ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

AN IMMERSED BOUNDARY IMPLEMENTATION USING A HIGH ORDER COMPACT SCHEME ON A GRAPHICS PROCESSING UNIT

M.Sc. THESIS

Ufuk ÖZCAN

Department of Aeronautical and Astronautical Engineering

Aeronautical and Astronautical Engineering Programme

MAY 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

AN IMMERSED BOUNDARY IMPLEMENTATION USING A HIGH ORDER COMPACT SCHEME ON A GRAPHICS PROCESSING UNIT

M.Sc. THESIS

Ufuk ÖZCAN
(511111169)

Department of Aeronautical and Astronautical Engineering

Aeronautical and Astronautical Engineering Programme

Thesis Advisor: Prof. Dr. Fırat Oğuz EDİS

MAY 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GRAFİK İŞLEME BİRİMİ ÜZERİNDE YÜKSEK MERTEBE KOMPAKT ŞEMA KULLANILARAK GÖMÜLÜ SINIR UYGULAMASI

YÜKSEK LİSANS TEZİ

Ufuk ÖZCAN
(511111169)

Uçak ve Uzay Mühendisliği Anabilim Dalı

Uçak ve Uzay Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Fırat Oğuz EDİS

MAYIS 2015

Ufuk Özcan, a **M.Sc.** student of ITU **Graduate School of Science Engineering and Technology** student ID **511111169**, successfully defended the **thesis** entitled "**AN IMMERSED BOUNDARY IMPLEMENTATION USING A HIGH ORDER COMPACT SCHEME ON A GRAPHICS PROCESSING UNIT**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Prof. Dr. Fırat Oğuz EDİS**
Istanbul Technical University

Jury Members : **Prof. Dr. Aydın Mısırlıoğlu**
Istanbul Technical University

Assistant Prof. Levent Ali Kavurmacıoğlu
Istanbul Technical University

Date of Submission : 27 April 2015
Date of Defense : 08 May 2015

FOREWORD

I would like to offer my sincerest gratitude to my thesis supervisor Prof. Fırat Oğuz Edis who has supported me throughout this study with his invaluable guidance and experience.

A very special thanks goes to Dr. Bülent Tutkun whose pioneering work during his PhD. study about computational fluid dynamics on graphics processing units caused a jump start for me and made this thesis possible for a relatively shorter time. I am incredible grateful for his invaluable guiding, support and patience throughout this study. Considering all his contributions, I can hardly call this my work.

I am also thankful to my fiancée Arch. Nevriye Öztürk for all her patience, support throughout this study and assistance during the organization of this thesis.

May 2015 Ufuk ÖZCAN

TABLE OF CONTENTS

Page

FOREWORD ..... **vii**
TABLE OF CONTENTS ..... **ix**
ABBREVIATIONS ..... **xi**
LIST OF TABLES ..... **xiii**
LIST OF FIGURES ..... **xv**
SUMMARY ..... **xvii**
ÖZET ..... **xix**
1. INTRODUCTION **1**
2. GOVERNING EQUATIONS **5**
3. NUMERICAL METHODS **7**
3.1 Temporal Discretization 7
3.2 Immersed Boundary Method 7
3.2.1 A direct forcing approach 9
3.2.2 Interpolation scheme 10
3.3 Compact Finite Difference Schemes 12
3.4 Conjugate Gradient Method 16
4. GENERAL PURPOSE COMPUTING ON GRAPHICS PROCESSING UNIT (GPGPU) **19**
4.1 GPU Architecture and Code Design 20
5. APPLICATIONS **23**
5.1 Flow Over A Flat Plate 23
5.2 Lid Driven Cavity Flow 26
5.3 Flow Around Circular Cylinder 33
6. CONCLUSION **39**
REFERENCES **41**
CURRICULUM VITAE **47**

ABBREVIATIONS

CFD : Computational Fluid Dynamics
CPU : Central Processing Unit
CUDA : Compute Unified Device Architecture
GFLOPS : Giga Floating-Point Operations per Second
GPGPU : General-Purpose Computing on Graphical Processing Units
GPU : Graphics Processing Unit
IBM : Immersed Boundary Method

LIST OF TABLES

Page

Table 4.1: General specifications of Tesla C1060 20
Table 5.1: Reference results for flow around cylinder..... 36

LIST OF FIGURES

Page

Figure 1.1: Preparation steps of the body-fitted mesh and IBM 2
Figure 3.1: Treatment of the interface cells. 11
Figure 3.2: Schematic of the interpolation stencil in two dimensions. (a)–(c) Possible choices depending on the boundary topology and local grid size..... 12
Figure 3.3: Resolution characteristics of various central schemes for the first derivative 15
Figure 3.4: Resolution characteristics of various central schemes for the second derivative..... 15
Figure 3.5: Solution algorithm for conjugate gradient method..... 17
Figure 4.1: A Tesla C1060 GPU 20
Figure 4.2: A representation of CUDA structure 21
Figure 4.3: Representation of CUDA grid 22
Figure 4.4: Computation structure for a 3D flow domain 22
Figure 5.1: Contours of velocity [m/s]..... 24
Figure 5.2: Boundary layer thickness along the flat plate..... 24
Figure 5.3: Profiles of x and y components of velocity inside boundary layer at various Reynolds numbers. 25
Figure 5.4: Schematic diagram of lid driven cavity flow. 26
Figure 5.5: Results for v-velocity along horizontal line through geometric center of cavity for Re=100..... 27
Figure 5.6: Results for u-velocity along vertical line through geometric center of cavity for Re=100..... 27
Figure 5.7: Velocity contours inside the cavity for Re=100; top: u-velocity [m/s]; bottom: v-velocity [m/s]..... 28
Figure 5.8: Results for v-velocity along horizontal line through geometric center of cavity for Re=400..... 29
Figure 5.9: Results for u-velocity along vertical line through geometric center of cavity for Re=400..... 29
Figure 5.10: Results for v-velocity along horizontal line through geometric center of cavity for Re=1000. 30
Figure 5.11: Results for u-velocity along vertical line through geometric center of cavity for Re=1000. 30
Figure 5.12: Velocity contours inside the cavity for Re=400; top: u-velocity [m/s]; bottom: v-velocity [m/s]. 31
Figure 5.13: Velocity contours inside the cavity for Re=1000; top: u-velocity [m/s]; bottom: v-velocity [m/s]. 32
Figure 5.14: The xy-plane of the computational domain..... 33
Figure 5.15: The development of flow field; a)0s, b)2s, c)4s, d)6s, e)8s, f)14s, g)17s, h)20s, i)23s..... 34
Figure 5.16: Flow field behind the cylinder; 141s – 153s. 34

Figure 5.17: Time history of v-velocity. 36
Figure 5.18: Pressure field for half period of vortex shedding cycle [Pa]: a) t = 150 s. b) t = 151 s. c) t = 152 s. d) t = 153 s. 37
Figure 5.19: Velocity field for half period of vortex shedding cycle [m/s]: a) t = 150 s. b) t = 151 s. c) t = 152 s. d) t = 153 s..... 38

AN IMMERSED BOUNDARY IMPLEMENTATION USING A HIGH ORDER COMPACT SCHEME ON A GRAPHICS PROCESSING UNIT

SUMMARY

Computational Fluid Dynamics (CFD) has become a major tool for simulating the fluid flow in various areas. It is a cost and time effective way compared to experimental methods. Even though it is not always possible to obtain realistic solutions concerning complex high Reynolds number flows due to the hardware problems, numerous researches have been dedicated to find ways to overcome this shortness.

Numerical methods that are used to perform CFD analyses define the accuracy and duration of simulations. For instance, in order to get accurate results with low order numerical methods, very high grid resolution is required and high grid resolution yields to long computation periods with high memory usage. To overcome this drawback higher order schemes are used commonly to get reliable solutions with coarser grids. Spectral methods are more reliably than standard finite difference methods but the computational cost of these schemes are very high. Compact higher order finite difference schemes have spectral like resolution ability but simpler structure. Compared to standard finite difference schemes, compact schemes use smaller stencil and give better resolution with the same order of accuracy. In this study a sixth order compact scheme is used to obtain results of several incompressible flow test cases; flow over a flat plate, lid driven cavity flow and flow around a circular cylinder.

Mesh generation is another time consuming part of CFD simulations. As an alternative to the body-fitted meshes, immersed boundary method (IBM) which is a very efficient and straightforward way to generate grid has been used widely. With IBM one does not need to generate mesh that conforms the body, but change the algorithm of the solution to impose the boundaries. In this method a simple mesh such as cartesian mesh is generated without the body and then in the solution step the presence of the body is introduced by the modifications of the governing equations. Compared to body-fitted structured or unstructured grids, in IBM less memory is required and grid generation is very easy. An IB method is applied in the test cases of this thesis study to represent solid boundaries.

Another achievement of this study is that it uses a graphics processing unit (GPU) as a computational platform. General purpose computing on GPUs has recently become a very efficient way compared to central processing unit (CPU) in computational sciences. The computations of the present study are performed on one of NVIDIA's scientific computing GPUs Tesla C1060 which has 30 streaming multi-processors, each containing 8 scalar processors clocked at 1.3 GHz with a bandwidth of 102 GB/s to 4 GB GDDR3 global memory. NVIDIA's Compute Unified Device Architecture (CUDA) toolkit which is a complete software development solution for programming CUDA-enabled GPUs is utilized as the code design medium.

In conclusion, three test cases, flow over a flat plate, lid driven cavity flow and flow around a circular cylinder, are solved using immersed boundary method and sixth order compact finite difference scheme on a GPU. Results obtained from flow over a flat plate test case are compared to Blasius boundary layer theorem solutions. Lid driven cavity flow and flow around a circular cylinder results are compared to the ones found on literature. All results obtained in this study are in very good agreement with theoretical and reference results.

GRAFİK İŞLEME BİRİMİ ÜZERİNDE YÜKSEK MERTEBE KOMPAKT ŞEMA KULLANILARAK GÖMÜLÜ SINIR UYGULAMASI

ÖZET

Hesaplamalı akışkanlar dinamiği (HAD) yöntemi, akademik ve endüstriyel çalışmalarda akışkan hareketinin incelenmesinde kullanılan önemli bir araç haline gelmiştir. HAD deneysel yöntemlere kıyasla maliyet ve süre açısından daha avantajlı bir yöntemdir. Karmaşık ve yüksek Reynolds sayılı akışların simülasyonu henüz donanımsal yetersizlikler yüzünden her zaman mümkün olmasada, güvenilir sonuçların alınabilmesi için bir çok akademik ve endüstriyel çalışma yapılmaktadır. Simülasyonlarda kullanılan sayısal yöntemler, bu yöntemlerin doğruluk, hız ve hafıza kullanımı açısından performansa etkileri oldukça çok araştırılan konulardır. Analizler için gerekli hesaplama ağları ve bu hesaplama ağlarının oluşturuluş biçimi çok fazla araştırılan başka bir konudur. Grafik işleme birimlerinin (GPU) bilimsel hesaplamalarda kullanılmaya başlandığı 2000'li yılların başından beri incelenen başka bir konu ise simülasyonların yapıldığı hesaplama platformunun performansa etkileridir.

HAD analizlerinde kullanılan sayısal yöntemler sonuçların doğruluğunu ve simülasyon süresini etkiler. Örneğin düşük mertebeli şemalar ile yüksek doğruluk elde edebilmek için çok yoğun grid yapısına ihtiyaç vardır. Bu durum da büyük miktarda hafızaya ihtiyaç duyulması ve hesaplamanın uzun sürmesi gibi olumsuzluklara neden olmaktadır. Bu sorunun çözümü için, daha az yoğun grid yapısıyla, düşük mertebeli şemalara kıyasla, daha güvenilir sonuçlar verebilen yüksek mertebe şemalar sıkça kullanılmaktadır. Oldukça yüksek doğruluk veren başka bir yöntem de, çözüm aşamasında hesaplama ağındaki tüm noktalardaki türev değerlerini kullanan, spektral yöntemlerdir. Spektral yöntemler standart sonlu farklar yöntemine kıyasla daha güvenilir sonuçlar vermektedir, ancak spektral şemaların hesaplama maliyeti çok yüksektir. Kompakt şemalar, spektral şemalara benzer çözüm yeteneğine sahiptir ve daha basit bir yapıdadır. Kompakt sonlu farklar şemaları, aynı doğruluk mertebesinde standart sonlu farklar yöntemine kıyasla daha iyi sonuç verir ve daha düşük sayıda düğüm noktası kullanırlar. Daha düşük düğüm noktası sayısıyla daha iyi sonuç vermeleri, kompakt şemaları HAD simülasyonlarında oldukça cazip kılmaktadır. Bu çalışma kapsamında, konumsal ayrıklaştırmalar için altıncı mertebe kompakt sonlu farklar şeması kullanılarak, çeşitli sıkıştırılamaz akış tiplerinin HAD analizleri yapılmıştır. Simülasyonu gerçekleştirilen test akışları; ince levha üzerindeki akış, üst duvarı hareketli kavite içerisindeki akış ve dairesel silindir etrafındaki akıştır.

Mesh oluşturma işlemi HAD simülasyonlarının önemli ve zaman alıcı bir bölümüdür. Yapı üzerine giydirilen gridler, HAD analizlerinde en çok kullanılan hesaplama ağlarıdır. Bu tip gridlerde oluşturulan hesaplama ağı incelenen katı sınırları sarmaktadır ve çözüm esnasında koordinat dönüşümü yapılarak gerekli sınır şartları sağlanmaktadır. Ancak geometri karmaşıklaştıkça bu grid yapısını oluşturmak oldukça zahmetli bir hale gelmekte ve çok uzun süreler almaktadır.

Ayrıca oluşturulan ağ yapısı her zaman tatmin edici olmamaktadır. Bu grid yapısına alternatif olarak kulanılan başka bir yöntem, gömülü sınır yöntemi son zamanlarda oldukça tercih edilen bir mesh oluşturma yöntemidir ve yapı üzerine giydirilen gridlerin aksine oldukça kolay hazırlanmaktadır. Gömülü sınır yöntemi ile oluşturulan hesaplama ağının cismi sarması gerekmez. Akış içerisindeki duvar sınır şartları, çözüm algoritmasının değiştirilmesiyle elde edilir. Bu yöntemde kartezyen grid gibi çok basit bir hesaplama ağı oluşturulduktan sonra, cismin varlığı çözüm aşamasında yönetici denklemlerin modifikasyonlarıyla belirtilmektedir. Yapılandırılmış ya da yapılandırılmamış cisim üzerine giydirilen hesaplama ağlarına kıyasla, gömülü sınır yöntemi düşük boyutta hafızaya ihtiyaç duyar ve oluşturulması çok basittir. Bu tez çalışmasında incelenen akış analizlerindeki katı sınırlar gömülü sınır yöntemi kullanılarak belirtilmiştir.

Bu çalışmadaki hesaplamalar görüntü işleme birimi (GPU) üzerinde gerçekleştirilmiştir. Görüntü işleme birimleri son zamanlarda video ve oyun alanları dışında bilimsel çalışmalardaki hesaplamalı yöntemlerde de kullanılmaya başlanmıştır. Yüksek paralel hesaplama gücü sayesinde GPU'lar merkezi işleme birimlerine (CPU) kıyasla çok daha yüksek performansa sahip olabilmektedir. Bu çalışmadaki hesaplamalarda kullanılan GPU, NVIDIA tarafından bilimsel hesaplamalarda kullanılmak üzere üretilmiş Tesla C1060'dır. Bu GPU'da herbiri 8 skalar işlemci içeren 30 adet çoklu-işlemci bulunmaktadır. Üzerinde bulunan işlemcilerin çalışma frekansları 1.3 GHz'dir ve 102 GB/s veri taşıma hızında 4 GB boyutunda GDDR3 hafızaya sahiptir. Analizler için hazırlanmış kodlar, grafik işleme birimlerinin programlanması için NVIDIA tarafından geliştirilen CUDA (Compute Unified Device Architecture) ortamı kullanılarak, C tabanlı bir programlama dili ile oluşturulmuştur.

Bu tez çalışması kapsamında üç sıkıştırılamaz akış testi (ince levha üzerindeki akış, üst duvarı hareketli kavite içerisindeki akış ve dairesel silindir etrafındaki akış) gömülü sınır metodu ve kompakt sonlu farklar şeması kullanılarak incelenmiştir. Bu akış testleri için 3 boyutlu Navier-Stokes denklemleri daha önce bahsedilen sayısal yöntemler kullanılarak çözülmüştür. Birinci ve ikinci derece konumsal türevlerin hesabında altıncı mertebe kompakt, merkezi, sonlu farklar şeması kullanılmıştır. Katı sınırlar gömülü sınır yöntemi ile ifade edilmiştir. Poisson denkleminin çözümü için konjuge gradyen metodu kullanılmıştır. Hesaplamalar Poisson denkleminin çözümü haricinde tek hassasiyetle yapılmıştır. Poisson denkleminin çözümünde kullanılan konjuge gradyen metodunda çift hassasiyet kullanılmıştır. Uzaysal türev terimlerinin kompakt şema kullanılarak ayrıklaştırmasından elde edilen lineer denklem sistemi direk yöntemle çözülmüştür. Bunun için her analizin başında gerekli katsayı matrisi oluşturulmuş, bu matrisin tersi hesaplanarak depolanmış ve analiz boyunca her zaman adımında, hesaplanan sağ taraf setleri ile tekrar tekrar çarpılmıştır. Zaman integrasyonu için ikinci mertebe Adams-Bashfort metodu kullanılmıştır.

