Large Eddy Simulation of Turbulent Incompressible Flow using GPU with multigrid algorithm

By
Mudassar Shaikh
Roll No. ME11M16
M.Tech II Yr.-Thermofluids

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

Department Of Mechanical Engineering
Indian Institute Of Technology Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

JULY 2013

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas and words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

[illegible]

(Signature)

Mudassar Shaikh

(Student Name)

ME11M16

(Roll No.)

Approval Sheet

This thesis entitled "Large Eddy Simulation of Turbulent Incompressible Flow using GPU with multigrid algorithm " by Mudassar Shaikh is approved for the degree of Master of Technology from IIT Hyderabad.

[illegible]

(Dr. Vinod Janardhanan) Examiner
Dept. of Chemical Engg.
IITH

[illegible]

(Dr. K. Venkatasubbaiah) Examiner
Dept. of Mechanical Engg.
IITH

[illegible]

(Dr. Raja Banerjee) Adviser
Dept. of Mechanical Engg.
IITH

Acknowledgements

I would like to thank my guide Dr Raja Banerjee for giving me an opportunity to work with him.His guidance in my research work has been invaluable. He has managed to be strict and patient at the same time, during the most demanding times of the work. He is a great person and one of the best mentors, I will always be thankful to him.

I would also like to thank Sathi Rajesh Reddy for his friendly support and encouragement through the entire work of this dissertation. A special note of thanks to all my classmates from Thermal Engineering for inspiring me to do my best always. I would also like to thank all my juniors especially Vatsalya Sharma and Varun Jadon for being a greate support especially during culmination phase of this dissertation. I would also like to thank Mr Madhu Pandicheri for his timely support in the CAE lab. A special note of thanks to all my dear friends in the hostel especially my room mate Pankaj Kumar for making my stay in IIT Hyderabad memorable and enjoyable.

I would like to dedicate this thesis to my amazingly loving and supportive family who has always been with me, no matter where I am.

Mudassar Shaikh

To my family ...

Abstract

The research presented in this dissertation is divided into two parts,the first part focuses purely on computational science, and the second part focuses on fundamental fluid dynamics. The computational science aspect involves implementing an incompressible Navier-Stokes solver on Graphics Processing Units (GPUs), with optimized algorithm like multigrid algorithm to achieve the goal of enhancing performance of existing computational facilities and enabling them to solve more complex and computationally expensive fluid flow problems. The fluid dynamics aspect involves using the GPU-based Navier-Stokes solver to study turbulent flows. Large Eddy Simulation (LES) turbulence model has been implemented successfully to analyze 2d incompressible flow.

Computational Fluid Dynamics (CFD) simulations can be very computationally expensive,especially for Large Eddy Simulations (LES). In LES the large, energy containing eddies are resolved by the computational mesh, but the smaller (sub-grid) scales are modeled. Clusters of CPUs have been the standard approach for such simulations, but an emerging approach is the use of Graphics Processing Units (GPUs), which deliver impressive computing performance compared to CPUs. Recently there has been great interest in the scientic computing community to use GPUs for general-purpose computation (such as the numerical solution of PDEs) rather than graphics rendering. To explore the use of GPUs for CFD simulations, an incompressible Navier-Stokes solver was developed for a GPU. The Navier-Stokes equation is solved via a SMAC algorithm and is spatially discretized using the finite volume method on a rectangular collocated grid. Validation of the GPU-based solver was performed for fundamental bench-mark problems, and a performance assessment indicated that the solver was over an order-of-magnitude faster compared to a CPU. This solver was later extended to perform an LES of turbulent flows. LES has been known to be sensitive to inlet boundary conditions, the effect of different inlet boundary conditions has been observed and summarized for a mixing layer problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Introduction

In recent years the power and diversity of flow simulations as a tool for exploring science and technology involving flow applications have reached new levels. This offers a promising future for providing cost effective alternatives to laboratory experiments, field tests and prototyping. This new power of flow simulation is based mainly on recent developments in high performance computing. Advanced algorithms capable of accurately simulating complex, real world problems and advanced hardware and networking with sufficient memory and bandwidth to execute those simulations are being developed. The flow simulation solvers have come a long way to a point of being a practical tool for solving complex real world problems in different fields of engineering and applied fluid mechanics.

