to Van Dommelen and Rundensteiner (1989) for the computation of the velocity field. Taylor and Vezza (1999*b*,*a*) used a different $O(N \log\ N)$ method for the velocity computation. Clarke and Tutty (1994) used a higher order panel method (parabolic geometry) to satisfy the no-penetration boundary condition. They also used a diffusion velocity method to simulate diffusion near the body. The RVM was used in the wake region. Taylor and Vezza (1999*b*,*a*) used a linear geometry panel method. Their simulations are in general of a lower resolution as compared to those of Koumoutsakos and Leonard (1995) who uses a deterministic diffusion scheme.

At the Department of Aerospace Engineering, IIT-Madras, Shashidhar (1998) investigated the vortex blob method and applied the RVM to the flow past zero thickness flat plates. He also studied the problem of vortex sheet roll-up in some detail. However, his implementation of the RVM was not designed for generic flows and used a conformal transformation to apply the no-penetration condition. His work also did not employ fast multipole methods. The work focussed on the theory and implementation of the RVM.

1.3 Motivation

Most vortex method implementations either do not use panel methods or use lower order panel methods which have difficulties due to the edge effect. From a study of the literature, it seems that only Clarke and Tutty (1994) and Takeda *et al.* (1999) have implemented a higher-order (parabolic) panel method to eliminate the edge effect and obtain high accuracy. However, they do not use fast multipole methods to compute the panel velocities rapidly. This makes the method fairly inefficient. Thus, it is important to develop an accurate and efficient panel method.

In the RVM, particles move randomly. These particles should not penetrate solid walls. Enforcing this efficiently in the presence of complex geometries is difficult. This problem does not appear to have been addressed in the literature. Developing a fast algorithm for this purpose is of importance.

An efficient and accurate vortex based solver for general flows involves several algorithms. Some of these algorithms can be quite complex and a significant amount of programming effort is required to implement them. Therefore, it is difficult to develop the solver from scratch. However, due to the Lagrangian nature of vortex methods there is significant commonality between the key algorithms. Abstracting these common algorithms minimizes code duplication. It also makes the code much easier to understand and extend. The use of object-oriented design makes this abstraction possible. Thus, it is of great use to apply object-oriented design principles to this problem.

From a review of the existing literature, it is clear that the RVM has been widely applied over the years. However, it also appears that most of the computations are of a low resolution and only represent engineering approximations. This is understandable because of the constraints on computational power and memory at the times these studies were made. Many of these studies also predated the fast multipole algorithms.

All the published computations employing the RVM to simulate the initial transients for the impulsively translated cylinder (Cheer, 1983, 1989; Smith and

Stansby, 1988, 1989) are of a low resolution as compared to similar computations where a deterministic diffusion scheme is used. Koumoutsakos and Leonard (1995), Shankar (1996), Shiels (1998) and Ploumhans and Winckelmans (2000) all simulate the same problem using deterministic diffusion schemes[4]. In particular, the results of Smith and Stansby (1988) compare very poorly with any of their computations.

Shankar (1996) also compares his results with unpublished data from the RVM. The agreement is not very good. He does mention that the RVM is a usable method that requires a large number of particles. However, there is a clear lack of published evidence that demonstrates that the RVM can be used to perform high-resolution computations. If anything, all existing results point to just the opposite. Due to these reasons, some researchers argue that the RVM is only a low-resolution technique and cannot be reliably used for high-resolution computations.

The present work seeks to address these concerns.

1.4 Objective of present work

There are two components of the present work. The first is to develop and implement a two-dimensional vortex method employing the random vortex method for diffusion. The second is to study the random vortex method and investigate the possibility of performing high-resolution simulations with it. The following are the key goals of the work.

- Develop and implement the essential algorithms and tools useful for 2D vortex methods.
- Provide an object-oriented design and implementation for vortex methods.
- Use the Random Vortex Method (RVM) (Chorin, 1973, 1978) to handle diffusion.
- Study the issues involved in obtaining high-resolution results with the RVM. Importantly, it is shown that the method is indeed capable of high-resolution.

---

[4] The results of Koumoutsakos and Leonard (1995) are considered to be direct numerical simulations (DNS) for the flow past an impulsively started circular cylinder.

The following important algorithms are developed and implemented in this work.

- The adaptive fast multipole method (AFMM) (Carrier *et al.*, 1988).
- An efficient and accurate panel method for fairly generic geometries.
- Efficiently performing diffusion using the RVM in the presence of complex geometries.
- Efficient merging of particles based on a proximity criterion.

The AFMM (Carrier *et al.*, 1988) is implemented using an object-oriented design that allows for easy extension and code re-use.

A cubic panel method that eliminates the edge effect is developed. The method is both accurate and efficient. Efficiency is achieved by the use of a fast multipole algorithm to compute the panel velocity field. This also necessitates a generalization of the AFMM to handle passive particles.

A fast algorithm to handle a large number of particles moving randomly in the presence of complex geometries is developed.

A simple technique called "annihilation" is used to eliminate the parasitic elements that are generated in RVM simulations. The method deletes nearby vortex sheets of opposite sign based on a proximity criterion. This enables for a large reduction in the number of particles and improves the accuracy of the simulation. The merging of like signed vorticity is also used to reduce the number of interacting particles. Almost an order of magnitude reduction in the number of particles is observed when these techniques are used.

Object oriented design (OOD) allows software developers to manage large and complex software. It allows one to develop code with a high degree of abstraction that enables easier understanding and extension of the developed software. Apart from the obvious benefits in the programming of such codes, OOD sometimes enables a clearer understanding of the physical problem. Thus the present work provides an object-oriented design and implementation for vortex methods.

In section 1.3 it was seen that the RVM is considered to be a low-resolution technique useful only for engineering approximations. In the present work it is shown that it is certainly possible to use the RVM to perform high-resolution simulations. For high Reynolds numbers, a simple variance reduction technique is introduced. The method takes advantage of the parallelism inherent in the RVM. This enables the present work to obtain excellent results for high Reynolds numbers. Exhaustive comparisons are made with the available results of Koumoutsakos and Leonard (1995), Shankar (1996) and Ploumhans and Winckelmans (2000) for the flow past an impulsively started cylinder. Some of the results of Shiels (1998) are also used for comparison. These comparisons conclusively demonstrate that the RVM can indeed be used for high-resolution simulations. It is to be noted that the objective of these comparisons is not to claim that the RVM is superior to a deterministic scheme but to illustrate that accurate and useful results are certainly possible with the RVM.

1.5 Overview of thesis

Chapter 2 provides theoretical details on vortex methods with specific reference to the schemes used in the present work. The numerical implementation and overall details of the algorithms are discussed in chapter 3. Issues with advection and diffusion are discussed. The panel method as a means to ensure the no-penetration boundary condition is also developed. The new variance reduction scheme is also introduced. Chapter 4 discusses the application of the AFMM to accelerate the computation of the velocity field due to the vortex panels used in the panel method. A new method that elegantly handles passive particles with the AFMM is also developed. Chapter 5 discusses some of the other useful fast algorithms used in the present work. Notably the algorithm to handle randomly moving particles in the vicinity of complex geometries is discussed. Chapter 6 discusses the object-oriented design for vortex methods developed in the present work. The chapter also illustrates the power and benefits that were obtained by using an object-oriented design. In chapter 7 the flow past an impulsively started circular cylinder is used as a benchmark problem. The different parameters involved in the method are varied and their influence on the results is studied. Several recommendations on the optimal choice of the parameters are made. In chapter 8, results for the simulation of the flow past an impulsively started circular cylinder at Reynolds numbers in the range 40–9500 are presented. The results of chapter 7 are used to choose the parameters suitably. The results obtained with the RVM are compared with available computational data. Finally, chapter 9 summarizes the work and mentions many issues, problems and improvements that could be pursued in the future.

Appendix A provides a detailed description of the adaptive fast multipole method (AFMM). The theoretical and implementation details of the method are brought out. Appendix B describes how common diagnostic quantities are computed.

CHAPTER 2

PROBLEM FORMULATION

In this chapter, the governing equations for the problem considered are presented. Some basic theoretical details of vortex methods are provided. The advection and diffusion steps of the vortex method are discussed in some detail.

2.1 Governing equations

Consider an incompressible, Navier-Stokes fluid with constant kinematic viscosity, ν and density ρ. As shown in Fig. 2.1, let the domain of the fluid be denoted as D and the boundary of an immersed solid body as B. At any point, let p be the pressure and $\vec{V}$ the velocity of the fluid. The fluid flow is assumed to be isothermal. The flow over the solid body moving in the unbounded fluid is governed by the conservation of mass and linear momentum equations. Conservation of mass results in the following equation,

$$\text{div}\vec{V} = 0. \tag{2.1}$$

Conservation of linear momentum results in the following equation,

$$\frac{\partial \vec{V}}{\partial t} + \vec{V} \cdot \text{grad } \vec{V} = -\frac{1}{\rho}\text{grad } p + \nu \nabla^2 \vec{V}. \tag{2.2}$$

As shown in Fig 2.1, the boundary and initial conditions are,

$$\vec{V}(\vec{r}, 0) = \vec{V}_0(\vec{r}), \tag{2.3a}$$

$$\vec{V}(\vec{r}, t) = 0 \quad \text{as } \vec{r} \rightarrow \infty, \tag{2.3b}$$

$$\vec{V}(\vec{r}, t) = \vec{V}_B \quad \text{on } B. \tag{2.3c}$$

$\vec{V}_B$ is the velocity of the boundary, and in general is a function of space and time. The boundary condition on B ensures no-penetration and no-slip on solid walls. These equations along with the boundary conditions are illustrated in Fig. 2.1.

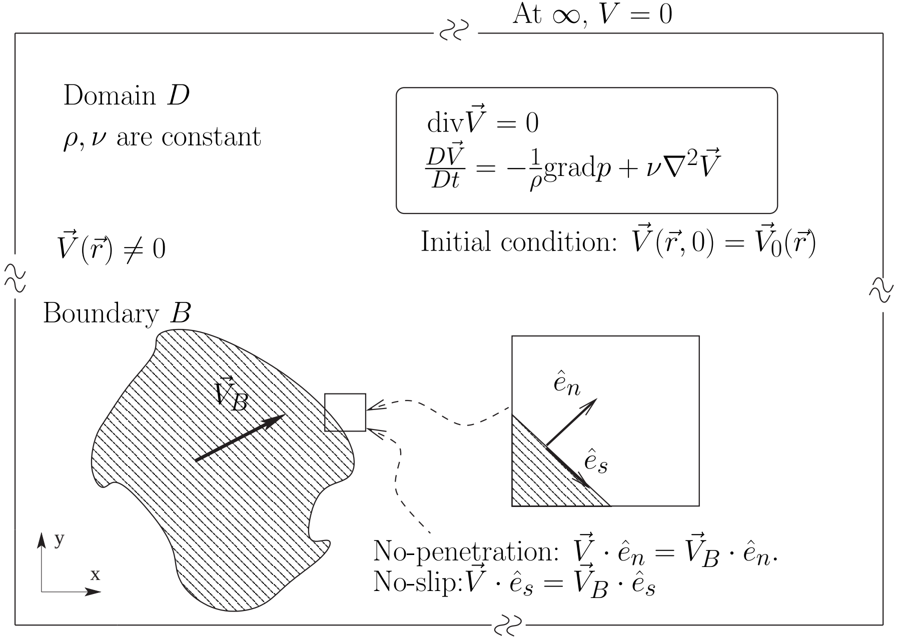
At ∞, $\vec{V}=0$
Domain D
ρ, ν are constant
$\mathrm{div}\vec{V}=0$
$\frac{D\vec{V}}{Dt} = -\frac{1}{\rho}\mathrm{grad}p + \nu\nabla^2\vec{V}$
$\vec{V}(\vec{r}) \neq 0$
Initial condition: $\vec{V}(\vec{r},0) = \vec{V}_0(\vec{r})$
Boundary B
$\vec{V}_B$
$\hat{e}_n$
$\hat{e}_s$
y
x
No-penetration: $\vec{V}\cdot\hat{e}_n = \vec{V}_B\cdot\hat{e}_n$.
No-slip: $\vec{V}\cdot\hat{e}_s = \vec{V}_B\cdot\hat{e}_s$

Figure 2.1: Problem specification employing primitive variables.

In vortex methods the above equations are represented in terms of the vorticity, $\vec{\omega}$. Vorticity is defined as the curl of the velocity, $\vec{\omega} = \text{curl } \vec{V}$. If the curl operator is applied to equation (2.2) the following equation is obtained,

$$\underbrace{\frac{D\vec{\omega}}{Dt}}_{\text{Advection}} = \underbrace{\vec{\omega}\cdot\text{grad }\vec{V}}_{\text{Stretching}} + \underbrace{\nu\nabla^2\vec{\omega}}_{\text{Diffusion}}, \tag{2.4}$$

where,

$$\frac{D}{Dt} = \frac{\partial(\ )}{\partial t} + \vec{V}\cdot\text{grad }(\ ),$$

is the material derivative. As can be seen, the pressure term is eliminated from this equation.

For a two-dimensional flow $\vec{\omega}$ is normal to the plane of the flow and therefore $\vec{\omega} = \omega\hat{k}$. Consequently, the vortex stretching term in equation (2.4) is zero. Therefore, the governing differential equations in vorticity-velocity form for a two-

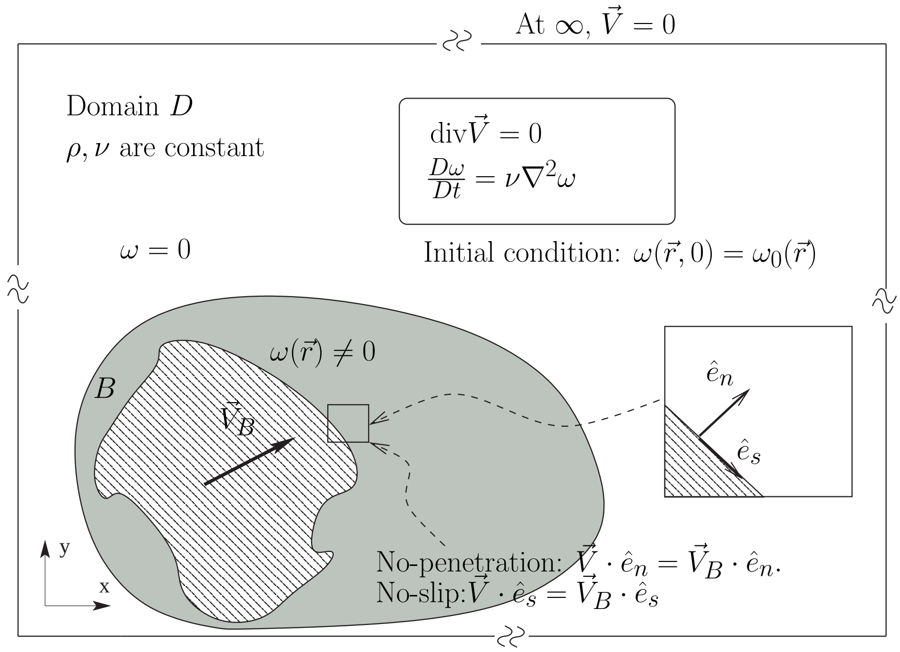
At ∞, $\vec{V}=0$
Domain D
ρ, ν are constant
$\mathrm{div}\vec{V}=0$
$\frac{D\omega}{Dt}=\nu\nabla^2\omega$
$\omega=0$
Initial condition: $\omega(\vec{r},0)=\omega_0(\vec{r})$
$\omega(\vec{r})\neq 0$
B
$\vec{V}_B$
$\hat{e}_n$
$\hat{e}_s$
y
x
No-penetration: $\vec{V}\cdot\hat{e}_n=\vec{V}_B\cdot\hat{e}_n$.
No-slip:$\vec{V}\cdot\hat{e}_s=\vec{V}_B\cdot\hat{e}_s$

Figure 2.2: Problem specification in vorticity-velocity form.

dimensional fluid flow are as follows.

$$\frac{\partial\omega}{\partial t}+\vec{V}\cdot\text{grad }\omega=\nu\nabla^2\omega, \qquad \text{(Vorticity transport)} \tag{2.5a}$$

$$\omega=\text{curl }\vec{V}\cdot\hat{k}, \qquad \text{(Vorticity)} \tag{2.5b}$$

$$\text{div }\vec{V}=0, \qquad \text{(Mass conservation)} \tag{2.5c}$$

$$\omega(\vec{r},0)=\omega_0(\vec{r}), \qquad \text{(Initial condition)} \tag{2.5d}$$

$$\vec{V}(\vec{r},t)=0 \qquad \text{as } \vec{r}\to\infty, \qquad \text{(Infinity BC)} \tag{2.5e}$$

$$\vec{V}(\vec{r},t)\cdot\hat{e}_n=\vec{V}_B\cdot\hat{e}_n \qquad \text{on } B, \qquad \text{(No-penetration BC)} \tag{2.5f}$$

$$\vec{V}(\vec{r},t)\cdot\hat{e}_s=\vec{V}_B\cdot\hat{e}_s \qquad \text{on } B. \qquad \text{(No-slip BC)} \tag{2.5g}$$

where $\hat{e}_n$ and $\hat{e}_s$ are the normal and tangential unit vectors on B. These are illustrated in Fig. 2.2. In the figure it can be seen that the region of non-zero vorticity for the flow past a body in an unbounded fluid is compact.

In the context of vortex methods equation (2.5a) is solved in two steps.

$$\frac{D\omega}{Dt}=0, \qquad \text{(Advection)} \tag{2.6a}$$

$$\frac{\partial\omega}{\partial t}=\nu\nabla^2\omega. \qquad \text{(Diffusion)} \tag{2.6b}$$

Equation (2.6a) is the advection equation. Equation (2.6b) is the diffusion equation. During each time step, the advection and diffusion equations are solved separately. This is called the method of fractional steps or an operator/viscous splitting technique (Beale and Majda, 1981). This approach enables a Lagrangian method of solution for the equations.

2.2 Advection

The advection equation (2.6a) corresponds to the Euler equation in vorticity form. It implies that vorticity is a fluid property and remains constant along a particle path. The advection equation along with the boundary conditions are given by,

$$\begin{aligned}
&\frac{\partial \omega}{\partial t} + \vec{V} \cdot \text{grad}\ \omega = 0, && && \text{(Advection)} && (2.7a)\\
&\omega = \text{curl}\ \vec{V} \cdot \hat{k}, && && \text{(Vorticity)} && (2.7b)\\
&\text{div}\ \vec{V} = 0, && && \text{(Mass Conservation)} && (2.7c)\\
&\omega(\vec{r}, 0) = \omega_0(\vec{r}), && && \text{(Initial condition)} && (2.7d)\\
&\vec{V}(\vec{r}, t) = 0 && \text{as } \vec{r} \rightarrow \infty, && \text{(Infinity BC)} && (2.7e)\\
&\vec{V}(\vec{r}, t) \cdot \hat{e}_n = \vec{V}_B \cdot \hat{e}_n && \text{on } B. && \text{(No-penetration BC)} && (2.7f)
\end{aligned}$$

Given the vorticity field, the velocity field that satisfies the boundary conditions is to be found. This velocity field can then be used to advect the vorticity using the advection equation (2.7a).

2.2.1 Obtaining the velocity field from ω

The velocity field can be computed from the vorticity field as follows. If ψ is the stream-function then it is known that

$$\vec{V} = (u, v) = \left(\frac{\partial \psi}{\partial y}, -\frac{\partial \psi}{\partial x} \right) = (\partial_y, -\partial_x)\psi. \tag{2.8}$$

From the definition of vorticity it can be seen that

$$\nabla^2\psi = -\omega. \tag{2.9}$$

This equation can be solved to obtain the stream-function from the known vorticity. The boundary conditions are satisfied by using the homogenous part of the solution. Therefore, ψ is written as,

$$\psi = \psi_\omega + \psi_p, \tag{2.10}$$

where ψ_p is the homogenous solution (corresponding to the irrotational flow) and ψ_ω is the particular solution (corresponding to the rotational flow). ψ_p is chosen such that the boundary conditions are satisfied. Thus, the velocity field can be split into two parts, $\vec{V}_\omega$, the rotational velocity arising from ψ_ω and $\vec{V}_p$, the potential part from ψ_p. Since $\vec{V}_p$ is irrotational, there exists a corresponding potential function, ϕ, satisfying $\nabla^2\phi = 0$, that can be used to obtain the velocity field. Therefore,

$$\vec{V}(\vec{r}, t) = \vec{V}_\omega + \vec{V}_p = \vec{V}_\omega + \text{grad}\phi \tag{2.11}$$

The manner in which $\vec{V}_p$ is found is discussed in section 2.2.3. This section discusses how $\vec{V}_\omega$ is obtained.

If $G(\vec{x})$ is the Green's function for the $-\nabla^2$ operator,

$$\psi_\omega = \int_{\mathcal{R}_n} G(\vec{x} - \vec{x'})\omega(\vec{x'})d^n x' = G * \omega \tag{2.12}$$

where $*$ denotes convolution. In two-dimensions, $n = 2$ and $d^n x' = dx'\ dy'$. From equation (2.8),

$$\begin{aligned}\vec{V}_\omega &= ((\partial_y, -\partial_x)G) * \omega \\ &= K * \omega \\ &= \int_{\mathcal{R}_n} K(\vec{x} - \vec{x'})\omega(\vec{x'})d^n x'. \end{aligned} \tag{2.13}$$

This equation is the Biot-Savart law. K is called the Cauchy velocity kernel. For the two dimensional case, K is given as

$$K(x,y) = \frac{(-y,x)}{2\pi r^2}, \tag{2.14}$$

where $r^2 = x^2 + y^2$. In two-dimensions it is convenient to use complex notation. If $z = x + iy$ is the complex co-ordinate, with $i = \sqrt{-1}$, the Cauchy kernel in complex co-ordinates is given by

$$K(z) = u - iv = \frac{-i}{2\pi z}. \tag{2.15}$$

Using equation (2.14) or (2.15) in equation (2.13), the velocity field is obtained from the vorticity field. It is to be noted that the above Cauchy velocity kernel corresponds to the velocity field of a point vortex with unit strength and located at the origin. This velocity field is singular at the origin and the corresponding vorticity field is given by the Dirac distribution (referred to commonly as the delta function).

The integral in equation (2.13) can be discretized to obtain a numerical approximation of the velocity field. If N point vortices, at positions $\vec{x}_j$, with circulations Γ_j, are used to discretize the vorticity field then the discretized vorticity is given by

$$\omega(\vec{x}) = \sum_{j=0}^{N} \delta(\vec{x} - \vec{x}_j)\omega_j h^2 = \sum_{j=0}^{N} \delta(\vec{x} - \vec{x}_j)\Gamma_j, \tag{2.16}$$

where $\delta(\vec{x})$ is the Dirac distribution, ω_j is the vorticity at the position $\vec{x}_j$, h is the grid spacing used to discretize the vorticity. Hence, the discretized velocity field is given by

$$\vec{V}_\omega(\vec{x}, t) = \sum_{j=0}^{N} K(\vec{x} - \vec{x}_j)\Gamma_j. \tag{2.17}$$

This is called the "point vortex method". The method is known to be second order (Goodman *et al.*, 1990) accurate. However, numerical problems with the method requires desingularization of the Cauchy kernel. This is discussed in the next section.

2.2.2 Desingularized velocity kernels

The Cauchy velocity kernel poses computational difficulties[1]. The singular velocity field of the point vortices introduces large errors in the vicinity of the point vortices (Beale and Majda, 1985). This motivates the use of desingularized velocity kernels. Additionally, higher order accuracy is possible if vortex blobs are used.

Chorin (Chorin, 1973; Chorin and Bernard, 1973) proposed the use of a modified, desingularized velocity kernel, K_δ, to obtain the velocity field as

$$\vec{V}_\omega(\vec{x}, t) = \sum_{j=0}^{N} K_\delta(\vec{x} - \vec{x}_j)\Gamma_j. \tag{2.18}$$

This desingularized kernel can be obtained by introducing smoothing functions f_δ, which are approximations to the Dirac distribution. Consider the function,

$$f_\delta(r) = \frac{1}{\delta^2} f(r/\delta)$$

where f is a symmetric, real valued smoothing function that approaches a Dirac delta as the parameter $\delta \to 0$ such that

$$2\pi \int_0^\infty f(r) r dr = 1.$$

The parameter δ is called the *core radius* or *cutoff radius*. The smoothing function f_δ is also called a core function or cutoff function. The smoothing function need not be symmetric but usually is assumed to be so. In the following, it is assumed to be symmetric.

The cutoff function serves to obtain a better approximation of the vorticity field. The discretized vorticity is obtained similar to equation (2.16) as

$$\omega(\vec{x}) = \sum_{j=0}^{N} f_\delta(\vec{x} - \vec{x}_j)\omega_j h^2. \tag{2.19}$$

---

[1] See Krasny (1987, p. 124) for a discussion of the problems with point vortices as applied to the simulation of the evolution and roll-up of vortex sheets.

The desingularized velocity kernel is found by convolving the cutoff function with the Cauchy kernel which results in,

$$K_\delta = K * f_\delta.$$

Following the derivation in Beale and Majda (1985), a simple relationship between the Cauchy kernel and the smoothing function can be obtained if f_δ is radially symmetric. Define $G_\delta = G * f_\delta$. Since G is the Green's function for the $-\nabla^2$ operator it is easy to see that $\nabla^2(G * f_\delta) = -f_\delta$. Since f_δ is radially symmetric,

$$\nabla^2 G_\delta = \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial G_\delta}{\partial r}) = -f_\delta.$$

Therefore,

$$\frac{\partial G_\delta}{\partial r} = -\frac{1}{r}\int_0^r f_\delta \; r' \; dr' \tag{2.20}$$

From equation (2.20), and (2.13) it is easy to see that

$$\begin{aligned} K_\delta &= (\partial_y, -\partial_x)G_\delta \\ &= \left(\frac{y}{r}, \frac{-x}{r}\right)\frac{\partial G_\delta}{\partial r} \\ &= \frac{(-y, x)}{2\pi r^2} \;\; 2\pi \int_0^r f_\delta \; r' \; dr' \\ &= K(x, y) \;\; k(r) \end{aligned} \tag{2.21}$$

Therefore, for a given smoothing function, f_δ, it is possible to obtain $k(r)$. With this, the desingularized velocity field can be obtained using equation (2.18). This discretization of the vorticity and velocity field results in the "vortex blob method".

Some commonly used smoothing functions are given in Table 2.1. The velocity field induced by these blobs are all radially symmetric and some of them are plotted in Fig. 2.3. In the figure, the blob core radius $\delta = 0.5$. The Chorin blob is a second order blob and is often used in implementations of the random vortex method. Many deterministic vortex method implementations use the Beale Majda 2nd order blob.

Table 2.1: Smoothing functions

Blob type	$f_\delta(r)$	$k(r)$
Chorin	$(2\pi r\delta)^{-1} \quad r < \delta$ $0 \quad r \geq \delta$	$r/\delta \quad r < \delta$ $1 \quad r \geq \delta$
Saffman (Rankine)	$(\pi\delta^2)^{-1} \quad r < \delta$ $0 \quad r \geq \delta$	$(r/\delta)^2 \quad r < \delta$ $1 \quad r \geq \delta$
Krasny	$\delta^2/(\pi(r^2+\delta^2)^2)$	$r^2/(r^2+\delta^2)$
Beale Majda (2nd order)	$e^{-r^2/\delta^2}/(\pi\delta^2)$	$1 - e^{-r^2/\delta^2}$
Beale Majda (4th order)	$(2e^{-r^2/\delta^2} - 0.5e^{-r^2/2\delta^2})/(\pi\delta^2)$	$(1 - 2e^{-r^2/\delta^2} + e^{-r^2/2\delta^2})$

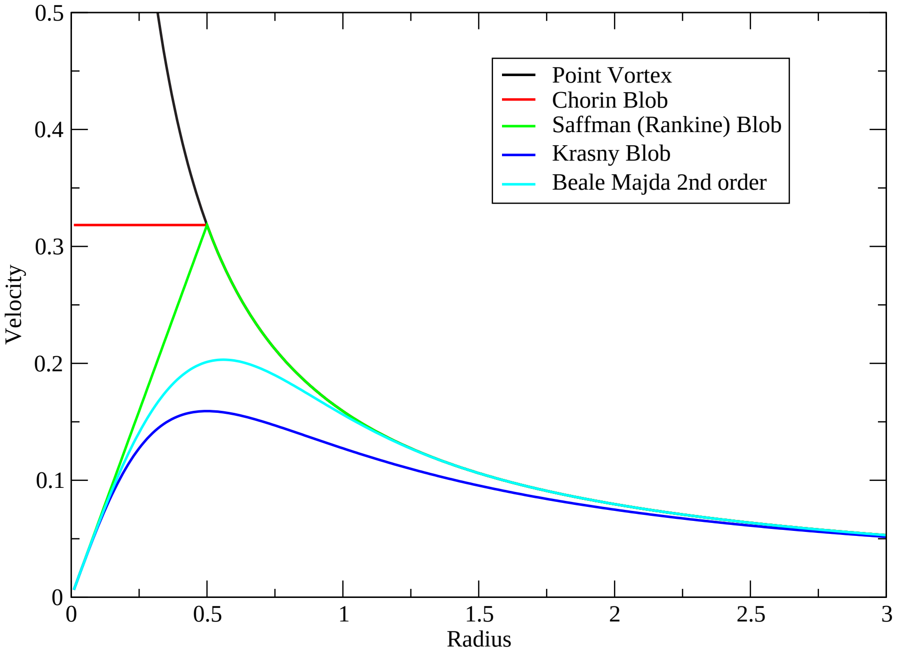

Figure 2.3: Velocity field due to different vortex blobs.

It is to be noted that the accuracy and order of convergence of the vortex blob method are related to the order of the smoothing function. Puckett (1991) and Cottet and Koumoutsakos (2000) provide exhaustive references and material on the topic. Thus, the velocity field, $\vec{V}_\omega$, is obtained using desingularized velocity kernels.

2.2.3 Boundary conditions

If the vorticity field has compact support or is rapidly decaying[2], the velocity field induced by the vorticity field will be zero at infinity. Therefore the infinity boundary condition (2.7e) will be identically satisfied.

The inversion of the vorticity equation (2.9) to obtain the rotational part of the velocity field, $\vec{V}_\omega$, was discussed in sections 2.2.1 and 2.2.2. The resulting velocity field will in general violate the no-penetration boundary condition in equation (2.7f). As discussed in section 2.2.1, a potential velocity, $\vec{V}_p$, is to be added to the rotational velocity field to satisfy the no-penetration boundary condition. In section 2.2.1 it was also seen that this velocity field can be expressed as the gradient of a potential, ϕ, that satisfies the Laplace equation.

A commonly used approach used to satisfy the no-penetration condition is the method of images. For example, consider a point vortex in the complex plane having circulation Γ, with position z, in the presence of an infinite wall along the x-axis. The no-flow boundary condition across this wall is readily ensured by adding an image vortex having strength $-\Gamma$, and position z^*, where z^* is the complex conjugate of z. Similarly, if the point vortex is in the presence of a circular cylinder of radius, R, the no-flow condition is satisfied by adding an image vortex with circulation $-\Gamma$ at position R^2/z^*. More complex geometries can be handled by using conformal transformations. This has probably been the most widely used technique (Cheer, 1989; Sethian and Ghoniem, 1988; Ghoniem and Gagnon, 1987; Ghoniem and Ng, 1987; Baden and Puckett, 1990; Shashidhar, 1998) in the past to satisfy the no-penetration boundary condition in vortex methods. The

---

[2] This is true for all problems considered in the present work.

advantage of the method is that it satisfies the boundary condition exactly. The disadvantages are the following.

- Cannot handle arbitrary geometries.
- Significantly increases the number of particles and is therefore inefficient.

Some researchers use a grid-based fast Poisson solver to apply the no-flow boundary condition as done by Sethian (1984) and others (Smith and Stansby, 1988, 1989). The problem with this approach is that it necessitates the use of a computational grid.

In the present work, tools are developed for vortex methods in the presence of complex geometry. It would be ideal to use a grid free technique to enforce the boundary condition. In the context of a vortex method it is also important to be able to obtain an accurate velocity at an arbitrary point in the fluid. Panel methods (Katz and Plotkin, 1991) are an attractive numerical technique that satisfy these requirements. Panel methods are discussed in detail in section 3.6.

Bakhoum and Board (1996) develop an interesting geometrical method to obtain solutions to Laplace's equation. The method is designed to map the entire potential field inside/outside a boundary. However, for vortex methods one requires the potential or velocity at an arbitrary collection of points. Hence the panel method appears more attractive for vortex methods. The authors in (Bakhoum and Board, 1996) show that their method is much faster than the traditional panel methods. However, the techniques described in chapter 4 make the panel method comparably efficient. Therefore the present work uses a panel method to satisfy the no-penetration boundary condition.

Thus, $\vec{V}_p$ is obtained using the panel method. This along with $\vec{V}_\omega$ gives the total velocity field (equation (2.11)). Using this velocity field, the convective displacement of the particles can be computed.

2.2.4 Convective displacement of the vorticity

The incompressible Euler equations (2.7) transport the vorticity along with the flow. Let the initial vorticity, $\omega_0 = \omega(\vec{x}, 0)$, be discretized into vortex blobs placed at $\vec{x}_j(0)$ and having circulations Γ_j. Then, the Euler equations are solved by computing the path of these vortex blobs with their circulations remaining constant.

The convective displacement of the vortex blobs are computed using the velocity field in equation (2.11). Thus, the system of ODEs describing the position of the vortex blobs is given by,

$$\begin{aligned}\frac{d\vec{x_j}}{dt} &= \vec{V}(\vec{x_j}, t) \\ &= \vec{V}_\omega(\vec{x_j}, t) + \vec{V}_p(\vec{x_j}, t) \\ &= \sum_{k=0;k\neq j}^{N} K_\delta(\vec{x}_j - \vec{x}_k)\Gamma_k + \vec{V}_p(\vec{x_j}, t), \end{aligned} \tag{2.22}$$

where $\vec{V}_p(\vec{x_j}, t)$ is the potential velocity field used to satisfy the no-penetration boundary condition. The $k \neq j$ condition arises due to the fact that a radially symmetric vortex blob does not induce a velocity on itself. These ODEs are integrated as described in section 3.3. Thus, advection can be implemented.

2.3 Diffusion

The second step in the solution of the governing differential equations (2.5) is the solution of the diffusion equation. The governing equations for diffusion are given below.

$$\frac{\partial \omega}{\partial t} = \nu \nabla^2 \omega, \tag{2.23a}$$

$$V(\vec{r}, t) \cdot \hat{e}_s = \vec{V}_B \cdot \hat{e}_s \qquad \text{on } B, \tag{2.23b}$$

Equation (2.23) is the heat equation. Its solution can be obtained by noting that the Green's function for the heat equation in two dimensions is given by,

$$G(\vec{r}, t) = \frac{1}{4\pi\nu t} e^{-\frac{r^2}{4\nu t}}. \tag{2.24}$$

Therefore the solution to equation (2.23) is given as,

$$\omega(\vec{r}, t) = \frac{1}{4\pi\nu t} \int_{\mathcal{R}_2} e^{-\frac{(r-r')^2}{4\nu t}} \omega(\vec{r'}) dx' dy'. \tag{2.25}$$

There are several ways of solving this equation in a Lagrangian framework. The earliest of these is the random vortex method due to Chorin (1973).

2.3.1 Random vortex method

This method simulates diffusion by letting the individual vortex blobs undergo independent random walks with the displacement obtained from a Gaussian distribution having zero mean and variance $2\nu\Delta t$. Since the method is stochastic[3], the solution is noisy and some amount of averaging (ensemble, space or time averaging) is necessary to obtain smooth solutions. The method was numerically shown to have an $O(\sqrt{\nu/N})$ rate of convergence by Roberts (1985), where N is the number of vortex blobs. This is a slow rate of convergence. However, as discussed in section 1.2, the method has been used to solve a variety of problems. The idea of using a random walk to simulate diffusion is based on the fact that the integral in equation (2.25) can be written along with equation (2.19) as,

$$\omega(\vec{r}, t) = \frac{1}{4\pi\nu t} \int_{\mathcal{R}_2} e^{-\frac{r'^2}{4\nu t}} \sum_{j=0}^{N} f_\delta(\vec{r} - (\vec{r_j} + \vec{r'}))\Gamma_j dx' dy'. \tag{2.26}$$

This can interpreted as the expected value of the integral taken over Gaussian random variables $\vec{r'}$ with zero mean and variance $2\nu t$. The random walk method works by giving each vortex blob an independent Gaussian random displacement with zero mean and variance $2\nu\Delta t$, in each co-ordinate direction during each time step. Since the resultant displacement of the particles is the sum of several independent

---

[3] Gardiner (1985) provides a general and detailed discussion on stochastic methods.

normal distributions the resultant distribution is another normal distribution with the net variance being the sum of the individual variances.

It is to be mentioned that the method described above is for the simulation of diffusion using random walks in free space. On the surface of a solid boundary, the no-slip boundary condition is to be satisfied[4]. This is done by releasing new vortices of appropriate strength such that the slip is offset. Further, vortex particles striking a solid wall during the random walk are reflected specularly.

The boundary layer region around the body can also be modeled more accurately by using a vortex sheet algorithm as proposed by Chorin (1978). In order to use this approach, the domain of the fluid, D, is split into two regions. A "sheet region" (or sheet layer or numerical layer) around the body, D_s, where the vorticity is modeled in form of vortex sheets and a "blob region", D_b, that contains only vortex blobs.

2.4 Vortex sheets

In order to satisfy the no-slip boundary condition at each time step, vorticity is created in the sheet layer, D_s. This vorticity is modeled in the form of vortex sheets. The sheet region (or numerical layer) is a computational artifice. The governing equations in the numerical layer are the Prandtl boundary layer equations and in terms of vorticity and velocity are given as,

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial s} + v\frac{\partial \omega}{\partial n} = \nu\frac{\partial^2 \omega}{\partial n^2} \tag{2.27a}$$

$$\omega = -\frac{\partial u}{\partial n} \tag{2.27b}$$

$$\frac{\partial u}{\partial s} + \frac{\partial v}{\partial n} = 0 \tag{2.27c}$$

$$(u, v) = (0, 0) \quad \text{on } B \tag{2.27d}$$

$$\lim_{n\to\infty} u(s, n, t) = U_e(s, t), \tag{2.27e}$$

---

[4] The no-penetration condition is satisfied in the advection step.

where $U_e(s,t)$ is the velocity component along s at the edge of the numerical layer, s, n are the co-ordinates tangential and normal to the boundary respectively and u, v are the velocity components along these directions. The vorticity in this region is approximated by sheets of vorticity which are straight line segments. Then,

$$\omega(s,n) = \sum_j \gamma_j b_l(s - s_j)\delta(n_j - n), \tag{2.28}$$

where γ_j is the strength of the vortex sheet, l is the length of the sheet and (s_j, n_j) are the co-ordinates of the center of the sheet and $\delta(n)$ is a Dirac distribution. b_l is the smoothing function. Chorin (1978) proposed the use of a hat or tent function for b_l. The present work uses both a tent and a Haar function. The Haar function is defined as follows,

$$b_l(s) = \begin{cases} 1, & |s/l| < 1/2, \\ 0, & \text{otherwise.} \end{cases} \tag{2.29}$$

The tent function proposed by Chorin is given by,

$$b_l(s) = \begin{cases} 1 - |s/l|, & |s/l| < 1, \\ 0, & \text{otherwise.} \end{cases} \tag{2.30}$$

Note that for the tent function, the total sheet length is $2l$.

The velocity field produced by the sheets is given as,

$$u(s,n) = U_e(s,t) + \sum_j \gamma_j b_l(s - s_j) H(n_j - n), \tag{2.31}$$

where $U_e(s,t)$ is the tangential component of the velocity at the edge of the numerical layer. Equation (2.31) is obtained by integrating equation (2.27b) where ω is given by equation (2.28). $H(n)$ is the Heaviside function defined as

$$H(n) = \begin{cases} 1 & n > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.32}$$

The vertical velocity component, v, is found by using equations (2.27c) and

(2.27d) to get,

$$v(s,n) = -\int_0^n \frac{\partial u(s,n)}{\partial s} dn. \tag{2.33}$$

Using equation (2.31) and approximating $\frac{\partial u}{\partial s}$ using a central difference, v is obtained as,

$$v(s,n) = -\frac{\partial U_e(s,t)}{\partial s} n - \frac{1}{l} \sum_j \gamma_j \left(b_l(s + l/2 - s_j) - b_l(s - l/2 - s_j) \right) \min(n, n_j). \tag{2.34}$$

More details on the derivation may be had from Chorin (1978), Puckett (1991) and also Shashidhar (1998).

There are complications involved in the computation of v using equation (2.34). Unlike a vortex blob, for which the velocity needs to be found at a single point (its center), the sheet requires the knowledge of the velocity at its center and two additional points at, $(s_j \pm l/2, n_j)$. The approach of using a central difference is inelegant and also appears to be against the general spirit of vortex methods. Additionally, it is well known that v in the boundary layer is usually much smaller than u. Consequently, in the present work v is not computed. If h_{num} is the height of the numerical layer, the v velocity of a sheet at (s,n) is set equal to the v velocity at the point (s, h_{num}). This simplifies the implementation of vortex sheets.

For future reference the following sheet types are defined.

- Sheet1 – this sheet uses the Haar function for b_l as defined in equation (2.29). Equation (2.34) is *not* used to compute the v velocity.
- Sheet2 – Similar to Sheet1 except that the tent function for b_l as defined in equation (2.30) is used.
- Sheet3 – Similar to Sheet1 with v computed using equation (2.34).
- Sheet4 – Similar to Sheet2 with v computed using equation (2.34).

Of these four sheets, only Sheet1 and Sheet2 are considered in the present work. Neglecting v is justified in this case because h_{num} is usually chosen to be very small.

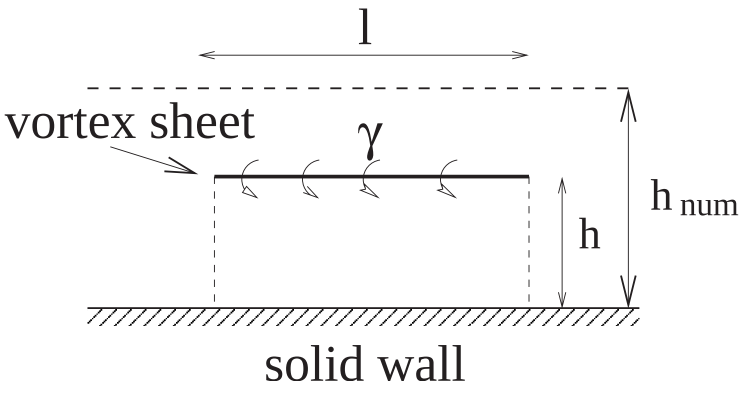

Figure 2.4: Illustration of a vortex sheet.

A typical sheet in the numerical layer is illustrated in Fig. 2.4. The height of the numerical/sheet layer, h_{num} is sometimes assumed to be $k\sqrt{2\nu\Delta t}$, where k is a constant. This is a multiple of the diffusion length scale. Another scheme involves choosing h_{num} as a fraction of an estimated maximum boundary layer height. These possibilities are explored in chapter 7.

The boundary layer equations (2.27) are split into a convection and diffusion step as done with the NS equations (2.5). In order to convect the sheets, the velocity of the sheets are computed using the expressions in equations (2.31) and (2.34). The numerical implementation of the convection is described in section 3.3. The sheets are then diffused using random walks. In this case, since the diffusion is only along the n direction, the random walk is performed only along the n direction.

In this fashion vortex sheets are used to discretize the vorticity in the numerical layer.

Thus the governing differential equation (2.5) is solved using the hybrid RVM. The numerical implementation of the scheme is described in detail in the next chapter.

CHAPTER 3

NUMERICAL IMPLEMENTATION

In the previous chapter the relevant theoretical details of the random vortex method were outlined. This chapter explores the numerical implementation of the random vortex method.

3.1 Operator splitting

As mentioned in section 2.1, the vortex method is typically solved in two fractional steps, advection and diffusion. This splitting of the operator involves an error. Let the solution of the advection equation be obtained using the operator $E(\cdot)$ and the solution to the diffusion equation using the operator $H(\cdot)$. Let the initial condition be ω_0 and the approximate solution at any time t be $\tilde{\omega}(t)$. Standard viscous splitting obtains $\tilde{\omega}(t)$ as,

$$\tilde{\omega}(n\Delta t) = (H(\Delta t)E(\Delta t))^n \omega_0, \tag{3.1}$$

where Δt is the time step used for the integration. Beale and Majda (1981) prove that this type of viscous splitting involves an error of $C\nu\Delta t$. C depends only on the total time interval and the smoothness of the flow. An alternative scheme called Strang-discretization or Strang-type splitting approximates the solution to the NS equations by solving,

$$\tilde{\omega}(n\Delta t) = (H(\Delta t/2)E(\Delta t)H(\Delta t/2)^n \omega_0. \tag{3.2}$$

The error involved in Strang-type viscous splitting is proved (Beale and Majda, 1981) to be $C\nu\Delta t^2$.

In the present work both schemes are explored. Computationally, Strang-type splitting is slightly more expensive but allows for larger Δt values. The two approaches are studied numerically in chapter 7.

3.2 Discretization of vorticity

The vorticity field is to be discretized into vortex blobs and vortex sheets. In applications where a known vorticity field exists the vorticity field is represented in the form of vortex blobs. Equation (2.19) is reproduced below,

$$\omega(\vec{x}) = \sum_{j=0}^{N} f_\delta(\vec{x} - \vec{x}_j)\omega_j h^2.$$

The value of ω_j can be found by either setting $\omega_j = \omega(\vec{x_j})^1$ or by solving the system of equations for the unknown ω_j values based on the known $\omega(\vec{x})$ and f_δ as done by Ghoniem *et al.* (1988) and others (Krishnan and Ghoniem, 1992).

In the present work, the problem studied is the flow past an impulsively started cylinder where there is no initial vorticity field. Vorticity is released in the form of vortex sheets from the surface of the body. The numerical method used to release the sheets is described in section 3.4.

3.3 Advection

As discussed in section 2.2.4, the solution of the advection equation involves solving a set of ordinary differential equations (ODEs). The solution of these ODEs requires the knowledge of the velocity of the particles. The velocity due to the vortex blobs is obtained from equation (2.18). The velocity due to the sheets is obtained from equations (2.31) and (2.34). The computation of the velocity field due to the blobs can be accelerated using a fast summation technique. The AFMM (Carrier *et al.*, 1988) is implemented and used for this purpose. This technique is described in detail in appendix A. The velocity field of the sheets can also be accelerated using the technique described in section 5.2. The velocity field

---

[1] This scheme and an alternative are discussed by Perlman (1985).

is also required to satisfy the no-penetration boundary condition. This boundary condition is satisfied using a panel method. The method is described in section 3.6.

Once the velocity field due to the blobs, sheets and the panel method are known, it is possible to integrate the ODEs using an appropriate numerical scheme. Thus, the new positions of the particles are computed. In the present work these ODEs are integrated using a second order Runge-Kutta scheme as follows,

$$\begin{aligned} \vec{x'}_j(t+\Delta t/2) &= \vec{x}_j(t) + \vec{V}(\vec{x}_j, t)\Delta t/2 && (3.3a) \\ \vec{x}_j(t+\Delta t) &= \vec{x'}_j(t) + \vec{V}(\vec{x'}_j, t+\Delta t/2)\Delta t. && (3.3b) \end{aligned}$$

$\vec{x}_j$ are the positions of the particles and $\vec{V}(\vec{x}_j, t)$ represents the velocity of the particles. Higher order schemes can also be used. The Runge-Kutta fourth order scheme is also constructed in a similar fashion to the above. However, for most of the simulations in the present work the second order Runge-Kutta scheme is used.

The implementation of the second order Runge-Kutta scheme for the vortex sheets is tricky since sheets that leave the numerical layer are to be converted to blobs and vice versa. Since this is common to both diffusion and advection, a discussion is provided in section 3.4.3.

3.4 Diffusion

As discussed in section 2.3.1, the random vortex method simulates diffusion by making each vortex particle perform a random walk. For vortex blobs the displacements are along the co-ordinate directions. For the vortex sheets the displacement is along the local normal.

Computationally, a pseudo-random number generator is used to generate the random numbers. In the present work the random number generator of L'Ecuyer with a Bays-Durham shuffle (Press *et al.*, 1992) is used to generate uniform deviates. This generator has a period greater than 2×10^{18} which suffices for all the calculations performed in the present work. Gaussian deviates are generated from the uniform ones using the Box-Muller transformation (Press *et al.*, 1992).

3.4.1 No-slip boundary condition

The no-slip boundary condition is satisfied by releasing vortex sheets every time the diffusion equation is solved. Numerically, the boundary is split into N equally sized linear segments. The centers of each of these segments (control points) are used to satisfy the no-slip boundary condition. The velocity due to all the constituents (blobs, sheets, free stream etc.) are computed at these points. Vortex sheets are introduced at these locations to satisfy the no-slip boundary condition. If the standard viscous splitting approach, equation (3.1), is used, the newly introduced sheets are only diffused in the first time interval. From the next time step onwards, they are both convected and diffused.

Let γ_{max} be the maximum magnitude of the strength of any vortex sheet. Let u_s be the slip velocity at a particular control point. Let n be the number of sheets released at the control point. In this work, the new sheets are created in three different ways:

1. $n = [|u_s|/\gamma_{max}]$, with each sheet having magnitude of strength γ_{max}. Where $[x]$ is the largest integer less than x. This is called sheet release style 1.
2. $n = [|u_s|/\gamma_{max} + 0.5]$, with each sheet having a magnitude of γ_{max}. This is called sheet release style 2.
3. $n = [|u_s|/\gamma_{max}]$, with each sheet having magnitude γ_{max} plus one additional sheet having magnitude of strength, $u_s - n\gamma_{max}$. This additional sheet satisfies the no-slip condition exactly. This is called sheet release style 3.

The sign of the strength of the sheets are chosen so as to nullify the slip velocity.

In chapter 7 these three sheet release styles are studied numerically.

3.4.2 Sheet tagging

Equation (2.27a) indicates that the diffusion is independent of the s direction. This implies that a vortex sheet introduced along the body surface diffuses as a single unit. In order to perform this correctly, the following tagging procedure is performed. Consider the computation at a particular time, t. At a typical control point let q be the number of new sheets being released. Each of these new sheets is tagged with a sequence of integers in the range $m+1, \ldots, m+q$ where m is the largest tag value of the old sheets. This process is repeated for all the newly created sheets with m being the same for all the control points. After the tagging is complete, when displacing sheets with the random displacement along n, all sheets with the same tag are given the same random displacement. In this manner sheets at different spatial locations are made to diffuse as one single unit.

Puckett (1989) numerically shows that sheet tagging does not improve the accuracy of the vortex sheet method and recommends that the method need not be used. In section 7.7 this is explored and the result obtained is similar in that no special improvement is seen when tagging is performed.

3.4.3 Conversion of sheets to blobs

As a sheet moves it will eventually leave the numerical layer. Similarly, a moving blob is likely to enter the numerical layer. Any sheet having length λ, that leaves the numerical layer is converted to a Chorin blob (see Table 2.1) of strength, $\Gamma = \gamma\lambda$, with a core radius δ such that there is no jump in the velocity at the control point on the boundary due to the conversion. That is,

$$\frac{\Gamma}{2\pi\delta} = \frac{\gamma}{2}$$

since $\Gamma = \gamma\lambda$,

$$\delta = l/\pi$$

Since the Chorin blob is used, the velocity inside the core region is constant in magnitude and this conversion may be performed even if the core intersects the boundary as would happen if $h_{num} < \delta$. Similarly if a blob enters the numerical layer it is converted into a sheet with a corresponding strength and length. It is given a new tag value.

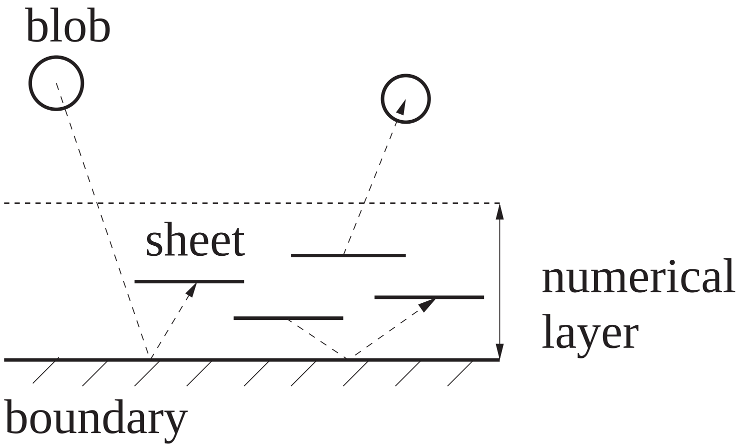

Figure 3.1: Reflection and conversion of vortex sheets and blobs.

As the sheets and blobs are given random displacements they are likely to strike the solid wall. If this happens, they are all reflected specularly, i.e. if the solid wall is along the x-axis and the particle's position after being randomly displaced is at (x, y), then its reflected position is $(x, -y)$. Reflecting particles in this fashion correctly and efficiently for complex geometries is not trivial. An efficient algorithm for this is discussed in section 5.1. The process of conversion and reflection is illustrated in Fig. 3.1.

As mentioned earlier, the sheets and blobs are convected using a second order Runge-Kutta scheme. As seen in equation (3.3a), this involves computing an intermediate set of positions for the blobs and sheets. It is possible for sheet-blob conversion to occur at this point. If this happens, these converted sheets and blobs are converted back to their original state and then displaced in the final step, equation (3.3b). Due to the complexity of the process, this is generally not done in traditional RVM implementations. In the present work no attempt has been made to test if this indeed improves accuracy. However, multi-step methods are used when convecting the vortex sheets and this procedure is used to handle the sheet-blob conversions.

3.4.4 Other techniques to satisfy no-slip

For completeness, it is to be noted that other researchers use different approaches to handle the no-slip condition. Chorin (1973), Lin *et al.* (1997), Clarke and

Tutty (1994), Shankar (1996) and various others (Smith and Stansby, 1988, 1989) generate a row of vortex blobs on the surface that satisfy the no-slip boundary condition. Koumoutsakos *et al.* (1994) and others (Koumoutsakos and Leonard, 1995; Shiels, 1998; Cottet *et al.*, 2000; Ploumhans and Winckelmans, 2000) on the other hand, generate a vortex sheet at the surface and then diffuse this vorticity to nearby vortex blobs by simulating the diffusion of this sheet in a semi-infinite domain. This approach yields accurate results in comparison to schemes that release vorticity in the form of vortex blobs above the boundary. Bernard (1995) introduces a deterministic sheet method and also experiments with a different type of sheet whose velocity field is computed by integrating the Biot-Savart law on the sheet element. This is similar to the rectangular anisotropic vortex sheet elements of Huyer and Grant (1996). The advantage with this type of sheet element is that it behaves like any other vortex element and induces a global velocity field. Teng (1982) uses an elliptically shaped blob core. Such blobs can be used in the solution of the NS equations and therefore can be used inside and outside the numerical layer. However, it appears that other researchers have not used this type of blob in simulations of the NS equations. Marshall and Grant (1996) also develop a method to find the velocity field due to anisotropic vortex blobs. They find that the computation of the velocity field is very expensive. However, as compared to the isotropic blobs, much fewer anisotropic elements are needed to obtain good agreement with the Blasius boundary layer solution.

Summers (2000) provides an interesting discussion on boundary conditions for viscous vortex methods using the idea of Hodge decomposition and impulses generated at the wall. Impulses differ from the rotational velocity ($\vec{V}_\omega$) by the gradient of a scalar function. The boundary condition at the wall is satisfied by the generation of impulses at the surface. Two different schemes are considered and one of these correspond to the generation of vortex sheets at the surface. These vortex sheets are like the anisotropic elements discussed above. This gives vortex sheets an interesting impulse based interpretation. Summers (2000) also discusses another approach where a vortex-dipole distribution at the surface is generated to satisfy the boundary condition. Cortez (2000) uses impulse elements to simulate the motion of thin flexible boundaries. The impulse elements are created because of forces generated at the boundaries. This serves as a means to connect kinematics to dynamics. Hence, impulse elements offer an interesting approach to satisfy the boundary conditions for vortex methods.

In the present work the traditional vortex sheet algorithm introduced by Chorin (1978) and described in earlier sections is used to satisfy the no-slip boundary condition.

In the next section, a simple technique used to reduce the number of interacting particles is described.

3.5 Merging and annihilation of vorticity

The RVM simulates diffusion by giving vortex particles a random displacement. During a numerical simulation it is often found that large numbers of particles having a vorticity of equal magnitude and opposite sign are very close to each other. These particles effectively contribute nothing to the simulation since they cancel each other's effects. These particles are called "parasitic particles" (Choi *et al.*, 1988). In chapter 7 it is numerically demonstrated that these parasitic particles make the solution inaccurate. In the present work, parasitic particles are removed by annihilating any two particles of opposite sign having the same magnitude of circulation provided they lie within a distance of $R_a\ \lambda$ of each other. λ is the length of a vortex sheet and R_a is a non-dimensional number. If the two particles are of opposite sign and are not of equal strength, the particle with a smaller strength is deleted with the larger particle absorbing its strength. This annihilation incurs a small error in the moments of the vorticity. However, as will be demonstrated in later chapters, the benefits of annihilation more than compensate for this loss of accuracy.

If there are a large number of particles near each other having the same sign but with small circulations, then they can be merged to reduce the number of particles. While merging the particles, it is possible to conserve the first moment of the vorticity by moving the position of the merged particle appropriately. Consider two similarly signed particles having strengths, Γ_1, Γ_2 and positions $\vec{x}_1, \vec{x}_2$, that are within a prescribed distance, $R_m\ \lambda$ with $|\Gamma_1+\Gamma_2| < \gamma_{max}\lambda$. These two particles are merged and the merged blob's strength is $\Gamma_1+\Gamma_2$ with position $(\Gamma_1\vec{x}_1+\Gamma_2\vec{x}_2)/(\Gamma_1+\Gamma_2)$. A more sophisticated scheme specifically tailored to Gaussian vortices is presented by Rossi (1997) where the second moment is also conserved by varying the core-radius of the merged blob. However, in the present work the simpler scheme is used.

The next section describes the panel method that is used to satisfy the no-penetration boundary condition.

3.6 The panel method

Panel methods provide a means to solve Laplace's equation in two and three dimensions[2]. They can be used effectively to satisfy the no-penetration condition discussed in section 2.2.3. Panel methods reduce the dimensionality of the problem by one and in two dimensions only require a one dimensional surface grid. The method works by discretizing the body into elements called panels. An unknown amount of singularity is distributed on each of these panels. The unknown strengths of these singularities are solved for using an appropriate boundary condition. Thus, the singularity distribution is obtained as the solution of a boundary value problem. The geometry of the panels can be linear, parabolic or higher order. For the singularity distribution, sources, doublets or vorticity can be used. The singularity can be lumped at a point but this results in very low accuracy. Usually a constant, linear or a higher order function is used. The boundary conditions used can be represented either in terms of the normal velocity on the surface, called the Neumann condition, or in terms of the potential/stream function on the body, called the Dirichlet condition. Katz and Plotkin (1991) provide a comprehensive overview of panel methods.

---

[2] The panel method has also been used to study the evolution of vortex sheets (Peters and Hoeijmakers, 1995).

Yon (1990) performs an extensive study of nine different panel methods and shows that for difficult geometries such as airfoils with cusped trailing edges or very thin airfoils, only the Neumann formulation using a linear distribution of vorticity produced satisfactory results. A distribution of doublets could have also be used to simulate lifting bodies. However, it can be shown that a polynomial distribution of doublets of order q can be represented by a distribution of vorticity of order $q-1$. Another advantage of using a distribution of vorticity is that it can be used to explain the kinematic motion of the rigid body in addition to solving the fluid flow (Rajan, 1994). Based on the above reasons and the fact that using a vorticity distribution is natural for a vortex method, the present work uses a vorticity distribution.

The boundary condition can be applied using either a no-penetration condition or a no-tangency condition just inside the body. This condition is to be applied at a specific location on each panel called the "control point". For the no-penetration condition one enforces that the normal component of the velocity at the control point should be the same as that of the body. For the no-tangency condition the tangential component of the velocity is equated to that of the tangential velocity of the body. It can be shown (refer Shiels (1998) for a sample proof) that the no-tangency condition automatically satisfies a no-penetration condition. The no tangency condition also results in better conditioned matrices. The problem with the no-tangency condition is that thin shapes can cause problems requiring corrections (Lewis, 1991). It is also not possible to incorporate the no-tangency condition for bodies with zero thickness (like a zero thickness flat plate). For this reason the present work uses the more traditional no-penetration condition.

In the recent past, several researchers (Clarke and Tutty, 1994; Lin *et al.*, 1997; Kim and Flynn, 1995; Takeda *et al.*, 1999; Taylor and Vezza, 1999*a*,*b*) have employed the panel method in the context of vortex methods. Clarke and Tutty (1994) and Takeda *et al.* (1999) use a parabolic panel method where the geometry of the body is discretized into parabolic segments. The others use a linear approximation for the geometry. The present work develops a very accurate cubic panel method.

In the following, the velocity and potential due to a linear vortex panel is derived. The issues with this approach are considered and then a cubic panel method is developed. These derivations are largely reproduced from the work of Ramachandran *et al.* (2000*a*, 2003).

3.6.1 Linear panels and the edge effect

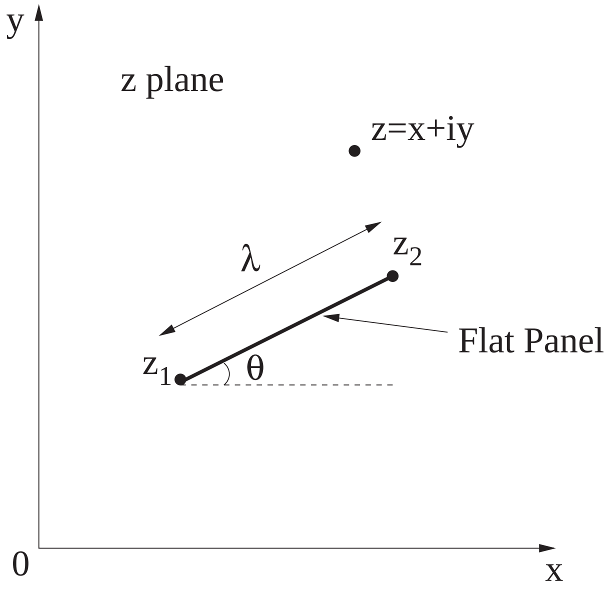

Figure 3.2: A flat panel in the complex plane.

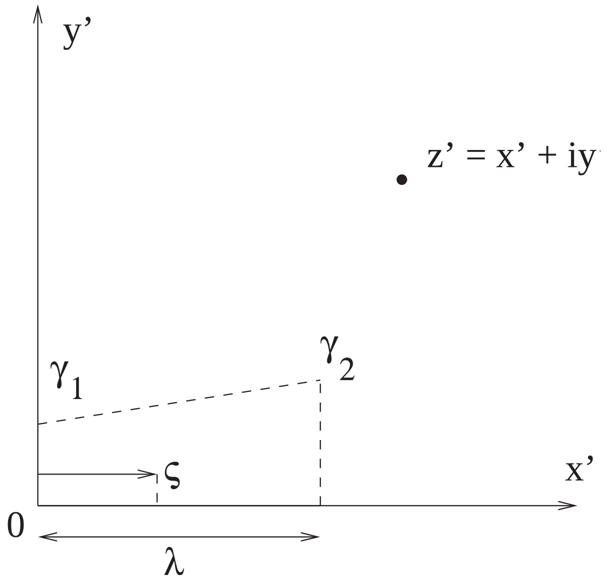

Figure 3.3: A flat panel in the $z' = (z - z_1)e^{-i\theta}$ plane.

As mentioned earlier, the present work uses a vortex panel method. The body is split into a collection of linear segments called panels. As done by the authors in (Ramachandran *et al.*, 2000*a*), a linear distribution of vorticity is chosen on each panel. A panel with a linear (flat) geometry, having length λ, is considered. This is illustrated in Fig. 3.2. The local co-ordinate system with respect to the panel shown in Fig. 3.3 as the z' plane. γ is the vorticity at any point on the panel. γ_1 and γ_2 are the values of γ at the ends of the panel. The vorticity γ at any point on the panel (ζ) is given by,

$$\gamma = \gamma_1 + \frac{\gamma_2 - \gamma_1}{\lambda}\zeta. \tag{3.4}$$

The velocity due to the panel at a point z' is obtained as follows,

$$V(z') = u' - iv' = \frac{-i}{2\pi}\int_0^{\lambda} \frac{\gamma \, d\zeta}{z' - \zeta} \tag{3.5}$$

where $i = \sqrt{-1}$. Integrating the above equation after substituting equation (3.4) and transforming the velocity back to the z plane gives the velocity due to the panel at an arbitrary point z as,

$$\begin{aligned} V(z) &= u - iv \\ &= \frac{-i}{2\pi}\left\{\gamma_1\left[\left(\frac{z'}{\lambda} - 1\right)\ln\left(\frac{z' - \lambda}{z'}\right) + 1\right]\right. \\ &\qquad \left. -\gamma_2\left[\frac{z'}{\lambda}\ln\left(\frac{z' - \lambda}{z'}\right) + 1\right]\right\} e^{-i\theta}. \end{aligned} \tag{3.6}$$

where $z' = (z - z_1)e^{-i\theta}$. In a similar fashion the complex potential due to the linear vortex panel can be found as,

$$\begin{aligned} \Phi(z) &= \phi + i\psi \\ &= \frac{-i}{2\pi}\int_0^{\lambda} \gamma \ln(z' - \zeta)\, d\zeta \\ &= \frac{i\gamma_1}{2\pi}\left[(z' - \lambda)\ln(z' - \lambda) - z'\ln(z') + \lambda\right] \\ &\quad - \frac{i(\gamma_2 - \gamma_1)}{4\pi\lambda}\left[(\lambda^2 - z'^2)\ln(z' - \lambda) - \lambda z' - \frac{\lambda^2}{2} + z'^2\ln(z')\right]. \end{aligned} \tag{3.7}$$

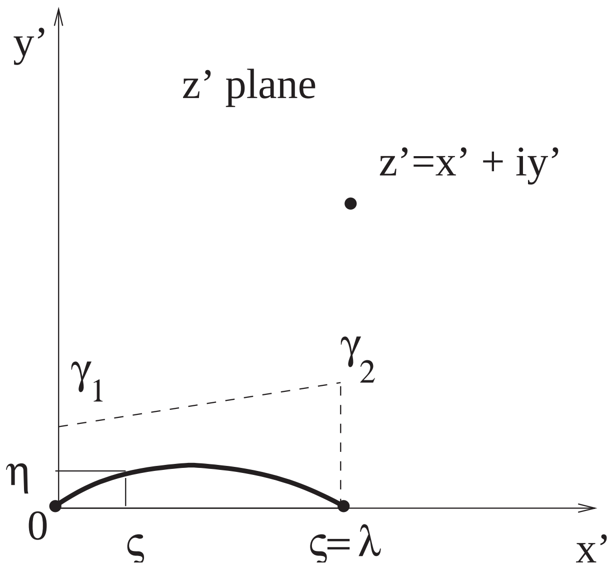

Figure 3.4: A cubic panel having chord length λ in the z' plane.

It can be seen that equation (3.6) for the velocity due to the flat panel diverges at the end points, z_1 and z_2. For a curved body, the singularity is not nullified by the adjacent panels due to the linear discretization of the body geometry and the consequent discontinuity in the slopes between two adjacent panels. This is known as the "edge effect". This edge effect would exist even if a higher order singularity distribution were used on the surface of the panel.

3.6.2 Cubic panels

Ramachandran *et al.* (2000*a*) show that the edge effect can be removed if the body geometry is discretized in terms of cubic panels. In this approach, the slope is continuous across adjacent panels (if the body shape is smooth). Let the chord of the cubic panel be oriented along the x'-axis. As shown in figure 3.4, let ζ and η respectively be the x' and y' coordinates of the panel surface. Being a cubic, the equation of the panel is given by $\eta = a_1\zeta + a_2\zeta^2 + a_3\zeta^3$. a_1, a_2 and a_3 depend on the slopes of the panel at the end points and the chord length, λ. To keep the integral tractable, the vorticity is distributed on the surface of the panel but is assumed to be linear in ζ. The equation for the velocity due to such a panel at a point z' is given by,

$$\begin{aligned} V(z') &= \frac{-i}{2\pi}\int_0^\lambda \frac{(\gamma_1 + k\zeta)}{(z' - (\zeta + i\eta))} d\zeta \\ &= \frac{-i}{2\pi}\int_0^\lambda \frac{(\gamma_1 + k\zeta)}{(z' - \zeta - i(a_1\zeta + a_2\zeta^2 + a_3\zeta^3))} d\zeta \\ &= \frac{k}{2\pi a_3}\int_0^\lambda \frac{(\zeta + \gamma_1/k)}{\left(\zeta^3 + \frac{a_2}{a_3}\zeta^2 + \frac{(a_1-i)}{a_3}\zeta + \frac{iz'}{a_3}\right)} d\zeta \end{aligned} \tag{3.8}$$

where $k = (\gamma_2 - \gamma_1)/\lambda$. In order to obtain the solution, the cubic in the denominator is expressed as,

$$\zeta^3 + \frac{a_2}{a_3}\zeta^2 + \frac{(a_1 - i)}{a_3}\zeta + \frac{iz'}{a_3} = (\zeta - a)(\zeta - b)(\zeta - c) \tag{3.9}$$

where a, b and c are the complex cube roots of the cubic. These roots are computed using the algorithm given in (Press *et al.*, 1992). After obtaining the roots, equation (3.8) is integrated using the method of partial fractions. After integration and simplification, the velocity due to the cubic panel is obtained as

$$\begin{aligned} V(z) =& \frac{-\gamma_2}{2\pi a_3\lambda}\left[\frac{a\log\left(\frac{a-\lambda}{a}\right)}{(a-c)(a-b)} + \frac{b\log\left(\frac{b-\lambda}{b}\right)}{(b-c)(b-a)} + \frac{c\log\left(\frac{c-\lambda}{c}\right)}{(c-a)(c-b)}\right] e^{-i\theta} \\ &- \frac{\gamma_1}{2\pi a_3\lambda}\left[\frac{(\lambda - a)\log\left(\frac{a-\lambda}{a}\right)}{(a-c)(a-b)} + \frac{(\lambda - b)\log\left(\frac{b-\lambda}{b}\right)}{(b-c)(b-a)} + \frac{(\lambda - c)\log\left(\frac{c-\lambda}{c}\right)}{(c-a)(c-b)}\right] e^{-i\theta}. \end{aligned} \tag{3.10}$$

θ is the angle between the chord of the panel and the x-axis.

The complex potential due to a cubic panel is given by,

$$\Phi = \phi + i\psi = \frac{-ik}{2\pi}\int_0^\lambda (\zeta + \frac{\gamma_1}{k})\ \ln\left(ia_3(\zeta - a)(\zeta - b)(\zeta - c)\right) d\zeta$$

which upon simplification and integration becomes

$$\Phi = \frac{-i\lambda}{4\pi}(\gamma_1 + \gamma_2)\ln(-ia_3) + I_1 + I_2 + I_3 \tag{3.11}$$

where

$$
\begin{aligned}
I_1 = \frac{-ik}{8\pi}\left[(\lambda - a)^2(2\ln(\lambda - a) - 1) - a^2(2\ln(-a) - 1)\right] \\
- \frac{ik}{2\pi}\left[(a + d)((\lambda - a)(\ln(\lambda - a) - 1) + a(\ln(-a) - 1))\right],
\end{aligned}
$$

and $d = \gamma_1/k$. I_2 and I_3 are obtained from I_1 by replacing a respectively with b and c.

Care must be taken in the implementation of the cubic panel method. If the panel is parabolic (or close to parabolic), $a_3 \to 0$, and there can be severe numerical errors in the evaluation of the velocity and complex potential. Similarly, if the panel is linear, care must be taken to avoid numerical problems with a small value of a_2. In the present work, if the coefficients in the cubic equation are very small then an appropriate parabolic or linear panel velocity is used.

3.6.3 Solving the boundary value problem

From the known expression of the velocity field due to a single panel it is easy to create a system of linear equations for the unknown values of γ on the panel surface. The solution of this linear system of equations gives the solution of the boundary value problem. Let N_p be the number of panels. Let the velocity field due to a panel, i, at the j'th panel control point be $\vec{V}_{ij}$. The following set of equations are obtained by setting up N_p no-penetration boundary conditions, one for each panel,

$$
\sum_{i=0}^{N_p} \vec{V}_{ij} \cdot \hat{e}_{nj} = (\vec{V}_{Bj} - \vec{V}_j) \cdot \hat{e}_{nj}, \tag{3.12}
$$

where $\hat{e}_{nj}$ is the unit normal vector at the control point of the j'th panel. $\vec{V}_{Bj}$ is the velocity of the solid surface at the j'th panel control point and $\vec{V}_j$ is the velocity due to the other constituents of the flow (vortex blobs, vortex sheets, free stream etc.). Hence, the quantities in the right-hand side of equation (3.12) are known. $\vec{V}_{ij}$'s are governed by either equation (3.6) or (3.10) depending on the panel geometry. These involve the unknown vorticity of the panel, γ. For a linear distribution of vorticity there are two unknown strengths per panel, γ_1 and γ_2.

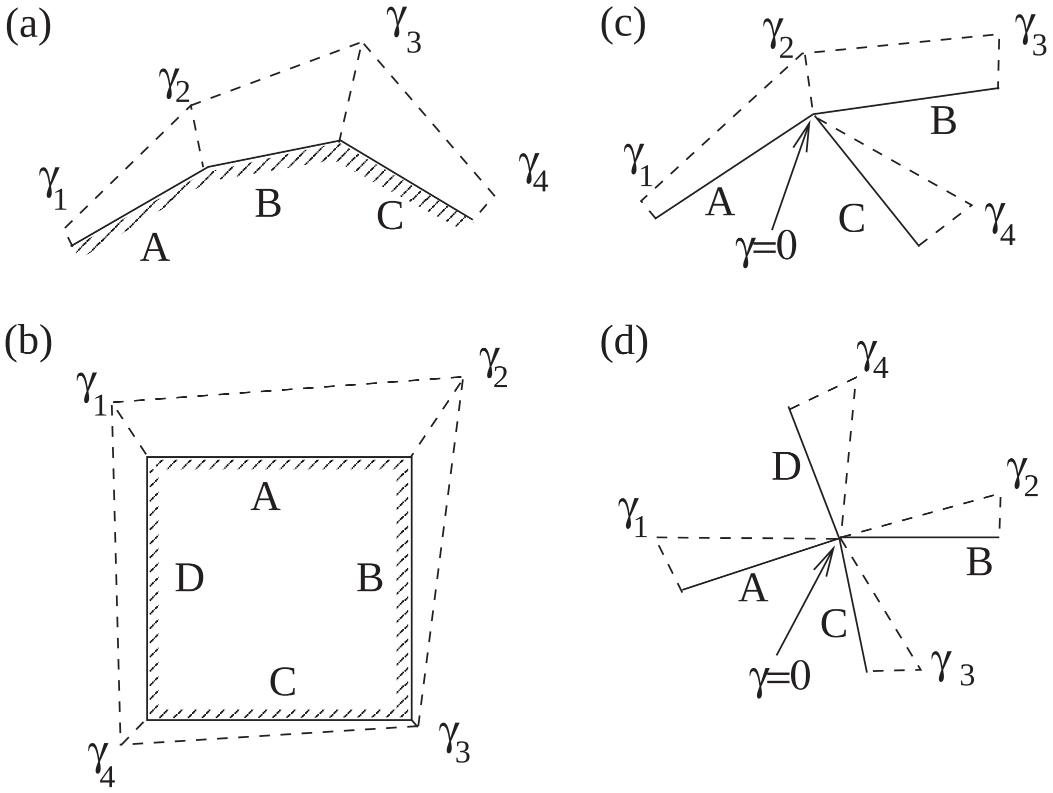

Figure 3.5: Vorticity distribution for different types of panel attachments.

Depending on the geometry of the body, there are simplifying assumptions one can make regarding the vorticity at the edge of two adjacent panels. Some of the important possibilities are illustrated in Fig. 3.5. Consider the cases (a) and (b) in the figure. Clearly, if only two panels are attached at a particular point, there cannot be a jump in the vorticity. For this case, adjacent panels share the same value of γ at their attachment point. If there are more than two panels attached at a point there are complications. For the case (c) shown in Fig. 3.5, it is evident that the point where panel C is attached to panels A and B is a stagnation point due to the acute angle between the panels. This requires that the value of γ of panel C at the point be zero. On the other hand the region above panels A and B is not a stagnation region since the angle between the panels is obtuse. Additionally, the value of vorticity there must also be continuous. Hence, the panels A and B share a common value γ_2. For the case (d) the point where all the panels A, B, C and D meet is a stagnation point. Hence, the value of γ for all the panels at that point is zero. Such cases introduce difficulties in the solution of the system of equations (3.12). The different possibilities discussed above are listed below.

- If the body is a simple closed surface, there are N_p boundary conditions and N_p unknowns as illustrated by case (b) of Figure 3.5.
- If the body is an open arc, like a zero thickness flat plate, there are $N_p + 1$ unknowns and only N_p boundary conditions. This is illustrated in case (a) of Figure 3.5.
- The different types of attachments of one body to another are illustrated in cases (c) and (d) of Figure 3.5. The intensity of vorticity of the panel at the stagnation point is zero. Hence, panels attached at a stagnation point should have a zero vorticity intensity at the point of attachment. This is illustrated in case (d) of Figure 3.5. Attached panels that subtends an obtuse angle should have a continuous vorticity strength at the common point as illustrated for panels A and B in case (c) of Figure 3.5.

In addition to the above complications, one must ensure that the circulation around each body is equal to the amount due to the the rotation of the body. For a non-rotating body it is zero. This is one additional equation per body. Hence, only for the case of a simple open body are there an equal number of equations and unknowns. Every other case has more equations than unknowns. The matrix embodying the linear system of equations is also dense. In the present work a singular value decomposition (Press *et al.*, 1992) is used to solve the resulting matrix with more conditions than variables. The method is computationally expensive but provides the best solution in the least squares sense. Also, for rigid body motion with no relative motion between parts of a body, the generalized inverse need be computed only once. Therefore, this approach provides the best results and handles all the above complications without incurring a large computational cost. If there is a need to handle moving bodies and a large number of panels, it is practical to use an iterative matrix solver. However, this is not done in the present work. In this manner, the no-penetration condition is accurately imposed by computing a potential velocity field.