Yapılan analizlerde grafik işleme birimi, merkezi işleme birimine yardımcı işlemci olarak kullanılmıştır. Önişlem ve ardişlem olarak nitelendirilebilecek olan adımlar (hesaplama ağının oluşturulması, katsayı matrisinin oluşturulup tersinin hesaplanması ve depolanması, analiz sonucunda elde edilen verilerin uygun bir formatta kaydedilmesi ve bu sonuçların incelenmesi) merkezi işleme birimi tarafından, diğer tüm işlemler (kompakt şema ayrıklaştırılmasından elde edilen lineer denklem sisteminin sağ tarafının hesaplanması, analiz başlangıcında kaydedilen katsayı matrisinin tersi ile çarpılması, gömülü sınır metodu için gerekli işlemler,

Poisson denkleminin çözümü, zaman integrasyonu) grafik işleme birimi üzerinde gerçekleştirilmiştir.

Oluşturulan kodlarla simülasyonu gerçekleştirilen test çalışmalarından ilki olan ince levha üzerindeki akış analizinden elde edilen sonuçlar, Blasius sınır tabaka teoremi sonuçlarıyla kıyaslanmıştır. Bir diğer test çalışması olan tüm duvar uzunlukları eşit ve üst duvarı hareketli kavite içerisindeki akış analizlerinden elde edilen sonuçlar açık literatürde bulunan benzer analiz sonuçlarıyla kıyaslanmıştır. Son test çalışması dairesel silindir etrafındaki akış analizinin sonuçları da benzer şekilde açık literatürde bulunmuş deneysel ve sayısal test sonuçlarıyla kıyaslanmıştır. Bu tez çalışması kapsamında yapılan test çalışmalarında, teorik ve referans verilerle uyumlu sonuçlar elde edilmiştir.

1. INTRODUCTION

Computational Fluid Dynamics (CFD) has been used as a cost effective alternative to experimental methods by scientists and engineers in various industries. Performing time efficient, accurate simulations is the primary objective of CFD analyses. Since time used to perform simulations and reliability of the results are regarded as the most important issues, the problems concerning both the duration and accuracy have been subjects of the researchers.

Mesh generation is an essential and often the most time consuming part of CFD. Although automated and adaptive methods are possible, mesh generation is still largely manual. Over complex bodies automatically generated body-fitted meshes are often not possible or unsatisfactory and since generation of good mesh is problem dependent, expert knowledge is required throughout the process. Examples of body-fitted grid generation over complex bodies which take weeks even months can be easily given.

The immersed boundary method (IBM) is an alternative way to body-fitted meshes which in contrast to latter does not require the computational grid to fit the physical boundaries. Instead of conforming the grid to boundaries, in the IBM desired boundary conditions are imposed during the solution step by locally modified solution algorithm. As seen in Figure 1.1, in the IBM the mesh is constructed and the boundary condition is imposed in the solution step whereas in the body fitted meshes the mesh is generated over the specified geometry. The main advantages of the IBM are less memory requirement, CPU savings and ease of grid generation compared to body-fitted structured or unstructured grids [3].

The IBM was first introduced by Peskin [1] for simulation of blood flow in cardiovascular system. Since then there has been a great many work concerning the accuracy, stability and applicability of IBM and different ways to impose boundary conditions have been investigated. IB methods have been applied to a wide range of applications; compressible flows [5, 6], particulate flows [7, 8], micro-scale flows

[9], interaction with solid bodies [10, 11], multi-phase flows [12], conjugate heat transfer [13, 14], environmental flows [15], bio-fluids [16], etc. There will be a brief description of existing categories of IBM in the following sections. Detailed information about previous studies can be found in the articles [3] and [4]. In this study an application of an IBM for the solution of incompressible Navier-Stokes equations is presented.

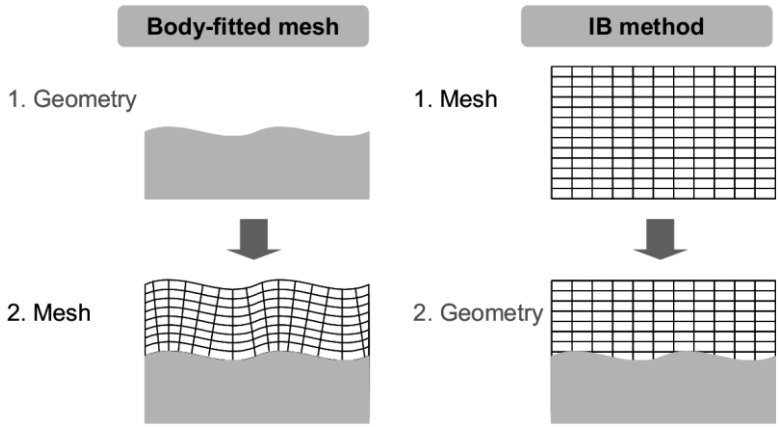

Figure 1.1: Preparation steps of the body-fitted mesh and IBM [2].

When dealing with transitional and turbulent fluid flows, numerical methods used for CFD must represent necessary time and length scales. Resolving required length scales depends on the numerical schemes. Accurate results with low order numerical methods require high grid resolutions which yield to high memory usage and long computation periods. Using high order spectral methods can overcome this drawback since reliable solutions can be achieved by relatively coarser grids, but computational cost of these schemes is too high and non-periodic boundaries need special treatments. Compact higher order finite difference schemes have a spectral like resolution ability but simpler structure. With the same order of accuracy compact schemes use smaller stencil and give better resolution compared to the finite difference schemes.

Although compact approximations have been previously used by other authors [17 - 21], the interest in compact schemes for turbulence and aeroacoustic simulations seems to have started from the pioneering work of Lele [22]. Lele developed compact approximations for the first and second derivatives and carried out extensive study and discussion of the resolution characteristics of the higher order compact schemes on a uniform grid. Since then, due to their benefits in solution accuracy and time requirement, high order compact schemes have received much attention and

comprehensive studies have been conducted concerning the resolution characteristics of these schemes [23 - 30]. Since derivatives are computed implicitly, compact finite difference schemes require derivative values as well as function values at the adjacent nodes. In the present study, a sixth order compact central finite difference scheme is used and will be discussed briefly in the following sections.

With the continuous increase in computational power of graphics processing units (GPU), recently they have started to be used as a scientific computing platform for computational sciences. Detailed and chronological reviews of studies conducted on GPUs are presented in [31] and [32]. In the field of CFD several promising implementations have been carried out [32 - 38]. The present study is conducted on a GPU in order to benefit from the immense parallel computational power of GPU.

In the context of this thesis study 3D incompressible Navier-Stokes equations are solved for various test cases; flow over a flat plate, lid driven cavity flow and flow around a circular cylinder. These simulations are carried out with previously mentioned numerical techniques; high order compact scheme and immersed boundary method, on a graphics processing unit because of their high computational power. Calculations are conducted on a Tesla C1060, one of NVIDIA's scientific computing GPUs. Compute Unified Device Architecture (CUDA) toolkit which is a complete software development solution for programming CUDA-enabled GPUs is utilized as the code design medium.

2. GOVERNING EQUATIONS

Simulations of the test cases, flow over flat plate, lid driven cavity flow and flow around a circular cylinder are conducted using incompressible Navier-Stokes equations. These equations are:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{2.1}$$

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right) \tag{2.2}$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right) \tag{2.3}$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right) \tag{2.4}$$

Here u, v and w are cartesian velocity components in x, y, z directions and ρ, p and μ represent density, static pressure and molecular viscosity respectively.

The periodic boundary conditions are applied in y and z directions for the simulations. If $i=1$ and $i=n$ are the first and last nodes along any periodic direction, then the formulation used for any flow variable denoted by f on the periodic boundaries are:

$$f_0=f_n \text{ and } f_{-1}=f_{n-1} \tag{2.5}$$

$$f_{n+1}=f_1 \text{ and } f_{n+2}=f_2 \tag{2.6}$$

On the left boundary along the x direction (inlet), the Drichlet boundary conditions are applied for the velocity components and Neumann boundary condition is applied for the pressure:

$$u=U_\infty, \quad v=0, \quad w=0, \quad \partial p/\partial x=0. \tag{2.7}$$

On the right boundary along the x direction (outlet), the Neumann boundary conditions are applied for velocity components and Drichlet boundary condition is applied for the pressure:

$$\partial u/\partial x = 0, \quad \partial v/\partial x = 0, \quad \partial w/\partial x = 0, \quad p = 0. \tag{2.8}$$

3. NUMERICAL METHODS

3.1 Temporal Discretization

In this study, the second order Adams-Bashfort scheme [39] is applied for the time integration. This is an explicit, multi-point scheme which uses the results of two previous time steps $n-1$ and n to compute the time step $n+1$. Since it uses current and prior to current time steps to get the next one, it cannot start from the initial conditions. Thus another method is needed to get the computation started. The formulation for the second order Adams-Bashfort method is given below.

$$Q^{n+1} = Q^n + \Delta t\left[\frac{3}{2}L(E^n) - \frac{1}{2}L(E^{n-1})\right] \tag{3.1}$$

Here *L(E)* is the discrete representation of all terms other than the term including time derivative.

3.2 Immersed Boundary Method

As mentioned before, rather than generating the computational grid over body, in IBM the solution algorithm is modified to capture desired boundary conditions. How the boundary conditions are imposed distinguishes IB methods from one another. The governing equations for incompressible viscous flow, the Navier-Stokes equations, follow as,

$$\rho\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) + \nabla p - \mu \Delta u = 0 \tag{3.2}$$

$$\nabla \cdot u = 0 \text{ in } \Omega_f \tag{3.3}$$

$$u = u_\Gamma \text{ on } \Gamma_b \tag{3.4}$$

where $u(x,t)$, $p(x,t)$ and the coefficients ρ , μ are fluid velocity, pressure, density and viscousity respectively. The solid body occupies the domain Ω_b, with boundary denoted by Γ_b , and the surrounding fluid domain denoted by Ω_f .

In an IBM, equation (3.2) is discretized on a cartesian grid without a body and then this discretized equation is modified to impose the boundary conditions. This modification yields to a forcing function that produces the effect of the boundary.

Including the forcing function, denoted by f, yields to equations,

$$\rho\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) + \nabla p - \mu \Delta u = \mathrm{f} \tag{3.5}$$

$$\nabla \cdot u = 0 \text{ in } \Omega_f \tag{3.6}$$

$$u = u_\Gamma \text{ on } \Gamma_b \tag{3.7}$$

The way to introduce this forcing function divides IB methods in two groups [3], continuous forcing and discrete (or direct) forcing. In continuous forcing approach forcing function, f, is included into the momentum equation and then applied to the entire domain [40]. This technique was first used in fluid simulations with elastic boundaries [1, 41]. Then, it was extended to rigid boundaries by several approaches [42 - 45]. In this method, appropriate forcing function should be chosen to represent the external body force.

In direct (discrete) forcing approach, the momentum equation is discretized without the body and then the presence of the body is introduced by adjusting the discretization in cells near the immersed body. Velocity boundary conditions are imposed at the immersed body without inserting or computing any forcing term. In this approach, usually an interpolation scheme is used to compute for the node values near the IB. Different approaches are presented in references [46 - 52] for computations of boundary values, for example, determining a forcing term by the difference between the interpolated velocities in the boundary points and the desired (physical) boundary velocities, i.e. mirroring the velocity field across the immersed boundary [46, 47], reconstructing the solution in vicinity of boundary nodes by linear interpolation of the values along the line normal to the body [48, 50], ghost cell approach, as opposed to previous approach, interpolating values inside the solid body with an interpolation scheme that implicitly incorporates the boundary condition on the IB [51, 52].

There is also another IBM approach, called cut-cell approach, in which the Cartesian cells are truncated to create new cells in order to form the immersed body [53, 54]. This approach does not require any forcing function.

3.2.1 A direct forcing approach

In the present study the direct forcing method, explained in detail in [38], is used. Values at the cells in vicinity of the immersed body are computed by an interpolation scheme.

Incompressible Navier-Stokes equations with direct forcing vector F to impose proper boundary condition on the surface of a solid body are given below.

$$\nabla \cdot V = 0 \tag{3.8}$$

$$\frac{\partial V}{\partial t} + V \cdot \nabla V = -\frac{1}{\rho}\nabla p + \vartheta \nabla^2 \mathrm{V} + \mathrm{F} \tag{3.9}$$

Representing the convection and diffusion terms by A, equation (3.10), the time advancement of the momentum equation (3.9) with the use of previously described second order Adams-Bashfort method may be shown as:

$$A = -(V \cdot \nabla V) + \vartheta \nabla^2 \mathrm{V} \tag{3.10}$$

$$\frac{V^* - V^n}{\Delta t} = \frac{3}{2}A^n - \frac{1}{2}A^{n-1} - \frac{1}{\rho}\nabla p^n + F^{n+1} \tag{3.11}$$

$$\frac{V^{**} - V^*}{\Delta t} = \frac{1}{\rho}\nabla p^n \tag{3.12}$$

$$\frac{V^{n+1} - V^{**}}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1} \tag{3.13}$$

The forcing term related to IBM can be given by:

$$F^{n+1} = \varepsilon\left(-\frac{3}{2}A^n + \frac{1}{2}A^{n-1} + \frac{1}{\rho}\nabla p^n + \frac{V_b^{n+1} - V^n}{\Delta t}\right) \tag{3.14}$$

Where ε is 1 in solid body region and 0 everywhere else. Inserting above term into equation (3.11), satisfies the velocity boundary condition in the region prescribed by ε.

Normally in a standard fractional step method, the incompressibility condition, $\nabla \cdot V^{n+1} = 0$, may be satisfied with the solution of a Poisson equation in the form of:

$$\nabla \cdot \nabla p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot V^{**} \tag{3.15}$$

When IBM is included in the computational method, equation (3.15) transforms into equation (3.16),

$$\nabla \cdot \nabla p^{n+1} = \frac{\rho}{\Delta t} \nabla \cdot [(1-\varepsilon)V^{**}] \tag{3.16}$$

In the solid body region, $\varepsilon = 1$, above equation gives the Laplace equation, anywhere else, $\varepsilon = 0$, the standart Poisson equation is obtained. This Poisson equation is solved with the use of conjugate gradient method.

In the flow over flat plate and flow around circular cylinder test cases ε is chosen to be 1 inside the solid body, i.e. the grid points that are on or inside the flat plate and circular cylinder. In the lid driven cavity flow, since the solid boundaries are represented with IBM, the computational grid extends outside the cavity and in this case ε is 1 outside the solid boundary, i.e. the grid points that are on or outside cavity.

3.2.2 Interpolation scheme

In the scope of this work, a direct forcing IBM is incorporated into the solution of the incompressible Navier-Stokes equations. In this approach velocity of the immersed boundary, V_b, needs to be imposed at every fractional step. When immersed boundary coincides with cartesian grid, the Drichlet boundary condition is enforced at this point, i.e. the velocity of the immersed boundary is imposed. However if the body does not coincide with the grid, then an appropriate interpolation scheme is required to compute values at the cells in vicinity of the immersed body. In the present study a second order accurate interpolation scheme which was suggested in [47] and described in detail in [48] is used. In this approach the imposition of the velocity is done on the grid points nearest to the solid body. It is proposed to compute $\overrightarrow{V_1}$ by interpolating along the line normal to the solid boundary as seen in Figure 3.1. The value of the virtual point $\overrightarrow{V_2}$, which is not located on a grid point in this case, is interpolated from the surrounding points.

In this method, the optimal location of the virtual point that has a compact stencil without involving other points in vicinity of the boundary should be determined [48]. In order to do that the author suggests tagging the points in vicinity of the body with a -1 flag.

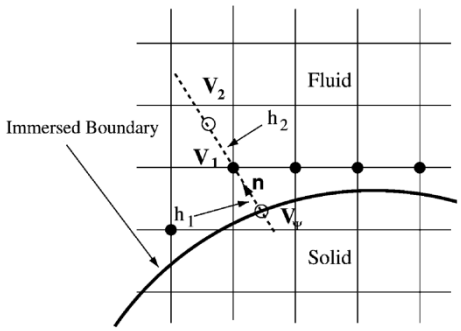

Figure 3.1: Treatment of the interface cells [48].