In turbulence modeling RANS approach is useful for qualitative descriptions of flows. However it is unreliable for extracting detailed flow physics. RANS simulation gives a profile which is averaged over time and their solutions do not resolve the wide range of length and time scale that is typically seen in turbulent flows. In various engineering applications scientists or engineers are more interested in the description of large scale fluctuations of flow which contain the desired information of turbulent transfer of momentum, heat and other scalars. This can be achieved by directly solving Navier Stokes equation at all scales as in Direct Numerical Simulations(DNS) or resolving large scale fluctuations as in Large Eddy Simulation (LES). However, DNS and LES are computationally expensive.The computational requirement for LES & DNS is illustrated in Table 1.1.(Wilcox [16]) for a 3D channel flow. As can be seen from the table, there is

Re_H	Re_τ	N_{DNS}	N_{LES}
12300	360	$6.7X10^6$	$6.1X10^5$
30800	800	$4X10^7$	$3.0X10^6$
616000	1450	$1.5X10^8$	$1X10^7$
230000	4650	$2.1X10^9$	$1X10^8$

Table 1.1: Grid Size for DNS and LES

rapid increase in computational cost with increase in Reynolds number. In such high fidelity computation, multicpu based High Performance computing cluster has been traditionally used.

1.2 CFD using Graphics Processing Units

Overview

Driven by the demands of the gaming industry, the graphics hardware has substantially evolved over the years with remarkable floating point arithmetic performance. Small-footprint desktop supercomputers with hundreds of cores that can deliver teraflops peak performance at the price of conventional workstations have been realized. Recently there has been interest in the scientific computing community to use GPUs for general-purpose computation, such as in the numerical solution of PDEs. This interest has arisen out of the high-performance offered by GPU technology at a relativley lower cost. For many years, numerical simulations have been successfully performed using single CPUs for serial execution of programs, CPU clusters for parallel execution across a network of processors (for example, using Message Passing Interface (MPI)), or multiple/multi-core CPUs in a single machine (for example, using POSIX Threads or OpenMP). CPU performance has, however, increased only marginally in recent years, whereas GPU performance has increased dramatically. In the early years, these operations had to be programmed indirectly, by mapping them to graphic operations and using graphic libraries such as OpenGL and DirectX. GPU programming was greatly promoted with the development of Compute Unified Device Architecture (CUDA) by NVIDIA CUDA [8] is a general purpose parallel computing architecture with a new parallel programming model. It comes with a software environment that allows developers to use C/C++ as a high-level programming language. Using CUDA, the latest

NVIDIA GPUs become easily accessible for general purpose applications. The present work involves implementation of a Navier-Stokes solver for incompressible fluid flow based on cpu architecture. Specifically, NVIDIAs CUDA programming model is used to implement the discretized form of the governing equations. The projection algorithm to solve the incompressible fluid flow equations is solved using CUDA kernels, and is implemented so as to exploit the memory hierarchy of the CUDA programming model.

1.3 Multigrid Technique

Overview

The multigrid method is one of the most efficient iterative method known today. The objective of the multigrid method is to accelerate convergence of an iterative scheme. Multigrid consists of the use of hierarchy of grids. The grid is coarsened by a factor of 2 at each level for a structured grid as shown in figure given below. The coarsened grid quickly transmits the information from boundary to achieve a “dirty” solution. This solution provides a better initial guess for the finer grid hence acclerating the convergence by reducing the number of iterations by the order of two. The multigrid technique can be applied using any of the iterative schemes, although the Gauss Seidel algorithm has been used in the current study. In the present work, the pressure poisson equation in the projection algorithm for solving incompressible Navier Stokes equation has been solved using Multigrid algorithm on GPU architecture provided by NVIDIA CUDA.