3.6.4 Numerical results

The cubic panel method eliminates the edge effect and is also more accurate than the linear panel technique. In order to demonstrate this, the error in the computed

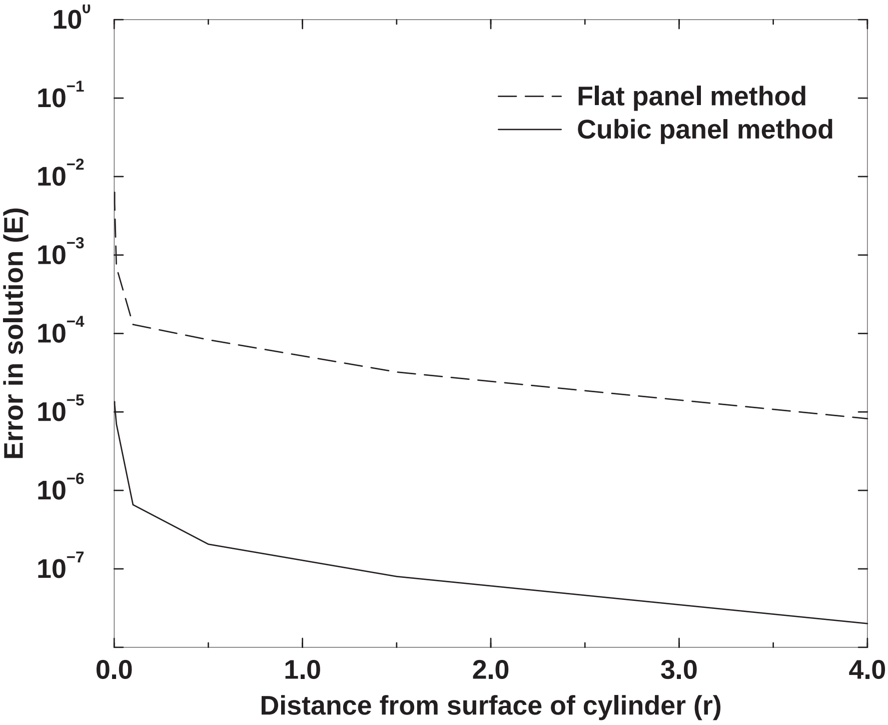

Figure 3.6: Comparison of the error E versus distance from the surface of a circular cylinder obtained by using flat and cubic panels.

solution is defined as,

$$E = \left(\frac{\sum_{j=1}^{N} |v_j - \tilde{v_j}|^2}{\sum_{j=1}^{N} |v_j|^2} \right)^{1/2}, \tag{3.13}$$

where v_j is the exact velocity, $\tilde{v_j}$ is the computed velocity and N is the number of points at which the error is computed. Consider the flow past a circular cylinder of unit radius as the test problem. The exact solution is compared with that obtained by using 200 flat and cubic panels. The no-penetration boundary condition is applied at the center of each panel. A ring of about 1000 points at a distance r from the surface of the cylinder is considered and the value of E computed. Figure 3.6, plots the error E versus r for both the flat and cubic panel methods. As is evident, the cubic panel is at least two orders of magnitude more accurate. Even extremely close to the surface, the error is relatively small. Unlike the cubic panel case, when the flat panels are used, the error grows without bound as one approaches the edge of the panels.

The difficulty with the cubic panel method is that for the evaluation of the velocity due to each panel, the three roots of a complex cubic equation are to be computed. This makes the method computationally expensive. Based on the present implementation, the cubic panel method is found to be about 4 times slower than the flat panel method. Techniques to accelerate the computation of the velocity field due to the panels are discussed in chapter 4.

Thus, the developed cubic panel method can be used to accurately satisfy the no-penetration condition. The next section describes how the various components described in this chapter are put together to form the random vortex method.

3.7 Computational procedure

Using the techniques described in this chapter it is possible to implement the random vortex algorithm. For the standard viscous splitting scheme, equation (3.1), the random vortex method proceeds as follows.

1. The slip velocity on the bounding solid surfaces is computed.
    - The slip velocity is computed at a set of control points on the boundary.
    - The velocity field due to the free stream, vortex blobs and vortex sheets are computed.
    - The no-penetration condition is enforced using a panel method as discussed in section 3.6.
2. The existing sheets and blobs are convected using an appropriate ODE integration scheme (usually a second order Runge-Kutta scheme, equation (3.3), is used).
    - The velocity field is computed as discussed in sections 2.2.
    - The no-penetration boundary condition is satisfied using the panel method discussed in section 3.6.
    - The calculation of the velocity field is computationally expensive. Techniques to accelerate this computation are discussed in the next chapter.
    - As mentioned earlier in section 3.4.3, the conversion of blobs to sheets and vice versa is handled during the intermediate steps of the integration.
3. Vortex sheets are added just above the surface of the body to offset the slip velocity computed in step 1. This is discussed in section 3.4.1.
4. The vortex sheets and blobs are diffused as discussed in sections 2.3.1 and 3.4. Blobs and sheets are inter-converted as necessary.

5. The vortex particles are merged and annihilated as described in section 3.5. An efficient algorithm for this is described in section 5.4.
6. The process repeats from step 1.

The specific details of the algorithm depend on the implementation. However, the general manner in which a vortex method simulation proceeds is explained above. As seen, the algorithm is very elegant in the sense that it mimics the actual physical process.

3.8 Ensemble averaging

The RVM is a stochastic method. For a particular simulation, obtaining results from a single trial using the RVM may not produce conclusive results. Making several trials with different seqences of random numbers[3] and ensemble averaging the trials produces much more reliable results.

Ensemble averaging brings out an important feature of the RVM viz. the ability to trivially parallelize the trials. Given a serial random vortex code it is possible to use the same program on another computer with a different random seed and then ensemble the results. This provides improved accuracy in the same time. The convergence rate of the random vortex method is slow. However, by using this approach the accuracy can be improved considerably with minimum effort in the same time. In section 7.7, ensemble averaged results are used to make several recommendations on the optimal choice of various computational parameters.

3.9 The ensembled RVM

In this work a new and simple variance reduction scheme is introduced. The method is based on the idea that ensemble averaging improves the quality of the solutions. The method also takes advantage of the inherent parallelism in the

---

[3] The random number generator produces different sequences of random numbers when it is seeded with a different initial value.

RVM. The new method is called the Ensembled RVM (ERVM) and works as follows.

Assume there are n_{proc} processors simulating the same problem with random number generators initialized using different seeds. At the end of n_{sync} time steps, the data (blobs and sheets) from the processors are assembled together. The circulation of each particle is divided by n_{proc}. The resulting particles are then merged and annihilated as they would be during the course of the simulation. Each of the processors then resumes the computations using the new distribution of particles. Particles having a circulation of magnitude less than one thousandth of the largest circulation ($\gamma_{max}\lambda$) are removed from the computation. Thus each processor resumes its simulation using a more accurate intermediate solution. This approach considerably reduces the standard deviation and errors in the solution.

In section 7.8, simulations are made using the ERVM. An explanation of how the relevant parameters are to be chosen for optimal results is also presented. The results show that the method works very well. In chapter 8 and specifically in section 8.2, it is shown that the ERVM performs quite favorably as compared with the deterministic diffusion schemes in terms of the quality of the results and the computational effort.

In the next chapter, fast summation techniques that are central to modern vortex methods are discussed.

CHAPTER 4

FAST SUMMATION TECHNIQUES

In the previous chapter an overview of the numerical implementation of the RVM was provided. In this chapter details on the fast algorithms used to rapidly evaluate the velocity field due to different entities in the vortex method are provided. Fast summation techniques are first introduced. Background material on the theory and implementation of the AFMM are provided in appendix A. The extensions of the AFMM developed in the present work are then discussed.

4.1 Introduction

The advection of vorticity requires the knowledge of the velocity field. The velocity field due to a collection of vortex blobs is given by equation (2.18). If there are N blobs, finding the velocity at a point requires an $O(N)$ number of operations. Therefore, finding the velocity of all the blobs clearly requires an $O(N^2)$ number of operations. The accuracy of vortex methods increase as N is increased, consequently the $O(N^2)$ method is prohibitively expensive. This can be reduced to either an $O(N \log N)$ or $O(N)$ operation count by using special fast summation techniques. The use of these techniques to perform the velocity computation is therefore a must for a vortex method implementation.

The adaptive fast multipole method (Carrier *et al.*, 1988) is an algorithm that enables the computation of the velocity field in an $O(N)$ number of operations. The algorithm is well known. However, it is not easy to implement. In appendix A the method is described in considerable detail. Background information on fast summation techniques and theoretical details of the algorithm as applied to vortex blobs are also presented. Appendix A also provides pseudo-code useful to implement both body-cell treecodes (Barnes and Hut, 1986) and the AFMM. The appendix also develops a simpler method to evaluate one of the complicated list of cells used in the AFMM. Thus, using the AFMM described there, it is possible to rapidly evaluate the velocity field of the vortex blobs.

This chapter focuses on the application of the AFMM to vortex panels in order to rapidly compute the velocity due to the vortex panels on a collection of passive particles. The key contributions are as follows.

- Extension of the AFMM to handle linear vortex panels developed in section 3.6.1.
- An elegant generalization of the AFMM algorithm in order to handle passive particles.
- Development of an accelerated higher order panel method using the AFMM.
- Adaptation of the "fast multipole method without multipoles" (Anderson, 1992) to the AFMM framework. Application of the method to the cubic panels developed in section 3.6.2.
- Comparison of all the developed methods.

4.2 AFMM for linear vortex panels

The panel method can be used to apply the no-penetration solid wall condition for the flow of an incompressible fluid. Some details of the panel method were discussed in section 3.6.

Given the singularity distribution on the surface of the panels, it is easy to evaluate the velocity of the body on a collection of particles. If there are M panels and N vortex blobs, obtaining the velocity of the panels on the blobs is an $O(MN)$ computation. Usually the number of vortex blobs used in a vortex method is much larger (by one or two orders of magnitude) than the number of panels used to discretize the geometry. Despite the fact that $M << N$, the use of the AFMM for the blob-blob interactions makes it much faster than the panel-blob interactions. It is found that the time taken to evaluate the velocity of 400 linear panels on 10000 blobs is anywhere between 4 to 16 times (depending on the type of blob) more than the computation of the velocity due to the 10000 vortex blobs using the AFMM. The cubic panels (section 3.6.2) would require four times as much computational effort. Hence, it is imperative to improve the computational efficiency of the panel method. The panel-blob interactions can be significantly accelerated by adapting the AFMM for the vortex panels as done by Ramachandran *et al.* (2003). In this section, the significant results from that work are reproduced.

The approach used is to find expressions for the multipole expansions and local expansions for linear vortex panels. Direct velocity computations are performed using cubic panels. This eliminates the edge effect while speeding up the computation considerably. This approach is also called the "hybrid cubic/flat" panel method. The linear vortex panels were discussed in section 3.6.1 and cubic panels in section 3.6.2.

4.2.1 Multipole and local expansions

For the multipole expansion and local expansions, the equation for the velocity due to a linear panel as given in equation (3.6) is considered. Upon simplification this reduces to

$$V(z) = u - iv = \frac{i}{2\pi} A, \tag{4.1}$$

where A is given as

$$\begin{aligned} A &= (\gamma_2' - \gamma_1') \left[\frac{(z - z_2)}{\lambda'} \ln(z - z_2) - \frac{(z - z_1)}{\lambda'} \ln(z - z_1) \right] \\ &\quad + \gamma_2' \ln(z - z_2) - \gamma_1' \ln(z - z_1) + (\gamma_2' - \gamma_1'), \end{aligned} \tag{4.2}$$

where $\gamma_2' = \gamma_2 e^{-i\theta}$, $\gamma_1' = \gamma_1 e^{-i\theta}$, and $\lambda' = \lambda e^{i\theta}$. The terms $\gamma_1, \gamma_2,\ z_1,\ z_2, \lambda$, and θ are shown in Figs. 3.2 and 3.3. This form is very elegant from the perspective of the AFMM. The multipole expansion and other expressions for the terms of the form $q_i \ln(z - z_i)$ are already derived by Greengard and Rokhlin (1987). All that is required is to derive similar expressions for the terms of the form $q_i(z - z_i) \ln(z - z_i)$. With this, the AFMM can be applied to the linear vortex panel method. In the following the required expressions are derived.

A particle of singularity strength q located at z_1 is considered. The “velocity” that it induces at a point z is given as

$$V(z) = q(z - z_1)\ln(z - z_1). \tag{4.3}$$

This can be expressed as a multipole expansion about a circle centered at z_0 as follows:

$$\begin{aligned}
B(z) &= q(z - z_1)\ln(z - z_1) \\
&= q(z - z_0 - (z_1 - z_0))\left[\ln(z - z_0) - \sum_{k=1}^{\infty} \frac{1}{k}\left(\frac{z_1 - z_0}{z - z_0}\right)^k\right] \\
&= q(z - z_0)\left[\ln(z - z_0) - \sum_{k=1}^{\infty} \frac{1}{k}\left(\frac{z_1 - z_0}{z - z_0}\right)^k\right] \\
&\quad - q\left[(z_1 - z_0)\ln(z - z_0) - \sum_{k=1}^{\infty} \frac{(z_1 - z_0)^{k+1}}{k(z - z_0)^k}\right].
\end{aligned}$$

If there are m singularities of strength q_i located at positions z_i inside a circle of radius R with center z_0, then the total velocity is found from the above equation as a simple sum of the various singularities, which after some simplification reduces to

$$\begin{aligned}
B(z) &= \sum_{i=1}^{m} q_i(z - z_i)\ln(z - z_i) \\
&= a_1 + (a_0(z - z_0) - d_0)\ln(z - z_0) + \sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{(z - z_0)^k},
\end{aligned} \tag{4.4}$$

where

$$a_0 = \sum_{i=1}^{m} q_i, \qquad a_k = \sum_{i=1}^{m} \frac{-q_i}{k}(z_i - z_0)^k \quad \text{for} \quad k > 0$$

and

$$d_0 = \sum_{i=1}^{m} q_i(z_i - z_0), \qquad d_k = \sum_{i=1}^{m} \frac{-q_i}{k}(z_i - z_0)^{k+1} \quad \text{for} \quad k > 0.$$

Equation (4.4) is the multipole expansion for a set of m singularities that have a "velocity" field given by (4.3) and are located in a circle centered at z_0.

In order to shift the multipole expansion about the center z_0 to any other center, (4.4) is expanded to obtain a multipole expansion about the origin, as performed in (Carrier *et al.*, 1988) and (Greengard and Rokhlin, 1987). Using this it is easy to generalize the shift to any center. Hence, expanding (4.4) about the origin and carrying out simplifications results in the following:

$$\begin{aligned} B(z) &= a_1 - a_0 z_0 + (a_0 z - a_0 z_0 - d_0)\ln z \\ &\quad + \sum_{l=1}^{\infty} \frac{1}{z^l}\left[\left(\sum_{k=1}^{\infty} z_0^{l-k} \begin{pmatrix} l-1 \\ k-1 \end{pmatrix} (a_{k+1} - d_k)\right) + \frac{z_0^l}{l}\left(\frac{a_0 z_0}{l+1} + d_0\right)\right]. \end{aligned} \tag{4.5}$$

This expression governs the transfer of the multipole expansion given by (4.4) and is similar to Lemma 2.3 in (Greengard and Rokhlin, 1987).

Given a multipole expansion about a center one has to be able to convert it to a local (Taylor) expansion in a circular region of analyticity. Therefore, from (4.4) a power series representation has to be obtained. (4.4) can be split into two parts. The first part given below can be written as

$$\begin{aligned} a_1 + (a_0(z - z_0) - d_0)\ln(z - z_0) &= a_1 - (a_0 z_0 + d_0)\ln(-z_0) \\ &\quad + z\left(a_0 \ln(-z_0) + \frac{1}{z_0}(a_0 z_0 + d_0)\right) \\ &\quad - \sum_{k=2}^{\infty} \frac{1}{k}\left(\frac{z}{z_0}\right)^k \left(\frac{a_0 z_0}{k-1} - d_0\right), \end{aligned} \tag{4.6}$$

and the second part can be written as

$$\begin{aligned} \sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{(z - z_0)^k} &= \sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{z_0^k}(-1)^k + \frac{z}{z_0}\left(\sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{z_0^k} k(-1)^k\right) \\ &\quad + \sum_{l=2}^{\infty} \left(\frac{z}{z_0}\right)^l \left(\sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{z_0^k} \begin{pmatrix} l+k-1 \\ k-1 \end{pmatrix} (-1)^k\right). \end{aligned} \tag{4.7}$$

The right-hand sides of (4.6) and (4.7) are obtained using the result from

Lemma 2.4 in (Greengard and Rokhlin, 1987). Combining (4.6) and (4.7) and arranging the terms based on powers of z, the following power series is obtained:

$$\begin{aligned}
B(z) \;=\;& a_1 - (a_0 z_0 + d_0)\ln(-z_0) + \sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{z_0^k}(-1)^k \\
&+ z\left[a_0 \ln(-z_0) + \frac{1}{z_0}\left(a_0 z_0 + d_0 + \sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{z_0^k} k(-1)^k\right)\right] \qquad (4.8) \\
&+ \sum_{l=2}^{\infty}\left(\frac{z}{z_0}\right)^l \left[\sum_{k=1}^{\infty} \frac{a_{k+1} - d_k}{z_0^k} \left(\begin{array}{c} l+k-1 \\ k-1 \end{array}\right)(-1)^k - \frac{1}{l}\left(\frac{a_0 z_0}{l-1} - d_0\right)\right].
\end{aligned}$$

This equation corresponds to the final equation in Lemma 2.4 in (Greengard and Rokhlin, 1987). The first term is a constant, the second is linear in z, and the third term includes all the higher powers of z.

The last equation needed for the AFMM is the expression that transfers the Taylor's series about a center z_0 to that about any other center. Since this is nothing but a transfer of a generic Taylor's series, equation (A.9) can be used.

With equations (4.4), (4.5), (4.8), and (A.9) the AFMM can be used for a singularity that has a velocity field given by equation (4.3). Greengard and Rokhlin (1987) provide the corresponding expressions for a singularity that behaves as $q_i \ln(z - z_i)$. Combining their results with the ones derived above, it is possible to use the AFMM for the velocity due to a panel that has a linear geometry.

It can be seen from (4.1) and (4.2) that the velocity field due to the panels can be represented as the sum of a set of singularities that are of the form $z \ln z$ and $\ln z$ that are located at the end points, z_1 and z_2, of the panels. There are two singularities having strength $(\gamma_2' - \gamma_1')/\lambda'$ and $(\gamma_1' - \gamma_2')/\lambda'$ of the form $z \ln z$, and two more singularities of the form $\ln z$ with strengths γ_2' and $-\gamma_1'$ placed, respectively, at the ends of the panel, z_2 and z_1. Note that the coefficients due to the $\ln z$ singularity is added to that of d_k. Hence, for a single panel, it can be shown that

$$
\begin{aligned}
a_0 &= 0, && \text{(4.9a)}\\
a_k &= \frac{-(\gamma_2' - \gamma_1')}{k\lambda'}\left((z_2 - z_0)^k - (z_1 - z_0)^k\right), && \text{(4.9b)}\\
d_0 &= 0, && \text{(4.9c)}\\
d_k &= \frac{1}{k}\left[\frac{(\gamma_1' - \gamma_2')}{\lambda'}\left((z_2 - z_0)^{k+1} - (z_1 - z_0)^{k+1}\right)\right. && \text{(4.9d)}\\
&\qquad \left. + \gamma_2'(z_2 - z_0)^k - \gamma_1'(z_1 - z_0)^k\right],\\
a_{k+1} - d_k &= \frac{1}{k}\left[\frac{(\gamma_2' - \gamma_1')}{\lambda'(k+1)}\left((z_2 - z_0)^{k+1} - (z_1 - z_0)^{k+1}\right)\right. && \text{(4.9e)}\\
&\qquad \left. - \gamma_2'(z_2 - z_0)^k + \gamma_1'(z_1 - z_0)^k\right],
\end{aligned}
$$

where $k > 0$. Since a_0 and d_0 are zero for a single panel, (4.4), (4.5), and (4.8) simplify greatly. It is now possible to use the AFMM for any number of panels by adding to the above coefficients, (4.9), the effect of each panel and substituting the result in (4.4), (4.5), (4.8), and (A.9). The series expansions need to be truncated up to some order p depending on the accuracy desired. In Carrier *et al.* (1988), given an accuracy ϵ, the number of terms to be considered in the multipole expansion is given as $p = \ln_2 \epsilon$. Since there is a multiplicative factor of z in the case of a panel, for the same accuracy to be obtained, p must be chosen as $p = 1 + \ln_2 \epsilon$.

4.2.2 AFMM with passive particles

In section A.2.1 the traditional algorithm for organizing the particles into a quad-tree structure was discussed. This method only handles the case where the particles influence each other. Usually, in a vortex method, the effect of vortex blobs on vortex panels, or on other particles in the fluid is required. In the present case, the effect of vortex panels on vortex blobs or other particles in the fluid is required. This needs to be handled efficiently in the context of the AFMM. It is possible to generalize the tree generation algorithm in order to handle this efficiently. This generalization also has other applications. In (Ramachandran *et al.*, 2001, Under review) this approach of generalizing the tree algorithm is used to handle random walks in the presence of arbitrary two-dimensional geometries. This section discusses a generalized tree structure to group particles.

The basic idea is simple. The particles involved in the computation are viewed as *causes* or *effects*. As *causes*, they influence the *effects*. The *effects* are merely influenced by the *causes*. Treating the particles in this manner results in a very general algorithm for the domain decomposition that can be used efficiently in a wide variety of schemes. Consider the following example where the AFMM is used to compute velocities due to vortex blobs. The computational domain contains blobs and passive particles. The passive particles by definition are *effects* as they do not induce a velocity on anything. The blobs influence all particles (blobs and passive particles). Therefore, the blobs are *causes* and both the blobs and passive particles are *effects*. This example brings out the fact that the interacting elements have two types of manifestations, *causes* and *effects*.

Using the above, a domain decomposition algorithm for the particular case of the AFMM as applied to blobs and tracer particles can be expressed as follows.

A. If there are a large number of *causes* and *effects* in a cell, then it would be inefficient to perform a direct computation between the *cause* and *effect* elements in the cell. Therefore, the cell needs to be split into child/daughter cells.

B. If there are a very small number of *causes* and *effects* in the cell, then it is faster to perform the direct computation between the *cause* and *effect* elements in the cell rather than further subdividing the cell. Therefore, the cell should be left alone and stored as a leaf/childless cell.

C. If there are a large number of *cause* elements but small number of *effects* in the cell b, then the decision to split depends on the nature of its colleagues. This is because if there are a larger number of *effects* in any of the colleagues, c, then it would be expensive to compute the interactions between the *causes* in cell b and the *effects* in c. Hence, if any of the colleagues c, of cell b have a large number of *effects*, then cell b is split into child cells. This will reduce the number of computations that have to be performed between the *causes* of cell b and the *effects* in c. If there are no such colleague cells then the cell b can be stored as a leaf/childless cell.

D. Similarly, if there are a large number of *effect* elements but small number of *causes* in the cell b, then it should be split if any of its colleagues c, have a large number of *causes*. If there are no such colleague cells then the cell b can be stored as a leaf/childless cell.

The notion of what is a "large" number of causes or effects can be determined from numerical experiments. The present work uses two parameters called `max_cause` and `max_effect` to specify the allowed number of causes or effects per cell. Depending on the nature of the interactions between the causes and effects and their distribution, these parameters take different optimal values.

As opposed to the traditional domain decomposition employed in the AFMM, where only causes (blobs, charges, etc.) are considered, the above domain decomposition (A through D) can handle situations where the the velocity on blobs and passive particles are computed. Strickland and Baty (1998) also propose an alternative scheme to handle independent "source" (cause) and "target" (effect) fields. Their scheme creates separate cells for the sources and targets whereas the present scheme creates a single set of cells with each cell containing both causes and effects.

In the case of the AFMM applied to vortex panels, it is clear that the panels are the causes and other particles are effects. Hence, by using the above algorithm, it is possible to efficiently handle the effect of the panels on other particles.

Using the above approach it can be shown that the evaluation of the velocities of N cause particles on M effects is an $O(N + M)$ operation. Table 4.1 shows the time taken to compute the velocity due to N vortex blobs of known strengths placed on the surface of an ellipse (with a $3 : 1$ axis ratio) on M randomly distributed particles inside a square of side 10 units. The effect of the blobs on themselves are not computed. As can be seen, despite the fact that both N and M double, only an approximately two-fold increase is seen in the computational time indicating that the computation is $O(N + M)$.

This is an important result since the naive approach of computing the velocity of the N causes on the M effects results in an $O(NM)$ computation. For this reason, Bakhoum and Board (1996) find that the AFMM is not suitable for problems where N is small and M very large. However, the present approach of extending the AFMM using cause and effects clearly eliminates this problem by making the computation $O(N + M)$.

Table 4.1: CPU time for velocity of N cause particles on M effects.

N (cause)	M (effect)	CPU time (secs)
100	100	0.00502
200	200	0.01326
400	400	0.02683
800	800	0.06241
1600	1600	0.10450
3200	3200	0.24375
6400	6400	0.40861
12800	12800	0.95652
25600	25600	1.67095

4.2.3 Numerical results

Table 4.2: Comparison of CPU times for cubic, flat, and fast panel methods.

Time taken (seconds)	Cubic panel	Flat panel	Fast panel method
Case 1	43.88	11.18	0.42
Case 2 (wake region)	43.53	10.95	0.12

The case of flow past a circular cylinder, centered at the origin with a radius of 1 unit, with 400 panels is considered. A set of 10000 uniformly spaced particles in a square region is considered. The coordinates of two diagonally opposite corners of the region are $(-2.0, -2.0)$ and $(2.0, 2.0)$. This is called Case 1. The velocity at each of these 10000 points is computed using the cubic panel method, the linear panel method, and finally using the developed fast summation technique. In the above computation the accuracy for the FMM is chosen as $\epsilon = 10^{-6}$. `max_cause` and `max_effect` are chosen as 7 panels and particles, respectively. The corresponding hierarchical mesh is shown in Figure 4.1. As is evident from Table 4.2, the fast panel method is highly efficient. There is a 100 fold increase as compared to the cubic method and a 26 fold increase as compared to the traditional flat panel method. It is significant to note that if the domain of interest (the region where the particles are distributed) is further away from the body, a much greater increase is seen. To demonstrate this, a square region having left bottom corner $(1.0, -1.0)$ and top right corner $(3.0, 1.0)$ with 10000 uniformly spaced points (case 2), is considered. This region is in the wake of the cylinder. For case 2 the results

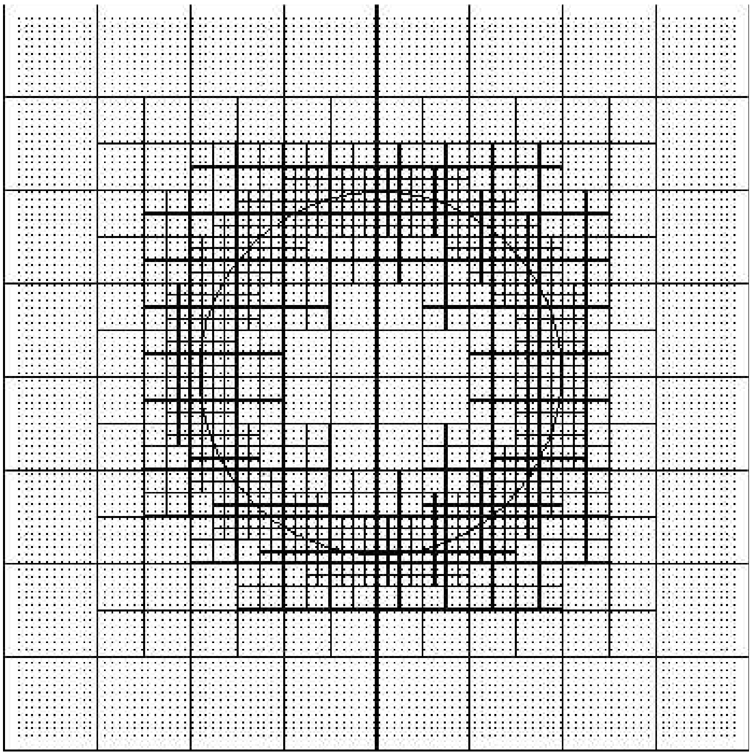

Figure 4.1: Hierarchical mesh for the flow past a circular cylinder with 400 panels and 10000 passive particles for case 1.

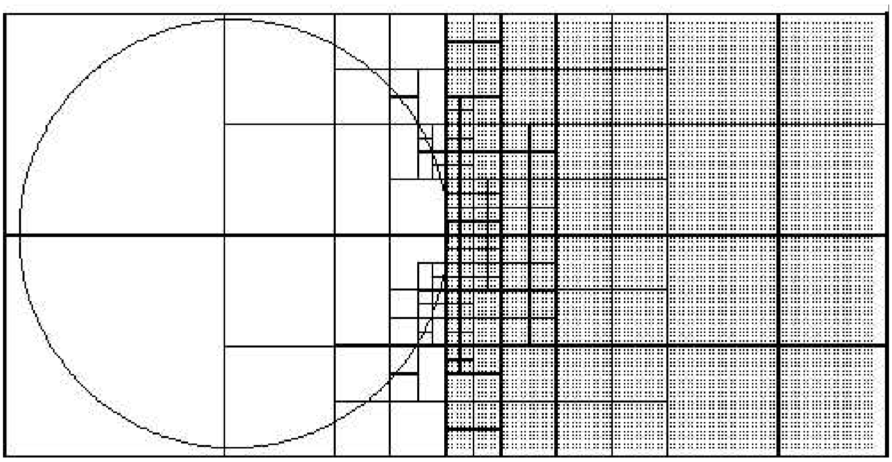

Figure 4.2: Hierarchical mesh for the flow past a circular cylinder with 400 panels and 10000 passive particles for case 2.

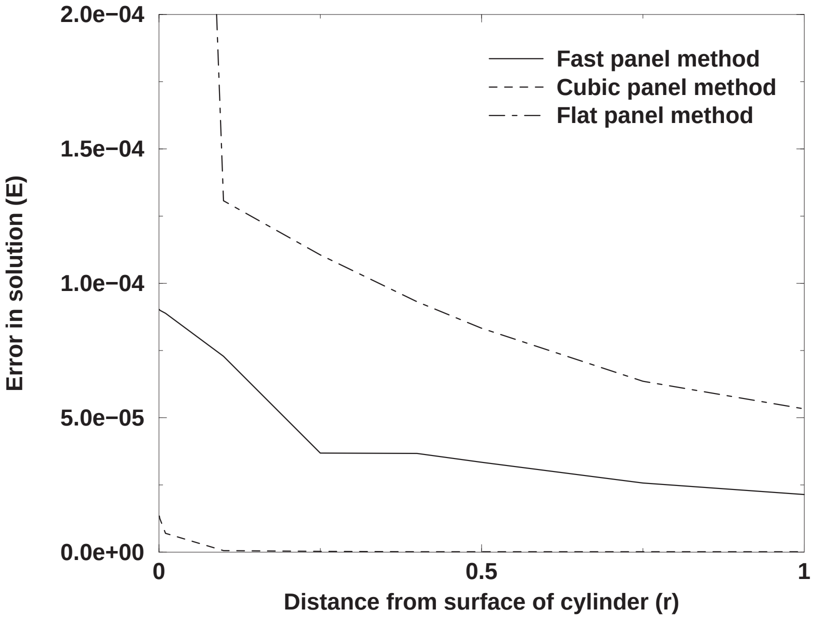

Figure 4.3: Comparison of the error E versus distance from the surface of a circular body obtained by using the fast panel method, the flat panel method, and the cubic panel method.

are again shown in Table 4.2. The corresponding mesh is shown in Figure 4.2. It is clear from Figure 4.2 that the refinement of the mesh becomes finer as the particles get closer to the body and coarser as the particles move away from the body. From the results it can be observed that as one moves away from the body, the computation becomes more efficient (in this case by a factor of 3.5). There is certain to be an upper limit of efficiency gained for a given computation but as is evident from the above results, more than two orders of magnitude in speed increase are achievable.

The fast panel technique has very significant advantages from the perspective of computational efficiency. In the following, the accuracy of the method is compared to that of the cubic panel technique. The flow past a cylinder is used as a benchmark. Equation (3.13) is used to compute the error on a ring of particles away from the cylinder. The ring is made up of around 1000 particles. 200 panels are used for the cylinder. Figure 4.3 plots the variation of the error, E, versus the distance of the ring from the surface the cylinder, r. The computation was performed using a maximum of 7 panels and particles per cell. The fast panel method is more accurate than the flat panel technique because (a) the cubic panel method is used to compute the strengths on the panel and (b) in the vicinity of the panels the cubic panel method is used to find the velocity.

As can be clearly seen from the above results, the hybrid cubic/flat panel method is efficient and eliminates the edge effect. It is to be noted that the method developed allows one to rapidly compute the velocity field due to a known distribution of singularity on the panels. The approach can therefore be used to solve for the unknown strengths of the panels iteratively as done by Rokhlin (1985). The idea is to compute increasingly accurate approximations of the panel strengths by rapidly computing the velocity field due to the panels at the control points. This approach is typically useful when there are a large number of panels and when the geometry of the body is changing. However, this is not implemented in the present work as described in the end of section 3.6.3.

4.3 AFMM for higher order panels

The hybrid cubic/flat fast panel method discussed in the previous section is not as accurate as the cubic panel method. This is seen in figure 4.3. It is therefore important to extend the AFMM to work with cubic geometry panels instead of linear panels. In (Ramachandran *et al.*, In press) a technique to perform a fast multipole summation using panels of any order is developed. This allows for a highly accurate and fast algorithm for two-dimensional panel methods. The method developed there is reviewed in this section. The algorithm is demonstrated using cubic panels. The method developed is compared with the hybrid algorithm (Ramachandran *et al.*, 2003) developed in section 4.2 and also compared with Anderson's technique (Anderson, 1992) extended to the current problem.

4.3.1 Multipole and local expansions

Consider a higher order panel as shown in Figure 4.4. Figure 3.4 plots the same in the z' plane with its chord oriented along the x'-axis. Given a vorticity distribution $\gamma(\zeta)$ the complex velocity at a point z' (in the z' plane) due to the panel is given

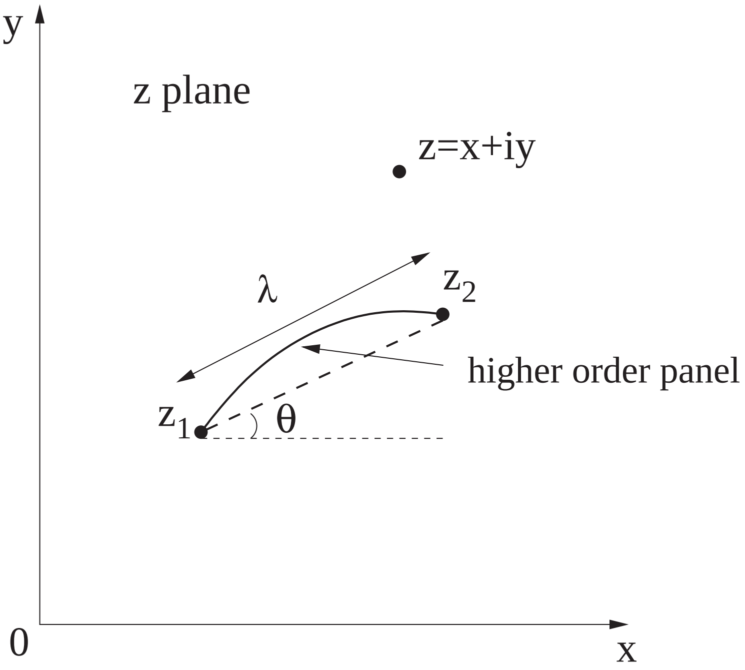

Figure 4.4: Higher order panel having a chord length λ in the z plane.

as,

$$V(z') = u' - iv' = \frac{-i}{2\pi}\int_0^{\lambda} \frac{\gamma(\zeta)}{z' - (\zeta + i\eta)} d\zeta. \tag{4.10}$$

Substituting $\xi = \zeta + i\eta$, and performing a binomial expansion results in,

$$\begin{aligned} u' - iv' &= \frac{-i}{2\pi}\int_0^{\lambda} \frac{\gamma(\zeta)}{z' - \xi} d\zeta \\ &= \frac{-i}{2\pi}\int_0^{\lambda} \frac{\gamma(\zeta)}{z'}\left[1 + \frac{\xi}{z'} + \frac{\xi^2}{z'^2} + \dots\right] d\zeta \\ &= \frac{-i}{2\pi}\sum_{j=1}^{\infty}\int_0^{\lambda} \frac{\gamma(\zeta)\xi^{j-1}}{z'^j} d\zeta \end{aligned} \tag{4.11}$$

Without loss of generality, if z_1 of the panel is assumed to be at the origin, then $z' = ze^{-i\theta}$ and $u - iv = e^{-i\theta}(u' - iv')$, and the above equation reduces to,

$$u - iv = \frac{-i}{2\pi}\sum_{j=1}^{\infty} \frac{e^{i(j-1)\theta}}{z^j}\int_0^{\lambda} \gamma(\zeta)\xi(\zeta)^{j-1} d\zeta. \tag{4.12}$$

This can be written as,

$$u - iv = \frac{-i}{2\pi}\sum_{j=1}^{\infty} \frac{A_j}{z^j}, \tag{4.13}$$

where,

$$A_j = e^{i(j-1)\theta} \int_0^{\lambda} \gamma(\zeta)\xi(\zeta)^{j-1} d\zeta. \tag{4.14}$$

The complex potential of the higher order panel can also be obtained as follows,

$$\begin{aligned}
\Phi = \phi + i\psi &= \frac{-i}{2\pi} \int_0^{\lambda} \gamma \ln(z' - \xi) d\zeta \\
&= \frac{-i}{2\pi} \left(\ln(z') \int_0^{\lambda} \gamma d\zeta - \sum_{k=1}^{\infty} \frac{1}{z'^k} \int_0^{\lambda} \frac{\xi^k}{k} d\zeta \right) \\
&= \frac{-i}{2\pi} \left[P_0 \ln(z') - \sum_{k=1}^{\infty} \frac{P_k}{z'^k} \right] \qquad (4.15)
\end{aligned}$$

where,

$$P_0 = \int_0^{\lambda} \gamma d\zeta \quad ; \quad P_k = \frac{1}{k} \int_0^{\lambda} \xi^k \; \gamma \; d\zeta. \tag{4.16}$$

In the present work one is interested in evaluating the velocity field due to the panels. Hence, the multipole method is developed with that in mind. The analysis of the truncation errors is performed only for the velocity field. The truncation error for the complex potential can also be easily computed in a similar fashion as done for the velocity field.

Given $\xi(\zeta)$, $\gamma(\zeta)$ and θ, A_j can be readily computed. The series (4.13) converges if $|\xi| < |z|$. It is reasonable to assume that the panel is completely contained inside a circle centered at the origin having radius λ, i.e. $|\xi(\zeta)| < \lambda$. Given this, it is easy to see from (4.14) that the error involved in truncating the series to a finite number of terms p is,

$$\left| V(z) + \frac{i}{2\pi} \sum_{j=1}^{p} \frac{A_j}{z^j} \right| \leq \frac{\Gamma}{2\pi\lambda(\varrho - 1)} \left(\frac{1}{\varrho} \right)^p \tag{4.17}$$

where,

$$\Gamma = \int_0^{\lambda} |\gamma(\zeta)| d\zeta$$

and $\varrho = |z|/\lambda$.

Hence, equations (4.13) and (4.14) can be used to obtain a fast multipole expansion for higher order panels. Note that no assumptions on the nature of

$\gamma(\zeta)$ or $\xi(\zeta)$ are made at this point and the condition $|\xi(\zeta)| < \lambda$ for $0 \leq \zeta \leq \lambda$ is only used to bound the truncation error. The coefficients A_j are to be computed by numerical integration. It is to be noted that equation (4.13) is a multipole expansion about the point z_1 (the first point) of the panel as shown in figure 4.4. For a cubic panel as used in section 3.6.2, $\xi = \zeta + i(a_1\zeta + a_2\zeta^2 + a_3\zeta^3)$. It is also to be noted that if the panels do not deform or change orientation, the coefficients, A_j, are constant and need be computed only once. If the panels only change in orientation, the entire integral need not be evaluated and the coefficients need to be multiplied by a different value of $e^{i(j-1)\theta}$. If the panels deform, the coefficients must be recomputed. Given equations (4.13) and (4.14), the various expressions for the fast multipole method can be derived as follows.

Multipole expansion for a collection of panels

Given n panels placed at points z_k inside a circle of radius R, the multipole expansion for the velocity field of the panels is,

$$V(z) = u - iv = \frac{-i}{2\pi}\sum_{k=1}^{n}\sum_{j=1}^{\infty}\frac{A_{kj}}{(z - z_k)^j}$$

where A_{kj} are the coefficients as given in equation (4.14). The above equation can be expressed as a multipole expansion about a circle centered at z_0 as follows,

$$\begin{aligned} V(z) &= \frac{-i}{2\pi}\sum_{k=1}^{n}\sum_{j=1}^{\infty}\frac{A_{kj}}{((z - z_0) - (z_k - z_0))^j} \\ &= \frac{-i}{2\pi}\sum_{k=1}^{n}\sum_{j=1}^{\infty}\frac{A_{kj}}{(z - z_0)^j}\left[1 - (z_k - z_0)/(z - z_0)\right]^{-j}. \end{aligned}$$

By grouping powers of $(z - z_0)$ it can be seen that,

$$V(z) = \frac{-i}{2\pi}\sum_{j=1}^{\infty}\frac{a_j}{(z - z_0)^j} \tag{4.18}$$

where,

$$a_j = \sum_{k=1}^{n}\sum_{m=1}^{j} A_{km}\begin{pmatrix} j-1 \\ m-1 \end{pmatrix}(z_k - z_0)^{j-m} \tag{4.19}$$

Equations (4.18) and (4.19) are equivalents to those of Lemma 2.1 in Carrier *et al.* (1988). Equation (4.18) is nothing but the sum of the transfers of the multipole expansions of the n panels, each starting at z_k, to a circle of radius R centered at z_0. Thus, when this expression is truncated to p terms, the error in the velocity is,

$$\left| V(z) + \frac{i}{2\pi} \sum_{j=1}^{p} \frac{a_j}{(z-z_0)^j} \right| = \left| \frac{i}{2\pi} \sum_{j=p+1}^{\infty} \frac{a_j}{(z-z_0)^j} \right|,$$

where a_j is given in equation (4.19). From equation (4.14) it can be seen that,

$$|A_{km}| \leq \Gamma_k \lambda_k^{m-1}, \quad \Gamma_k = \int_0^{\lambda_k} |\gamma_k(\zeta)| d\zeta,$$

where λ_k is the chord length of the k'th panel. Given this and the fact that all the z_k's lie inside a circle of radius R, a_j can be clearly bounded as,

$$\begin{aligned} |a_j| &\leq \sum_{k=1}^{n} \sum_{m=1}^{j} \Gamma_k \lambda_k^{m-1} \begin{pmatrix} j-1 \\ m-1 \end{pmatrix} R^{j-m} \\ &\leq \Delta \sum_{m=1}^{j} \lambda^{m-1} \begin{pmatrix} j-1 \\ m-1 \end{pmatrix} R^{j-m} \end{aligned}$$

where $\Delta = \sum_{k=1}^{n} \Gamma_k$ and $\lambda = \max(\lambda_k)$. Hence,

$$|a_j| \leq \Delta (R+\lambda)^{j-1} \tag{4.20}$$

Therefore,

$$\begin{aligned} \left| V(z) + \frac{i}{2\pi} \sum_{j=1}^{p} \frac{a_j}{(z-z_0)^j} \right| &\leq \left| \frac{i\Delta}{2\pi} \sum_{j=p+1}^{\infty} \frac{(R+\lambda)^{j-1}}{(z-z_0)^j} \right| \\ &= \frac{\Delta}{2\pi(R+\lambda)(c-1)} \left(\frac{1}{c} \right)^p \end{aligned} \tag{4.21}$$

where $c = |z-z_0|/(R+\lambda)$. It is to be noted that in this case, any panel having a starting point z_k, lying inside the circle D, centered at z_0 with radius R, is considered for the multipole expansion. Clearly, the circle of radius $R+\lambda$ will completely contain all the panels that have a starting point inside D. Let the radius of a cell containing panels be R_{cell}. If the panel mid-chord position is used as the determining criterion for a panel to be part of a cell, then clearly the radius of the circle that completely contains all the panels in that cell is $R_{cell} + \lambda/2$. This is because only half a panel length will be partly outside the cell. From this it is easy to see that

$$R + \lambda \leq R_{cell} + \lambda/2. \tag{4.22}$$

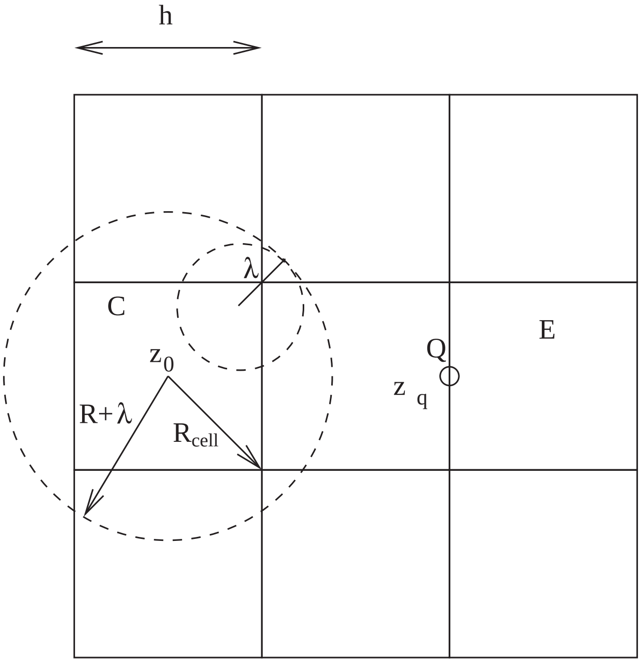

Figure 4.5: Illustration of radius of convergence for panels.

While the result in equation (4.21) has the same form ($O(c^{-p})$) for the error term as in the fast multipole method of (Greengard and Rokhlin, 1987; Carrier *et al.*, 1988), the value of c is different. The reason for the difference can be explained as follows. Consider the cell C centered at z_0, shown in Figure 4.5. The cell contains a panel of length λ, with its mid point at the corner of the cell and at an angle 45° to the horizontal. The size of the cell is h. The radius of the cell $R_{cell} = h\sqrt{2}/2$. From equation (4.17) it is clear that the panel's multipole expansion about its starting point converges only outside a circle of radius λ centered about its starting point. Thus, it is easy to see that the multipole expansion for the cell about its center converges outside a circle of radius $R_{cell} + \lambda/2$. This is the same result as in equation (4.22).

While implementing the adaptive fast multipole algorithm, the multipole expansion of the cell C can be evaluated on cells that are well separated from it. The cell E in Figure 4.5 contains z_q, the closest point to z_0. The error in the multipole expansion at this point is governed by,

$$c = \left| \frac{z_q - z_0}{R + \lambda} \right| = \frac{3h}{h\sqrt{2} + \lambda}$$

If $h = \beta\lambda$, then,

$$c = \frac{3\beta}{\beta\sqrt{2} + 1}. \tag{4.23}$$

Generally, c is not greater than 2 as in the case of the original fast multipole method where $c = 3/\sqrt{2}$. Given a value of c and a desired precision, ϵ, the number of terms in the series necessary, p, can be determined from equation (4.21). In order to employ a smaller number of terms p, β needs to be chosen carefully. It is important to note that the size of the cells used in the multipole method should be limited by $\beta\lambda$. This result also applies to the fast multipole method of Ramachandran *et al.* (2003) which is discussed in section 4.2.

Shifting the center of a multipole expansion

Equation (4.18) is the multipole expansion for a collection of panels in a circle of radius R centered at z_0. This expansion, when translated to the origin, produces a multipole expansion that converges outside a circle centered at the origin with radius $R + \lambda + |z_0|$. The resulting multipole expansion is given by

$$V(z) = \frac{-i}{2\pi} \sum_{j=1}^{\infty} \frac{b_j}{z^j} \tag{4.24}$$

where,

$$b_j = \sum_{k=1}^{j} a_k \begin{pmatrix} j-1 \\ k-1 \end{pmatrix} z_0^{j-k} \tag{4.25}$$

and a_k is as given in equation (4.19). This expression is equivalent to Lemma 2.2 in (Carrier *et al.*, 1988). From equation (4.20) a bound for b_j can be obtained as,

$$\begin{aligned} |b_j| &\leq \Delta \sum_{k=1}^{j} (R+\lambda)^{k-1} \begin{pmatrix} j-1 \\ k-1 \end{pmatrix} |z_0|^{j-k} \\ &= \Delta(R+\lambda+|z_0|)^{j-1} \end{aligned} \tag{4.26}$$

From the above, the error in truncating equation (4.24) to p terms is bounded as,

$$\begin{aligned} \left| V(z) + \frac{i}{2\pi} \sum_{j=1}^{p} \frac{b_j}{z^j} \right| &= \left| \frac{i}{2\pi} \sum_{j=p+1}^{\infty} \frac{b_j}{z^j} \right| \\ &\leq \frac{\Delta}{2\pi(|z| - (R+\lambda+|z_0|))} \left(\frac{R+\lambda+|z_0|}{|z|} \right)^p \end{aligned} \tag{4.27}$$

Conversion of multipole expansion to a local expansion

Given a multipole expansion (4.18) about a circle D_0 of radius R and centered at z_0 such that $|z_0| > (c+1)R$ with $c > 1$, the multipole expansion can be described by a power series in z that converges inside a circle, D_2, centered at the origin having radius R,

$$V(z) = \sum_{j=0}^{\infty} c_j z^j \tag{4.28}$$

where,

$$c_j = \frac{-i}{2\pi z_0^j} \sum_{k=1}^{\infty} (-1)^k \frac{a_k}{z_0^k} \begin{pmatrix} j+k-1 \\ k-1 \end{pmatrix}. \tag{4.29}$$

This is equivalent to Lemma 2.3 in (Carrier *et al.*, 1988). The derivation for the error term is similar to that derived in (Greengard and Rokhlin, 1987) for the local expansions. However, the extent of the panels modifies the results slightly. The error introduced when the series in equation (4.28) is truncated to p terms is bounded by,

$$\left| V(z) - \sum_{j=1}^{p} c_j z^j \right| \leq \frac{2\Delta e(p(R+\lambda)+cR)(c+1)}{\pi c R(c-1)(R+\lambda)} \left(\frac{1}{c} \right)^{p+1}, \tag{4.30}$$

where $p \geq 2c/(c-1)$, and e is the base of the natural logarithm. The derivation is very similar to that in (Greengard and Rokhlin, 1987) but is a little involved. The full derivation is available in Ramachandran *et al.* (In press). The expression is very similar to equation (A.8). Also note that c is as defined in (Greengard and Rokhlin, 1987) and used in equation (A.8) and is *not* a function of λ.

Finally, given any local expansion centered about z_0 and the coefficients a_k, the center of the local expansion can be shifted to the origin without loss of any accuracy using the equation (A.9).

Using equations (4.18), (4.24), (4.28) and (A.9) the fast multipole algorithm can be applied to higher order panels. As detailed in section 4.3.1, it is to be noted that the size of the cell is limited by the length of the panels. The parameter c in equation (4.21) is modified as in equation (4.23). The direct computation of the velocity is dependent on the actual higher order panel chosen. In the present work a cubic panel geometry with a linear distribution of vorticity is used. The velocity field due to this type of panel is obtained using equation (3.10).

Implementation issues

The expression for the multipole coefficients for a panel in equation (4.14) seems to indicate that they need re-computation when the value of $\gamma(\zeta)$ changes. For a linear distribution of vorticity, the expression for A_j is given below,

$$A_j = e^{i(j-1)\theta}\left(\gamma_1 \int_0^{\lambda} \xi(\zeta)^{j-1} d\zeta + \frac{\gamma_2-\gamma_1}{\lambda}\int_0^{\lambda} \zeta\ \xi(\zeta)^{j-1} d\zeta\right), \tag{4.31}$$

where γ_1 and γ_2 are the values of the vorticity at the ends of the panel. Clearly the integrals are independent of γ_1 and γ_2 and need not be re-computed unless the geometry of the panel or its orientation change. In the present work the integrals in the equation are evaluated using Simpson's rule.

At the end of section 4.3.1 it was mentioned that in order to obtain accurate results the cell size needs to be limited based on the panel size. This can be done efficiently in the following manner. The maximum and minimum panel lengths

$(\lambda_{max}, \lambda_{min})$ are computed when assigning the panels to the cell at level 0. If there is a significant length variation in the panels, then the λ_{max} at higher levels must be recomputed. If not, the same value of λ_{max} can be used at all levels. In the present implementation, it is assumed that if $\lambda_{max}/\lambda_{min} > 1.5$ then there is reasonable variation in the panel lengths. When splitting a cell C, into four daughter cells, the λ_{max} of cell C is set as the λ_{max} of the panels in the children. When splitting the newly created daughter cells, if the cell size h is such that $h < 2\beta\lambda_{max}$, then λ_{max} is recomputed for the cell. After re-computation of λ_{max}, if $h > 2\beta\lambda_{max}$ then the cell can be split. If not the split cell will be smaller than the required length and is not split. The factor 2 arises because the split cell's length will be $h/2$. Using such a scheme, the need to compute the maximum panel length at each level is eliminated and is computed only when necessary.

4.3.2 Anderson's FMM without multipoles

In (Ramachandran *et al.*, In press) the authors also use the "FMM without multipoles" method (Anderson, 1992) to accelerate the cubic panel method. The method uses Poisson's integral formula in order to obtain equivalents for the multipole expansion. Given a collection of point charges or vortices, the method uses inner and outer ring approximations of the potential computed using Poisson's integral formula to represent the cluster of particles as a single computational entity and accelerate the computations. The advantage of this approach is that there is no need to obtain expressions for the multipole expansions as done normally with the FMM. Anderson describes the algorithm and its implementation from a multi-grid perspective. Ramachandran *et al.* (In press) implement Anderson's algorithm in the context of the AFMM.

Anderson's method works by approximating the potential or stream function of a cluster of particles by inner and outer ring approximations using a careful discretization of the Poisson's integral. Poisson's integral formula is used for particles that have a $\log(r)$ potential. For vortex blobs and panels this corresponds to their stream function. Anderson defines an outer ring approximation for the stream function as follows,

$$
\begin{aligned}
\psi(r,\theta) &\approx \kappa\log(r) + \frac{1}{2\pi}\sum_{i=1}^{K} f(s_i)F_i h \qquad (4.32)\\
F_i &= \frac{1-\left(\frac{a}{r}\right)^2 - 2\left(\frac{a}{r}\right)^{M+1}\cos((M+1)(\theta-s_i)) + 2\left(\frac{a}{r}\right)^{M+2}\cos(M(\theta-s_i))}{1-2\left(\frac{a}{r}\right)\cos(\theta-s_i)+\left(\frac{a}{r}\right)^2},
\end{aligned}
$$

where, r, θ are the co-ordinates of a point where the stream function is approximated. $K = 2M+1$, s_i is the angle of the integration point on the ring of radius a, $h = 2\pi a/K$, $\kappa = \sum_{i=1}^{N}(\kappa_i/2\pi)$, $f(s_i) = \Psi(a, s_i) - \kappa\log(a)$ and Ψ is the stream function induced by N vortex panels (or vortex blobs) of strengths κ_i. The inner ring approximation is similar and given as,

$$
\begin{aligned}
\psi(r,\theta) &\approx \frac{1}{2\pi}\sum_{i=1}^{K} f(s_i)G_i h \qquad (4.33)\\
G_i &= \frac{1-\left(\frac{r}{a}\right)^2 - 2\left(\frac{r}{a}\right)^{M+1}\cos((M+1)(\theta-s_i)) + 2\left(\frac{r}{a}\right)^{M+2}\cos(M(\theta-s_i))}{1-2\left(\frac{r}{a}\right)\cos(\theta-s_i)+\left(\frac{r}{a}\right)^2}
\end{aligned}
$$

where, $f(s_i)$ is the stream function due to particles outside the inner ring evaluated on the inner ring. For a ball of radius R, containing a collection of particles (or panels) Anderson suggests using a value of $a = 2R$ for the outer ring approximation. For an inner ring approximation the value of $a = R/2$ can be chosen. In the implementation of this technique, the coefficients $f(s_i)$ are stored at the integration points on the inner and outer rings.

Given the above, it is possible to use Anderson's method with the AFMM. The procedure is simple and is performed as follows.

- The multipole expansions for the childless cells are computed using outer ring approximations of the stream function.
- The multipole expansions from daughter cells, D, are are shifted to their parents, P, by evaluating the outer ring approximation of D on P's outer integration points and accumulating the outer ring coefficients of P.
- Given a multipole expansion (or an outer ring approximation) of a cell C, a local expansion (inner ring approximation) at a well separated cell S, can be obtained by computing the outer ring approximation of the stream function due to C on S's inner ring integration points and adding them to the inner

ring coefficients of S. This corresponds to obtaining the interactions due to cells in the V_b list as described in (Carrier *et al.*, 1988).

- A local expansion is translated from a cell P, to a child cell C, by evaluating the inner ring approximation of P at the inner integration points of C and adding the result to the inner ring approximation coefficients of C.
- The interactions due to a cell W, in the W_b list of a cell B are computed by evaluating the stream function due to the outer ring of W on particles inside B.
- The X_b interactions on cell B due to a cell X belonging to the X_b list is found by evaluating the stream function due to each panel in X on the inner ring integration points of B.

The algorithm does not require the use of explicit expressions for the multipole expansions and the local expansions as done for the AFMM in sections 4.2 and 4.3. This makes the method easy to extend to situations where expressions for the multipole expansion are not easy to derive.

Before implementing Anderson's method a few issues must be noted. The expression for F_i and G_i in equations (4.32) and (4.33) are indeterminate when the evaluation point is at any of the integration points, i.e. when $r = a$ and $\theta = s_i$. However, the functions are very well behaved and it can be shown that

$$\lim_{r\to a;\ \theta\to s_i} F_i = \lim_{r\to a;\ \theta\to s_i} G_i = 2M + 1. \tag{4.34}$$

It is to be noted that the limiting value is the same independent of the order in which the limits are taken.

If one is interested in the evaluation of the velocity field and not the stream function then some care must be taken. In order to obtain the velocity field one can differentiate equations (4.32) and (4.33). From the expressions for G_i, F_i and the equation (4.34) it can be shown that the derivatives of F_i and G_i will be singular near the integration points. If the inner ring lies inside a cell, the points at which the velocities are evaluated may be near the integration points (i.e. $r \to a$ and $\theta \to s_i$). Hence, inaccurate results will be obtained for such points. This problem is demonstrated in Ramachandran *et al.* (In press). In order to avoid such problems, the ring radii must be chosen carefully. This must be done without loss of accuracy. The inner ring radius must be chosen such that it is outside the cell where the velocity is to be evaluated. Similarly, the outer ring radius must be chosen such that it does not intersect any well separated cell. It is easy to see that the inner ring radius should be such that $a > R$ (R is the the cell radius) and for the outer ring it must be such that $a < 3R$. In Ramachandran *et al.* (In press) it is shown that given a cell of length h and radius $R = h\sqrt{2}/2$, choosing $a = 0.75h$ for the inner ring radius works well without any significant loss of accuracy. Similarly, for the outer ring radius a value of $a = 1.4h$ is chosen. This ensures that the outer ring does not lie inside any well separated cells. In this manner it is possible to obtain accurate velocity fields using Anderson's technique.

4.3.3 Influence of cell size on accuracy

Consider a panel placed at the edge of a cell as shown in Figure 4.6. The cell C contains a single panel at one corner such that the mid-chord of the panel is just inside it. The cells, $L1$, $L2$ and $L3$ are well separated from C. The multipole expansion due to C is evaluated on these cells. The multipole expansion of C is also transferred to these cells as a local expansion and this local expansion is evaluated inside these cells. The contours of the relative error in the velocity due to the panel in these cells is plotted using the cubic fast panel method (section 4.3), hybrid cubic/flat method (section 4.2) and Anderson's scheme (section 4.3.2). As would be expected, it is found that the point z_m as shown in the figure has the maximum error. The cells $X1$ and $X2$ are cells which are in the W_b list of C. Hence, C is in the X_b list of $X1$ and $X2$. The X_b interactions on these two are evaluated from cell C. For this case the maximum error occurs at the point shown as z_x in figure 4.6. Again this is to be expected because this point is closest to the circle centered at C and containing the panel completely.

The relative error, E_{rel}, at a point is defined as,

$$E_{rel} = \left| \frac{v_{exact} - v_{computed}}{v_{exact}} \right| \tag{4.35}$$

Note that v_{exact} is never zero at the points being considered. For a panel of

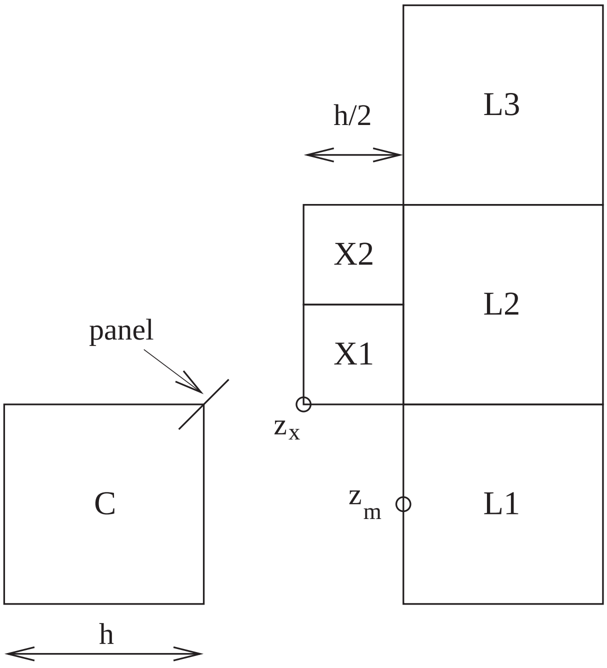

Figure 4.6: Illustration of a panel in a cell and the location of maximum error. The cell C contains a panel at the corner. The cells $X1$ and $X2$ are cells where the X_b special local expansion is performed, $L1, L2, L3$ are cells where local expansions and multipole expansions due to C are computed.

fixed chord length, λ, different cells, C, with length given as $h = \beta\lambda,\ \beta > 1$ are considered. For each of these cases the maximum relative error inside the cells $X1$ and $L1$, which are at the points z_x and z_m respectively, are computed. The number of terms, p, necessary to obtain an accuracy of around 10^{-6} are considered for the computation. At z_m both the multipole expansion due to C and the local expansion due to $L1$ are computed. At z_x the X_b interaction due to C is computed. Figure 4.7 plots these errors as β is varied when using the multipole expansions for the higher order panels developed in section 4.3. At the chosen scale, the local expansion error curve is indistinguishable from the multipole expansion error curve. This occurs because the order of the local expansion error is independent of β as seen in equation (4.30). Therefore, the local expansion is as inaccurate as the multipole expansion that was used to compute its coefficients. From the discussion in section 4.3.1 it is expected that the expansions become increasingly accurate as β increases. As is expected, good accuracy is obtained only when $\beta > 5$. The results are similar for the hybrid cubic/flat panel method (section 4.2) and also for Anderson's method applied to panels (section 4.3.2). The results for

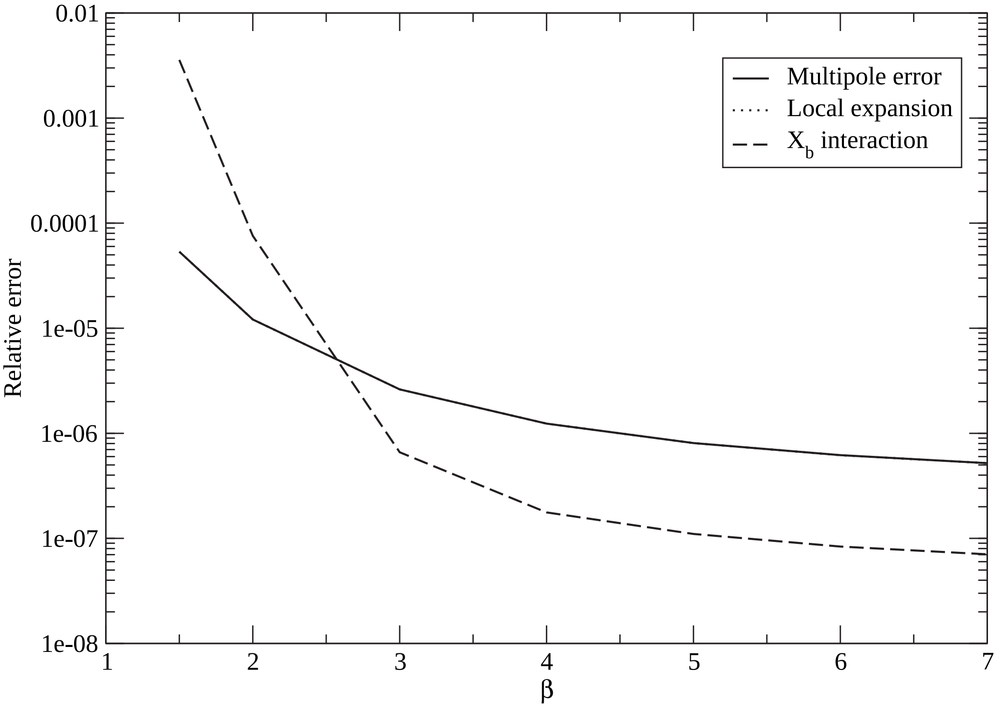

Figure 4.7: Maximum relative error due to the multipole expansion, local expansion and X_b interaction versus change in β for the AFMM for higher order panels. Note that the curves for the multipole expansion and the local expansion coincide.

these cases are presented in Ramachandran *et al.* (In press). This illustrates the importance of limiting the cell size based on panel length.

4.3.4 Numerical results

Comparison of accuracy

In order to demonstrate the accuracy of the AFMM adapted to higher order panels, the flow past a circular cylinder is used for comparison. This problem is chosen because it has an exact solution, the geometry is simple and has bounded values of vorticity. The flow past a circular cylinder of radius 1 unit with 400 cubic panels is considered. The panel strengths are obtained using an LU decomposition. A circular ring of 10000 particles around the cylinder is considered at various distances from the surface of the cylinder. The velocity due to the panels is

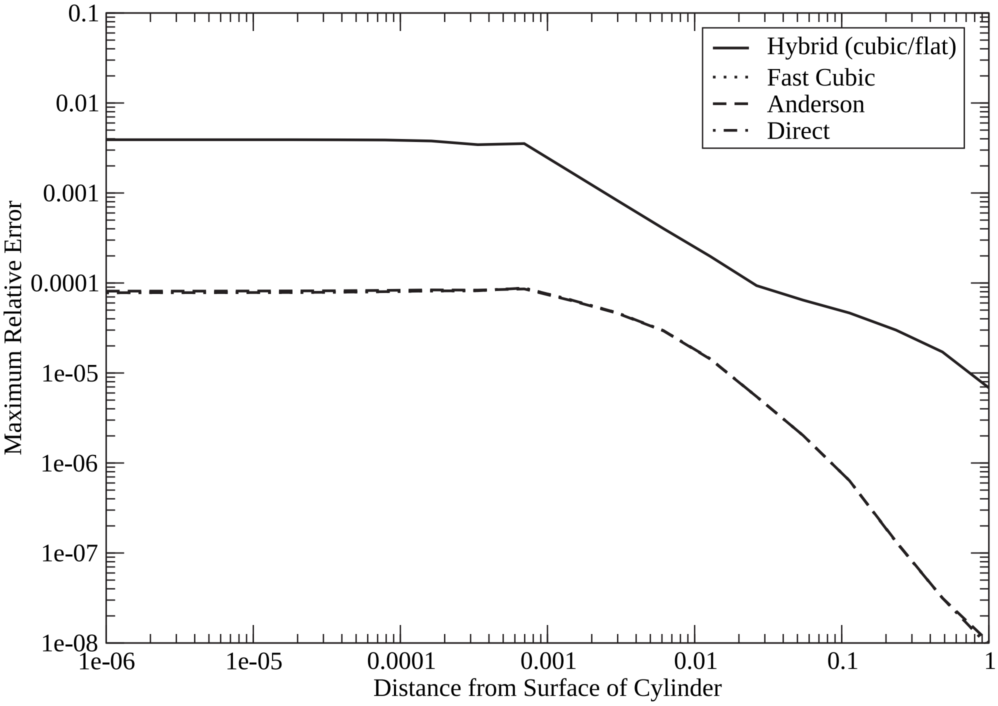

Figure 4.8: Plot of the maximum relative error due to various methods versus distance from the surface of the cylinder. 400 panels are used for the cylinder. The curves for the fast cubic method and Anderson's scheme coincide with the curve for the direct computation.

computed using the direct method (for cubic panels), the hybrid cubic/flat fast multipole panel approach developed in section 4.2, the higher order cubic panel method (section 4.3) and Anderson's scheme (section 4.3.2). The relative error at each evaluation point is computed using equation (4.35). The maximum of these is plotted as the ring radius is varied in Figure 4.8. It is to be noted that when the exact velocity is very small, the point is not considered for the error computation. For the scale used in the graph, the results for the direct method, the AFMM for higher order panels and Anderson's scheme all coincide. This indicates that the higher order panel method and Anderson's scheme produce very accurate results.

Comparison of computational efficiency

A few general observations are made first. If the causes and effects in the quadtree mesh are well separated then the interactions between these causes and effects can be evaluated without direct computations. It is easy to see that this is the most efficient case. For example, consider the flow past a circular cylinder. Let the velocity or potential due to this cylinder be computed at a cluster of particles. If the cluster is at least two diameters away from the center of the cylinder, then there are no direct computations to perform between these particles and the panels that represent the cylinder. Clearly such a computation would be highly efficient. On the other hand if the cluster is such that there are a large number of particles that are distributed along the surface of the cylinder, then there are a large number of direct computations to be performed. Given that the size of a cell is restricted by the length of the panels (section 4.3.3), it is easy to imagine situations where a small number of panels are used along with a large number of particles distributed near them. In such cases, due to the size of the cell, there will be a very large number of effect particles per cell. This will make the multipole computations inefficient. In order to overcome this one should reduce the size of the panels such that the ratio of the number of effects divided by the number of causes in a cell are kept as small as possible. So, on the one hand the number of panels must increase and on the other hand the main reason why higher order panels are chosen is to use a small number of panels. By using a small number of panels the matrix used to solve for the singularity distribution is small and hence easier and faster to solve. Given such a conflicting requirement it is still possible to use a smaller matrix by using two different representations for the body. One representation with larger panels could be used to solve for the singularity strengths and another representation with smaller panels, having interpolated strengths obtained from the larger panels, can be used for the AFMM. Using such an approach, it should be possible solve for the panel singularities efficiently and also perform the AFMM efficiently.

In order to demonstrate the efficiency of the AFMM for higher order panels, the flow past a circular cylinder of unit radius, centered at the origin is considered. The cylinder is discretized into 400 cubic panels. A uniform grid of particles in the square region $z_{min} = -2.0 - 2.0i, z_{max} = 2.0 + 2.0i$ is considered. The number of particles used is varied. The time taken by the direct method, the hybrid flat/cubic method, the higher order panel method and Anderson's method are

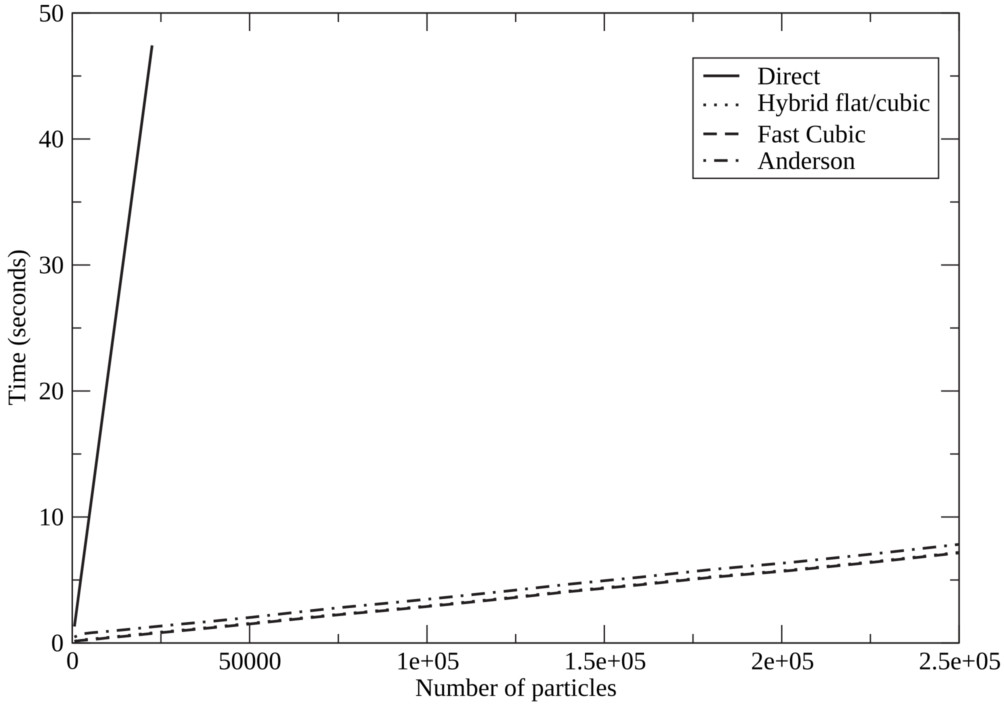

Figure 4.9: Time taken versus number of particles in a square region. 400 panels are used for the cylinder. $\beta = 7.0$. Note that the curves for the fast cubic method (higher order panels) and the hybrid cubic/flat method almost coincide.

plotted in Figure 4.9. The maximum number of cause and effects are chosen as 13 and 13 respectively for Anderson's scheme and 7 and 7 for all the others. β is chosen as 7.0. As can be seen, all the fast multipole methods are significantly (factor of 50) faster than the direct method. It is also seen that the fast cubic method is slightly faster than Anderson's scheme.

It is to be noted that the above computations are not optimized to suit the particle and panel distribution because the number of panels and the grid size is fixed whereas the number of particles is increasing. This results in a very large number of effects per cell. Despite this, the fast multipole schemes are much faster than the direct method.

An estimate for the number of effects per cell can be obtained in the following manner. Given the radius of the circle r, β, and the number of panels, N_{panel}, the length of the smallest cell is clearly limited by $h = 2\pi r\beta/N_{panel}$. If a uniform

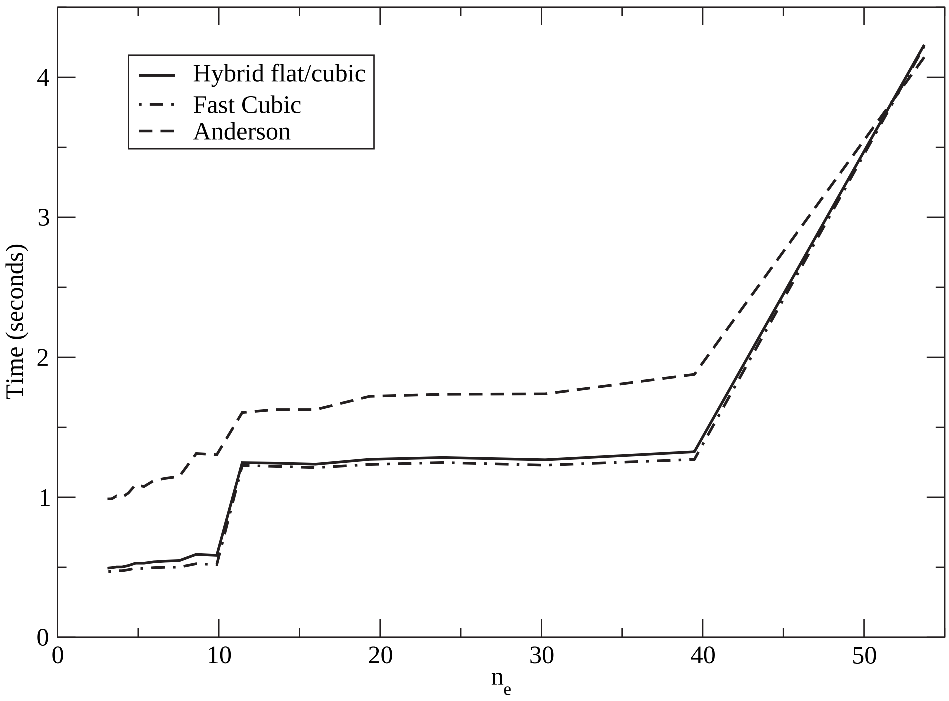

Figure 4.10: Time taken as n_e is varied. n_e is reduced by increasing A. 400 panels are used for the cylinder. $\beta = 7.0$.

grid of N_p particles enclosed in an area A is used, then the estimated number of particles in the smallest cell, n_e, is

$$n_e = \frac{N_p}{A} h^2 = \frac{4 N_p \pi^2 r^2 \beta^2}{A N_{panel}^2} \tag{4.36}$$

In order to demonstrate how significantly the number of effects per cell affects the computational time, the time taken for the presently developed algorithm for a fixed number of particles is computed as n_e is changed. For a given N_p, β and a given body, n_e can be varied by changing either A (the area of the uniform grid of particles) or by changing N_{panel}.

Figure 4.10 plots the variation of the time taken as the area of the uniform grid is gradually increased. The number of tracer particles in the grid is 40000. The stair stepping occurs in the plot because the cell size is limited by $\beta\lambda$ and the way in which the cells are split. If the total length of the side of a cell at level 0 is L, then the length of the side of the cell at level l is, $L/2^l$. In the range

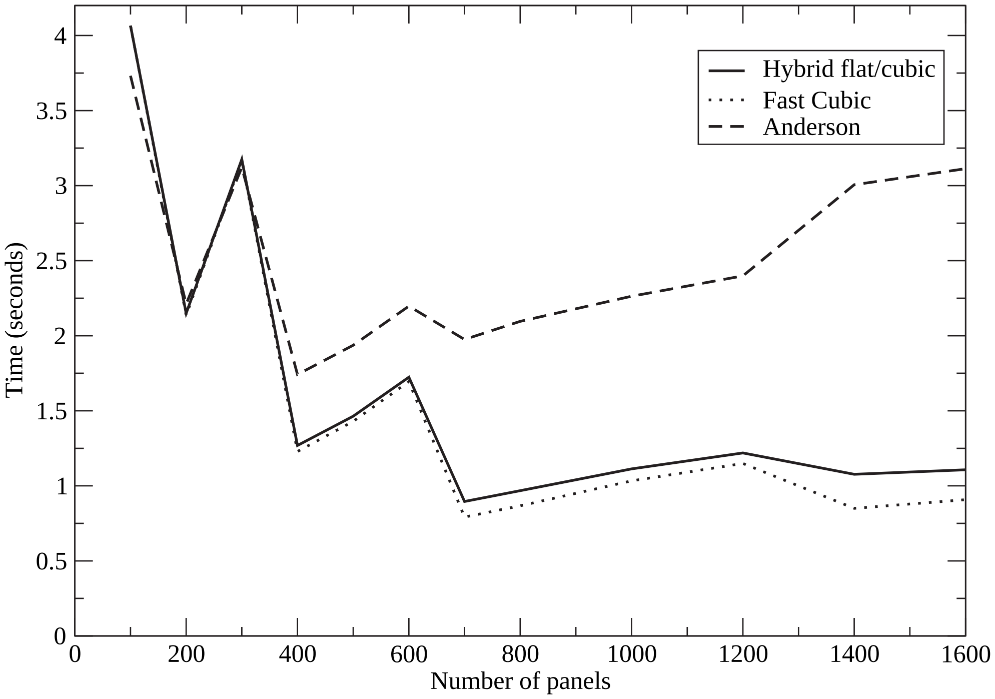

Figure 4.11: Time taken as the number of panels is increased. 40000 points in a uniform grid are considered inside a square region. $\beta = 7.0$.

$9 < n_e < 11$, the length L of the level 0 cell and the panel length are such that it is possible to refine the grid by one more level. This is the reason for the stair stepping in the plot. However, the point being made here is that as the number of effects per smallest cell reduces, the time taken reduces significantly, anywhere between a factor of 4 to 8. It is also seen that for most part, the fast cubic panel method is about 1.5 to 2 times faster than Anderson's scheme.