The interpolation procedure involves following steps [48]:

- Find the virtual point on the line normal to the boundary in a way that $h_1 = h_2$.
- Check if any of the grid points 1 - 4, surrounding the virtual point has a -1 flag.
- If none of these points have a -1 flag, then the interpolation stencil is the one shown in the Figure 3.2a and using this stencil any flow variable ϕ at point ib can be found by:

$$\phi_{ib} = \frac{h_2}{h}\phi_{\psi} + \frac{h_1}{h}\phi_{virtual} \quad \textbf{(3.17)}$$

  and

$$\phi_{virtual} = \sum_{1}^{4} \alpha_i \phi_i \quad \textbf{(3.18)}$$

  Where $h = h_1 + h_2$ and α_i are the coefficients of a standart bilinear interpolation involving points 1 - 4.
- If any of points 1 – 4 coincide with the boundary point, ib, but none of the remaining ones are tagged with a -1 flag, then the interpolation stencil is the one shown in Figure 3.2b. Again, in this case despite the reduced stencil size the value of ϕ_{ib} is obtained from equations (3.17) and (3.18).
- If any other point (except for ib) with a -1 flag is involved in the stencil, then the virtual point is relocated in a way that $h_2 > h_1$. This is done iteratively by increasing the initial distance $h_2 = h_1$ by $h_1/4$ intervals until one of the above scenarios is ‘‘true’’. Referring to Figure 3.2c the interpolation is performed using equations (3.17) and (3.18).

This procedure can be easily extended to three-dimension by replacing the bilinear interpolation with a trilinear.

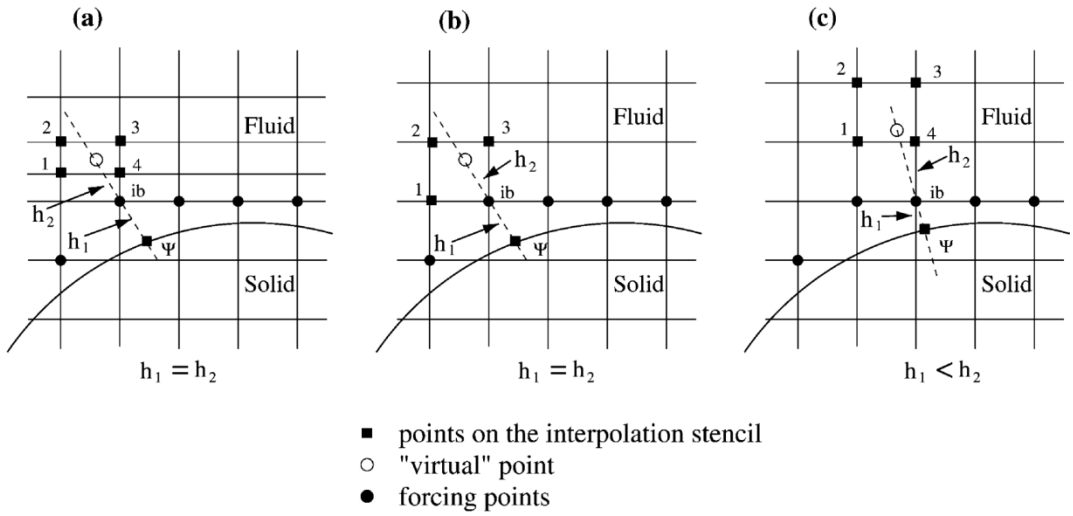

Figure 3.2: Schematic of the interpolation stencil in two dimensions. (a)–(c) Possible choices depending on the boundary topology and local grid size [48].

Since linearization of velocity component in the vicinity of the immersed boundary reduces the momentum equation normal to the boundary to $dp/dn = 0$, on the immersed boundary no imposition is done for the pressure boundary condition [46].

3.3 Compact Finite Difference Schemes

Throughout the present study, sixth order central compact finite difference scheme is used for the spatial discretization of the governing equations. Detailed information about compact schemes and implemented Fourier analyses to investigate the resolution characteristics are presented in [22] and [38]. The compact approximations for first and second derivatives on a uniform grid, given in [22], will be discussed here.

For the first derivative, the generalized central compact finite difference schemes on a uniformly spaced mesh are derived from the form:

$$\beta f'_{i-2} + \alpha f'_{i-1} + f'_i + \alpha f'_{i+1} + \beta f'_{i+2} = c\frac{f_{i+3} - f_{i-3}}{6h} + b\frac{f_{i+2} - f_{i-2}}{4h} + a\frac{f_{i+1} - f_{i-1}}{2h} \quad \textbf{(3.19)}$$

Here, f is the function value of node indexed by i, f' is the derivative of the function, h is the grid spacing and $a, b, c, \alpha\ and\ \beta$ are coefficients. The constraints for these

coefficients, attained by matching Taylor series coefficients of various orders, follow as:

$$a + b + c = 1 + 2\alpha + 2\beta \text{ (second order)} \quad \textbf{(3.20)}$$

$$a + 2^2 b + 3^2 c = 2\frac{3!}{2!}(\alpha + 2^2\beta) \text{ (fourth order)} \quad \textbf{(3.21)}$$

$$a + 2^4 b + 3^4 c = 2\frac{5!}{4!}(\alpha + 2^4\beta) \text{ (sixth order)} \quad \textbf{(3.22)}$$

$$a + 2^6 b + 3^6 c = 2\frac{7!}{6!}(\alpha + 2^6\beta) \text{ (eighth order)} \quad \textbf{(3.23)}$$

$$a + 2^8 b + 3^8 c = 2\frac{9!}{8!}(\alpha + 2^8\beta) \text{ (tenth order)} \quad \textbf{(3.24)}$$

For a periodic dependent variable in any direction, if β is chosen to be zero then the linear system of equations generated by writing the equation (3.19) for each node will be cyclic tridiagonal, otherwise it will be cyclic pentadiagonal. If the dependent variable is not periodic then special treatment will be required for the values of near boundary nodes. The order of scheme is reached by the imposition of the constrainments defined by the equations (3.20) – (3.24) to the general form given by equation (3.19). For instance, a fourth order accurate scheme can be reached by satisfying the general relation between the equations (3.19), (3.20) and (3.21). If the equation (3.22) is also imposed, then a sixth order accurate scheme is obtained. Extensions to eighth or tenth order may be achieved by the same procedure.

For a tridiagonal scheme generated by $\beta = 0$, if another choice is made for $c = 0$, then a fourth order accurate one-parameter (α) scheme is obtained. The constraints for these schemes follow as:

$$\beta = 0,\ a = \frac{2}{3}(\alpha + 2),\ b = \frac{1}{3}(4\alpha - 1),\ c = 0 \quad \textbf{(3.25)}$$

If α is chosen to zero, then classic fourth order central difference scheme is achieved. For $\alpha = 1/3$ the scheme is sixth order accurate and its coefficients are:

$$\alpha = \frac{1}{3},\ \ \beta = 0,\ \ \alpha = \frac{14}{9},\ \ b = \frac{1}{9},\ \ c = 0 \quad \textbf{(3.26)}$$

In this study sixth order compact central differencing scheme is used. The derivations of pentadiagonal schemes $(\beta \neq 0)$ and eighth-order tridiagonal scheme $(\beta = 0)$ can be found in [22].

Approximations of the second derivative can be done in a way very similar to the first derivative. The starting point for the second derivative follows as:

$$\beta f''_{i-2} + \alpha f''_{i-1} + f''_i + \alpha f''_{i+1} + \beta f''_{i+2} = c\frac{f_{i+3} - 2f_i + f_{i-3}}{9h^2} + b\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} + a\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \quad \textbf{(3.27)}$$

Again, the constraints for the coefficients $(a, b, c, \alpha\ and\ \beta)$, attained by matching Taylor series coefficients of various orders, follow as:

$$a + b + c = 1 + 2\alpha + 2\beta \text{ (second order)} \quad \textbf{(3.28)}$$

$$a + 2^2 b + 3^2 c = \frac{4!}{2!}(\alpha + 2^2\beta) \text{ (fourth order)} \quad \textbf{(3.29)}$$

$$a + 2^4 b + 3^4 c = \frac{6!}{4!}(\alpha + 2^4\beta) \text{ (sixth order)} \quad \textbf{(3.30)}$$

$$a + 2^6 b + 3^6 c = \frac{8!}{6!}(\alpha + 2^6\beta) \text{ (eighth order)} \quad \textbf{(3.31)}$$

$$a + 2^8 b + 3^8 c = \frac{10!}{8!}(\alpha + 2^8\beta) \text{ (tenth order)} \quad \textbf{(3.32)}$$

Similar to first derivative the order of the scheme is determined by meeting these constraints. For a periodic dependent variable in any direction, if α and β is chosen to be zero then the linear system of equations generated by writing the equation (3.27) for each node will be cyclic tridiagonal, otherwise it will be cyclic pentadiagonal. If the dependent variable is not periodic then special treatment will be required for the values of near boundary nodes.

For a tridiagonal scheme generated by $\beta = 0$, if $c = 0$, then a fourth order accurate one-parameter (α) scheme is obtained. The constraints are:

$$\beta = 0, \quad c = 0, \quad a = \frac{4}{3}(1 - \alpha), \quad b = \frac{1}{3}(-1 + 10\alpha) \quad \textbf{(3.33)}$$

If α is chosen to zero, then classic fourth order central difference scheme is achieved. For $\alpha = 2/11$ the scheme is sixth order accurate and its coefficients are:

$$\alpha = \frac{2}{11}, \beta = 0, \alpha = \frac{12}{11}, b = \frac{3}{11}, c = 0 \quad \textbf{(3.34)}$$

In this study, for the second order spatial derivatives in incompressible Navier-Stokes equations sixth order compact central differencing scheme is used. The derivations of pentadiagonal schemes ($\beta \neq 0$) can be found in [22].

For the investigations of resolution characteristics of finite differencing approximations, performed Fourier analyses by [38] is presented in Figure 3.3 and Figure 3.4. It is clearly seen from the comparison as the order increases the scheme becomes more compatible with the exact differentiation.

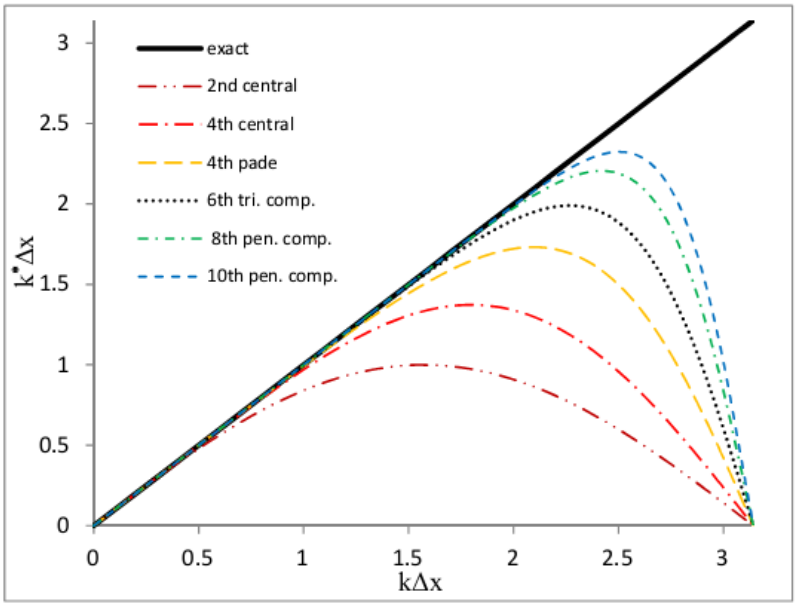

Figure 3.3: Resolution characteristics of various central schemes for the first derivative [38].

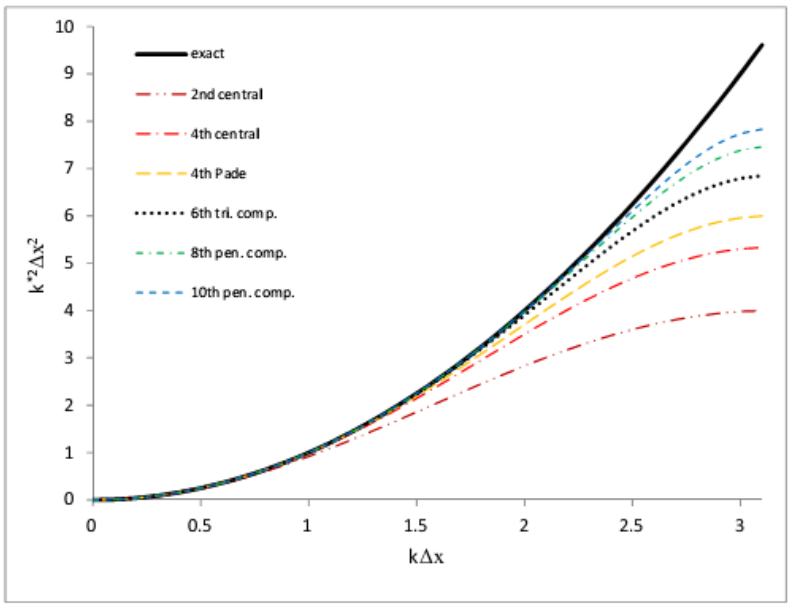

Figure 3.4: Resolution characteristics of various central schemes for the second derivative [38].

The second and fourth order schemes display large inconsistency from exact solution. Although the eighth and tenth order schemes give better resolution characteristics, they require larger bandwidth and larger stencils which produce larger coefficient matrices. Sixth order compact scheme is a compromise between opposing factors [38]. The first and second derivatives for the sixth order central compact scheme may be seen below.

$$\frac{1}{3}f'_{i-1} + f'_i + \frac{1}{3}f'_{i+1} = \frac{1}{9}\left(\frac{f_{i+2} - f_{i-2}}{4h}\right) + \frac{14}{9}\left(\frac{f_{i+1} - f_{i-1}}{2h}\right) \tag{3.35}$$

$$\frac{2}{11}f''_{i-1} + f''_i + \frac{2}{11}f''_{i+1} = \frac{3}{11}\left(\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2}\right) + \frac{12}{11}\left(\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}\right) \tag{3.36}$$

Since the stencil of the central compact scheme extend out of the boundary, for the non-periodic boundaries and the points near these boundaries third order one sided and fourth order central compact scheme are applied respectively. These schemes follow as:

1st derivative, i=1 (3rd order) $$f'_1 + 2f'_2 = \frac{1}{2h}(-5f_1 + 4f_2 + f_3) \tag{3.37}$$

1st derivative, i=2 (4th order) $$\frac{1}{4}f'_1 + f'_2 + \frac{1}{4}f'_3 = \frac{3}{4h}(f_3 - f_1) \tag{3.38}$$

2nd derivative, i=1 (3rd order) $$f''_1 + 11f''_2 = \frac{1}{h^2}(13f_1 - 27f_2 + 15f_3 - f_4) \tag{3.39}$$

2nd derivative, i=2 (4th order) $$\frac{1}{10}f''_1 + f''_2 + \frac{1}{10}f''_3 = \frac{6}{5h^2}(f_3 - 2f_2 + f_1) \tag{3.40}$$

3.4 Conjugate Gradient Method

To solve the Poisson equation which yields to a very large sparse system that cannot be solved by direct methods, conjugate gradient method is used. This is an iterative method used for the solution of symmetric positive definite systems [54]. Solution algorithm of conjugate gradient method for the solution of system $Ax = b$ may be seen in Figure 3.5.

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$	$\boldsymbol{x}_0$ is an initial approximate solution vector and $\boldsymbol{r}_0$ is a residual.
$\boldsymbol{l}_0 = \boldsymbol{r}_0$	$\boldsymbol{l}$ is an auxilary variable.
$k = 0$	k is an iteration step indicator.
repeat	
$\boldsymbol{q}_k = \boldsymbol{A}\boldsymbol{l}_k$	$\boldsymbol{q}_k$ is k_{th} iteration residual.
$\alpha_k = \dfrac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{l}_k^T \boldsymbol{q}_k}$	α_k is a correction factor.
$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{l}_{\boldsymbol{k}}$	$\boldsymbol{x}_{k+1}$ is the solution at $(k+1)_{th}$ iteration.
$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k$	$\boldsymbol{r}_{k+1}$ is the residual at $(k+1)_{th}$ iteration. If $\boldsymbol{r}_{k+1}$ is small enough, then exit loop and the solution is $\boldsymbol{x}_{k+1}$.
$\beta_k = \dfrac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k}$	β_k is a correction factor.
$\boldsymbol{l}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{l}_{\boldsymbol{k}}$	$\boldsymbol{l}_{k+1}$ is the auxilary variable at $(k+1)_{th}$ iteration.
$k = k + 1$	
end	

Figure 3.5: Solution algorithm for conjugate gradient method.

4. GENERAL PURPOSE COMPUTING ON GRAPHICS PROCESSING UNIT (GPGPU)

Prior to development of multi-core processor technology of central processing units (CPU), sequential codes were implemented in the applications running in the computers. Chips including multiprocessor units allowed benefiting increased performance via parallelization of applications. At the beginning of the 2000s, the rapid advancement regarding CPUs' performance and multi-core technology slowed due to issues like energy consumption and heat generation of processors. In contrast to CPUs having conventional multi-cores, graphics processing units (GPU) have many cores with lesser capability. For instance the GPU that has been used in this study, NVIDIA Tesla C1060, has 240 scalar processors at 1.3 GHz with a bandwidth of 102 GB/s, capable of 933 GFLOPs/s with thousands synchronous working threads. This particular GPU is a relatively old one. Peak performance of one of the newer GPUs, NVIDIA Tesla K40, has reached to 4290 and 1430 GFLOPS for single and double precision floating-point operations respectively with 2880 scalar processors at 3.0 GHz.