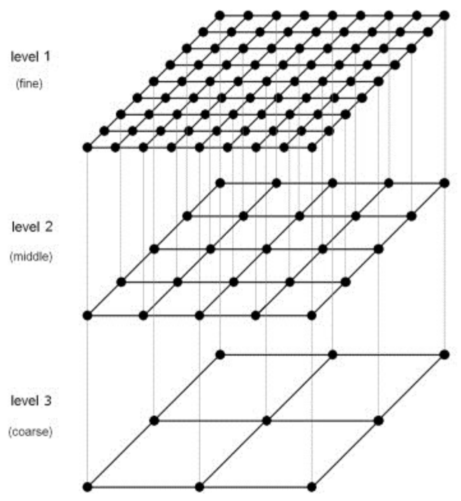

Figure 1.1: Heirarchy of Levels

1.4 A brief overview of turbulence

Turbulence is one of the unsolved problems in the physical sciences. It is characterized by an unsteady, random, chaotic, highly nonlinear, whirling motion of the fluid, which can easily be observed around us. Most of fluid flows and heat transfer problems generally encountered in engineering applications are turbulent in nature. The very unpredictable nature of turbulent flows makes it one of the most challenging problems in engineering sciences. Whenever turbulence is present in a certain flow it appears to be the dominant factor over all other flow phenomena. That is why successful modeling of turbulence greatly increases the quality of numerical simulations. Turbulent flows exibit the following features:

- disorganised, chaotic, seemingly random behavior
- nonrepeatability(sensitivity to initial conditions)
- extremely large range of length and time scales
- enhanced diffusion and dissipation
- time dependence, vorticity and rotationality

Efforts to get computationally cheap analysis of turbulent flows led to the solving of Reynolds Averaged Navier-Stokes equation (RANS). The RANS equation are derived by performing a time averaging of the Navier-Stokes equations after replacing each variables by sum of mean and fluctuating components of the variable. Averaging of Navier-Stokes equation introduces an unknown term in equations ($\overline{\rho u_i u_j}$) known as the Reynolds-stress tensor (τ_{ij}) which cannot be determined analytically. Therefore the RANS equations are not closed, i.e.the number of unknowns are larger than the number of equations. For solving the RANS equation,the Reynolds stress tensor needs to be modeled. The Boussinesq eddy-viscosity is widely used to model Reynold's stress tensor.

Boussinesq Eddy Viscosity Approximation

The similarity between molecular mixing and turbulent fluctuations forms the basis of Boussinesq Eddy Viscosity hypothesis .According to this hypothesis Reynolds stress tensor is linearly proportional to the mean strain-rate tensor which is expressed as

$$-\rho\overline{u_i'u_j'} = \mu_t \left[\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right] - \frac{2}{3}\rho\delta_{ij}k \tag{1.1}$$

where μ_t is the dynamic eddy-viscosity and k represents the specific turbulent kinetic energy of the uctuations and is given by

$$k = \frac{1}{2}u_i'u_j' = \frac{1}{2}\left(\overline{u_1'^2} + \overline{u_2'^2} + \overline{u_3'^2}\right) \tag{1.2}$$

Kinematic eddy-viscosity is not the property of fluid, but is of the flow. For calculating the mean fluid flow variables, μ_t needs to be evaluated. Once the Bousseinesq approach is adopted the main purpose of the turbulence model is the determination of the turbulent viscosity μ_t .Based on the approach followed for modeling μ_t, turbulence models are classified as

- Algebraic or Zero-Equation Models
- One-Equation Models
- Two-Equation Models

Algebraic or Zero Equation Models

Models where the eddy-viscosity is completely determined in terms of the local mean velocity and prescribed parameters are referred to as zero equation, or algebraic models. Most of these models use Prandtl mixing length hypothesis and compute the eddy viscosity as an analogy to the molecular viscosity found in kinetic theory of gases. In these models, the eddy viscosity is defined as:

$$\mu_t = l_{mix}^2 \frac{\partial u_i}{\partial u_j} \tag{1.3}$$

Some of the popular zero equation models are Cebeci-Smith[2] and the Baldwin-Lomax [1] .In general, zero equation model perform reasonably well for free shear flows.

One Equation Models

In these models, a transport equation is solved for turbulent kinetic energy.The turbulent length scale is then algebraically prescribed in one equation models. The kinematic eddy viscosity μ_t is calculated using an algebraic equation, which is given as

$$\mu_t = \rho l_{mix} \sqrt{k} \tag{1.4}$$

Baldwin-Barth [3] and Spalart-Almaras[4] models are two commonly used one equation models. These models give better results than zero equation models.