Figure 4.11 plots the variation of the time taken as the number of panels is increased. 40000 tracer particles are considered inside the square region ($z_{min} = -2.0 - 2.0i, z_{max} = 2.0 + 2.0i$) for all the computations. As can be seen from equation (4.36), n_e drops as the number of panels increases. The time taken drops first and then increases as the number of panels is increased because initially the ratio of the length of the cell and the panel size is such that the first increase in the number of panels triggers a split in the cell. Subsequently, reducing the panel length does not trigger a split until a threshold is crossed. During this the computational time increases. As can be seen in the figure, even though the number of panels has increased by a factor of 16 the time taken to perform the computation has reduced by a factor of 4. This illustrates the importance of choosing the right number of panels.

The time taken to re-compute the multipole coefficients (equation (4.14) and (4.31)) for each panel is small compared to the other computations. In the Figure 4.11 the fraction of time taken to recompute the multipole is around 8% of the total time when the number of panels are 1600. With 400 panels the time taken to re-compute the multipoles is around 1.5% of the total time. This indicates that the higher order panel method is efficient even if the panel geometry changes significantly in time.

It is to be noted that Anderson's scheme has been implemented with some care. The only difference between the newly developed algorithm and the implementation of Anderson's method is in the functions governing the computation of the multipoles at the finest level, transfer of multipole expansion and evaluation and transfer of the local expansion. The expressions given in Anderson's work are used in these functions. The velocity field is computed by analytically differentiating these expressions. All quantities that are constant (like $\log(a)$, where a is the radius of the ring and the various sines and cosines) are pre-computed and stored in order to avoid unnecessary re-computation. It is also to be noted that the evaluation of the stream function did not amount to more than 5% of the total computational time taken to evaluate the velocity on a 101x101 grid of points. It is possible that the implementation of Anderson's method can be further optimized. However, it must be noted that the expressions in equations (4.32) and (4.33) require the evaluation of the arctangent (to compute the angle θ) and the various cosines. By precomputing e^{iMs_i} and e^{is_i}, it is possible to obtain the other cosine terms using arithmetic operations and the value of $e^{i\theta}$ and $e^{iM\theta}$. However, the multipole expansions derived in the present work require purely arithmetic operations. Therefore, it appears that there is a small price to pay for the generality that Anderson's scheme provides and the ease of its deployment and use in different situations. The gain in using the expressions in section 4.3.1 over Anderson's scheme depends on the choice of parameters and the problem chosen. The present implementation suggests that a factor of two improvement is possible.

4.4 Summary

The following were discussed in considerable detail in this chapter.

- An adaptation of the AFMM suitable for vortex panels with a linear geometry (Ramachandran *et al.*, 2003) and linear vorticity distribution.
- An elegant generalization of the AFMM to handle passive particles.
- An adaptation of the AFMM to handle higher order vortex panels. This was specifically demonstrated for cubic panels with a linear vorticity distribution (Ramachandran *et al.*, In press).
- The use of Anderson's "FMM without multipoles" method to obtain the velocity field due to higher order vortex panels. The modifications necessary to use it in the context of the AFMM were discussed.
- The importance of limiting the cell size based on the length of the panels inside it was demonstrated.
- The accuracy and efficiency of the methods developed were demonstrated.

The developed techniques are centrally important to vortex method based solvers because they enable one to use a large number of interacting particles. The next chapter investigates other important fast algorithms used in this work.

CHAPTER 5

OTHER FAST ALGORITHMS

In the previous chapter, the adaptive fast multipole method (AFMM) was applied to accelerate the computation of the velocity field due to the vortex panels. The AFMM is central to a vortex method because it enables the rapid evaluation of the velocity field. However, there are aspects of a vortex method that require the use of other fast algorithms. The following fast algorithms that are of relevance to the present work are discussed in this chapter.

- An algorithm to move vortex particles in the vicinity of complex geometries.
- An algorithm to compute the velocity induced by the sheets on each other and on the boundaries.
- An algorithm for the inter-conversion of sheets and blobs.
- An algorithm to merge and annihilate vortex particles in the context of the random vortex method.

Of these, the algorithm to move particles in the vicinity of complex geometries is the most challenging. The other algorithms can be developed easily using the techniques developed for it. Hence, the algorithm for moving particles efficiently in the presence of complex geometries is first discussed.

5.1 Particle motion and complex geometry

A moving particle should not penetrate a solid body. Therefore, care is to be taken during numerical simulation to ensure that no particle penetrates a solid wall. This needs to be done efficiently. The problem is further compounded when the geometry is complex.

In the RVM, particle motion occurs in two different ways. The first is due to the convection of the particles. The second is due to the random displacements given to the particles during diffusion. If the time step is small, the displacements due to convection will be small. The no-penetration condition enforced on the velocity field will in general prevent the particles from entering the surface. However, if the time steps are large, the situation can be different. On the other hand, the displacement of the particles during diffusion is random. It is therefore not easy to ensure that the particles do not penetrate solid bodies. Detecting collisions of particles with solid walls for simple geometries is trivial and usually not an issue. However, it is not easy to handle complex geometries efficiently.

Given the possibility that a vortex particle path intersects a solid wall, the case can be handled using the following two approaches. In one approach the particle is reflected specularly. In the other, the particle is removed from the computation as done by Clarke and Tutty (1994); Smith and Stansby (1989) and others. Ghoniem *et al.* (1982) use a hybrid approach and either reflect or absorb particles depending on their initial position with respect to the wall. In the present work the particles are reflected specularly for both convective and diffusive displacements.

In the following, an efficient methodology is developed to move particles in the presence of arbitrary two-dimensional shapes. Certain ideas from the fast multipole algorithms are used in order to make the computations more efficient. This is based on the work of Ramachandran *et al.* (2000*b*, 2001, Under review).

In the context of collision detection of particles, Lubachevsky (1991) presents an event driven algorithm to efficiently handle the interactions of hard spheres. This algorithm divides the domain of the particles into sectors to speed up the computations. The algorithm is applicable to a more general problem domain. However, the algorithm developed in the present work is simpler and designed to be used in the context of vortex methods. The simplicity arises from the fact that unlike the hard sphere problem, there are no sphere-sphere collisions to consider.

It is to be noted that the algorithm developed here is also of use in the context of deterministic diffusion schemes in the presence of complex geometry. While deterministic diffusion schemes do not involve a random displacement of the particle, they do require the computation of a list of nearby particles with which the particle will interact. That is, particles that are separated by a solid wall cannot diffuse vorticity to each other. It is clear that the ideas developed in the present case can be used for this.

The basic idea used in the algorithm is to discretize the geometry into linear segments. The domain containing the particles and solid boundaries is then decomposed into a quad-tree of cells, much like the computational cells of the AFMM discussed in section A.2.1. The particles are then tracked along the cells as they move. If the particle strikes a body, it is reflected appropriately. Sheet-blob conversion issues are also considered. The algorithm is designed so that it is possible to extend an existing AFMM implementation. A simple measure of the geometric complexity of a shape is also provided. The algorithm developed is shown to scale efficiently as this complexity increases.

In order to perform intersection checks of particle paths with the solid walls efficiently, it is imperative that the domain decomposition be done carefully. This is detailed in the following.

5.1.1 Domain decomposition and cell generation

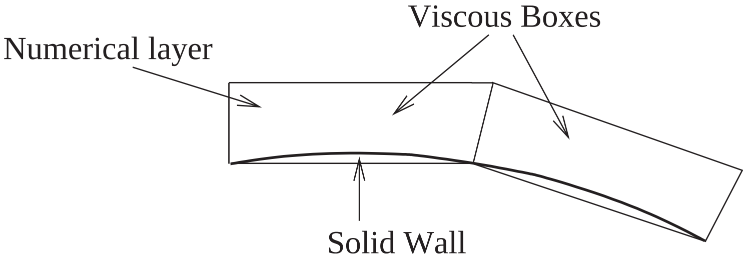

Figure 5.1: Discretization of the numerical layer into viscous boxes.

The geometry is first discretized into segments or panels. Each panel has extent and has a numerical layer associated with it. As seen earlier, vorticity in this numerical layer around the body is represented in the form of vortex sheets. Blobs that enter this region are converted to sheets and sheets leaving the region are converted to blobs. In order to handle this correctly, the numerical layer is discretized into *viscous boxes*. As illustrated in Fig. 5.1, each viscous box is essentially a trapezium with one side along the solid surface and the other at the

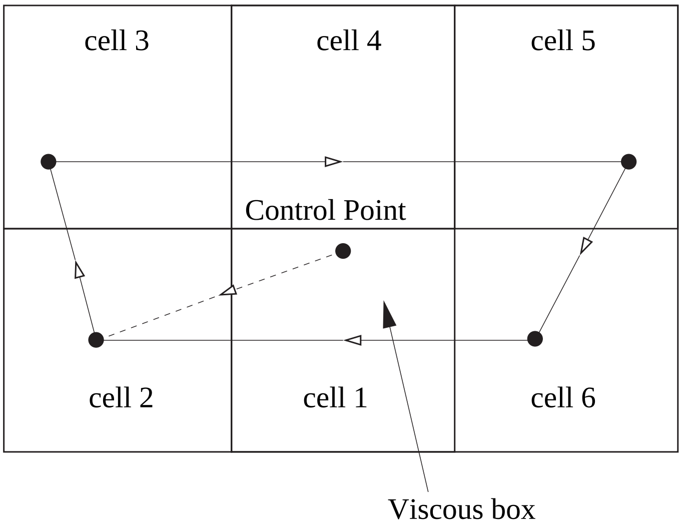

Figure 5.2: A viscous box passing through various cells.

edge of the numerical layer. Once the body geometry is specified, the geometry of the viscous boxes are known. The domain of particles is organized into a quad-tree as discussed in section A.2.1 and 4.2.2. The viscous boxes are treated as causes and all moving particles as effects. The domain is first decomposed in terms of the panel centers or some representative point, called the control point, inside the viscous box. Thus, each viscous box is initially identified with one childless cell. Due to the extent of the viscous box, parts of it will also be in other cells. In Fig. 5.2 the viscous box is initially assigned to cell 1 alone. All its neighboring cells also contain parts of the viscous box and this information is to be updated in those cells. A cell is said to contain a viscous box if a side of the box passes through it. All the childless cells are modified based on this.

The following procedure is applied to modify the childless cells that are created by the tree generation algorithm. A viscous box is considered. The control point of the viscous box is within some cell created by the domain decomposition. Starting at the control point, a path is traced to an edge of the box. The path continues along lines constituting the trapezium. In this process, the different childless cells that are traversed are modified to reflect the existence of a part of this viscous box in them. Care is to be taken to avoid repetitions when the cell already contains the particular box. The procedure is illustrated in Fig. 5.2, where the viscous box passes through the cells numbered 1 through 6. After performing the domain decomposition only cell 1 will contain the particular viscous box. After performing the above procedure, cells 2, 3, 4, 5 and 6 are also made to contain part of the viscous box. The process is repeated for all the viscous boxes. This algorithm will fail if a cell is completely enclosed by the viscous box because in that case no side of the trapezium will pass through the cell. This can be easily prevented by enforcing the condition that every cell has a side larger than the smallest side of all the trapeziums.

The pseudo-code for tracking a side of the viscous box is given in algorithm 5.1. z_1 and z_2 are the end points of the line (side of the viscous box) that are being tracked. The most vital component in this algorithm is the one that tracks the side of the trapezium through the various cells using the **FindNextCell** function. It is evident that this tracking algorithm is also important for the case of the random walk of particles. The details of the algorithm for tracking a line segment are elucidated first.

Algorithm 5.1 TrackLine(box, z_1, z_2, C)

```
$NextCell$ = FindNextCell ($z_1, z_2, C$)
if $NextCell \neq C$ then
  if $NextCell$ does not contain $box$ then
    Set $box$ in $NextCell$.
  end if
  TrackLine($box, z_1, z_2, NextCell$)
end if
```

The current discussion is restricted to two dimensions and hence the complex plane is used to describe the algorithm. Let the start and end points of the straight line being tracked be labelled z_1 and z_2 respectively, where z is the complex coordinate. Let z_{12} be the line joining z_1 and z_2. Assume that the tracking is started at a childless cell C which contains the point z_1.

If the point z_2 lies within the current cell C, then the line segment z_{12} is wholly in the cell C. If it does not, then it crosses over into one of the colleagues of C, intersecting the side of the cell at z_{tmp}. This side identifies the colleague C_1 as shown in Fig. 5.3. Then, the childless cell C_{tmp} that contains z_{tmp} is found. This will be either C_1 or one of C_1's descendents or one of its ancestors. In Fig. 5.3, C_{tmp} is one of C_1's descendents. In order to continue tracking from the cell C_{tmp},

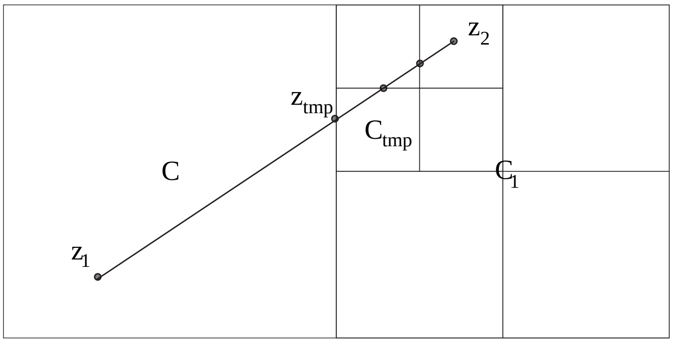

Figure 5.3: Schematic of a line passing through various cells. The line joining z_1 and z_2 originates in the cell C and passes through the descendents of its colleague C_1.

the process is repeated with $C = C_{tmp}$ and $z_1 = z_{tmp}$. In this fashion, the cells through which z_{12} passes can be found. The pseudo-code to find the next cell (**FindNextCell**) is given in algorithm 5.2.

Algorithm 5.2 FindNextCell(z_1, z_2, C)

```
if Line $z_{12}$ crosses any side of $C$ then
    Find the side, $S$, of $C$, that the line $z_{12}$ crosses.
    Find intersection of $S$ and $z_{12}$ and store as $z_{tmp}$.
    Find the colleague, $C_1$, of $C$, that shares $S$.
    Find the childless ancestor or descendent $C_{tmp}$, of $C_1$, that contains $z_{tmp}$.
    $z_1 = z_{tmp}$
    return $C_{tmp}$.
else
    return $C$
end if
```

Using implementations of the above algorithms it is easy to find all the cells that contain parts of a viscous box. After applying the algorithms to all the viscous boxes, it is possible to check for intersections of particle paths with the viscous boxes in each cell. The details of this methodology as applied to the RVM are discussed next.

5.1.2 Diffusion using the RVM

Diffusion of blobs and sheets in the RVM is accomplished by giving them random displacements. The sheets are given displacements perpendicular to the local panel surface and the blobs are given random displacements in each co-ordinate direction

(x and y in two dimensions). The paths of the diffusing blobs and sheets are to be checked for intersections with any panel. Clearly, the displacement of the blob or sheet at a given time step is a straight line. If a blob enters a viscous box it is to be converted to a sheet and vice versa. A variant of the algorithms 5.1 and 5.2 are used here and described below. The following observations are made before the algorithm is presented. Every blob is initially associated with a childless cell of the mesh. It is also to be noted that one side of each viscous box is associated with a linear panel that represents the solid surface. The other sides of the box are used to represent the numerical layer.

A blob in a childless cell, C, is considered. Its initial position is z_1 and the final position is z_2. The line joining the two points, z_{12}, is tracked through all the childless cells in a manner similar to the algorithm 5.1. This is also illustrated in Fig. 5.3. First, it is determined if the point z_2 lies within the current cell C. If it does then the blob is checked for collisions with panels inside the current cell. If the point z_2 is not inside the cell C, then the line segment z_{12} crosses the cell. Therefore, the segment of z_{12} inside the current cell is found using z_{tmp} obtained from algorithm 5.2. This segment is checked for intersections with all panels in the current cell. If there is no intersection then the algorithm proceeds from the next cell. If there is an intersection, the reflected path is computed. The same algorithm is then applied to the reflected path. The pseudo-code embodying the above discussion is given in algorithm 5.3.

The final value of z_2 is the final position of the particle. The methodology to check for intersections between the particle path and the panels is discussed next. For the case illustrated in Fig. 5.4, the panel associated with the viscous box and the relevant points are transformed to a local co-ordinate system. If the line joining z_1 and z_2 is the path of the particle, it is evident that it is not always necessary to compute the value of x_{int}, the point where the line intersects the x axis. One can eliminate many intersection checks using the relative signs of y_1 and y_2. Therefore, in the present implementation, the first check is to see if an intersection is possible. If it is, then the value of x_{int} is computed and compared to see if $0 \leq x_{int} \leq l$ where l is the length of the panel. In some cases there

Algorithm 5.3 CheckPath(z_1, z_2, C)

```
if Line $z_{12}$ crosses any side of $C$ then
    Find the side, $S$, of $C$, that the line $z_{12}$ crosses.
    Find intersection of $S$ and $z_{12}$ and store as $z_{tmp}$.
    $Last = false$
else
    $z_{tmp} = z_2$
    $Last = true$
end if
if Line joining $z_1$ and $z_{tmp}$ intersects any panel in $C$ then
    Find the panel that is intersected first.
    Find reflected ray and store the path in new $z_1, z_2$.
    CheckPath($z1, z2, C$)
else if $Last == false$ then
    Find the colleague, $C_1$, of $C$, that shares $S$.
    Find childless descendent or ancestor $C_{tmp}$, of $C_1$, that contains $z_{tmp}$.
    $z_1 = z_{tmp}$
    CheckPath($z1, z2, C_{tmp}$)
end if
```

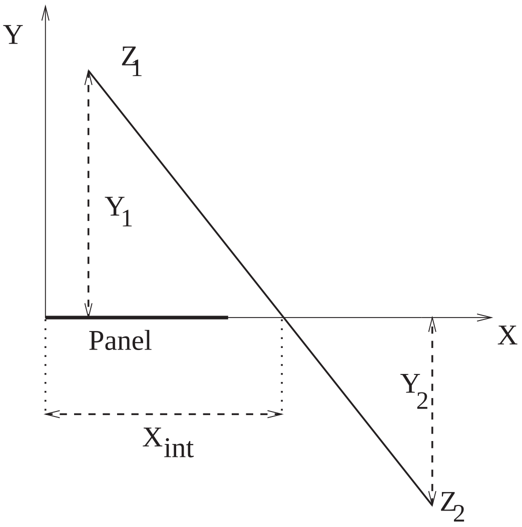

Figure 5.4: Illustration for algorithm used to check intersections of a particle path with a panel.

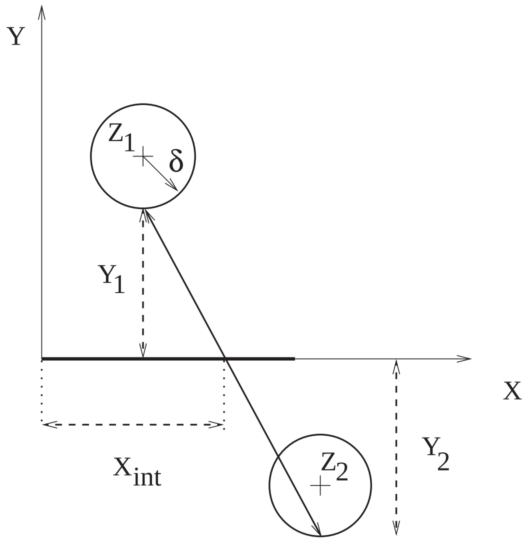

Figure 5.5: Illustration for algorithm used to check intersections of blobs having a finite core radius, δ, with a panel.

can be more than one panel in the same cell that the line crosses. In such a case all the intersecting panels are considered and the panel that has the closest intersection point is chosen for the reflection. The corresponding distance is given by, $\sqrt{(x_{int} - x_1)^2 + y_1^2}$. The intersection point is given as $(x_{int}, 0)$ and the final point is reflected such that $z_2 = (x_2, -y_2)$. It is possible to use a different reflection scheme or even delete the intersected blob, as done by Clarke and Tutty (1994) and Smith and Stansby (1989).

The same algorithm can be used for the sheets. However it can be simplified since the displacement of the sheets is only perpendicular to the local surface. The terminal position of the particle determines whether it is a sheet or a blob. If sheets are not used at all as in (Clarke and Tutty, 1994; Lin *et al.*, 1997; Taylor and Vezza, 1999*b*,*a*) and others, then the complications due to the conversion are avoided.

Usually, vortex blobs have a finite core and it may be necessary to ensure that the blob core does not penetrate a panel. In such a case the intersection check must be done by using the points closest to the panel and not the center of the blob. In local co-ordinates this reduces to the bottom of the blob as shown in the Fig. 5.5. This can be easily incorporated in the present algorithm by doing the following. In Fig. 5.5, instead of considering the actual heights in the local co-ordinate system, the radius of the blob, δ, is subtracted from the heights and the resulting line is used to perform the check. This merely requires two extra subtractions. However, a blob with a core does require little more care than this. Since the blob has extent it is not sufficient to check for intersections inside the current cell alone. One must also check for intersections in all cells that the blob will pass through by virtue of its extent. This is not difficult to do but does require a little effort if efficiency is desired. It also increases the computational time taken by the algorithm since the number of cells to consider is larger.

One final point to note is that there are cases where the angle between two adjacent panels is very small and forms a concave region. If a particle having a finite core-radius performs a random walk into this region, it is possible it becomes stuck between the two panels as it approaches the intersection of the two panels. To avoid this case one must store the length of the path between two consecutive reflections. If the length is reducing and tending to zero the particle could be placed at the corner without undergoing any further reflections. Alternatively, the particle may be reflected along the bisector of the angle between the two panels.

In this fashion, the collisions of the moving particles with the solid boundary are handled. Using the above ideas, it is clear that vortex diffusion in the presence of arbitrary boundaries can be carried out when using the RVM.

5.1.3 Numerical results

In order to demonstrate the efficiency of the algorithm developed, a complex body geometry is considered. Particles are distributed on the surface of the body and diffused using random walks. The computational time taken by the algorithm for a fixed number of time steps is plotted as various parameters are varied. In order to characterize the geometric complexity of the body, a measure for the geometric complexity is necessary. Using a fractal dimension in this context is not useful because geometries considered in such computations are not truly fractal and can at best be finite iterations of a fractal construction. Using a complexity measure from information theory (e.g. Kolmogorov complexity) also does not seem appropriate for the present study. The following properties for the geometric complexity are desirable:

1. The measure must be purely geometric and must not depend on the number of panels used to discretize the body.
2. It must be a scalar and insensitive to rotation, translation and scaling.
3. If the number of bodies is multiplied by an integer k, then the measure must also scale by k.
4. The measure must be easy to compute.

A simple measure for the geometric complexity, $\mathcal{C}$, of a body that satisfies all of the above can be defined as follows. Let $\mathcal{G}$ be the sum of the absolute change in the angle of inclination of the tangent as the contour of the body is traced. Consider an equilateral triangle, if one starts at a vertex and traces the contour of the body, it is clear that at each vertex, the inclination of the tangent changes by $2\pi/3$ radians. There are three such vertices and therefore for an equilateral triangle, $\mathcal{G} = 2\pi$. In similar fashion it can be seen that for simple geometries like triangles, rectangles, closed convex polygons and circles, $\mathcal{G} = 2\pi$. Hence, the geometric complexity is defined in terms of $\mathcal{G}$ as $\mathcal{C} = \mathcal{G}/2\pi$

Evidently, this measure is incapable of differentiating between a circle and a convex polygon. However, for a closed body that has concave depressions and convex projections, it does show an increase in complexity. Any concave geometry or projection complicates the intersection algorithm because such regions would increase the number of intersections that a randomly diffusing particle would make with the body.

For the present study, in order to construct geometries with varying complexities easily, a fractal construction is chosen. At level 0 of the construction, an equilateral triangle is considered. Each side of the triangle is split in the manner of a Koch snowflake (see Peitgen *et al.* (1992) for details). The first and second levels of construction are shown in Figure 5.6. One can easily compute the complexity of such a body at any level of construction. Table 5.1 shows the variation

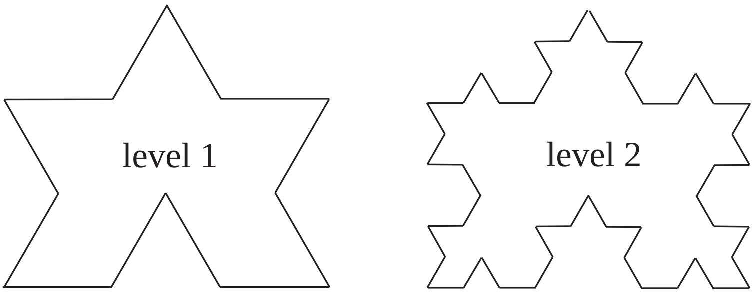

Figure 5.6: Geometry chosen for study at level 1 and 2.

Table 5.1: Geometric complexity of the body.

Construction level	Number of sides	Geometric complexity, $\mathcal{C}$
0	3	1
1	12	3
2	48	11
3	192	43
4	768	171

of complexity as the number of levels of construction of the body shown in Fig. 5.6 increases. As is evident, the complexity of the body increases rapidly.

In order to test the present scheme, point vortices are distributed on the inner or outer surface of the body. The particles are diffused and the computational time taken is plotted as different parameters are varied. Since the primary interest is in testing the random walk algorithm alone, no convection is performed. Fig. 5.7 and 5.8 show the particles and the body used after 50 time steps. In Fig. 5.7 the particles are placed on the outer surface of the body and diffused. In Fig. 5.8 the particles are distributed on the inner surface of the body. 49152 particles are used and the body has 576 equal sized panels. The body chosen is at level 3 and has a geometric complexity of 43. As is evident, none of the particles cross the body. The maximum number of *cause* and *effect* particles allowed per cell is fixed for all simulations as 10. The non-dimensional length of a side of the equilateral triangle that is used to generate the body (i.e. the body at level 0) is 1. For simplicity, no sheet-blob conversion is performed. In order to stress-test the algorithm the viscosity, ν, and the time step Δt are chosen such that large random displacements are generated. In the present work ν and Δt are chosen such that the standard deviation of the random displacement is $\sigma/L = \sqrt{2\nu\Delta t}/L = 0.1$, where L is the

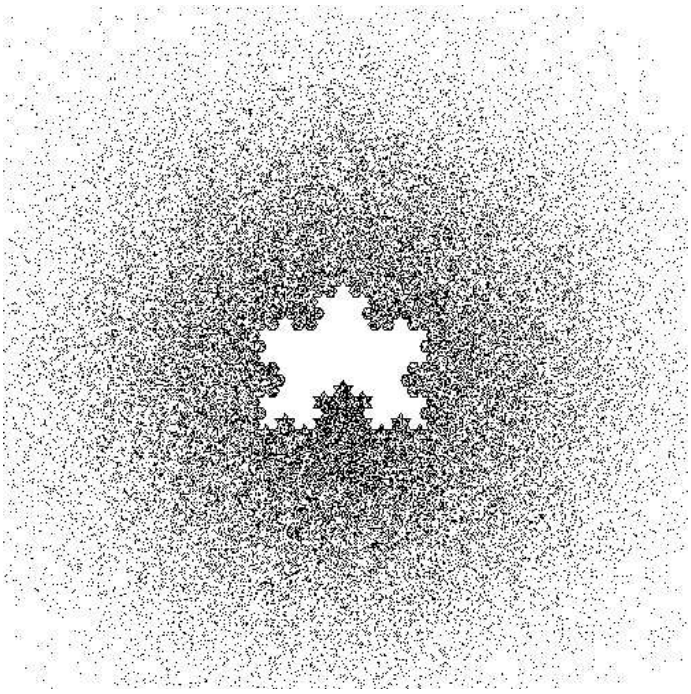

Figure 5.7: Simulation of diffusion outside the body at level 3 of its construction. 49152 vortex blobs are used in the simulation. The blobs are initially distributed on the outer surface of the body. The non-dimensional width of the body is 1. The figure is a plot at the end of 50 time steps.

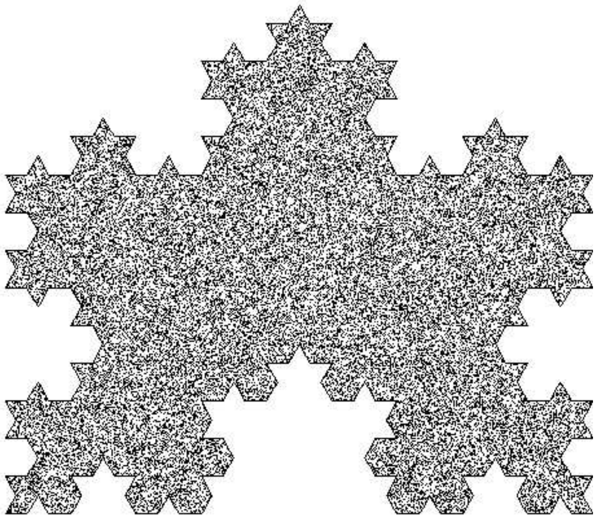

Figure 5.8: Simulation of diffusion inside the body at level 3 of its construction. 49152 vortex blobs are used in the simulation. The blobs are initially distributed on the inner surface of the body. The non-dimensional width of the body is 1. The figure is a plot at the end of 50 time steps.

length of the side of body. This is 10% of the size of the body and is quite a large displacement. The computations in (Koumoutsakos and Leonard, 1995) for the flow past a cylinder at a Reynolds number of 40 use a Δt of 0.02. Simulating this using the RVM would therefore require the generation of random displacements having a σ/L of about 0.032. The present displacement is about 3 times larger and therefore seems a reasonable choice to use when testing the diffusion algorithm.

To determine the efficiency of the algorithm, the computational time is plotted against the complexity of the body in Fig. 5.9. For each body of given complexity, the number of panels used is 768. The panels are equally sized. 49152 vortex particles are used. The complexity of the body is varied from 1 to 171. All other parameters are held fixed. The dashed line is for the internal diffusion case and the solid line is for the external diffusion case. It is clear that despite the great increase in complexity, there is only a 15% increase in the computational time. Due to a larger number of reflections, the internal diffusion case takes more computational time.

Fig. 5.10 plots the variation of the computational time as the number of panels is changed for a given number of particles (49152) and complexity (43). Here again, despite a seven fold increase in number of panels there is only a 25% change in the time taken. As the number of panels increases, the time taken increases almost linearly. This indicates that the algorithm is efficient. Fig. 5.11 plots the variation of the computational time versus number of blobs used in the simulation. As expected this is linear. 576 equal sized panels are used for the body and the complexity of the body chosen is 43.

For a simulation of diffusion inside the body at level 3 (complexity 43) with 768 panels and 49152 particles the maximum number of cause and effect particles is chosen as 1000. This ensures that there is only one cell in the computation and hence all the particles are checked for intersections with all the panels. It is seen that this computation takes more than 45 times the time taken when the maximum number of causes and effects is set to 10. This clearly demonstrates the efficiency of the new procedure as compared to naive intersection checks without any adaptive domain decomposition.

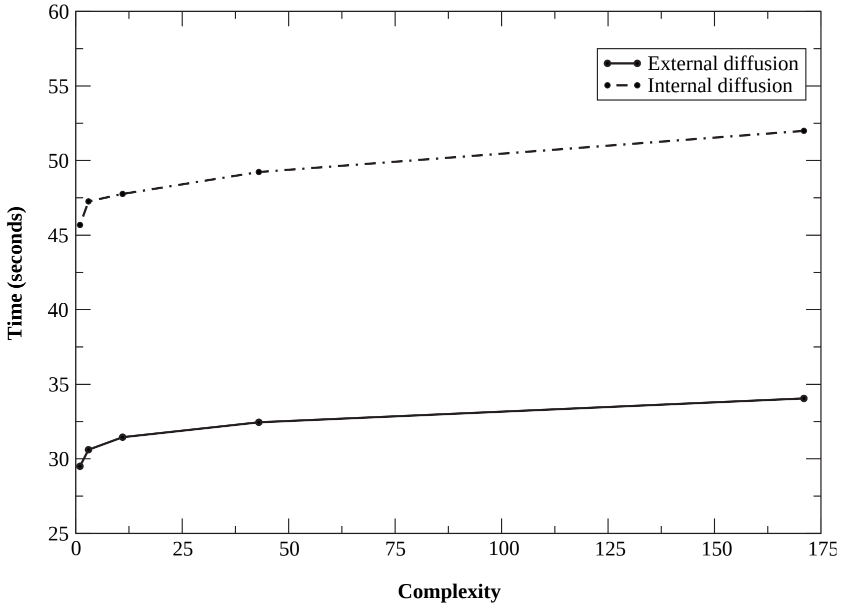

Figure 5.9: Variation of time taken for 50 time steps versus geometric complexity of the body. 49152 vortex blobs are used in the simulation and 768 equal sized panels are used for the body.

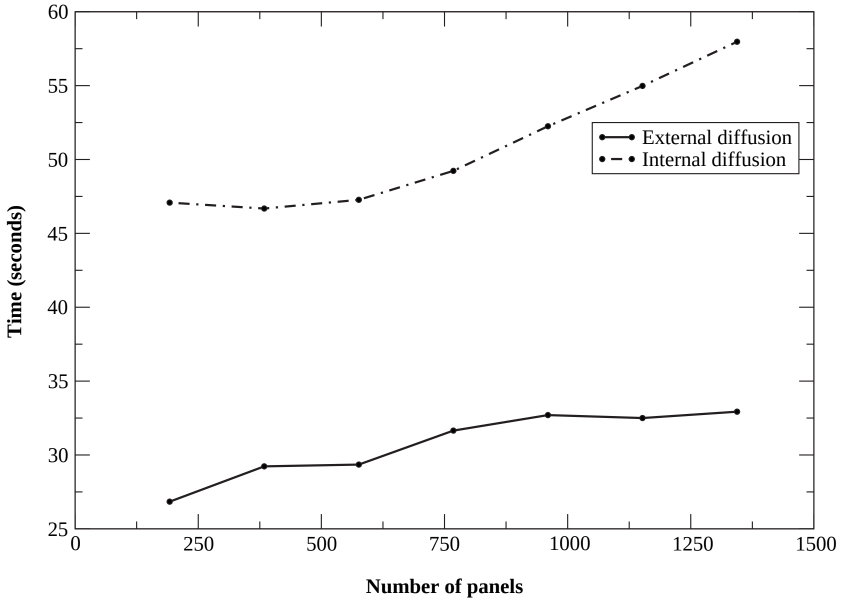

Figure 5.10: Variation of time taken for 50 time steps versus number of panels used. 49152 vortex blobs are used in the simulation and the complexity of the body is 43 (i.e. the body is at level 3 of the fractal construction).

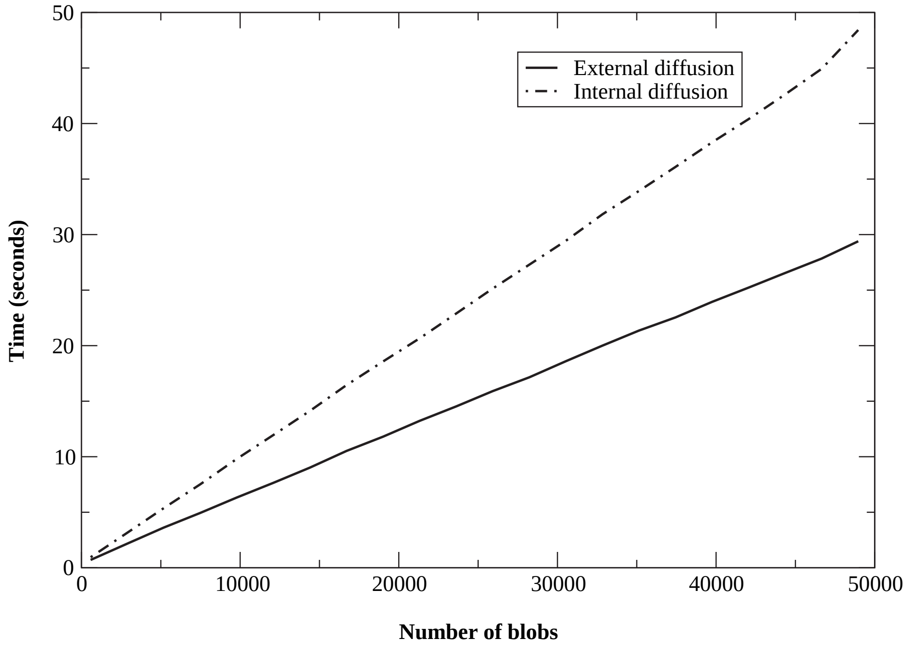

Figure 5.11: Variation of time taken for 50 time steps versus number of blobs used in the simulation. 576 equal sized panels are used in the simulation and the complexity of the body is 43 (i.e. the body is at level 3 of the fractal construction).

In order to demonstrate the speed of the algorithm in realistic situations the uniform flow past two different complex body shapes is simulated. There are two reasons why the simulations are made. The first is to demonstrate that the algorithm works in a realistic situation with advection and sheet blob conversions even when complex shapes are considered. The second is to provide an estimate of the efficiency of the algorithm as compared to the rest of the computations (like the fast multipole velocity evaluation) involved in the simulation. For the simulation, the adaptive fast multipole method (Carrier *et al.*, 1988), using the modified domain decomposition of section 4.2.2 has been implemented to compute the velocity due to the blobs on each other. A linear vortex panel method is used to satisfy the no penetration condition on the boundary. The fast multipole technique (Ramachandran *et al.*, 2003) developed in section 4.2 is also used to evaluate the velocity field due to the panels. In the first case, the body has a geometry as shown in Fig. 5.12. 434 equal sized panels are used to discretize the shape. For the simulation, the relevant parameters are chosen as $Re = 1000$ and $\Delta T = 0.0044$, where $T = Ut/R$, U is the free stream velocity and R is the radius of the cylinder. The numerical layer height is taken as $\epsilon/R = 0.011$. The

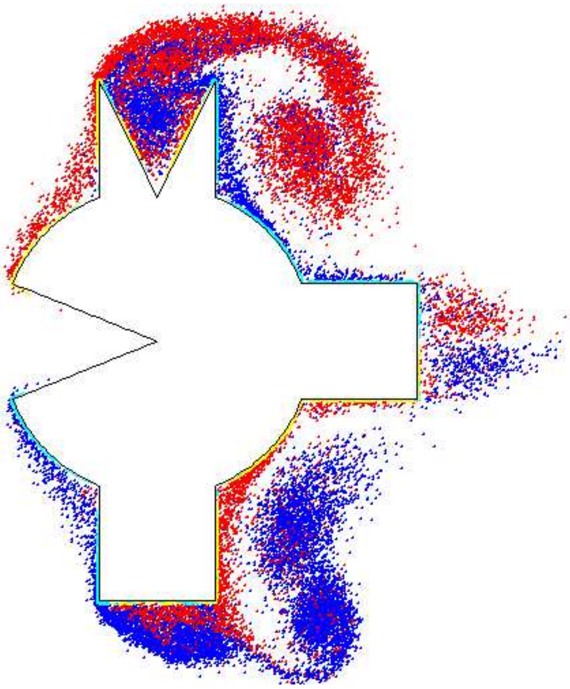

Figure 5.12: Vorticity distribution for flow past the body at $T = 1.32$.

resulting vorticity distribution at the end of 301 time steps is shown in Fig. 5.12. There are about 25000 particles in the flow at this time. The red colored blobs and yellow colored sheets represent clockwise vorticity. The blue blobs and cyan sheets represent anti-clockwise vorticity. The strength of each blob is 7.868×10^{-4} units. As is evident, none of the particles have entered the body, indicating a successful implementation of the diffusion algorithm. At the end of 301 time steps, (i.e. a total time of $T = 1.32$), the computational time taken by the diffusion algorithm is about 3.44% of the the total time taken by the fast multipole method. This corresponds to 2.5% of the time taken for the entire simulation.

In Fig. 5.13 the vorticity distribution for the flow past the shape given in Fig. 5.6 at level 2 and having complexity of 11 is shown. 576 equal sized panels are used to discretize the body. The flow is at an angle of 45° to the horizontal. The Reynolds number is based on the width of the body. The flow parameters are chosen such that the Reynolds number is 1000. The figure shows the vorticity distribution at the end of 1000 time steps using a $\Delta T = 0.0025$. At this time there are about 45000 particles in the flow. It is evident that none of the particles have entered the body. For this simulation the time taken for the diffusion algorithm is about 2.1% of the time taken for the fast multipole algorithm and only 1.6% of

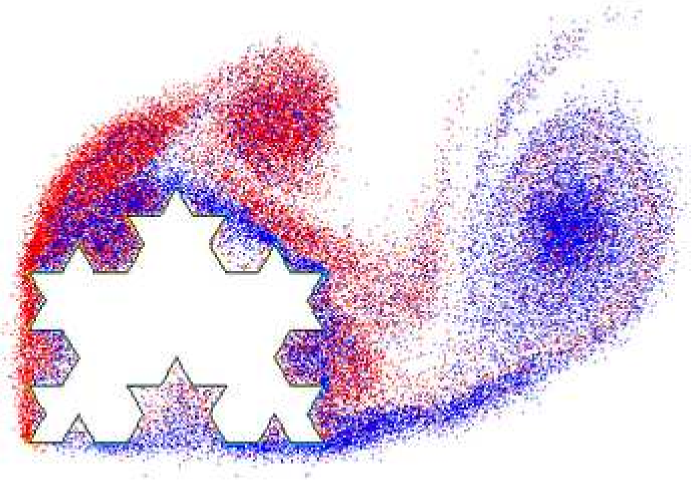

Figure 5.13: Vorticity distribution for flow past a complex shape at $T = 2.5$.

the total simulation time.

Considering the fact that the shapes are arbitrary, the above times for the two different simulations are certainly acceptable. From the earlier discussions, it is evident that an increase in the number of panels or geometric complexity will not change the computational time of the diffusion significantly. It is also evident that once a wake structure is created, the diffusion algorithm will perform more efficiently. This is because a greater fraction of the particles are in the wake and these particles require less intersection checks due to their distance from the body.

Thus, an efficient algorithm to move particles in the vicinity of complex geometries has been developed. The algorithm handles random displacements of particles. It is therefore easy to see that it is also capable of handling any arbitrary displacement of the particles. Therefore, the same algorithm is used to handle the convective displacement of the particles. This enables for the use of larger time steps (if necessary) during convection.

5.2 Computing the sheet velocity field

The velocity induced by a vortex sheet is highly local and influences only particles inside the numerical layer that are below the sheet. It is therefore inefficient to compute the velocity field of the sheets in a naive manner. In the present work a simple method is used to efficiently evaluate the sheet velocities.

The algorithm first generates a hierarchy of cells that are organized as a quad-tree as done in section 5.1.1. The basic tree generation algorithm is described in sections A.2.1 and 4.2.2. The cause elements are the viscous boxes and the sheets and other particles are the effects. All cells containing part of a viscous box are updated as described in section 5.1.1. To find the velocity field due to the sheets, all the sheets in each childless cell are considered. The viscous boxes inside which each sheet extends is found by considering the extremities of the sheet. Once the extent of influence of each sheet is known, the velocity field can be rapidly computed. Thus each sheet only influences a few sheets and particles in its vicinity.

5.3 Conversion of sheets and blobs

The conversion of sheets to blobs and vice versa is also an important component of the hybrid RVM. This can potentially be computationally expensive. While the conversion of particles to blobs and sheets is automatically handled when the particles are undergoing a displacement as discussed in section 5.1, it is useful to be able to convert blobs and sheets outside of the context of particle motion. For example, when blobs and sheets are merged, their position might change to conserve the moments of the vorticity. Therefore, it is possible for a sheet to have moved outside the numerical layer and for a blob to have moved into the numerical layer. Hence, it is useful to have a fast algorithm to handle these conversions.

It is easy to see that an algorithm similar to the one used to compute the sheet velocities can be used here. A quad tree of cells is generated for the viscous boxes, sheets and the blobs. This algorithm is described in sections A.2.1 and 4.2.2. The cause elements are the viscous boxes and the sheets and blobs are treated as effects. The cells are modified based on the extent of the viscous boxes as described in section 5.1.1. Each sheet in a cell is considered. It is determined if the sheet lies within any of the viscous boxes inside the cell. If no viscous box is found that contains the sheet, then the sheet is converted to a blob. Similarly, if a blob lies inside a viscous box, it can be converted to a sheet. Thus by using the quad tree of cells it is possible to perform the conversion of sheets and blobs efficiently.

5.4 Annihilation and merging of vorticity

The idea of annihilation and merging of particles was introduced in section 3.5. This annihilation and merging of the particles can be performed efficiently using a fast algorithm. The algorithm is similar to those described in sections 5.2 and 5.3. The sheets and blobs are annihilated separately. In each case there are only "cause" particles in the annihilation or merging because there is only one species involved. That is, either the sheets or the blobs alone are involved in the annihilation and merging. The fast algorithm works by first organizing the particles into a quad tree of cells as described in sections A.2.1 and 4.2.2. The particles in each cell are considered and nearby particles in the same cell and inside the neighboring cells are considered for merging or annihilation. Using the tree structure makes this computation highly efficient.

5.4.1 A numerical study of annihilation and merging

A simple one-dimensional problem is considered to numerically study the annihilation and merging algorithms. Two Gaussian distributions of opposite sign are considered. The diffusion of these two Gaussian vortices is numerically studied. The exact solution is known and is the sum of the solutions of the two diffusing Gaussian distributions. Consider a Gaussian vorticity distribution,

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}.$$

The exact solution for the diffusion of this distribution is obtained if at any time t, $\sigma(t)^2 = \sigma^2 + 2\nu t$, where ν is the kinematic viscosity. Since the heat equation is linear, the exact solution for the diffusion of two oppositely signed vortices is easily obtained as the sum of the two Gaussian distributions.

Two vortices with an initial standard deviation, $\sigma = 0.1$ spaced 0.8 units apart are considered. Each Gaussian is discretized into 201 individual particles inside a region $-5\sigma \leq x - x_0 \leq 5\sigma$ around the center of the Gaussian, x_0. The strength of each particle is the value of the function at the point into the interval length between the points. The simulation is run for 10 seconds with $\Delta t = 0.5s$ and $\nu = 0.01$. 8 trials are made and the ensemble of these is considered for the computation of the error. The vorticity distribution is obtained from the new position of the particles using the optimal smoothing approach discussed in section B.1.5. The L^2 error is computed between the exact solution and the computed solution as the merging and annihilation parameters are varied. The standard deviation between the various trials is also computed. Let the annihilation radius[1] be denoted as R_a and the merging radius, R_m and the number of particles in the simulation as N. Two same signed particles are not merged if the sum of their strengths is larger than the maximum strength of the initial distribution of particles.

Table 5.2: Error, standard deviation and number of particles as the annihilation radius, R_a is varied. No merging is performed.

R_a	Error	Deviation	N
0.0000	0.08767	0.05964	402
0.0005	0.08023	0.07072	299
0.0010	0.08348	0.06292	275
0.0025	0.09154	0.06064	248
0.0050	0.08372	0.05567	238
0.0100	0.08830	0.06281	238
0.0200	0.07875	0.05003	227
0.0400	0.09441	0.07167	225
0.0800	0.10493	0.06559	220
0.1600	0.11508	0.05528	204
0.3200	0.17397	0.04277	175
0.6400	0.30842	0.03441	112

Table 5.2 shows the errors for the case where R_a is varied and no merging of the particles is performed. It is clearly seen that annihilation reduces significantly

---

[1] R_a and R_m are defined as non-dimensional lengths and in this case, they are multiplied by 1 to give an absolute distance

the number of particles. If R_a is reasonably chosen, the errors are slightly better than those without any annihilation. However increasing R_a too much introduces significantly larger errors.

Table 5.3 shows the errors for the case where R_m is varied and no annihilation is performed. The results are not completely conclusive and it appears that the merging does not introduce any significant errors.

Table 5.3: Error, standard deviation and number of particles as the merging radius, R_m is varied. No annihilation is performed.

R_m	Error	Deviation	N
0.0000	0.08767	0.05964	402
0.0005	0.10711	0.07512	222
0.0010	0.07724	0.06273	175
0.0025	0.12383	0.07891	140
0.0050	0.11032	0.08056	126
0.0075	0.09315	0.09034	122
0.0100	0.06698	0.05986	120
0.0200	0.07368	0.06853	117
0.0400	0.09047	0.07354	115
0.0800	0.08874	0.07283	113
0.1600	0.09375	0.07180	111
0.3200	0.09671	0.08665	108
0.6400	0.08808	0.08846	108
1.0000	0.07557	0.07623	109
2.0000	0.10095	0.09038	108

Table 5.4 shows the errors as R_m varies with R_a fixed at 0.02. Once again, there is a large reduction in the number of particles with no appreciable increase in the errors or in the standard deviation except for increases in the deviation for the extremely large R_m values. It is therefore clear that using a reasonably small annihilation distance along with a similar value for R_m (=0.02) reduces significantly the number of particles by a factor of five without introducing large errors.

Table 5.4: Error, standard deviation and number of particles as the merging radius, R_m is varied and $R_a = 0.02$.

R_m	Error	Deviation	N
0.0000	0.07875	0.05003	227
0.0005	0.07440	0.05851	137
0.0010	0.10125	0.06857	110
0.0025	0.08256	0.06176	88
0.0050	0.10204	0.06520	78
0.0075	0.09530	0.07307	78
0.0100	0.08630	0.06901	79
0.0200	0.07599	0.06484	76
0.0400	0.08994	0.06697	73
0.0800	0.08616	0.06142	72
0.1600	0.07810	0.06642	72
0.3200	0.08777	0.08258	71
0.6400	0.07864	0.07728	71
1.0000	0.08215	0.08292	71
2.0000	0.09982	0.08378	67

5.4.2 Illustration of utility of annihilation

To illustrate the utility of annihilation of vorticity, the case of an impulsively started circular cylinder at $Re = 3000$ is considered. A non-dimensional time $T = Ut/R$ is defined, where U is the velocity of the cylinder and R is its radius. Fig. 5.14 plots the vortex particles at $T \approx 2$ for the case where no annihilation is performed. The red colored dots (blobs) and yellow colored lines (sheets) represent vorticity of a clockwise sense. The blue colored dots and cyan lines represent anti-clockwise vorticity. All the particles have the same magnitude of vorticity. As can be seen, there are a large number of parasitic particles. There are about 330000 blobs and 65000 sheets in the simulation. Fig. 5.15 plots the particles at the same value of T for the case where sheet annihilation alone is performed. The number of particles reduces significantly (by a factor of 8) and the features of the flow are much clearer than in the case where no annihilation is performed. Fig. 5.16 plots the particles at the same value of T with both sheet and blob annihilation. The number of particles reduces further with attendant improvement in the quality of the vortex formation. In chapter 7 it is also shown that the annihilation improves the quality of the results significantly. The order of magnitude reduction in the

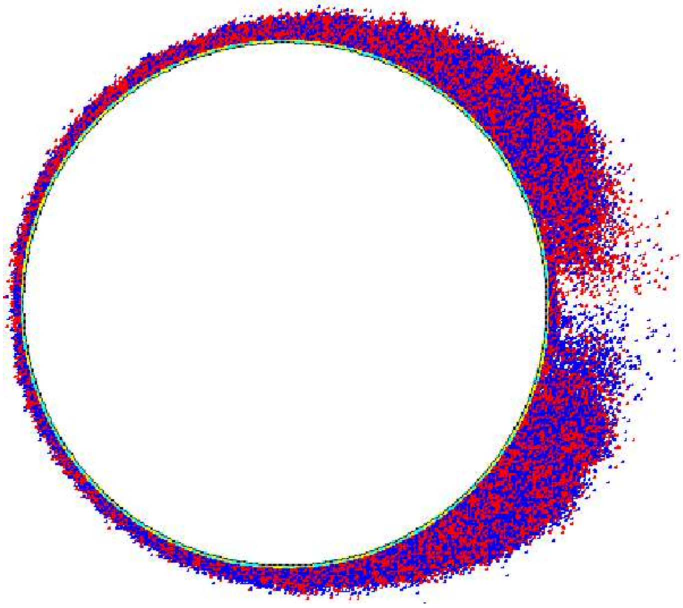

Figure 5.14: Flow past a circular cylinder at $Re = 3000$, with no annihilation at $T \approx 2$. 336340 vortex blobs and 65323 sheets are present.

number of particles in this case is significant because it results in an order of magnitude improvement in the computational efficiency of the simulation.

Thus annihilation and merging of the vorticity can be performed efficiently using the method described here. As shown, the annihilation also results in a significant improvement in computational efficiency because it reduces the number of particles in the simulation without introducing much error in the computation.

5.5 Summary

As seen in this chapter, there are other fast algorithms that are used in vortex methods. Using these fast algorithms, along with the AFMM described in appendix A and chapter 4, it is possible to simulate fluid flows efficiently using vortex methods.

It is observed that all of the fast algorithms discussed thus far rely on an organization of the particles into a tree structure. In the next chapter an object oriented design for vortex methods is developed. The important components involved in these algorithms are carefully designed and re-used.

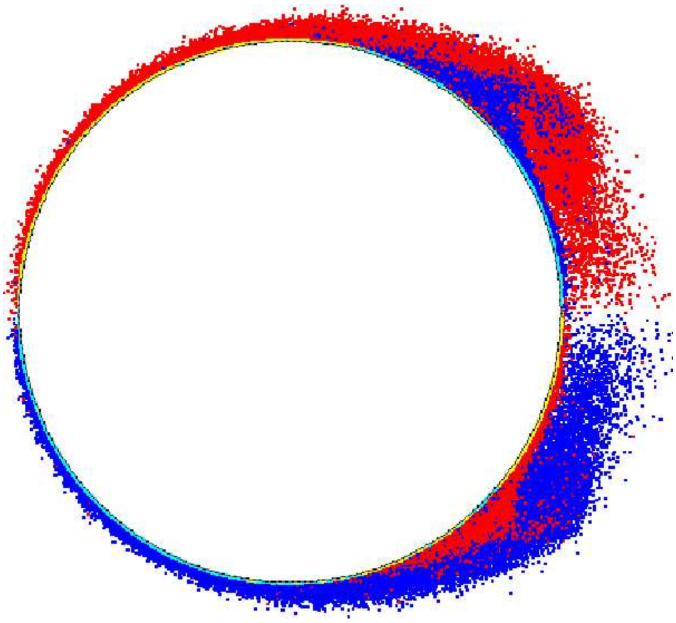

Figure 5.15: Flow past a circular cylinder at $Re = 3000$, with sheet annihilation at $T \approx 2$. 43826 vortex blobs and 6897 sheets are present.

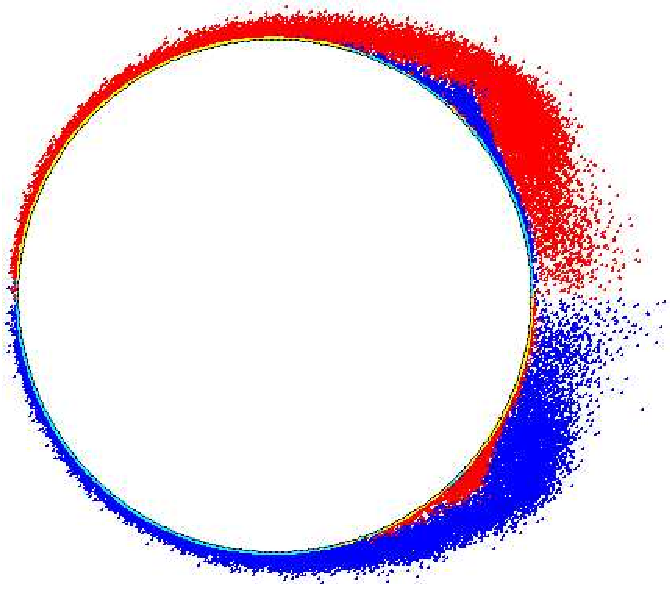

Figure 5.16: Flow past a circular cylinder at $Re = 3000$, with sheet and blob annihilation at $T \approx 2$. 36552 vortex blobs and 6031 sheets are present.

CHAPTER 6

OBJECT-ORIENTED DESIGN

In chapter 4 and 5, the important fast algorithms involved in the development of the vortex method were discussed. This chapter provides a brief overview of object-oriented design. This is followed by a presentation of the object-oriented design used for vortex methods in the present work. The importance and benefits obtained by the use of such a design are brought out.

6.1 Introduction

In a vortex method, the need for efficiency motivates the use of complex data structures and algorithms. In appendix A and chapter 4 it is seen that the AFMM is a fairly complex algorithm to implement. The AFMM together with the panel method and the algorithms discussed in chapter 5 require a reasonably large programming effort[1].

Object-oriented design (OOD) enables one to develop and handle complex software. More importantly, the design feeds back into the understanding and development of the algorithms involved. This enables one to make important improvements to algorithms. One centrally important aspect of OOD is the creation of *objects* or *templates* that mirror the entities and processes involved in the actual physical problem. That is, when programming using an object-oriented language, one tends to think more about the actual problem rather than the implementation details. A good design also allows the programmer to re-use important parts of the code. Therefore, object orientation results in the code being easier to understand, read and maintain. The approach may be likened to the development of an

---

[1] For example, around 36000 lines of C++ are used to write the libraries developed in the present work. This excludes the unit-test programs (about 12000 lines of code) and the main programs.

elegant and powerful notation as done in mathematics. The notation and power of manipulating abstract symbols in a convenient fashion enables the researchers to handle complexity and express themselves with a greater degree of clarity. As with any design process, object-oriented program design is also an iterative one. A good design therefore requires at least a few iterations of the design process. To paraphrase Paul Valery, *"A design is never finished, it is only abandoned."*

In the present work, the C++ programming language (Stroustrup, 1998) is used. C++ is efficient and enables object-oriented and generic programming. The difficulty with C++ is that it is a complex language with many features. However, it is probably the only commonly available object-oriented language that produces efficient machine code. This is important for numeric computation.

The Python programming language (van Rossum *et al.*, 1991–) is an excellent, free, dynamically typed, object-oriented, scripting (Ousterhout, 1998) language that mixes very well with C++. Python is highly expressive and easy to understand. This motivated its use for the pseudo-code in appendix A. The difficulty with Python and other scripting languages is that they are too slow to be used natively for numerical algorithm development. However, using tools such as SWIG (Beazley *et al.*, 1995–; Beazley, 1996) and Boost.Python (Abrahams *et al.*, 2000–; Abrahams and Grosse-Kunstleve, 2003) it is possible to "wrap" existing C/C++ libraries and use them from Python. This enables one to write scripts in Python (or other scripting languages) using libraries developed in C++. In the present work, the C++ libraries developed have also been exposed to Python using SWIG.

This chapter describes the design and abstractions used in the developed vortex based solver. The object design is discussed without providing distracting details on the specifics of the implementation. For a general introduction to object-oriented programming the reader is referred to Budd (1991).

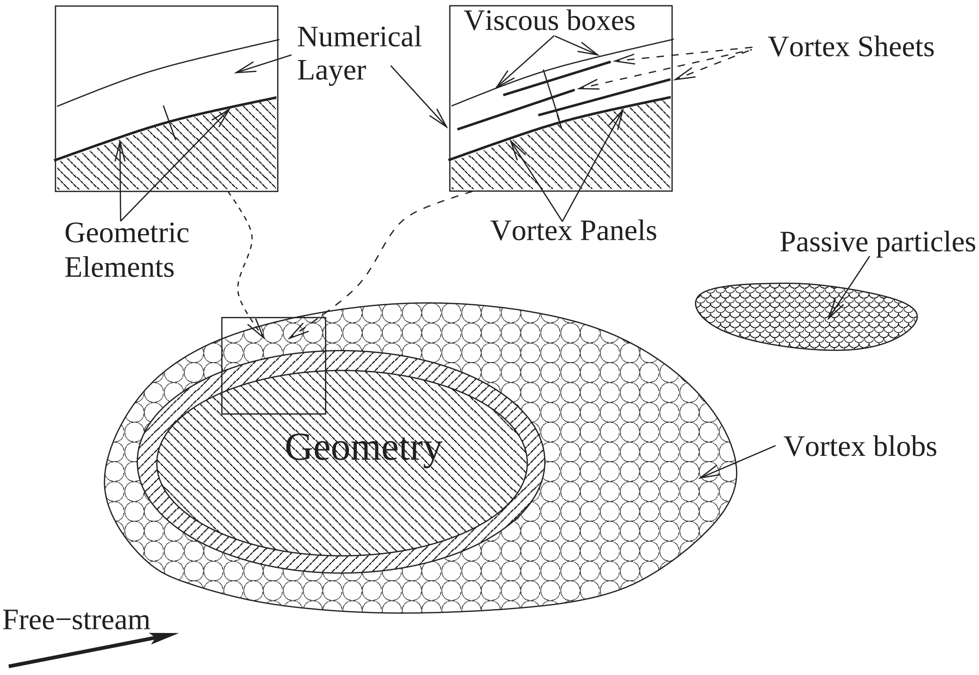

Figure 6.1: Various entities involved in a vortex method.

6.2 Design overview

The easiest way to understand the design is to first consider the various entities involved in the vortex method. The basic and most important ones are illustrated in Fig. 6.1. A detailed discussion of the entities in the figure is provided in earlier chapters. The first step of the design involves the creation of *classes* encapsulating the behavior of these entities. Objects are created as instances of these classes. The controlled interaction of these objects results in the vortex method simulation. The algorithms involved in the computation of these interactions are shown in Fig. 6.2. The items colored red require special algorithms for efficiency.

Thus, the most natural design would be one where the illustrations in Figs. 6.1 and 6.2 directly translate into the program design and code. This is exactly what object-oriented programming and design allows for. As the description of the design of the code proceeds, the manner in which the design fits in with general description of the problem illustrated in the figures will be shown. Before the design is discussed in detail it is important to note some key issues that drive the design.

1. All the particles used in the computational scheme can be thought of as

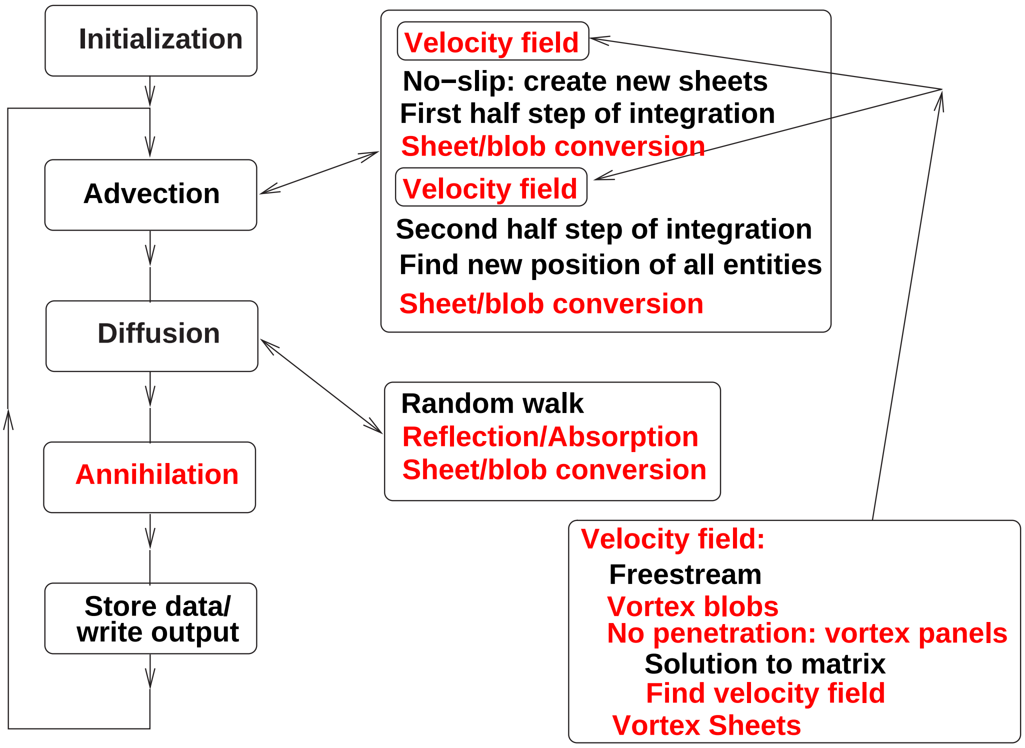

Figure 6.2: A schematic of the vortex method algorithm. Red items require special techniques for computational efficiency.

"fluid elements". These elements are defined by a position, a velocity and usually carry some fluid properties (like circulation/vorticity). For example, a vortex blob carries a circulation while a passively advected tracer particle does not carry any special fluid property.

2. Providing an accurate representation for the geometry is important. A one-dimensional curve can be represented as a collection of several linear/-parabolic or higher order elements connected to one another. However, bodies with distinct parts and multiple bodies can pose complications. These must be handled appropriately.

3. As seen in appendix A and chapters 4, 5, identifying clusters of particles based on relative distances is an important and commonly used algorithm. It is important to abstract this algorithm.

4. Appendix A and chapter 4, show that that there are several flavors of the AFMM. Hence, abstracting a generic AFMM algorithm is of considerable use.

The next section describes the notation used to illustrate the relationship of the various classes used in the design.

6.3 Notation for class diagrams

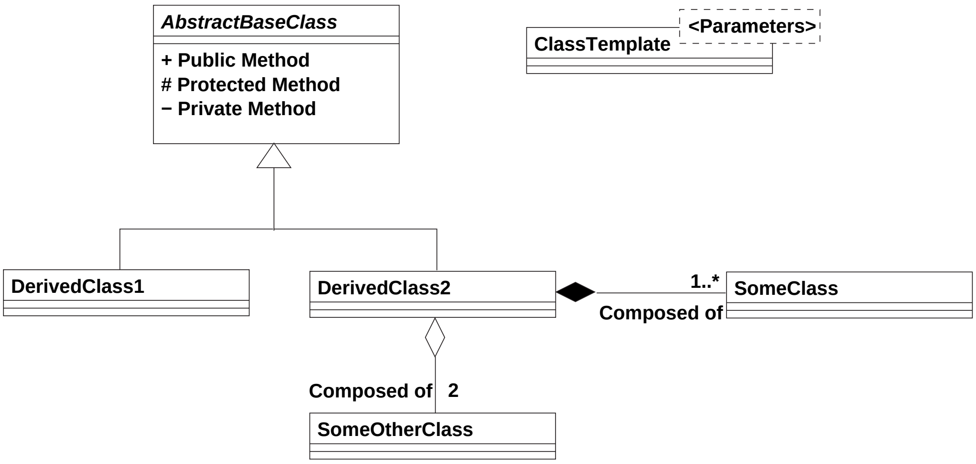

Figure 6.3: Legend for the symbols used in class diagrams.

Simple UML (Unified Modeling Language) class diagrams are used to elaborate the design. Fig. 6.3 demonstrates the basic ideas. In the figure, `AbstractBaseClass`, `DerivedClass1` and `DerivedClass2` depict an inheritance relationship. `AbstractBaseClass` is abstract and this is indicated by the fact that its name is italicized. `DerivedClass1` and `DerivedClass2` derive from this abstract base class. The methods of the abstract base class are prepended with different symbols (`+`, `#` and `-`) to indicate if they are public, protected or private members respectively. The `ClassTemplate` shows how a class template is depicted in the figure along with its parameters. `DerivedClass2` is composed of (or contains) `SomeClass` and `SomeOtherClass` objects. The white diamond for `SomeOtherClass` indicates a lack of ownership. That is, if the memory occupied by an instance of the `DerivedClass2` is freed, the `SomeOtherClass` objects it is composed of are not deleted. The black diamond indicates ownership and when an instance of `DerivedClass2` is deleted, the `SomeClass` objects it contains are all deleted. The number near the arrow indicates the number of items the class contains. In the illustration, `DerivedClass2` will contain one or more `SomeClass` objects and exactly two `SomeOtherClass` objects.

The next section describes the design of the vortex based solver in detail.

6.4 Design details

An object which is an instance of a "solver" class manages the entire simulation. The behavior of this object is very similar to the illustration in Fig. 6.2. The solver marshals the simulation. For the initialization, the solver object reads a data file describing the geometry, the input vortex particles, vortex sheets and various other parameters used in the simulation. The simulation involves the advection and diffusion of the particles. There are two member functions of the class that are responsible for the advection and diffusion. These in turn use other helper objects to divide the work. The solver creates "managers" for the geometry, vortex blobs, sheets and the panel method. These are described in detail in the following subsections.

6.4.1 Geometry

In the present work only solid bodies are considered. The geometry is split into several logical units. A `Geometry` object describes the geometry completely. Each `Geometry` contains several `Body` objects. A `Body` in turn is a collection of connected `Part` objects. Each `Part` is composed of several `CubicGeomElem` objects. A `CubicGeomElem` is a piecewise cubic element and is at the lowest level of the hierarchy. A `Part` is basically a simple curve consisting of connected `CubicGeomElem` objects. Clearly, by using this approach one can describe highly complex geometries and yet retain information on the logical parts of the geometry. Fig. 6.4 illustrates the case of two bodies (Body1 and Body2). Body1 has four parts (Part1 to Part4). Body2 has two parts. Body1 and Body2 together form the geometry.

It is sometimes useful to obtain a representation of this geometry using linear elements rather than cubic elements[2]. In order to do this, a separate set of flat elements are generated and one `LinearCover` object for the geometry is generated. This represents the geometry in terms of linear elements.

As described in section 5.1.1, when vortex sheets are used, it is useful to define

---

[2] This is useful when detecting particle collisions as described in section 5.1 and interpolating the vorticity to a grid as described in section B.1.4.

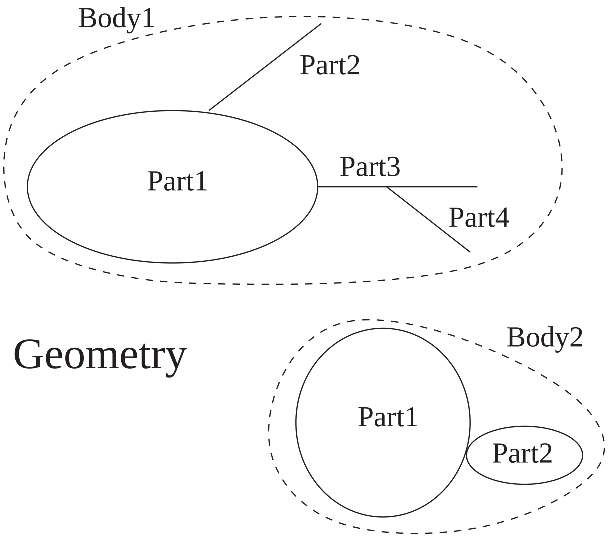

Figure 6.4: Geometry consisting of various bodies and parts.

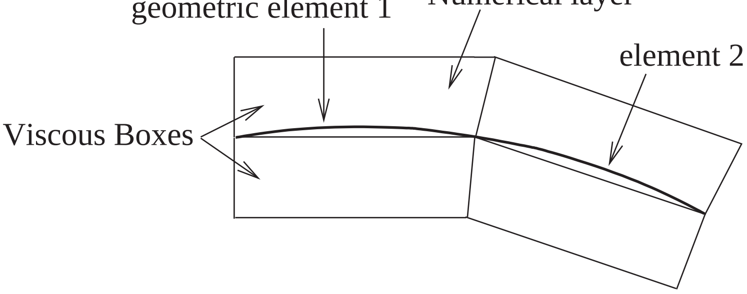

Figure 6.5: Illustration of viscous boxes for curved geometric elements.

a "viscous box" that is used to discretize the numerical layer. Fig. 6.5 illustrates two `CubicGeomElem`'s (geometric element 1 and element 2) that discretize part of an open body with zero thickness. The geometry is curved and has two sides. It is covered with four viscous boxes. Each viscous box can be defined as a trapezium with height equal to the numerical layer height, h_{num}, and with one side as the chord of the geometric element. A `ViscBox` class represents this viscous box. Instances of the `ViscBox` class are stored in a `ViscBoxManager` that manages their construction and destruction. The viscous boxes are constructed from a given `Geometry` object. For closed bodies, `ViscBox` objects are generated on the side of interest and for open bodies they are generated on both sides as shown in Fig. 6.5.

6.4.2 Fluid particles

As illustrated in Fig. 6.1, the vorticity field is discretized into blobs of vorticity. In the vicinity of the body, the vorticity is discretized into vortex sheets. Vortex blobs are circular in shape and have a finite radius called the "core radius". Vortex sheets on the other hand are flat in shape and have zero thickness and a finite length. Apart from vortex particles, tracer particles that are passively advected by the fluid are also useful. As mentioned earlier, these different types of particles have a velocity and a position associated with them. This suggests a convenient abstraction in the form of an abstract base class called the `FluidElement`. The `FluidElement` class merely defines an interface for any object that has a position and a velocity. A concrete instance of this class is a `FluidParticle`. This is a simple class that encapsulates a position and a velocity vector. Instances of the class can be used as passive tracer particles in a simulation. Derived from the `FluidParticle` is an abstract `VortexElement` class that abstracts the behavior of a vortex blob. A `VortexElement` is essentially a particle that has a total circulation and a core radius. A `VortexElement` can induce a velocity at any point in space. The complex potential induced by it at a point can also be computed. Concrete classes derived from the `VortexElement` class are implemented as the various blobs found in literature (point vortex, Saffman blob, Chorin blob, Krasny blob, etc.)

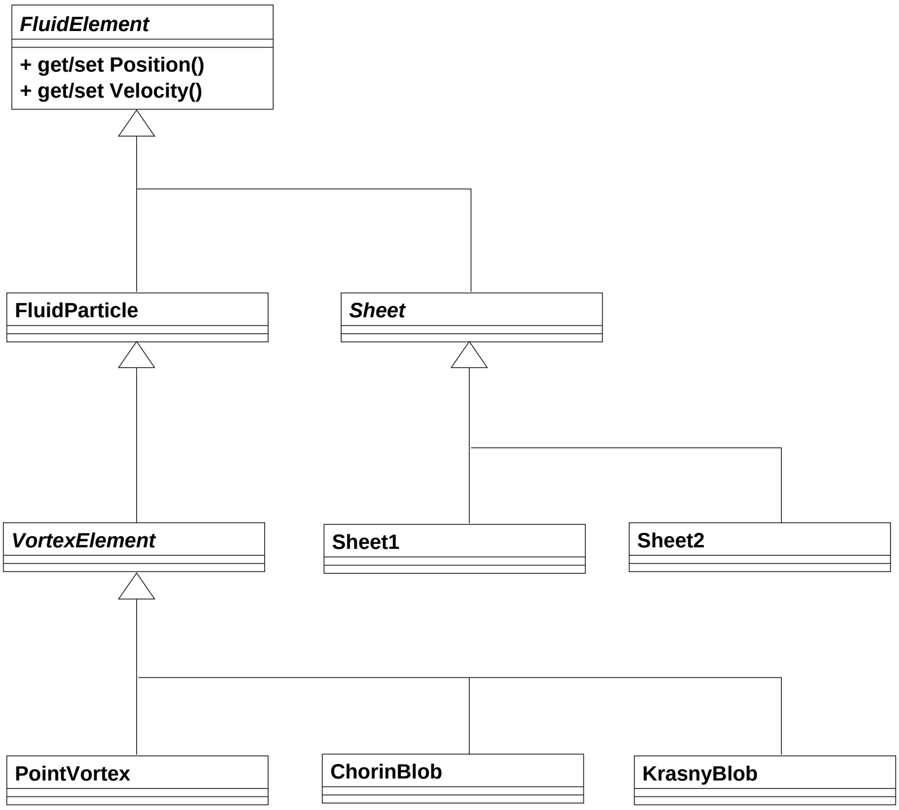

Figure 6.6: Class diagram for the fluid elements.

as shown in Table 2.1.

A vortex sheet is a slightly different entity from the vortex blob in that it is anisotropic and has an orientation. However, it too shares the same properties as a fluid particle in that it has a position and a velocity. Similar to a `VortexElement`, it induces a velocity on another point and also has a known circulation. A `Sheet` is therefore an abstract base class derived from a `FluidElement`. Concrete subclasses of the sheet are derived as `Sheet1`, `Sheet2` etc. as discussed in section 2.4.

The class diagram and relationship between the various classes is illustrated in Fig. 6.6. This is not a comprehensive list or description of the respective classes but is indicative of the general idea. As can be seen, these classes mirror the physical entities illustrated in Fig. 6.1.

During a simulation, a very large numbers of the basic particles may be created. These particles need to be managed. For each particle class, a separate manager object is created. An abstract base class `FluidElementManager` specifies the interface of any particle manager by providing a method to obtain any of

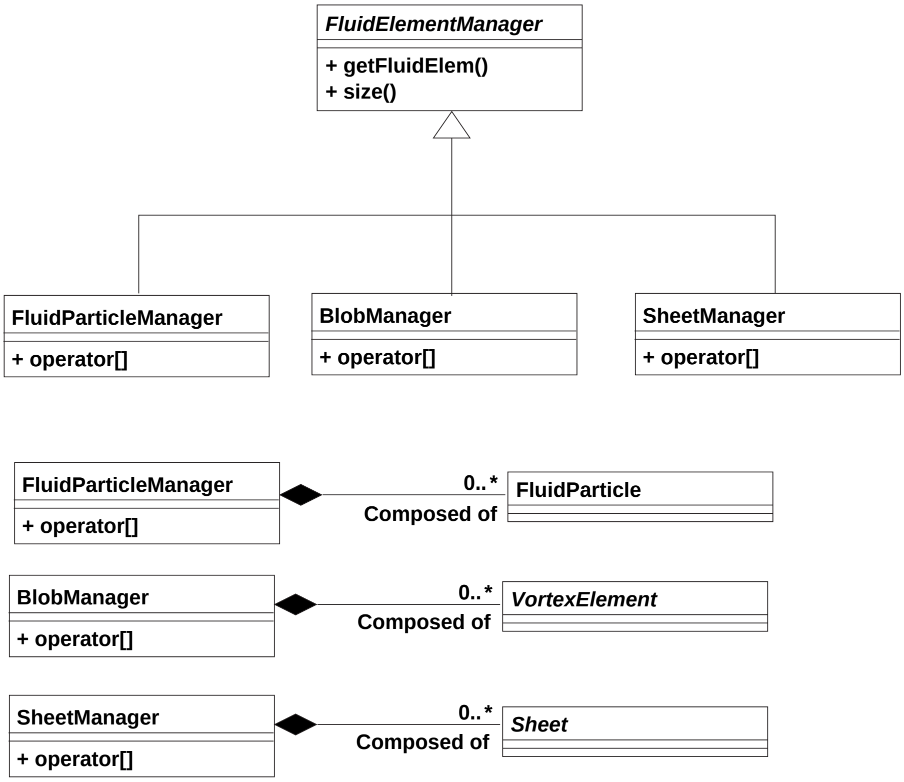

Figure 6.7: Class diagram and containership relation for the fluid element managers.

the contained particles as a `FluidElement`. This implies that all objects managed with such a manager can be treated like a `FluidElement`. This is a key and useful abstraction that is used to simplify several algorithms. Having one abstract interface that provides a common interface also means that functions can be passed pointers to this type of a manager and they will work for any of the derived managers so long as the notion of a `FluidElement` is sufficient. This will be illustrated with more examples in the sections that follow.

Concrete subclasses of the `FluidElementManager` are created to contain and manage the `FluidParticle`, `VortexElement`, `Sheet`, and other objects. These are called `FluidParticleManager`, `BlobManager`, `SheetManager` etc. Each of these has a method called `operator[]` that allows one to obtain the contained object pointer. The classes and their relationship to the other objects are illustrated in Fig. 6.7.

6.4.3 No-penetration BC: the panel method

The panel method (section 3.6) is used to satisfy the no-penetration boundary condition on the solid body. The panel method discretizes the body into panels. Vorticity is distributed linearly on the surface of these panels. The panels can have either a linear or cubic geometry.