Because of their immense parallel computational power, and recent advancements on floating point operations, GPUs can be more efficient compared to CPUs. Starting from the beginning of the 2000s GPUs have begun to be used as a scientific computing platform priorly handled by CPUs. GPUs have been used by many researchers from different fields; life sciences [56, 57], computational electrodynamics [58], data mining [59, 60], medical imaging [61, 62], molecular dynamics [63, 64] and computational fluid dynamics. Early implementations of CFD simulations on GPUs were generally appeared on gaming applications. One of the first CFD implementations, a smoke simulation by solving 2D Navier-Stokes equations, is done by M. Harris [65]. In the beginning, the CFD applications on GPUs mainly focused on performance comparison against CPU codes. Several works including performance comparisons between GPU and CPU can be seen in [66 - 69]. Solving 2D/3D compressible Euler equations, compared to CPU 25x and 14x speed-

ups are attained in [66] for two and three dimensional simulations respectively. Another comparison between CPU and GPU is presented in [67, 68] with Euler equations and 29x and 16x speed-ups are attained for two and three dimensional simulations respectively. A hypersonic flow simulation is conducted on GPU in [69], and 40x and 20x speed-ups are obtained for the simple and complex geometries respectively. The work done by Tutkun, [38], shows 20x speed-up for the solution of sparse linear systems on GPU against a CPU code.

4.1 GPU Architecture and Code Design

The GPU used in this study, NVIDIA Tesla C1060, may be seen in Figure 4.1 and general specifications of this GPU are presented in Table 4.1.

Figure 4.1: A Tesla C1060 GPU [38].

Table 4.1: General specifications of Tesla C1060

Number of processor cores	240
Processor core clock	1.296 GHz
Global Memory (GDDR3)	4 GB
Internal bandwidth	102 GB/s
Peak performance (single precision)	933 GFLOPs/s
Peak performance (double precision)	78 GFLOPs/s
Typical power requirement	160 W
Peak power requirement	200 W

In this study NVIDIA's C programming language based CUDA toolkit is used. GPU, designated as device, is used as a coprocessor to the main CPU, designated as host. Tasks are carried out by many threads on the device. Threads are arranged in blocks and blocks are arranged in grids. A general representation of this structure can be seen in Figure 4.2.

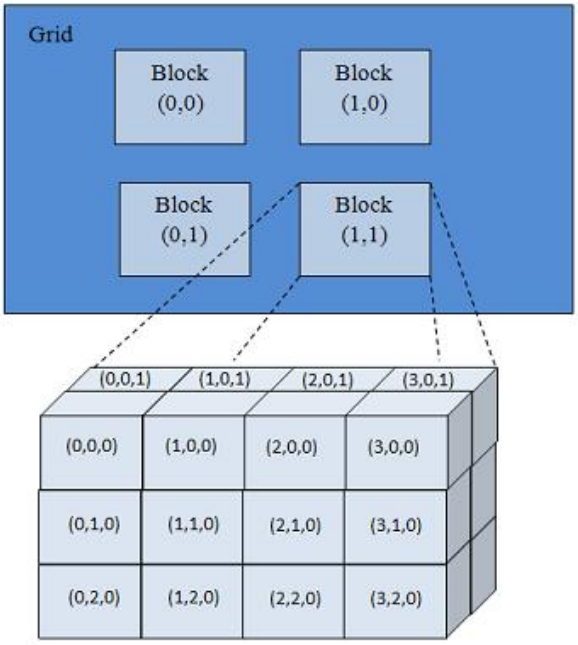

Figure 4.2: A representation of CUDA structure [38].

In this study computational structure on GPU, explained in [70], contains 1D grid which is composed of 1D block ordered along y axis. The blocks contain threads ordered along x direction. As a result the 3D flow domain is represented by a 2D computation structure with the help of indexing. Each node in the computational domain is assigned to a thread in order to perform all the calculations on any node by the same thread. The computation structure of threads, blocks and grid for a 3D flow domain with N grid points in each direction is shown in Figure 4.3. The CUDA computation structure has N^2 blocks along the Y-axis and N^3 threads in total each representing a grid point in the flow domain. The physical coordinates x and y match with the CUDA coordinates X and Y, and N block slices represents the physical z coordinate, i.e., the first physical xy-plane is represented by the first N blocks (from the block "0" to the block "N-1"), and the second physical xy-plane is represented by the second N blocks (from the block "N" to the block "2N-1"), and so on. This approach is chosen for the general code structure because Tesla C1060 GPU allows maximum 512 threads in a block and 64 threads in Z-axis of the block [70].

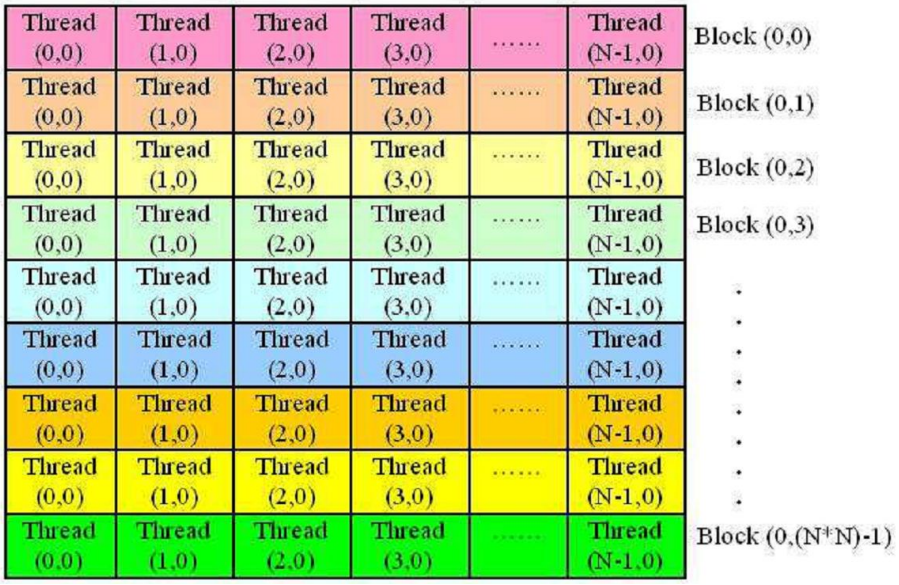

Figure 4.3: Representation of CUDA grid [38].

A general computation grid for a 3D flow domain generated from the 2D CUDA grid may be seen in Figure 4.4. The built-in CUDA variables “blockDim.x”, “blockIdx.x” and “Thread.idx” represent the number of threads along the X-axis in the block, the place of the block in the X-axis within the grid and the place of the thread in the X-axis within the block respectively. Similarly, “blockIdx.y” shows the place of the block in the Y-axis within the grid and “Thread.idy” and “Thread.idz” show the global places of the threads on the Y- and Z-axis respectively.

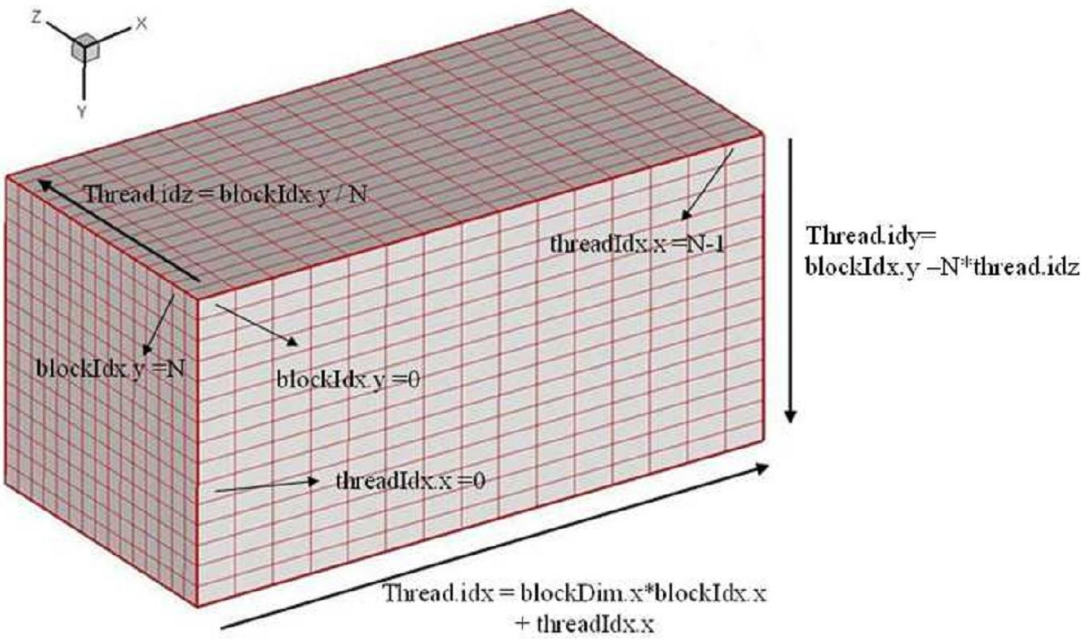

Figure 4.4: Computation structure for a 3D flow domain [38].

5. APPLICATIONS

3D incompressible Navier-Stokes equations are solved for three basic test cases; flow over a flat plate, lid driven cavity flow and flow around a circular cylinder. For the first and second spatial derivatives a sixth order central compact finite difference scheme is used. The solid boundaries are represented by immersed boundary method. Conjugate gradient method is used for the solution of the Poisson equation. Calculations are carried out with using single precision floating point format except for the solution of Poisson equation. For the iterations of conjugate gradient method double precision floating point format is employed. The linear system of equations of compact schemes is solved with direct method. The coefficient matrix is generated and its inverse is calculated and stored at the beginning of each simulation and then repeatedly multiplied with multiple right hand side sets at every time step. The second order Adams-Bashfort method is used for the time integration.

5.1 Flow Over A Flat Plate

The first test case considered is a laminar flow over a flat plate. At the inlet, the Drichlet boundary conditions are applied for the velocity components and Neuman boundary condition is applied for the pressure; $u = U_\infty$, $v = 0$, $w = 0$ and $\partial p/\partial x = 0$. At the outlet, for the velocity components the Neuman boundary conditions are applied and for the pressure the Drichlet boundary condition is applied; $\partial u/\partial x = 0$, $\partial v/\partial x = 0$, $\partial w/\partial x = 0$ and $p = 0$. The periodic boundary conditions are applied in y and z directions for the simulation. The grid spacings along the x- and y-axis are $\Delta x = 0.015\ m$, $\Delta y = 0.01\ m$.

The velocity field obtained from the simulation is presented in Figure 5.1. The boundary layer thickness along the flat plate is compared to the Blasius boundary layer theorem results. Figure 5.2 shows the comparison between present results and Blasius results along the flat plate between the Reynolds numbers $Re_L = 5000$ and

$Re_L = 10000$ calculated by the distance from the leading edge of the plate. The results are in good agreement.

Figure 5.1: Contours of velocity [m/s].

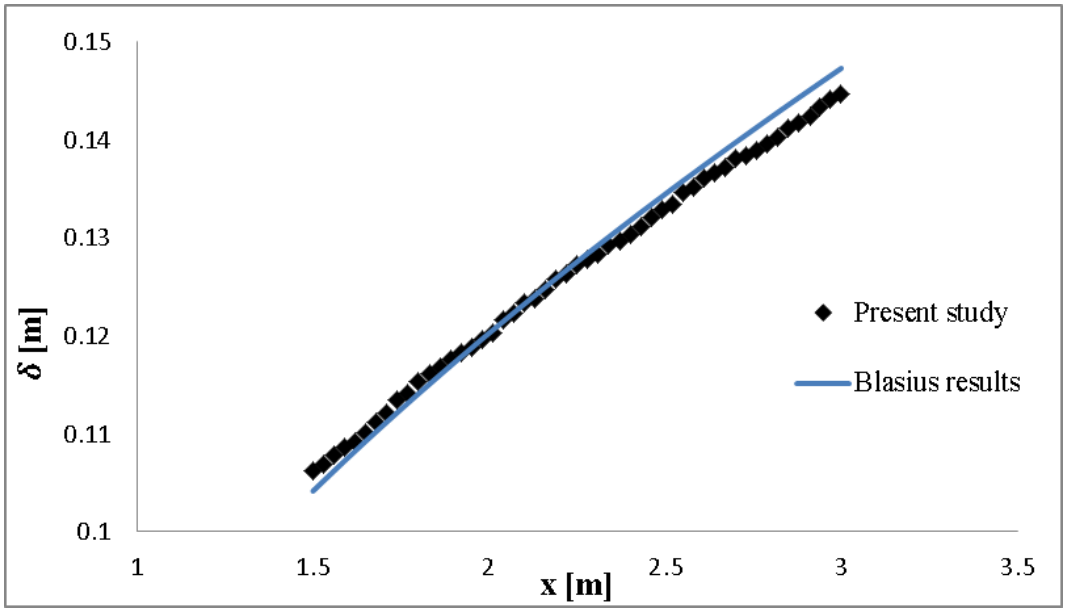

Figure 5.2: Boundary layer thickness along the flat plate.

The profiles of x and y components of the velocity inside the boundary layer at various Reynolds number calculated by the distance from the leading edge of the plate are presented in Figure 5.3. The results are compared with Blasius solutions. The results obtained by the present study are in good agreement with theoretical data.

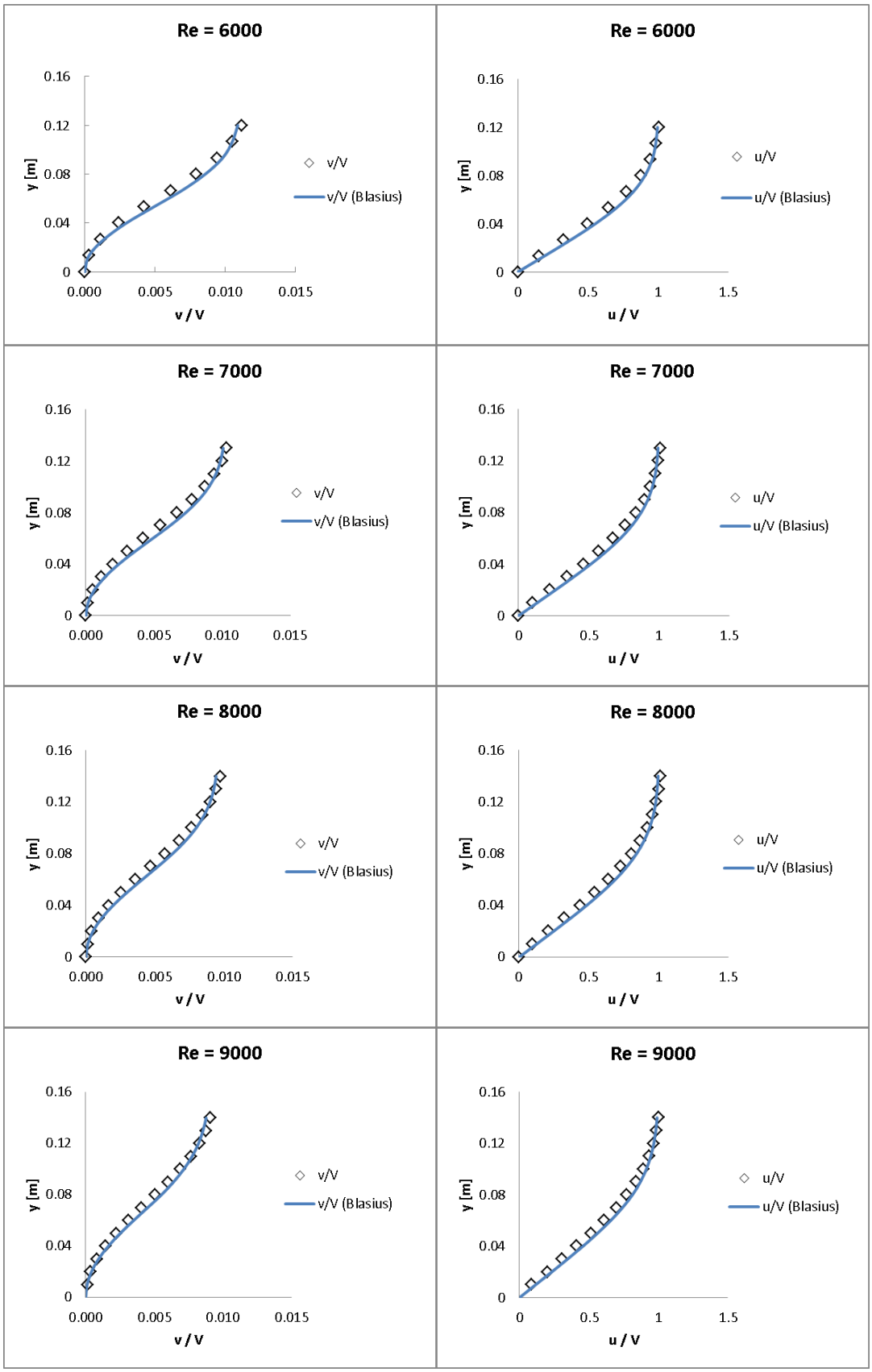

Figure 5.3: Profiles of x and y components of velocity inside boundary layer at various Reynolds numbers.