Two Equation Models

The two equation models have served as foundation for most of the turbulence modelling research during the past two decades. In these models not only is the turbulent kinetic energy equation solved, but another equation for turbulence length scale or some other equivalent quantity is solved. Hence these models are called complete, i.e., they can be used to predict properties of a given flow with no prior knowledge of the turbulence structure. In these models two separate transport equations are solved for these quantities. The first quantity is turbulent kinetic energy (k). For the second quantity mainly rate of dissipation of turbulent kinetic energy (ϵ) or frequency of dissipation(ω) is used. Based on the second parameter used two equation models are classified as $k - \epsilon$ and $k - \omega$.
For the $k - \epsilon$ based models

$$\mu_t \sim \frac{\rho k^2}{\epsilon} \tag{1.5}$$

For the $k - \omega$ based models

$$\mu_t \sim \frac{\rho k}{\omega} \tag{1.6}$$

1.5 Introduction to Large Eddy Simulation

In Large Eddy Simulation large eddies are resolved and small eddies are modeled.Behaviour of the large-scale eddies depends strongly on the forces acting on the flow and on initial and boundary conditions and therefore they are flow-dependent. On the other hand small-scale eddies are generally independent of the larger scale statistics. They are flow-independent. Hence large eddies are directly resolved while small eddies are modeled. Since LES involves modeling the smallest eddies, the size of the smallest cell can be larger than the size of the smallest eddy ie Kolmogorov length scale. Also the timescale required is much larger than Kolmogorov time scale. Hence LES is more economical than DNS(Direct Numerical Simulation which resolves eddies at all scales) in terms of computational cost. Typically the computational cost for LES is only 5-10% of that of DNS. LES is an invaluable tool for deciphering the vortical structure of turbulence, since it allows us to capture deterministically the formation and ulterior evolution of coherent vortices and structures. It also permits the prediction of numerous statistics associated with turbulence and induced mixing. LES applies to extremely general turbulent flows (isotropic, free-shear, wall-bounded, separated, rotating, stratied, compressible, chemically reacting, multiphase, magnetohydrodynamic, etc.). LES has contributed to a blooming industrial development in the aerodynamics of cars, trains and planes, propulsion, turbo-machinery, thermal hydraulics, acoustics and combustion.

The large scale and small scale fluctuations are separated by a process called low-pass filtering .A cut off length or filter width Δ is defined such that the fluctuations falling above this length scale are termed as large scaled fluctuations and are directly resolved and fluctuations below this length scale are termed as subgrid scaled fluctuations and are modeled using a suitable subgrid scale model.The filter width $\Delta$is typically taken to be $(\Delta_x \Delta_y \Delta_z)^{1/3}$. In the present work filtered Navier Stokes has been solved for the large scaled fluctuations and since solver solves for incompressible flow the turbulent stress tensor is the only subgrid term which needs to be modeled. Smagorinsky Lilly Subgrid model [5, 6] has been used to model this term and provide a closure to the problem.

1.6 Literature Survey

GPU & Multigrid Implementation

Using GPUs for CFD applications has been shown by several works done previously. Before the advent of CUDA, programmers had to cast their applications in terms of graphics processing operations. The use of CUDA and modification of Gauss Siedel using red black algorithm in order to solve Pressure Poisson equation has been specified by Shinn et al.[7]. The multigrid technique is one of the most iterative technique known till date. The theory behind the technique has been clearly explained as per Briggs et al [11]. The multigrid technique in combination with red black Gauss siedel in order to perform Direct Numerical Simulation of incompressible turbulent flow has been done by Shinn et al [24].

k-ϵ and Large eddy simulation

The first two equation model was proposed by Kolmogorov (1942). After that various improvements were suggested.The $k-\epsilon$ model was suggested by Launder and extensive work was done on it by Launder & Spalding [25].

The Smagorinsky model for LES was proposed by Smagorinsky in 1963 [5]. In Smagorinsky's work the subgrid scale term was modeled using a model similar to mixing length. The drawbacks in the Smagorinskys model were noted by Germano et. al.[18] and a dynamic Smagorinsky model was proposed. The effect of inlet conditions on the flow downstream was investigated by Smirnov et.al.[22]. Suitable random fluctuations in order to provide realistic inlet boundary were also suggested by Smirnov et al.[22]. Large eddy simulation of turbulent mixing layer were conducted by Vreman et al.[20] using different subgrid scale models and their results were compared with DNS.