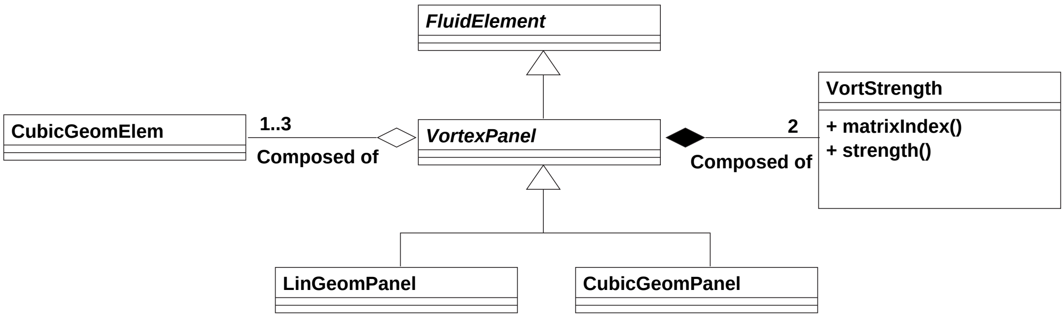

Figure 6.8: Class diagram for the vortex panels and related classes.

The behavior of a panel is encapsulated in an abstract `VortexPanel` class. As shown in Fig. 6.8, it is derived from the `FluidElement` class. This is because each `VortexPanel` has a control point that has a position and a velocity induced on it. Derived from `VortexPanel` are the `LinGeomPanel` and `CubicGeomPanel` concrete classes that use a linear geometry and a cubic geometry respectively.

Each panel contains two `VortStrength` objects that store information on the strength of the vorticity at each edge of the panel. The `VortStrength` object also stores a position index to the location on the matrix used to solve for the unknown strengths. This eases the generation of the influence matrix for the panel method. It also makes it easy to handle attachments and complex shapes. The `VortexPanel` objects also contain pointers to the underlying `CubicGeomElem` objects that define the `Geometry`.

An instance of the `PanelManager` class manages all the `VortexPanel` objects. As illustrated in Fig. 6.9, the `PanelManager` class is derived from the `FluidElementManager` class. The panels are organized in the same manner as the `Geometry` object into `Part` and `Body` objects. A `PartPM` object is similar to the corresponding `Part` object and contains a list of pointers to `VortexPanel` objects that represent the panels for a particular part. A `BodyPM` manages a col-

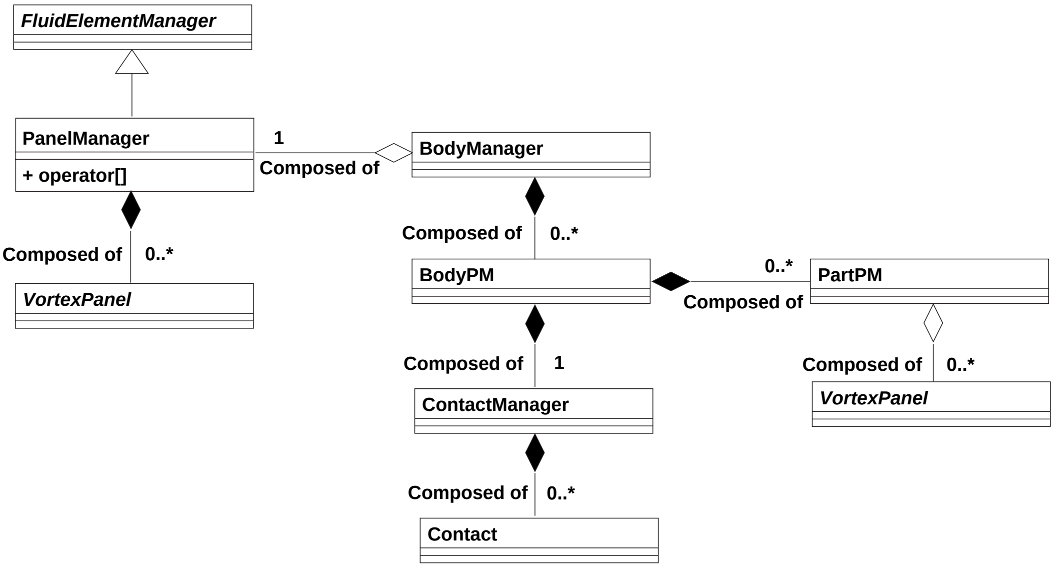

Figure 6.9: Class diagram for the panel method and related classes.

lection of `PartPM` objects. The `BodyManager` class is equivalent to the `Geometry` class. In order to handle complex attachments, two additional classes are defined called a `Contact` and a `ContactManager`. Each `BodyPM` manages a collection of attached `PartPM` objects. Each `PartPM` of a `BodyPM` is attached to other `PartPM` objects at a `Contact`. The `ContactManager` manages all the contacts for a particular `BodyPM`. The `ContactManager` is responsible for generating extra conditions to solve the panel method correctly when complex geometries are used. These classes and their relationships to other classes are illustrated in Fig. 6.9.

The `BodyManager` provides a simple but high level interface to the panel method. It is capable of using a `Geometry` object and generating the necessary `BodyPM` and `PartPM` objects. It also provides member functions to solve the matrix and obtain the value of the vorticity of the panels.

With the classes described up to this point it can be seen that all the entities in Fig. 6.1 with the exception of the `FreeStream` have been defined above. The `FreeStream` class is described in the section on advection (6.4.6). Hence, by using the classes described above it is possible for the programmer to describe the physical problem in the computer program.

6.4.4 Algorithms for hierarchically organizing particles

Given a collection of `FluidParticle`, `VortexElement` and `VortexPanel` objects, it is in theory possible to develop solvers employing the vortex method. As discussed in appendix A, computing the velocity field of the vortices on each other is an $O(N^2)$ operation. This can be reduced to an $O(N)$ or $O(N \log N)$ computation using one of many Adaptive Fast Multipole Methods (AFMM). The AFMM algorithm requires that the particles be organized into clusters of particles. As seen in chapter 5, organizing particles into clusters based on their relative proximity also turns out to be very useful in other contexts for a vortex method. Particles are organized into clusters of particles using a quad tree as discussed in section A.2.1.

In order to organize particles into clusters, an abstract base class called the `BasicCell` is defined. The class is illustrated in Fig. 6.10. The `BasicCell` defines methods to obtain the center and level of the cell (in the hierarchy). The class also defines methods to indicate if the particular cell is a parent or not. Concrete subclasses of the `BasicCell` will contain particles of two flavors, "causes" and "effects". Therefore, the `BasicCell` class provides methods to find the number of causes (`nCause()`) and effects (`nEffect()`) inside it. The concept of causes and effects is elaborated in considerable detail in section 4.2.2.

Each instance of a concrete subclass of the `BasicCell` class additionally provides methods to obtain the parent of the cell, the children of the cell and the associates of the cell (a maximum of eight associate cells exist per cell). The concrete subclass also defines a method called `createChildren()`, using which the cell will split itself into four daughter cells of the same type. The cells also provide methods to set and get the causes and effects in the cell. For brevity, these are illustrated in Fig. 6.10 only for the `AnnihilateCell` class. However, the methods are common to all concrete subclasses of `BasicCell`.

Derived from the `BasicCell` are the `MultipoleCell` and `GeneralCell` classes which are abstract. The `MultipoleCell` abstracts some functions that are useful for all multipole computations. Specifically, in the fast multipole computations, the cell treats all effects as `FluidElement` objects. Thus, the class provides meth-

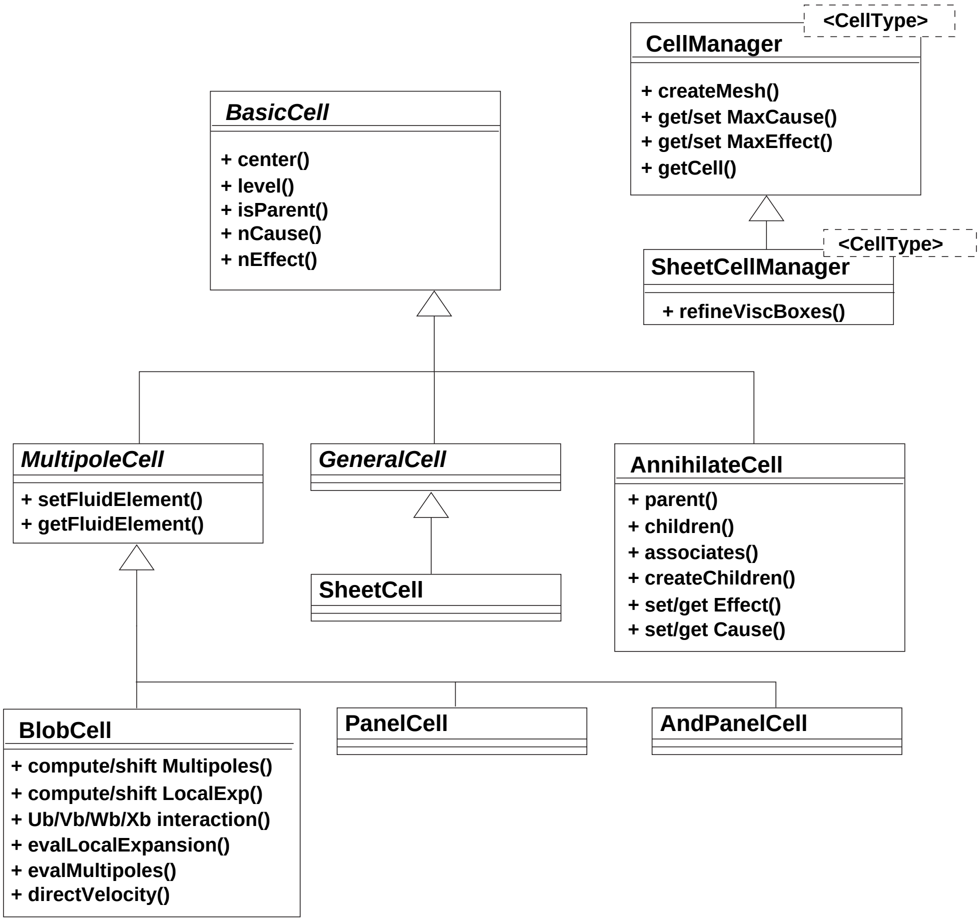

Figure 6.10: Class diagrams for the different flavors of cells.

ods to set and get the `FluidElements` inside it. Derived from the `MultipoleCell` are concrete classes that implement the specifics for a particular type of fast multipole computation. There are several flavors of the cell types like `BlobCell`, `PanelCell` etc. For a `BlobCell`, `VortexElement` objects are the cause particles and everything else is treated as a `FluidElement` (an effect). Similarly for the `PanelCell`, `VortexPanel` objects are the causes and the rest are effects. Each of these concrete subclasses define the methods relevant for the AFMM. This is illustrated in Fig. 6.10 for the `BlobCell` class. The `PanelCell` implements methods similar to the `BlobCell` that are suitable for the hybrid flat/cubic panels as described in section 4.2. The `AndPanelCell` implements the methods suitable for Anderson's method as applied to panels. This is described in section 4.3.2. There are two other AFMM related cell classes that are not illustrated in the figure. One is `FCPanelCell` which implements the methods suitable for the fast cubic panels described in section 4.3 and the other is called `AndBlobCell` which implements Anderson's method for blobs. `AndBlobCell` and `AndPanelCell` are used to compute the stream function due to the blobs and panels rapidly.

The `GeneralCell` is different in the way it is implemented. The other cells store pointers to the various elements contained inside them. However, sometimes it is useful to store indices of the entities rather than their pointers. This is used in the context of the conversion of sheets to blobs and vice-versa, to find the velocity of sheets on each other and to perform a random walk in the presence of arbitrarily complex geometries. The `SheetCell` class is used for these.

The `AnnihilateCell` is a concrete class that enables the annihilation of nearby vortex sheets in the numerical layer. This cell is special because effects are identical to the causes in its context.

Since all of these cell types are derived from a `BasicCell`, they all have member functions that specify the number of cause and effects in the cell. They also provide methods to access to the colleagues of the cell. Therefore, it is possible to use a `CellManager` class template with template parameter `<CellType>`, that will organize the corresponding cells into a quad-tree based on the number of causes and effects in the cell and its colleagues. The algorithm is itself described in section 4.2.2

and in (Ramachandran *et al.*, 2001, 2003). The `CellManager<CellType>` therefore creates and manages cells. The important methods defined by this class are illustrated in Fig. 6.10.

A `SheetCell` is slightly different because it organizes `Sheet` and `ViscBox` objects. These objects have extent and an orientation. In order to handle this a `SheetCellManager<CellType>` is derived from the `CellManager`. This defines a special method that updates the generated cells with information on which `ViscBox` object passes through the cells. The specifics of the algorithm are detailed in section 5.1.1.

Thus, in this work cells are used to hierarchically organize a distribution of particles. Each specialized cell class implements the specifics of the algorithm in which one is interested. The `CellManager` generates and organizes these cells. Once the cells are organized it is possible to use them in a variety of fast algorithms. Therefore, with one implementation of the `CellManager` class template (implemented as one header file and one source file), it is possible to build the cell structure for eight different algorithms. Thus, object-oriented design and generic programming enables a large amount of code re-use. This enhances the readability and maintainability of the code.

6.4.5 Adaptive Fast Multipole Method

In order to reduce the $O(N^2)$ computation of the blob velocities to an $O(N)$ one, an adaptive fast multipole method (AFMM) (Carrier *et al.*, 1988) is used. This is described in detail in section A.3. There are several flavors of the AFMM that are developed and used in the present work. Two algorithms to find the blob velocities have been developed, one using the AFMM and the other using Anderson's FMM without multipoles approach (Anderson, 1992). Three methods are implemented to find the effect of the panels on other particles. One using the approach detailed in section 4.2, the second using cubic panels as done in section 4.3 and the third using Anderson's scheme as described in section 4.3.2.

All of these use the same algorithm for the AFMM and require a similar cell structure. As seen in the previous section, the different cell types handle the specifics of the algorithm and the `CellManager` generates the quadtree. A `FastMultipoleManager` class template with parameters, `<DataManager, CellType>` is used to implement the AFMM. The `DataManager` parameter is either a `BlobManager` or `PanelManager`. The parameter `CellType` can be any subclass of the `MultipoleCell` that implements the necessary methods for the AFMM. The `FastMultipoleManager` instantiates the appropriate `CellManager` and uses it internally to generate and manage the cells. The class then uses these organized cells to implement the AFMM.

The key idea used in the abstraction is in the treatment of the effects by using a `FluidElement`. For example, consider the case where the velocity of the blobs is to be computed on the panels, sheets and passive particles. The `velocity` method of the `FastMultipoleManager` is given a list of pointers to `FluidElementManager` objects. Pointers to `PanelManager`, `SheetManager` and `FluidParticleManager` objects are passed to the method. Since all of them are sub-classed from the `FluidElementManager` class, the velocity on all of them can be easily computed in one stroke. The implementation of the `FastMultipoleManager` closely mirrors the pseudo-code given in section A.3.5.

Hence, using these abstractions it is possible to use one implementation for the `CellManager` and one for the `FastMultipoleManager` and yet develop five different flavors of the AFMM. In order to obtain a fast velocity due to a large number of particles the user does not even need to know about the `CellManager` since the `FastMultipoleManager` provides a high level interface that hides these details. Thus, the object-oriented design developed here results in code that is fairly easy to understand, use and extend. As shown above, it is possible to develop different flavors of the algorithms easily and yet re-use large parts of the key algorithms.

6.4.6 Advection

In order to advect the particles based on the velocity field, the `FastMultipoleManager` class template is used to obtain the velocities due to the blobs and vortex panels. The velocity of the `Sheet` objects on other sheets and particles is obtained using a `SheetVelocityManager` object. As described in section 5.2, this object also organizes the particles into clusters of cells using the `SheetCell` class and then computes the velocities. Using a quad tree structure to locate the sheets and particles makes the computation very efficient.

In order to simulate a free-stream, an abstract class called `FreeStream` is created. `ConstFreeStream` is a constant free stream class that is a concrete subclass of the `FreeStream` class. The `FreeStream` can set the velocity on any given `FluidElementManager` object. Thus, the velocity due to the `FreeStream` can be computed on any of the particles in the vortex method.

Using the computed velocity due to all the interacting species, the ODEs governing the particle motion are solved to obtain the new particle positions. The ODE solver implemented is not general and is a fairly straight forward implementation. Euler's first order method, Runge-Kutta second order and fourth order methods are implemented. During the intermediate steps of the advection (in the second order and fourth order method) it is possible for sheets to be converted to blobs and vice-versa. The manner in which this is handled is discussed in the next section.

6.4.7 Diffusion

In order to simulate diffusion the blobs and sheets are made to undergo a random walk. The details of the algorithms used in this context are discussed in section 3.4. The motion of the particles is random. If the particle strikes a solid wall it is to be reflected off the surface of the wall. This is done efficiently using a `DiffusionManager` class. The algorithms to perform this use a quad-tree structure to accelerate the computation and are described in considerable detail in

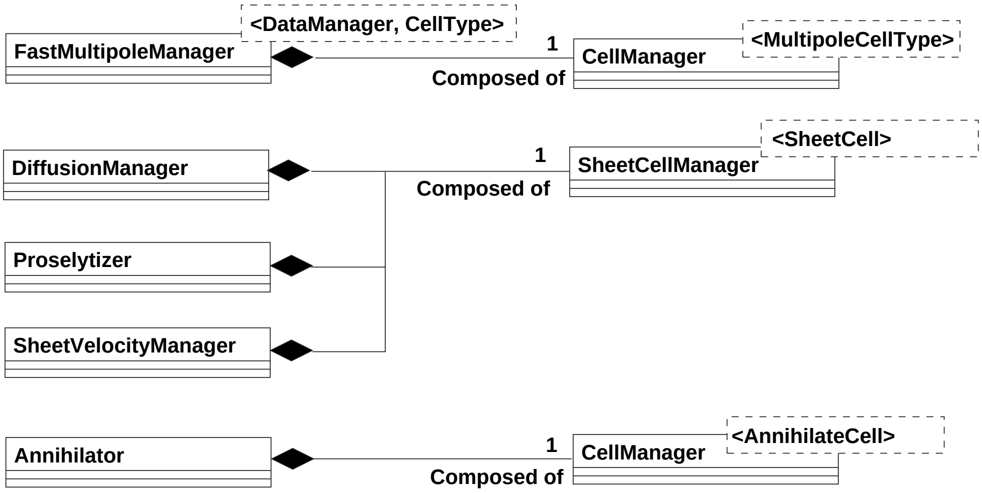

Figure 6.11: Relationship of fast algorithms and the cell managers.

section 5.1. The class uses the `SheetCells` and a `SheetCellManager` to manage them.

In order to convert blobs to sheets and vice-versa, a `Proselytizer` class is used. This class also uses a quad tree structure via the `SheetCellManager` class template for efficiency. An overview of how the class works is provided in section 5.3.

Thus, by handling the diffusion and inter-conversion of sheets and blobs the diffusion equation is solved.

Annihilation and merging of the sheets and blobs are handled using an `Annihilator` class. This class also uses a `SheetCellManager` to efficiently organize the particles into clusters. The algorithmic details of the annihilation and merging process are described in sections 3.5 and 5.4.

6.4.8 Vortex methods

The aspects of the vortex method algorithm that require the use of fast algorithms are are colored red in Fig. 6.2. The various classes responsible for these fast algorithms have been described above. These classes and their relationship to the `CellManager` class are illustrated in Fig. 6.11.

By putting together all the described components it is possible to develop an efficient vortex method. In the present work the main driver class (or solver class) is called `Vebtifs` which stands for Vortex Element Based Two-dimensional Incompressible Flow Solver. This class is responsible for the entire algorithm as described in section 6.2. Its structure is similar to that depicted in Fig. 6.2.

Apart from the classes described in the previous sections there are other helper classes used to compute the diagnostic quantities, transfer the vorticity to a grid, perform graphics, generate data files etc. For brevity and clarity these have not been discussed here.

6.4.9 Unit testing

The code developed is tested fairly rigorously using unit tests written with the CppUnit (Feathers *et al.*, 2000–) library. A unit test is essentially a program where the various objects and their methods are tested systematically. Each object is set to a known state and the results obtained by invoking different methods are checked against expected values. If the values match, the test passes, if not the test fails. The CppUnit library formalizes this by providing a set of useful classes that are used to write the tests. The library makes it easy to see where the error occurs. It also helps to automate the test by creating "test suites". Thus, it is possible to check the state of the developed code. Writing tests is usually not a pleasant activity and for complex libraries it involves a reasonable amount of work[3]. However, writing tests is extremely important to ensure program correctness. The tests also make it easy to change the library. The unit tests ensure that program correctness is maintained as changes are made to the code. This allows the developer to be confident that the changes have not affected any of the tested functionality.

---

[3] In the present work about 12000 lines of code have been written exclusively to test the library.

6.5 Summary

In this chapter an object-oriented design for vortex methods was presented. Central to the implementation are several abstractions. The interacting particles are represented as subclasses of `FluidElement` which are managed using subclasses of the `FluidElementManager` class. The use of the `CellManager` class template abstracts the centrally important algorithm which generates the hierarchy of cells used to identify clusters of particles. The `FastMultipoleManager` class template defines the adaptive fast multipole method. This class template is used by various flavors of the AFMM. The overall design of the code maps directly into the physical process of the vortex method simulation. This makes the code easy to understand and develop. Thus, using the developed object-oriented design it is possible to implement an efficient, understandable and easy to extend vortex method based flow solver.

CHAPTER 7

PARAMETRIC STUDY OF THE RVM

In the previous chapters the theoretical, implementation and design details of vortex methods were described. There are various parameters involved in an implementation of the random vortex method. In this chapter, a parametric study is carried out to determine the optimal choice of these parameters. The computational scheme and the parameters to be varied are reviewed first. Schemes to relate the parameters to each other are investigated. The flow past an impulsively started circular cylinder at $Re = 3000$ is considered. The different parameters are varied and the results compared with available data. Using the study, several recommendations are made regarding the optimal choice of the parameters.

7.1 Computational method

Consider the flow past an impulsively started circular cylinder of radius, R. The circular cylinder is discretized into N viscous boxes of equal size. These viscous boxes are used to release vortex sheets from the surface of the cylinder. If the standard viscous splitting procedure (equation (3.1)) is used, the vortex method proceeds as follows:

1. Compute the slip velocity due to all the vorticity, free stream and vortex panels at the control point of each of the N viscous boxes.
    - The velocity due to the vorticity is computed using an adaptive fast multipole method (AFMM) discussed in section A.3.
    - The vortex panels are responsible for satisfying the no-penetration condition. A panel method with cubic panels and a linear vorticity distribution as described in section 3.6.2 is used for this. A fast multipole technique is used to accelerate the velocity computation as described in section 4.3.
    - The velocity due to the free stream is trivial to compute since it is a constant.

2. Based on the slip velocity at the control point of each viscous box, sheets are released in order to offset the slip. The details of the sheet release are described in section 3.4.1. These new sheets are created just on the surface of the viscous boxes.
3. The existing blobs and sheets (excluding the newly created ones) are then convected using the velocity field. A second order Runge-Kutta scheme (equation (3.3)) is used to integrate the ODEs. Other integration schemes are also studied.
4. All the blobs and sheets (including the newly created sheets) are then diffused using a random walk.
    - The blobs are given a random displacement along both x and y directions.
    - Sheets are given random displacements along the normal to the surface. The displacement of the sheets uses a tagging procedure as described in section 2.4. Subsequently, the influence of tagging is studied by disabling it.
    - If a sheet leaves the numerical layer it is converted to a blob and it looses its tag value. If a blob enters the numerical layer it is converted to a sheet with a new tag value that is not equal to any of the existing tag values of the other sheets.
5. The sheets and blobs are annihilated and merged as discussed in section 3.5. The algorithm used for the merging and annihilation is discussed in section 5.4.
6. The algorithm then repeats from step 1.

If Strang discretization (equation (3.2)) is employed the algorithm is as follows.

1. Compute the slip velocity and release new sheets to satisfy the no-slip boundary condition.
2. Diffuse the sheets (both newly created and existing) and blobs with a random walk using a time step of $\Delta t/2$.
3. Convect the particles by integrating the ODEs using an appropriate scheme.
4. Compute the slip velocity and release new sheets to satisfy the no-slip boundary condition.
5. Diffuse the sheets (both newly created and existing) and blobs with a random walk using a time step of $\Delta t/2$.
6. Annihilate and merge the vorticity.
7. Repeat from step 1.

Unless otherwise mentioned, Strang discretization is not used.

7.2 Computational parameters

The parameters and key elements involved in the computational scheme are listed below.

- Type of blob: the Chorin blob is used (see Table 2.1).
- δ is the core radius of a blob.
- N is the number of viscous boxes on the body. Note that the viscous boxes are equal sized and all of them release sheets.
- R is the radius of the cylinder.
- Δt is the time step.
- ν is the kinematic viscosity (μ/ρ).
- Re is the Reynolds number.
- γ_{max} is the maximum strength of an individual vortex sheet. The sheet strength is quantized by this number and integral multiples of these sheets of strength γ_{max} is released to satisfy the no slip condition. In some cases an extra sheet having a fraction of the strength γ_{max} are released to satisfy the slip exactly.
- h_{num} is the numerical layer height. The numerical layer is the layer inside which the sheets exist. When a sheet leaves this layer it is converted to a blob and when a blob enters this layer it is converted to a sheet.
- U is the free stream velocity of the fluid far upstream of the body.
- R_a is the annihilation distance factor and R_m the merge distance factor. If two sheets of length λ are to be considered for annihilation, $R_a\lambda$ is the largest allowed distance between them. Similarly, $R_m\lambda$ is the largest allowed distance for the merging of two sheets. The same radii are used for the blobs also.

These are a large number of parameters. It is possible to relate these parameters in two different ways. This is done in the following section.

7.3 Determination of parameters

In this section, two ways to inter-relate these parameters are proposed called scheme A and scheme B. This results in four parameters which can be used to determine the rest. The length scale is determined by a parameter k (or k_1 if scheme B is used). The time scale is determined by a parameter C. γ_{max} and R_a are the other free parameters. Thus, by using these four parameters, with the respective scheme, all the other parameters can be readily obtained. The following sub-sections describe the schemes in detail.

7.3.1 Scheme A

The definition of Re is given by,

$$Re = \frac{2RU}{\nu}. \tag{7.1}$$

In general, a vortex blob cannot penetrate the body because that would introduce vorticity into the solid body. In traditional RVM implementations this is not enforced. In the present work this condition is enforced resulting in the following condition,

$$\delta \leq h_{num}. \tag{7.2}$$

The numerical layer height can chosen to be some multiple of the diffusion length scale ($\sqrt{2\nu\Delta t}$), therefore,

$$h_{num} = k\sqrt{2\nu\Delta t}. \tag{7.3}$$

The above choice is usually made in implementations of the RVM. The length of a viscous box side (or the length of the sheet) is given by $\lambda = 2\pi R/N$. The velocity due to a blob at a distance equal to the core radius (along with its image vortex) must match the velocity due to a sheet. Therefore,

$$\gamma = \frac{2\Gamma}{2\pi\delta}, \tag{7.4}$$

where γ is the vortex sheet intensity and Γ is the strength of the vortex blob. The factor 2 in the numerator of the right-hand-side arises due to the image vortex. Each sheet has the same length, λ, as the viscous box on the body. Hence, to conserve circulation, $\Gamma = \gamma\lambda$. Therefore, from equation (7.4) it is seen that,

$$\delta = \frac{\lambda}{\pi}. \tag{7.5}$$

Using equation (7.5), equation (7.2) now becomes,

$$\begin{aligned} \frac{2\pi R}{N\pi} &\leq k\sqrt{2\nu\Delta t} \\ \frac{2R}{N} &\leq k\sqrt{2\nu\Delta t}. \end{aligned} \tag{7.6}$$

Puckett (1991) suggests that a condition on Δt be imposed such that the distance traveled by a sheet in one time step is less than the length of the sheet. This can be expressed as,

$$MU\Delta t = C\lambda$$

where M is a upper bound on the velocity in the flow and C is a user determined constant. Therefore,

$$\Delta t = \frac{C2\pi R}{MUN} \tag{7.7}$$

Using equation (7.6) and the definition of the Reynolds number, equation (7.1), the following is obtained,

$$\begin{aligned} \frac{2R}{N} &\leq k\sqrt{2\frac{2RU}{Re}\frac{2C\pi R}{MUN}}, \\ N &\geq \frac{MRe}{2\pi k^2 C}. \end{aligned} \tag{7.8}$$

Re is known and M can be estimated. This means that the number of viscous boxes, N, is directly proportional to the Reynolds number. This is interesting to note since it is normally argued (Leonard, 1980) that the number of particles released should be proportional to $Re^{1/2}$. In the present case $N \propto Re$ because there is a strong restriction on Δt as given in equation (7.7) and the numerical layer height is in turn tied to Δt.

7.3.2 Scheme B

Scheme B differs from A in relating h_{num} to the other parameters. Physically, the numerical sheet layer is the same as the boundary layer. Therefore, it is natural to relate h_{num} to the boundary layer height. From boundary layer theory, h_{num} can be expressed as,

$$h_{num} = k_1 \frac{L}{\sqrt{Re}}, \tag{7.9}$$

where L is a length scale and in this case equals $2\pi R$. k_1 is a numerical factor that can be chosen. k_1 is similar to k used in scheme A. Using the above relation and equations (7.2) and (7.5) results in the following,

$$\begin{aligned} \frac{L}{\pi N} &\leq k_1 \frac{L}{\sqrt{Re}} \\ N &\geq \frac{\sqrt{Re}}{\pi k_1} \end{aligned} \tag{7.10}$$

From equation (7.7) and (7.10) it is clear that,

$$\begin{aligned} \Delta t &= \frac{CL}{MUN} \\ &\leq \frac{CL\pi k_1}{MU\sqrt{Re}} \end{aligned}$$

The above relations show that in scheme B, N is independent of Δt and Δt is dependent on C and k_1. This approach in general will require fewer particles. The other advantage with this scheme is that C can be changed without changing the number of panels N. However, it is noted that most high-resolution vortex method implementations tend to limit the core radius of the blob as done in scheme A. Shiels (1998), for example, explicitly mentions that the core radius can be chosen as a multiple of the viscous diffusion length scale.

There is some flexibility in choosing k, k_1 and C. Clearly, once N is known from either equation (7.8) or (7.10) the other parameters can be found. The only other free parameters are γ_{max} and R_a, which are also varied. R_m is not considered here since R_a is the more sensitive parameter. This was seen in the numerical experiments performed in section 5.4.1. Hence, the variation of these parameters (k (or k_1), C, γ_{max} and R_a) are studied. Initially scheme A is used. Subsequently, scheme B is used and the results of both are compared. It is to be noted that the results for the variation in γ_{max} and R_a are independent of the scheme chosen.

7.4 Primary diagnostics

To study the results obtained, the drag coefficient, C_d, versus time is used as the primary diagnostic quantity. It is well known (Koumoutsakos and Leonard, 1995; Shiels, 1998; Ploumhans and Winckelmans, 2000) in the vortex methods community that the drag history is a good diagnostic for the accuracy of such a simulation. Ploumhans and Winckelmans (2000) also use the production of circulation on the upper surface of the cylinder as a diagnostic. However, in this chapter only the drag history is used.

The drag coefficient is obtained using the hydrodynamic impulse. The resulting curve is smoothed using piecewise polynomials as described in section B.1.2.

The noise level in the force history is also obtained by smoothing the computed load using an appropriate scheme and finding the difference between the smoothed and noisy curves using equation (B.5). It is to be noted that the initial singular part of the drag curve is ignored for this computation since it introduces spurious errors.

Since the number of parameters and possible combinations of them is large, the results are studied by comparing the drag coefficient, C_d, with the results of Koumoutsakos and Leonard (1995) (referred to hereinafter as KL) for the same Reynolds number. The data from their curves have been extracted. The process of extracting the data introduces errors and simple estimates suggest that up to 1% error is possible in this process. The errors are significant near $t = 0$ due to the singular behavior of the curve. Any small errors in the extraction of the correct value of t near this singular region can prove expensive. Thus, the forces are compared only when $T = Ut/R > 0.5$ when the data is close to a minimum and is well behaved. The comparison of the present data with that of KL is done by first interpolating the values from KL to the values of t for the present computations. The difference in the solutions is then found using equation (B.5), which is an L^2 norm. Thus, using this approach it is possible to easily compare the present results with those of KL. It is to be noted that the results from KL are only available up to $T = 6$. Therefore, the error is computed for $0.5 \leq T \leq 6$.

Initially, the computations are performed with a single random number seed. Subsequently, several computations are made with different random number seeds and the results of these are ensemble averaged. In these cases the standard deviation, σ, of the various runs are computed. If the expected value of a measured quantity is $\bar{L} = \sum_1^N L_i/N$, where the individual values are L_i, then the standard deviation, σ, for N values is given as

$$\sigma = \sqrt{\frac{\sum_1^N (L_i - \bar{L})^2}{N-1}}. \tag{7.11}$$

Vorticity contours are also plotted but in the present case they are only used to provide a qualitative feel for the results.

7.5 Parametric study with scheme A

For all the cases considered in this section, sheet release style 1 (section 3.4.1) is used. All the vortex sheets have the same strength. Thus no merging of vortex sheets or blobs is possible or necessary. The annihilation is only performed for the sheets.

7.5.1 Variation of C

Given that $Re = 3000$, the values of U and R are assumed to be unity. From equation (7.1), $\nu = 1/1500$. For a circular cylinder, $M = 2$.

In order to reduce C, k is held fixed and Δt is reduced gradually. This approach limits the motion of the sheet or blob relative to the size of the sheet. It also reduces the numerical layer height. This makes the computations more accurate. If it is assumed that $\delta = h_{num}$, from equation (7.6) and (7.7) it can be seen that,

$$N = \frac{2R}{k\sqrt{2\nu\Delta t}},$$

$$C = \frac{N\Delta t}{\pi R}.$$

Hence N increases as Δt and C are decreased. Since R and ν are known it is seen that,

$$N = \sqrt{\frac{3000}{k^2\Delta t}}.$$

This value of N is truncated to the nearest integral value and k is set to 3.

The Δt values considered are shown in Table 7.1. R_a is kept fixed at 0.25 and γ_{max} at $0.05m/s$. The accuracy of the AFMM is set to 10^{-6}. Second order integration is used. Sheet release style 1 is used. The simulations are run up to $T = Ut/R = 30$. Sheet 1 is used.

Table 7.1: Parameters chosen as C is varied for scheme A.

Case	Δt (s)	N	h_{num} (m)	C
1	0.015	150	0.01342	0.71620
2	0.01	184	0.010954	0.58569
3	0.0075	212	0.0094868	0.50611
4	0.005	260	0.007746	0.41380
5	0.00375	300	0.00670820	0.35810
6	0.0025	366	0.00547723	0.29125
7	0.001875	422	0.00474342	0.25186
8	0.00125	518	0.00387298	0.20611

For the problem considered, it is known that the total circulation in the flow should be zero. This is used as a first test for the code. When sheets are used for the simulation there is a peculiar problem. Vortex sheets only influence the flow locally whereas blobs influence the flow globally. The velocity field of the blobs cannot introduce any circulation around the body. The influence of vortex sheets is local and the circulation due to the sheet on the body is not zero. If the total strength of the sheets around a body is not zero (due to a random walk or numerical errors), these sheets will introduce a non-zero total slip on the body. In order to satisfy the no-slip condition, new sheets will be released that have a non-zero total circulation. Hence, the total circulation due to the blobs and sheets will be non-zero. There is no means for this to be corrected in the simulation. Consequently, the total circulation in the flow will remain non-zero. In order to correct this behavior a circulation equal to that of the negative of the total blob circulation is applied around the body (i.e. the panel method is solved with a net circulation). This tends to bring the circulation back to zero. Unfortunately, this approach is not easy to apply when multiple bodies are involved since it is unclear what circulation must be assumed around each body. Another problem with this is that the change in circulation due to an excess of sheets is a local effect but the scheme to stabilize it is not local at all. This is a serious disadvantage of using sheets. Nonetheless, it is important to systematically study the different approaches to satisfying this total circulation condition. It is to be noted that this issue with the vortex sheet method has been specifically mentioned in the work of Cheer (1983). The correction employed there is identical in principle to the one used in this work.

If the total circulation in the flow is known at each time instant as Γ_i, the average error in the circulation, E_Γ, is given by,

$$E_\Gamma = \frac{\sum_{i=0}^{N_t} \mid \Gamma_i \mid}{N_t}, \tag{7.12}$$

where N_t is the number of time steps.

Consider Case1 from Table 7.1, initially the circulation of the panel method is set to the strength of the blobs in the flow with a negative sign. E_Γ is computed at $T = 30$. It is found that $E_\Gamma = 0.007942$ and the maximum value of the circulation is 0.03351. When the total circulation is set to zero without any correction to account for the sheets, $E_\Gamma = 0.55651$ and the maximum value of the circulation is 1.48063. When the total circulation of the circular cylinder is set to the negative of the total strength of the sheets it is fond that $E_\Gamma = 0.75453$ and the maximum value of the circulation is 1.6775. When the total circulation of the circular cylinder is set to the negative of the total strength of the sheets and the blobs it is fond that $E_\Gamma = 0.061432$ and the maximum value of the circulation is 0.30575.

This clearly indicates that the best approach to set the total circulation to the negative of the total strength of the blobs in the flow. This is also what is suggested by Cheer (1983).

Initially, the Park and Miller random number generator with Bays-Durham shuffle was used. Details on this algorithm are available in Press *et al.* (1992). This generator has a period of about 2.1×10^9. Case8 has about 145000 particles at the end of $T = 30$, taking a total of 24000 time steps. If an average of 70000 particles is assumed throughout the computation then this computation would require the generation of about $2 \times 24000 \times 70000 = 3.36 \times 10^9$ random numbers which exceeds the period of the generator. Hence, the random number of generator of L'Ecuyer with Bays-Durham shuffle (Press *et al.*, 1992) is implemented. This generator has a period greater than 2×10^{18}. This is sufficient for all the calculations performed in the present work.

Table 7.2: Error in circulation, E_Γ and maximum circulation, Γ_{max} for different cases.

Case	E_Γ	Γ_{max}
1	0.007942	0.03351
2	0.007765	0.03415
3	0.006686	0.02815
4	0.005722	0.02537
5	0.005743	0.02409
6	0.004724	0.02490
7	0.005234	0.02456
8	0.004527	0.01820

Table 7.2 shows E_Γ and Γ_{max} for the various cases. From the computed data, the drag coefficient, C_d, is computed using the vortex momentum approach as described in section B.1.2. Piecewise polynomials are used to obtain the vortex momentum data. This is differentiated to obtain the load. For Case1, 60 points per piecewise polynomial are chosen. The polynomial order is chosen to be 6 with

an overlap of 50%.

The results for cases 1 to 4 are plotted in Fig. 7.1. As can be seen, the drag forces are considerably different after $T = 2.5$. Fig. 7.2 plots a zoomed view of the load for small times. Clearly there is much better agreement at smaller times. Similarly, Figs. 7.3 and 7.4 show the variation of C_d for cases 5 through 8. It is seen that there are small variations in the load starting from $T = 2.5$ onwards. Subsequently these differences become significant. There is sensitivity to initial condition at this Reynolds number and different random seeds tend to take different paths. This manifests in the differences beyond $T = 2.5$. Table 7.3 shows the variation of the error and the noise level for each case. These are computed as indicated in section 7.4. The noise level clearly increases as Δt drops. This is easy to explain since the load is computed using a central difference. Due to the random walk, there is an error in the numerical value of the hydrodynamic impulse computed using equation (B.1). This error due to the random walk behaves as $O(\sqrt{\Delta t})$. Therefore, differentiating the hydrodynamic impulse results in an $O(1/\sqrt{\Delta t})$ behavior in the drag coefficient. Thus reducing Δt produces more noisy C_d curves. However, the error in column 3, for Table 7.3 is harder to explain. A bi-modal behavior is seen. When cases having odd or even numbers are considered separately, there is a clear reduction in the error. However, scheme A is peculiar in that as Δt reduces, so does the spatial resolution of the problem. Additionally, due to the random nature of the diffusion, it is hard to determine if this is a result of the sensitivity to initial conditions. Only if a reasonably large number of runs are made and the ensemble considered is it possible to make any conclusion on this matter. This is explored subsequently.

7.5.2 Variation of k

The variation of k is studied next, keeping all other parameters fixed. C is fixed at 0.4, R_a at 0.25 and γ_{max} at $0.05 m/s$. The accuracy of the fast multipole method is set to 10^{-6}. The simulations are run only up to $T = 10$ because it is clear from the results of section 7.5.1 that running the simulation for longer only produces

Table 7.3: Error as compared to KL and noise level as Δt is varied for scheme A.

Case	Δt	Error	Noise Level
1	0.015000	0.09541	0.29523
2	0.010000	0.06611	0.38207
3	0.007500	0.09508	0.46202
4	0.005000	0.05303	0.56456
5	0.003750	0.07328	0.65465
6	0.002500	0.05472	0.85738
7	0.001875	0.08877	0.92061
8	0.001250	0.04721	1.19986

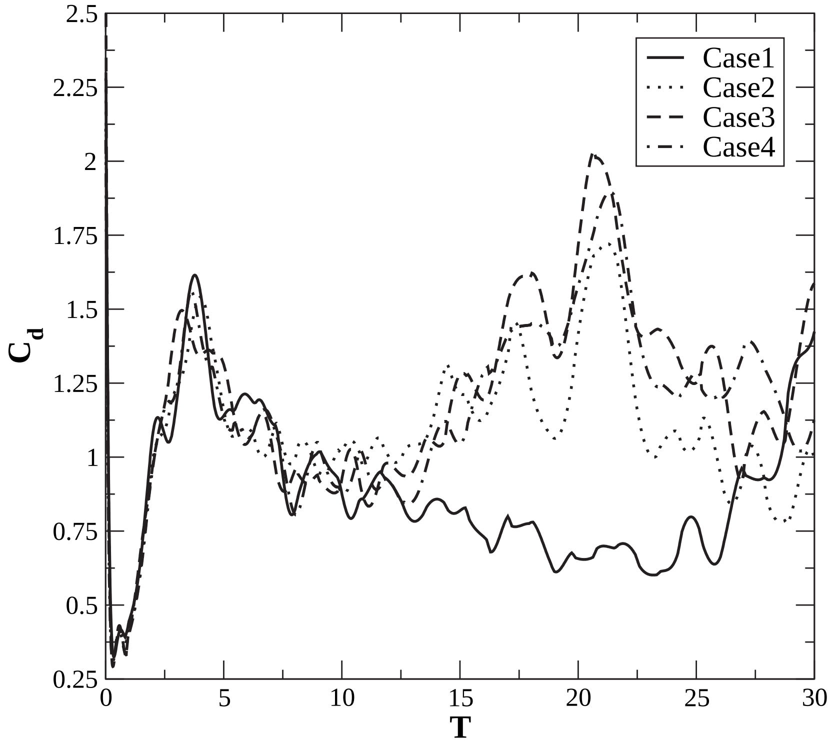

Figure 7.1: C_d versus T for cases 1 to 4.

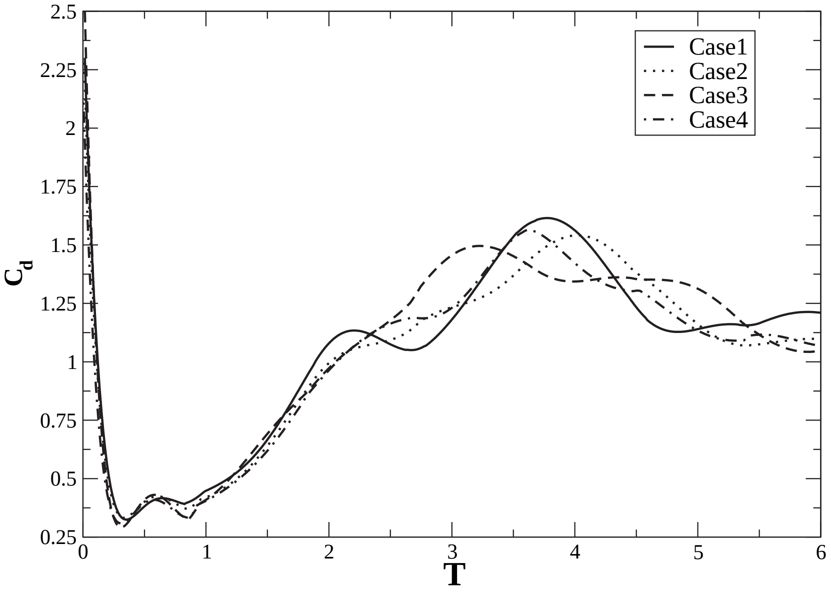

Figure 7.2: Zoomed view of C_d versus T for cases 1 to 4.

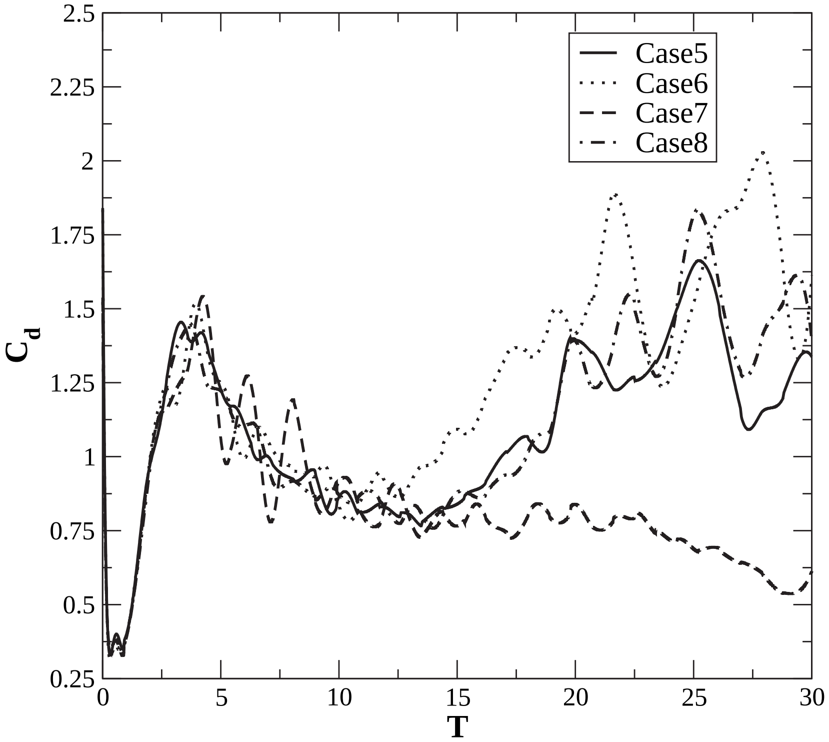

Figure 7.3: C_d versus T for cases 5 to 8.

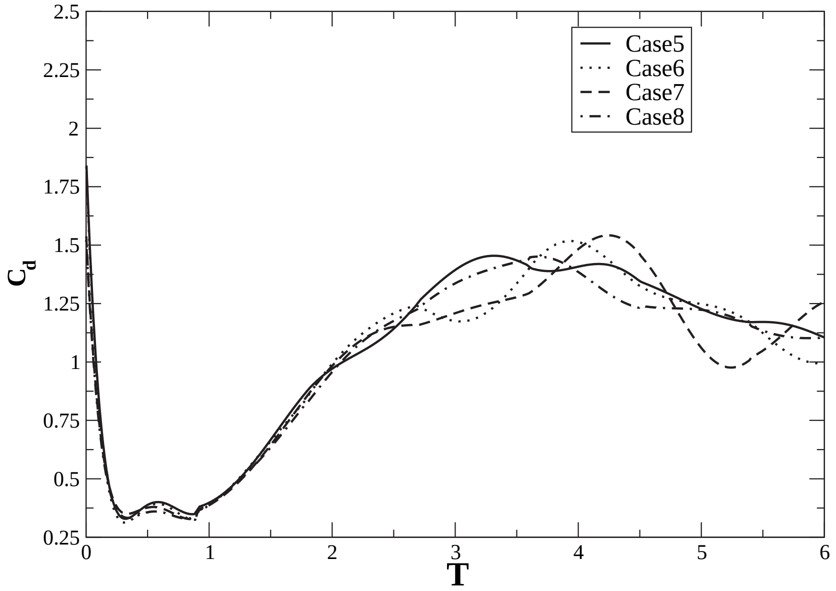

Figure 7.4: Zoomed view of C_d versus T for cases 5 to 8.

erratically different results.

From equation (7.8) and noting that $M = 2$ for the flow past a circular cylinder it is seen that,

$$N = \frac{Re}{k^2 \pi C}.$$

In the present case, $Re = 3000$. Given N,

$$\Delta t = \frac{\pi C}{N}.$$

The cases presented in Table 7.4 are considered. The value of C is approximately 0.4. N is found by rounding the value obtained using $C = 0.4$ to the nearest even integer.

It is obvious that as k increases, the number of blobs and sheets used in the simulation will reduce. Thus the resolution of the simulation drops as k increases.

Table 7.5 shows the variation of the number of blobs and sheets as the value of k increases. The variation of the number of blobs is plotted in Fig. 7.5. As can be seen the variation is not linear. Table 7.5 also shows the L^2 error in the drag curve as compared to KL and also the noise levels. The results clearly indicate that as k reduces the error in the solution drops significantly. The increase in the

Table 7.4: Number of panels, N and Δt for different k values given that $C \approx 0.4$.

k	N	Δt	h_{num}
1.5	1062	0.00118435	0.00188496
2.0	596	0.00210552	0.00335103
3.0	266	0.00473741	0.00753982
4.0	150	0.00842206	0.01340413
5.0	96	0.01315947	0.02094395
6.0	66	0.01894964	0.03015929
7.0	50	0.02579257	0.04105014
8.0	38	0.03368825	0.05361651

Table 7.5: Number of blobs and sheets at $T \approx 10$ for various values of k. Also shown is the error as compared to KL and the noise level in the curves.

k	N_{blob}	N_{sheet}	Error	Noise level
1.5	141827	1227	0.03606	1.03058
2.0	72618	994	0.04220	0.81930
3.0	30425	976	0.05737	0.61076
4.0	16541	915	0.10574	0.50551
5.0	10523	751	0.15057	0.48245
6.0	7077	555	0.24447	0.45836
7.0	5222	508	0.32075	0.43205
8.0	3822	426	0.29403	0.41546

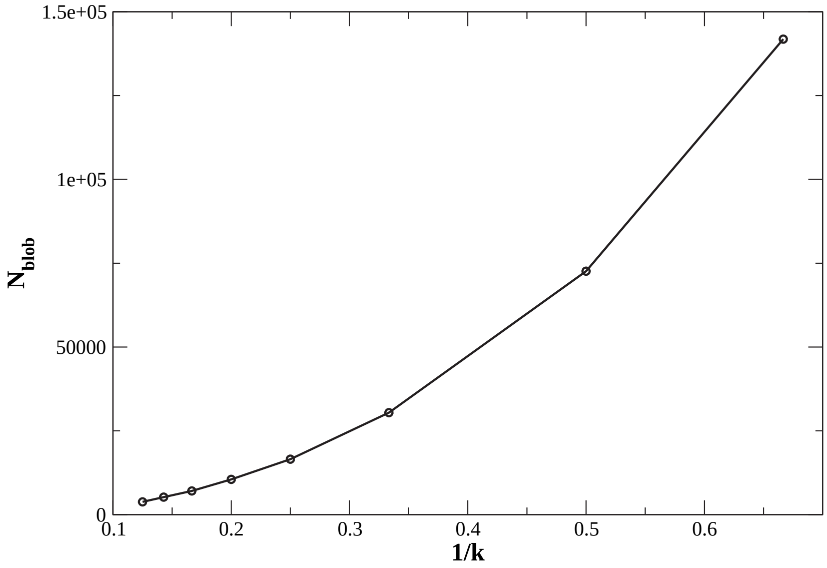

Figure 7.5: Number of blobs versus $1/k$ at the end of $T = 10$.

noise levels is only due to the reduction in Δt.

Figs. 7.6 and 7.7 plot the variation C_d versus T as k is changed. The curves are smoothed using running averages because only the trends are of interest here. The plots for k between 1 and 4 were generated using the same averaging parameters while the plots for $k = 5$ and 7 were generated with a different averaging window since the number of data points in these cases is much less. As can be seen from the figures, using values of k up to 4 are reasonable with 2-3 being the best compromise whereas using $k = 4$ produces oscillatory loads after 4 seconds. When k is greater than 3 the results are poor and show spurious oscillations. From table 7.5, it is seen that the $k = 3$ case is the best compromise, with few number of particles and reasonably accurate results.

Figs. 7.8, and 7.9, plot the vorticity field, ω, at $T \approx 5$ with k as 1.5 and 3. As can be seen, the vorticity distribution for the $k = 1.5$ case is better than the case where $k = 3$. However, this case requires a very large number of particles.

7.5.3 Variation of γ_{max}

To study the effect of the variation of γ_{max}, the following values are chosen. $C \approx 0.4$, $k = 3$ and $R_a = 0.25$. γ_{max} is varied from 0.00625 to 0.4.

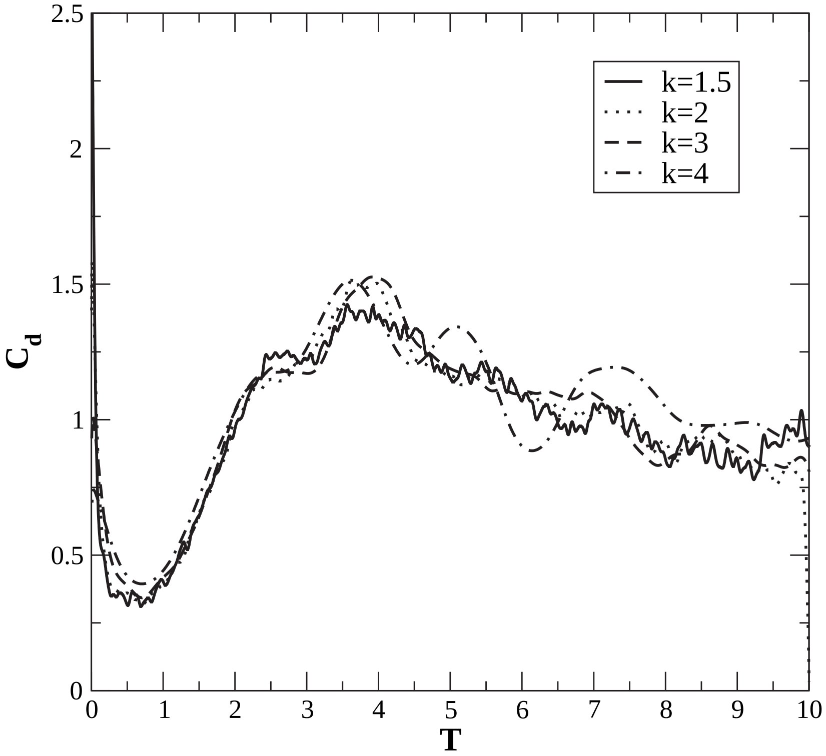

Figure 7.6: C_d versus T as k varies between 1.5 to 4.

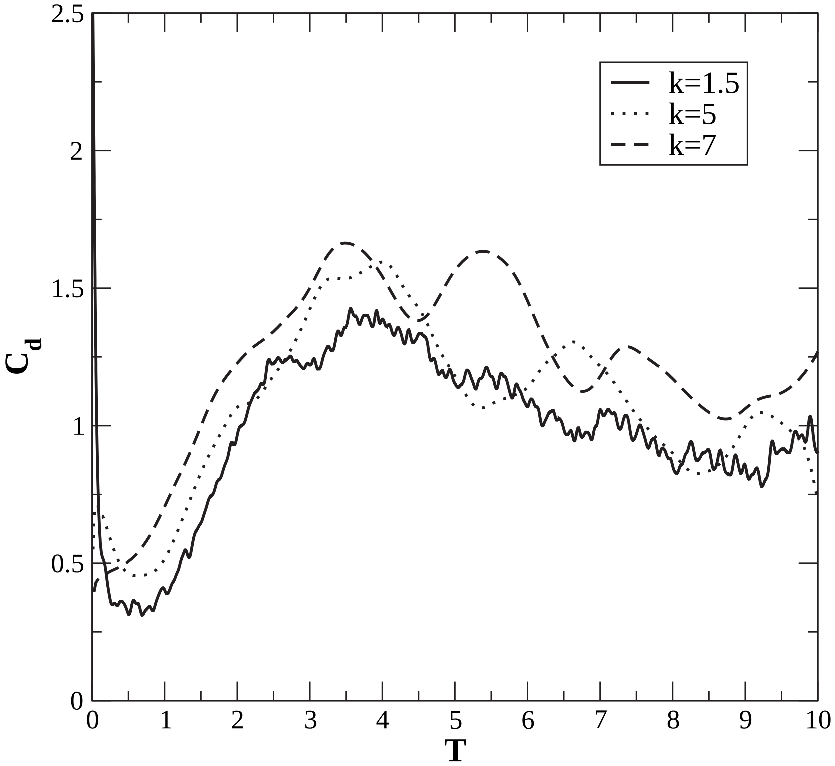

Figure 7.7: C_d versus T as k varies between 5 to 7.

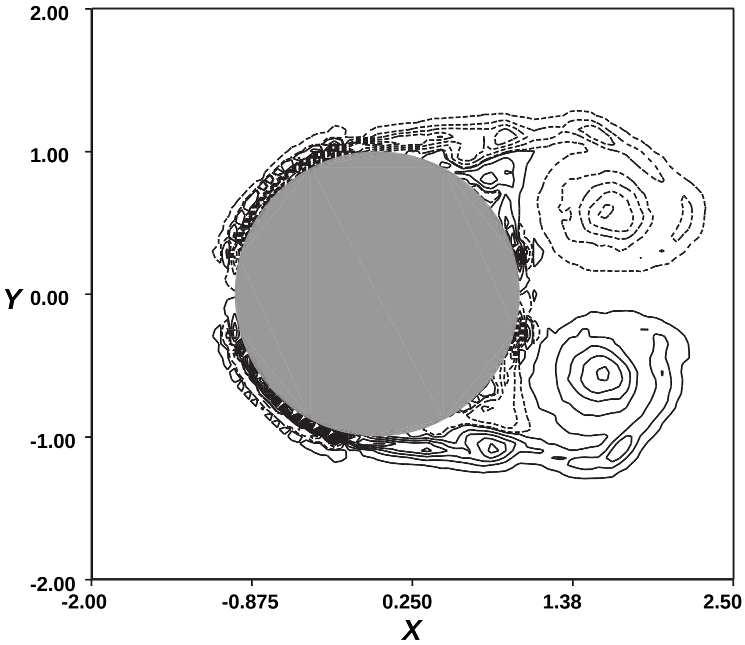

Figure 7.8: Vorticity contours at $T \approx 5$ with $k = 1.5$.

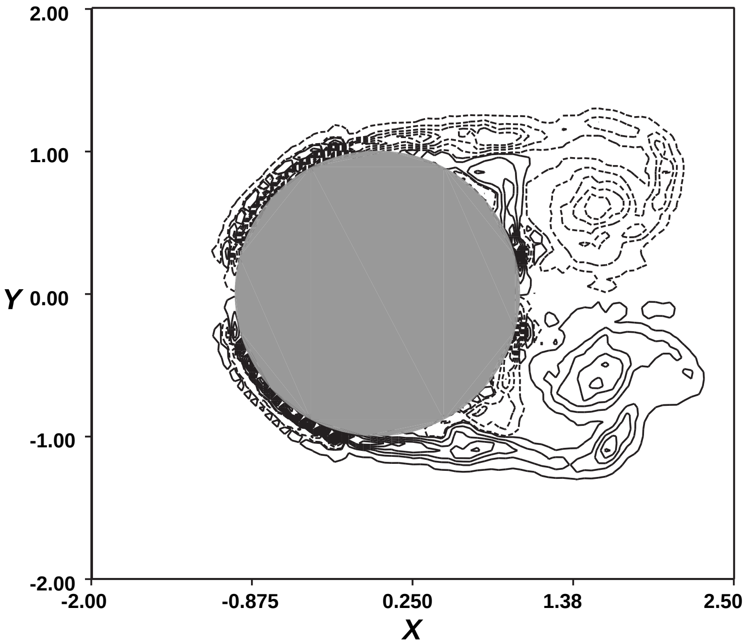

Figure 7.9: Vorticity contours at $T \approx 5$ with $k = 3$.

Table 7.6: Number of blobs and sheets at $T \approx 10$ for various values of γ_{max}. Also shown are the error and noise level in C_d versus T.

γ_{max}	N_{blob}	N_{sheet}	Error	Noise level
0.00625	235646	7440	0.04715	0.13266
0.01250	118917	3726	0.04653	0.21589
0.02500	59587	1991	0.08181	0.35279
0.05000	30425	976	0.05839	0.61045
0.10000	15617	510	0.10192	1.08369
0.20000	8235	268	0.09549	1.83416
0.40000	4227	131	0.19750	2.96449

Table 7.6 shows the variation of the number of blobs and sheets as γ_{max} is increased. As can be expected, the variation is almost linear. The noise levels in the load and the error as compared to KL are are also presented in Table 7.6. It is clear that reducing γ_{max} reduces the noise levels and the errors. The reduction in noise is almost linear. A decreasing trend in the errors is also observed.

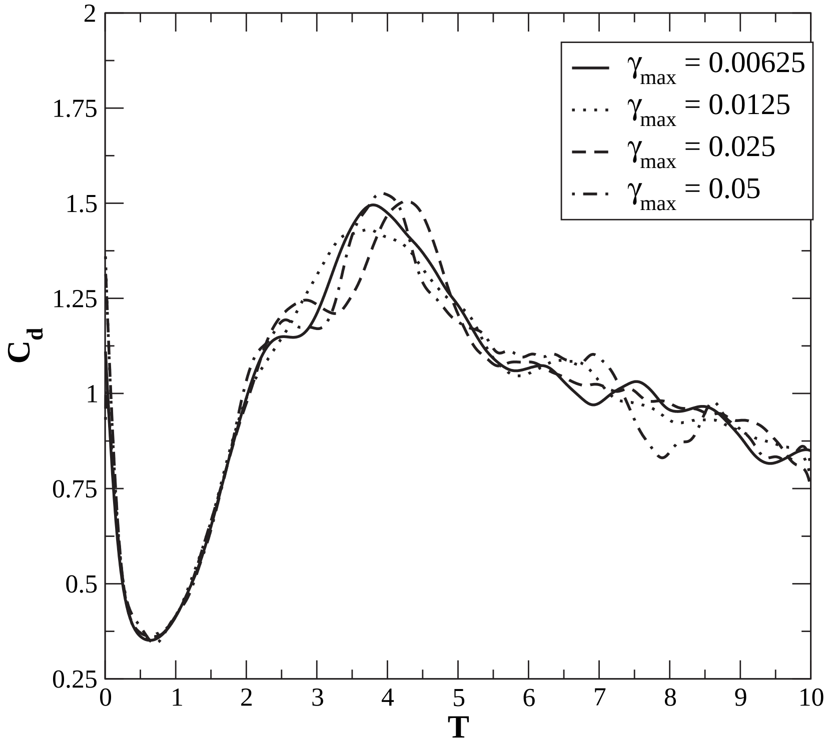

Figure 7.10: Smoothed C_d versus T as γ_{max} is changed.

Figs. 7.10 and 7.11 plot the variation of C_d versus T (processed using a running average with a 15 point window) as γ_{max} is varied. Up to a γ_{max} of 0.05 it can be observed that the initial behavior is picked up very well by all the cases. However,

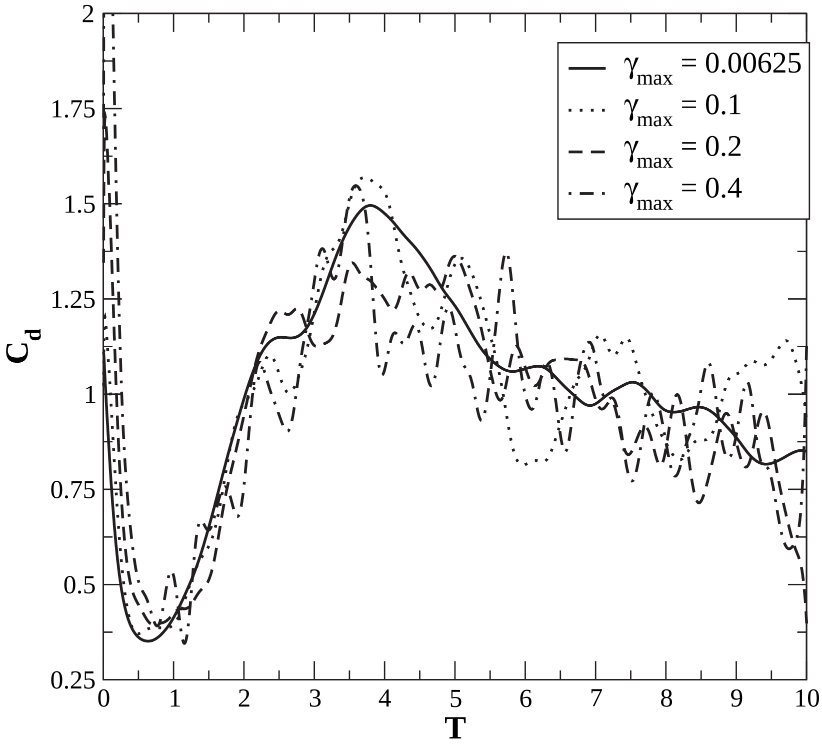

Figure 7.11: Smoothed C_d versus T as γ_{max} is changed.

for $T > 3$, there are small differences. It is to be noted that the results are fairly close to each other. Once γ_{max} increases beyond 0.05 there are significant differences.

Figs. 7.12 and 7.13 plot the vorticity field for the $\gamma_{max} = 0.00625$ and 0.0125 cases. The plot for the γ_{max}=0.05 case is seen in Fig. 7.9. The plots in Fig. 7.12 and 7.13 clearly show the high resolution achievable with the RVM.

7.5.4 Variation of R_a

To study the variation of R_a the parameters are chosen as follows. $C \approx 0.4$ and $k = 3$ as used in the previous section. $\gamma_{max} = 0.0125$ and R_a is varied from 0.0 to 0.4. When $R_a = 0.0$ there is no annihilation performed. In these cases only the vortex sheets are annihilated.

Table 7.7 shows the variation of the number of blobs and sheets as the parameter R_a is varied. $R_a\lambda$ is the radius of the circle inside which sheets are considered for annihilation. The length of a sheet is λ. As can be seen, when there is no anni-

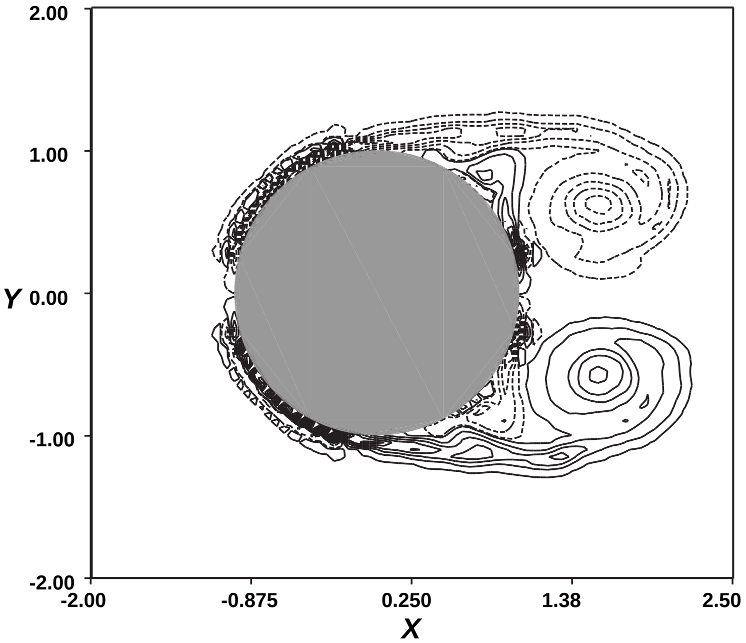

Figure 7.12: Vorticity contours at $T \approx 5$ with $\gamma_{max} = 0.00625$.

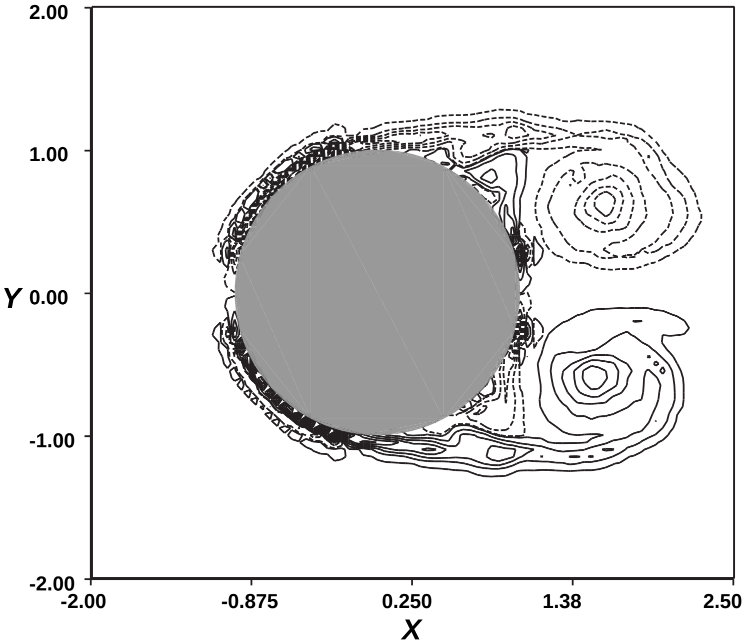

Figure 7.13: Vorticity contours at $T \approx 5$ with $\gamma_{max} = 0.0125$.

Table 7.7: Number of blobs and sheets at $T = 6$ for various values of R_a. Also shown are the error and noise level in the curve for C_d.

R_a	N_{blob}	N_{sheet}	Error	Noise level
0.00000	1360403	93094	0.09236	0.54561
0.00625	320503	18919	0.04988	0.31804
0.01250	168885	9347	0.05591	0.26440
0.02500	111465	5739	0.04941	0.23737
0.05000	93587	4579	0.05588	0.22007
0.10000	86702	4284	0.04154	0.21936
0.15000	85129	3811	0.03809	0.22105
0.20000	84265	4020	0.03884	0.21396
0.25000	83856	3929	0.04653	0.21589
0.30000	83111	4024	0.04200	0.20756
0.35000	82985	4207	0.04495	0.21780
0.40000	82674	4087	0.07329	0.21674

hilation, the number of particles generated is extremely large. Even the smallest value for R_a significantly reduces the number of particles. When $R_a = 0$, the flow contains a large number of blobs and sheets having opposite sign and very close to each other. These do not contribute significantly to the flow simulation and make the simulation noisy. Table 7.7 clearly shows the reduction in the noise level as R_a increases. Beyond a R_a of 0.1 there is no significant reduction in the noise. These results suggest that despite an order of magnitude reduction in the number of particles (and therefore computational time), there is a significant improvement in the results.

Figs. 7.14 and 7.15 plot the smoothed (using a 15 point sliding-average) C_d versus T curves as R_a is varied. As can be seen, the results improve when R_a is increased. However increasing it by too much also introduces errors of its own. This happens because as the distance is increased, the errors introduced in the moment of the vorticity will increase. The load for the $R_a = 0$ case is only available for around 6 seconds. This is because the number of particles is extremely large at this time.

Figs. 7.16, 7.17 and 7.18 plot the vorticity field, ω, at $T \approx 5$ for $R_a = 0$, $R_a = 0.0125$ and $R_a = 0.05$ respectively. As can be seen, the $R_a = 0$ plot is very noisy. The $R_a = 0.05$ is smooth but is not symmetric whereas $R_a = 0.0125$ looks

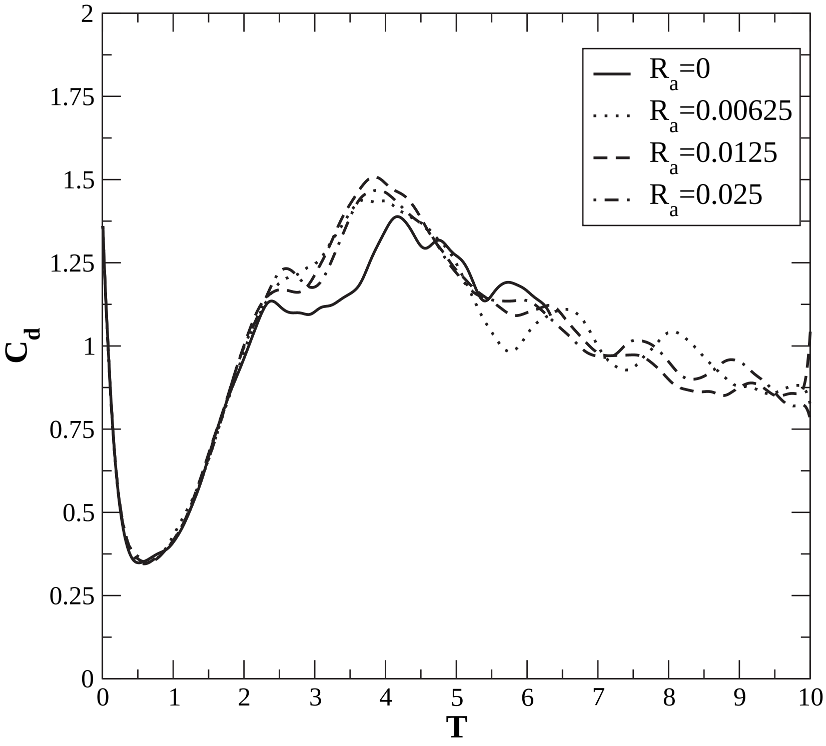

Figure 7.14: Smoothed C_d versus T as R_a is varied between 0 to 0.025.

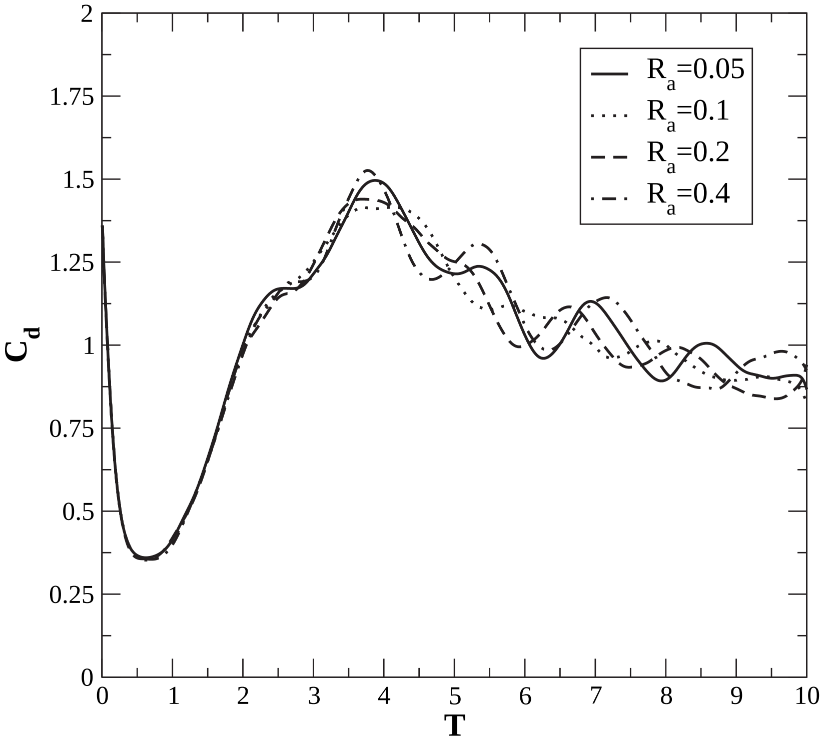

Figure 7.15: Smoothed C_d versus T as R_a is varied between 0.05 and 0.4.

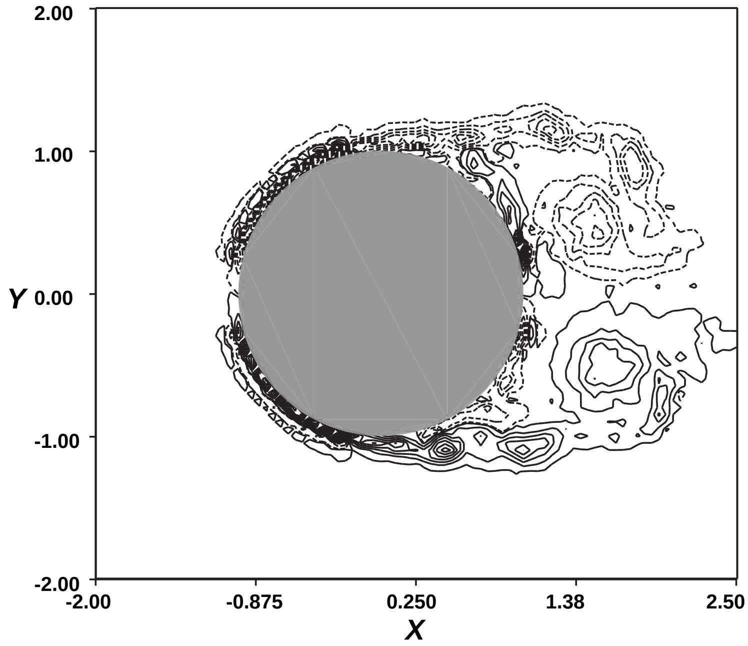

Figure 7.16: Vorticity contours at $T \approx 5$ with $R_a = 0$.

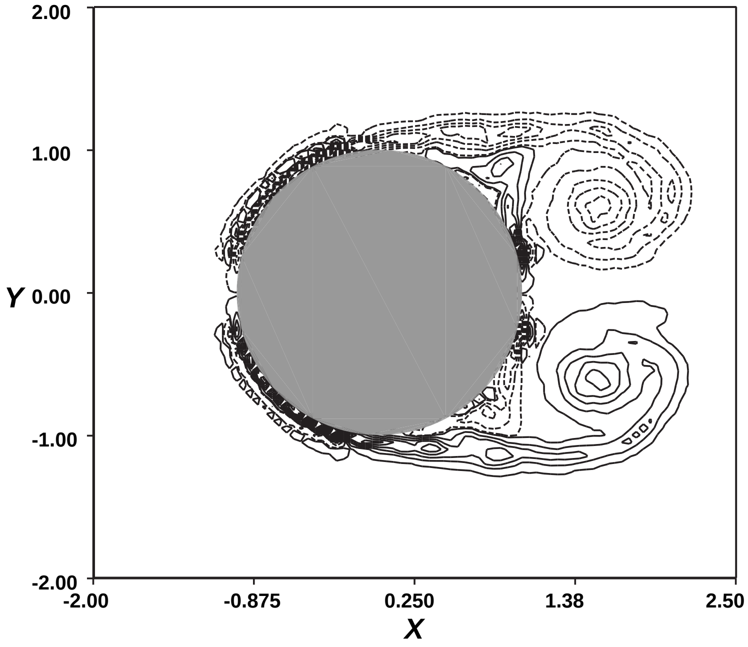

Figure 7.17: Vorticity contours at $T \approx 5$ with $R_a = 0.0125$.