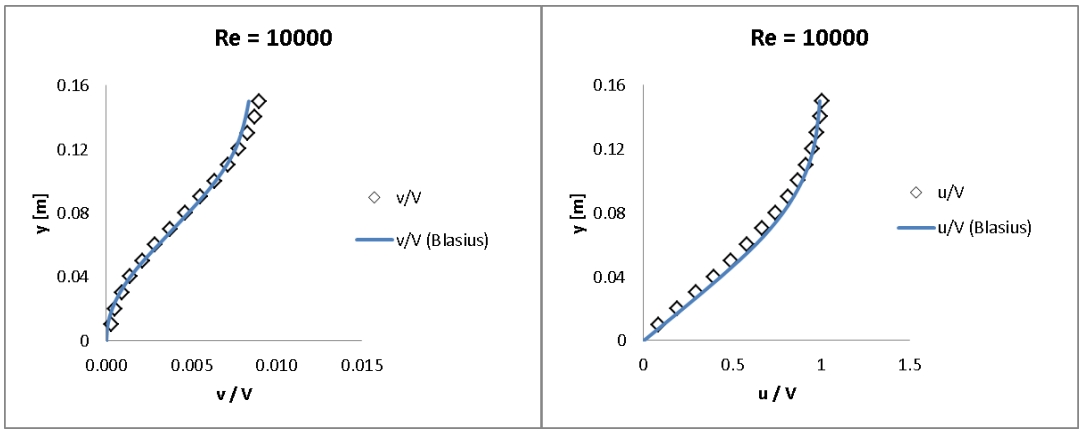

Figure 5.3 (continued): Profiles of x and y components of velocity inside boundary layer at various Reynolds numbers.

5.2 Lid Driven Cavity Flow

The second test case is the laminar incompressible flow in a square cavity of which top wall moves with a uniform velocity. This problem is widely used for testing and evaluating numerical techniques because of its simple geometry and complex flow structure. The schematic representation of the problem is shown in Figure 5.4.

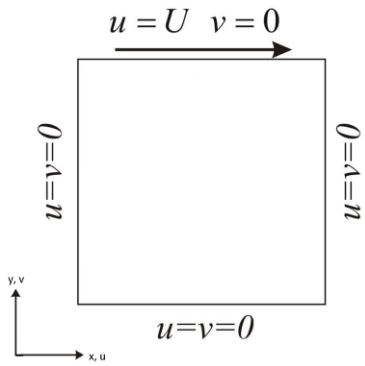

Figure 5.4: Schematic diagram of lid driven cavity flow.

The simulations for incompressible, viscous, laminar lid driven cavity flow is conducted for three different Reynolds number; $\mathrm{Re} = 100$, $\mathrm{Re} = 400$, and $\mathrm{Re} = 1000$. For each simulation grid independency test is carried out.

For Reynolds number $\mathrm{Re} = 100$, results from conducted simulations for different grid numbers are shown in Figure 5.5 and Figure 5.6. It is apparent that the flow becomes independent from the grid with the use of 41 X 41 grid points inside the cavity. The results are compared with the solutions obtained in [71]. Contours of velocity components inside the cavity are shown in Figure 5.7.

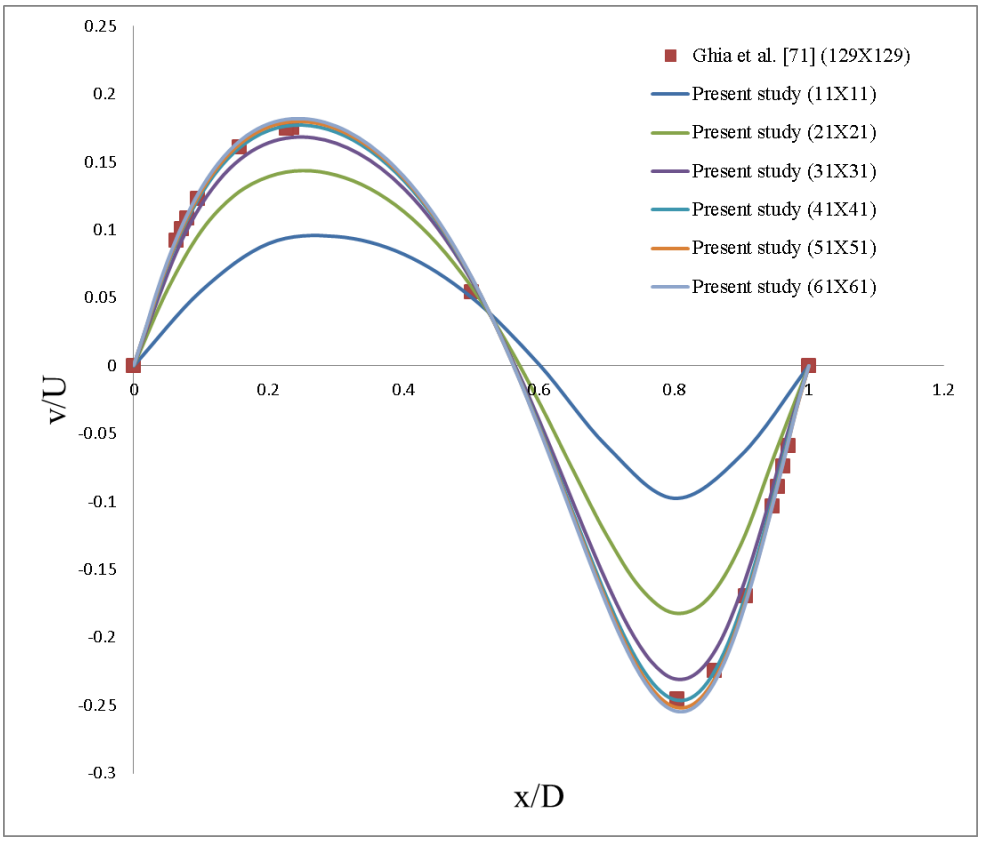

Figure 5.5: Results for v-velocity along horizontal line through geometric center of cavity for Re=100.

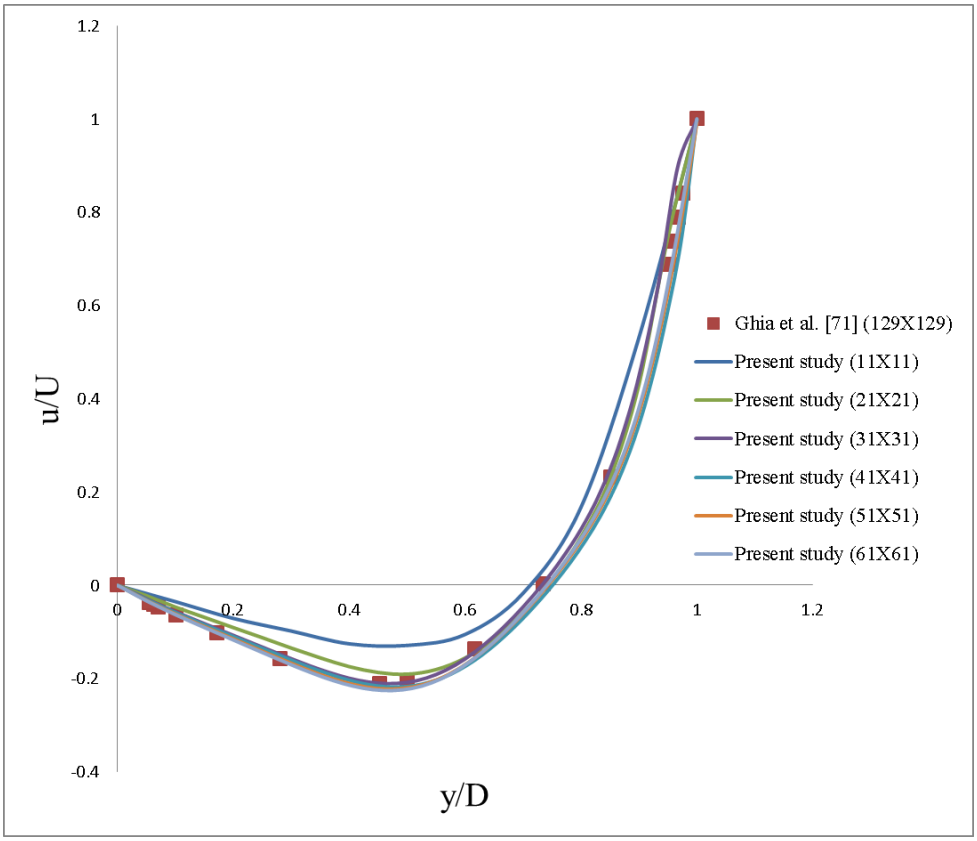

Figure 5.6: Results for u-velocity along vertical line through geometric center of cavity for Re=100.

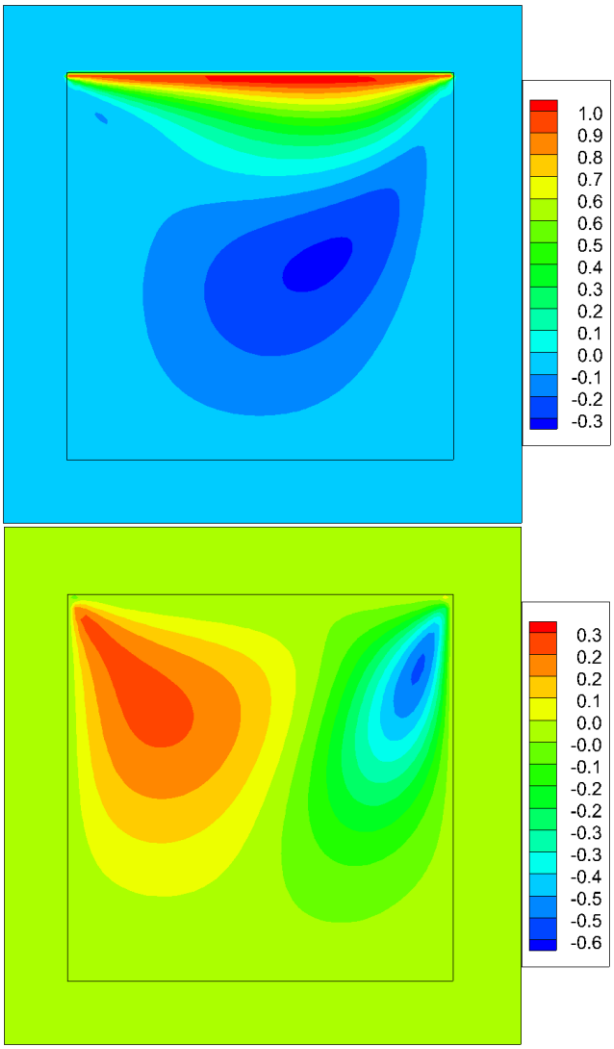

Figure 5.7: Velocity contours inside the cavity for Re=100; top: u-velocity [m/s]; bottom: v-velocity [m/s].

For Reynolds number Re = 400, conducted simulations for different grid numbers are shown in Figure 5.8 and Figure 5.9. As seen in the figures as Reynolds number increases, accurate results cannot be obtained from coarse grid. For Re = 400, the mesh generated with 81 X 81 grid points inside the cavity is adequate. The results are in good agreement with the solutions obtained in [71].

For Reynolds number Re = 1000, with similar approach, simulations are repeated for different grid numbers. The results are shown in Figure 5.10 and Figure 5.11. It is apparent the flow becomes independent from the grid with the use of 121 X 121 grid points inside the cavity. The obtained results are compared with the solution of [71]. Results for v-velocity along horizontal line through geometric center of cavity agree well with the reference results. However there is slight difference for the results of u-velocity along vertical line through geometric center of cavity near the stationary bottom wall.

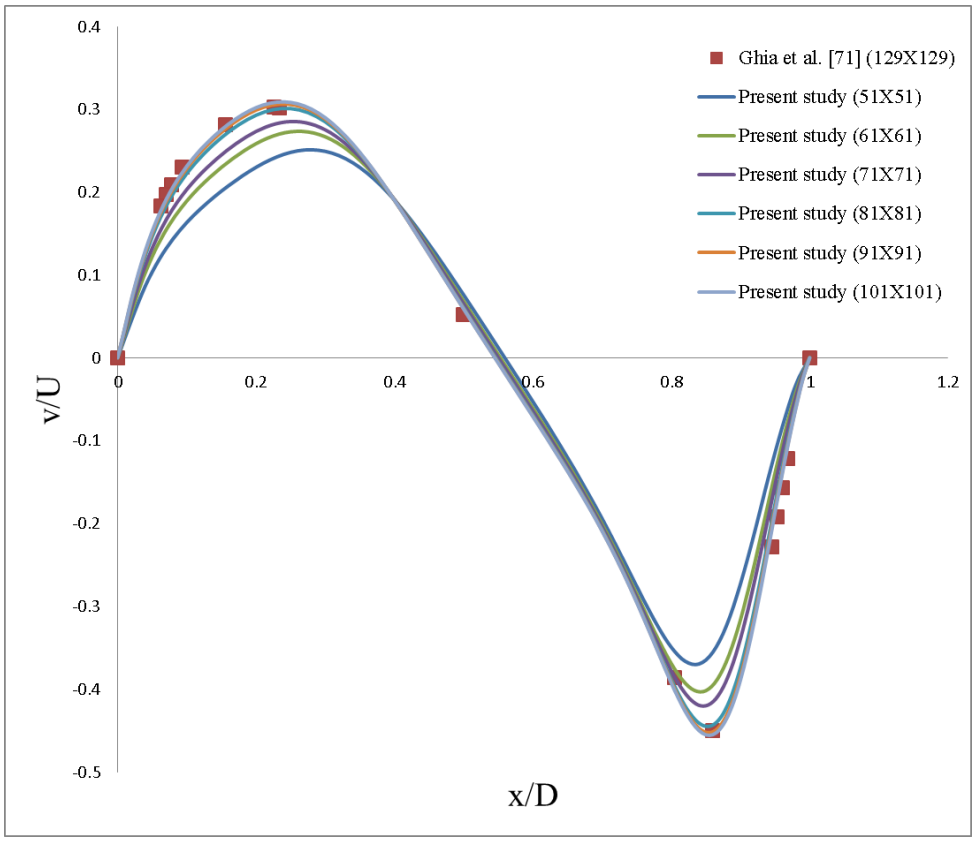

Figure 5.8: Results for v-velocity along horizontal line through geometric center of cavity for Re=400.

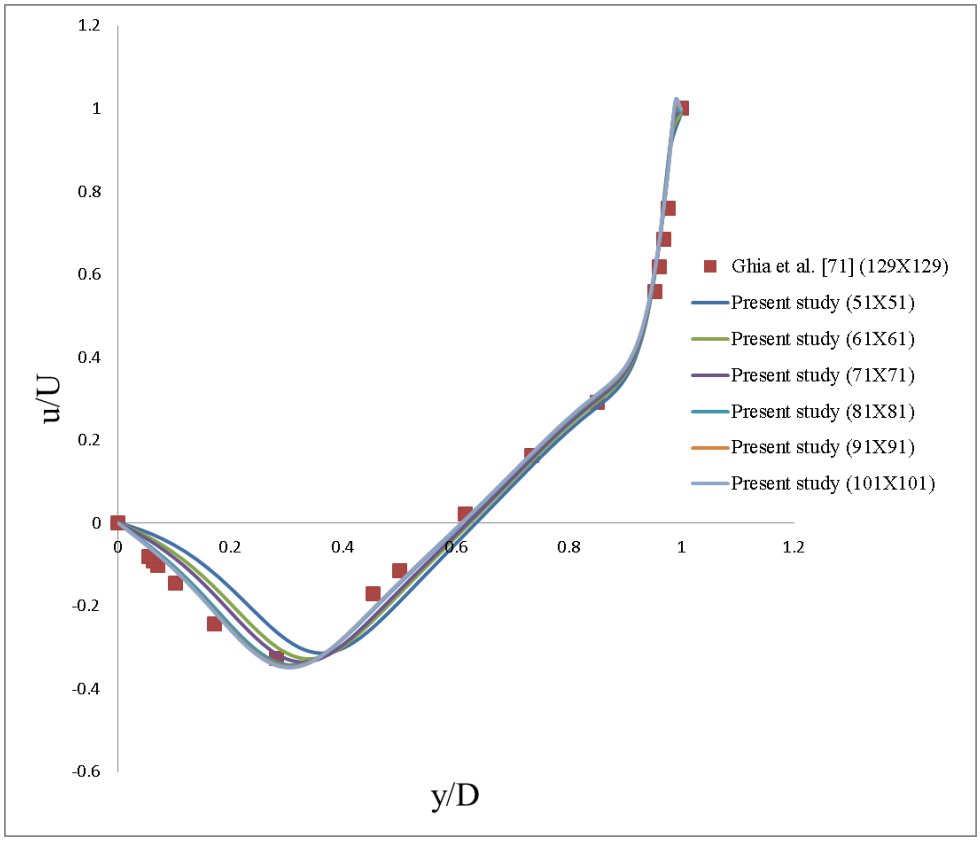

Figure 5.9: Results for u-velocity along vertical line through geometric center of cavity for Re=400.