1.7 Objective of the present work

The following list summarizes the objectives of the present research work

- To explore GPUs for CFD applications, a 2D incompressible Navier Stokes solver has been implemented for a single GPU.The code is written using CUDA(Compute Unied Device Architecture), which is a programming paradigm developed by NVIDIA. Lid driven cavity has been used as a benchmark problem.

- To extend the solver for solving RANS to enable it to solve turbulent ows. The solver has been validated against standard results for a mixing layer for a turbulent flow case.The GPU-based solver has shown impressive speedup compared to a similar CPU-based flow solver.

- To implement Large Eddy Simulations model, solver which uses Smagorinsky's Eddy Viscosity model for modeling the subgrid term has been developed and has been validated for a 2D mixing layer problem.

- To test the effect of inlet boundary conditions, LES is known to be sensitive to inlet conditions, hence effect of various inlet conditions has been observed.

Chapter 2

GPU and Multigrid

2.1 GPU Architecture

The GPU can be thought essentially as a massively parallel computer, capable of simultaneously executing instructions on a large number of arithmetic units. However, due to the special architecture of the GPU, it is necessary to devise the numerical algorithm and the program structure such that the communication, computation and data access are executed optimally. The architecture of a GPU is quite different from that of a CPU. A GPU is designed with more transistors dedicated for computation and has less resources dedicated for data caching and flow control compared to a CPU. A GPU contains a large number of streaming multiprocessors or cores. It should be noted that the GPU is used as a co-processor in conjunction with the CPU. Typically, the main program is executed on a CPU, and the GPU is utilized by launching kernels from the main program. Thus, usage of the CPU is not eliminated but rather is minimized. The GPU architecture can be classied as SIMD (single-instruction, multiple-data) or SIMT(single-instruction, multiple-thread) [8]. In addition to the architectural differences between CPUs and GPUs, the memory band-width is another important difference. Memory bandwidth is defined as the rate (say in gigabytes per second) at which data is moved into and out of the random access memory (RAM). Modern GPUs have memory bandwidths an order of magnitude greater than CPUs. This is because CPUs have to satisfy constraints of legacy applications and operating systems, which makes increasing memory bandwidth difficult. However, GPUs have less legacy constraints resulting in more memory bandwidth. This increase in memory bandwidth is another factor contributing to the favorable performance of GPUs.

2.2 CUDA programming model

CUDA allows the programmer to define special functions called kernels, such that, when called, are executed N times in parallel by N different CUDA threads [8]. A CUDA program is organized into a host program, consisting of one or more sequential threads running on a host. All threads generated by a kernel define a grid and are organized in blocks. A grid consists of a number of blocks (all equal in size), and each block consists of a number of threads as shown in Figure 2.1 [8]. Each block within the grid is identified through the built-in variable 'blockIdx', accessible within the kernel. The dimension of the thread block is accessed through the built-in variable 'blockDim'. Each thread in a particular thread block is identified using 'threadIdx' built-in variable. By using these built in variables each thread determines which element it should process.

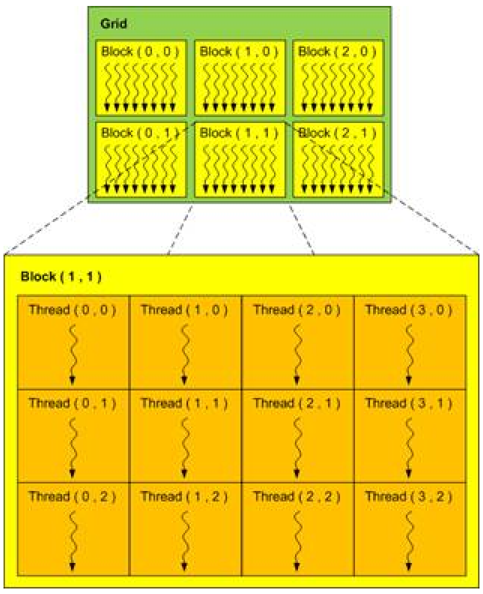

Figure 2.1: GPU Thread Hierarchy

CUDA devices have different types of memories that can be utilized by programmers in order to achieve high performance as illustrated in Figure 2.2 [8]. The global memory is the memory which every thread can access. The data to be processed by the GPU is first transferred from the host memory to the device global memory. Each block has shared memory visible to all threads of the block and each thread has private local memory visible to that thread only. There are also two additional memory spaces accessible by all threads, the constant memory and texture memory. Constant memory provides faster and more parallel data access paths for CUDA kernel execution than the global memory [9]. Texture memory is used to accelerate frequently performed operations and also provides a way to interact with the display capabilities of the GPU.