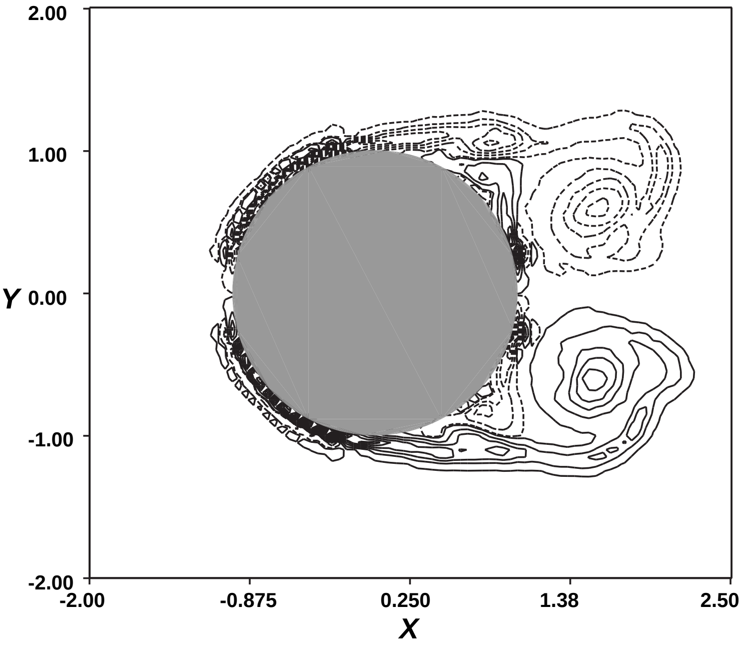

Figure 7.18: Vorticity contours at $T \approx 5$ with $R_a = 0.05$.

both smooth and symmetric. These results indicate that choosing an optimal R_a influences the computation considerably. The value of $R_a = 0.0125$ reduces the number of particles by a factor of 8 and produces far better results than the case without any annihilation.

It is clear that the annihilation of vorticity inside the numerical layer improves the quality and efficiency of the computation. It is of interest to study how annihilation outside of the numerical layer influences the computation. This is investigated subsequently. Merging of vortices of equal sign will not achieve the same effect because it effectively increases γ_{max} and thereby introduces noise. It will also increase the errors in the solution.

The results shown above clearly demonstrate that RVM is capable of producing high resolution results when a large number of particles are used and also when the parameters are carefully chosen. It is also seen that annihilation significantly improves the results in terms of quality and efficiency.

7.6 Parametric study with scheme B

Scheme B is considered in this section and the effect of the variation of C and k_1 are studied. There is no need to consider the variation of R_a and γ_{max} since in these cases, the value of C and k used in sections 7.5.3 and 7.5.4 can be easily related to a corresponding C and k_1 in scheme B. Similarly, for the case where k_1 is varied, it is possible to find the corresponding k_1 values from the values of k taken in section 7.5.2. If C is fixed as 0.4 and the values of N are taken from Table 7.4, the appropriate value of k_1 can be found as

$$k_1 = \frac{\sqrt{Re}}{\pi N}.$$

Table 7.8 shows the values of k_1 corresponding to the values of k from Table 7.4. In scheme B since N is not related to C (see equation (7.10)), once C is fixed, Δt can be found directly from equation (7.7). Hence, the values of Δt and h_{num} are slightly different from those in Table 7.4. These differences are very little and show-up only in the third or fourth decimal place. Therefore, the results of section 7.5.2 are applicable here.

Table 7.8: Values of k_1 for scheme B, given corresponding k values from scheme A in Table 7.4.

k	N	k_1
1.5	1062	0.016417
2.0	596	0.029253
3.0	266	0.065543
4.0	150	0.116230
5.0	96	0.181610
6.0	66	0.264160
7.0	50	0.348691
8.0	38	0.458804

7.6.1 Variation of C

To study the effect of variation of C, the case of $k = 3$ is considered. This results in a value of $k_1 \approx 0.065543$, $N = 266$, and $h_{num} = 0.0075188$. This corresponds to the cases for which γ_{max} and R_a were varied in sections 7.5.3 and 7.5.4. C values are considered as shown in Table 7.9. The values of Δt are chosen to match those in Table 7.1. R_a is fixed at 0.25 and γ_{max} at 0.05. Also shown are the error in the drag coefficient as compared to KL and the noise level in the curve. As is expected, the noise level increases as Δt drops. The error is large for the largest value of Δt. However, there is no clear trend seen in the reduction of the error. This is quite unlike scheme A where a bimodal behavior with a small reduction was seen as Δt was reduced. The results of table 7.9 indicate that the reason for the reduction in the error in scheme A is due to the fact that the resolution has changed. It also is clear that scheme B is a better approach to use since it is easy to separate errors in spatial resolution as against errors in the time integration of the vortex method. These results also show that reducing Δt below a point is meaningless and only produces more noisy loads.

Table 7.9: Values of Δt and corresponding C value for scheme B. Also shown are the error and noise level in the curve for C_d.

Δt	C	Error	Noise Level
0.06	5.0802	0.14107	0.07831
0.04	3.2868	0.06455	0.09151
0.03	2.5401	0.08392	0.11773
0.02	1.6934	0.05498	0.17124
0.015	1.2701	0.09558	0.22069
0.01	0.8467	0.06405	0.31137
0.0075	0.6350	0.06853	0.41911
0.005	0.4233	0.07738	0.56824
0.00375	0.3175	0.07077	0.72401
0.0025	0.2117	0.07025	0.99874
0.001875	0.1588	0.06360	1.24475
0.00125	0.1058	0.06199	1.71327

7.7 Parametric study using ensemble averaging

The results obtained up to this point were based on single runs. Many of the results were inconclusive as regards the error of the solution as compared to those of KL. As mentioned in section 3.8, using an ensemble average of various trials is a much more reliable way to determine the optimal parameters. This is explored in this section.

From the results of the previous sections it is clear that scheme B is a better approach than scheme A. Reducing k or equivalently k_1 improves the accuracy of the simulation. Similarly, reducing γ_{max} also improves accuracy. It has also been seen that sheet annihilation improves the results while reducing computational effort. Reducing Δt beyond a point is not advantageous and produces increasingly noisy results. In this section, ensemble averaging is used to further study the variation of these parameters. Additionally, several other quantities are varied and studied. These are listed below.

- Sheet1 or Sheet2 (as described in section 2.4) can be used. Thus far Sheet1 alone has been used.
- The vortex sheets can be created in three different ways as listed in section 3.4.1. These are called sheet release style 1, 2 and 3.
- The collisions due to the convective displacements can be reflected across solid surfaces as discussed in section 5.1. For larger Δt values this is a must. However, for small Δt this may be unnecessary. Thus, it is of use to study the effects of this.
- Annihilation can be performed for either or both the sheets and the blobs.
- Each sheet can induce a velocity of $-\gamma_{max}/2$ on itself. This is to ensure that the sheets velocity is similar to that of a blob at the same position. The blob will be influenced by an image blob which contributes a velocity of this amount near the wall. It is investigated if the self-induced sheet velocity is of any significance. All computations up to this point of time do not compute a self induced velocity.
- Sheet tagging can be enabled or disabled. Puckett (1989) shows that sheet tagging does not produce any improved results for the vortex sheet method.
- Different integration schemes can be used. A first order Euler or second order Runge-Kutta or fourth order Runge-Kutta method can be used.
- Strang type discretization can be used.

Clearly, these are a large number of variations. These necessitate a large number of runs. To compare the results, the error, noise level and standard deviation of the trials are used. These are computed as discussed in section 7.4.

First, the importance and advantage of ensemble averaging is discussed.

7.7.1 Ensemble averaging

Consider the case where $k_1 = 0.065543$ (or equivalently $k = 3.0$), with $\Delta t = 0.01$ ($C = 0.8467$), $N = 266$, $h_{num} = 0.0075188$, $\gamma_{max} = 0.05$ and $R_a = 0.25$. Sheet1 is used with sheet release style 1. No sheet self induced velocity is computed. Only sheets are annihilated. Sheet tagging is performed. Second order integration without Strang type discretization is used. In the following, different number of trials per ensemble are considered and the error and standard deviation (σ) are studied. The runs are made up to $T = 10$ and the standard deviation is computed for the total time. The error is computed only up to $T = 6$ since the results of KL are only available up to that time.

Table 7.10: The error, standard deviation and noise level for different number of trial runs.

Trials	Error	σ	Noise Level
1	0.08282	-	0.31683
2	0.06324	0.08055	0.21592
4	0.04726	0.06761	0.15204
8	0.04728	0.07300	0.10562
16	0.04088	0.07929	0.08415
24	0.03821	0.07994	0.06572

As shown in Table 7.10, the error decreases appreciably as the number of trials in the ensemble increases. The noise level also decreases noticeably. The standard deviation, σ is around 0.075 and does not change appreciably. This indicates that despite a fairly large standard deviation (indicating a large amount of fluctuation between runs), that the averaged results are quite close to the results of KL. This is shown in Fig. 7.19 for the drag coefficient. The ensembled results for 8 and 24 trials are plotted as compared to the results of KL. The case with 8 trials is quite close to the case using 24 trials. Thus, henceforth, unless specifically mentioned, 8 trials are used per ensemble. The parameters are studied carefully using the ensemble averages.

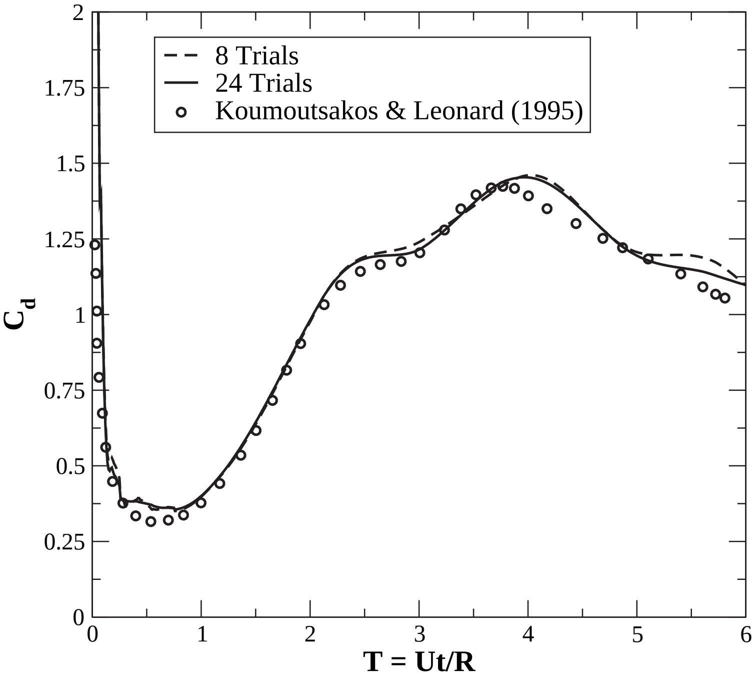

Figure 7.19: Smoothed C_d versus non-dimensional time for different ensemble averages as compared to Koumoutsakos and Leonard (1995).

7.7.2 Parametric study

Table 7.11 presents the various cases considered. The first case, case A, is identical to the one used in the previous section, 7.7.1. The subsequent cases are presented as modifications of this case. Therefore, case A2 is identical to case A with the only difference that annihilation is performed for blobs in addition to sheets. Unless otherwise mentioned, eight trials are made for each case. The standard deviation is only computed up to $T = 6$. N_b is the number of blobs and N_s is the number of sheets at $T = 6$, averaged over all the trial runs. In order to avoid problems with the initial singularity in the curve, the noise level is computed after ignoring the values up to $T = 0.5$. The merging radius used, R_m, is the same value as the annihilation radius. The results for all the cases are presented in Table 7.12.

From the results in Table 7.12 it can be seen that annihilation of blobs and sheets improves the results while reducing the number of particles (see results for A, A2, A3 and B, B1, B2 and C, C1). This is the same conclusion as made earlier in section 7.5.4. Computing the sheet self-induced velocity appears to reduce the error without changing the standard deviation (see A5). Since the standard

Table 7.11: Various combinations of parameters investigated.

Case	Parameters varied
A	Sheet1; sheet release style 1; no sheet self induced velocity; annihilation: sheets; sheet tagging; second order integration; no Strang discretization; no reflection during convection; $\Delta t = 0.01$, $\gamma_{max} = 0.05$, $k_1 = 0.06554$, $R_a = 0.25$
A1	Reflection during convection
A2	Annihilation: sheets and blobs
A3	Annihilation: only blobs
A4	Sheet2
A5	Sheet self induced velocity
A6	Sheet release style 2
A7	Sheet release style 3
B	Sheet release style 2; sheet self induced velocity; annihilation: blobs and sheets; reflection during convection; $\Delta t = 0.01$, $\gamma_{max} = 0.05$, $k_1 = 0.06554$, $R_a = 0.25$
B1	$R_a = 0.01$
B2	$R_a = 0.05$
C	Sheet 2; sheet release style 2; sheet self induced velocity; annihilation: blobs and sheets; reflection during convection; $\Delta t = 0.01$, $\gamma_{max} = 0.05$, $k_1 = 0.06554$, $R_a = 0.25$
C1	annihilation: only sheets
C2	$R_a = 0.05$
C3	$R_a = 0.05$, $\Delta t = 0.005$
C4	$R_a = 0.05$, $\gamma_{max} = 0.0125$
C5	$R_a = 0.05$, $k_1 = 0.032772$, $\Delta t = 0.005$
C6	$R_a = 0.1$, $k_1 = 0.032772$
C7	$R_a = 0.1$, $k_1 = 0.026416$
C8	Sheet release style 3; $R_a = 0.1$, $k_1 = 0.026416$
C9	$R_a = 0.1$, $k_1 = 0.021848$
D	Sheet 2; sheet release style 3; sheet self induced velocity; annihilation: blobs and sheets; reflection during convection; $\Delta t = 0.01$, $\gamma_{max} = 0.05$, $k_1 = 0.06554$, $R_a = 0.25$
D1	$R_a = 0.1$, $k_1 = 0.021848$, ; 8 trials
D2	$R_a = 0.1$, $k_1 = 0.021848$, $\gamma_{max} = 0.1$; 16 trials
D3	$R_a = 0.1$, $k_1 = 0.021848$, $\gamma_{max} = 0.2$; 32 trials
D4	$R_a = 0.1$, $k_1 = 0.021848$, $\gamma_{max} = 0.4$; 64 trials

Table 7.12: The error, standard deviation, noise level and number of parameters for the different cases presented in Table 7.11.

Case	Error	σ	Noise Level	N_b	N_s
A	0.04728	0.06499	0.10562	22154	1035
A1	0.03413	0.08142	0.11537	22039	1076
A2	0.03506	0.05190	0.11730	16066	990
A3	0.03437	0.05313	0.12074	15939	1726
A4	0.04513	0.06463	0.11101	21657	1024
A5	0.02841	0.06122	0.11443	21980	1039
A6	0.03929	0.06241	0.11789	22438	1055
A7	0.03418	0.05939	0.06279	27700	1399
B	0.02731	0.05385	0.11101	16042	1029
B1	0.06493	0.08898	0.16630	79556	9055
B2	0.05845	0.05671	0.12131	17952	1963
C	0.03907	0.05438	0.12567	16003	1037
C1	0.03871	0.06396	0.12569	21997	1025
C2	0.04845	0.06598	0.11415	17414	1497
C3	0.04241	0.06360	0.23511	16828	1385
C4	0.04793	0.04104	0.04403	65671	4252
C5	0.02972	0.04908	0.15831	37299	2225
C6	0.02722	0.04731	0.07649	34667	1424
C7	0.03525	0.03902	0.09289	43813	1693
C8	0.02708	0.03846	0.05710	53187	2540
C9	0.02562	0.03406	0.10894	57840	2053
D	0.04028	0.05573	0.06768	18392	1301
D1	0.02506	0.04580	0.07870	66379	2689
D2	0.03032	0.04958	0.08391	40760	2304
D3	0.01823	0.08110	0.09604	29002	2151
D4	0.05907	0.11087	0.13631	26204	1995

deviation is unchanged, further study is necessary to confirm this. The variation of Δt is not too critical for the ranges considered (C, C3, C5, C6). It is clearly seen (C2, C4) that reducing γ_{max} reduces the standard deviation and the noise level. However, the averaged error does not improve by much. This requires more careful analysis. It is definitively seen from the cases C5, C6, C7, C8 and C9, that reducing k_1 consistently improves the results by reducing both the error and the standard deviation, σ. Using sheet release style 2 improves the results slightly and sheet release style 3 significantly lowers the noise level (C7, C8). Sheet2 has no significant effect (see A, A4). Sheet2 is used in subsequent cases because the velocity field due to Sheet2 is smoother than that of Sheet1.

The results for cases D, D1, D2, D3 and D4 clearly show that as γ_{max} increases, the standard deviation increases. However, the use of a large number of trials does tend to give acceptable errors. This indicates that there are limits to how much parallelization is possible with the method. Increased errors are seen because the cell or mesh Reynolds number, $Re_h = \gamma_{max}\lambda/\nu$, increases with γ_{max}. Ploumhans and Winckelmans (2000) note that the mesh Reynolds number must be $O(1)$. The D4 case has $Re_h \approx 5$.

Using the results for the case D4 with 64 trials, the relationship between the number of trials, the error, standard deviation and noise level is studied. The results are shown in Table 7.13. The results and conclusions are similar to those presented in Table 7.10. The results indicate that as the number of trials increases, the error drops gradually but as would be expected, the standard deviation stays fixed. The noise level does drop significantly.

Given the results of Table 7.12, further refinements are made to case C9 and studied. Sheet 2 along with sheet release style 2 and the sheet self induced velocity is used. Annihilation of blobs and sheets are performed and reflection is performed during convection. The various parameters chosen are $\gamma_{max} = 0.05$, $R_a = 0.1$, $k_1 = 0.021848$. The results for the variation of Δt are shown in Table 7.14. A slightly different method is used to generate the random seeds while using the same random number generator. Consequently, the numbers for the case where $\Delta t = 0.01$ are different from those of case C9.

Table 7.13: The error, standard deviation and noise level for different number of trial runs for case D4.

Trials	Error	σ	Noise Level
1	0.16609	-	1.09676
2	0.17239	0.07829	0.71172
4	0.09922	0.11461	0.52690
8	0.07027	0.10710	0.35749
16	0.05676	0.11167	0.26736
32	0.06083	0.10629	0.20644
64	0.05907	0.11087	0.13631

Table 7.14: The error, standard deviation, noise level and number of parameters for the variation of Δt for case C9 in Table 7.11.

Δt	Error	σ	Noise Level	N_b	N_s
0.0025	0.02894	0.03772	0.37818	53252	1879
0.0050	0.02072	0.04026	0.21215	55408	1967
0.0100	0.03103	0.03901	0.11545	58086	2081
0.0200	0.03607	0.04561	0.05523	60515	1988
0.0400	0.03863	0.04210	0.02952	63748	1616
0.0500	0.04680	0.04143	0.02436	65353	1479
0.1000	0.20359	0.13961	0.04041	114313	2690

It is evident that the error does not drop by much once Δt is lower than 0.02. The standard deviation appears to be the same all the way up to $\Delta t = 0.05$. At this point it is possible that the discretization errors will make a difference and the use of Strang type discretization might mitigate the errors. This is investigated subsequently. The case where $\Delta t = 0.01$ has $C = 2.54$. Therefore, it is clear that values for C up to around 10 are acceptable without introducing much error. Values higher than that will introduce large errors.

Table 7.15 presents the results for the variation of k_1 for case C9 with all other parameters fixed. It is easy to see that as k_1 drops, the error and standard deviation drops. The number of particles also increases more or less proportionally. The second row considers the case where $\gamma_{max} = 0.0125$ with $k_1 = 0.131087$. As expected, the standard deviation drops considerably but the error remains large. Indicating that the error is due to a lack of spatial resolution.

k_1 has the interesting feature that it is a fraction of the estimated maximum boundary layer height. For a circular cylinder the maximum boundary layer height is of the order of $\pi R/\sqrt{Re}$. Thus, if a value of k_1=0.5 were to be chosen the numerical layer height would equal the estimated maximum boundary layer height. Thus, when $k_1 = 0.131087$ the size of the numerical layer is more than 25% of the maximum boundary layer height. This implies that the blobs are quite large. Reducing k_1 improves the spatial resolution and at 5-3% of the estimated boundary layer height, excellent results are obtained.

Table 7.15: The error, standard deviation, noise level and number of parameters for the variation of k_1 for case C9 in Table 7.11.

k_1	Error	σ	Noise Level	N_b	N_s
0.131087	0.07039	0.09107	0.18965	7982	961
γ_{max}= 0.0125	0.08697	0.05942	0.07925	31275	3501
0.065543	0.04846	0.06834	0.12081	16484	1180
0.032772	0.02706	0.05740	0.10875	34300	1580
0.021848	0.03103	0.03901	0.11545	58086	2081
0.016386	0.02513	0.03750	0.11320	82026	2349

Table 7.16 presents the results for the variation of γ_{max}. As is to be expected, the number of particles increases in proportion to the reduction in γ_{max}. The

error, standard deviation and noise levels reduce as γ_{max} reduces. However, below a point, it is clear that the returns are diminishing, requiring a reduction in k_1 and possibly Δt.

Table 7.16: The error, standard deviation, noise level and number of parameters for the variation of γ_{max} for case C9 in Table 7.11.

γ_{max}	Error	σ	Noise Level	N_b	N_s
0.0125	0.01582	0.01794	0.04382	205055	4910
0.0250	0.01974	0.02514	0.06675	106708	3002
0.0500	0.03103	0.03901	0.11545	58086	2081
0.1000	0.02344	0.06334	0.19241	34788	1744
0.2000	0.04150	0.10335	0.40570	25290	1603
0.4000	0.09681	0.12128	0.74118	24953	1341

Table 7.17 presents the results for the variation of R_a. Clearly, the annihilation improves the results and beyond a point does not improve anything significantly. However, it is important to note that even for very large values of R_a the error and standard deviation are not large. Although the annihilation introduces an error in the first moment of the vorticity it does more good than harm by reducing the noise levels and errors due to a large number of oppositely signed but overlapping particles. The reduction in the number of particles is an enormous benefit as demonstrated in section 7.5.4.

Table 7.17: The error, standard deviation, noise level and number of parameters for the variation of R_a for case C9 in Table 7.11.

R_a	Error	σ	Noise Level	N_b	N_s
0.025	0.02985	0.06361	0.15188	157011	6710
0.050	0.03288	0.04564	0.12783	79712	3568
0.100	0.03103	0.03901	0.11545	58086	2081
0.150	0.02883	0.03674	0.10592	53693	1626
0.200	0.02586	0.03371	0.10250	52097	1369
0.250	0.02587	0.03188	0.09695	51165	1312
0.300	0.02341	0.02856	0.09888	50783	1216
0.400	0.02172	0.03751	0.10235	50442	1165
0.600	0.02174	0.03437	0.10248	49902	1205
0.800	0.03150	0.03399	0.10360	49460	1159
1.000	0.02138	0.03890	0.10464	49248	1238

Given that the case C9 produces results that agree well with the results from KL, other modifications are made to the case to study their influence on the errors. This is similar to the study conducted in Tables 7.11 and 7.12. The results of this study are presented in Table 7.18. Most of the results have the same order of error and standard deviation. The results for the Strang type discretization are of considerable interest since it suggests that a five fold increase in Δt is possible without any significant reduction in the error or standard deviation. Strang discretization does increase the computational effort by about 25%. However, this is acceptable since a 5 fold reduction in computational effort results by increasing Δt. On the other hand, Euler integration produces larger noise levels and uses a $\Delta t = 0.005$ with slightly smaller standard deviation. The error in the solution is also much worse. Runge-Kutta fourth order integration does not have an appreciable effect indicating that a second order integration scheme is optimal in this case.

Table 7.18: The error, standard deviation, noise level and number of parameters for the different perturbations on case C9 in Table 7.11. The term RK4 refers to Runge-Kutta fourth order integration.

Parameters	Error	σ	Noise Level	N_b	N_s
Strang; $\Delta t = 0.05$	0.03017	0.03622	0.02176	71975	2109
Strang; $\Delta t = 0.01$	0.02535	0.03633	0.11493	60994	2826
Strang; RK4; $\Delta t = 0.05$	0.02458	0.04481	0.02629	71962	2276
Euler; $\Delta t = 0.005$	0.04021	0.03160	0.20309	55263	2021
No sheet self velocity	0.03090	0.04578	0.10351	58212	2160
Sheet1; release style 1	0.03159	0.03931	0.09186	56026	1615
Sheet1; release style 2	0.02517	0.03870	0.11220	59851	2536
Sheet1; release style 3	0.02463	0.04274	0.09030	66959	3215
Sheet2; release style 1	0.03983	0.04367	0.07887	54362	1357
Sheet2; release style 3	0.01656	0.04117	0.07850	66162	2986
No sheet tagging	0.02730	0.04627	0.11096	58109	2053

In Table 7.19 a more serious study of Strang discretization is made. The parameters listed are the differences with respect to the C9 configuration. It is clear from the results that it is feasible to use a large Δt such that C as specified in equation (7.7) is as large as 10 (for $\Delta t = 0.05$, $C = 12.7$). Thus it is possible to obtain results of similar accuracy in almost a quarter of the time.

Based on the extensive study of the various parameters conducted above, the

Table 7.19: The error, standard deviation, noise level and number of parameters for combinations of parameters when using Strang discretization.

Parameters	Error	σ	Noise Level	N_b	N_s
$\Delta t = 0.05$, $\gamma_{max} = 0.05$	0.03017	0.03622	0.02176	71975	2109
$\Delta t = 0.05$, $\gamma_{max} = 0.025$	0.02931	0.02276	0.01323	118951	3290
$\Delta t = 0.05$, $\gamma_{max} = 0.0125$	0.02049	0.01818	0.01170	217447	5283
$\Delta t = 0.10$, $\gamma_{max} = 0.05$	0.04081	0.04479	0.03223	76043	1878
$\Delta t = 0.10$, $\gamma_{max} = 0.025$	0.04684	0.02438	0.03125	125595	2970
$\Delta t = 0.10$, $\gamma_{max} = 0.0125$	0.04728	0.01463	0.02538	226330	4972

following general recommendations are made.

- Scheme B is a more flexible approach than scheme A.
- Ensemble averaging produces good results with non-optimal parameters. It also allows for a trivial parallelization of the code. By computing the standard deviation for the trials, one can estimate the error bound of the computations. From computations presented here it is clear that ensembles of 2 or 4 trials provide reasonable estimates for the deviation. The error also reduces as the number of trials increases.
- Annihilation of both sheets and blobs of opposite sign is very important and reduces the errors, standard deviation and the noise level. It also reduces the number of particles and thereby makes the computation considerably more efficient (sometimes by over an order of magnitude). Annihilation does introduce an error in the moment of the vorticity. Therefore, a large value of R_a will introduce inaccuracies. It is suggested that R_a be chosen so that $\lambda R_a < \sqrt{2\nu\Delta t}$. Where λ is the length of a vortex sheet.
- The use of a second order integration scheme is suggested since it allows for a larger Δt. It also introduces less noise in the solution than a first order scheme.
- Strang discretization is recommended since it allows for larger Δt with no appreciable reduction in accuracy.
- Δt can be chosen as given in equation (7.7) with the value of $C \leq 10$. The normally suggested choice is $C \leq 1$. However, it appears that this is unnecessary for the type of simulations performed here. Further, it is recommended that the following condition be an upper limit for Δt,

  $$\Delta t < \frac{\lambda}{\gamma_{max}}.$$

  This condition ensures that the slowest sheet does not move by more than a sheet length.

- As γ_{max} reduces, the noise, error and standard deviation reduce. However, the number of particles increases with a reduced γ_{max}. Thus it is worth reducing this parameter as much as feasible. It is also important to note that γ_{max} should be chosen such that the cell Reynolds number, $Re_h = \gamma_{max}\lambda/\nu$, be $O(1)$ or less.
- k_1 can be chosen such that the numerical layer height is less than 10% of an estimated maximum boundary layer height. As k_1 get smaller, the number of particles increase almost in inverse proportion. In the present work it is seen that values less than 0.025 produce very good results.
- Sheet2 produces smoother velocity fields and can be used instead of Sheet1. However, it is to be noted that it does not appreciably improve the accuracy or standard deviation of the results.
- Results indicate that sheet release style 3, where the boundary condition is satisfied exactly, reduces noise the most at the cost of introducing a few more particles. All of the sheet creation methods have similar characteristics. It is noted that with sheet release style 3 it is imperative to merge sheets/blobs having small strengths to reduce the number of particles.
- Adding a self induced velocity for the sheets has no appreciable effect.
- Performing sheet tagging also does not improve the results as mentioned by Puckett (1989).

The recommendations for Δt and γ_{max} do not agree with those of Puckett (1989). However, he only studied the boundary layer equations and compared his results with those of the Blasius profile. The present work considers an unsteady flow where the NS equations are valid with both vortex sheets and blobs. The numerical layer height is also very small.

7.8 Study of the ERVM

In the previous section it was seen that ensemble averaging improves the accuracy of the simulations. Based on this, a simple and new variance reduction technique was proposed in section 3.9. In this section the ERVM is used and the results obtained are studied.

Case C9 in Table 7.11 is considered with $R_a = 0.25$. The ERVM is used with the number of processors, $n_{proc} = 6$. The simulation produces a solution with an error of 0.01903 with $\sigma = 0.01262$ and the noise level at 0.11604. The standard deviation has reduced by almost a factor of three as compared to the results without the new method. This value is also less than that produced by the case where $\gamma_{max} = 0.0125$ (Table 7.16). The error itself is close to those of the cases where $\gamma_{max} = 0.025, 0.0125$. In this particular case none of the vorticity was removed based on the magnitude of its strength. The total run time was about 1.5 times that of the case where the original RVM scheme is used. Removing the vortex particles having circulation below a threshold value makes the computations almost as efficient as the original RVM.

Table 7.20 presents the results for the case where n_{sync} is varied with the number of processors, $n_{proc} = 6$. $R_a = R_m = 0.6$. Sheet2 is used along with release style 3. Strang discretization is used with $\Delta t = 0.05$ and $\gamma_{max} = 0.05$. Except for R_a, the parameters are similar to those used for Table 7.19. The results clearly show that the new method reduces the error and standard deviation. It is noted that in these tables, the average number of particles for the entire simulation (in the ensemble) are shown and not the number at the end of $T = 6$. This number is computed by the sum of the number of particles at each time step divided by the total number of time steps. This approach is used because each synchronization results in a sudden jump in the number of particles in each processor. Thus the average number of particles is a better indicator to use.

In Table 7.20, as n_{sync} reduces the error increases. This can be explained if one considers the viscous diffusion length scale and compares it with the merge/annihilation radius. Consider the value of $l_r = \sqrt{2\nu\Delta t n_{sync}}/(R_a\lambda)$. When $n_{sync} = 2$, $l_r = 2.44$ and increases to 7.73 when $n_{sync} = 20$. Clearly, if n_{sync} is small, the merge length scale is close to the diffusion length scale, thus the effect of merging negates the effect of the increased number of samples. Thus, it is recommended that n_{sync} not be chosen too small. If smaller n_{sync} values are desirable, R_m and R_a should be reduced. However, reducing R_a and R_m too much is also not a good idea since this usually results in a very large increase in the number of particles. It is therefore a good idea to choose a reasonable R_m and R_a and then choose n_{sync} such l_r is at least above 5.

Table 7.20: The error, standard deviation, noise level and average number of particles for different values of n_{sync} with n_{proc} fixed at 6.

n_{sync}	Error	σ	Noise Level	Average N_b	Average N_s
2	0.02208	0.01289	0.01242	53591	1980
5	0.01801	0.01127	0.01737	50048	1935
10	0.01800	0.01308	0.01667	47966	1887
20	0.01361	0.01222	0.01802	46808	1831

In Table 7.21 the results for the variation of n_{proc} is shown as n_{sync} is held fixed at 5. All the other parameters are the same as used for Table 7.20. As is clearly demonstrated, increasing the number of processors considerably reduces the error, standard deviation and noise level. The convergence of the results indicates that when 16 processors are used the errors are due to other sources. Δt can be reduced and so can γ_{max}. Δt in this case is very large since $C \approx 12$. However, the results presented here clearly indicate that the new method produces very accurate results with little computational overhead. It is of importance to carefully analyze and understand the new method. This is beyond the scope of the present work and will be investigated in future.

Table 7.21: The error, standard deviation, noise level and average number of particles for different values of n_{proc} with n_{sync} fixed at 5.

n_{proc}	Error	σ	Noise Level	Average N_b	Average N_s
2	0.05810	0.01788	0.02769	46025	1848
4	0.03215	0.01365	0.01864	48382	1891
8	0.01486	0.00891	0.01406	50225	1905
16	0.01330	0.00927	0.01213	50269	1902

7.9 Summary

The chapter first described the computational algorithm. The key parameters involved were listed. Two schemes, scheme A and scheme B, to tie the parameters to each other were discussed. The impulsively started flow past a circular cylinder at $Re = 3000$ was considered as the benchmark problem. The parameters discussed were systematically varied and their effects on the results were studied. The error in the solution was found by comparing the drag history with the high-resolution results of Koumoutsakos and Leonard (1995). This error was used to compare the results of the parameter variation. Scheme B was found to be more flexible and better than scheme A. Ensemble averaging was used to further refine the parameter range. The ensemble averaging reduced both the error and noise levels. Various parameters were changed and systematically studied. Over 500 different individual runs were made in the process with around 50 different cases studied. Based on this study, several recommendations for the choice of optimal parameters were suggested. Finally, the new method for variance reduction was proposed and studied. For the same parameters, the method considerably reduces the error and standard deviation without incurring a significant computational cost. When used carefully, the method produces excellent results that are comparable to the results of Koumoutsakos and Leonard (1995). In the next chapter the RVM is applied to the flow past an impulsively started cylinder at various Reynolds numbers. The results and recommendations of the present chapter are used in the study.

CHAPTER 8

RESULTS AND DISCUSSION

In the previous chapter, the numerical parameters used in the RVM were described in detail. The impulsively started flow past a circular cylinder at $Re = 3000$ was considered as a benchmark problem. The computational parameters were varied systematically. Several recommendations were made regarding the optimal choice of parameters. In this chapter, the impulsively started flow past a circular cylinder in the Reynolds number range 40–9500 is studied in considerable detail. The results are compared with available data from other computations employing vortex methods.

8.1 Flow past a circular cylinder

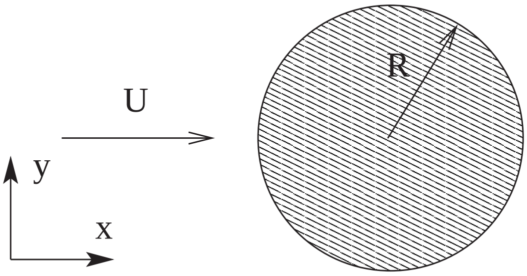

Figure 8.1: Impulsively started flow past a circular cylinder.

The problem considered involves a circular cylinder of radius R, placed in an infinite mass of initially stationary fluid. This is illustrated in Fig. 8.1. At $t = 0^+$ the body (or alternatively the fluid) is imparted a constant velocity, U. A non-dimensional time, T is defined as $T = Ut/R$. The computations aim to simulate the initial transients of this flow with high-resolution. The problem is challenging and of interest for the following reasons.

- In the vortex methods community, this is considered to be a standard benchmark problem for the NS equations. Several researchers (Cheer, 1983, 1989;

Smith and Stansby, 1988; Koumoutsakos and Leonard, 1995; Shankar, 1996; Shankar and van Dommelen, 1996*a*; Shiels, 1998; Ploumhans and Winckelmans, 2000) have studied the problem extensively.

- Capturing all the features of the flow involves considerable effort. Therefore, the problem is a good test for the code developed in the present work.
- The only significant comparisons of the results of the RVM with that of a deterministic diffusion scheme are by Koumoutsakos and Leonard (1995) and Shankar (1996). Koumoutsakos and Leonard (1995) employ the PSE (Degond and Mas-Gallic, 1989) technique and compare some of their results with those of Smith and Stansby (1988) and find the agreement to be very poor. The comparisons made by Shankar (1996) with the vorticity redistribution technique put the RVM in better light. However, there were significant differences between the results of the VRT and the RVM. Therefore, it is of interest to see if the RVM can compete with the deterministic methods.

For the present computations, various diagnostic quantities are computed as outlined in appendix B. Several of these diagnostic quantities are computed and presented in literature. Perhaps the most systematic study, with a large number of diagnostic quantities is available in the work of Koumoutsakos and Leonard (1995). They plot the drag force history, the pressure and friction drag histories, vorticity contours, streamlines, body vorticity, vorticity flux and the rate of circulation production. All of these quantities are computed in the present work and compared with the available results of other researchers. Shankar (1996) also plots the velocity distribution in the wake of the cylinder. This is also computed and compared. Thus, it is possible to make a comprehensive comparison of a modern implementation of the RVM with a vortex method using a deterministic diffusion scheme.

8.1.1 $Re = 40$

The computational parameters chosen for this case are, $\Delta t = 0.05$ ($C \approx 1.27$), $k_1 \approx 0.025$, $\gamma_{max} = 0.00625$ and $R_a = R_m = 0.25$. The body is discretized into 80 panels and viscous boxes. Second order Runge-Kutta integration is used along with Strang discretization. Sheet2 with a release style of 3 (section 3.4.1) is used. Both blobs and sheets are annihilated and merged. Eight independent trials are made and the ensemble of these is considered. At the end of $T = 15$ there are

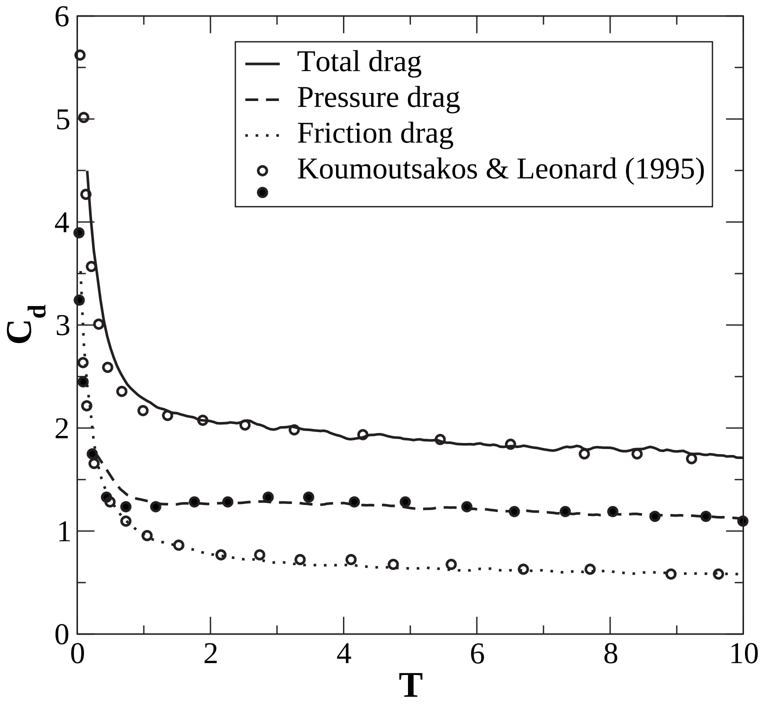

Figure 8.2: Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 40$. Solid line is the total drag force. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

around 57000 blobs and 900 sheets in the flow. A single trial takes 52 minutes to execute on a Pentium-IV 1.7Ghz machine. A larger Δt could be chosen since C is small. However, this would increase the viscous splitting error due to the high viscosity.

Fig. 8.2 plots the drag coefficient for the total, pressure and frictional forces. The curves represent the present simulations and the symbols represent those of Koumoutsakos and Leonard (1995). The total force is computed using the hydrodynamic impulse. The pressure and friction forces are computed using the approach described in section B.1.3. The curves are smoothed using a 7 point sliding average. Apart from small differences in the total and pressure forces at small times, the agreement is good.

Fig. 8.3 shows the variation of the body vorticity versus the angle measured from the trailing stagnation point of the cylinder and traversing in an anti-clockwise sense. The lines correspond to the results of the present work and the symbols to those of Koumoutsakos and Leonard (1995). The data is smoothed using a 7 point sliding average.

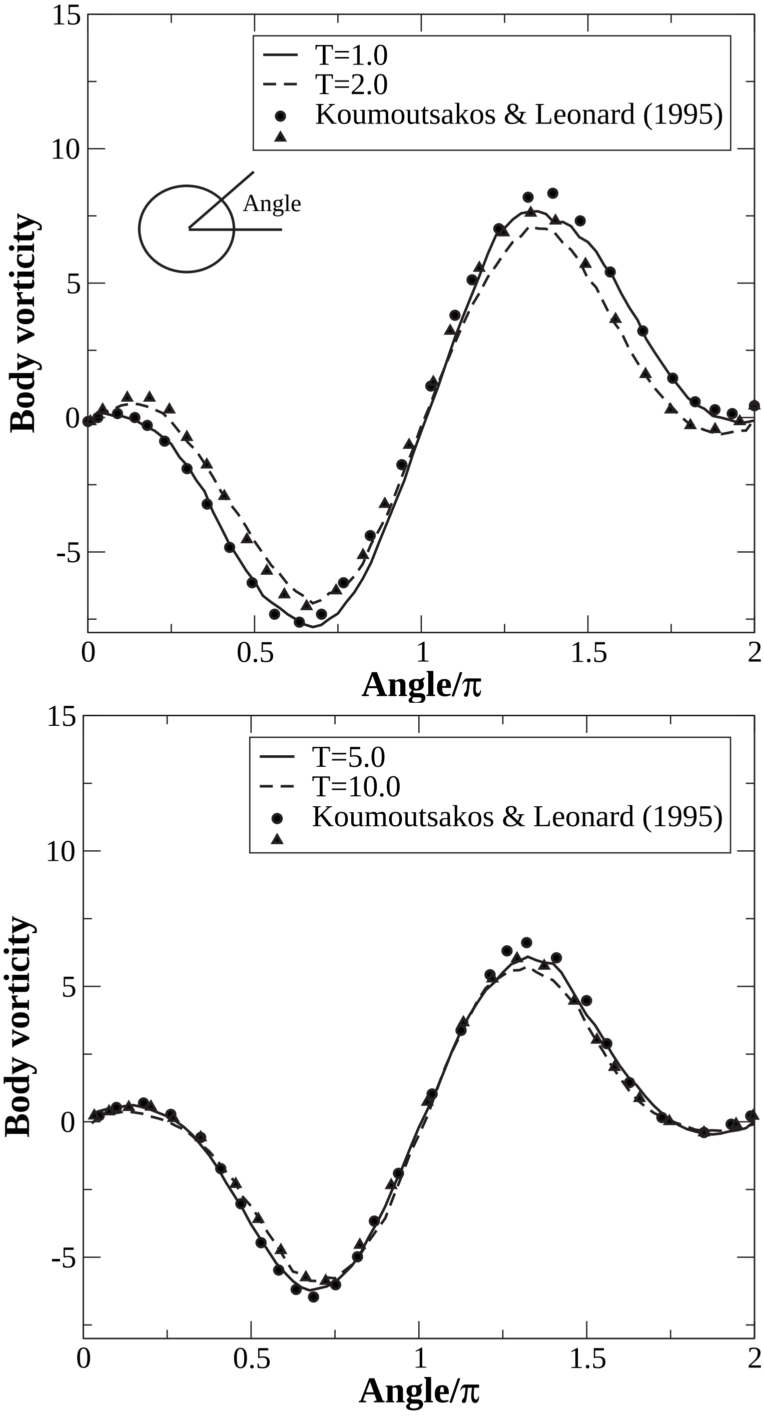

Figure 8.3: Body vorticity for impulsively translated cylinder at $Re = 40$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

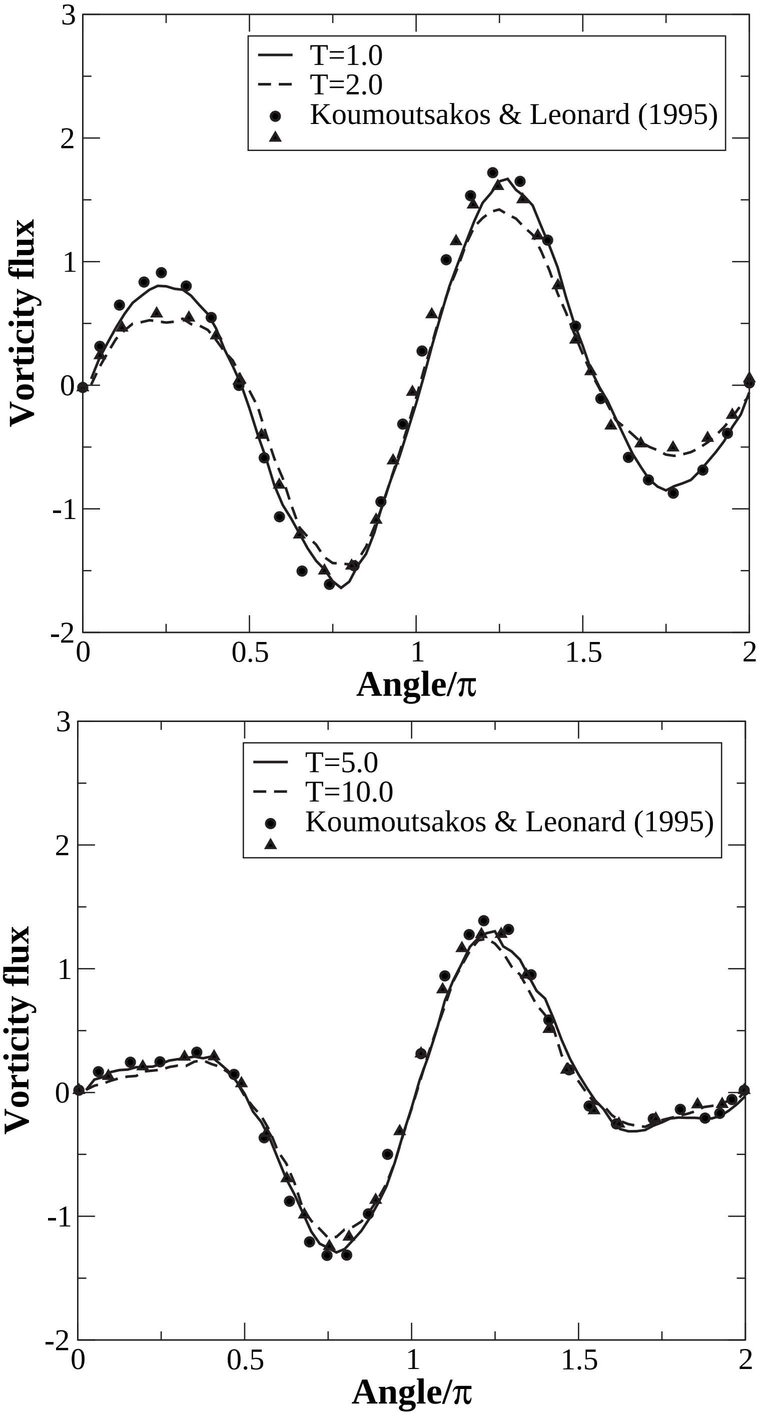

Figure 8.4: Vorticity flux for impulsively translated cylinder at $Re = 40$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

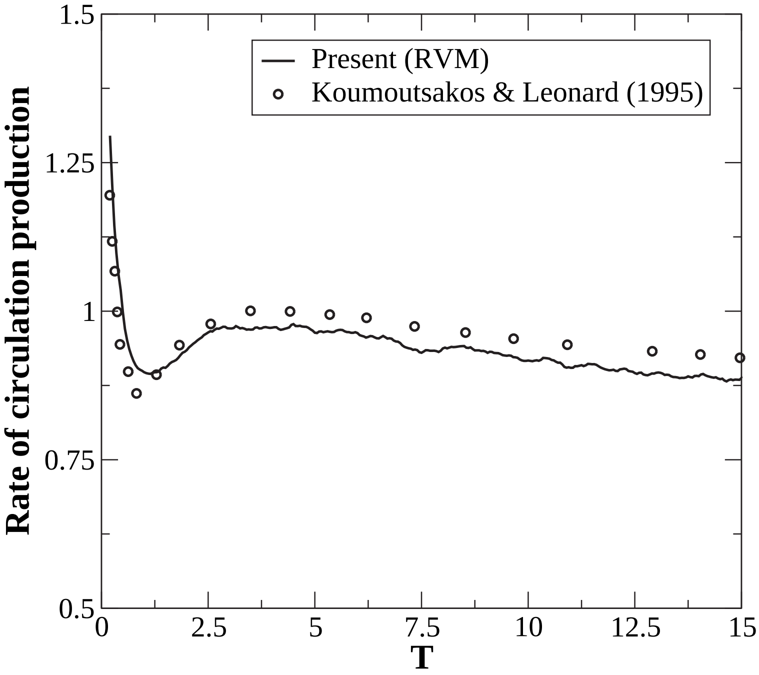

Figure 8.5: Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 40$. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

Fig. 8.4 shows the variation of the vorticity flux versus the angle measured from the rear stagnation point of the cylinder and traversing in an anti-clockwise sense. The values plotted in Koumoutsakos and Leonard (1995) are dimensional and the viscosity, diameter of the cylinder and other values chosen for the computations are not mentioned. In order to compare the present results with theirs the value of ν/D is guessed based on curves presented by them. The guess that seems to fit the data best is that of $\nu/D = 10^{-4} m/s$. Therefore this value is used consistently when comparing the vorticity flux values given by Koumoutsakos and Leonard (1995).

The overall agreement of the results is good. However, there are some discrepancies in the results for the $T = 1, 2$ cases at the peaks. The overall agreement is quite remarkable considering that the RVM works best for high Reynolds number flows.

Fig. 8.5 plots the rate of circulation production, $\frac{d\Gamma}{dt}$, from the lower half of the cylinder versus T. The values obtained by Koumoutsakos and Leonard (1995) are also shown. Their values are dimensional and as done for the vorticity flux

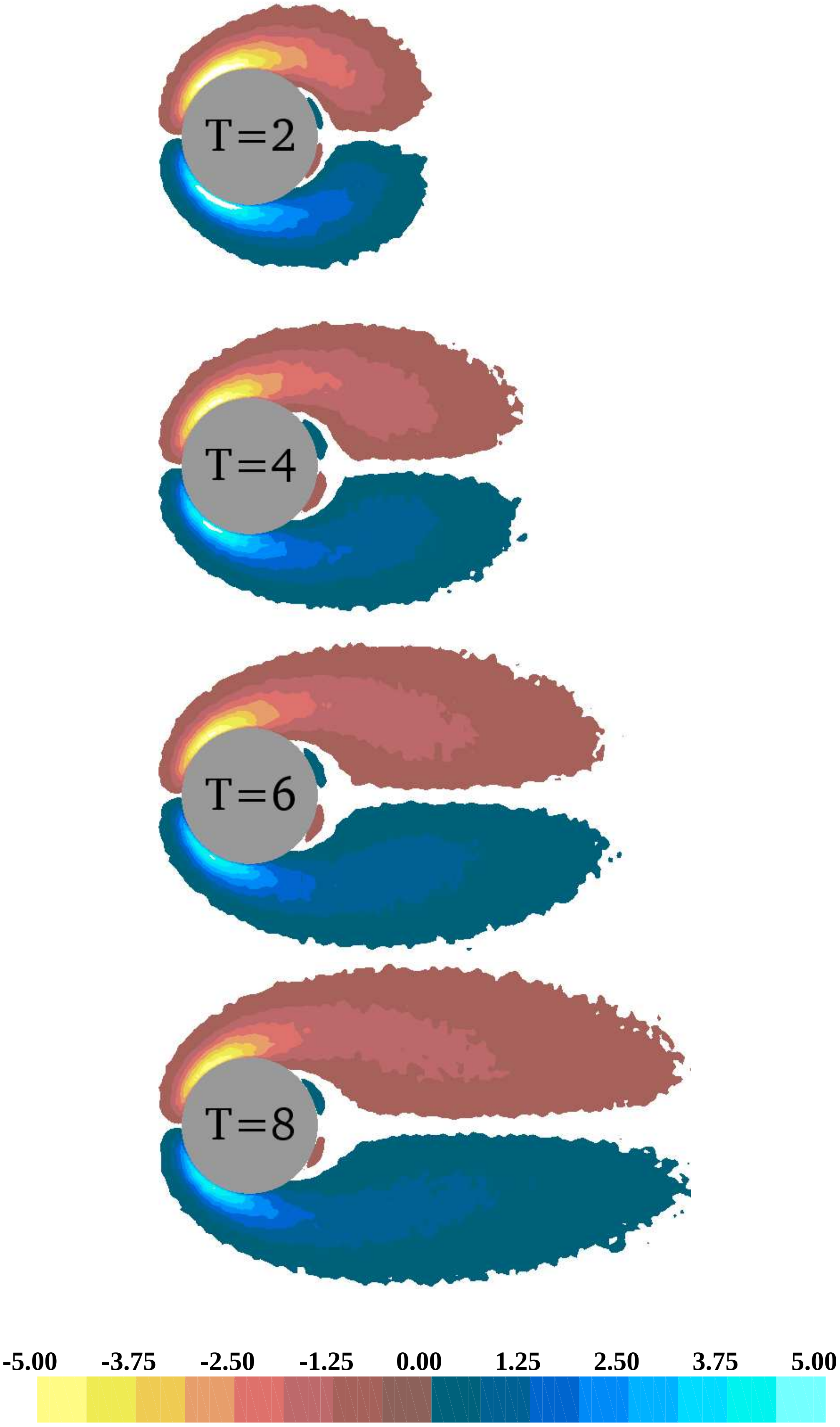

Figure 8.6: Iso-vorticity contours for an impulsively translated cylinder at $Re = 40$.

plots, their values are scaled assuming that $\nu/D = 10^{-4} m/s$. The agreement in the values is reasonable.

Fig. 8.6 plots the iso-vorticity contours at various times. The vorticity is transferred to a regular grid with spacing, $h = 0.04$. Noise is reduced by performing a Laplace smoothing operation three times.

8.1.2 $Re = 550$

The computational parameters chosen for this case are, $\Delta t = 0.05$ ($C = 4.77$), $k_1 \approx 0.025$, $\gamma_{max} = 0.00625$ and $R_a = R_m = 0.25$. The body is discretized into 300 panels and viscous boxes. Second order Runge-Kutta integration is used along with Strang discretization. Sheet2 with a release style of 3 (section 3.4.1) is used. Both blobs and sheets are annihilated and merged. Eight trials are made and the ensemble of these is considered. At the end of $T = 7$ there are around 150000 blobs and 4000 sheets in the flow. A single trial takes 75 minutes to execute on a Pentium-IV 1.7Ghz machine. On an eight machine cluster the 8 trials essentially take a total time of around 75 minutes.

Fig. 8.7 plots the drag coefficient, C_d, versus non-dimensional time, T. The solid line corresponds to the results from the present work. The curve is obtained using a piecewise 6th order polynomial fit of the data with 23 points per piecewise polynomial. The symbols correspond to the results from Koumoutsakos and Leonard (1995), Shankar (1996) and Ploumhans and Winckelmans (2000). The figure clearly shows that the present results agree well with those of Shankar (1996) and Ploumhans and Winckelmans (2000). There is a small discrepancy between the present results and those of Koumoutsakos and Leonard (1995).

It is to be noted that there is very little noise in the results and a simple five point sliding average for the curves would also produce smooth results. The noise level in the curves is computed using the equation (B.5). The noise level is found to be 0.00772 and the standard deviation, σ, of the trials is 0.01153. The same case was also run with a value of $\gamma_{max} = 0.0125$ and the results were almost identical ($\sigma = 0.01396$, noise level 0.01). For this case, the number of particles was close

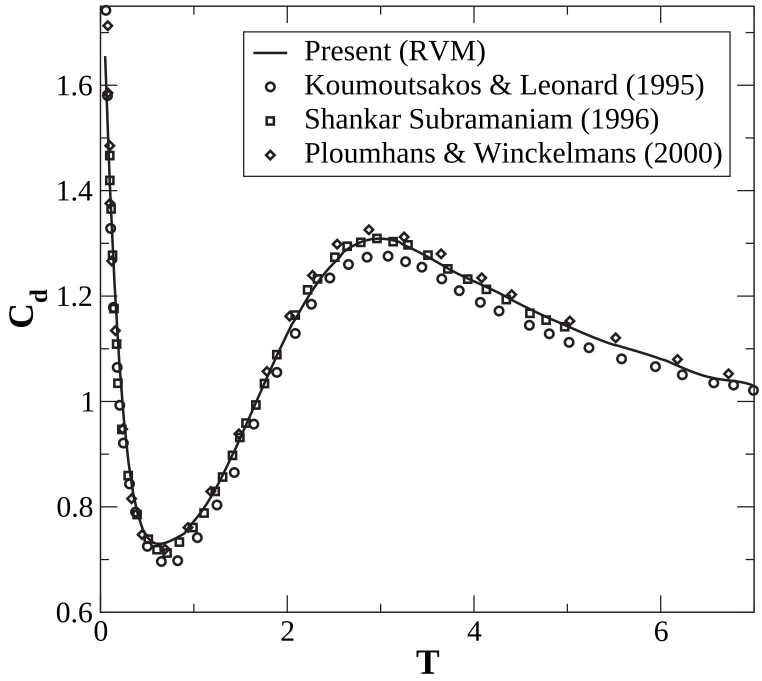

Figure 8.7: Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 550$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995), Shankar (1996) and Ploumhans and Winckelmans (2000).

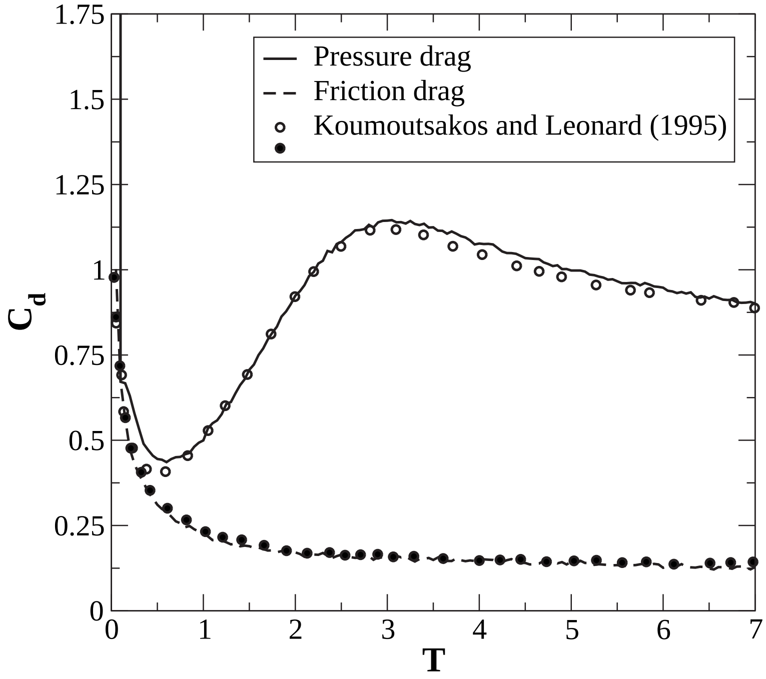

Figure 8.8: Pressure and friction drag versus T for impulsively translated cylinder at $Re = 550$. Solid line: pressure drag, dashed line: friction drag. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

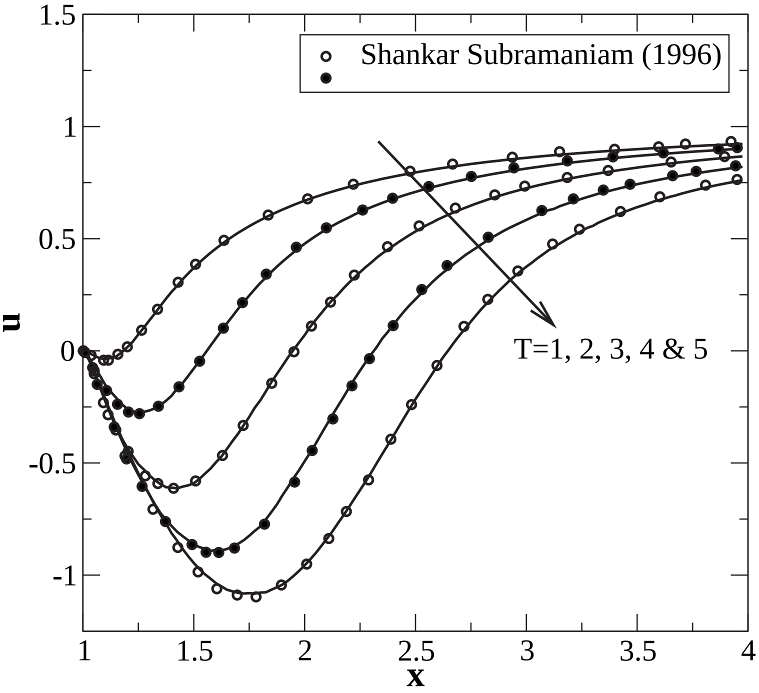

Figure 8.9: Radial velocity along the axis of symmetry on the rear side of an impulsively translated cylinder at $Re = 550$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996).

to half of the $\gamma_{max} = 0.00625$ case and the run time was also halved at around 39 minutes.

Fig. 8.8 plots the variation of the pressure and friction drag versus T. The symbols correspond to the results of Koumoutsakos and Leonard (1995). No smoothing of the data is done for the plots. As seen, the agreement is very good except for the slight discrepancy in the pressure load. This discrepancy is similar to the one seen in Fig. 8.7.

Fig. 8.9 plots the variation of the radial velocity along the axis of symmetry on the rear side of the cylinder. The symbols correspond to the results of Shankar (1996). The agreement is almost total. In the RVM, the velocity field obtained is fairly smooth. Therefore, this agreement with the results of Shankar (1996) is not surprising.

Fig. 8.10 shows the variation of the body vorticity versus the angle measured from the rear stagnation point of the cylinder and traversing in an anti-clockwise sense. The lines correspond to the results of the present work and the symbols

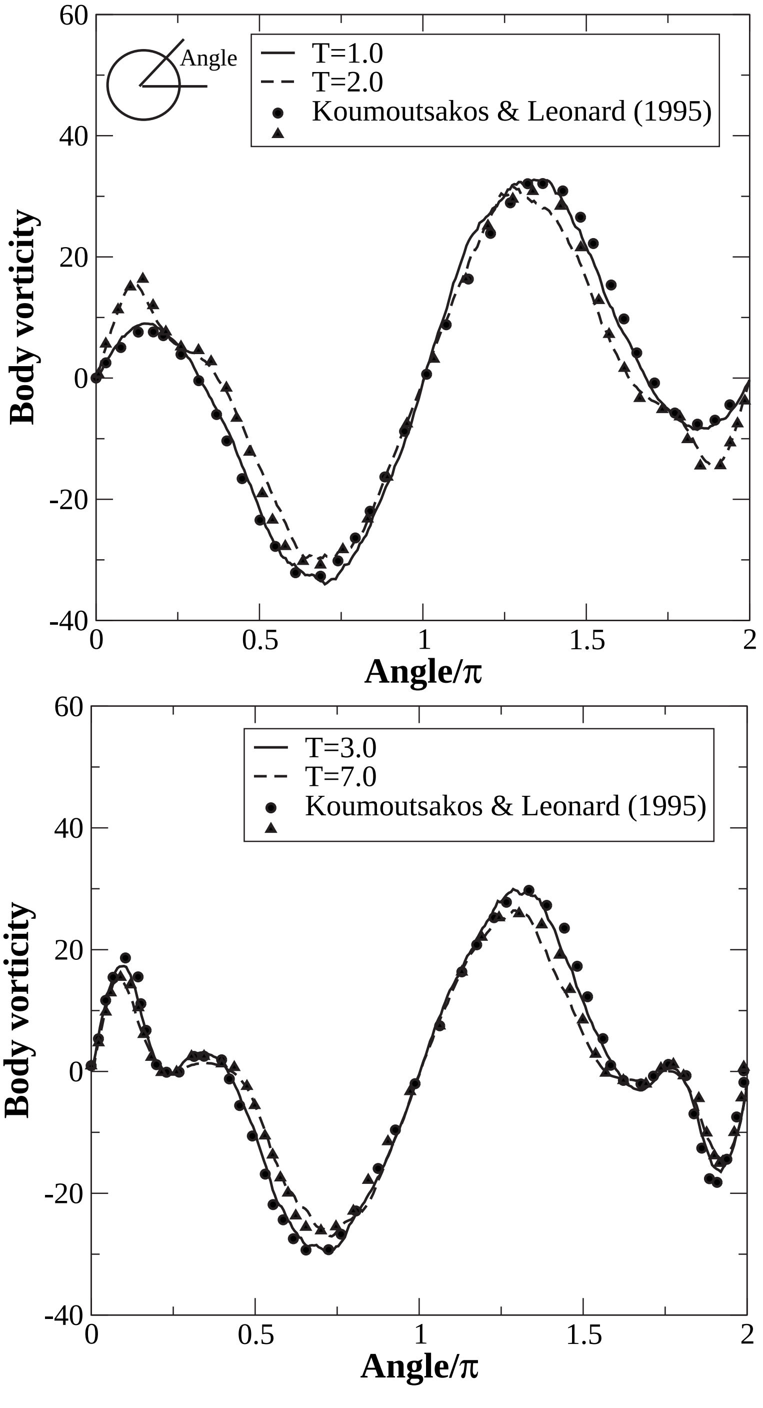

Figure 8.10: Body vorticity for impulsively translated cylinder at $Re = 550$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

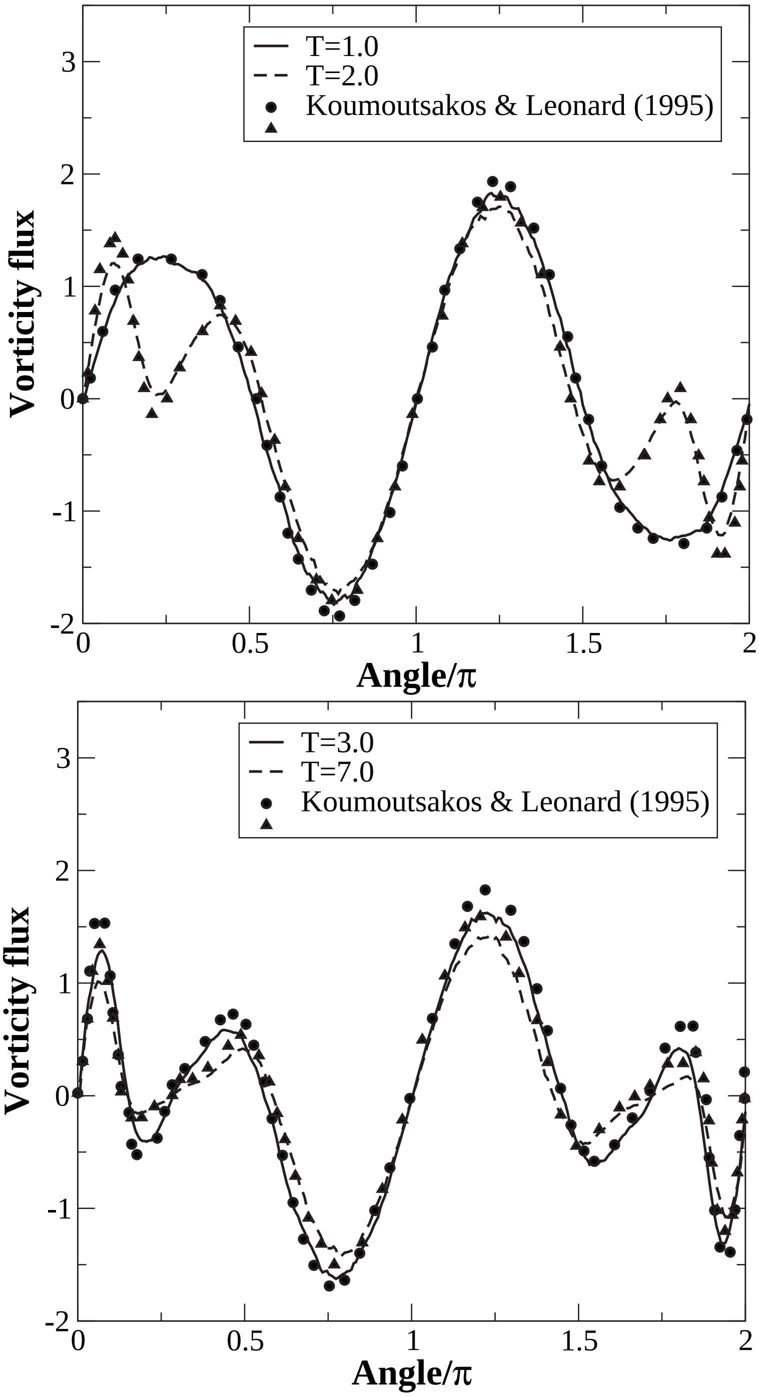

Figure 8.11: Vorticity flux for translated cylinder at $Re = 550$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

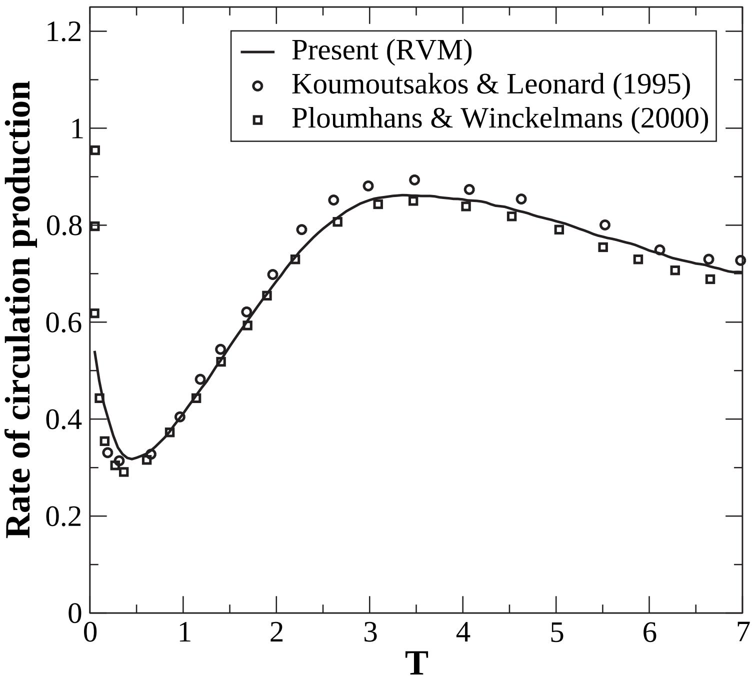

Figure 8.12: Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 550$. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000).

to that of Koumoutsakos and Leonard (1995). The data is smoothed using a 11 point sliding average. Fig. 8.11 shows the variation of the vorticity flux versus the angle. This data is also smoothed using a 11 point sliding average.

Fig. 8.12 plots the rate of circulation production from the lower half of the cylinder versus T. The values obtained by Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000) are also shown. The agreement is very good with the present values being in-between those of the other computations.

Fig. 8.13 plots the iso-vorticity contours at various times. The vorticity is transferred to a regular grid with spacing, $h \approx 0.015$. The data is smoothed once using a Laplace smoothing operation. Although not shown here, these results also match well with available plots of the vorticity contours using different diffusion schemes.

Fig. 8.14 plots the streamlines for the flow. The range of x is $[-1.5, 3.0]$ and the range of y is $[-1.4, 1.4]$. The values of the streamfunction contours are, 0 , ± { 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35,

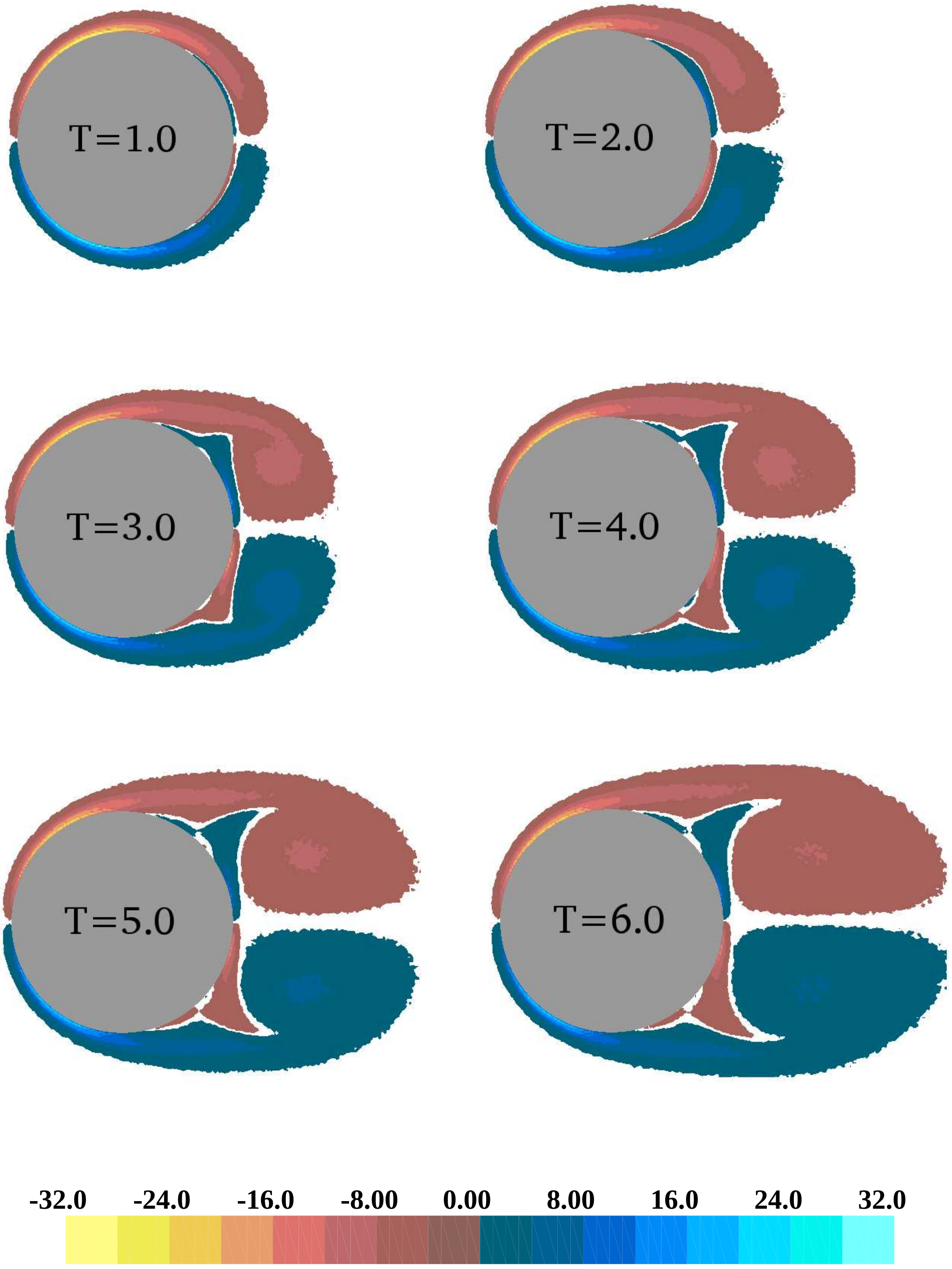

Figure 8.13: Iso-vorticity contours for an impulsively translated cylinder at $Re = 550$.

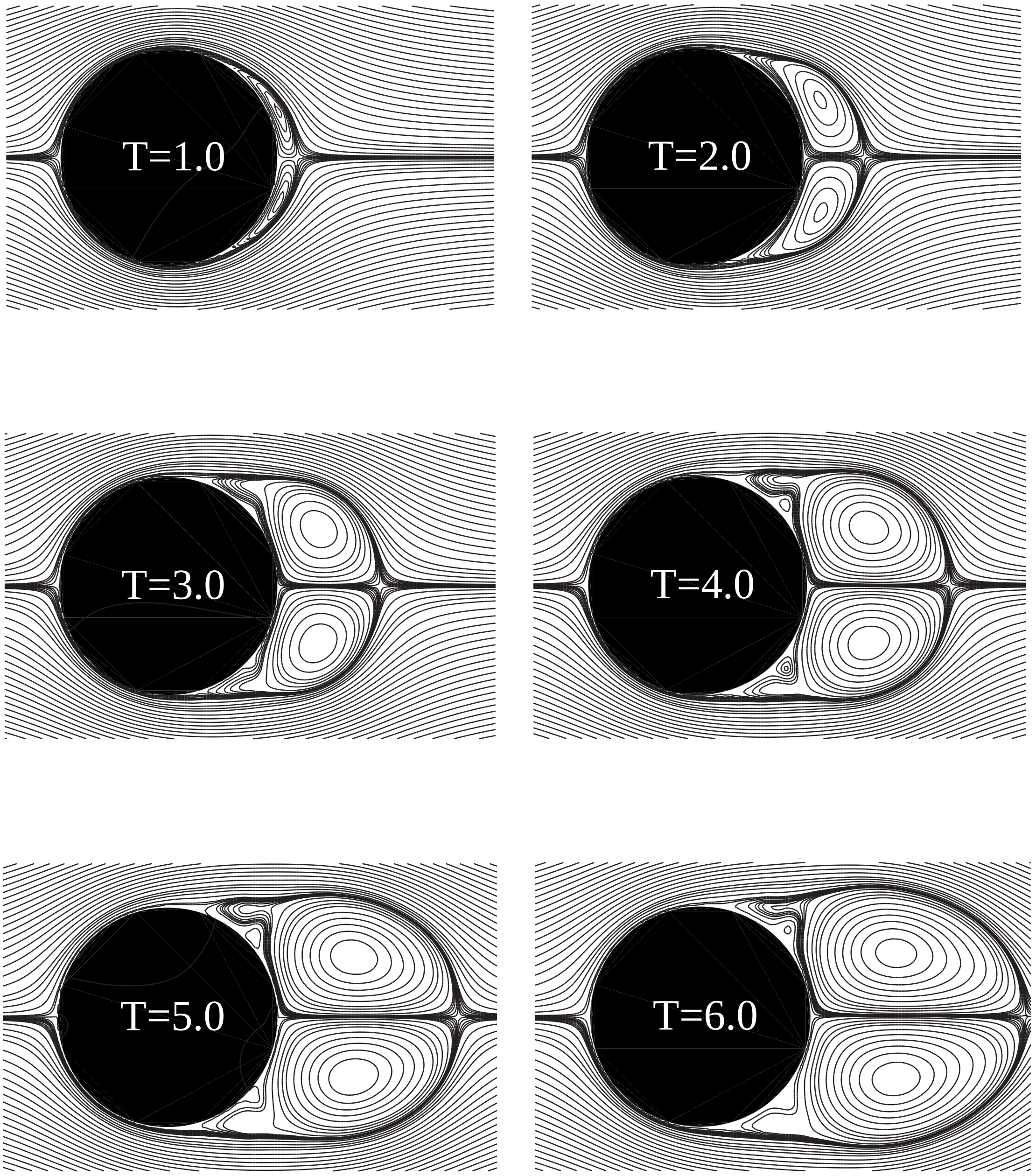

Figure 8.14: Streamlines for an impulsively translated cylinder at $Re = 550$.

0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0, 1.05, 1.1, 1.15}. These results also agree well with available results from other computations.

8.1.3 $Re = 1000$

The computational parameters chosen for this case are, $\Delta t = 0.05$ ($C = 6.37$), $k_1 \approx 0.025$, $\gamma_{max} = 0.0125$ and $R_a = R_m = 0.25$. The body is discretized into 400 panels and viscous boxes. Second order Runge-Kutta integration is used along with Strang discretization. Sheet2 with a release style of 3 is used. Both blobs and sheets are annihilated and merged. Eight trials are made and the ensemble of these is considered. At $T = 6$, there are around 100000 vortex blobs and 3000 vortex sheets. A single trial takes 44 minutes to execute on a Pentium-IV 1.7Ghz machine.