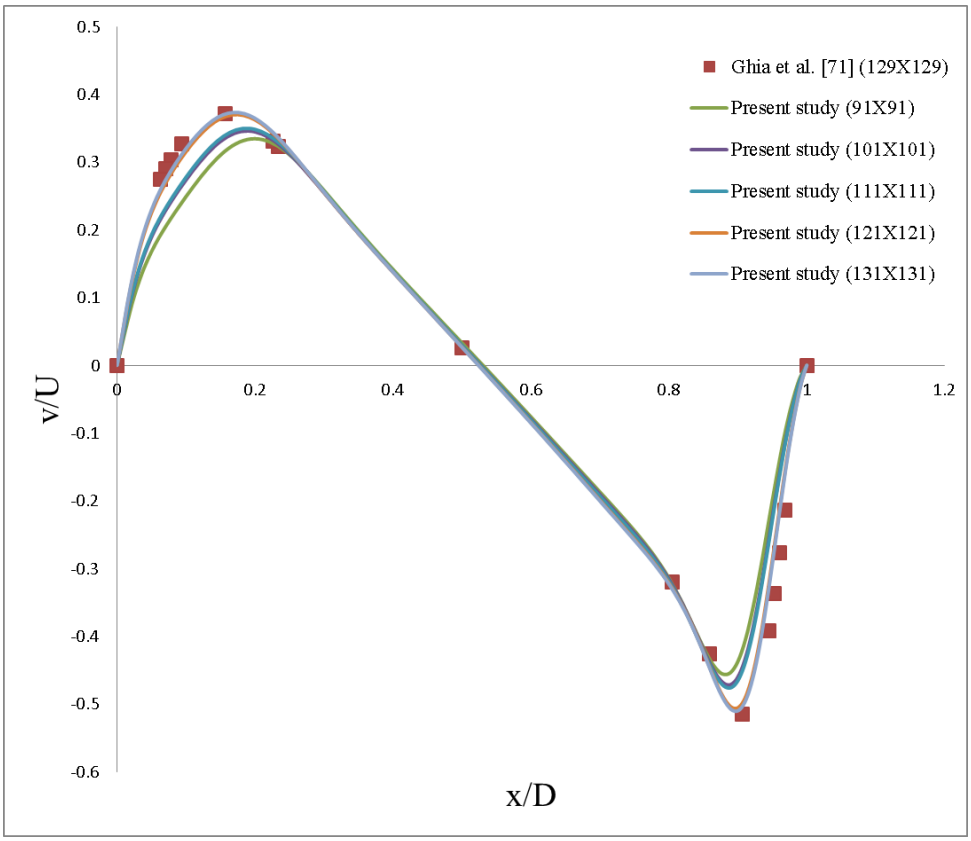

Figure 5.10: Results for v-velocity along horizontal line through geometric center of cavity for Re=1000.

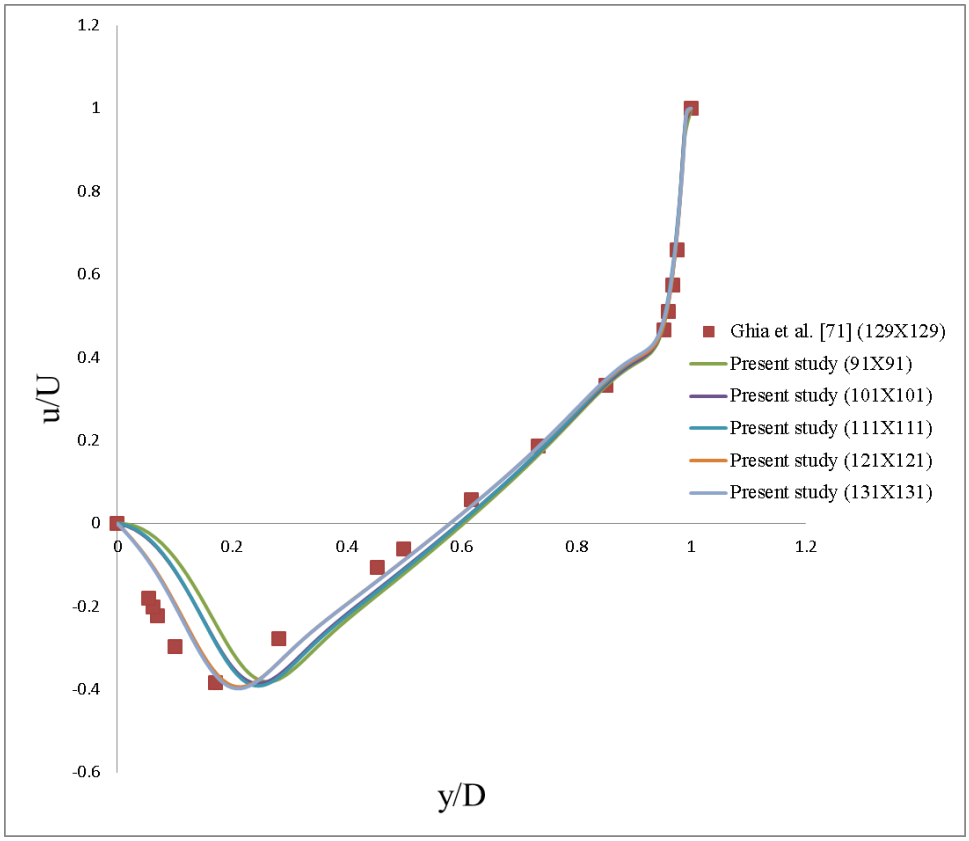

Figure 5.11: Results for u-velocity along vertical line through geometric center of cavity for Re=1000.

Velocity field inside the cavity for Re = 400 and Re = 1000 are presented in Figure 5.12 and Figure 5.13 respectively. Since the solid boundaries of the cavity are defined with the use of immersed boundary method, the computational domain extends the boundaries of cavity.

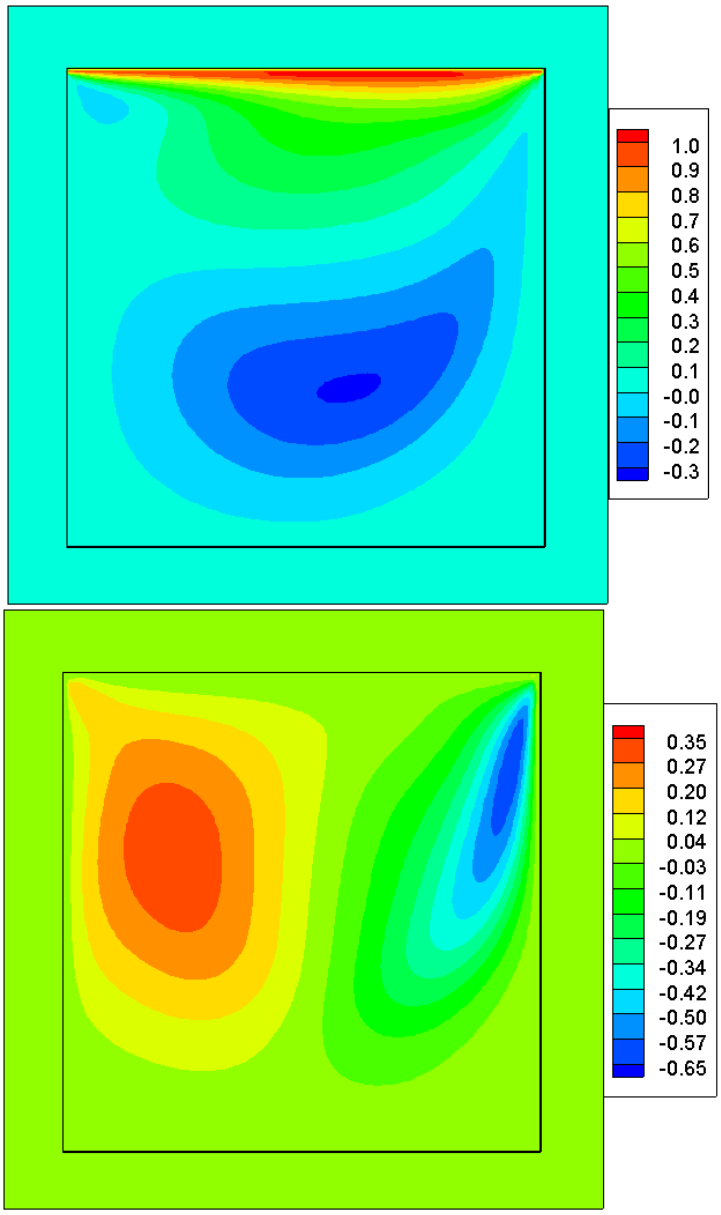

Figure 5.12: Velocity contours inside the cavity for Re=400; top: u-velocity [m/s]; bottom: v-velocity [m/s].

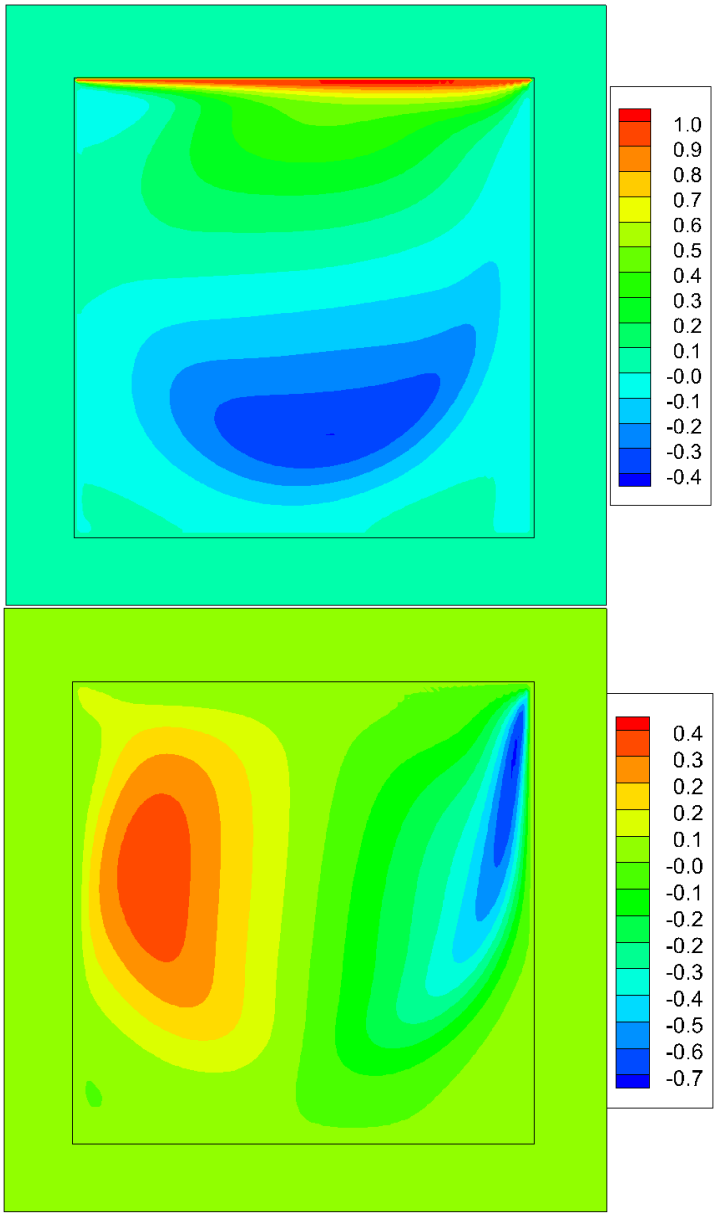

Figure 5.13: Velocity contours inside the cavity for Re=1000; top: u-velocity [m/s]; bottom: v-velocity [m/s].

5.3 Flow Around Circular Cylinder

In this test case, the unsteady nature of flow around a bluff body is investigated. For this purpose flow around a circular cylinder is simulated by solving 3D incompressible Navier-Stokes equations on a uniformly spaced grid. In the simulations of vortex dominated flow fields, aside from the near body region, the wake region of the flow must be accurately resolved. Using sixth order compact scheme on a uniform grid provides an opportunity to accurately capture the downstream flow field.

The xy-plane of the computational domain with dimensions compared to diameter of circular cylinder is presented in Figure 5.14. Freestream Reynolds number is $Re_\infty = 100$.

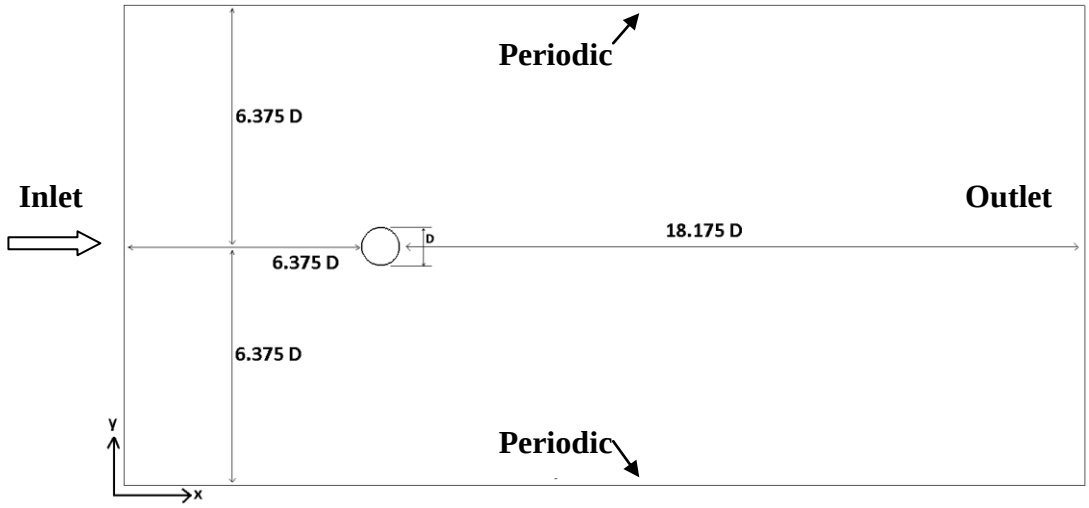

Figure 5.14: The xy-plane of the computational domain.

Periodic boundary conditions are applied in y and z directions for the simulation. The boundary conditions applied at the inlet are $u = U_\infty$, $v = 0$, $w = 0$ and $\partial p/\partial x = 0$. At the outlet, the boundary conditions are $\partial u/\partial x = 0$, $\partial v/\partial x = 0$, $\partial w/\partial x = 0$ and $p = 0$. The grid spacings along the x and y directions are $\Delta x = \Delta y = 0.05\, m$.

Freestream values are imposed as initial conditions. At the beginning, symmetrical vortex regions form behind the circular cylinder. As the time passes, this symmetrical structure deteriorates and the vortices begin to shed. The development of the flow field starting from initial time is presented in Figure 5.15.

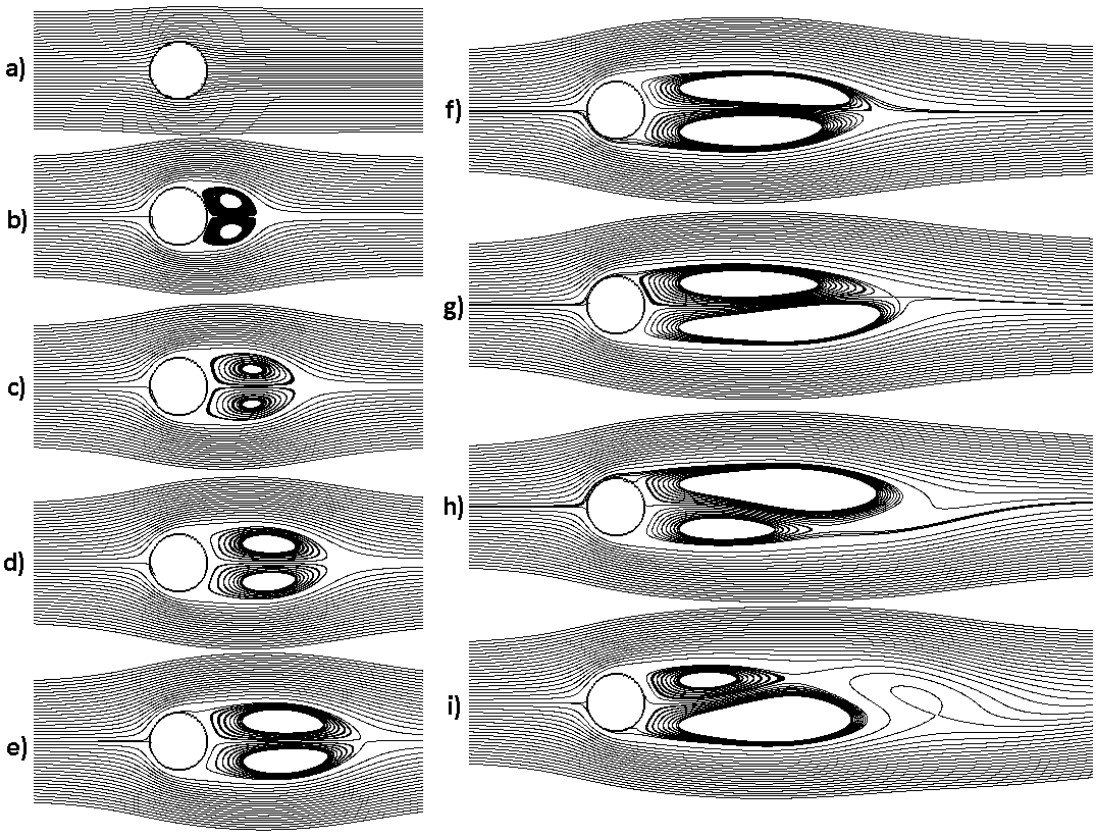

Figure 5.15: Development of flow field; a)0s, b)2s, c)4s, d)6s, e)8s, f)14s, g)17s, h)20s, i)23s.