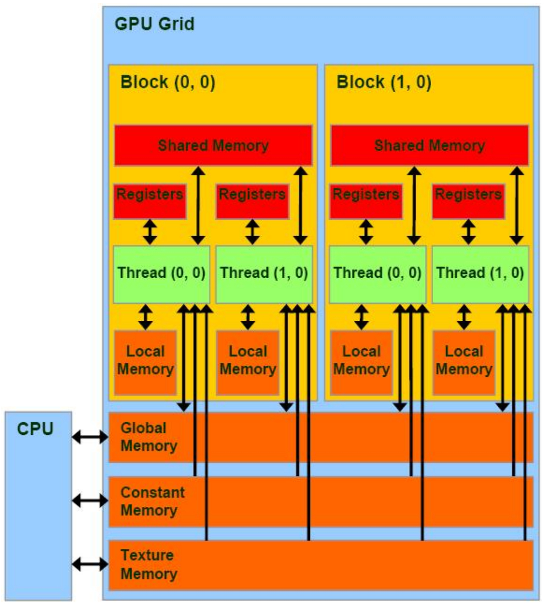

Figure 2.2: Memory model of GPU

2.3 Implementation in CUDA

The projection algorithm is adopted to find a numerical solution to the Navier -Stokes equation for incompressible fluid flows. In the projection algorithm, the velocity field u* is predicted using the momentum equations without the pressure gradient term. The predicted velocity field u* does not satisfy the divergence free condition because the pressure gradient term is not included in the predictor step. By enforcing the divergence free condition on the velocity field at time (t+1), the pressure Poisson equation can be derived from the momentum equations as per Thibault et al [10].The pressure-Poisson equation , which is fully implicit is the most time consuming step. It requires convergence to a high degree (mass error), and consumes nearly 80% of the total computational time. Hence it is necessary to port it on a GPU. The discrete pressure-Poisson equation represents a system of linear equations. A number of iterative linear solvers are available, but the selected solver should suit the parallel nature of the GPU. For an implicit algorithm variable at every point is calculated from its neighboring points at the same time step. For the algorithm to run on parallel threads it is necessary that there are no dependencies between variables on different threads. For a second ordered stencil used to solve 2-D Pressure Poisson equation two colors (say red and black) are required to generate two set of points which are independent of each other. The points with same color are processed simultaneously. A thread is created corresponding to every point of a color. The thread-id and block-id provided by CUDA can be used to map a thread to a point of a color and each color is processed sequentially.The Figure 2.3 [9]shows colored domain and thread configuration for processing the points. The threads are mapped to point of one color at a time. For higher-order stencils, more colors will be needed and that will further complicate the code structure.

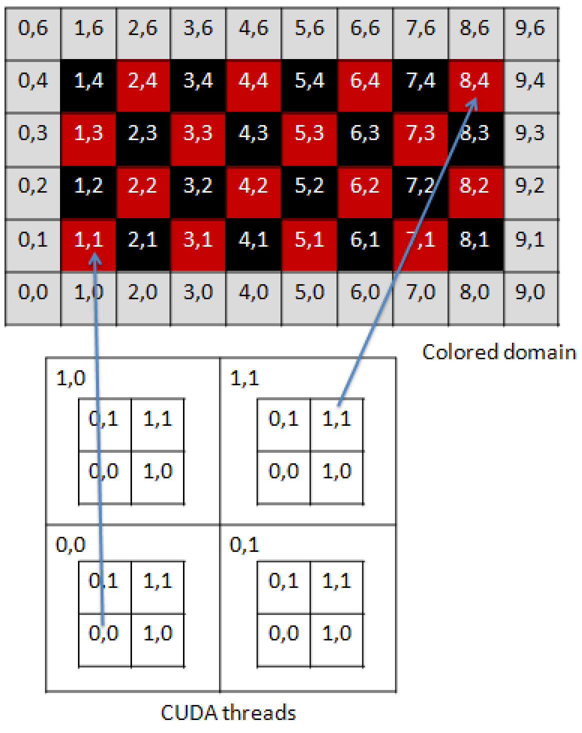

Figure 2.3: CUDA thread mapping arrangement

2.4 Multigrid technique

As already mentioned above the Pressure Poisson Equation is the most computationally expensive step in solving the incompressible Navier-Stokes equation hence it is necessory to accelerate the convergence rate to reduce the overall runtime of the solver. A geometric multi-grid method is employed for this purpose using a V-cycle, as shown in Figure 2.4.