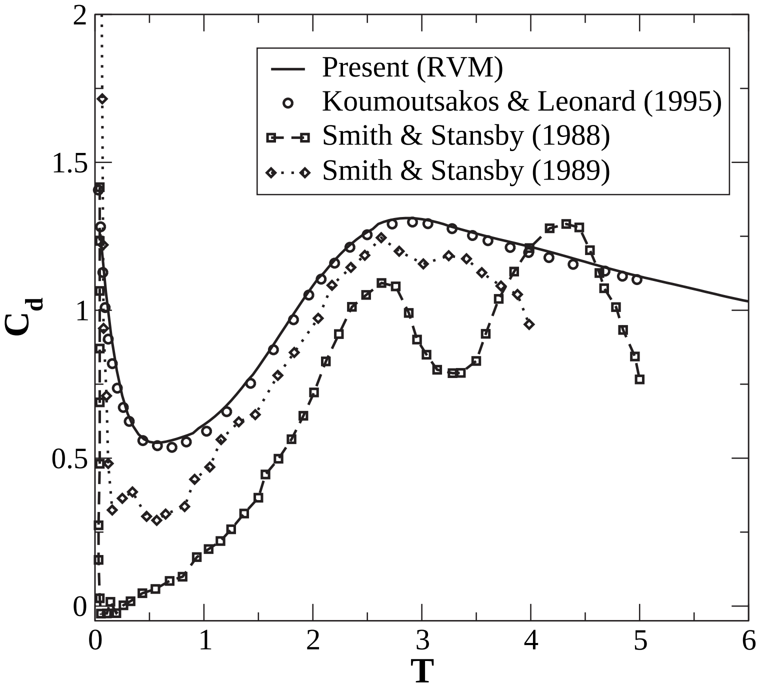

Figure 8.15: Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 1000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Smith and Stansby (1988, 1989).

Fig. 8.15 plots the drag coefficient, C_d versus non-dimensional time, T. The solid line corresponds to the results from the present work. The curve is obtained using a piecewise 6th order polynomial fit of the data with 23 points per piece.

The circular symbols correspond to the results from Koumoutsakos and Leonard (1995) and the square and diamond symbols to the results of Smith and Stansby (1988, 1989). Shankar (1996) does not present results for this Reynolds number. The figure clearly shows that the present results agree very well with those of Koumoutsakos and Leonard (1995). It is clear that the results of Smith and Stansby (1988, 1989) do not agree well with the present results. They also use the RVM for their computations. The poor agreement suggests that too few particles were used by them to resolve the features of the flow. It is also clearly seen that the results of Smith and Stansby (1989) are better than those of Smith and Stansby (1988). In their paper, Koumoutsakos and Leonard (1995) compare their results with those of Smith and Stansby (1988) and conclude that the poor agreement is due to the RVM. However, it is clear from the results of the present work that this is simply not the case and that it is possible to obtain excellent results with the RVM.

Like the $Re = 550$ case, there is very little noise in the results and a simple five point sliding average for the load curve would also produce excellent results. The noise level in the curves is 0.00651 and the standard deviation, σ, between trials is 0.01260.

Fig. 8.16 plots the variation of the pressure and frictional drag versus T. The symbols correspond to the results of Koumoutsakos and Leonard (1995). A seven point average was used to smooth the slight oscillations in the curve. As seen, the agreement is very good except for the slight discrepancy in the pressure load at initial times.

Fig. 8.17 shows the variation of the body vorticity versus the angle measured from the rear stagnation point of the cylinder and traversing in an anti-clockwise direction. Fig. 8.18 shows the variation of the vorticity flux on the body versus the angle. The lines correspond to the results of the present work and the symbols to that of Koumoutsakos and Leonard (1995). The data is smoothed using a 11 point sliding average. The body vorticity curves are in excellent agreement while the peaks for the flux curves are lower. The smaller peaks arise due to the smoothing. Without the smoothing the curves are noisy. However, the overall agreement is

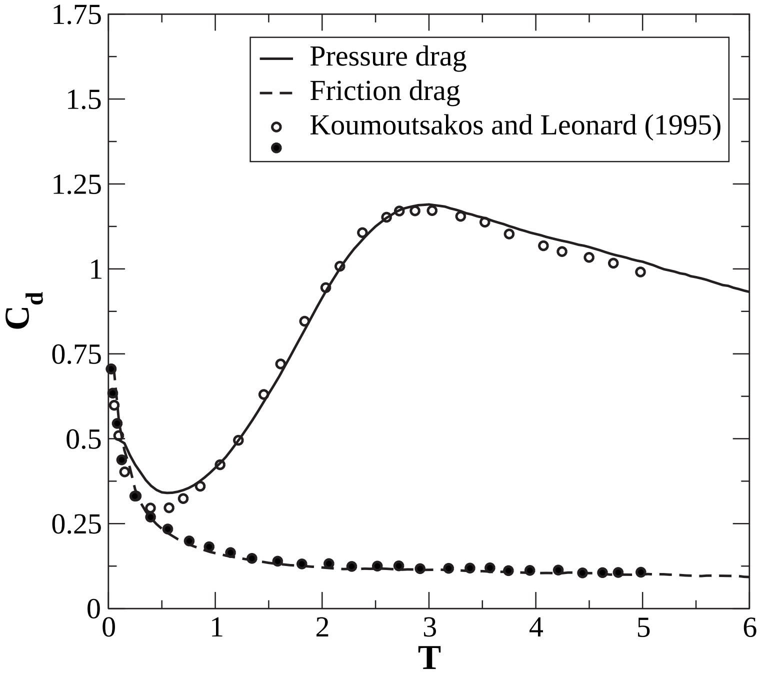

Figure 8.16: Pressure and friction drag versus T for impulsively translated cylinder at $Re = 1000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

good.

Fig. 8.19 plots the rate of circulation production from the lower half of the cylinder versus T. The values obtained by Koumoutsakos and Leonard (1995) are also shown. The agreement is good and there is a 5% difference in the peak value. Given the discrepancy between the results of Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000) for the $Re = 550$ case, this appears quite acceptable.

Fig. 8.20 plots the iso-vorticity contours at various times. The vorticity is transferred to a regular grid with spacing, $h \approx 0.015$. The data is smoothed once using a Laplace smoothing operation.

Fig. 8.21 plots the streamlines for the flow at times of $T = 1, 2, 3, 4, 5$ and 6. The range of x and y and the contours considered are the same as chosen for the $Re = 550$ case.

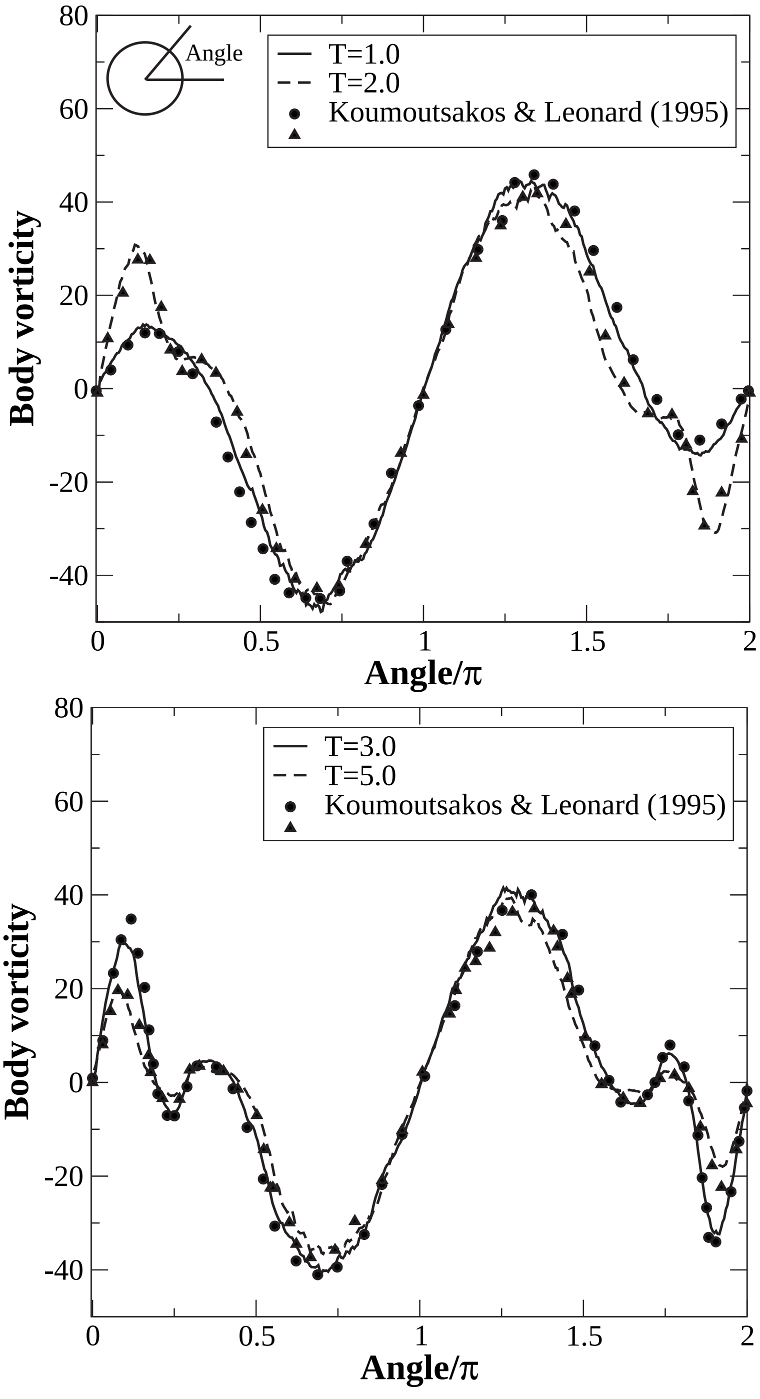

Figure 8.17: Body vorticity for impulsively translated cylinder at $Re = 1000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

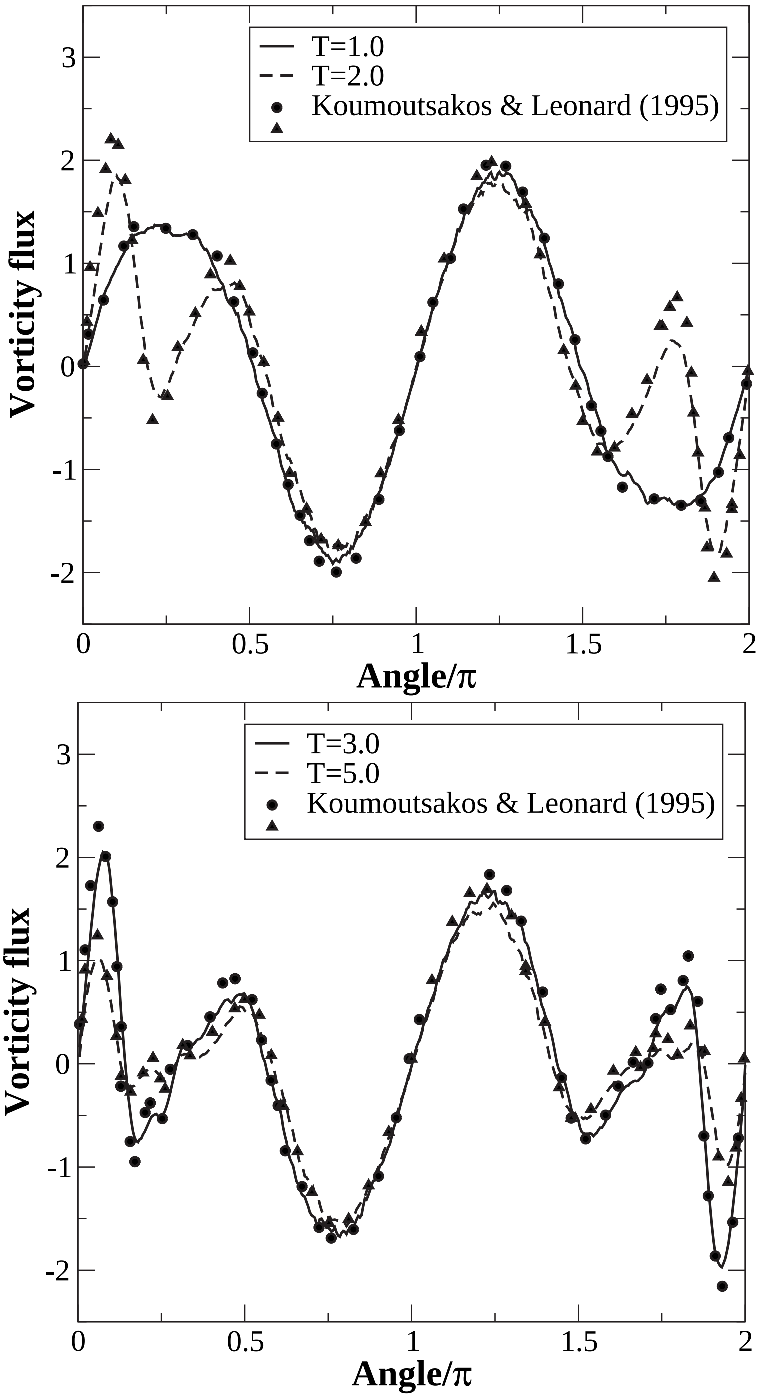

Figure 8.18: Vorticity flux for impulsively translated cylinder at $Re = 1000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

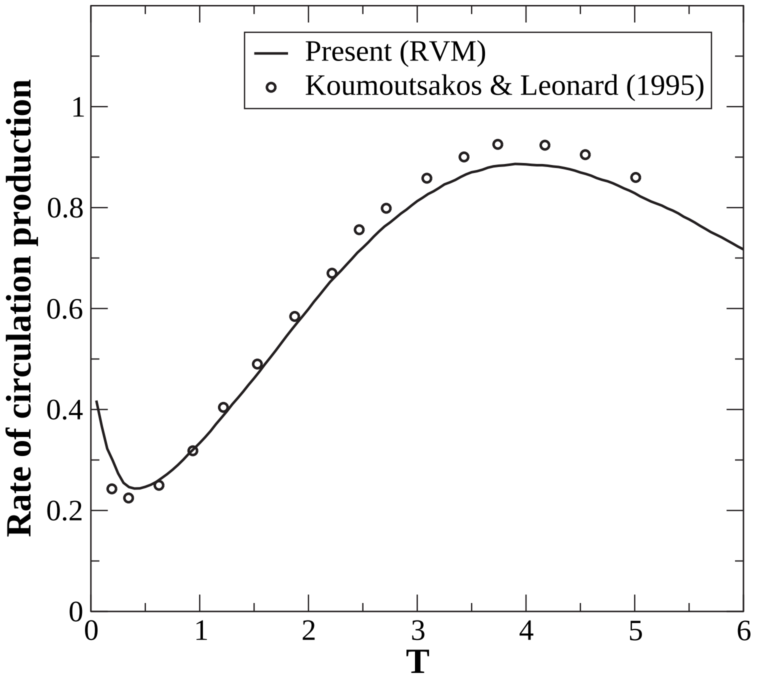

Figure 8.19: Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 1000$. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

8.1.4 $Re = 3000$

The computational parameters chosen for this case are, $\Delta t = 0.025$ ($C = 6.35$), $k_1 \approx 0.022$, $\gamma_{max} = 0.025$ and $R_a = 0.6$. The body is discretized into 266 panels with 798 ($= 266 \times 3$) viscous boxes. Second order Runge-Kutta integration is used along with Strang discretization. Sheet2 with a release style of 3 is used. Both blobs and sheets are annihilated and merged. Eight trials are made in parallel using the ERVM. The ensembling occurs every 10 iterations ($n_{sync} = 10$) in the manner described in section 3.9. The ensemble of these runs is considered for the comparison. At $T = 6.25$, an average[1] of around 90000 vortex blobs and 3500 sheets are used in the simulation. The entire run takes 224 minutes on a cluster of eight Pentium-IV class machines.

Fig. 8.22 plots the drag coefficient, C_d versus non-dimensional time, T. The solid line corresponds to the results from the present work. The curve is obtained using a piecewise 6th order polynomial fit of the data with 45 points per piecewise polynomial. The symbols correspond to the results from Koumoutsakos and

---

[1] Averaged over each simulation starting from $T = 0$ and then averaged over the trials.

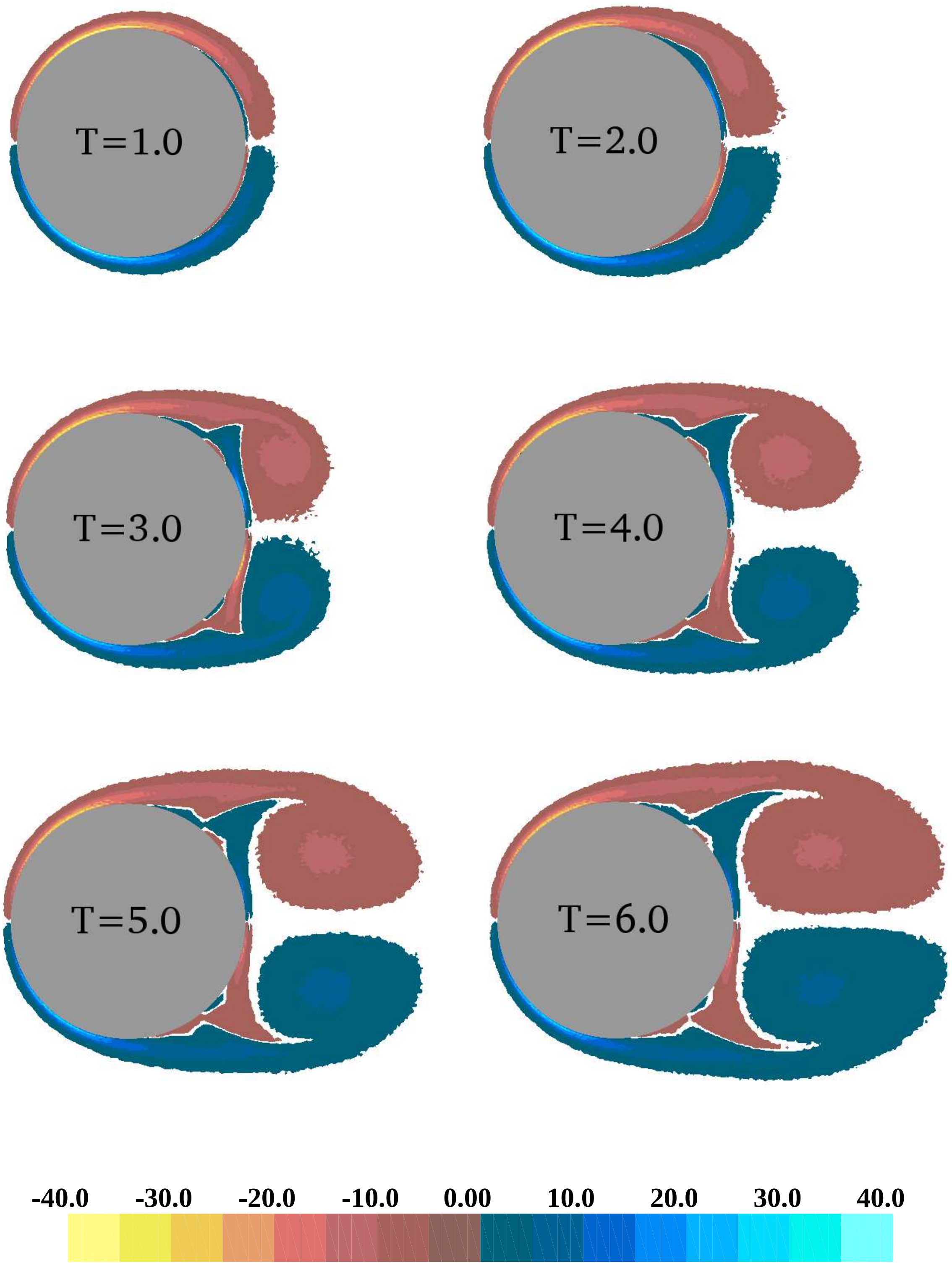

Figure 8.20: Iso-vorticity contours for an impulsively translated cylinder at $Re = 1000$.

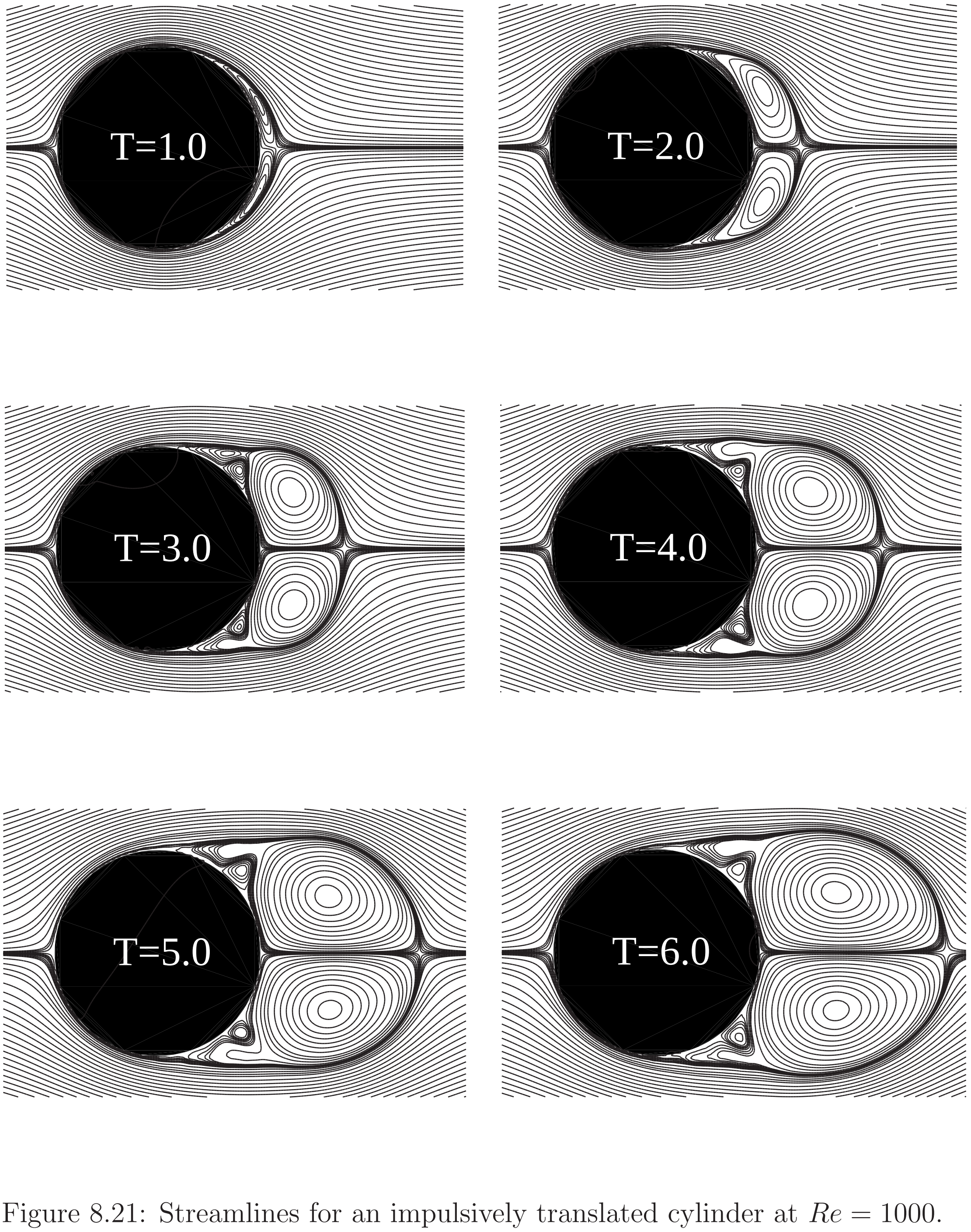

Figure 8.21: Streamlines for an impulsively translated cylinder at $Re = 1000$.

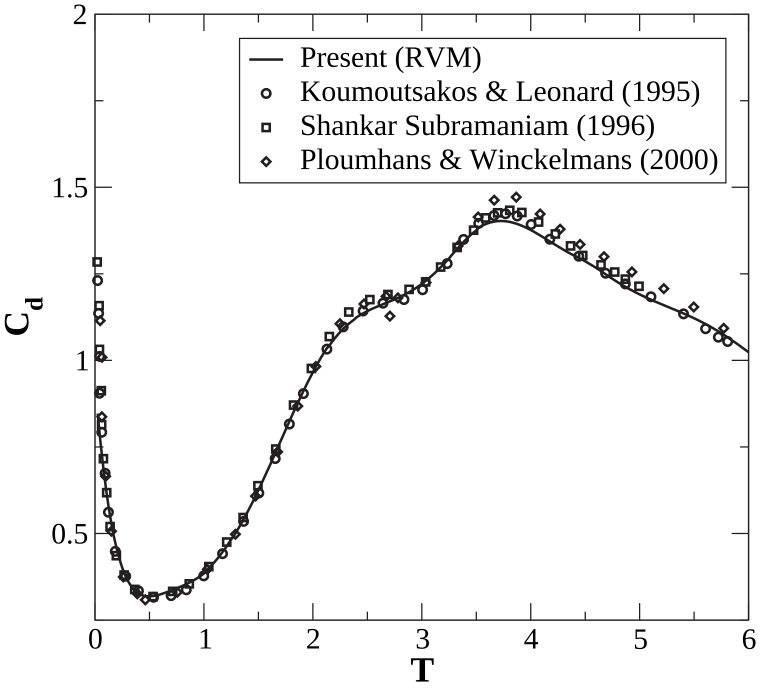

Figure 8.22: Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 3000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995); Shankar (1996); Ploumhans and Winckelmans (2000).

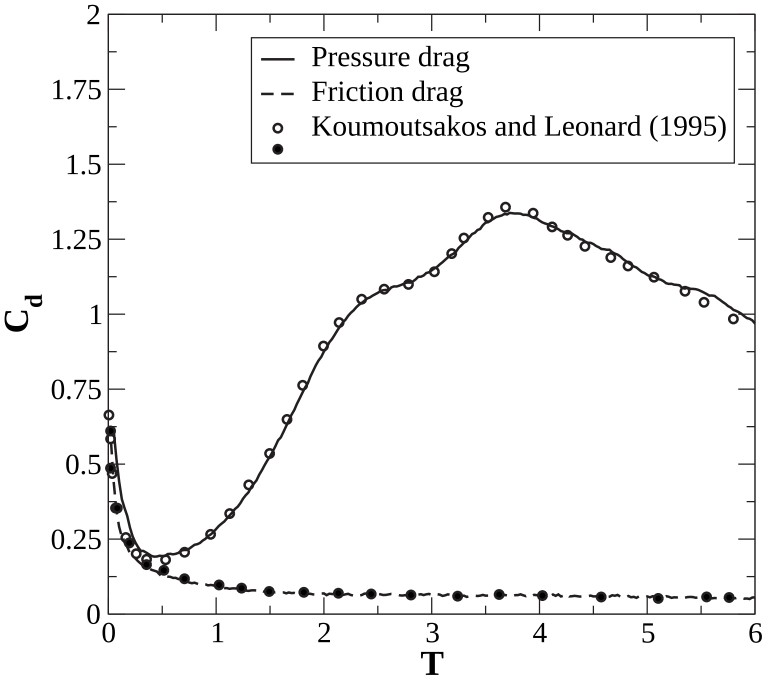

Figure 8.23: Pressure and friction drag versus T for impulsively translated cylinder at $Re = 3000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

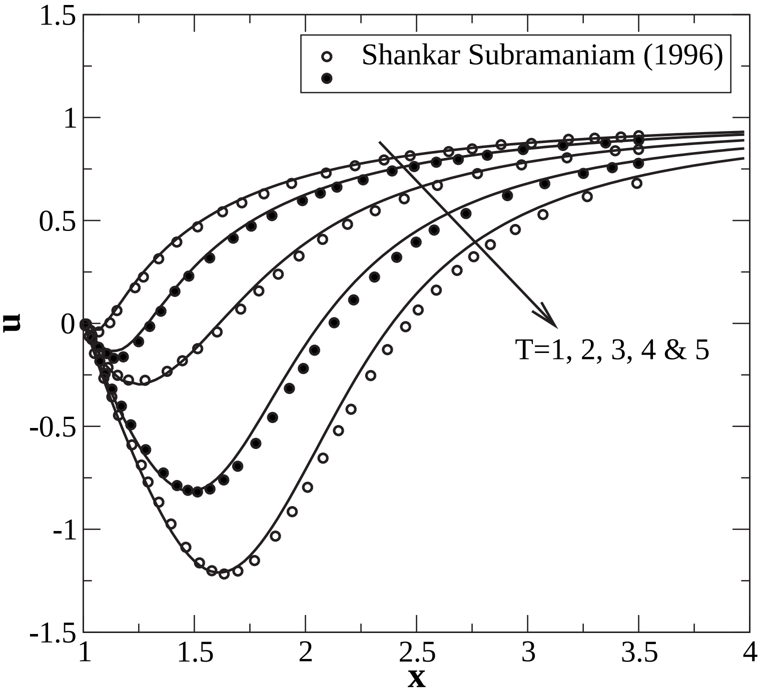

Figure 8.24: Radial velocity along the axis of symmetry on the rear side of an impulsively translated cylinder at $Re = 3000$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996).

Leonard (1995), Shankar (1996) and Ploumhans and Winckelmans (2000). The figure clearly shows that the present results agree well with those of the deterministic diffusion scheme results. The noise level in the load curve is 0.02428 and the standard deviation, $\sigma = 0.01092$.

Fig. 8.23 plots the pressure and friction drag versus T. The solid curve represents results from the present computations and the symbols correspond to the results of Koumoutsakos and Leonard (1995). An eleven point sliding average is used to smooth the oscillations in the curve for the pressure drag. As seen, the agreement is excellent.

Fig. 8.24 plots the variation of the radial velocity along the axis of symmetry on the rear side of the cylinder. The symbols correspond to the results of Shankar (1996). The agreement is good.

Fig. 8.25 shows the variation of the body vorticity versus the angle measured from the rear stagnation point of the cylinder and traversing in an anti-clockwise sense.

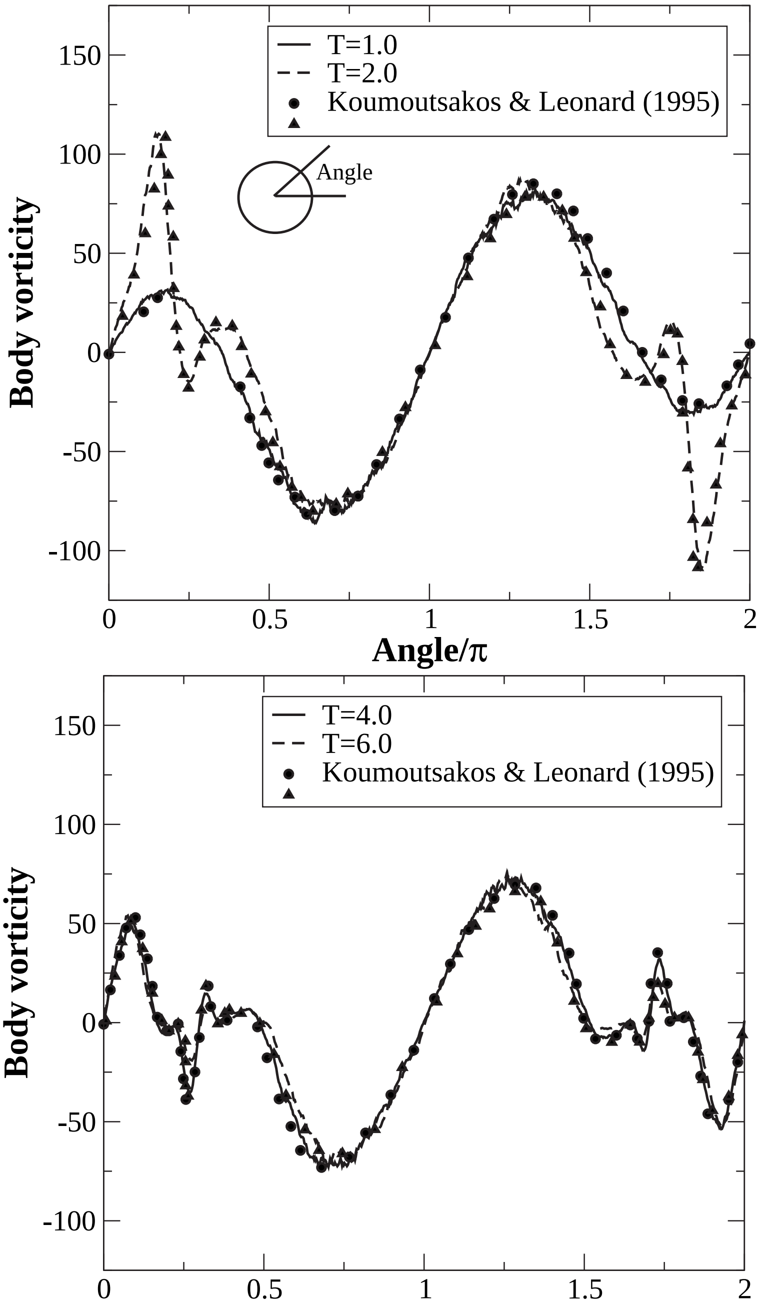

Figure 8.25: Body vorticity for impulsively translated cylinder at $Re = 3000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

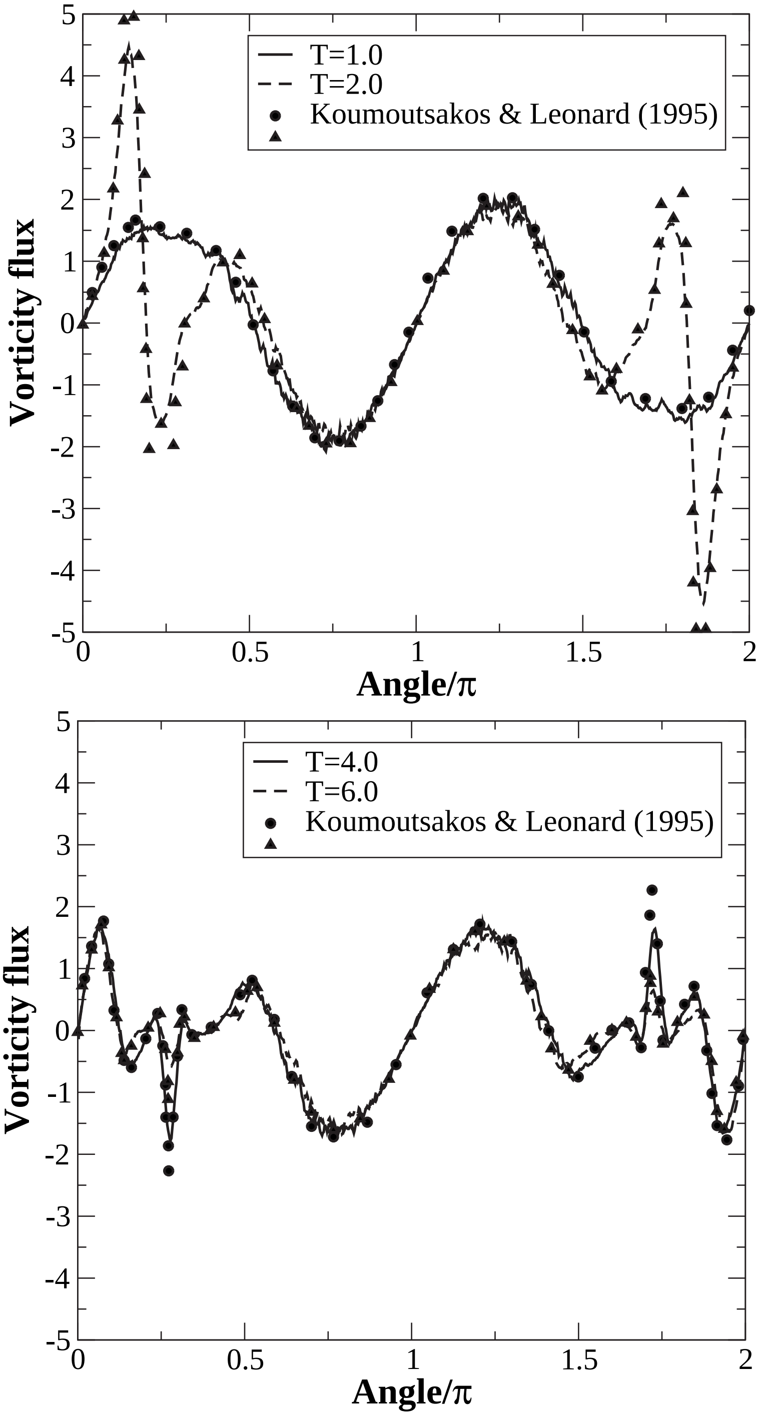

Figure 8.26: Vorticity flux for impulsively translated cylinder at $Re = 3000$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

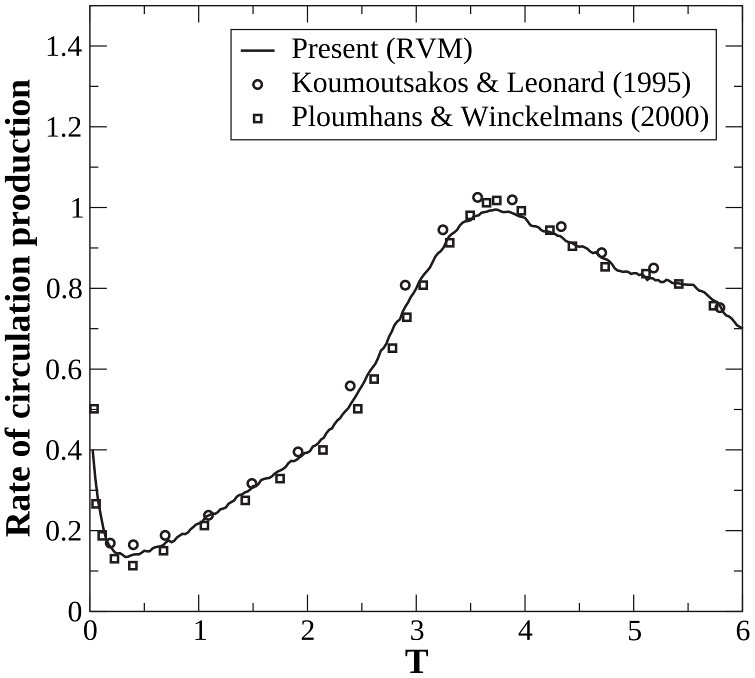

Figure 8.27: Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 3000$. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000).

Fig. 8.26 shows the variation of the vorticity flux versus the angle. The lines correspond to the results of the present work and the symbols to that of Koumoutsakos and Leonard (1995). The data is smoothed using a 15 point sliding average. The body vorticity curves are in excellent agreement while the peaks for the flux curves computed using the RVM are lower. Once again, the results are in very good overall agreement.

Fig. 8.27 plots the rate of circulation production from the lower half of the cylinder versus T. The values obtained by Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000) are also shown. The agreement is clearly very good.

Fig. 8.28 plots the iso-vorticity contours at various times. The vorticity is transferred to a regular grid with spacing, $h \approx 0.01$. The data is smoothed twice using a Laplace smoothing operation.

Fig. 8.29 plots the streamlines for the flow at times of $T = 1, 2, 3, 4, 5, 6$. The range of x and y and the contours considered are the same as chosen for the

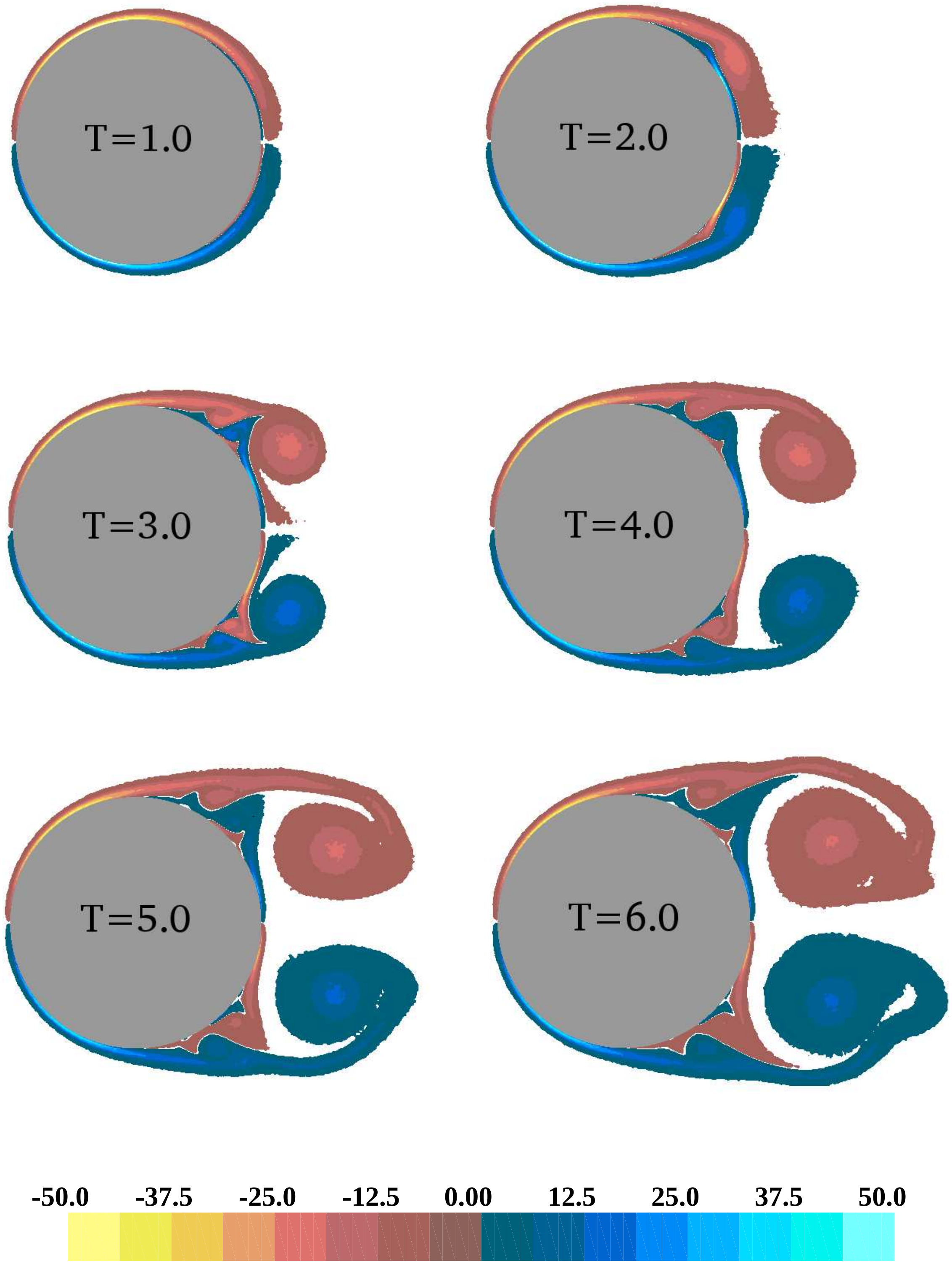

Figure 8.28: Iso-vorticity contours for an impulsively translated cylinder at $Re = 3000$.

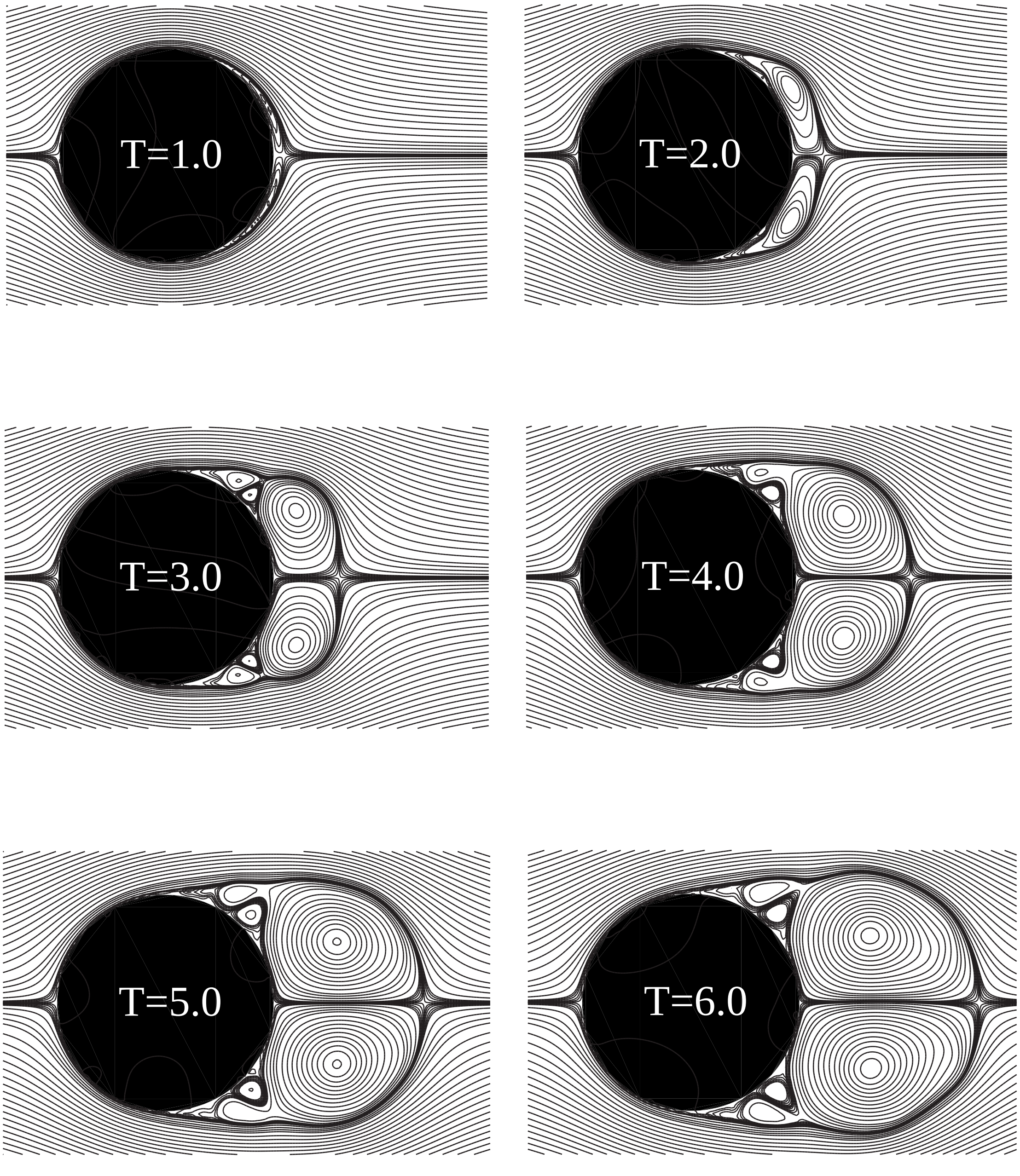

Figure 8.29: Streamlines for an impulsively translated cylinder at $Re = 3000$.

$Re = 550$ and $Re = 1000$ cases.

8.1.5 $Re = 9500$

The computational parameters chosen for this case are, $\Delta t = 0.0125$ ($C = 4.94$), $k_1 \approx 0.025$, $\gamma_{max} = 0.025$, $R_a = 0.15$ and $R_m = 0.3$. The body is discretized into 620 panels with 1240 ($= 620 \times 2$) viscous boxes. Second order Runge-Kutta integration is used along with Strang discretization. Sheet2 with a release style of 3 is used. Both blobs and sheets are annihilated and merged. Eight trials are made in parallel using the ERVM. The ensembling is done every 40 iterations ($n_{sync} = 40$) in the manner described in section 3.9. The ensemble of these runs is considered for the results. At $T = 6.0$, 203500 vortex blobs and 6100 sheets are present. An average of 148000 vortex blobs and 8100 sheets have used been during the entire course of the simulation. The entire run takes around 480 minutes (8 hours) to execute on a cluster of eight Pentium-IV class machines.

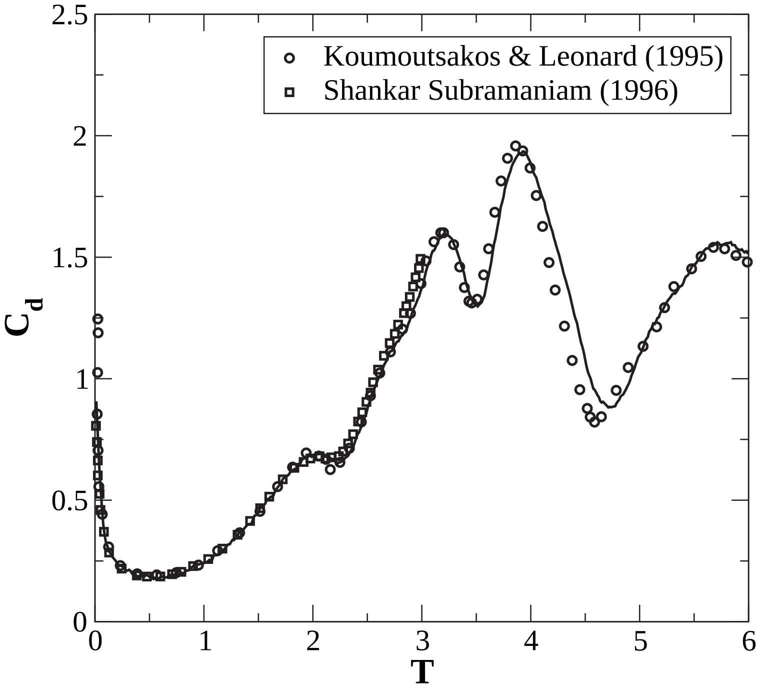

Figure 8.30: Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 9500$. Solid line is the present solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995); Shankar (1996).

It is important to note that obtaining a symmetric flow at this Reynolds num-

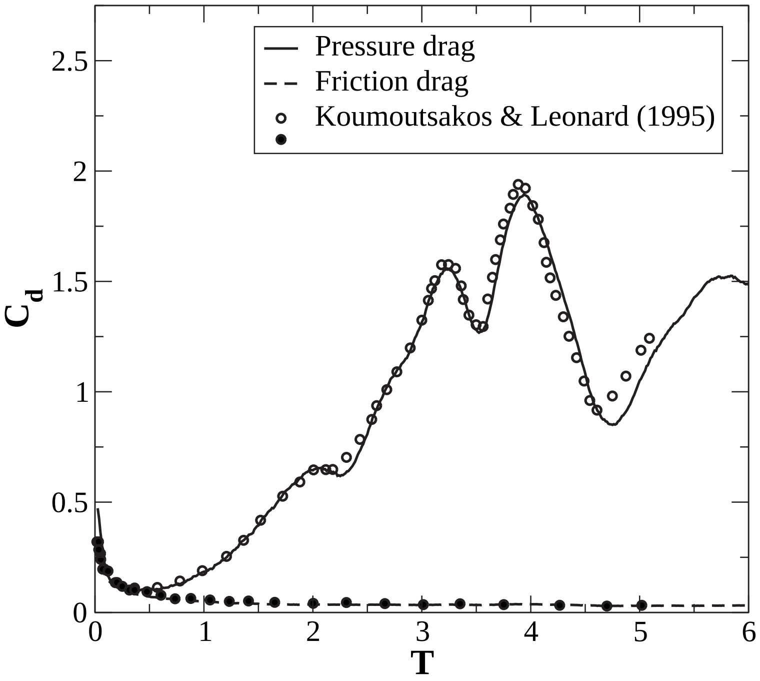

Figure 8.31: Pressure and friction drag versus T for impulsively translated cylinder at $Re = 9500$. Solid line and dashed lines correspond to the solution using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

ber proved very challenging. The inherently noisy nature of the RVM coupled with the unstable nature of the flow at this Reynolds number makes it difficult to obtain symmetric results beyond $T = 3$. A fair amount of fine tuning of parameters was necessary. Most of the cases produced reasonable results up to a time of $T \approx 3$. However, it proved difficult to obtain symmetric results beyond this time. A more serious discussion on the issue is provided in section 8.2.

Fig. 8.30 plots the drag coefficient, C_d versus non-dimensional time, T. The solid line corresponds to the results from the present work. The curve is smoothed using a 7 point sliding average. The symbols correspond to the results from Koumoutsakos and Leonard (1995) and Shankar (1996). The figure clearly shows that the present results agree well with those of the deterministic diffusion scheme results. The noise level in the load curve is 0.02222 and the standard deviation, $\sigma = 0.00639$ (this is computed using a 6th order piecewise polynomial with 23 points per piecewise polynomial).

Fig. 8.31 plots the pressure and friction drag versus T. The solid curve represents results from the present computations and the symbols correspond to the

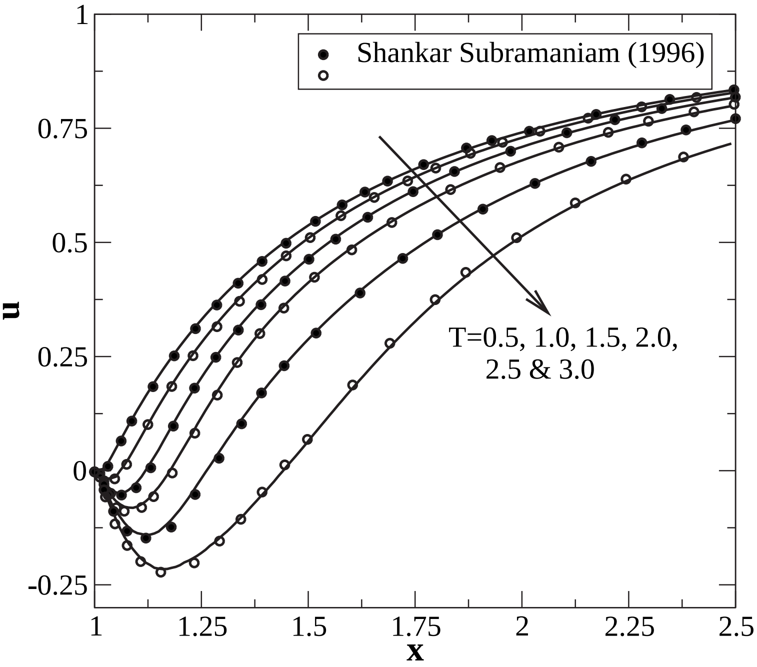

Figure 8.32: Radial velocity along the axis of symmetry on the rear side of an impulsively translated cylinder at $Re = 9500$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996).

results of Koumoutsakos and Leonard (1995). A seven point sliding average is used to smooth the oscillations in the curve for the pressure drag. As seen, the agreement is good.

Fig. 8.32 plots the variation of the radial velocity along the axis of symmetry at the rear of the cylinder. The symbols correspond to the results of Shankar (1996). The agreement is excellent.

Fig. 8.33 plots the variation of the tangential component of the velocity along radial lines at different angles, θ, measured clockwise from the front symmetry line. The symbols correspond to the results from Shankar (1996). The velocity is computed as the ensemble average of all the 8 trials. The agreement between the computations is very good.

Fig. 8.34 shows the variation of the body vorticity versus the angle measured from the rear stagnation point of the cylinder and traversing in an anti-clockwise sense. Fig. 8.35 shows the variation of the vorticity flux versus the angle. The lines correspond to the results of the present work and the symbols to that of

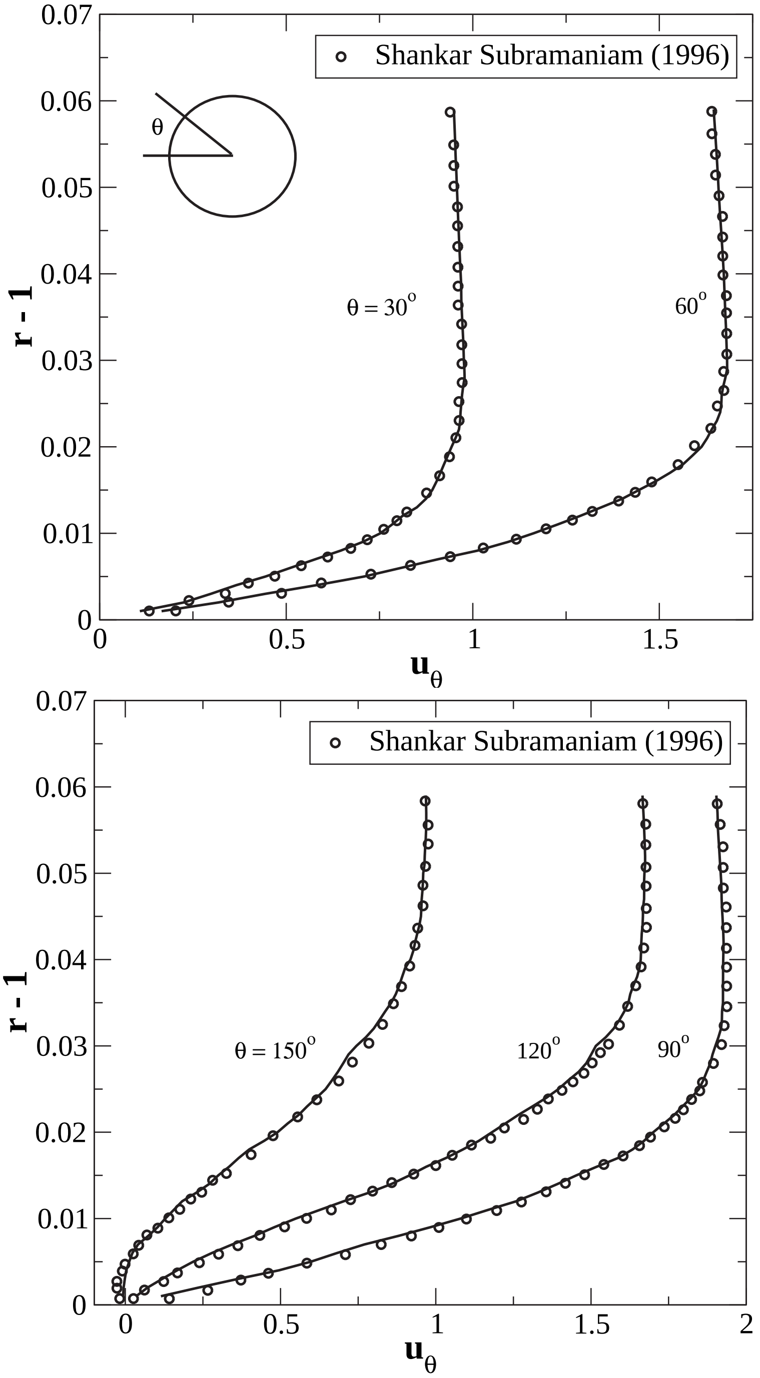

Figure 8.33: Tangential velocity component along radial lines at different angles for an impulsively translated cylinder at $Re = 9500$. Solid line is the present solution using the RVM. Symbols correspond to the results of Shankar (1996).

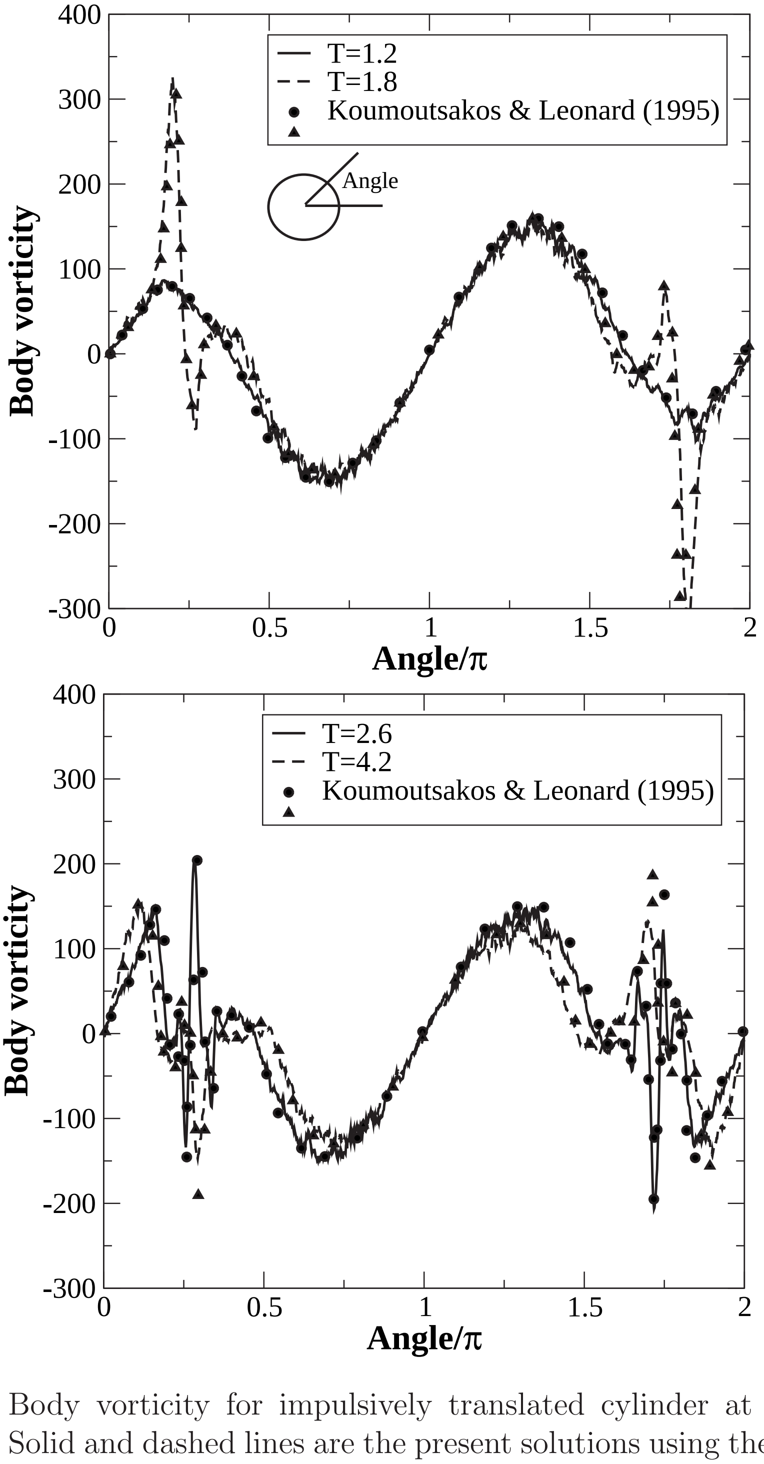

Figure 8.34: Body vorticity for impulsively translated cylinder at $Re = 9500$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

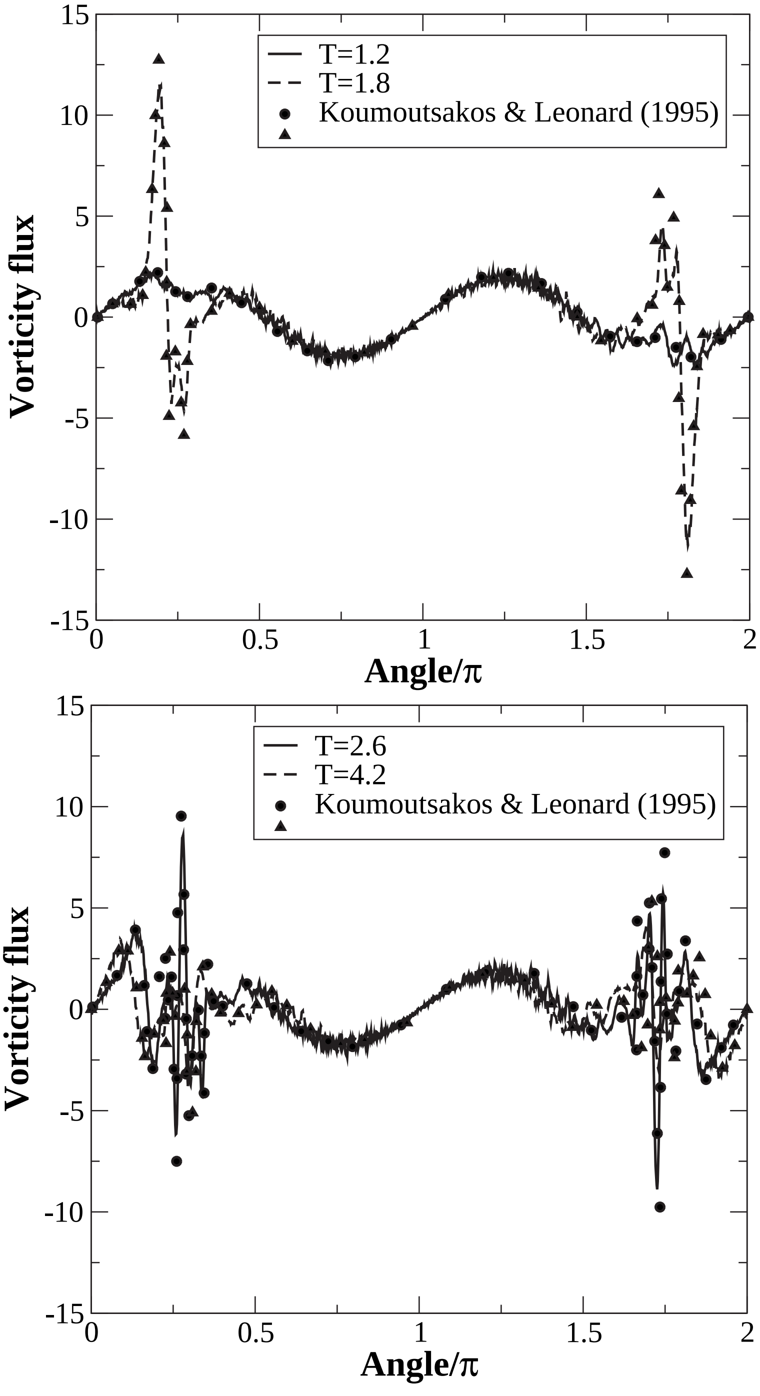

Figure 8.35: Vorticity flux for impulsively translated cylinder at $Re = 9500$. Solid and dashed lines are the present solutions using the RVM. Symbols correspond to the results of Koumoutsakos and Leonard (1995).

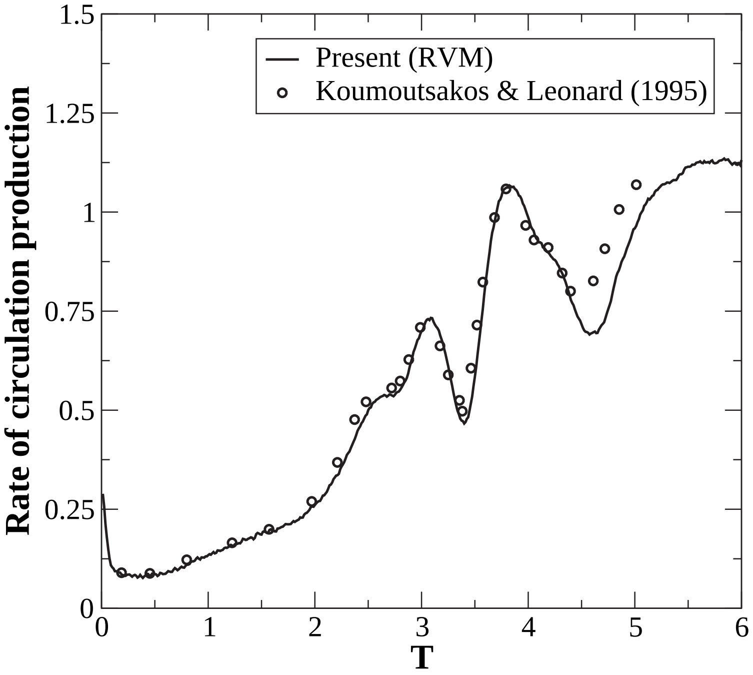

Figure 8.36: Rate of circulation production from the lower half of the cylinder versus T for impulsively translated cylinder at $Re = 9500$. Symbols correspond to the results of Koumoutsakos and Leonard (1995) and Ploumhans and Winckelmans (2000).

Koumoutsakos and Leonard (1995). The data is smoothed using a 15 point sliding average. The body vorticity curves are in excellent agreement while the peaks for the flux curves are lower. The overall agreement is very good.

Fig. 8.36 plots the rate of circulation production from the lower half of the cylinder versus T. The curve is smoothed using a 5 point sliding average. The values obtained by Koumoutsakos and Leonard (1995) are also shown. The agreement is very good up to about $T = 4.25$ beyond which there is some discrepancy due to the difficulty in maintaining symmetry in the present computations.

Fig. 8.37 plots the iso-vorticity contours at various times. The vorticity is transferred to a regular grid with spacing, $h = 0.01$. The data is smoothed once using a Laplace smoothing operation.

Fig. 8.38 plots the streamlines for the flow at times of $T = 1, 2, 3, 4, 5, 6$. The range of x is $[-0.75, 1.75]$ and y is $[-1.05, 1.05]$. The contour levels chosen are the same values as specified by Shankar (1996): 0 , ± { 0.001, 0.003, 0.005, 0.007, 0.01, 0.015, 0.027, 0.04, 0.06, 0.08, 0.1, 0.125, 0.15, 0.18, 0.21, 0.24, 0.27, 0.3, 0.35,

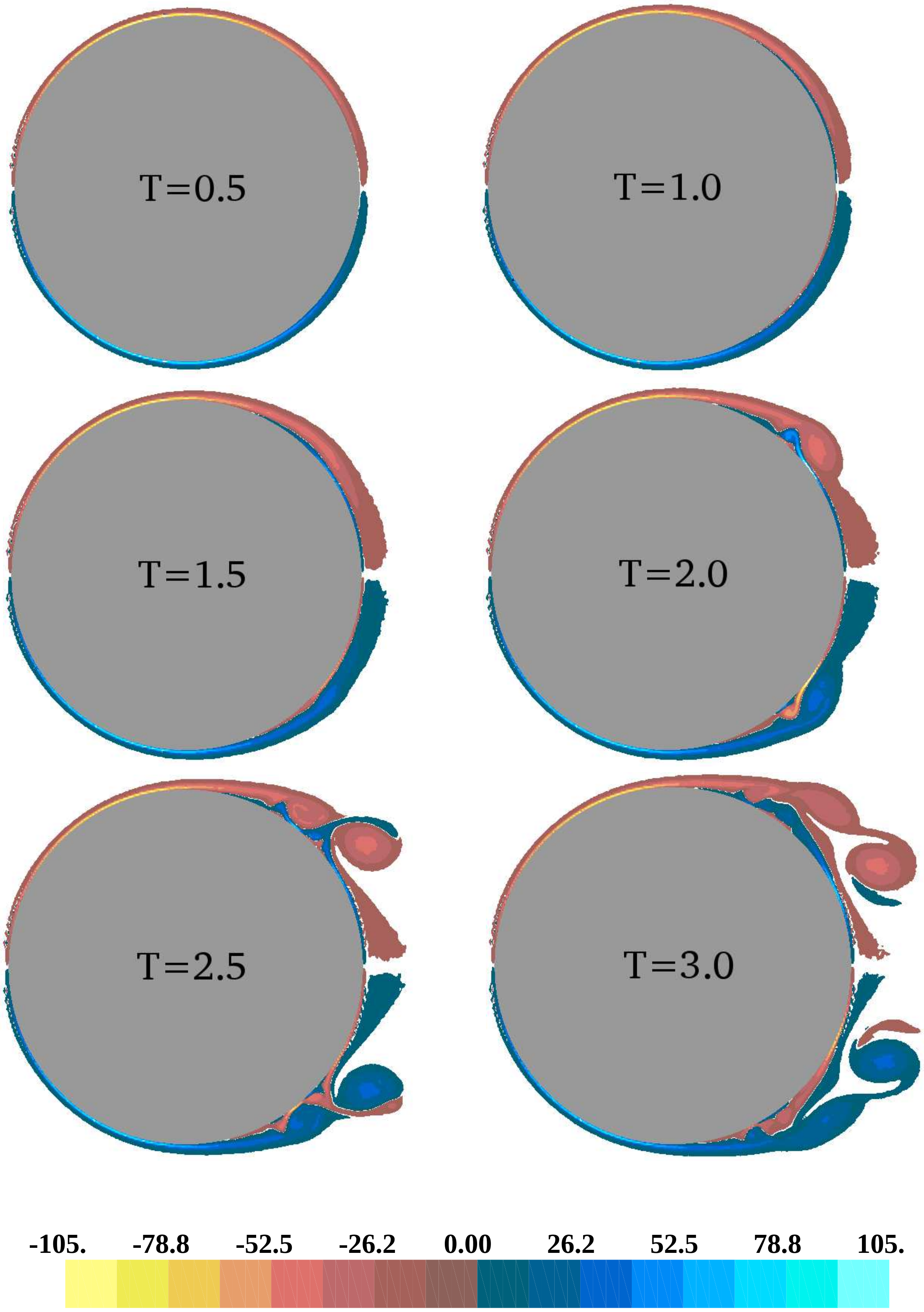

Figure 8.37: Iso-vorticity contours for an impulsively translated cylinder at $Re = 9500$.

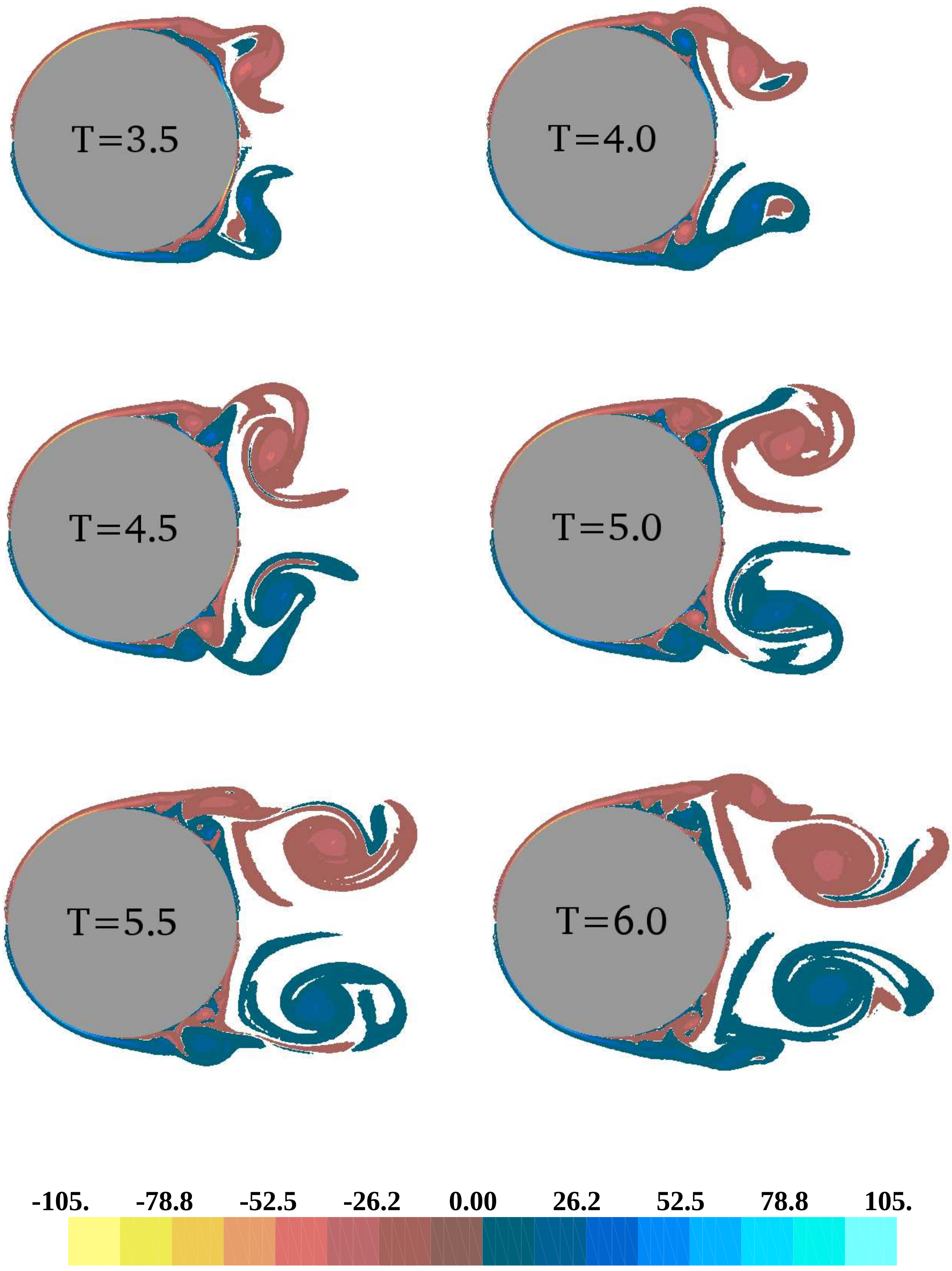

Figure 8.37: **(contd.)**: Iso-vorticity contours for an impulsively translated cylinder at $Re = 9500$.

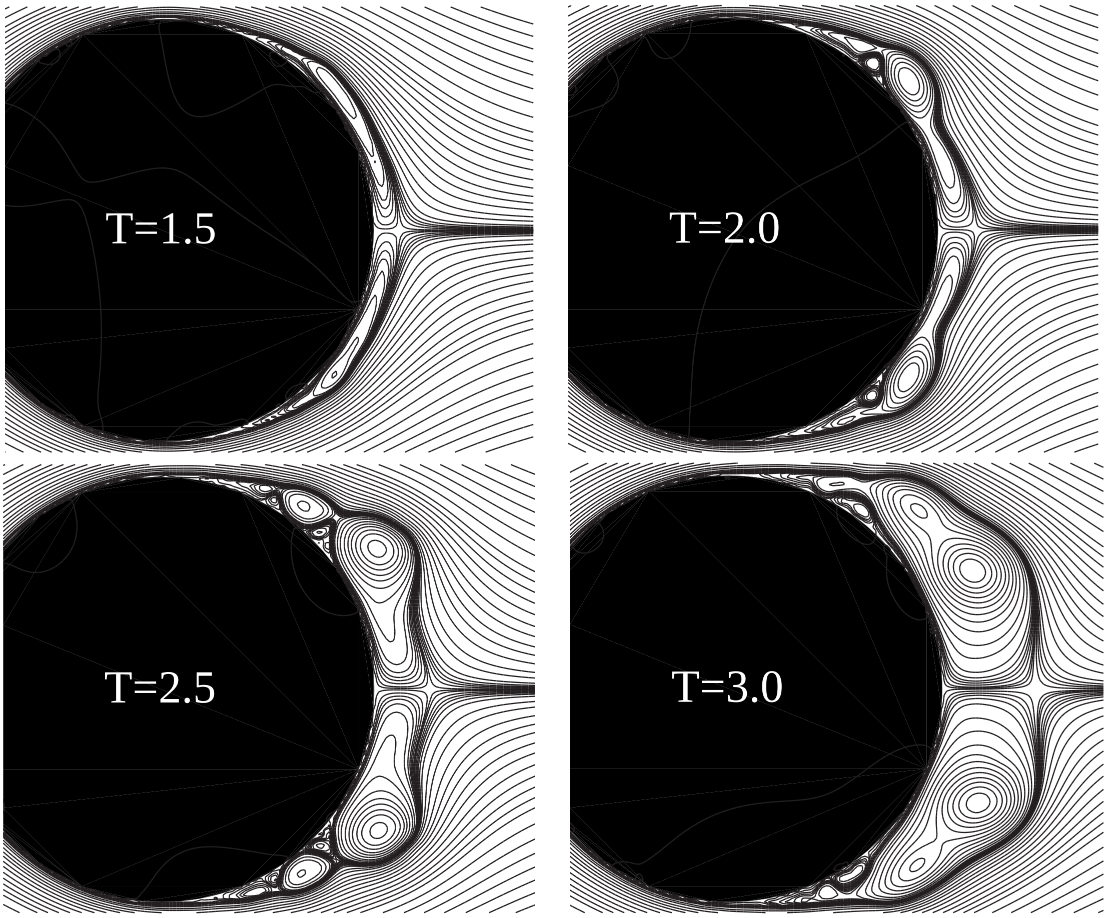

Figure 8.38: Streamlines for an impulsively translated cylinder at $Re = 9500$.

0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7}.