As more time passes a periodic vortex shedding process develops behind the circular cylinder. Two vortex shedding cycle is shown in Figure 5.16. The process of instantaneous "alleyways" mentioned in [72] may be seen in this figure. At the beginning of a period a vortex region forms from the lower side of the cylinder, expands and moves forward (sheds). During 3this time, the flow is sucked from the top in to this vortex region and another vortex region forms at the upper side. As the vortex at the lower side moves forward, the vortex at the top expands and begins to shed. This time the flow is sucked from the bottom and the process repeats itself.

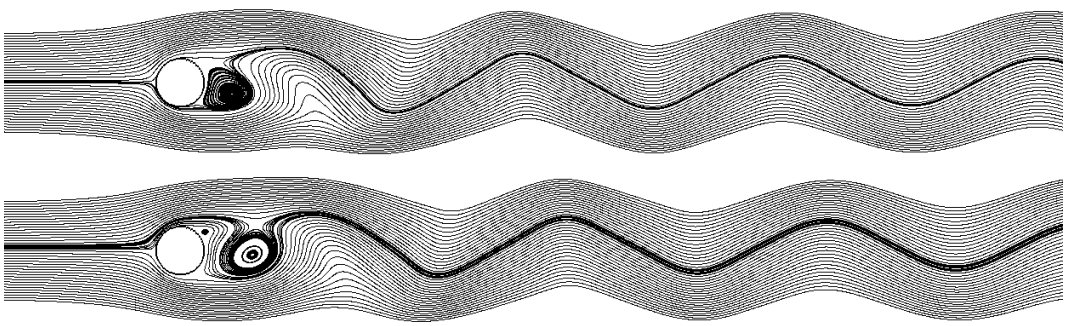

Figure 5.16: Flow field behind the cylinder; 141s – 153s.

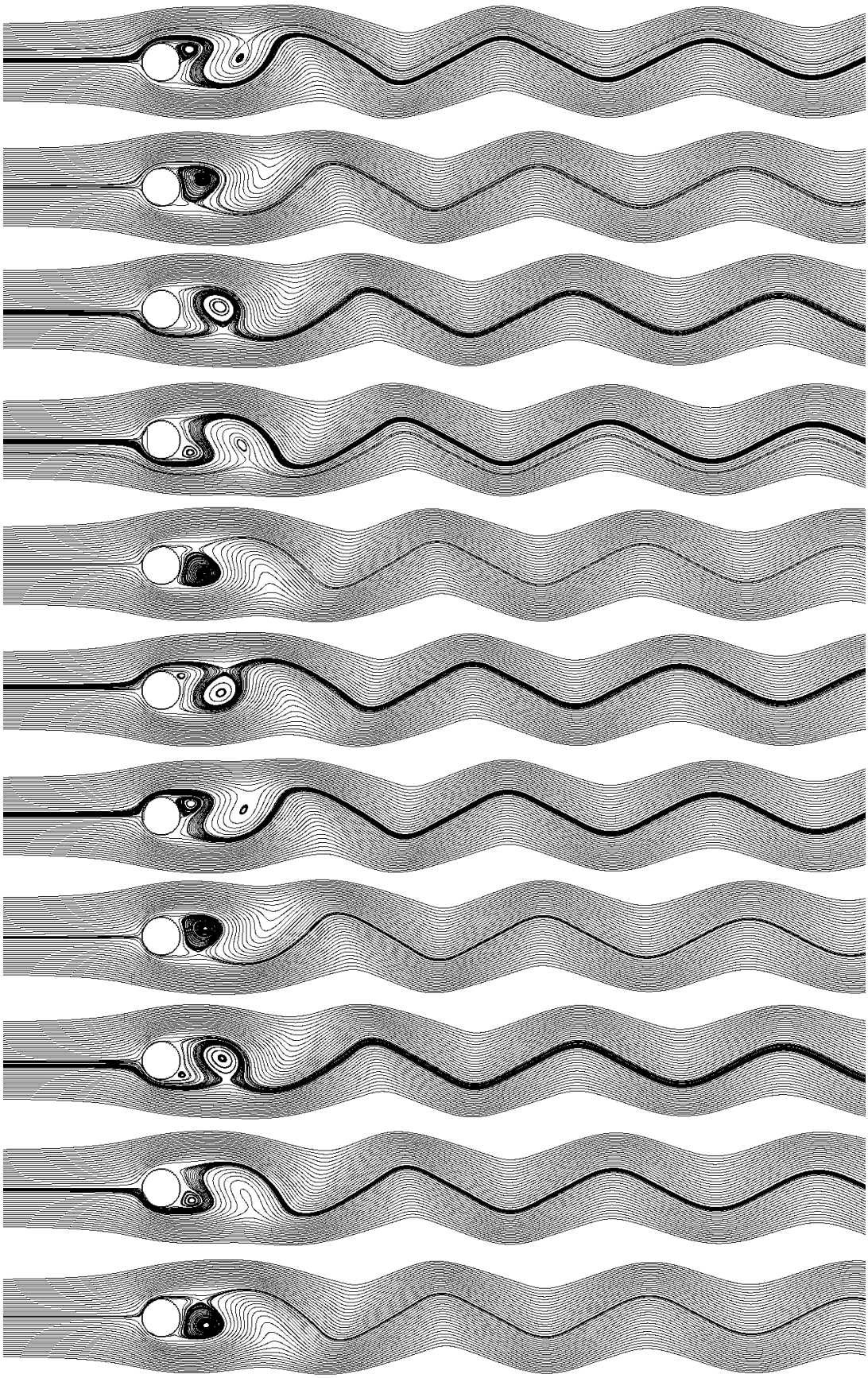

Figure 5.16 (continued): Flow field behind the cylinder; 141s – 153s.

Strouhal number is a non-dimensional parameter that gives information about the frequency of the vortex shedding. It depends on Reynolds number and freestream velocity. It is defined as:

$$St = \frac{fD}{U_\infty} \tag{5.1}$$

Here, f, D and U_∞ are the shedding frequency, cylinder diameter and freestream velocity respectively. The shedding frequency is calculated from the feedback of flow variables at the downstream of the cylinder. During the simulation, the time histories of flow variables at a point on the centerline of the cylinder are reported. The point is located at the downstream with a distance of radius from the cylinder. The variation of v-velocity with time at this point is presented in Figure 5.17.

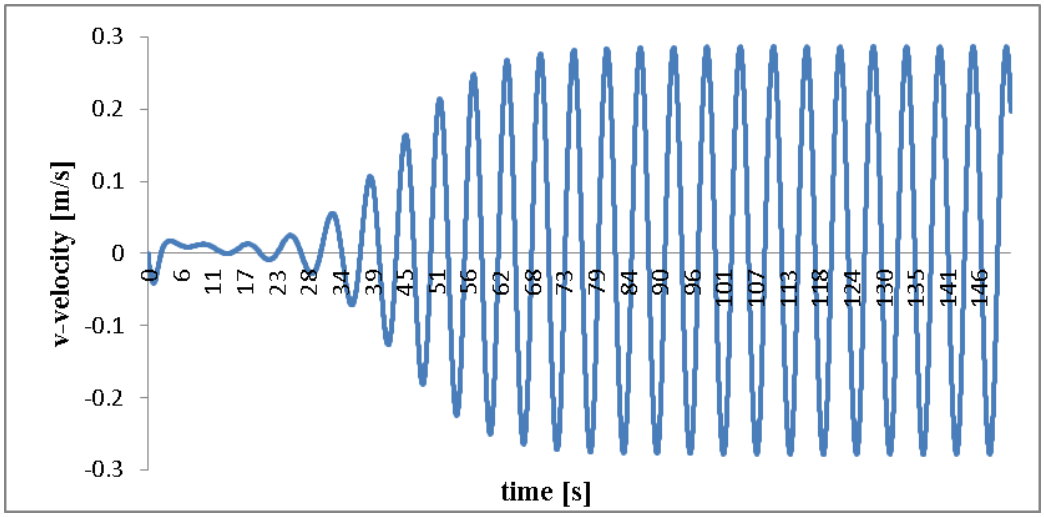

Figure 5.17: Time history of v-velocity.

The flow becomes periodic after 70s and its period is 5.86 s. Strouhal number calculated using this period is $St = 0.1706$. Several results obtained by others are shown in Table 4.1.

Table 5.1: Reference results for flow around cylinder.

Reference	Strouhal number, St
Willimson [72]	0.165
Engelman and Jamnia [73]	0.173
Behr et al. [74]	0.1711
Liu et al. [75]	0.164
Present study	0.1706

The pressure and u-velocity fields for half period of vortex shedding cycle with 1 s intervals are presented in Figure 5.18 and Figure 5.19. Since the solid boundaries are represented by immersed boundary method, a flow field develops inside the cylinder too. However, the flow around the cylinder is independent from the one inside.

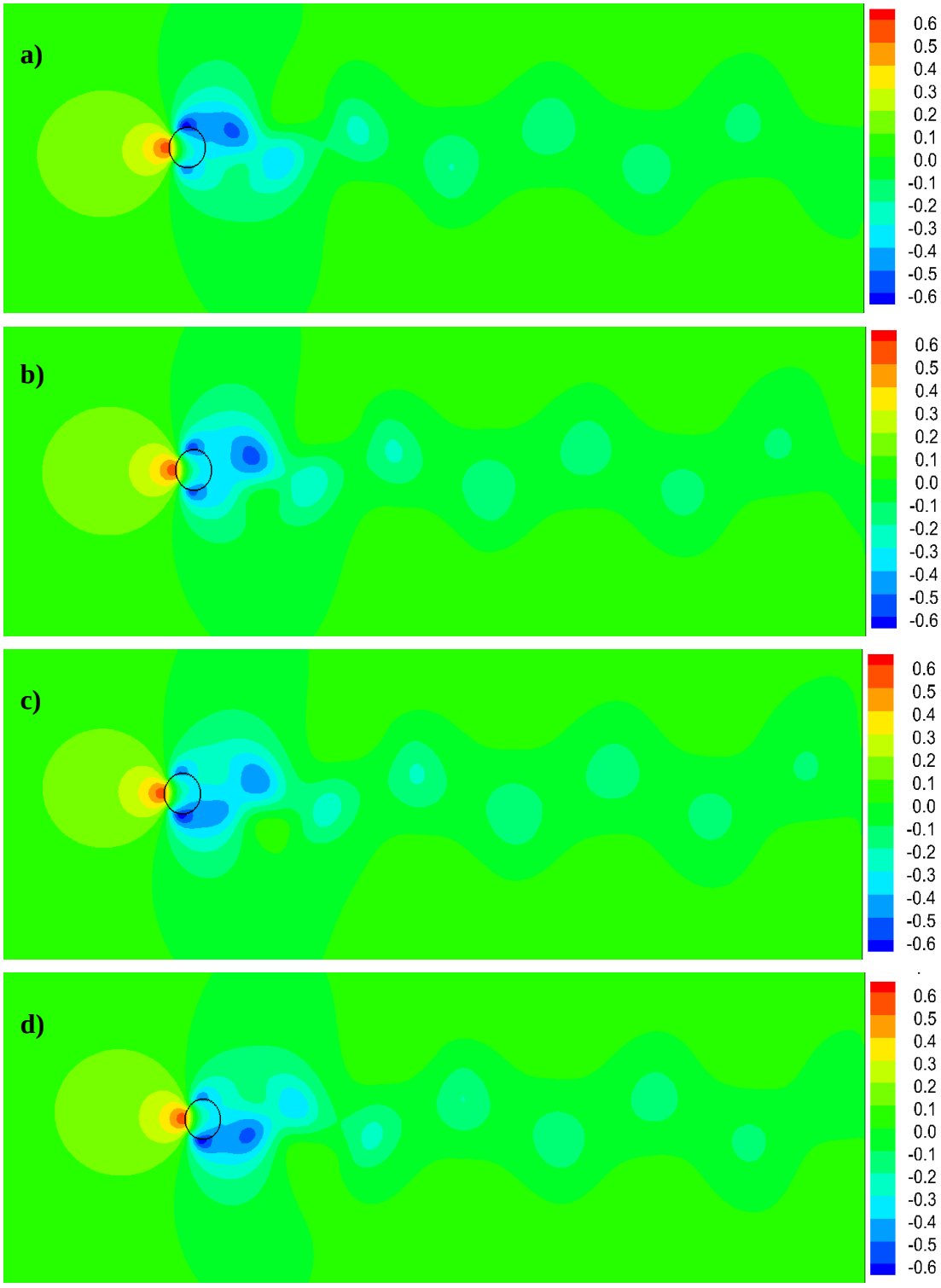

Figure 5.18: Pressure field for half period of vortex shedding cycle [Pa]: a) t = 150 s. b) t = 151 s. c) t = 152 s. d) t = 153 s.

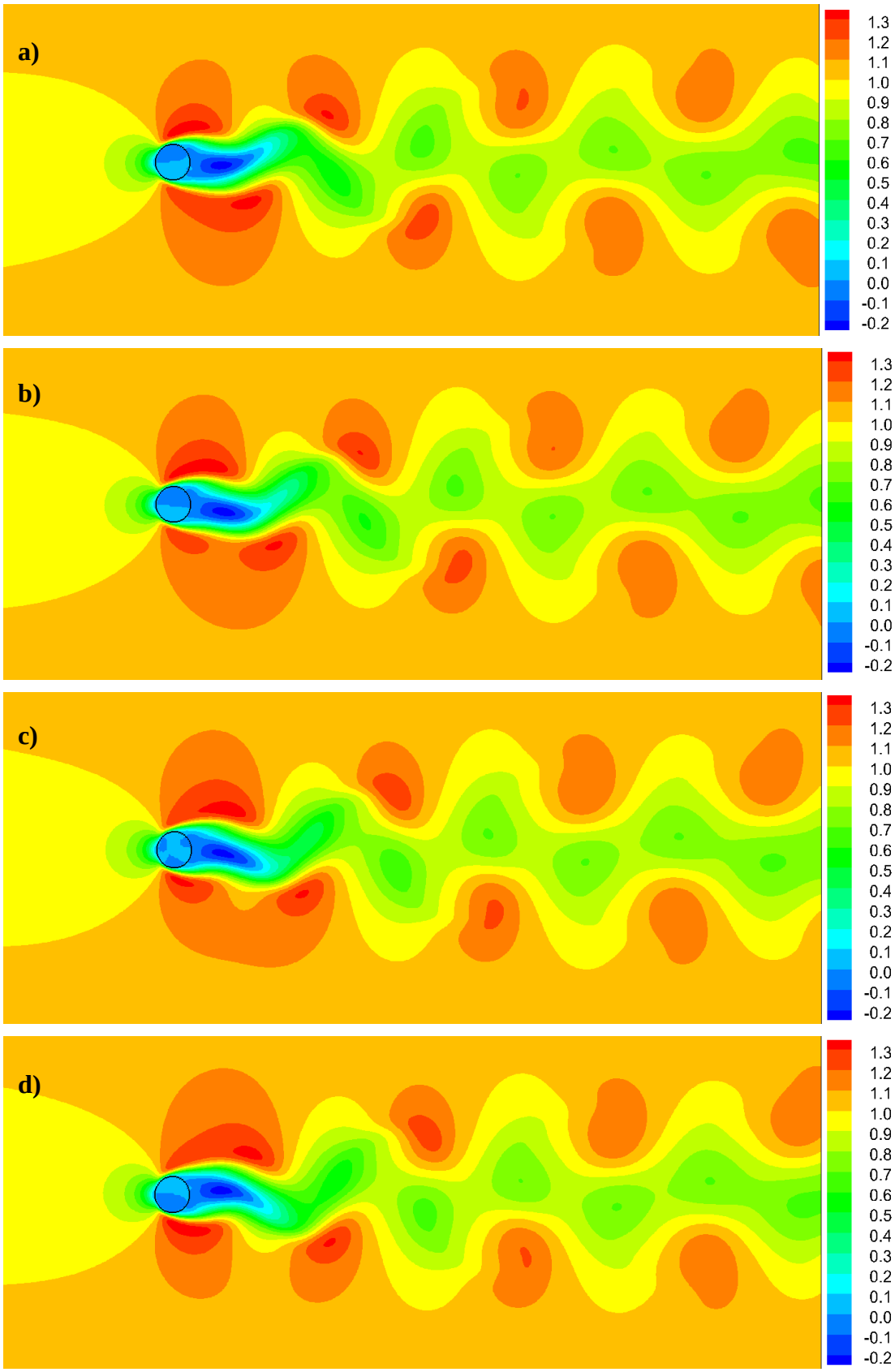

Figure 5.19: Velocity field for half period of vortex shedding cycle [m/s]: a) t =150 s b) t = 151 s. c) t = 152 s. d) t = 153 s.