8.2 Discussion

In this work an efficient implementation of the random vortex has been provided and studied. Chorin's blob (Chorin and Bernard, 1973; Chorin, 1973) along with vortex sheets (Sheet2, section 2.4) in the numerical layer are used to discretize the vorticity. A second order Runge-Kutta scheme is used to integrate the ODEs for particle motion. A fast multipole method is used to accelerate the velocity computations. An accelerated panel method is used to satisfy the no-penetration boundary condition. Strang discretization (Beale and Majda, 1981) is used for operator splitting. The random vortex method is used to simulate diffusion. Particles are specularly reflected when they strike a solid surface. Annihilation and merging are performed to reduce the number of particles. Based on extensive numerical experimentation several recommendations were suggested in chapter 7. These recommendations are used to choose the parameters. A new and simple variance reduction scheme is used to reduce the noise levels and improve results significantly (section 7.8).

The following are important points to note.

1. The Chorin blob is used in the present work. This is a second order blob that is used frequently in random vortex simulations. However, most researchers using deterministic diffusion schemes seem to use the Gaussian blob.
2. Sheet2 as defined in section 2.4 does not satisfy the mass conservation equation since it induces no vertical velocity field in its region of influence. It is also questionable to use the vortex sheet method for the class of flows considered here since it assumes that the Prandtl boundary layer equations are valid. However, this error is a higher order one and only influences the numerical layer which is an extremely small region of the flow.
3. For the simulation of the Euler equations using vortex methods, it is known (Beale and Majda, 1985; Perlman, 1985; Nordmark, 1991) that accuracy and higher order convergence demand that the blob core-radii overlap. The present work does not use any special technique to ensure overlap.
4. The no-slip boundary condition is satisfied by introducing vortex sheets on the surface. While physically appealing, this method is considered to be

a simple approach as compared to that proposed by Koumoutsakos *et al.* (1994). Koumoutsakos and Leonard (1995) claim that the superiority of their results for lower Reynolds numbers is, among other things, also due to their sophisticated scheme to satisfy this boundary condition.

5. The random walk method has been used to simulate diffusion. This method is known to have a low rate of convergence (Roberts, 1985). The method also tends to produce noisy results due to its stochastic nature. It is also believed to require a large number of particles to produce high-quality results. With the advent of deterministic diffusion schemes, the RVM has been largely ignored for high-resolution simulations. It is only considered suitable for low-resolution, engineering approximations. Further, the poor agreement between the results of Koumoutsakos and Leonard (1995) and Smith and Stansby (1988) seem to validate these general conclusions. This motivates Koumoutsakos and Leonard (1995) to mention that the poor agreement is likely due to the slow convergence rates for the RVM and the simple means used to satisfy the no-slip boundary condition. With the exception of Shankar (1996) who notes that the RVM is capable of good results when a large number of particles are used, it is generally believed that the RVM is unsuitable for high-resolution computations.

In section 8.1, the results obtained for the flow past an impulsively started cylinder using the code developed were presented. The results were systematically compared with high-resolution results obtained by other researchers using deterministic diffusion schemes. The comparison is quite exhaustive with all well known quantities compared for a fairly wide range of Reynolds numbers. The agreement obtained is generally very good in almost all cases. This *definitively* indicates that the RVM can be used to perform high-resolution simulations. The present work therefore raises a few interesting issues and questions.

It appears safe to assume that the use of the Chorin blob is justified. Further experimentation with different blobs will clarify this point. The use of Sheet1 and Sheet2 also appears justified from the results obtained. Since the height of the numerical layer is very small (around 5% of an estimated maximum boundary layer height), it also appears that the choice of using the vortex sheet method in the numerical layer is not a serious issue.

It is often mentioned that maintaining core-overlap for the vortex particles is critical to obtaining good results for the NS equations. For example, Cottet *et al.* (2000) mention that the difficulty with maintaining core-overlap precludes the possibility of using the RVM for direct numerical simulations. Mathematical and numerical evidence exist to demonstrate that the convergence and higher order accuracy of vortex methods used to simulate the Euler equations for long times require particle overlap. However, Goodman *et al.* (1990) show that the point vortex method converges to solutions of the Euler equation. This result implies that even if vortex blobs were used without an explicit overlap, second order convergence is possible. It must also be mentioned that the PSE (Degond and Mas-Gallic, 1989) does have a strong stability requirement for particle overlap. Thus the need for core-overlap appears to be dictated by the diffusion scheme rather than by the solution of the Euler equations. The simulations with which the present results have been compared with (notably that of Koumoutsakos and Leonard (1995)) are considered to be direct numerical simulations. In the present work no explicit core-overlap is enforced and yet the results obtained are certainly comparable to theirs. Thus, it is not clear if the requirement for overlap is over-emphasized in the literature. There are problems involving high-resolution and high order convergence where particle overlap is a must as demonstrated by Perlman (1985); Nordmark (1991, 1996) and Koumoutsakos (1997). Specifically, the results of Nordmark (1996) are important in the context of the Navier-Stokes equations. However, given the results obtained in the present work for the simulation of bluff-body flows, it would appear that this is not an absolute requirement.

The difficulties in obtaining agreement with the results of Koumoutsakos and Leonard (1995) for the $Re = 9500$ case beyond $T = 3.0$ may indicate the need for overlap at high Reynolds numbers. However, from a study of the literature, it appears that no other researchers show agreement for the $Re = 9500$ case beyond $T = 3.0$. Shankar (1996) does show excellent agreement with the results of Koumoutsakos and Leonard (1995) but does not present results beyond $T = 3.0$. Further, the earlier results of Koumoutsakos (1993) are not symmetric beyond $T = 3.0$. This indicates that even while ensuring overlap and using a deterministic diffusion scheme with a large number of particles (Koumoutsakos (1993) used around half a million particles at $T = 4$), the results beyond $T = 3.0$ are not symmetric. Therefore, it appears that the difficulty of the $Re = 9500$ case is due to the inherent instability of the flow and not necessarily related to particle overlap. It is to be noted that in section 8.1.5 it is shown that with a careful choice of parameters through numerical experimentation, it is possible to obtain fair agreement even when $T > 3.0$ without ensuring explicit particle overlap. Further research would be necessary to make any conclusive comments on the matter.

In the present work, the no-slip boundary condition is satisfied by releasing vortex sheets to offset the slip. Even at low Reynolds numbers ($Re = 40$) the results of the present work agree well with established results. The results of Shankar (1996) also suggest that this approach of satisfying the boundary condition is adequate for the problem being studied.

The random vortex method is stochastic and produces noisy results. The present work uses a reasonable amount of smoothing to remove this noise. The numerical parameters are chosen based on the parametric study made in chapter 7. As discussed in section B.1.5, the vorticity contours are smoothed optimally based on the work of Fogelson and Dillon (1993). The noise is also reduced by ensembling the results of several trial runs. The results of the present work clearly indicate that by using these smoothing techniques, the random vortex method is capable of producing results that are comparable with those of other schemes. The reason why the present work succeeds where others failed is because a larger number of particles are used. Earlier computations used much fewer particles. The use of merging and annihilation allow for an effective utilization of the existing particles. The new variance reduction scheme also helps improve the accuracy significantly. Most of the well-known simulations for the flow past an impulsively translated cylinder using the RVM were performed in the 80's. The computational resources available then were nowhere near as plentiful as today.

Koumoutsakos and Leonard (1995) compare their results for the $Re = 1000$ case with those of Smith and Stansby (1988) and the agreement is very poor. However, as shown in Fig. 8.15, the agreement with the results of Smith and Stansby (1989) is much better. Both the computations of Smith and Stansby (1988, 1989) appear to use fewer particles than necessary. The results of section 7.5.2 and in particular Table 7.5 demonstrate that increasing the size of the blobs (by increasing k) produces increasingly erroneous results. Thus, if the size of the blob is chosen such that the necessary scales are resolved, then the RVM does indeed perform well.

The random walk method does have its share of problems. Most notable is the noise inherent in the method. The noise necessitates the use of smoothing. The smoothing does introduce some side effects. Most notable is the reduction in the magnitude of the peaks in the curves for the vorticity flux. For example, this is seen in the curves presented in 8.26. It is possible to reduce the amount of noise by using an optimal time step and using a larger number of particles or performing several trials and ensembling the results. The results presented in this chapter only use a fairly small amount of smoothing since the results are obtained by ensemble averaging 8 trials. While the noise is a problem it certainly does not preclude the possibility of performing high-resolution computations.

The use of vortex sheets introduces a large amount of complexity in the implementation. The conversion of blobs to sheets and vice-versa and the need to carefully choose different sheet related options is an additional problem. For computational convenience it is necessary to use sheets of the same size. If this were not the case, the complexity of the code would increase quite substantially for general flows. Multiple bodies also pose problems because of the issues mentioned in section 7.5.1. Specifically, there is a problem with conserving the total circulation. Conservation of the total circulation requires that the body be imparted a spin. It is not clear how this can be handled for multiple bodies with the current implementation of sheets having localized velocity fields. Therefore, it appears that using a sheet with a global velocity field and one that satisfies the NS equations instead of the Prandtl equations is necessary. An alternative would be to not use sheets at all. Further research is necessary to explore these options for the random vortex method.

The RVM also poses problems with randomly moving particles in the vicinity of complex bodies. The present works develops an efficient algorithm in section 5.1 to handle this.

The need to use a large number of particles to reduce the noise might be at first perceived as a significant disadvantage of the RVM. However, there are some points to consider. The number of particles used in the present work are comparable to those used by the PSE. For example, the runs presented in section 8.1.4 for $Re = 3000$ used around 125000 blobs and 2900 sheets at the end of $T = 6$ in each trial run. Eight trials were made and the results of these ensembled. For the same case, Koumoutsakos and Leonard (1995) use around 300000 particles at $T = 6$. It may be argued that the present work uses a total of 1 million particles to compute the ensemble. However, this would be a naive comparison. The RVM enables for a trivial parallelization. Thus, on a cluster of 8 desktop machines the present results achieve a *linear* scale-up. That is, the time taken for 8 runs is almost the same as that taken by one individual run. Further, very little special programming is necessary to enable this. Eight different seeds are considered and each processor merely runs a serial code. In the case where the new variance reduction scheme (see section 7.8) is used, a simple program is used to communicate and merge the results every n_{sync} time steps. Thus, the eight trial runs were made in parallel on a loosely-coupled cluster of desktop machines with a minimal amount of effort. Therefore, this computation cannot be equated to that of a parallelized vortex method code using one million particles running on a supercomputer. Further, as seen from the results in section 7.7.2, it is possible to obtain good results with a fewer number of trials. For example, four runs would produce similar results and reduce the total particle count. It is also possible to halve the number of particles by increasing γ_{max}.

Fig. 8.39 plots the drag coefficient versus T for $Re = 3000$. Two processors are used along with the new ERVM. $\Delta t = 0.025$ ($C = 6.35$), $k_1 \approx 0.022$ $\gamma_{max} = 0.05$, $R_a = 0.15$ and $R_m = 0.3$. The frequency of synchronization between trials, n_{sync} is 5. The total number of particles at the end of $T = 6$ is around 95000 blobs and 2400 sheets. The average number of particles is 65000 blobs and 2800 sheets. The total run time on a pair of Pentium IV machines (one running at 1.7Ghz and the other at 1.3Ghz) was 107 minutes. The agreement between the present computation and other results is very good. The number of particles used is comparable to that used by Ploumhans and Winckelmans (2000) with the non-

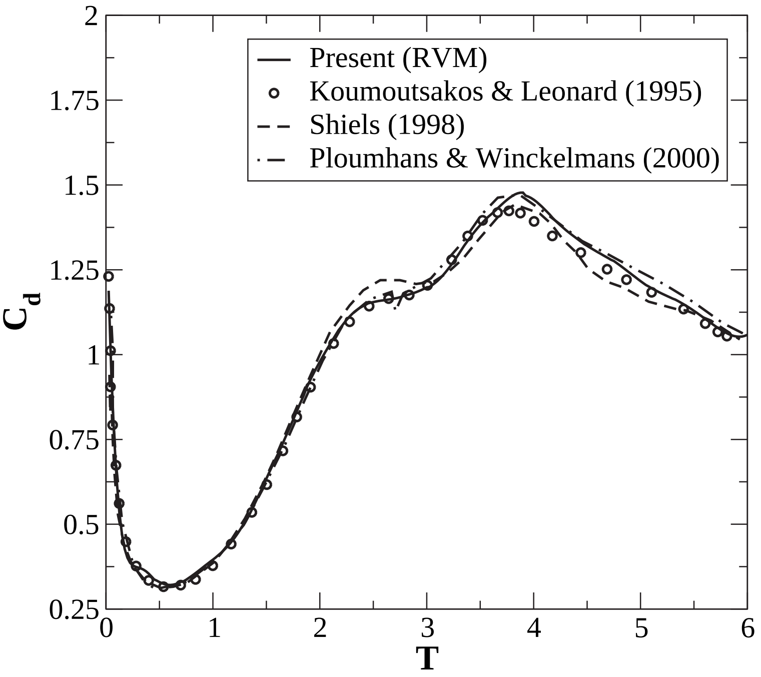

Figure 8.39: Drag coefficient C_d versus T for impulsively translated cylinder at $Re = 3000$. $n_{proc} = 2$, $n_{sync} = 5$ and $\gamma_{max} = 0.05$.

uniform resolution scheme at the same time. Thus, it is clear that the number of particles necessary is similar to that used in computations employing the PSE.

Shiels (1998) presents results for the drag coefficient computed using the corrected core-spreading vortex method (CCSVM) (Rossi, 1996) for $Re = 3000$. While fewer particles are necessary, the time step needs to be small ($\Delta t = 0.005$) for agreement with the results obtained using the PSE. As larger time steps are chosen there is a fair amount of variation in the drag coefficient. As seen in Fig. 8.39, the present results are as good if not marginally better than his results. It is also to be noted that the time step used for the RVM is 0.025 whereas the CCSVM appears to require a value that is five times smaller. Thus, these results show clearly that the computations using the RVM and those employing the PSE or the CCSVM are comparable in terms of both accuracy and efficiency.

A comparison of the results at $Re = 9500$ is a little more complicated due to the difficulty in maintaining symmetry beyond $T = 3$. The lack of data from other schemes beyond this time is also a problem. Koumoutsakos and Leonard (1995) use around 1 million particles at the end of $T = 6$. The present computations require around 1.5 million particles in the ensemble. It seems reasonable to assume that the linear speedup of the present method coupled with the slightly higher time step makes the present computations as efficient as those of Koumoutsakos and Leonard (1995).

Shankar (1996) employs the vorticity redistribution scheme for diffusion in his computations. For the $Re = 9500$ case he obtains converged results at $T = 3.0$ with just 60000 vortices. The computations of the present work require about 140000 particles per trial, at the same time. However, the present work goes on to obtain nearly symmetric results all the way up to $T = 6$. The VRT of Shankar (1996) is also a little more complicated to implement than the RVM and is more computationally expensive. He also uses a fourth order Runge-Kutta integration scheme with a smaller $\Delta t = 0.01$. Therefore the VRT appears to take the same computational effort that one trial would take using the RVM.

It is to be noted that the focus of the present work is not to minimize the number of particles used to obtain results but to show that high-resolution results are possible with the RVM. The fact that the RVM is trivially parallelizable on commodity computers is an interesting aspect of the method.

It is not the intention of this work to claim that the RVM is a superior method to the deterministic schemes. However, the results obtained here clearly demonstrate that the RVM is certainly a usable method for high-resolution computations. The RVM does perform quite favorably when a fair comparison is made with most deterministic diffusion schemes.

In the next chapter the contributions made in the present work are summarized and conclusions are drawn. Suggestions for future research are also made.

CHAPTER 9

SUMMARY AND CONCLUSIONS

In this work, a fairly general random vortex method based solver has been developed. The resulting code can be used to simulate two-dimensional, incompressible, viscous fluid flows. Much attention has been paid towards an efficient implementation. This chapter summarizes the important contributions made in the present work. Conclusions based on the work are presented followed by a list of suggestions for future work.

9.1 Contributions made

The key contributions of the present work are listed below.

- An accurate cubic panel method to eliminate the edge effect and satisfy the no-penetration boundary condition is developed.
- An original implementation of the adaptive fast multipole method (AFMM) (Carrier *et al.*, 1988) is provided.
- A generalization of the AFMM to handle passive particles efficiently is proposed.
- The AFMM is extended to accelerate the computation of the velocity field due to linear and higher order vortex panels.
- Anderson's FMM without multipoles (Anderson, 1992) is implemented in the context of the AFMM. This method is compared with the AFMM extended to higher order panels.
- An efficient fast algorithm to handle randomly moving particles in the presence of complex geometries is developed. A simple idea to characterize the complexity of a geometry is also proposed.
- Annihilation of oppositely signed vortex particles is used as a means to improve accuracy and computational efficiency. Merging of like signed vorticity is also used to reduce the number of particles.

- An object oriented design for vortex based flow solvers is developed and discussed.
- A study of the connection between the various parameters used in the RVM is carried out. Several recommendations on the optimal choice of these parameters are made.
- A simple and new variance reduction scheme is introduced. The resulting method is called the Ensembled RVM (ERVM). The method takes advantage of the inherent parallelism of the RVM. It is also fairly easy to implement.
- The flow past an impulsively started cylinder is simulated. An exhaustive comparison of the results are made with various high-resolution simulations that use deterministic diffusion schemes. It is clearly demonstrated that the RVM can be used to perform high-resolution simulations.

9.2 Conclusions

The significant conclusions that can be drawn from the present work are as follows.

- The random vortex method can certainly be used for high-resolution simulations provided due care is taken to resolve fine scale features. Despite being a stochastic method and requiring a large number of particles, the results obtained are quite comparable to most modern deterministic schemes in terms of accuracy and efficiency. The RVM is completely grid-free and relatively easy to implement. The method does have its share of problems as discussed in section 8.2. However, the present work clearly demonstrates that it is capable of high-resolution simulations. The number of particles is comparable to the numbers used by computations employing the particle strength exchange method (PSE) (Koumoutsakos and Leonard, 1995; Ploumhans and Winckelmans, 2000). For the same accuracy, it appears that the RVM is at least as efficient as the corrected core spreading model (Rossi, 1996; Shiels, 1998). However, the vorticity redistribution technique (Shankar, 1996; Shankar and van Dommelen, 1996*a*) requires much fewer particles at a slightly increased computational cost.
- The use of ensemble averaging in the random vortex method is important. A single trial can be inconclusive. Ensembling the outcome of several trials is an effective means to obtain more reliable results.
- The use of ensemble averaging allows for a trivial parallelization. This aspect appears to have been unexplored in the past. A simple and new variance reduction scheme is introduced that allows for a significant reduction in the error. This method is also inherently parallel in nature and quite easy to implement.

- For the problem considered, at Reynolds numbers less than 9500, good results were obtained by using the RVM along with routine ensemble averaging. In chapter 8, the results for the $Re = 40, 550$ and 1000 cases were obtained without the use of the ERVM. The results in chapter 7 indicate that this approach is sufficient for $Re = 3000$ also. However, for the $Re = 9500$ case, obtaining a symmetric flow beyond a time of $T = 2.5$ is very difficult. The use of the ERVM for this case becomes extremely important.

- As with any vortex method, it is imperative to use fast algorithms to make the computations efficient.

- When the RVM is used, it is important to optimally smooth (Fogelson and Dillon, 1993) the vorticity contours as discussed in section B.1.5. The curves obtained for the various diagnostic quantities also require smoothing. Some techniques for this are discussed in section B.1.2.

- The annihilation of oppositely signed vorticity is extremely important in an implementation of the random vortex method. Merging of like signed vortices is also of use but is not as critical as that of oppositely signed vorticity. In section 5.4.2 and section 7.5.4 it was shown that an *order of magnitude* reduction in the number of particles is possible if annihilation is performed. This is a significant improvement.

- The implementation of vortex sheets leads to several complications. The sheet-blob conversions are problematic. Sheets also introduce complications in ensuring that the total circulation in the flow is conserved. Using sheets of different lengths also introduces implementation difficulties. The sheets also introduce a number of parameters that need to be chosen carefully.

- The AFMM is a fairly versatile fast algorithm that has a very important place in vortex methods. By introducing "cause" and "effect" particles it is possible to elegantly generalize the method to handle passive particles. Decomposing the domain of interacting particles into causes and effects is a powerful and general multi-scale modeling concept that clearly goes beyond the AFMM. This idea is used in the implementation of *every* fast algorithm developed in the present work.

- Object-oriented design and analysis enables one to develop a large volume of powerful, readable, maintainable and extendible code. The design also enables for a clearer understanding of the algorithms. This directly results in elegant and powerful abstractions and significant amounts of code-reuse.

In section 7.7.2, several recommendations were made regarding the implementation of the RVM. The most significant of these are summarized below.

- A second order integration scheme is optimal and allows for larger time steps without increasing the noise levels.

- Strang type viscous splitting allows for larger time steps than the standard viscous splitting approach. For the same time step, a small performance penalty exists. However, the feasibility of larger time steps more than offsets this cost.

- Choosing too small a time step ($C < 1$) results in an increased noise in the results. In practice it is seen that choosing larger time steps with $C \leq 10$ works well.

- γ_{max} should be chosen such that the cell Reynolds number, $Re_h = \gamma_{max}\lambda/\nu$, be $O(1)$ or less. Reducing γ_{max} as much as possible is also recommended.

- The length scale of the vortex blobs, k_1 can be chosen such that the numerical layer height is less than 10% of an estimated maximum boundary layer height. Values of k_1 around 0.025 produce very good results.

- Sheet release style 3 (section 3.4.1) appears to produce the best results at little additional computational expense.

- Changing the sheet type between Sheet1 and Sheet2 (section 2.4) does not produce any significant improvements in the results. Sheet tagging also does not appear to be beneficial.

9.3 Suggestions for future work

There are several issues worth exploring in the future with the current code as a basis. These are listed below.

- The Ensembled RVM was introduced in section 3.9 and studied numerically in section 7.8. No theoretical analysis of the method is performed in this work. This appears to be an interesting area of study since the method is likely to be applicable to any Monte-Carlo technique.

- Vortex sheets introduce many difficulties in the implementation. It would be of interest to see if similar results are achieved without the use of vortex sheets. Several anisotropic vortex elements have been also developed and studied by researchers (Teng, 1982; Huyer and Grant, 1996; Bernard, 1995; Marshall and Grant, 1996; Summers, 2000). It would be of interest to compare these techniques with the traditional sheet method and choose the best of these in terms of efficiency and accuracy.

- The errors introduced by annihilation need to be carefully studied. More sophisticated merging and annihilation algorithms can be used.

- If the body geometry deforms in time, it is necessary to solve the system of equations used in the panel method at each time step. Iterative matrix solution techniques should be considered to improve efficiency. The iterative solvers should be able to handle complex geometries.

- The present mechanism for generating panels on the surface of the body to satisfy the no-penetration condition is non-adaptive and requires the user to generate the panels. An adaptive panel method that automatically adds and removes panels given an arbitrary geometry and a desired accuracy would be useful.

- Deterministic diffusion schemes could be implemented and compared directly to the RVM. The VRT (Shankar and van Dommelen, 1996*a*) and corrected core spreading (Rossi, 1996) model are of particular interest due to their grid-free nature.

- If higher order accuracy for long times is desired, the vorticity field needs to be periodically re-meshed in order to ensure particle overlap. No re-meshing has been performed in the present work. It is unclear as to how re-meshing is possible or of any use in the context of the RVM. This requires careful investigation. For higher order accuracy it is also of considerable interest to implement the fast adaptive vortex method of Strain (1997).

- The vorticity field in the form of particles is interpolated onto a regular grid as described in section B.1.4. Using a regular grid is expensive in terms of memory requirements. Interpolating this vorticity on an adaptive Cartesian mesh or an unstructured grid would be more efficient. One possible approach of doing this is to use the technique described by Strain (1996, 1997).

- It is of interest to apply the ideas of verification for CFD solvers (refer Roache (1998)) to vortex methods. This would allow for systematic testing of vortex method implementations.

- Optimization of several of the algorithms is of considerable practical use. Some of the techniques mentioned in Russo and Strain (1994) and also in section A.3.6 can be used to accelerate the AFMM.

- Fully parallelizing the algorithms developed is imperative if larger problems are to be handled.

The code developed in the present work can be used as a basis to explore these possibilities.

APPENDIX A

THE FAST MULTIPOLE METHOD

In the context of vortex methods, the fast multipole method (FMM) is a fast algorithm that can be used to compute the velocity or potential due to a collection of vortex particles. For N vortex blobs, a direct velocity computation requires an $O(N^2)$ computation. The FMM allows this to be computed in an $O(N)$ or an $O(N \log N)$ number of operations.

There are various flavors of the FMM. In this appendix, a brief review of the various flavors of fast algorithms is first provided. Subsequently, the $O(N \log N)$ body-cell treecodes are described. The adaptive fast multipole method (AFMM) of Carrier *et al.* (1988) is then described in considerable detail. A simple approach to compute the X_b list is also developed. Pseudo-code is provided for greater clarity.

A.1 Background

The Cloud-In-Cell (CIC)/Vortex-In-Cell (VIC) technique (Christiansen, 1973; Smith and Stansby, 1988; Tryggvason, 1989; Smith and Stansby, 1989; Cottet, 1990) was the earliest scheme used to obtain velocity fields rapidly. In these class of methods the vorticity field is interpolated onto a grid. A fast Poisson solver is used to obtain the stream function from which the velocity field is computed and interpolated back to the particle positions. The method requires an $O(N) + M \log(M)$ work, where M is the number of nodes in the grid. Anderson's method of local corrections (Anderson, 1986) extends this technique so that it may be used with higher order accurate vortex blobs. The difficulty with these methods is the requirement of a fixed grid.

The first of a class of grid-free, tree-based, techniques was developed by Appel (1985) and was used to accelerate the N-body gravitational problem. These class of methods are called *treecodes* and employ a tree data structure to organize clusters of interacting particles. This method was also developed independently by Barnes and Hut (1986). Treecodes hierarchically organize clusters of particles and introduce particle-cluster interactions to reduce the computational cost. If the distance between a cluster and a particle is "large enough", then the interaction can be performed between the cluster as a whole and the particle. For particles that are nearby, the interactions are performed directly. This reduces the computational cost to $O(N \log N)$. Using the terminology of Salmon and Warren (1994), these methods are called *body-cell* treecodes. Compared to the direct $O(N^2)$ algorithm, which is trivial to program, a significant amount of programming effort is required to implement treecodes.

The fast multipole method (FMM) due to Greengard and Rokhlin (1987) is an $O(N)$ treecode based on the work of Rokhlin (1985). This method uses multipole expansions to represent the potential of clusters of charges accurately. The method reduces the operation count further by introducing cluster-cluster interactions in addition to particle-cluster interactions. It is ideally suited for uniform distributions of particles and can provide a highly accurate approximation to the velocity field. The adaptive fast multipole method (AFMM) of Carrier *et al.* (1988) is an adaptive $O(N)$ algorithm that works efficiently for both non-uniform and uniform particle distributions. This method introduces interactions between clusters that are reasonably close to each other but are of different sizes. This makes the algorithm more complicated but highly efficient. Such methods that employ cluster-cluster interactions are called *cell-cell* treecodes (Salmon and Warren, 1994).

There are several different flavors of treecodes apart from the ones mentioned above. Van Dommelen and Rundensteiner (1989) develop an $O(N \log N)$ algorithm using a Laurent series to represent the clusters of vortices. Anderson (1992) develops an interesting technique called the "fast multipole method without multipoles". This method uses Poisson's integral formula in order to obtain equivalents for the multipole expansion. The method is presented in the framework of a multigrid algorithm and can be used in both two and three-dimensions. Salmon and

Warren (1994) discuss body-cell treecodes and study the worst case error introduced by these schemes. They propose new criteria to find sufficiently far away clusters and develop an efficient and accurate technique to adaptively traverse the tree structure. Draghicescu and Draghicescu (1995) develop an algorithm to accelerate vortex blob interactions without assuming that the behavior of the the vortices far away from the center is like that of the point vortex. They use Taylor expansions specific to the particular velocity kernel being used in order to represent clusters of particles. Lustig *et al.* (1995) use Chebyshev economization to represent the multipoles in a more efficient manner and thereby improve computational efficiency of the AFMM (Carrier *et al.*, 1988). Makino (1999) develops another multipole-without-multipoles algorithm by using pseudo-particles to express the potential field of a cluster of particles. Vosbeek *et al.* (2000) present a hierarchical element method for use with contour dynamics (Zabusky *et al.*, 1979) simulations. Their method is an extension of Anderson's scheme (Anderson, 1992). They derive expressions for the Poisson integrals for the velocity field and also provide error bounds for the expressions.

The algorithms mentioned thus far are used to evaluate discrete sums of the free-space Green's function (or its derivatives). For the Green's function inside a cube in two or three dimensions, Strain (1992) develops fast algorithms to evaluate discrete sums and evaluate layer potentials on the boundary of a domain. This brief survey clearly indicates that a large number of fast algorithms have been developed and used.

Of the methods discussed above, the present work implements the adaptive fast multipole method (AFMM) of Carrier *et al.* (1988). While this algorithm is fairly complex to implement it is very efficient. The basic ideas used in the AFMM are discussed in greater detail in the following.

A.2 Body-cell treecodes

The original $O(N \log N)$ body-cell treecodes like the one developed by Barnes and Hut (1986) are discussed first since they are much easier to understand. In the following, a simple two-dimensional treecode is developed to compute the velocity field for vortex blob interactions. After this the AFMM algorithm is explained.

The general structure of the algorithm is as follows. A tree structure is built that identifies clusters of particles and organizes them in the tree. Multipole expansions for each cluster are constructed efficiently. Each particle is considered and the effect of all clusters is computed on the particle to obtain the total velocity on the particle. Each of these steps is discussed in the following sections.

A.2.1 Generating the tree

The hierarchy of particles is first generated in the following manner. A square region that contains all the interacting particles is identified. This is called a *cell* (or box) at level 0. This cell is divided or split into 4 children and these cells are called *daughter* or *child* cells. The four children generated from the level 0 cell are at level 1. A cell that has been split is called a *parent*. In general, a cell is split if it has more than n_p particles in it. Cells of the level $l+1$ are obtained by splitting each cell at level l with more than n_p particles into 4 children. This process is repeated till there are no more than n_p particles per cell. The non-empty, un-split cells are called *childless* or *leaf* cells. In this manner the collection of particles is distributed among a set of childless cells that are hierarchically organized in a quad-tree structure. Many standard tree-based methods use a binary-tree. A quad-tree is used here since the intention is to finally develop the AFMM, which uses the same structure.

The following definitions are useful to note.

Definition A.1 *A cell c is defined to be "adjacent" to another cell b if they share a side or corner.*

Definition A.2 *A "colleague" of a cell c at level l is defined as a cell at level l which is adjacent to c.*

Definition A.3 *An "associate" of a cell c at level l is defined as either a colleague* *of c or a childless cell at level less than l and adjacent to c.*

Clearly, the set of colleague cells is a subset of the set of associate cells. Also note that if an associate of a cell c is not a colleague, then it is childless and larger than c.

Once the clusters of particles are hierarchically organized into cells, the multipole expansion of the velocity due to the particles in each cell is to be computed.

A.2.2 Multipole expansions

The velocity at a point z, due to a point vortex having strength (circulation) Γ_1 and located at a position z_1 is given by,

$$V(z) = \frac{-i\Gamma_1}{2\pi(z - z_1)}.$$

This can be written as,

$$\begin{aligned} V(z) &= \frac{-i\Gamma_1}{2\pi((z - z_0) - (z_1 - z_0))} = \frac{-i\Gamma_1}{2\pi(z - z_0)}\left(1 - \frac{z_1 - z_0}{z - z_0}\right)^{-1} \\ &= \frac{-i}{2\pi}\sum_{j=1}^{\infty}\frac{a_j}{(z - z_0)^j} \quad ; \quad a_j = \Gamma_1(z_1 - z_0)^{j-1}, \end{aligned}$$

where z_0 is an arbitrary point. This summation clearly converges when z is such that $|z - z_0| > |z_1 - z_0|$. Depending on z, z_0 and z_1, the expansion can be truncated to p terms for a specified accuracy. This is the multipole expansion for a single point vortex.

Figure A.1 illustrates a circle C of radius R centered at z_0 containing m particles at z_j and having strengths Γ_j. The velocity induced by these particles at a point z can be given as,

$$V(z) = \sum_{j=1}^{m}\frac{-i\Gamma_j}{2\pi(z - z_j)}.$$

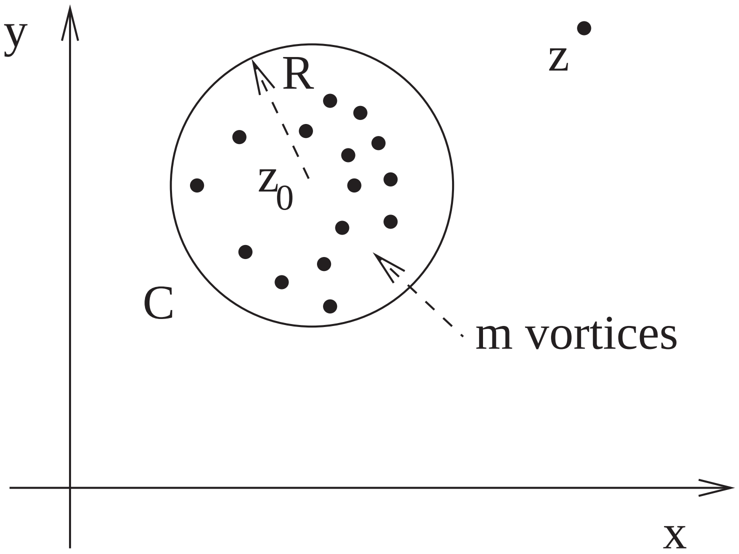

Figure A.1: A cluster of m point vortices considered for the multipole expansion.

The multipole expansion for this cluster of particles is given by,

$$V(z) = \frac{-i}{2\pi} \sum_{j=1}^{\infty} \frac{a_j}{(z-z_0)^j} \quad ; \quad a_j = \sum_{k=1}^{m} \Gamma_k (z_k - z_0)^{j-1} \tag{A.1}$$

This summation clearly converges for z outside the circle C. The summation can be truncated to p terms and the truncation error is bounded as,

$$\left| V(z) - \frac{-i}{2\pi} \sum_{j=1}^{p} \frac{a_j}{(z-z_0)^j} \right| \leq \frac{A}{c-1} \left(\frac{1}{c} \right)^p \tag{A.2}$$

where

$$c = \left| \frac{z}{R} \right| \quad ; \quad A = \frac{1}{2\pi} \sum_{j=1}^{m} |\Gamma_j|. \tag{A.3}$$

This error term is quite easy to derive[1]. The specialty of equation (A.1) is that the coefficients a_j are obtained by a summation of the coefficients due to individual blobs. From equation (A.2) it is easy to see that if $c \geq 2$, then ensuring an accuracy of ϵ requires the use of $p \geq -\log_2 \epsilon$ terms in the series.

---

[1] A more detailed analysis of the various error terms discussed in these sections is given in Greengard and Rokhlin (1987)

A.2.3 Shifting multipole expansions

A multipole expansion of a cluster of particles in the circle C, with center z_0, can be shifted to the origin resulting in the following,

$$V(z) = \frac{-i}{2\pi} \sum_{j=1}^{\infty} \frac{b_j}{z^j} \;\; ; \;\; b_j = \sum_{k=1}^{j} a_k \left(\begin{array}{c} j-1 \\ k-1 \end{array} \right) z_0^{j-k}. \tag{A.4}$$

This expression converges outside a circle centered at the origin with radius $R+ \mid z_0 \mid$. It can be used to shift the center of the multipole expansion from the center z_0 to any other point z_c by replacing z and z_0 with $z - z_c$ and $z_0 - z_c$ respectively. The truncation error for this is bounded as,

$$\left| V(z) - \frac{-i}{2\pi} \sum_{j=1}^{p} \frac{b_j}{z^j} \right| \leq \frac{A|z|}{|z| - |z_0| - R} \left| \frac{|z_0| + R}{z} \right|^{p+1}, \tag{A.5}$$

where A is as given in equation (A.3).

Hence, the multipole expansion for a cluster can be readily computed using equation (A.1) and then transfered from one center to another using equation (A.4).

Consider two clusters of particles enclosed within circles of the same radius, R. Let the center of these circles be separated by a distance of $3R$. Now, if the multipole expansion of the cluster of particles inside one circle is evaluated at any point in the other circle, it is guaranteed that $c \geq 2$, where c is given in equation (A.3). Such clusters of particles are called "well separated". Let one cluster have n_1 particles and the other have n_2 particles. The cost of computing the velocity due to the clusters on each other directly without the use of the multipole expansion is $O(n_1 n_2)$. Obtaining the multipole coefficients of p terms for each cluster would require $n_1 p + n_2 p$ computations. Then the interaction of each cluster on the other can be computed using the multipole expansions with $(n_1 + n_2)p$ computations. This results in an $O((n_1 + n_2)p)$ computational cost. Clearly, when n_1 and n_2 are large it is more efficient to compute the interactions using the multipole expansions. This idea is used systematically in treecodes.

A.2.4 The body-cell algorithm

The body-cell algorithm proceeds in three steps.

1. Generate the quad-tree using the algorithm described in section A.2.1.
2. Compute the multipole expansion for each cell.
3. Find the velocity at each particle.

Once the tree is generated, the multipoles for each cell are computed as follows. Let the total number of levels in the tree be n_{level}. For each childless (leaf) cell, compute the multipole expansion due to all the vortices in it about the center of the cell using equation (A.1). Transfer the multipole of each cell at a particular level to its parent cell using equation (A.4) and add the resulting coefficients to the parent's multipole coefficients. Keep adding the multipole coefficients from the children to the respective parents till level 2 is reached. This level will have at most 16 cells. Note that level 1 and level 0 have cells that are not well separated and therefore there is no point in computing the multipole expansion of these. Once this step is completed, all relevant cells will have multipole expansions for the respective particles that they contain. This completes the second step in the above sequence.

The above procedure is described in algorithm A.1. The Python programming language (van Rossum *et al.*, 1991–) is used to express the pseudo-code for the algorithms[2].

Algorithm A.1 ComputeMultipoles()

```
for cell in childless_cells :
    # compute multipole expansions of particles inside cell .
    cell .computeMultipoleCoefficients()
for level in range(n_level − 1, 1, −1):
    # for each cell ' cell ' in all cells at level ' level '
    for cell in cells [ level ]:
        for child in cell . children ():
            # transfer multipole from child and add to parent.
            child . transferMultipole( cell )
```

---

[2] Python is object oriented, highly expressive and freely available. This makes the pseudo-code relatively easy to understand while being expressive enough to explain the algorithms.

Definition A.4 *Consider a cell c centered at z_c, the radius of the cell, R_c, is defined as the radius of the circumcircle of the cell.*

Definition A.5 *A point z is considered "well separated" from the cell, c, centered at z_c, if $|z - z_c|/R_c \geq 2$.*

Definition A.6 *Two cells of radius R, centered at z_b and z_c respectively are said to be "well separated" if $|z_b - z_c| \geq 3R$.*

To compute the velocity, the following procedure is used. Consider each childless cell. Consider each particle of the cell. Let the position of the particle be z. Now, consider each cell in level two[3]. If the cell is well separated from z (Definition A.5), then compute the effect of this cell on the point using the multipole expansion. If the cell is not well separated, repeat the check for each of its children. If in this process one finds leaf/childless cells that are not well separated from z, then they are clearly too close to z and their influence on z is to be computed directly. This process is repeated for all the particles. Note that even the childless cell containing the particle will be traversed in this process. At the end of this procedure the velocity at each particle is known. One important restriction to note is that the minimum cell size should be such that the core radius of the blobs do not exceed the cell size. If this were to happen, the velocity due to the blobs will not behave like point vortices on well separated cells and the multipole expansion evaluation will be inaccurate. The pseudo-code for this algorithm is presented in algorithms A.2 and A.3. Algorithm A.2 is evidently a recursive one.

As can be seen above, the algorithms are fairly easy to understand and program. This approach of computing the velocity is generally how body-cell treecodes work. The definition of well-separatedness depends on the implementation (see Salmon and Warren, 1994). Instead of using multipole expansions it is also possible to also use other schemes. For example, Anderson (1992) uses Poisson's integral formula and Makino (1999) uses pseudo-particles.

---

[3] In general there will be more than 2 levels in the tree, for if there are less than 2 levels, there are not enough particles for a tree algorithm to be worthwhile

Algorithm A.2 ComputeCellVelocity(c, p)

```
# 'c' is a cell and 'p' is the particle at which the velocity is computed
    v = 0
    if abs(c.center() − p.position()) >= 2.0*c.radius():
        v = c.multipoleExpansion(p)
    elif c.isParent():
        for child in c.children():
            v += ComputeCellVelocity(child, p)
    else:
        v = c.directVelocity(p)
    return v
```

Algorithm A.3 ComputeVelocity()

```
    for cell in childless_cells :
        for p in cell.particles():
            # for cell 'b' in cells at level = 2
            for b in cells [2]:
                p.velocity += ComputeCellVelocity(b, p)
```

These fast algorithms are much faster than the $O(N^2)$ direct method when N is large. The advantage with these methods is that due to their simplicity they are easy to implement and parallelize. However, they are $O(N \log N)$ schemes and are not as efficient as the $O(N)$ AFMM.

A.3 Adaptive Fast Multipole Method (AFMM)

The adaptive fast multipole method (AFMM) is a cell-cell treecode. The collection of particles is organized into a hierarchical structure as done for the body-cell treecodes. The significant difference is that the AFMM uses cluster-cluster, particle-cluster and particle-particle interactions. The $O(N \log N)$ body-cell treecodes only use particle-cluster and particle-particle interactions. For each particle in a given childless cell the entire tree is traversed to search for well separated cells. This is optimized in the AFMM by considering cell-cell interactions. This results in a more complex but highly efficient algorithm. The key idea used is the transfer of the multipole expansion of a cell c, to a local expansion about the center of a well separated cell, b.

A.3.1 Local expansions

Given a multipole expansion (equation (A.1)) for particles inside a circle C of radius R and centered at z_0 such that $|z_0| > (c+1)R$ with $c > 1$, the multipole expansion can be described by a power series in z that converges inside a circle, C_1, centered at the origin, having radius R,

$$V(z) = \sum_{j=0}^{\infty} b_j z^j \tag{A.6}$$

where,

$$b_j = \frac{-i}{2\pi z_0^j} \sum_{k=1}^{\infty} (-1)^k \frac{a_k}{z_0^k} \begin{pmatrix} j+k-1 \\ k-1 \end{pmatrix}, \tag{A.7}$$

where a_k is given in equation (A.1). Note that if $c \geq 2$, the circles C and C_1 are well separated (Definition A.6). When equation (A.6) is truncated to p terms the error in the approximation is bounded by,

$$\left| V(z) - \sum_{j=1}^{p} b_j z^j \right| \leq \frac{A(4e(p+c)(c+1)+c^2)}{c(c-1)} \left(\frac{1}{c}\right)^{p+1}, \tag{A.8}$$

where A is given in equation (A.3), e is the base of natural logarithms and $p \geq \max(2, 2c/(c-1))$. The derivation of this error term is a little involved and is worked out by Greengard and Rokhlin (1987). Using the above, it is possible to compute the interaction between two well separated clusters. Instead of traversing the entire tree for each particle and finding well separated cell interactions, the local expansion is found and evaluated at each particle to obtain the velocity.

It is to be noted that if the multipole expansion (equation (A.1)) is for a particle that is exactly at z_0, then a_k is non-zero only when $k = 1$. In this case equation (A.7) simplifies considerably. Given this, the following definition proves useful subsequently.

Definition A.7 *Consider a cell c and another cell b. If the multipole expansion of each particle in c is expressed about the particle's position and converted to a local expansion in b then such a local expansion is called a "special local expansion".*

A.3.2 Shifting local expansions

Given any local expansion in a cell of radius, R centered at z_0, and coefficients a_k, the local expansion center can be transferred to the origin as,

$$\sum_{j=0}^{n} a_j (z - z_0)^j = \sum_{j=0}^{n} z^j \sum_{k=j}^{n} a_k \left(\begin{array}{c} k \\ j \end{array} \right) (-z_0)^{k-j} \tag{A.9}$$

This expression is exact and there is no error involved in the change of the center of a local expansion. This transfer is only performed from a parent cell to a child and therefore the transferred local expression will converge inside the child.

A.3.3 General idea

As seen earlier, the central idea used in treecodes is to represent clusters of particles as single computational units and define cluster-particle interactions in addition to particle-particle interactions. The AFMM additionally employs cluster-cluster interactions to accelerate the computations. Due to the adaptive nature of the mesh there are different flavors of cluster-cluster interactions. Consider two cells b and c. Let b be the childless cell on which the velocity due to cell c is computed. The important cases are categorized below.

- b and c are adjacent.
    - c is childless. These interactions must necessarily be computed directly and no multipole acceleration is possible.
    - c has children. The children must be considered for possible interactions with b in a similar manner.
- b and c have the same size and are well separated (Definition A.6). In this case one can compute the interaction of c on b by converting the multipole expansion of c as a local expansion on b. A hierarchy of well separated cells can also be handled using a hierarchical application of the same idea and making use of equation (A.9) to translate the local expansions.
- b is larger than c and not adjacent to it. In this case, due to the nature of the construction of the cells, it can be seen that any particle inside b is well separated from c (Definition A.5). Consequently, the effect of cell c can be found by evaluating the multipole expansion of c on each particle of b.

- b is smaller than c which is not adjacent to it and is childless. In this case it is possible to evaluate the effect of the local expansion of each particle in c about itself as a local expansion on b. This is the special local expansion as defined in A.7.

It is therefore clear that the following are essential in order to compute the velocity field using the AFMM,

- direct computation of the velocity field;
- evaluation of the multipole expansion about a given center (equation (A.1));
- transferring the multipole to a different center (equation (A.4));
- conversion of the multipole expansion to a local expansion about another center that converges in a radius around the new center (equation (A.6));
- transfer the local expansion to another center (equation (A.9)).

A.3.4 Definitions and notation

The particles in the domain are hierarchically organized as described in section A.2.1. The definitions of various cells noted in that section are of importance in the following discussion. The colleagues and associates of a cell b are illustrated in Figure A.2. The cell marked a is an associate cell. Recall that a colleague cell is also an associate cell.

For any cell b at a level l, 5 different lists are associated with it. The definition of these lists is mostly reproduced from Carrier *et al.* (1988).

U_b If b is childless, this list consists of cell b and all childless cells adjacent to b. If b is a parent then the list is empty.

V_b This list consists of all the children of the colleagues of b's parent that are well separated from b.

W_b If b is a parent, the list is empty. If b is childless, it consists of all descendants of b's colleagues that are not adjacent to b.

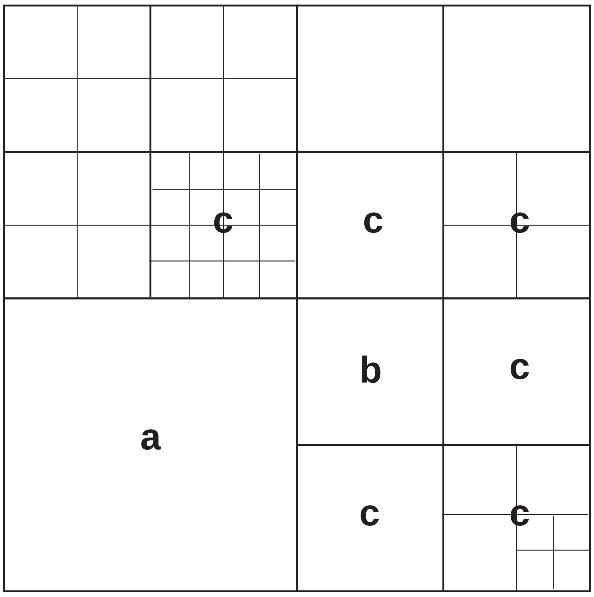

Figure A.2: Colleagues and associates of cell *b*. Cells marked *c* are colleagues (and therefore associates) and those marked *a* are only associate cells (and not colleagues).

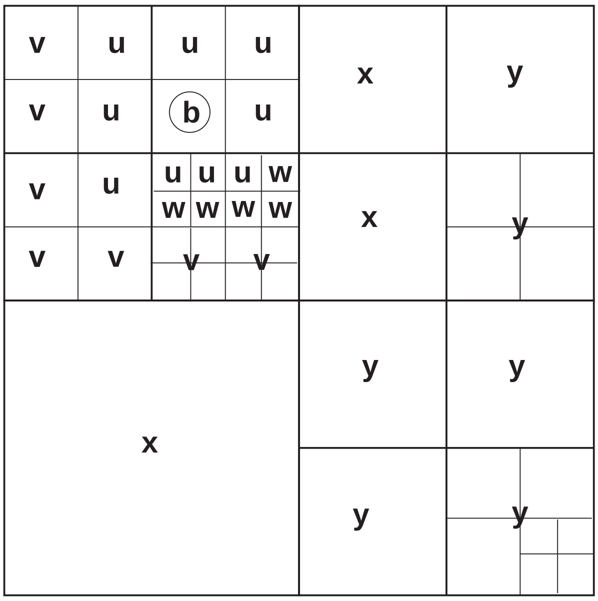

Figure A.3: The five different lists for a cell b. The cells marked 'u' belong to the U_b list, 'v' to the V_b list and similarly for the other lists.

X_b List of all cells c such that b is an element of W_c, i.e. if b is contained in c's W_c list defined above. This implies that c must be childless and must also be larger than b.

Y_b Consists of all cells that are well separated from b's parent.

The above lists are not trivial to determine. The W_b and X_b lists are especially difficult. The various lists for a particular cell b are illustrated in Figure A.3. Carrier *et al.* (1988) compute X_b for each childless cell. Y_b is never computed explicitly since it is already computed at coarser levels via the V_b lists. In the present work the direct computation of the X_b list is eliminated and a simpler Z_b list is defined. This list is similar to the V_b list.

Definition A.8 *The Z_b list for a cell b is defined as the set of all childless associates of b's parent that are not adjacent to b.*

It can be seen that Z_b is in some sense an inverse of the definition of the W_b list. Hence, the relevant lists to be noted are, U_b, V_b, W_b and Z_b. In Figure A.3 the Z_b list is the same as the X_b list with x replaced with by z. Figure A.4 illustrates these lists for a slightly different set of cells. This list of cells is obtained by splitting the cell b from Figure A.3 into four children. It can be seen from these figures that using the Z_b list lets one define X_b in a hierarchical manner in the same way that the V_b list lets one define the Y_b list easily. That is, the union of the Z_b list from Figure A.3 and A.4 gives one the X_b list for the cell marked b in Figure A.4. This makes it easy to implement the computation of the X_b list.

A.3.5 The adaptive fast multipole algorithm

In this section the information from the original paper (Carrier *et al.* (1988)) is supplemented so that it is easier to implement the AFMM. Provided are algorithms with pseudo-code that demonstrates how to compute the various lists and the actual algorithm.

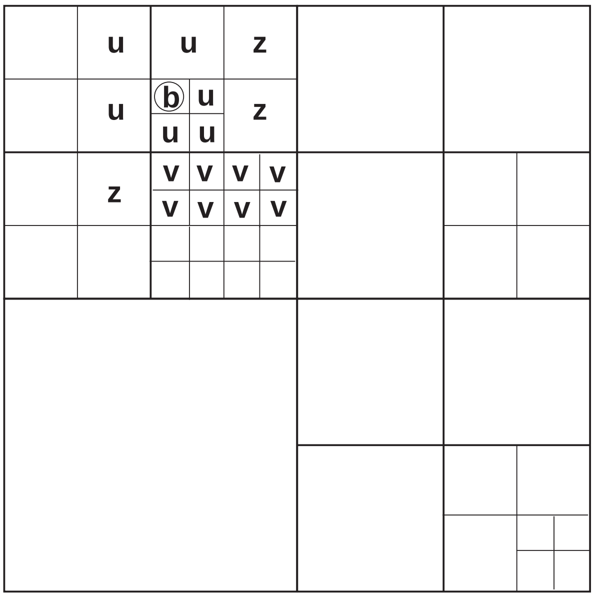

Figure A.4: The U_b, V_b, and Z_b lists for cell b. The cells marked 'u' belong to the U_b list, 'v' to the V_b list and 'z' to the Z_b list.

It is important to note the following in an implementation of the AFMM. All the key information (and the lists for a given cell) can be obtained for a given cell if the following information is available,

- level, center and length of the cell;
- its parent cell;
- its children;
- and its associates.

Associate cells naturally extend the concept of a colleague as defined in section A.2.1. This is especially useful when there is no colleague at a particular side of the cell (say for example the left colleague of a cell does not exist). Clearly, even if the colleague does not exist, the associate cell might exist. This associate cell, that is not a colleague, is obviously larger than the cell, as illustrated in Figure A.2. If the associate cell does not exist, there are no particles in that region. Note that, given the associates of a cell it is trivial to find the colleagues.

In addition to the above, each cell needs to store the multipole coefficients (a_j, equations (A.1)) for the p term multipole expansion and the p local expansion coefficients (b_j, equation (A.7).

Once the tree is generated, the general idea of the algorithm is as follows.

- Compute the multipoles for each cell as done for the body-cell treecodes. This is also called an upward pass since the calculation proceeds from the smallest cells (large levels) to the largest (smaller levels).
- Perform the first downward pass. Starting from level 2 and proceeding to all larger levels, do the following for each cell b,
  - transfer the multipole expansion of the cells in the V_b list to local expansions in b;
  - compute the "special local expansions" on b due to each of the cells in the Z_b list (Definitions A.8 and A.7). Note that these special local expansion coefficients are added to the local expansion coefficients of b.
- Perform the second downward pass. Starting from level 2 and proceeding through all levels, shift the local expansions of each cell to its children.
- Compute the velocity as follows. Consider each childless cell b. For each particle p in b, do the following,
  - add the direct velocity due to each cell in the U_b list to p;
  - add the velocity computed using the multipole expansion of all cells in the W_b list to p;
  - add the velocity computed using the local expansion coefficients of b on p.

At the end of this procedure the velocity on each particle has been obtained using the AFMM. In the pseudo-code provided below it is assumed that each cell has information about its parent, children and its associates. The length of a cell is the length of its side.

Algorithm A.4 FindUbWb(b, cell, Ub, Wb)

```
## cell is possibly in the Ub or Wb lists
## Ub and Wb are lists of the Ub and Wb cells.
d = 0.5*b.length() + cell.length()
dist = b.center() − cell.center()
if (abs(dist.real) > d) or (abs(dist.imag) > d):
    Wb.append(cell)
else:
    if cell.isParent():
        for child in cell.children():
            FindUbWb(b, child, Ub, Wb)
    else:
        Ub.append(cell)
```

Algorithm A.5 UbWbInteraction(b)

```
## b is the cell for which the lists are required
Ub, Wb = [], [] # initialization
if not b.isParent():
    Ub.append(b)
    for associate in b.associates():
        if associate.isParent():
            for child in associate.children():
                FindUbWb(b, child, Ub, Wb)
        else:
            Ub.append(associate)
for u in Ub:
    b.computeUbVelocity(u)
for w in Wb:
    b.computeWbVelocity(w)
```

Algorithm A.6 VbInteraction(b)

```
dist = 1.5*b.length()
for colleague in b.parent.colleagues():
    if colleague.isParent():
        for child in colleague.children():
            if abs(child.center() - b.center()) > dist:
                # set b's local expansion from child's multipoles
                b.setLocalExpansion(child)
```

Algorithm A.7 ZbInteraction(b)

```
for associate in b.parent.associates():
    if not associate.isParent():
        d = 0.5*associate.length() + b.length()
        dist = b.center() - associate.center()
        if (abs(dist.real) > d) or (abs(dist.imag) > d):
            # Set Xb local expansion of associate on b.
            b.setXbLocalExpansion(associate)
```

Algorithm A.8 ComputeLocalExpansions()

```
for level in range(2, n_levels+1):
    for cell in cells[level]:
        VbInteraction(cell)
        ZbInteraction(cell)

for level in range(2, n_levels):
    for cell in cells[level]:
        for child in cell.children():
            cell.shiftLocalExpansion(child)
```

Algorithm A.9 AFMMVelocity()

```
ComputeMultipoles()
ComputeLocalExpansions()
for cell in childless_cells :
    UbWbInteraction(cell)
    # evaluate the local expansion inside cell and add to velocity
    cell .evaluateLocalExpansion()
```

Algorithms A.4 and A.5 show how one can compute the U_b and W_b lists and find their interactions on a cell b. Algorithms A.6 and A.7 show how the V_b and Z_b interactions can be computed. Algorithm A.8 shows how the local expansions are computed and algorithm A.9 shows how the velocity is obtained. The function `ComputeMultipoles` used in algorithm A.9 is defined in algorithm A.1. It is best to read algorithm A.9 first and proceed from there in order to understand the algorithm. The specific details of how the various expansions are computed, shifted and evaluated are not provided since they are relatively straightforward to implement. The pseudo-code mainly shows how the lists are computed and illustrates the general structure of the algorithm.

Thus, it is possible to rapidly compute the velocity field using the AFMM.

A.3.6 Optimizations

It is possible to optimize the implementation of the AFMM by choosing the number of terms, p, used in the multipole and local expansions adaptively. The optimal p depends on the amount of separation between the interacting cells. In the computation of the V_b and W_b interactions, if the separation between cells is large, then fewer terms are necessary for the same accuracy. This can be used to accelerate the AFMM.

The choice of the maximum number of particles per childless cell, n_p, depends on both the distribution of the particles and the nature of the implementation. It is therefore hard to fix a priori. Usually, the optimal value is computed by performing a series of numerical experiments. In a vortex method the fast multipole computation is performed at least once (and often several times) during a single iteration. Consequently, it is possible to adaptively find the optimal number of particles by the following simple procedure. A reasonable first guess is made for n_p. During each fast multipole computation the average CPU time used per particle is computed. An incremental value of ± 1 is fixed and n_p incremented or decremented using it. During the subsequent computation, if the CPU time does not reduce, the sign of the increment is changed. These techniques are mentioned in Russo and Strain (1994). They are quite easy to incorporate into an AFMM implementation and have been reported to produce speed improvements by factors of two or three. The present work does not make use of these improvements. However, it is noted that it is easy to incorporate these in the future.

A.3.7 Numerical results

Figure A.5 compares the CPU time taken versus the number of vortices for the direct case and the AFMM for a cluster of point vortices. The number of terms, p, in the multipole expansions are chosen such that an accuracy of 10^{-6} is guaranteed. In practice at least an order of magnitude more accuracy is obtained than the value specified. It is to be noted that the direct velocity is computed while keeping in mind the fact that for point vortices, there is an anti-symmetry in evaluation of the velocity of one vortex on another. This halves the number of necessary velocity evaluations. The figure shows that the direct method behaves as if it were close to quadratic while the AFMM curve is clearly linear. It is also to be mentioned that the AFMM becomes faster than the direct method when there are more than around 100 to 200 particles (depending on the particle distribution, type of blob etc.). In Figure A.5, two AFMM curves are plotted, one for a particle distribution with point vortices placed randomly using uniform deviates inside a square region and the other with particles placed on an ellipse having an axis ratio of three is to one. It is clear that the latter will perform better since most of the cells are well separated and an almost two-fold speed increase is seen. There is no change in the behavior of the direct algorithm since it is completely insensitive to the distribution of the vortices. From the results, it is evident that the AFMM is a much faster algorithm. The ability to accurately and rapidly evaluate the velocity

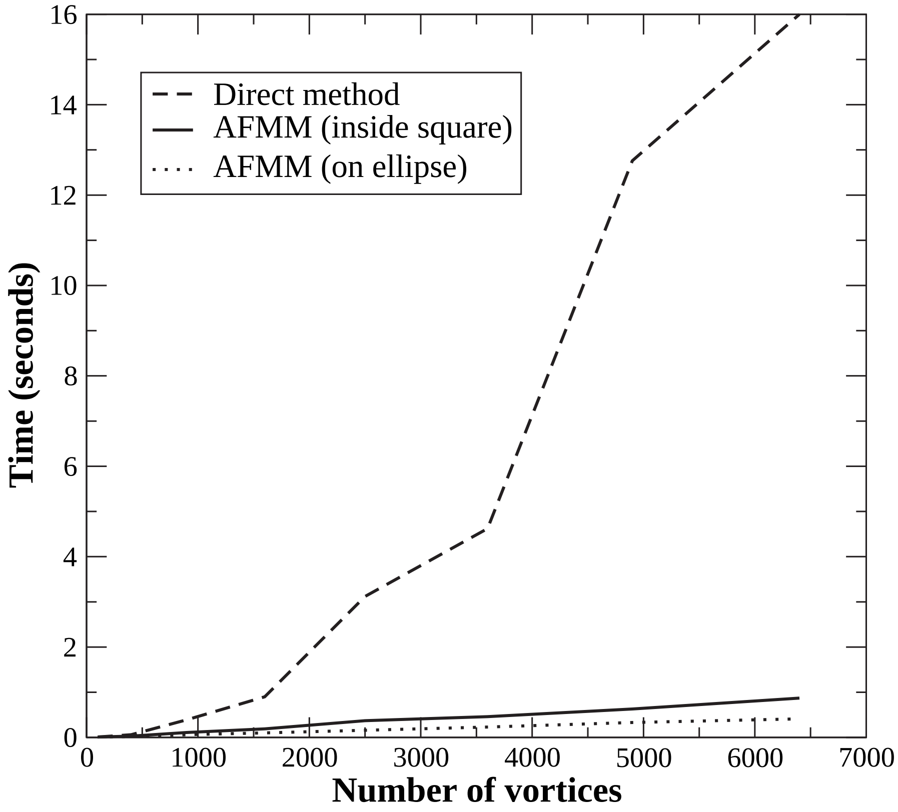

Figure A.5: Comparison of CPU time taken as the number of vortices is varied for the direct method and the AFMM. The dotted line plotting the AFMM results is for a case where the vortices are distributed on an ellipse. The solid line plots the variation when the particles are distributed randomly inside a unit square using uniform deviates.

field using the AFMM makes vortex methods much more effective and usable.

A.4 Summary

The theory and implementation of the adaptive fast multipole method (AFMM) have been discussed in considerable detail. The $O(N \log N)$ body-cell treecodes have also been described in order to facilitate easier understanding of the AFMM. A different approach to the AFMM algorithm was proposed that does not require the explicit computation of the X_b list as done in the original work of Carrier *et al.* (1988). This simplifies the implementation of the method.

APPENDIX B

COMPUTATION OF DIAGNOSTIC QUANTITIES

Diagnostic quantities serve three important purposes. They enable comparison of computed results with experimental and other computational data. They can be used to check the accuracy of the simulation. They are also directly useful in themselves. For example, the distribution of forces on a body is a diagnostic and is also useful to the engineer. This chapter discusses how several of the diagnostic quantities are computed in the context of a vortex method. Some commonly used quantities computed in numerical simulations are the following.

1. Forces and moments.
2. Vorticity field.
3. Streamlines, streaklines and path lines.
4. Velocity field.
5. Separation points.
6. Surface vorticity distribution.
7. Pressure field or surface pressure distribution.

Some of these are not easy to compute when a vortex method is used. The important quantities used in the present work are discussed in the subsequent sections.

B.1 Computation of diagnostics

B.1.1 Forces and moments

Computing the force on an arbitrary geometry using a vortex method can be somewhat involved. Two methods are discussed and used in the present work.

The first approach is to determine the forces on the body using the idea of vortex momentum. The difficulty with this approach is that only the total force can be computed. The distribution of the forces is not available. The pressure force and skin friction forces cannot be separated. Koumoutsakos and Leonard (1995) and Lin *et al.* (1997) discuss another scheme where the pressure and skin friction can be obtained for different shapes. The distribution of the forces can also be obtained. This method is more complicated than the the vortex momentum approach. Both of these approaches are discussed in the following two sections.

B.1.2 Force computation: vortex momentum

The vortex momentum or hydrodynamic impulse is defined as follows:

$$\begin{aligned}\vec{I} = (I_x, I_y) &= \rho \int_{\mathcal{R}^2} \vec{r} \times \vec{\omega} \; dx \; dy \\ &= \rho \sum_{i=1}^{N} \vec{r_i} \times \Gamma_i \vec{k} \ , \end{aligned} \tag{B.1}$$

where ρ is the density of the fluid and $\vec{r_i}$ is the position of a vortex element having circulation Γ_i. For a two dimensional flow the vorticity is along the z axis and hence along $\vec{k}$. The force can be obtained from the vortex momentum as follows:

$$\vec{F} = -\frac{d\vec{I}}{dt}. \tag{B.2}$$

Due to the stochastic nature of the RVM, the computed vortex momentum tends to have small high frequency oscillations. The noise has a broad band spectrum (white noise) and hence cannot be easily filtered out. When the $\vec{I}$ versus t curve is differentiated to obtain the load, spurious and very large oscillations are seen in the load. Fig. B.1 plots I_x and I_y for the flow past an impulsively started circular cylinder at $Re = 3000$. The solid line plots I_x and the dashed line plots I_y. The values of $\vec{I}$ are available in intervals of 0.01 seconds. At the resolution shown in the figure no oscillations are apparent. Fig. B.2 shows a zoomed view of

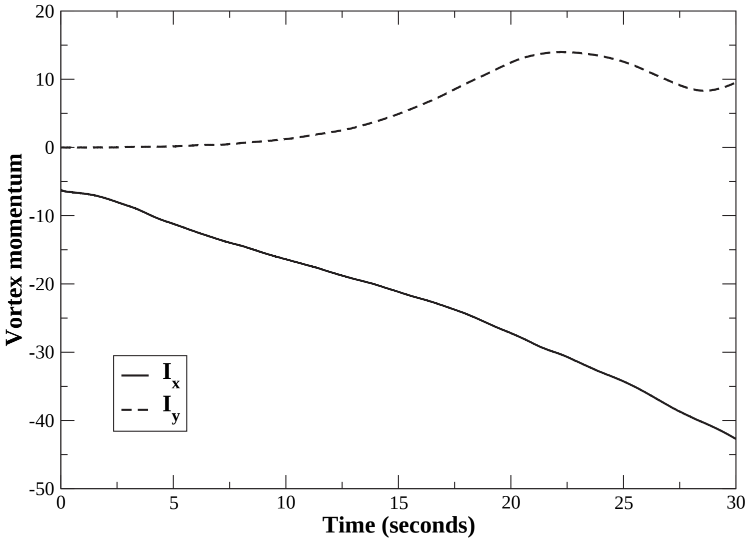

Figure B.1: Vortex momentum versus time for flow past an impulsively started cylinder at $Re = 3000$. The solid line plots I_x and the dashed line plots I_y.

the I_x versus time curve. As can be seen, there are significant oscillations at this resolution. Differentiating this curve using a centered difference with $\Delta t = 0.01$ will naturally produce extremely noisy forces as evidenced in Fig. B.3.

Many approaches can be used to obtain smooth and fairly accurate loads from this noisy data. The simplest approach is to use a sliding window average of the data. The value at each point is replaced by the average of the values in the vicinity of the point. Fig. B.4 plots the load, F_x versus time for a few different window sizes. The load is obtained by performing a central difference of the vortex momentum. At the end points a forward and backward difference is used. The black curve uses a 11 point average and the green a 21 point average. The red curve is the case of a 11 point average applied five times to the averaged data. As can be seen, the averaged data captures the trends in the load and eliminates the noise seen in Fig. B.3.

Another approach to obtain loads from the noisy data is to smooth the vortex momentum data using a piecewise least squares fit of the data. The data is split into several intervals. In each interval a polynomial of order n is fit to the data with a reasonable amount of overlap between intervals. This piecewise polynomial is then differentiated to obtain the load. Fig. B.5 plots the variation of F_x versus

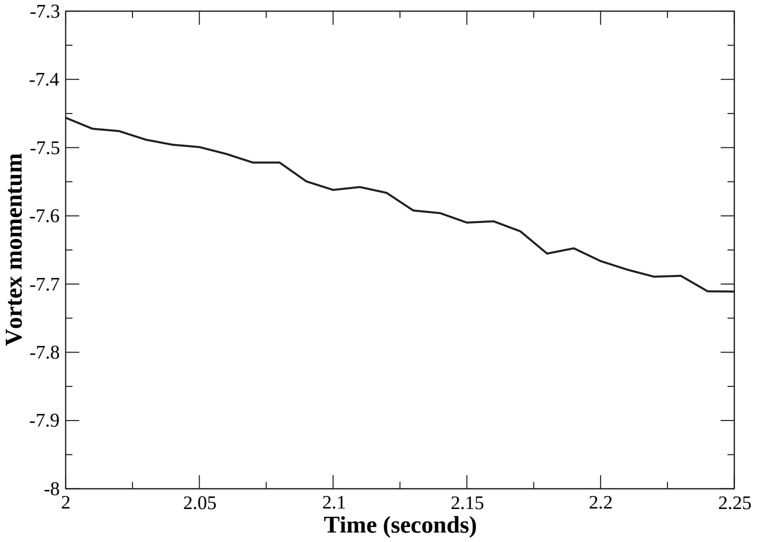

Figure B.2: Vortex momentum versus time for flow past an impulsively started cylinder at $Re = 3000$. This is a zoomed view of the I_x curve shown in figure B.1.

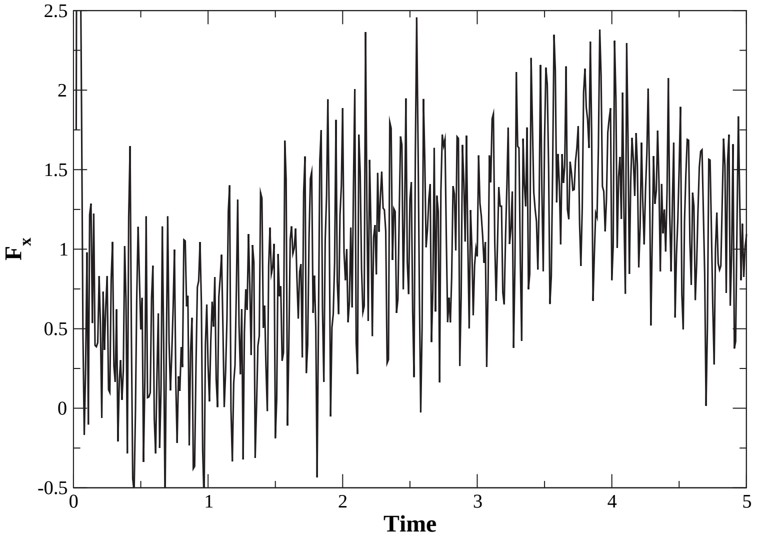

Figure B.3: Force along x axis for flow past an impulsively started cylinder at $Re = 3000$. The force was obtained by taking central differences directly on the data.

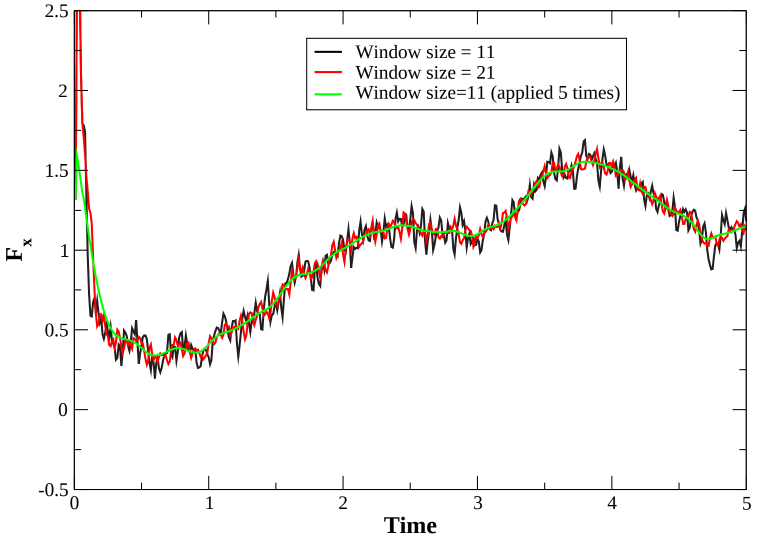

Figure B.4: Force along x axis obtained by using a sliding window average.

time. The force is obtained using a least squares fit with a 7th order polynomial for each 0.5 second interval of data. To smooth the edges between the intervals, each polynomial was made to overlap with 50% of the data from the adjacent intervals of data. In the overlapping region, the average of the overlapping polynomials is taken. The least square solution was obtained using SciPy's (Jones *et al.*, 2001–) optimize routines. As can be seen, the curve is much smoother than that seen in Fig. B.3. Also plotted (dashed line) is the load obtained by a sliding average with a window size of 11 that is applied 5 times.

The approaches described above produce results that match the data reasonably well. However, the choice of the averaging window size, order of polynomial, overlap etc. are done in an ad-hoc manner. In order to choose these parameters systematically, a known curve is considered and noise is added to it. The procedures discussed above are applied to this data and the smoothed data is compared to the exact curve. The curve chosen is given as,

$$y(x) = (\cos(2x) + \cos(4x))(1/x - 1) + x/10 + 2. \tag{B.3}$$

The reason for this choice is that the curve is very similar to the loads that one would expect for the flow past an impulsively started cylinder. The curve is singular near $x = 0$ and also exhibits a periodic behavior with steep gradients.

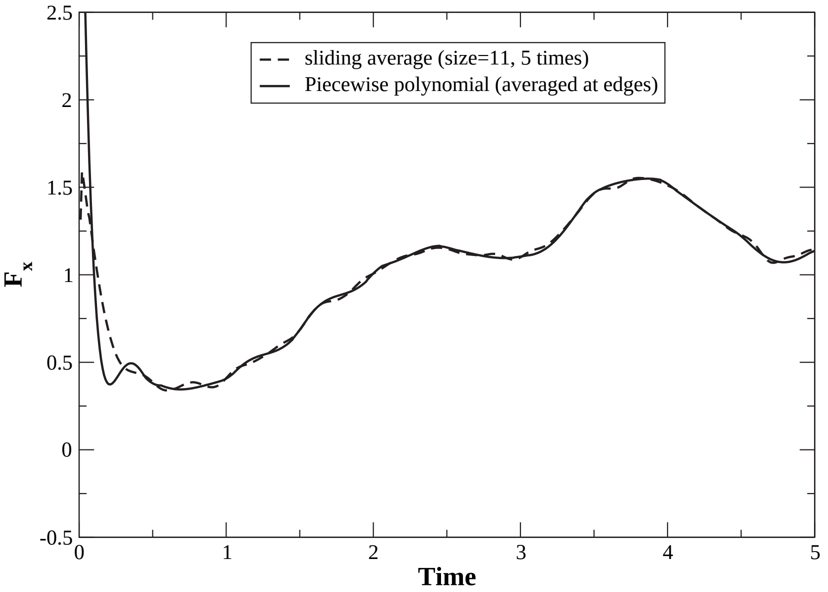

Figure B.5: Force along x axis. The solid curve is obtained using a least squares fit with a 7th order polynomial for each 0.5 second interval of data with 50% overlap. The dashed curve is obtained using a sliding average (window size = 11, applied 5 times).

This curve is sampled on 2990 points along x in the interval $0.1 \leq x \leq 30$. This function is plotted in Fig. B.6. To this sampled curve, random noise, generated using uniform deviates in the range $(-0.5, 0.5)$, is added. Different schemes are used to determine the actual curve from the noisy data and then the results are compared with the known curve. The error E_1 is computed as,

$$E_1 = \frac{\sum_{i=0}^{N} \mid \tilde{y}_i - y_i \mid}{N}, \tag{B.4}$$

where $\tilde{y}_i$ is the computed value (after smoothing), y_i is the exact value and N is the number of sample points. The error E_2 is computed as,

$$E_2 = \sqrt{\frac{\sum_{i=0}^{N} (\tilde{y}_i - y_i)^2}{N}}. \tag{B.5}$$

Using these it is possible to measure how well the techniques approximate the actual curve.

Consider the case of using the sliding window technique on the noisy data. Different window sizes are considered and for each window size the sliding average is applied various number of times. Figs. B.7 and B.8 plot the errors E_1 and E_2

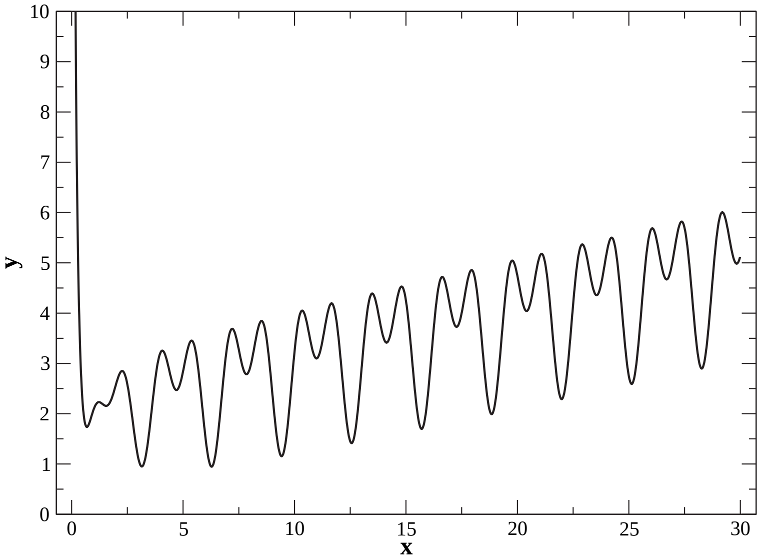

Figure B.6: A test function used to obtain the best scheme for removing noise from data.

respectively as the window size is changed. It is clearly seen in both the figures, that applying the averaging several times increases the error. It is also seen that the optimal window size depends on how the error is measured.

It is to be mentioned that if the amplitude of the noise increases by a factor of two the optimal window size changes again. The optimum size for the E_1 error changes from 23 to 29 and the E_2 error changes from 15 to 23. However, applying the average once continues to produce lower errors than when a smaller window is applied many times.

Considering the piecewise polynomial interpolation, it is found that there are a few parameters that can be varied. The order of the polynomial to be chosen, the size of each piece that is approximated using the polynomial and the amount of overlap between pieces. The amount of overlap is fixed at 50%. This seems a fairly reasonable choice. The overlapping data between two of the piecewise polynomials is not averaged. The order of the polynomial is varied from 1 to 7. The number of points per piece is varied from 10 to 50 in steps of 5. Figs. B.9 and B.10 plot the errors E_1 and E_2 as the order of the polynomial is varied along the x-axis. The various curves plotted are for different number of points per chosen interval of the approximated data. It is clear that both E_1 and E_2 are small for the case where the order of the polynomial is around 5 with around 35-45 points

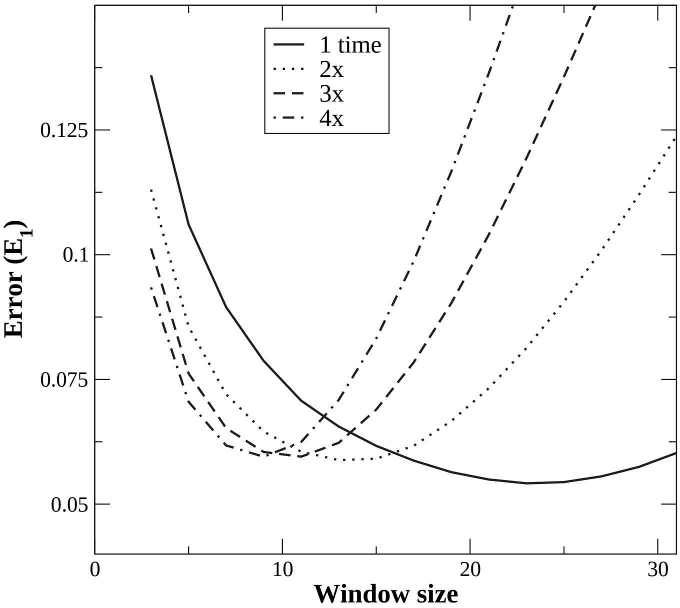

Figure B.7: Error (E_1) for the sliding window average with different window sizes applied various number of times.