6. CONCLUSION

In this thesis study, computational fluid dynamics (CFD) simulations are conducted with high order compact scheme and immersed boundary method (IBM) on a graphics processing unit (GPU). 3D incompressible Navier-Stokes equations are solved for three basic test cases; flow over a flat plate, lid driven cavity flow and flow around a circular cylinder. In order to take advantage of its easy and straightforward grid formation, the solid boundaries are represented with IBM in these test cases. For this purpose a direct forcing approach is practiced by inserting a forcing term into the momentum equations. Boundary velocity values are computed via interpolation of the grid point values inside the flow field.

Uniform grids are used for all test cases. A sixth order central compact finite difference scheme is used for the discretization of the first and second spatial derivatives. In the simulations of vortex dominated flow fields, aside from the near body region, the wake region of the flow must be accurately resolved. Using sixth order compact scheme on a uniform grid provides an opportunity to accurately capture the wake region. Conjugate gradient method is used for the solution of the Poisson equation. The second order Adams-Bashfort method is used for the time integration.

Calculations are carried out with using single precision floating point format except for the solution of Poisson equation. Double precision floating point format is employed for the iterations of conjugate gradient method. The linear system of equations of compact schemes is solved with direct method. The coefficient matrix is generated and its inverse is calculated and stored at the beginning of each simulation and then repeatedly multiplied with multiple right hand side sets at every time step.

The results obtained from the flow over a flat plate test case are compared to the Blasius boundary layer theorem solutions. For the lid driven cavity flow and flow around a circular cylinder test cases, the results are compared to the reference

solutions with similar cases found on literature. All results obtained in this study are in very good agreement with theoretical and reference results.

REFERENCES

[1] **Peskin**, **C.S.** *Flow patterns around heart valves: a numerical method*. Journal of Computational Physics, 1972. 10: p.252–271.

[2] **Kang**, **S.** An improved immersed boundary method for computation of turbulent flows with heat transfer, 2009. (Doctoral dissertation). Retrieved from web.stanford.edu/group/uq/pdfs/thesis/

[3] **Mittal, R., Iaccarino, G.** *Immersed boundary methods*. Annual Review of Fluid Mechanics, 2005. 37: p .239–261.

[4] **Iaccarino, G., Verzicco, R.** *Immersed boundary technique for turbulent flow simulations*. Applied Mechanics Review, 2003. 56(3): p. 331–347.

[5] **Ghias, R., Mittal, R., Dong, H.** *A sharp interface immersed boundary method for compressible viscous flows*. Journal of Computational Physics, 2007. 225: p. 528–553.

[6] **Liu, Q., Vasilyev, O.V.** *A Brinkman penalization method for compressible flows in complex geometries*. Journal of Computational Physics, 2007. 227: p. 946–966.

[7] **Uhlmann, M.** *An immersed boundary method with direct forcing for the simulation of particulate flows*. Journal of Computational Physics, 2005. 209: p. 448–476.

[8] **Yu, Z., Shao, X.** *A direct-forcing fictitious domain method for particulate flows*. Journal of Computational Physics, 2007. 227: p. 292–314.

[9] **Atzberger, P.J., Kramer, P.R., Peskin, C.S.** *A stochastic immersed boundary method for fluid-structure dynamics at microscopic length scales*. Journal of Computational Physics, 2007. 224: p. 1255–1292.

[10] **Gilmanov, A., Sotiropoulos, F.** *A hybrid cartesian/immersed boundary method for simulating flows with 3D geometrically complex moving bodies*. Journal of Computational Physics, 2005. 207: p. 457–492.

[11] **Zhao, H., Freund, J.B., Moser, R.D.** *A fixed-mesh method for incompressible flow-structure systems with finite solid deformations*. Journal of Computational Physics, 2008. 227: p. 3114–3140.

[12] **Ge, Y., Fan, L.S.** *Three-dimensional direct numerical simulation for film-boiling contact of moving particle and liquid droplet*. Physics of Fluids, 2006. 18, 117104.

[13] **Iaccarino, G., Moreau, S.** *Natural and forced conjugate heat transfer in complex geometries on cartesian adapted grids*. Journal of Fluids Engineering, 2006. 128: p. 838–846.

[14] **Yu, Z., Shao, X., Wachs, A.** *A fictitious domain method for particulate flows with heat transfer*, Journal of Computational Physics, 2006. 217: p. 424–452.

[15] **Smolarkiewicz, P.K., Sharman, R., Weil, J., Perry, S.G., Heist, D., Bowker, G.** *Building resolving large-eddy simulations and comparison with wind tunnel experiments*. Journal of Computational Physics, 2007. 227: p. 633–653.

[16] **Fauci, L.J., Dillon, R.** *Biofluidmechanics of reproduction*. Annual Review of Fluid Mechanics, 2006. 38: p. 371–394.

[17] **Hirsh**, **R.S.** *Higher order accurate difference solutions of fluid mechanics problem by a compact differencing technique.* Journal of Computational Physics, 1975. 19: p. 90–109.

[18] **Holly F.M., Usseglio P.J.M.** *Dispersion simulation in two-dimensional tidal flow*. Journal of Hydraulic Engineering 1984. 110(9): p. 05–26.

[19] **Dennis S.C.R., Hundson J.D.** *Compact h^4 finite difference approximations to operators of Navier–Stokes type*. Journal of Computational Physics, 1989. 85: p. 390–416.

[20] **Collatz, L.** The Numerical Treatment of Differential Equations. 1966, New York: Springer-Verlag.

[21] **Kopal Z.** Numerical Analysis. 1961, New York: Wiley.

[22] **Lele**, **S.K.** *Compact finite-difference schemes with spectral-like resolution*. Journal of Computational Physics, 1992. 103(1): p. 16–42.

[23] **Spotz W.F., Carey G.F.** *High-order compact scheme for the streamfunction vorticity equations*. International Journal of Numerical Methods in Engineering, 1995. 38(3): p. 497–512.

[24] **Spotz W.F., Carey G.F.** *A high-order compact formulation for the 3D Poisson equation*. Numerical Methods for Partial Differential Equations, 1996. 12(2): p. 35–43.

[25] **Weinan, E., Liu, J.G.** *Essentially compact schemes for unsteady viscous incompressible flows*. Journal of Computational Physics, 1996. 126: p. 122-138.

[26] **Gamet, L., Ducros, F., Nicoud, F., Poinsot, T.** *Compact finite difference schemes on non-uniform meshes. Application to direct numerical simulations of compressible flows*. International Journal for Numerical Methods in Fluids, 1999. 29: p. 159-191.

[27] **Meitz, H.L., Fasel, H.F.** *A compact-difference scheme for the Navier–Stokes equations in velocity-vorticity formulation*. Journal of Computational Physics, 2000. 157: p. 371-403.

[28] **Lee, C., Seo, Y.** *A new compact spectral scheme for turbulence simulations*. Journal of Computational Physics, 2002. 183: p. 438.

[29] **Nagarajan, S., Lele, S.K., Ferziger, J.H.** *A robust high order compact method for large eddy simulation*. Journal of Computational Physics, 2003. 191: p. 392-419.

[30] **Shukla, R.K., Tatineni, M., Zhong, X.L.** *Very high order compact finite difference schemes on non-uniform grids for incompressible Navier-Stokes equations*. Journal of Computational Physics, 2007. 224(2): p. 1064-1094

[31] **Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Kruger, J., Lefohn, A.E., Purcell, T.J.** "*A survey of general-purpose computation on graphics hardware*". Computer Graphics Forum, 2007. 26(1): p. 80-113.

[32] **Elsen, E., LeGresley, P., Darve, E.** *Large calculation of the flow over a hypersonic vehicle using a GPU*. Journal of Computational Physics, 2008. 227: p. 10148-10161.

[33] **Hagen, T.R., Lie, K.A., Natvig, J.R.** *Solving the Euler equations on graphics processing units*. Computational Science – Iccs 2006, Pt 4, Proceedings, 2006. 3994: p. 220-227.

[34] **Brandvik, T., Pullan, G.** "*Acceleration of a two-dimensional Euler flow solver using commodity graphics hardware*". Proceedings of the Institution of Mechanical Engineers Part C - Journal of Mechanical Engineering Science, 2007. 221(12): p. 1745-1748.

[36] **Corrigan, A., Camelli, F.F., Lohner, R., Wallin, J.** *Running unstructured grid-based CFD solvers on modern graphics hardware*. International Journal for Numerical Methods in Fluids, 2011. 66(2): p. 221-229.

[37] **Tolke, J., Krafczyk, M.** *TeraFLOP computing on a desktop PC with GPUs for 3D CFD*. International Journal of Computational Fluid Dynamics, 2008. 22(7): p. 443-456.

[38] **Tutkun**, **B.** Computational fluid dynamics analysis using a high order compact scheme on a GPU, 2012. *PhD Thesis*, ITU, Istanbul, Turkey.

[39] **Joel, M.P., Ferziger, H.** Computational Methods for Fluid Dynamics. 3th ed. 2002, New york: Springer-Verlag.

[40] **Peskin, C.S.** *The immersed boundary method*. Acta Numerica, 2002. p.1-39

[41] **Peskin, C.S.** *The fluid dynamics of heart valves: experimental, theoretical, and computation methods*. Annual Review of Fluid Mechanics, 1982. 14: p. 235-259

[42] **Goldstein, D., Handler, R., Sirovich, L.** *Modeling a no-slip flow boundary with an external force field*. Journal of Computational Physics, 1993. 105(2): p. 354-366.

[43] **Lai, MC., Peskin, C.S.** *An immersed boundary method with formal second-order accuracy and reduced numerical viscosity*. Journal of Computational Physics, 2000. 160(2): p. 705-719.

[44] **Saiki, E.M., Biringen, S.** *Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method*. Journal of Computational Physics, 1996. 123(2): p. 450-465.

[45] **Su, S.W., Lai, M.C., Lin, C.A.** *An immersed boundary technique for simulating complex flows with rigid boundary*. Computers and Fluids, 2007. 36(2): p. 313-324.

[46] **Mohd-Yusof., J.** *Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries*. Center for Turbulence Research Annual Research Briefs, 1997. 161(1): p. 317-327.

[47] **Fadlun, E.A., Verzicco, R., Orlandi, P., Mohd-Yusof, J.** *Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations*. Journal of Computational Physics, 2000. 161(1): p. 35-60.

[48] **Balaras, E.** *Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations*. Computers and Fluids, 2004. 33(3): p. 375-404.

[49] **Zhang, N., Zheng, Z.C.** *An improved direct-forcing immersed-boundary method for finite difference applications*. Journal of Computational Physics, 2007. 221(1): p. 250-268.

[50] **Gilmanov, A., Sotiropoulos, F., Balaras, A.** *A general reconstruction algorithm for simulating flows with complex 3d immersed boundaries on Cartesian grids*. Journal of Computational Physics, 2003. 191(2): p. 660-669.

[51] **Majumdar, S. Iaccarino, G. Durbin, P.** *Rans solvers with adaptive structured boundary non-conforming grids*. Center for Turbulence Research Annual Research Briefs, 2001. p. 353-366.

[52] **Tseng, Y.H., Ferziger J.H.** *A ghost-cell immersed boundary method for flow in complex geometry*. Journal of Computational Physics, 2003. 192(2): p. 593-623.

[53] **Ye, T., Mittal, R., Udaykumar, H.S., Shyy, W.** *An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries*. Journal of Computational Physics, 1999. 156(2): p. 209-240.

[54] **Udaykumar, H.S., Kan, H.C., Shyy, W., Tran-Son-Tay, R.** *Multiphase dynamics in arbitrary geometries on fixed Cartesian grids*. Journal of Computational Physics, 1997. 137(2): p. 366-405.

[55] **Ferziger, J.H., Peric, M.** Computational Methods for Fluid Dynamics. 2002, Berlin: Springer-Verlag.

[56] **Trapnell, C. and Schatz, M.C.** "*Optimizing data intensive GPGPU computations for DNA sequence alignment*". Parallel Computing, 2009. 35(8-9): p. 429-440.

[57] **Shi, H.X., Schmidt, B., Liu, W.G. and Muller-Wittig, W.** "*Accelerating Error Correction in High-Throughput Short-Read DNA Sequencing Data with CUDA*". 2009 Ieee International Symposium on Parallel & Distributed Processing, Vols 1-5, 2009: p. 1546-1553.

[58] **Klockner, A., Warburton, T., Bridge, J. and Hesthaven, J.S.** "*Nodal discontinuous Galerkin methods on graphics processors*". Journal of Computational Physics, 2009. 228(21): p. 7863-7882.

[59] **Ma, W.J. and Agrawal, G.** "*A Compiler and Runtime System for Enabling Data Mining Applications on GPUs*". Acm Sigplan Notices, 2009. 44(4): p. 287-288.

[60] **Sintorn, E. and Assarsson, U.** "*Fast parallel GPU-sorting using a hybrid algorithm*". Journal of Parallel and Distributed Computing, 2008. 68(10): p. 1381-1388.

[61] **Mcgraw, T. and Nadar, M.** "*Stochastic DT-MRI connectivity mapping on the GPU*". Ieee Transactions on Visualization and Computer Graphics, 2007. 13(6): p. 1504-1511.

[62] **Stone, S.S., Haldar, J.P., Tsao, S.C., Hwu, W.M.W., Sutton, B.P. and Liang, Z.P.** "*Accelerating advanced MRI reconstructions on GPUs*". Journal of Parallel and Distributed Computing, 2008. 68(10): p. 1307-1318.

[63] **Maintz, S., Eck, B. and Dronskowski, R.** "*Speeding up plane-wave electronic-structure calculations using graphics-processing units*". Computer Physics Communications, 2011. 182(7): p. 1421-1427.

[64] **Harvey, M.J. and De Fabritiis, G.** "*An Implementation of the Smooth Particle Mesh Ewald Method on GPU Hardware*". Journal of Chemical Theory and Computation, 2009. 5(9): p. 2371-2377.

[65] **Harris, M.J.** "*Fast Fluid Dynamics Simulation on the GPU*", in GPU Gems. 2004, Pearson Education: Boston. p. 637-665.

[66] **Hagen, T.R., Lie, K.A. and Natvig, J.R.** "*Solving the Euler equations on graphics processing units*". Computational Science - Iccs 2006, Pt 4, Proceedings, 2006. 3994: p. 220-227.

[67] **Brandvik, T. and Pullan, G.** "*Acceleration of a two-dimensional Euler flow solver using commodity graphics hardware*". Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science, 2007. 221(12): p. 1745-1748.

[68] **Brandvik**, **T. a. G. P.** "*Acceleration of a 3D Euler Solver Using Commodity Graphics Hardware*.", in AIAA Paper 2008-607, 46th AIAA Aerospace Sciences Meeting and Exhibit. 2008.

[69] **Elsen, E., Legresley, P. and Darve, E.** "*Large calculation of the flow over a hypersonic vehicle using a GPU*". Journal of Computational Physics, 2008. 227(24): p. 10148-10161.

[70] **Tutkun, B. and Edis, F.O.** "*A GPU Application for High-Order Compact Finite Difference Scheme*". Computers & Fluids, 2012. 55: p. 29-35.

[71] **Ghia, U., Ghia, K.N., Shin, C.T.** *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method.* Journal of Computational Physics, 1982. 48: p. 387–411.

[72] **Williamson, C.H.K.** *Vortex dynamics in the cylinder wake*. Annual Review of Fluid Mechanics, 1996. 28: p. 477–539.

[73] **Engelman MS, Jamnia M-A.** *Transient flow past a circular cylinder: a benchmark solution*. International Journal of Numerical Methods in Fluids, 1990. 11: p. 985-1000.

[74] **Behr, M., Hastreiter, D., Mittal S., Tezduyar, T.E.** *Incompressıble flow past a circular cylinder: Dependence of the computed flow field on the location of the lateral boundaries.* Computer Methods in Applied Mechanics and Engineering, 1995. 123: p. 309-316.

[75] **Liu, C., Zheng, X., Sung, C. H.** *Preconditioned multigrid methods for unsteady incompressible flows*. Journal of Computational Physics, 1987. 139: p. 35–57.

CURRICULUM VITAE

Name Surname: Ufuk ÖZCAN

Place and Date of Birth: Tunceli, TURKEY – 10.01.1987

E-Mail: ozcanuf@itu.edu.tr

EDUCATION:

B.Sc.: Istanbul Technical University, Faculty of Aeronautics and Astronautics, Astronautical Engineering Department (2012).

WORK EXPERIENCE

2012 – Present: EDS Aerospace Engineering Ltd. (R&D Engineer)