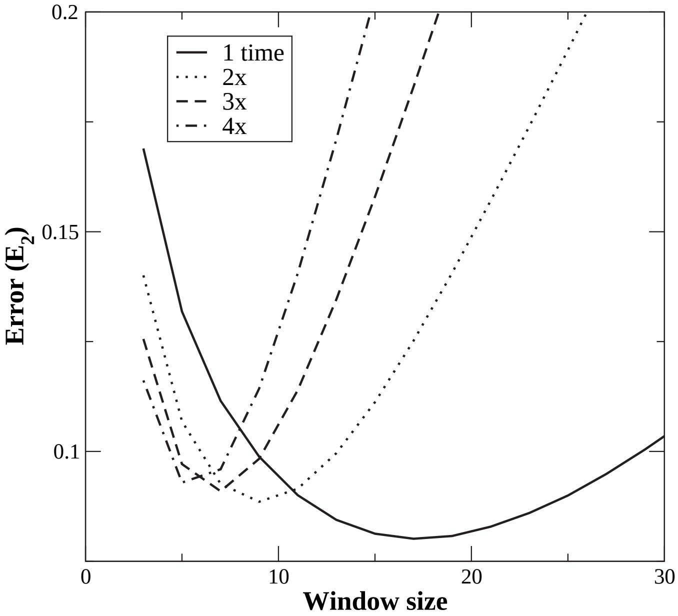

Figure B.8: Error (E_2) for the sliding window average with different window sizes applied various number of times.

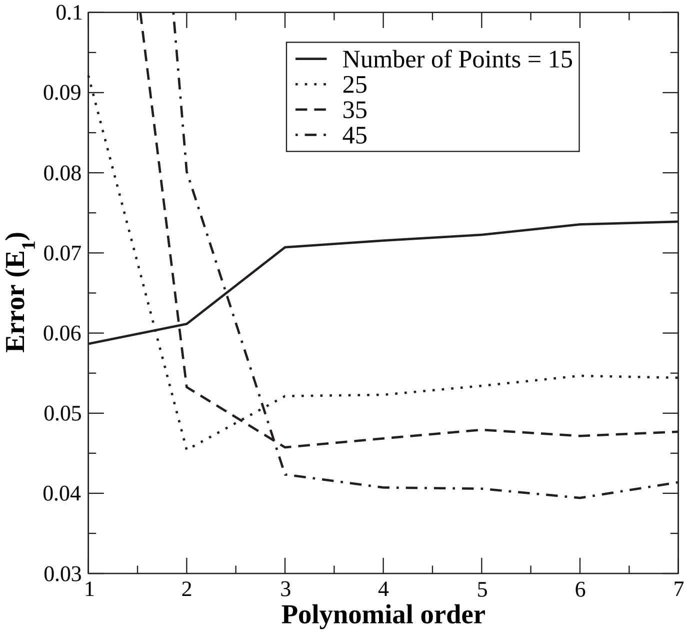

Figure B.9: Error (E_1) for piecewise polynomial least squares fit of the data as different order polynomials are used and as the number of points in the interval are varied.

per piecewise polynomial. It is observed that the errors are largely insensitive once a 4th or 5th order polynomial is used. It is also seen that errors are smaller as compared to the sliding window average case. Further, it is noted that these results do not change much when the amplitude of the noise is doubled, i.e. the best solution is still obtained with a piecewise polynomial of order around 5 with 35-45 points per interval.

One can expect that the number of points to be chosen for each piecewise polynomial depends on the nature of the curve being approximated and the number of points used to sample the entire curve. For example if the curve is approximated with 1000 points one would expect that the number of points per interval used in the interpolation would be different from the case when the number of sampled points is say 2000. Therefore, the effect of varying the number of points per piecewise polynomial for different number of sample points is studied. Fig. B.11 plots the variation of error E_2 versus number of points per piecewise polynomial for different number of sample points. It is clearly seen that as the number of sample points doubles, the number of points per piece necessary for the minimum error approximately doubles. This indicates that for generic data there is an element of uncertainty in the choice of the right number of points. No experimentation

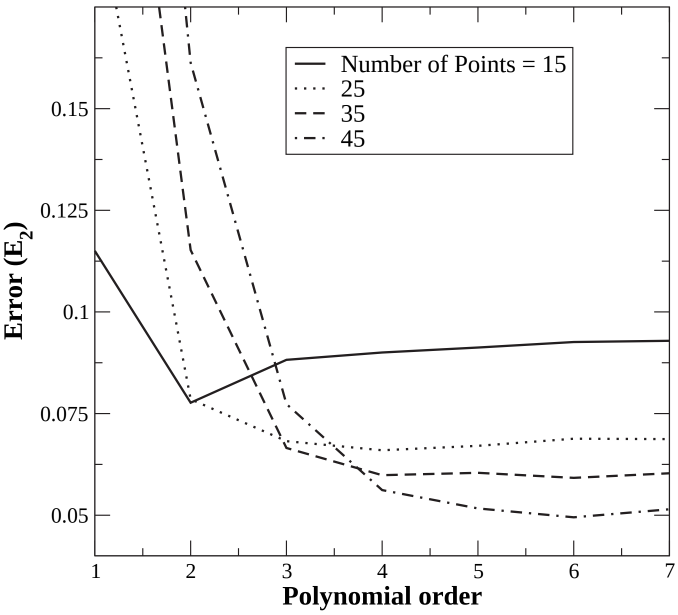

Figure B.10: Error (E_2) for piecewise polynomial least squares fit of the data as different order polynomials are used and as the number of points in the interval are varied.

can be done in these cases because the exact curve is not known. However, the user could visually investigate if the curve appears smoother as different number of points are used. It is also possible to use the sliding average to obtain a rough guideline for the optimal number of points to be used per piecewise polynomial.

From these results it is clear that using a piecewise polynomial least square fit is the best and most consistent option to extract the actual data from noisy data. However, the method is a little computationally expensive.

It is possible to use the Discrete Fourier Transform (Bracewell, 1986) of the given data and filter out the noise. The filtering can be done in either Fourier space by using a low pass filter or in real space by convolving the data with the Fourier transform of the appropriate low-pass filter. Computing running averages corresponds to convolving the time data with a gate filter, which corresponds to multiplying the Fourier transform of the data with an appropriately scaled *sinc* function in the Fourier domain. It is possible to use better low pass filters with suitable windowing functions. However, after considerable amount of experimentation it was found that the piecewise polynomial least square fit approach described above produced the best results with the least error. Computationally, it is much

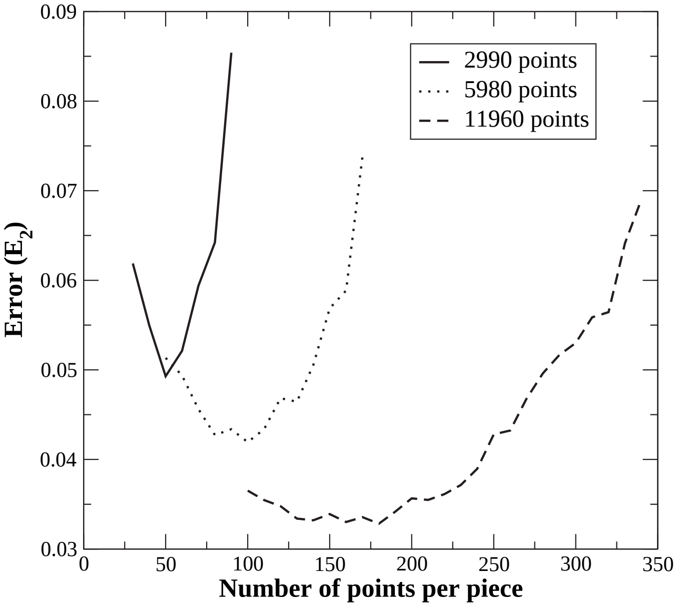

Figure B.11: Error (E_2) versus different number of points per piece as the number of sample points on the curve is varied. Piecewise polynomials are fit to the data using a least squares fit.

faster to use convolution than to use a piecewise polynomial fit. Therefore, it is sometimes useful to use a Gaussian filter to rapidly obtain smooth plots. These plots can be used as a first approximation and also to fine tune the polynomial fit parameters.

It is to be noted that if one is given a noisy curve then one can estimate the noise levels by smoothing the data using an appropriate method and then finding the average difference between the smoothed and noisy curves using equation (B.5). In the present work this idea is used to measure the noise levels in the curve.

B.1.3 Force computation: pressure and skin friction

Computing the force using the vortex momentum (or hydrodynamic impulse) provides only the total force and not the distribution of the force. In order to find the force distribution, the pressure and skin friction forces on the body are to be computed.

On the surface of the body, the boundary layer equations are assumed valid and it can be shown that the shear stress is,

$$\tau_{wall} = \nu \omega_{wall}, \tag{B.6}$$

along the tangential direction on the surface. ω_{wall} is the vorticity at the surface of the body.

The computation of the pressure is more involved. The approach of Lin *et al.* (1997) is used to compute the pressure. Consider the momentum equation as given in equation (2.2),

$$\frac{D\vec{V}}{Dt} = -\frac{1}{\rho}\text{grad}p + \nu\nabla^2\vec{V},$$

where p is the pressure and $\vec{V}$ is the velocity. Let $\hat{e}_n$, $\hat{e}_s$ be the local normal and tangential directions along the body surface respectively such that $\hat{e}_n = \hat{e}_s \times \hat{k}$, where $\hat{k}$ is the unit vector out of the plane. Now taking the dot product of the momentum equation with $\hat{e}_s$ gives,

$$\frac{Du_s}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial s} + \nu\nabla^2 u_s,$$

where u_s is the velocity component along the tangential direction $\hat{e}_s$. When this equation is specialized to the surface it can be seen that,

$$\frac{Du_s}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial s} - \nu\frac{\partial \omega}{\partial n}.$$

For the case of a non-accelerating, rigid body the left hand side is zero and the pressure can be obtained as,

$$\frac{1}{\rho}\frac{\partial p}{\partial s} = -\nu\frac{\partial \omega}{\partial n}. \tag{B.7}$$

If the body is accelerating, this equation changes due to the inclusion of the acceleration terms. The term on the right hand side corresponds to the rate of creation of vorticity at the boundary. Hence, if a vortex sheet of strength γ is released at a given time on the surface and Δt is the time step used in the

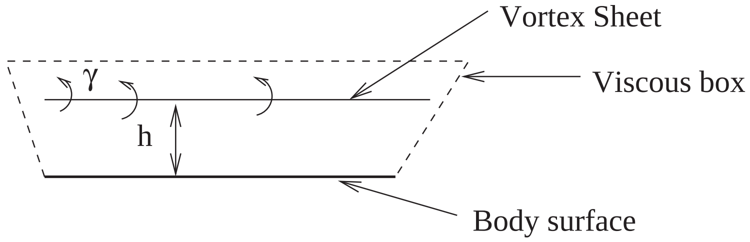

Figure B.12: Illustration of sheet inside a viscous box.

computation,

$$\nu \frac{\partial \omega}{\partial n} = \frac{\gamma}{\Delta t}. \tag{B.8}$$

Hence, using equations (B.8) and (B.7) the pressure distribution can be obtained. Assuming the pressure at a particular point on the surface, equation (B.7) can be easily integrated to obtain the pressure distribution on the body. Using the pressure distribution the pressure force can be computed using,

$$\vec{F_p} = \oint -p\hat{e}_n ds. \tag{B.9}$$

Similarly, the shear stress can be integrated to produce the net frictional force on the body.

Computationally, finding the wall vorticity (ω_{wall}) is not trivial. Koumoutsakos and Leonard (1995) generate a body fitted grid and solve for the vorticity on the wall by computing the stream function on the grid. In the present work the need for the grid is avoided. Instead a much simpler scheme is used. Since the present work employs vortex sheets in a thin numerical layer around the body, the sheets are used to determine the vorticity at the wall. The vorticity of the sheets is represented by Gaussian smoothing functions. As suggested by Fogelson and Dillon (1993), the smoothing parameter of the Gaussian, ϵ, is chosen as,

$$\epsilon = CN^{-q}, \tag{B.10}$$

where C is a constant, $q = 0.2$ and N is the number of sheets used to represent the vorticity. Consider a viscous box[1] as shown in Fig. B.12. The height of the viscous box is equal to the numerical layer height and the height of a sheet is h.

---

[1] The viscous box is defined in section 5.1.1.

The vorticity due to a collection of sheets having strength, γ_j, and height, h_j, is computed as,

$$\omega_{wall} = \sum_{j=0}^{N} 2 \frac{\gamma_j}{\epsilon\sqrt{\pi}} \, e^{-h_j^2/\epsilon^2}. \tag{B.11}$$

The factor of 2 in the above equation arises since one must consider the image sheets (of the same sign) in order to obtain the correct vorticity at the wall. Image vortices must be considered because the smoothing of the vorticity using Gaussian smoothing functions is incorrect in the vicinity of the solid wall. The part of the Gaussian piercing the solid wall must be "reflected" about the wall. This reflected vorticity can be thought of as arising from an equivalent image sheet of the same sign at a height of $-h_j$. Therefore this image sheet can be accounted for by multiplying the value by a factor of two.

The optimal value of C can be obtained by considering Stokes' first problem. For this problem, the exact wall vorticity due to a unit jump in the velocity can be shown to be,

$$\omega_{wall}(t) = \frac{1}{\sqrt{\pi \nu t}}.$$

After performing a few numerical experiments it was found that $C = 3\sqrt{2\nu\Delta t}$ produced good results. Fig. B.13 plots the variation of ω_{wall} versus t, when $N = 100$ for different viscosities. The numerical layer height chosen is $h_{num} = 3\sqrt{2\nu\Delta t}$. The computed results closely follow the exact curve. The $\nu = 0.001$ curve is noisy. However, as the number of particles is increased, the noise reduces. This is seen in Fig. B.14. Hence, using this approach it is possible to compute the wall shear stress with a reasonable amount of accuracy. In the computations for the flow past a circular cylinder the choice of h_{num} arises out of different considerations and using it as such works well in practice. The results in chapter 8 clearly demonstrate that this approach works well.

Finding the vorticity flux is fairly easy using equation (B.8) and the newly created sheets at each time step. The only difficulty with this approach and the hybrid vortex sheet/blob method is that the vorticity flux variation in time can be quite erratic. For the case of the flow past an impulsively started cylinder at $t = \Delta t$ a large number of sheets will be released to offset the initial slip. If Δt

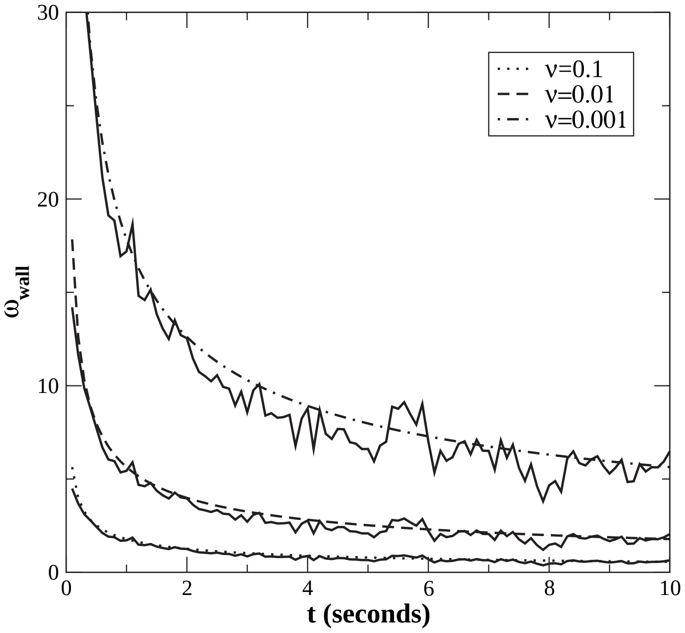

Figure B.13: ω_{wall} versus time for Stokes' first problem. The exact and computed results are plotted for different kinematic viscosities, ν, when $N = 500$.

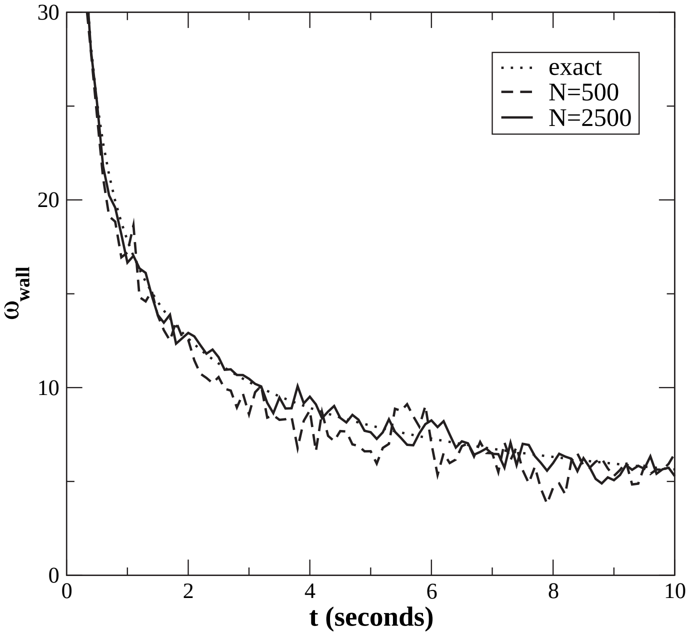

Figure B.14: ω_{wall} versus time for Stokes' first problem. The exact and computed results are plotted for different numbers of sheets used, N, when $\nu = 0.001$.

is sufficiently small such that at $t = 2\Delta t$ the sheets have not moved enough to create new sheets, then no new sheets will be created. This situation is highly likely in practice since Δt must be small enough to avoid a large motion for the particles. Therefore, naively computing the flux using this approach will result in a zero pressure distribution. This is incorrect and to correct this behavior it is better to compute the flux as the average of the present value and the previous value. This produces better results.

Unlike the frictional force, the pressure force curve is usually noisy like the force curve obtained from the vortex momentum. Therefore, the pressure forces need to be smoothed. The same techniques developed in section B.1.2 are used for this purpose.

The vorticity on the body and flux of the vorticity can also be plotted along the length of the body. Chapter 8 presents many results using the approach developed here. As seen there, the quality of the curves is very acceptable.

B.1.4 Vorticity distribution

This is the direct output of any vortex based code. The vortex method produces an unstructured collection of points representing the vorticity. Such data is not readily amenable for analysis or visualization since there is no structure in the organization of the particles. Due to the random walk of the particles, it is highly likely that the particles overlap one another. This overlapping makes a triangulation of the points difficult. In order for the data to be used for visualization, or further analysis, this vorticity needs to be interpolated to a grid that has a well defined structure.

Interpolating the vorticity from an unstructured collection of points onto a regular $h \times h$ grid is a common requirement and one for which solutions exist. The interpolation scheme is chosen such that p moments of the vorticity are conserved. Koumoutsakos and Leonard (1995) use a second order scheme ($p = 2$) where the circulation and two higher moments are preserved. In the vicinity of a wall the interpolation needs to be performed with care since no vorticity should be interpolated to a grid point that is separated from the vortex by a solid wall. Ploumhans and Winckelmans (2000) describe a third order interpolation scheme (preserves 3 moments apart from circulation) in the presence of a boundary.

In the present work both a second and third order interpolation scheme have been implemented. The scheme is capable of handling arbitrary geometries. The approach used is essentially the same as that detailed by Ploumhans and Winckelmans (2000). The only significant difference is that for complex geometries an attempt is made to perform the best possible interpolation given the constraints of the grid and geometry. The present implementation is robust and can handle situations where the grid is poor and the geometry is complex.

The basic idea is simple. A one dimensional interpolation kernel is used. The circulation of a vortex blob or sheet is first distributed to temporary particles created on the nearest grid lines parallel to the x-axis (i.e. a $y =$ const line). Then, the circulation of each of the temporary particles is distributed to grid points parallel to the y-axis (i.e. $x =$ const line). This is called an XY distribution. The same process can be applied along the y axis first and then along the x axis resulting in a YX distribution. In regions of space where there are no boundaries the XY and YX distributions are equivalent. In regions where there are boundaries, each of the distributions are assigned penalties depending on the interpolation scheme used. The best of the two schemes is chosen. More details are provided below. The interpolation kernels are first discussed followed by a detailed description of the algorithm.

A centered interpolation kernel of order p along the x co-ordinate is given as $\Lambda_p(x)$. A decentered kernel is denoted by $\Lambda'_p(x)$. Consider a particle located at x such that $-1/2 \leq x \leq 1/2$. The centered $\Lambda_3(x)$ interpolation kernel is given as,

$$\Lambda_3(x) = \begin{cases} (3-2x)(4x^2-1)/48 & \text{to particle at -3/2} \\ (1-2x)(9-4x^2)/16 & \text{to particle at -1/2} \\ (1+2x)(9-4x^2)/16 & \text{to particle at 1/2} \\ (3+2x)(4x^2-1)/48 & \text{to particle at 3/2.} \end{cases} \tag{B.12}$$

A decentered third order kernel $\Lambda_3'(x)$ is given as,

$$\Lambda_3'(x) = \begin{cases} (1-2x)(2x-5)(2x-3)/48 & \text{to particle at -1/2} \\ (2x-5)(2x-3)(1+2x)/16 & \text{to particle at 1/2} \\ (1-2x)(2x-5)(1+2x)/16 & \text{to particle at 3/2} \\ (1-2x)(3-2x)(1+2x)/48 & \text{to particle at 5/2.} \end{cases} \tag{B.13}$$

A centered second order kernel $\Lambda_2(x)$ is given as,

$$\Lambda_2(x) = \begin{cases} x(x-1)/2 & \text{to particle at -1} \\ (1-x^2) & \text{to particle at 0} \\ x(1+x)/2 & \text{to particle at 1.} \end{cases} \tag{B.14}$$

A decentered second order kernel $\Lambda_2'(x)$ is given as,

$$\Lambda_2'(x) = \begin{cases} (x-2)(x-1)/2 & \text{to particle at 0} \\ x(2-x) & \text{to particle at 1} \\ x(x-1)/2 & \text{to particle at 2.} \end{cases} \tag{B.15}$$

A first order interpolation kernel $\Lambda_1(x)$ is given as,

$$\Lambda_1(x) = \begin{cases} (1-x) & \text{to particle at 0} \\ x & \text{to particle at 1} \end{cases} \tag{B.16}$$

The grid used for the interpolations are illustrated in Fig. B.15.

The interpolation algorithm works as follows. The size and grid spacing of the grid is determined by the user.

1. Construct a mesh given the user parameters.
2. Consider the panels that constitute the solid walls and identify the cells of the grid that the panels pass through. Store information on which panel (corresponding to the geometry) each cell contains.
3. For each vortex blob/sheet, perform the following:
    (a) Perform an XY distribution which proceeds as follows:
        i. Find the cell that contains the particle.

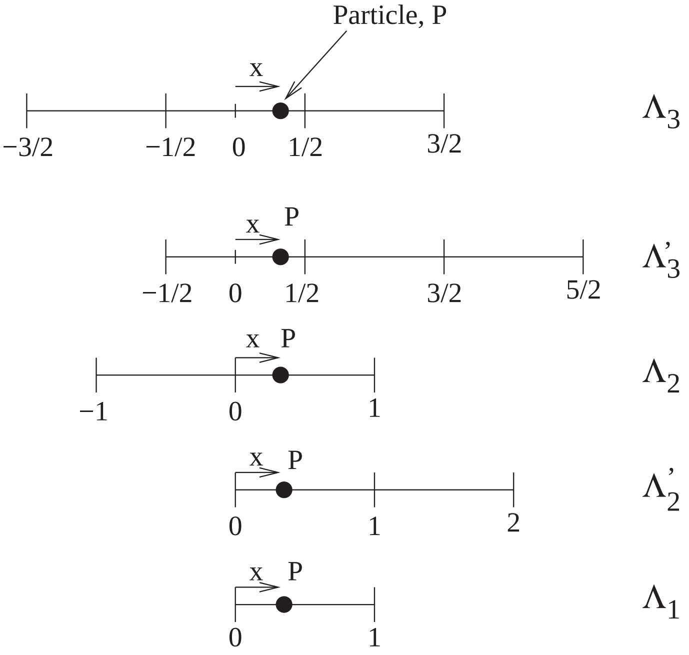

Figure B.15: Grid in relation to the interpolation schemes used.

   ii. Distribute the vorticity of the particle along the X direction to temporary particles placed on the nearest grid lines. Care is to be taken to avoid distributing vorticity to temporary particles that are across a solid wall.
   iii. Distribute each temporary particle to temporary particles at the grid points along the Y direction while considering the boundaries.
   iv. Compute the average penalty (discussed below) P_{XY}, for all the temporary particles created at the grid points.

(b) Perform a YX distribution which proceeds as follows:

   i. Find the cell that contains the particle.
   ii. Distribute the vorticity of the particle along the Y direction to temporary particles placed on the nearest grid lines while taking care of boundaries.
   iii. Distribute each temporary particle to temporary particles at the grid points along the X direction while considering the boundaries.
   iv. Compute the average penalty (discussed below) P_{YX}, for all the temporary particles created at the grid points.

(c) Choose the scheme (P_{XY}, P_{YX}) that has the least penalty. If they have the same penalties use the average of both schemes.

4. For each grid point, add the strength of the temporary particle that is located on it.

This scheme thus interpolates the vorticity onto the mesh using the best possible scheme given a grid, geometry and vorticity distribution. In the best case as

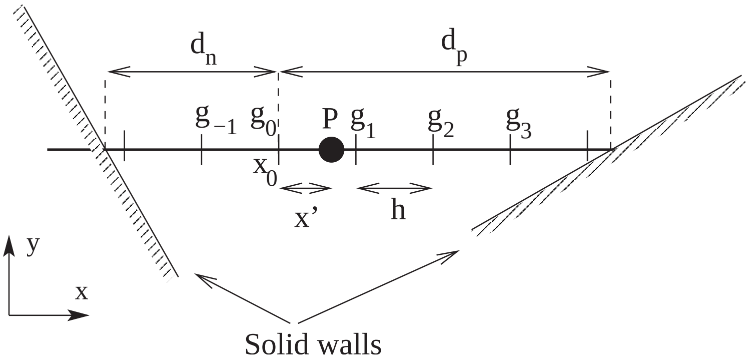

Figure B.16: Illustration of various parameters used for interpolation.

many as three moments are conserved.

When distributing the strength of the particle to the temporary particles at the grid points, a penalty is imposed for each type of interpolation used. If the Λ_3 or Λ_2 interpolation is used, then the penalty is zero. If Λ_3' or Λ_2' are used the penalty is 1. If Λ_1 is used the penalty is 2. If nearest point interpolation is used a penalty of 3 is assigned. The penalties along each direction accumulate, so a particle with a decentered interpolation along both x and y will have a total penalty of 2.

The strength of the particle is distributed to the temporary particles in the following manner:

1. Find the cell that contains the particle.
2. Consider the line along which the particle strength is distributed (horizontal when along X or vertical when along Y). This line is called "the line of distribution". Find the intersection of the line of distribution with the nearest panels.
3. Find the closest intersection of the line with the geometry on the left and right side of the particle.
4. Depending on the distance of the particle to the nearest wall on the left and right choose an appropriate scheme of distribution.

Consider Fig. B.16 which illustrates a simplified example of a particle, P, whose vorticity is to be distributed along the x-axis. The grid size is h. Consider the case where a third order scheme is used for interpolation. Let x_0 be the location of the nearest grid point to the left of the particle. Let the grid point associated

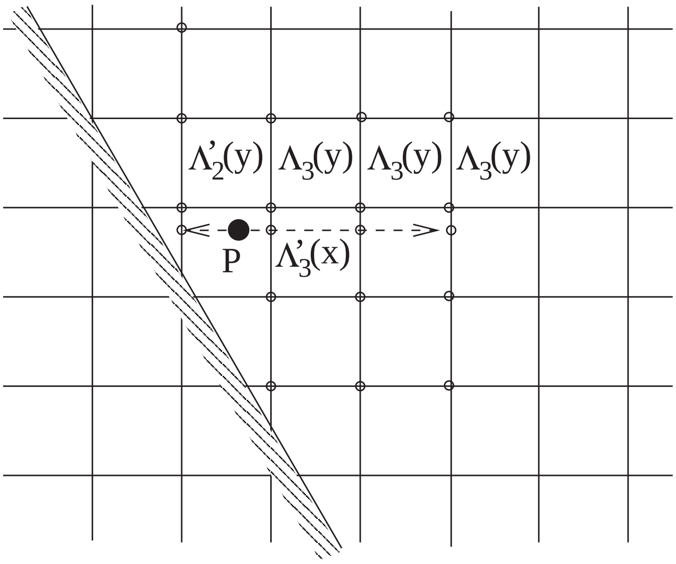

Figure B.17: Interpolation for a point P using the XY distribution. Temporary particles are first created along the $y=$ const. grid lines (along the dashed lines) and these particles are then distributed to the grid points along the $x=$ const. lines.

with x_0 be denoted as g_0. Let d_p be the distance to the nearest intersection to a wall on the right side of the particle from the point x_0. Similarly, let d_n be the distance to the nearest intersection to a wall on the left of the particle from x_0. That is, d_p and d_n are measured with respect to x_0. Let the position of the particle with respect to x_0 be x'. The third order interpolation equation (B.12) distributes vorticity when $-1/2 \leq x \leq 1/2$. Hence $x = x'/h - 0.5$. To clarify the discussion here is a concrete example. Let the particle be located at x_p and let x_0 and g_0 be known. Let x_{li} and x_{ri} be the closest left and right intersection points. Then:

$$\begin{aligned} x' &= x_p - x_0 \\ d_p &= x_{ri} - x_0 \\ d_n &= x_0 - x_{li} \\ x &= x'/h - 0.5 \end{aligned}$$

The pseudo-code embodying the distribution of the vorticity to the grid points is given in algorithm B.1. In the pseudo-code, the term "positive direction" indicates distribution to points to the right of the grid point (or above if the distribution is along the y axis) and "negative direction" indicates distribution to the points on the left (or below). The algorithm performs the best possible interpolation given the constraints of the grid and geometry. When no interpolation is possible (the function returns false) the total penalty is incremented by 10. The average penalty of all the particles is computed and the best scheme is used as described earlier.

The algorithm is also illustrated for one particular case in Fig. B.17. In the figure, the XY distribution for a particle P is illustrated. Temporary particles are first created along the $y =$ const. lines using a Λ_3' distribution. These temporary particles are then distributed to temporary particles along $x =$ const. (which are grid points) using appropriate distribution schemes.

Algorithm B.1 DistributeVorticity()

```
if $d_n > 2h$ then
  Case1()
else if $d_n > h$ then
  Case2()
else if $d_n > 0$ then
  Case3()
else
  Case4()
end if
return True
```

Algorithm B.2 Case1()

```
if $d_p > 2h$ then
  Use $\Lambda_3(x)$, eqn. (B.12), and distribute about $g_0$. Penalty = 0.
else if $d_p > h$ then
  Use $\Lambda_3'(x)$, eqn. (B.13), and distribute about $g_1$ in the negative direction. Penalty = 1.
else
  Use $\Lambda_2'(x)$, eqn. (B.15), and distribute about $g_0$ with $x = (x + 0.5)$. Negative direction. Penalty = 1.
end if
```

Using this scheme it is possible to interpolate the vorticity onto a $h \times h$ grid in the presence of complex geometries. This interpolated data can then be used for visualization (contouring) or for further analysis. The code developed for this task has been tested systematically with a suite of unit tests written in C++ using CppUnit (Feathers *et al.*, 2000–). Some of the critical functions have been

Algorithm B.3 Case2()

```
if $d_p > 2h$ then
    Use $\Lambda_3(x)$, eqn. (B.12), and distribute about $g_0$. Penalty = 0.
else if $d_p > h$ then
    Use $\Lambda_2(x)$, eqn. (B.14), and distribute about $g_0$ with $x = (x + 0.5)$. Penalty = 0.
else
    Use $\Lambda_1(x)$, eqn. (B.16), and distribute about $g_0$ with $x = (x + 0.5)$ in the negative direction. Penalty = 2.
end if
```

Algorithm B.4 Case3()

```
if $d_p > 3h$ then
    Use $\Lambda'_3(x)$, eqn. (B.13), and distribute about $g_0$. Penalty = 1.
else if $d_p > 2h$ then
    Use $\Lambda'_2(x)$, eqn. (B.15), and distribute about $g_0$ with $x = (x + 0.5)$ in the positive direction. Penalty = 1.
else if $d_p > h$ then
    Use $\Lambda_1(x)$, eqn. (B.16), and distribute about $g_0$ with $x = (x + 0.5)$ in the positive direction. Penalty = 2.
else
    Do nearest point interpolation. Penalty = 3.
    {Find nearest grid point and set its vorticity to be equal to the particles vorticity.}
end if
```

Algorithm B.5 Case4()

```
if $d_p > 3h$ then
    Use $\Lambda'_2(x)$, eqn. (B.15), and distribute about $g_1$ with $x = (x - 0.5)$ in the positive direction. Penalty = 1.
else if $d_p > 2h$ then
    Use $\Lambda_1(x)$, eqn. (B.16), and distribute about $g_1$ with $x = (x - 0.5)$ in the positive direction. Penalty = 2.
else if $d_p > h$ then
    Do nearest point interpolation. Penalty = 3.
else
    return False
    {Bad grid and geometry combination. No nearby grid point.}
end if
```

tested rigorously. As a basic test, various distributions of vortices in free space are interpolated onto regular grids with varying h values. The first three moments of the resulting distribution are compared with the exact moments. The different distributions of particles used in the testing are given below.

- A uniform distribution of particles inside a square with the vorticity of each particle being 1;
- a Gaussian distribution of particles;
- particles inside a square with positions generated using uniform random deviates with randomly generated vorticity;
- particles inside a square region with positions generated using Gaussian deviates and random vorticity (uniform deviates).

In order to test the ability of the code to handle boundaries, the case of a circle placed inside a square body is considered. Vortices are placed between the circle and the square using uniform random deviates and random vorticity values. The algorithm must not distribute vorticity outside the square and inside the circle. This is verified. The conservation of the two higher order moments are checked. Also considered is the case of a circle placed inside a rhombus. In this case, when the grid spacing is large and poorly chosen, it is possible that some vortices cannot be interpolated at all. The moments are not conserved in these cases. However, the algorithm should not distribute vorticity outside the rhombus and inside the circle. This is verified. These tests prove invaluable to find bugs in the algorithm. As seen above, the algorithm can be quite complex. The unit tests make it much easier to find bugs and document them when found. It also builds confidence and allows the programmer to change the implementation easily. This is because changes in implementation can be easily validated to produce the correct behavior by the tests.

Chapter 8 presents several vorticity contours for the impulsively started flow past a circular cylinder using this approach. Choosing the optimal mesh size, h, is not straight forward. This is addressed in the next section.

B.1.5 Optimal smoothing of vorticity

Fogelson and Dillon (1993) consider a one dimensional diffusion problem with a small diffusion coefficient. At $t = 0$, N particles are introduced within the support of the initial condition. The particles together discretize the initial function. Diffusion is then simulated using a random walk. They show that good results can be obtained if an optimal amount of smoothing is used. Specifically, if the initial condition is given as $f(x)$ and h_p is the spacing of the particles, each particle located at x_j is associated with a value of $h_p f(x_j)$. The function is approximated at later times by,

$$f(x,t) = \sum_{j=0}^{N} \frac{f_j h_p}{\epsilon\sqrt{\pi}} \, e^{(x-x_j)^2/\epsilon^2}, \tag{B.17}$$

where ϵ is given in equation (B.10) as $\epsilon = Ch^{-q}$. Fogelson and Dillon (1993) choose the value of C to be fixed at 1 and suggest using the value of $q = 0.2$. For the problems they solve, they show that using this approach produces very good results. They prove convergence as $N \to \infty$ and $\epsilon \to 0$. They also show convergence with numerical experiments as N increases and the viscosity drops. This is an important result and can be used to determine the optimal value of h used to interpolate the vorticity of vortex blobs to the grid as discussed in the previous section. However, there are three issues that need to be addressed before this approach can be used. The first issue is to determine the optimal value of C for a general problem. The second problem is to ascertain if the approach works when the data is interpolated in the manner done in the previous section. The third is to study how Laplace smoothing of the results affects the data and how it relates to optimal smoothing.

To address the choice of optimal C, the evolution of a Gaussian is studied for different viscosity values while varying the number of particles. As done by Fogelson and Dillon (1993), the function chosen is given as,

$$f(x) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} & -5\sigma < x < 5\sigma \\ 0 & \text{otherwise} \end{cases}$$

Different values are chosen for σ. The exact solution for this problem is known.

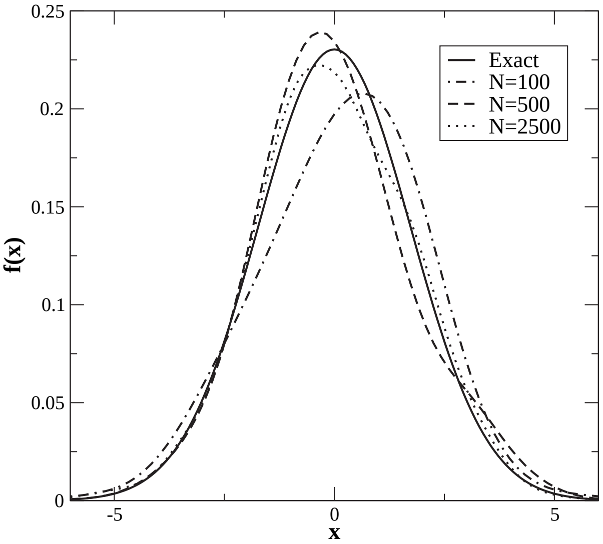

Figure B.18: Smoothed function as N varies, with $\nu = 0.1$ and $\sigma = 1$ at $t = 10$.

The solution remains a Gaussian with the value of $\sigma(t)$ given as $\sigma(t)^2 = \sigma(0)^2 + 2\nu t$, where ν is the viscosity used in the diffusion equation. After performing numerical experiments it is found that the best choice for C is a length scale associated with the function at the time of evaluation. Thus we choose $C = 2\sigma(t)$ and $q = -0.2$. This approach works even when the viscosity is quite large.

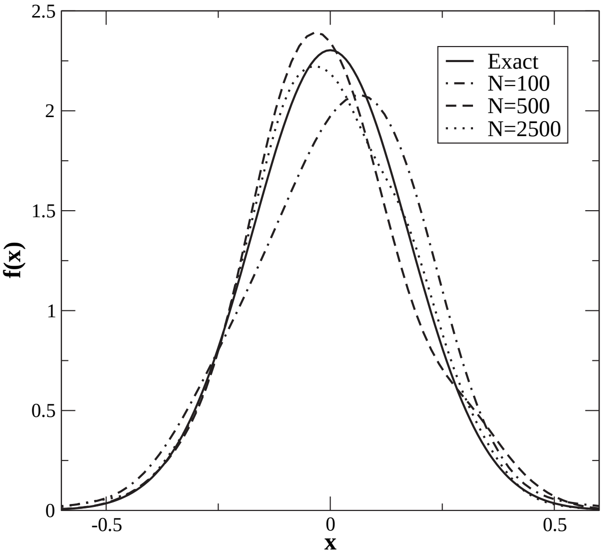

Figure B.19: Smoothed function as N varies, with $\nu = 0.001$ and $\sigma = 0.1$ at $t = 10$.

Fig. B.18 plots the approximate function at $t = 10$ for the case where $\nu = 0.1$, $\Delta t = 0.5$ and $\sigma = 1$. Fig. B.19 plots the approximate function at $t = 10$ for the case where $\nu = 0.001$, $\Delta t = 0.5$ and $\sigma = 0.1$. As can be seen, the approximating function approaches the exact solution as N increases. Though the viscosity is quite large in the case of Fig. B.18, the results continue to remain good. Despite the fact that the length scale in each figure has changed by an order of magnitude the agreement is good. Thus, the present approach of using $C = 2\sigma(t)$ works well. It is noted that if C was chosen significantly differently, the agreement was worse. Too small a C produced noisy curves and too large values produced smooth curves which had a larger error as compared to the exact solution.

Fig. B.20 plots the function obtained using linear interpolation such that the grid spacing $h = \epsilon$ as chosen for the smoothing case. As can be seen, the results are similar to Fig. B.19. The results for second order interpolation are also similar. If h is chosen significantly smaller, then the results are oscillatory as is the case with the smoothing case. However, there is one important difference between interpolation and smoothing. In smoothing, as C (or ϵ) is increased, the curve becomes smoother but is more inaccurate. With interpolation, the accuracy remains more or less the same. Fig. B.21 demonstrates this when the smoothing parameter and the interpolation grid spacing is approximately twice the optimal value ($\epsilon = h = 0.14$) with $N = 2500$. Therefore, interpolation appears to be a robust method to obtain the vorticity field given a distribution of vortex blobs. The minimum allowable grid size can be determined from the work of Fogelson and Dillon (1993) as $h = CN^{-0.2}$ with C chosen close to the length scale of the problem.

If the grid size, h, is chosen to be smaller than the optimal value, h_{opt}, one must resort to using some kind of smoothing to get good results. Laplace smoothing can be used for this purpose. Laplace smoothing works by replacing the value, y_j, at a grid point j by the following,

$$y_j = (y_{j-1} + 2y_j + y_{j+1})/4.$$

This smoothes out noisy oscillations in the curve. This smoothing can be repeatedly applied to remove noise. Consider the diffusion of the Gaussian used earlier with $\nu = 0.001$, $N = 2500$ and $\sigma = 0.1$. Second order interpolation is used to approximate the function at a time $t = 10$. Fig. B.22 plots the interpolated func-

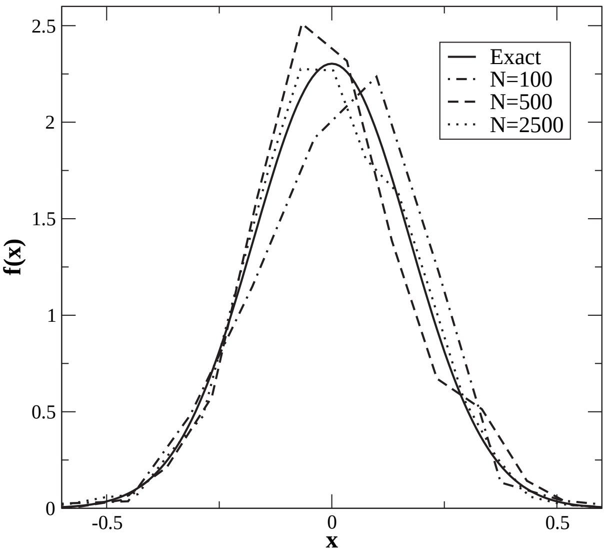

Figure B.20: Approximate function obtained using linear interpolation as N varies, with $\nu = 0.001$ and $\sigma = 0.1$ at $t = 10$.

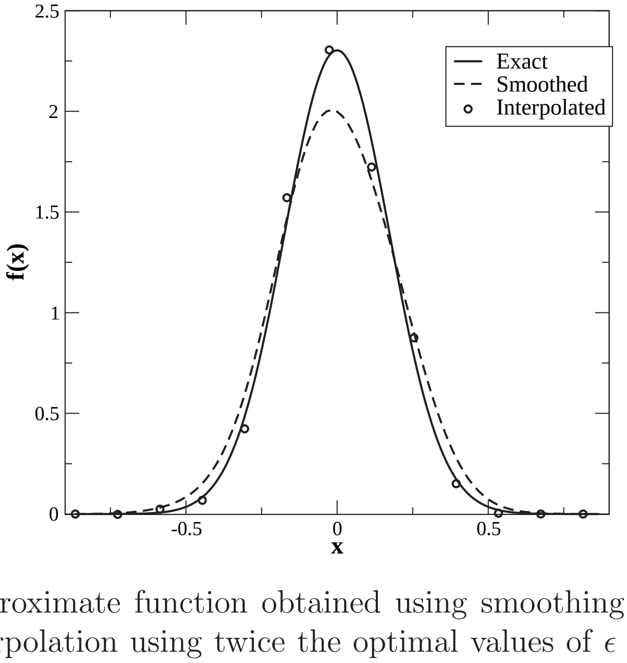

Figure B.21: Approximate function obtained using smoothing and second order interpolation using twice the optimal values of ϵ and h. $\nu = 0.001$, $\sigma = 0.1$, $t = 10$ and $N = 2500$.

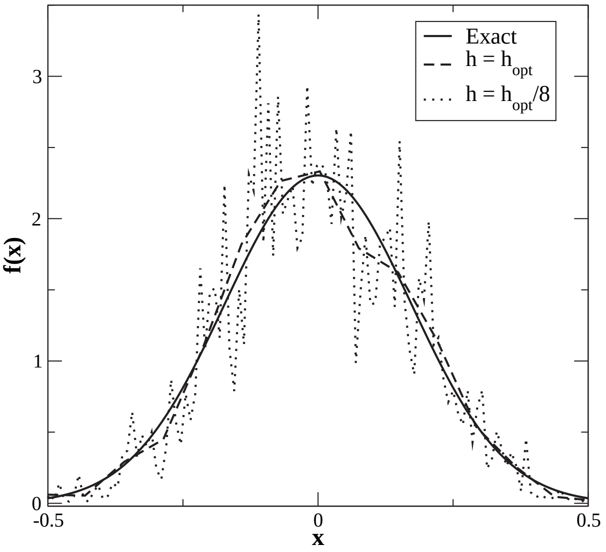

Figure B.22: Approximate function obtained using second order interpolation with different grid spacing. $\nu = 0.001$, $\sigma = 0.1$, $t = 10$ and $N = 2500$.

tion with no Laplace smoothing using the optimal value of $h = h_{opt}$ and a value eight times smaller. Clearly, the smaller h value produces very poor results. After numerical experimentation it is found that if the Laplace smoothing is applied S times with,

$$S = 2^{2n-1}, \tag{B.18}$$

where $n = \log(h_{opt}/h)/\log(2)$, excellent results are obtained. Fig. B.23 plots the results for different grid spacings using Laplace smoothing applied, S times with $h_{opt} = 0.072$. As can be seen, an optimal amount of Laplace smoothing removes the noise and produces curves that are very close to the optimally interpolated one. Thus interpolation along with Laplace smoothing is a robust and fairly simple way to obtain the approximate curve. Laplace smoothing is very useful when the grid spacing is much less than the optimal amount. These results also highlight the importance of h_{opt}.

The situation is similar for a two dimensional diffusion problem. C is again best chosen as a length scale relevant to the problem. The optimally smoothed solution produces good approximations to the exact solution. However, when

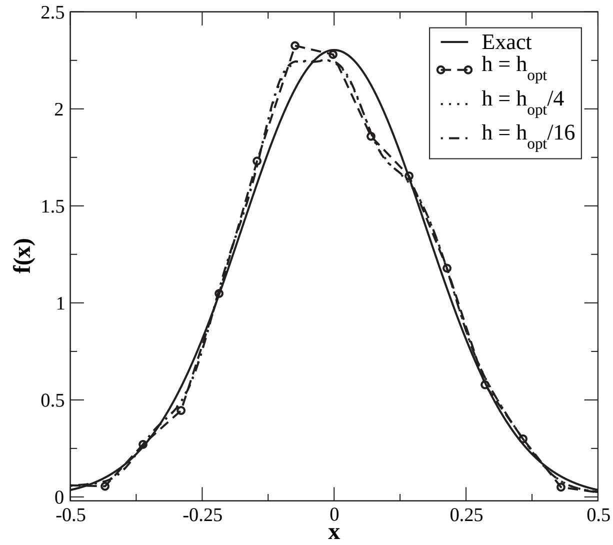

Figure B.23: Approximate function obtained using second order interpolation along with Laplace smoothing for different grid spacings h. $\nu = 0.001$, $\sigma = 0.1$, $t = 10$, $N = 2500$ and $h_{opt} = 0.072$.

using Laplace smoothing it is found that choosing S such that,

$$S = 3\ (2^{2n-2}), \tag{B.19}$$

where $n = \log(h_{opt}/h)/\log(2)$, produces better results. To demonstrate these results, a point vortex of unit circulation placed at the origin is considered at $t = 0$. It is discretized into $N = 2500$ particles carrying equal amounts of circulation. The two-dimensional diffusion problem is solved using the random walk method. The kinematic viscosity, ν is 0.001 and $\Delta t = 0.5$. At $t = 10$, the vorticity along the x axis is plotted and compared with the exact results. The interpolation is performed using the third order interpolation scheme described in section B.1.4. Fig. B.24 plots the results as different values of h are used. For the cases where $h < h_{opt}$, Laplace smoothing is performed with S as specified in equation (B.19). As seen, the results are in good agreement with the exact solution. Thus, it is possible to obtain smooth approximations to the function when using random walks to simulate diffusion.

For the flow past a cylinder it is possible to estimate both the optimal C and

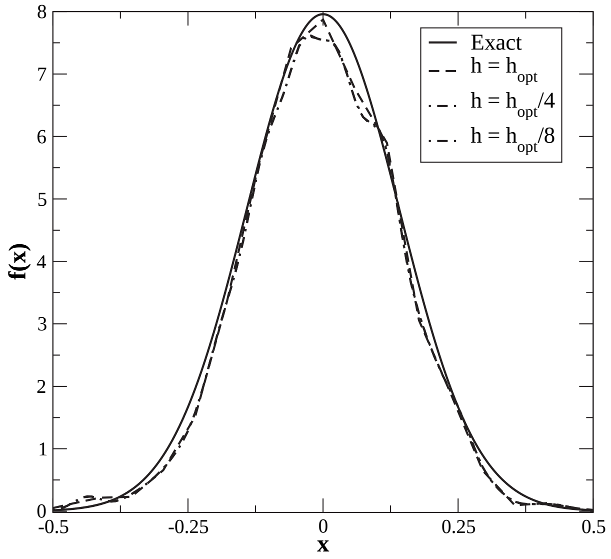

Figure B.24: Vorticity along the x axis for a two-dimensional diffusion problem along with Laplace smoothing for different grid spacings h. Third order interpolation is used. $\nu = 0.001$, $t = 10$, $N = 2500$ and $h_{opt} = 0.0592$.

N values as follows,

$$C = \sqrt{2\pi R\delta},$$

where R is the radius of the cylinder and δ is the core-radius of a blob. This value is also equal to the area of the numerical layer. N can be chosen as

$$N = \frac{V_\infty}{\gamma_{max}} N_p,$$

where V_∞ is the free stream velocity, γ_{max} is the maximum vorticity of each released vortex sheet and N_p is the number of panels used to discretize the cylinder. This value is essentially an order of estimate for the number of sheets shed for a slip velocity of V_∞. These values give an estimate for the optimal smoothing to use. Thus it is possible to employ an optimal amount of smoothing and obtain the vorticity field. The plots in chapter 8 employ these estimates to optimally smooth the vorticity. The MayaVi (Ramachandran, 2001) visualization software is used to generate the plots. The results are clearly good.

B.1.6 Streamlines, streaklines and path lines

These are relatively easy to compute and plot. They are not the best indicators of accuracy because they are integral quantities. However, they are useful diagnostics and are widely used.

Streamlines

It is possible to compute the streamlines in one of two ways.

1. Compute the velocity field at a given instant of time, seed the flow with tracer particles and integrate these particles in the flow with the known velocity field. This approach however introduces errors of its own.
2. Compute the stream function due to all constituents and contour the data to produce the streamlines.

The second approach is used in the present work. The stream function is computed using a fast multipole method. Since Anderson's fast multipole without multipoles method (Anderson, 1992) uses the stream function when applied to vorticity, this scheme is used to compute the stream function. Anderson's scheme has been implemented for both the vortex blobs and the vortex panels. The application of the method to cubic panels is described in section 4.3.2.

Inside the core of the blobs, the stream function is computed using the known exact stream function. The stream function for the Chorin, Saffman, and Krasny blobs are easy to derive. The implementation of the stream function for the higher order blobs (Gaussian and Beale Majda blobs) is involved due to the integrals that need evaluation. Hence, the stream function of the Saffman blob (also known as the Rankine blob) is used for these blobs for the near-field interaction. Outside the core-region, these blobs are treated like point vortices. For vortex panels, the stream function is known and this is used for the near-field computation. The far field interaction is performed using Anderson's scheme (Anderson, 1992) as applied to the panels.

Handling the vortex sheets is not straightforward. Currently the following simple approach is used. Each sheet is converted to an equivalent blob of appropriate strength and core radius. If the converted blob is less than one core radius from the surface of the body it is moved such that the distance to the wall is equal to the core radius. The stream function due to these blobs is then computed using the AFMM.

The code to generate the streamlines is tested using a suite of unit tests written in C++ using CppUnit (Feathers *et al.*, 2000–). The tests check the most important functions that are responsible for computing and transferring the multipoles. Both the velocity and streamlines generation are tested.

Chapter 8 shows several plots of the streamlines. The results are in excellent agreement with other computations.

Streaklines and pathlines

These are not plotted or computed at the present time but are not hard to compute. The only difficulty with these is that they need to be computed along with the fluid flow simulation and cannot be done as a part of the post processing of data unlike the streamlines.

B.1.7 Velocity vector plots

This is readily obtained for a vortex method since the velocity field is always computed for any vortex method. The velocity field is simply sampled onto a rectilinear grid and then visualized using some tool like MayaVi (Ramachandran, 2001).

B.1.8 Surface vorticity distribution

The surface vorticity distribution can be computed as described in section B.1.3 which deals with obtaining forces via pressure and skin friction. This surface vorticity distribution can be used to compute the steady separation points by finding where the skin friction goes to zero. Unsteady separation is considerably more involved and is not discussed further here.

B.1.9 Pressure distribution

The pressure term is eliminated in vortex methods. However, computing the pressure on the body surface is useful and is described in section B.1.3. Computing the pressure at an arbitrary point in the fluid is non-trivial and requires other techniques. For the purposes of the present work the pressure at an arbitrary point is not considered.

REFERENCES

1. **Abrahams, D.** and **R. W. Grosse-Kunstleve** (2003). Building hybrid systems with Boost.Python. URL `http://www.boost.org/libs/python/doc/PyConDC_2003/bpl.html`. Presented at PyCon.

2. **Abrahams, D.** *et al.* (2000–). Boost.Python. URL `http://www.boost.org/libs/python/doc/`.

3. **Anderson, C. R.** (1985). A vortex method for flows with slight density variations. *Journal of Computational Physics*, **61**, 417–444.

4. **Anderson, C. R.** (1986). A method of local corrections for computing the velocity field due to a distribution of vortex blobs. *Journal of Computational Physics*, **62**, 11–123.

5. **Anderson, C. R.** (1992). An implementation of the fast multipole method without multipoles. *SIAM J. Sci. Stat. Comput.*, **13**(4), 923–947.

6. **Appel, A. W.** (1985). An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, **6**, 85–103.

7. **Baden, S. B.** and **E. G. Puckett** (1990). A fast vortex method for computing 2d viscous flow. *Journal of Computational Physics*, **91**, 278–291.

8. **Bakhoum, E. B.** and **J. A. Board, Jr.** (1996). The geometric solution of Laplace's equation. *Journal of Computational Physics*, **123**, 274–295.

9. **Barnes, J.** and **P. Hut** (1986). A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, **324**, 446–449.

10. **Beale, J. T.** and **A. Majda** (1981). Rates of convergence for viscous splitting of the Navier-Stokes equations. *Mathematics of Computation.*

11. **Beale, J. T.** and **A. Majda** (1985). Higher order accurate vortex methods with explicit velocity kernels. *Journal of Computational Physics*, **58**, 188–208.

12. **Beazley, D.** *et al.* (1995–). SWIG: Simplified wrapper and interface generator. URL `http://www.swig.org`.

13. **Beazley, D. M.** (1996). SWIG : An easy to use tool for integrating scripting languages with C and C++. Presented at the 4th Annual Tcl/Tk Workshop, Monterey, CA.

14. **Bernard, P. S.** (1995). A deterministic vortex sheet method for boundary layer flow. *Journal of Computational Physics*, **117**, 132–145.

15. **Bracewell, R. N.**, *The Fourier transform and its applications*. McGraw-Hill Book Company, 1986, second edition.

16. **Budd, T.**, *An introduction to object-oriented programming*. Addison-Wesley, 1991, first edition. ISBN 0-201-54709-0.

17. **Carrier, J.**, **L. Greengard**, and **V. Rokhlin** (1988). A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Stat. Comput.*, **9**(4), 669–686.

18. **Chang, C.-C.** (1988). Random vortex methods for the Navier-Stokes equations. *Journal of Computational Physics*, **76**, 281–300.

19. **Cheer, A. Y.** (1983). Numerical study of incompressible slightly viscous flow past blunt bodies and airfoils. *SIAM Journal on Scientific and Statistical Computing*, **4**(4), 685–705.

20. **Cheer, A. Y.** (1989). Unsteady separated wake behind an impulsively started cylinder in slightly viscous fluid. *Journal of Fluid Mechanics*, **201**, 485–505.

21. **Chen, H.** and **J. S. Marshall** (1999). A Lagrangian vorticity method for two-phase particulate lows with two-way phase coupling. *Journal of Computational Physics*, **148**, 169–198.

22. **Choi, Y.**, **J. A. C. Humphrey**, and **F. S. Sherman** (1988). Random vortex simulation of transient wall-driven flow in a rectangular enclosure. *Journal of Computational Physics*, **75**, 359–383.

23. **Chorin, A. J.** (1973). Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, **57**(4), 785–796.

24. **Chorin, A. J.** (1978). Vortex sheet approximation of boundary layers. *Journal of Computational Physics*, **27**(3), 428–442.

25. **Chorin, A. J.** (1980). Vortex models and boundary layer instability. *SIAM Journal on Scientific and Statistical Computing*, **1**, 1–21.

26. **Chorin, A. J.** and **P. S. Bernard** (1973). Discretization of a vortex sheet, with an example of roll-up. *Journal of Computational Physics*, **13**(3), 423–429.

27. **Christiansen, J. P.** (1973). Numerical simulation of hydrodynamics by the method of point vortices. *Journal of Computational Physics*, **13**(3), 363–379.

28. **Clarke, N. R.** and **O. R. Tutty** (1994). Construction and validation of a discrete vortex method for two-dimensional incompressible Navier-Stokes equations. *Computers and Fluids*, **23**(6), 751–783.

29. **Cortez, R.** (2000). A vortex/impulse method for immersed boundary motion in high Reynolds number flows. *Journal of Computational Physics*, **160**, 385–400.

30. **Cottet, G.-H.** (1990). A particle-grid superposition method for the Navier-Stokes equations. *Journal of Computational Physics*, **89**, 301–318.

31. **Cottet, G.-H.** and **P. Koumoutsakos**, *Vortex methods: theory and practice*. Cambridge University Press, 2000.

32. **Cottet, G.-H.**, **P. Koumoutsakos**, and **M. L. O. Salihi** (2000). Vortex methods with spatially varying cores. *Journal of Computational Physics*, **162**, 164–185.

33. **Degond, P.** and **S. Mas-Gallic** (1989). The weighted particle method for convection-diffusion equations, part 1: The case of an isotropic viscosity, part 2: The anisotropic case. *Mathematics of Computation*, **53**(188), 485–526.

34. **Degond, P.** and **F.-J. Mustieles** (1990). A deterministic approximation of diffusion equations using particles. *SIAM Journal on Scientific and Statistical Computing*, **11**(2), 293–310.

35. **Draghicescu, C. I.** and **M. Draghicescu** (1995). A fast algorithm for vortex blob interactions. *Journal of Computational Physics*, **116**, 69–78.

36. **Eldredge, J. D.**, **T. Colonius**, and **A. Leonard** (2002*a*). A vortex particle method for two-dimensional compressible flow. *Journal of Computational Physics*, **179**, 371–399.

37. **Eldredge, J. D.**, **A. Leonard**, and **T. Colonius** (2002*b*). A general deterministic treatment of derivatives in particle methods. *Journal of Computational Physics*, **180**, 686–709.

38. **Feathers, M.** *et al.* (2000–). CppUnit: A C++ port of the JUnit framework for unit testing. URL `http://cppunit.sourceforge.net/`.

39. **Fishelov, D.** (1990). A new vortex scheme for viscous flows. *Journal of Computational Physics*, **86**, 211–224.

40. **Fogelson, A. L.** and **R. H. Dillon** (1993). Optimal smoothing in function-transport particle methods for diffusion problems. *Journal of Computational Physics*, **109**, 155–163.

41. **Gardiner, C. W.**, *A Handbook of Stochastic Methods*. Springer-Verlag, Berlin, 1985, second edition.

42. **Ghoniem, A. F.**, **A. J. Chorin**, and **A. K. Oppenheim** (1982). Numerical modelling of turbulent flow in a combustion tunnel. *Philosophical transactions of the royal society of London, Series A*, **304**, 303–325.

43. **Ghoniem, A. F.** and **Y. Gagnon** (1987). Vortex simulation of laminar recirculating flow. *Journal of Computational Physics*, **68**, 346–377.

44. **Ghoniem, A. F.**, **G. Heidarinejad**, and **A. Krishnan** (1988). Numerical simulation of a thermally stratified shear layer using the vortex element method. *Journal of Computational Physics*, **79**, 135–166.

45. **Ghoniem, A. F.** and **K. K. Ng** (1987). Numerical study of the dynamics of a forced shear layer. *Physics of Fluids*, **30**(3), 706–721.

46. **Ghoniem, A. F.** and **F. S. Sherman** (1985). Grid free simulation using random walk methods. *Journal of Computational Physics*, **61**, 1–37.

47. **Goodman, J.**, **T. Hou**, and **J. Lowengrub** (1990). Convergence of the point vortex method for the 2-D Euler equations. *Communications in Pure and Applied Mathematics*, **43**, 415–430.

48. **Greengard, C.** (1985). The core spreading vortex method approximates the wrong equation. *Journal of Computational Physics*, **61**, 345–348.

49. **Greengard, L.** and **V. Rokhlin** (1987). A fast algorithm for particle simulations. *Journal of Computational Physics*, **73**, 325–348.

50. **Heemink, A. W.** and **P. A. Blokland** (1995). On random walk models with space varying diffusivity. *Journal of Computational Physics*, **119**, 388–389.

51. **Huyer, S. A.** and **J. R. Grant** (1996). Computation of unsteady separated flow fields using anisotropic vorticity elements. *Journal of Fluids Engineering*, **118**, 839–849.

52. **Jones, E.**, **T. Oliphant**, **P. Peterson**, *et al.* (2001–). SciPy: Open source scientific tools for Python. URL `http://www.scipy.org/`.

53. **Katz, J.** and **A. Plotkin**, *Low-Speed Aerodynamics: From Wing Theory to Panel Methods*. McGraw–Hill Education, New York, 1991.

54. **Kida, T.** (1998). Theoretical and numerical results of a deterministic two-dimensional vortex model. *Sadhana*, **23**(5,6), 419–441.

55. **Kim, T.** and **M. R. Flynn** (1995). Numerical simulation of air flow around multiple objects using the discrete vortex method. *Journal of Wind Engineering and Industrial Aerodynamics*, **56**, 213–234.

56. **Koumoutsakos, P.** (1997). Inviscid axisymmetrization of an elliptical vortex. *Journal of Computational Physics*, **138**, 821–857.

57. **Koumoutsakos, P.** and **A. Leonard**, Direct numerical simulations using vortex methods. *In* **J. T. Beale**, **G. H. Cottet**, and **S. Huberson** (eds.), *Vortex Flows and Related Numerical Methods*. Kluwer Academic Publishers, Netherlands, 1993, 179–190.

58. **Koumoutsakos, P.** and **A. Leonard** (1995). High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *Journal of Fluid Mechanics*, **296**, 1–38.

59. **Koumoutsakos, P.**, **A. Leonard**, and **F. Pepin** (1994). Boundary conditions for viscous vortex methods. *Journal of Computational Physics*, **113**, 52–61.

60. **Koumoutsakos, P. D.** (1993). *Direct numerical simulations of unsteady separated flows using vortex methods*. Ph.D. thesis, California Institute of Technology, Pasadena, California.

61. **Krasny, R.** (1986). Desingularization of periodic vortex sheet roll-up. *Journal of Computational Physics*, **65**, 292–313.

62. **Krasny, R.** (1987). Computation of vortex sheet roll-up in the Trefftz plane. *Journal of Fluid Mechanics*, **184**, 123–155.

63. **Krishnan, A.** and **A. F. Ghoniem** (1992). Simulation of rollup and mixing in Rayleigh-Taylor flow using the transport element method. *Journal of Computational Physics*, **99**, 1–27.

64. **Leonard, A.** (1980). Vortex methods for flow simulation. *Journal of Computational Physics*, **37**, 289–335.

65. **Lewis, R. I.**, *Vortex Element Methods for Fluid Dynamical Analysis of Engineering Systems*. Cambridge University Press, Cambridge, UK, 1991.

66. **Lin, H.**, **M. Vezza**, and **R. A. McD. Galbraith** (1997). Discrete vortex method for simulating unsteady flow around pitching aerofoils. *AIAA Journal*, **35**(3), 494–499.

67. **Lubachevsky, B. D.** (1991). How to simulate billiards and similar systems. *Journal of Computational Physics*, **94**, 255–283.

68. **Lustig, S. R.**, **S. Rastogi**, and **N. Wagner** (1995). Telescoping fast multipole methods using Chebyshev economisation. *Journal of Computational Physics*, **122**, 317–322.

69. **Makino, J.** (1999). Yet another fast multipole method without multipoles — pseudo-particle multipole method. *Journal of Computational Physics*, **151**(2), 910–920.

70. **Marshall, J. S.** and **J. R. Grant** (1996). A method for determining the velocity induced by highly anisotropic vorticity blobs. *Journal of Computational Physics*, **126**, 286–298.

71. **Meiburg, E.** (1989). Incorporation and test of diffusion and strain effects in the two-dimensional vortex blob technique. *Journal of Computational Physics*, **82**, 85–93.

72. **Mortazavi, I.**, **P. Micheau**, and **A. Giovannini**, Numerical convergence of the random vortex methods for complex flows. *In Vortex flows and related numerical methods II, Proceedings of the second international workshop on vortex flows and related numerical methods*, volume 1. European Series in Applied and Industrial Mathematics (ESAIM), Montreal, Canada, 1996.

73. **Nordmark, H. O.** (1991). Rezoning for higher order vortex methods. *Journal of Computational Physics*, **97**, 366–397.

74. **Nordmark, H. O.** (1996). Deterministic high order vortex methods for the 2D Navier-Stokes equation with rezoning. *Journal of Computational Physics*, **129**, 41–56.

75. **Ogami, Y.** and **T. Akamatsu** (1991). Viscous flow simulation using the discrete vortex model – the diffusion velocity method. *Computers and Fluids*, **19**, 433–441.

76. **Ousterhout, J. K.** (1998). Scripting: higher level programming for the 21st century. *IEEE Computer*, **31**(3), 23–30.

77. **Peitgen, H.-O.**, **H. Jurgens**, and **D. Saupe**, *Chaos and Fractals: New Frontiers of Science*. Springer-Verlag, New York, 1992.

78. **Perlman, M.** (1985). On the accuracy of vortex methods. *Journal of Computational Physics*, **59**, 200–223.

79. **Peters, M. C. A. M.** and **H. W. M. Hoeijmakers** (1995). A vortex sheet method applied to unsteady flow separation from sharp edges. *Journal of Computational Physics*, **120**, 88–104.

80. **Ploumhans, P.** and **G. S. Winckelmans** (2000). Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry. *Journal of Computational Physics*, **165**, 354–406.

81. **Ploumhans, P.**, **G. S. Winckelmans**, and **J. K. Salmon**, Vortex particles and tree codes: I. flows with arbitrary crossing between solid boundaries and particle redistribution lattice; ii. vortex ring encountering a plane at an angle. *In* **A. Giovannini**, **G.-H. Cottet**, **Y. Gagnon**, **A. F. Ghoniem**, and **E. Meiburg** (eds.), *Vortex flows and related numerical methods III, Proceedings of the third international workshop on vortex flows and related numerical methods*, volume 7. European Series in Applied and Industrial Mathematics (ESAIM), Toulouse, France, 1999.

82. **Porthouse, D. T. C.** and **R. I. Lewis** (1981). Simulation of viscous diffusion for extension of the surface vorticity method to boundary layer and separated flows. *Journal Mechanical Engineering Science*, **23**(3), 157–167.

83. **Press, W. H.**, **S. A. Teukolsky**, **W. T. Vetterling**, and **B. P. Flannery**, *Numerical Recipes in 'C': The art of scientific computing*. Cambridge University Press, Cambridge, UK, 1992.

84. **Puckett, E. G.** (1989). A study of the vortex sheet method and its rate of convergence. *SIAM Journal on Scientific and Statistical Computing*, **10**(2), 298–327.

85. **Puckett, E. G.**, Vortex methods: An introduction and survey of selected research topics. *In* **R. A. Nicolaides** and **M. D. Gunzburger** (eds.), *Incompressible Computational Fluid Dynamics — Trends and Advances*. Cambridge University Press, 1991, 335.

86. **Qian, L.** and **M. Vezza** (2001). A vorticity-based method for incompressible unsteady viscous flows. *Journal of Computational Physics*, **172**, 515–542.

87. **Rajan, S. C.** (1994). Unification of ideal fluid and rigid body kinematics using vorticity. *Mechanics Research Communications*, **21**(5), 443–448.

88. **Ramachandran, P.**, MayaVi: A free tool for CFD data visualization. *In 4th Annual CFD Symposium*. Aeronautical Society of India, 2001. Software available at: http://mayavi.sf.net.

89. **Ramachandran, P.**, **S. C. Rajan**, and **M. Ramakrishna**, An accurate two-dimensional panel method. *In Seminar on advances in aerospace technologies, SAAT-2000*. Aeronautical Society of India, 2000*a*.

90. **Ramachandran, P.**, **S. C. Rajan**, and **M. Ramakrishna** (2003). A fast, two-dimensional panel method. *SIAM Journal on Scientific Computing*, **24**(6), 1864–1878.

91. **Ramachandran, P.**, **S. C. Rajan**, and **M. Ramakrishna** (In press). A fast multipole method for higher order vortex panels in two dimensions. *SIAM Journal on Scientific Computing*.

92. **Ramachandran, P.**, **M. Ramakrishna**, and **S. C. Rajan** (2000*b*). An efficient vortex diffusion implementation for flow past arbitrary bodies in two dimensions. Technical Report AE:CFL:TR:2000:1, IIT-Madras, Computers and Fluids Lab, Dept. Aerospace Eng. IIT-Madras, Chennai, INDIA - 600 036.

93. **Ramachandran, P.**, **M. Ramakrishna**, and **S. C. Rajan** (2001). An efficient vortex diffusion algorithm for flow past arbitrary bodies in two dimensions. Technical Report AE:CFL:TR:2001:1, IIT-Madras, Computers and Fluids Lab, Dept. Aerospace Eng. IIT-Madras, Chennai, INDIA - 600 036.

94. **Ramachandran, P.**, **M. Ramakrishna**, and **S. C. Rajan** (Under review). Efficient random walks in the presence of complex two-dimensional geometries.

95. **Roache, P. J.**, *Verification and Validation in Computational Science and Engineering*. Hermosa publishers, PO Box 9110, Albuquerque, New Mexico 87119-9110 USA, 1998. ISBN 0-913478-08-3.

96. **Roberts, S. G.** (1985). Accuracy of the random vortex methods for a problem with non-smooth initial conditions. *Journal of Computational Physics*, **58**, 29–43.

97. **Rokhlin, V.** (1985). Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, **60**, 187–207.

98. **Rossi, L.** (1996). Resurrecting core spreading vortex methods: A new scheme that is both deterministic and convergent. *SIAM Journal on Scientific Computing*, **17**(2), 370–397.

99. **Rossi, L.** (1997). Merging computational elements in vortex simulations. *SIAM Journal on Scientific Computing*, **18**(4), 1014–1027.

100. **Russo, G.** and **J. A. Strain** (1994). Fast triangulated vortex methods for the 2d Euler equations. *Journal of Computational Physics*, **111**, 291–323.

101. **Salmon, J. K.** and **M. S. Warren** (1994). Skeletons from the treecode closet. *Journal of Computational Physics*, **111**, 136–155.

102. **Sethian, J.** (1984). Turbulent combustion in open and closed vessels. *Journal of Computational Physics*, **54**, 425–456.

103. **Sethian, J. A.** and **A. F. Ghoniem** (1988). Validation study of vortex methods. *Journal of Computational Physics*, **74**, 283–317.

104. **Shankar, S.** (1996). *A new mesh-free vortex method*. Ph.D. thesis, The Florida State University, FAMU-FSU College of Engineering.

105. **Shankar, S.** and **L. van Dommelen** (1996*a*). A new diffusion procedure for vortex methods. *Journal of Computational Physics*, **127**, 88–109.

106. **Shankar, S.** and **L. L. van Dommelen**, A new diffusion scheme in vortex methods for three-dimensional incompressible flows. *In Vortex flows and related numerical methods II, Proceedings of the second international workshop on vortex flows and related numerical methods*, volume 1. European Series in Applied and Industrial Mathematics (ESAIM), Montreal, Canada, 1996*b*.

107. **Shashidhar, S.** (1998). *Simulation of two dimensional vortex flows*. Master's thesis, Department of Aerospace Enginneering, IIT-Madras.

108. **Shiels, D.** (1998). *Simulation of controlled bluff body flow with a viscous vortex method*. Ph.D. thesis, California Institute of Technology, Pasadena, California.

109. **Smith, P. A.** and **P. K. Stansby** (1985). Wave-induced bed flows by a Lagrangian vortex scheme. *Journal of Computational Physics*, **60**, 489–516.

110. **Smith, P. A.** and **P. K. Stansby** (1988). Impulsively started flow around a circular cylinder by the vortex method. *Journal of Fluid Mechanics*, **194**, 45–77.

111. **Smith, P. A.** and **P. K. Stansby** (1989). An efficient surface algorithm for random-particle simulation of vorticity and heat transport. *Journal of Computational Physics*, **81**, 349–371.

112. **Speziale, C. G.** (1987). On the advantages of the vorticity-velocity formulation of the equations of fluid dynamics. *Journal of Computational Physics*, **73**, 476–480.

113. **Strain, J.** (1992). Fast potential theory II. layer potentials and discrete sums. *Journal of Computational Physics*, **99**, 251–270.

114. **Strain, J.** (1996). 2D vortex methods and singular quadrature rules. *Journal of Computational Physics*, **124**, 131–145.

115. **Strain, J.** (1997). Fast adaptive 2D vortex methods. *Journal of Computational Physics*, **132**, 108–122.

116. **Strickland, J. H.** and **R. S. Baty** (1998). Modification of the Carrier, Greengard and Rokhlin FMM for independent source and target fields. *Journal of Computational Physics*, **142**(1), 123–128.

117. **Stroustrup, B.**, *The C++ programming language*. Addison-Wesley, 1998, third edition.

118. **Summers, D. M.** (1989). A random vortex simulation of Falkner-Skan boundary layer flow. *Journal of Computational Physics*, **85**, 86–103.

119. **Summers, D. M.** (2000). A representation of bounded viscous flow based on Hodge decomposition of wall impulse. *Journal of Computational Physics*, **158**, 28–50.

120. **Takeda, K.**, **O. R. Tutty**, and **D. A. Nicole**, Parallel discrete vortex methods on commodity supercomputers; an investigation into bluff body far wake behaviour. *In Vortex flows and related numerical methods III, Proceedings of the third international workshop on vortex flows and related numerical methods*, volume 7. European Series in Applied and Industrial Mathematics (ESAIM), Toulouse, France, 1999.

121. **Taylor, I.** and **M. Vezza** (1999*a*). Calculation of the flow around a square section cylinder undergoing forced transverse oscillations using a discrete vortex method. *Journal of Wind Engineering and Industrial Aerodynamics*, **82**, 271–291.

122. **Taylor, I.** and **M. Vezza** (1999*b*). Prediction of unsteady flow around square and rectangular cylinders using a discrete vortex method. *Journal of Wind Engineering and Industrial Aerodynamics*, **82**, 247–269.

123. **Teng, Z.-H.** (1982). Elliptic-vortex method for incompressible flow at high Reynolds number. *Journal of Computational Physics*, (46), 54–68.

124. **Tryggvason, G.** (1988). Numerical simulations of the Rayleigh-Taylor instability. *Journal of Computational Physics*, **75**, 253–282.

125. **Tryggvason, G.** (1989). Simulation of vortex sheet roll-up by vortex methods. *Journal of Computational Physics*, **80**, 1–16.

126. **van Dommelen, L.** and **E. A. Rundensteiner** (1989). Fast, adaptive summation of point forces in the two-dimensional Poisson equation. *Journal of Computational Physics*, **83**, 126–147.

127. **van Rossum, G.** *et al.* (1991–). The Python programming language. URL `http://www.python.org/`.

128. **Vosbeek, P. W. C.**, **H. J. H. Clercx**, and **R. M. M. Mattheij** (2000). Acceleration of contour dynamics simulations with a hierarchical-element method. *Journal of Computational Physics*, **161**, 287–311.

129. **Yon, S. A.** (1990). *Systematic Formulation and Analysis of 2-D Panel Methods*. Master's thesis, San Diego State University.

130. **Zabusky, N. J.**, **M. H. Hughes**, and **K. V. Roberts** (1979). Contour dynamics for the Euler equations in two dimensions. *Journal of Computational Physics*, **30**(1), 96–106.

LIST OF PAPERS BASED ON THESIS

1. Prabhu Ramachandran, S. C. Rajan, and M. Ramakrishna. An accurate two-dimensional panel method. In *Seminar on advances in aerospace technologies, SAAT-2000.* Aeronautical Society of India, January 2000.

2. Prabhu Ramachandran, M. Ramakrishna, and S. C. Rajan. Particle based flow solvers for incompressible flows in two dimensions: impulsively started flow past a circular cylinder. In *3rd Annual CFD Symposium.* Aeronautical Society of India, August 2000.

3. Prabhu Ramachandran, M. Ramakrishna, and S. C. Rajan. Particle based flow solvers for incompressible flows in two dimensions: impulsively started flow past a circular cylinder. *Journal of the Aeronautical Society of India*, 53(2), 102–110, May 2001.

4. Prabhu Ramachandran, S. C. Rajan, and M. Ramakrishna. A fast, two-dimensional panel method. *SIAM Journal on Scientific Computing*, 24(6), 1864–1878, 2003.

5. Prabhu Ramachandran, S. C. Rajan, and M. Ramakrishna. A fast multipole method for higher order vortex panels in two dimensions. *SIAM Journal on Scientific Computing*, In press.

6. Prabhu Ramachandran, M. Ramakrishna, and S. C. Rajan. Efficient random walks in the presence of complex two-dimensional geometries. Under review.

7. Prabhu Ramachandran. Python for CFD: a case study. In *SciPy'04 - Python for Scientific Computing.* CalTech, Pasadena, CA September 2-3, 2004.

CURRICULUM VITAE

1. NAME: Prabhu Ramachandran

2. DATE OF BIRTH: 8 September 1975

3. EDUCATIONAL QUALIFICATIONS

1997 Bachelor of Technology (B.Tech.)

Institution : Indian Institute of Technology, Madras
Specialization : Aerospace Engineering

Doctor of Philosophy (Ph.D.)

Institution : Indian Institute of Technology, Madras
Specialization : Aerospace Engineering
Registration date : 30-06-1999

DOCTORAL COMMITTEE

CHAIRPERSON:
Job Kurian
Professor and Head
Department of Aerospace Engineering

GUIDES:
S. C. Rajan
Assistant Professor
Department of Aerospace Engineering

M. Ramakrishna
Professor
Department of Aerospace Engineering

MEMBERS:
S. Santhakumar
Professor
Department of Aerospace Engineering

E. G. Tulapurkara
Professor
Department of Aerospace Engineering

V. Balabaskaran
Professor
Department of Mechanical Engineering

R. Usha
Professor
Department of Mathematics