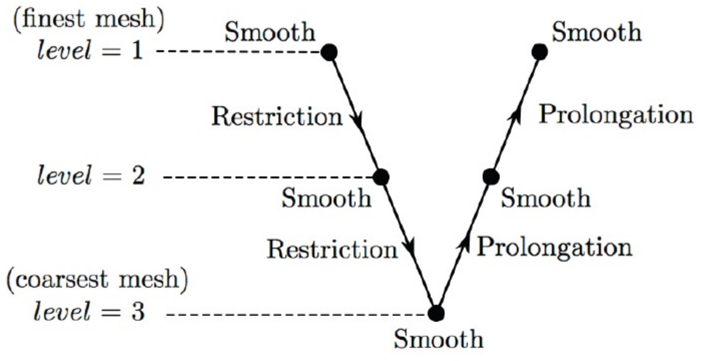

Figure 2.4: V cycle for 3 levels

Basic Theory

The convergence rate of standard iterative solvers (Gauss-Siedel, Jacobi, SOR) has a tendency to stall or to fail in effectively reducing errors after a few number of iterations. A close inspection of this behaviour reveals that the convergence rate is a function of the error field frequency, i.e. the gradient of the error from node to node. If the error is distributed in a high frequency mode, the convergence rate is fast. However, after the first few iterations, the error field is smoothed out i.e. converted to low frequency and the convergence rate deteriorates. Also the error in a low frequency mode on a finer grid gets converted to a high freqency mode on a coarser grid. Multigrid exploits this fact, thus after a few iterations of smoothening the error in high frequency mode it is transfered to coarser grid to convert low frequency components to high frequency components and smoothen it, hence convergence is achieved with lesser number of iterations. A system of linear equaton can be represented

$$Au = f \tag{2.1}$$

where u is the exact solution of the sytem.If v is an approximate solution to the system of equation the residual r is given by

$$Av - f = r \tag{2.2}$$

Also the correction Δu

$$\Delta u = u - v \tag{2.3}$$

Solving equation 2.1,2.2 and 2.3

$$A\Delta u = -r \tag{2.4}$$

$$v_{corrected} = v + \Delta u \tag{2.5}$$

as per Briggs et.al[11]

2.5 Solution Algorithm for Multigrid on GPU

The basic steps involved in the multigrid technique are as follows[12]

- Solve on the fine grid using scheme like Gauss Siedel for a few iterations in order to smoothen the high frequency error Compute and store the residual at each fine grid point.
- Interpolate the residual onto the coarse grid .This operation is called restriction. Since for a structured grid every point on the coarse grid has a corresponding point on the fine grid hence values of residuals can directly be transfered. Generally a weighted average of the neighbouring points is considered. The restriction operator I_1^2 is calculated as follows

$$\begin{aligned} I_1^2 R_{ij} = \frac{1}{8}&\left[R_{2i+1,2j+1} + R_{2i-1,2j+1} + R_{2i+1,2j-1} + R_{2i-1,2j-1}\right] \\ &\frac{1}{4}\left[R_{2i+1,2j} + R_{2i-1,2j} + R_{2i,2j+1} + R_{2i,2j-1}\right] + \frac{1}{2}\left[R_{2i,2j}\right] \end{aligned}$$

[Peiter Wessling][13]

- Solve for the corrections using the equation $\nabla^2(\delta P) = -R_{ij}$ on the coarser grid. These corrections are transfered from coarse grid to fine grid. The corrections for every point in the coarse grid are directly transfered to every point on fine grid, the corrections for other points are determined through interpolation. This operation is called prolongation.

- The corrections at the fine grid points are now added to the intermediate solution to provide a better initial guess and further smoothening is done at every level.
- Finally the prolongated correction is added to the intermediate solution on the finest grid and the convergence is checked after smoothening.
- If convergence is not achieved all the above steps are repeated.

2.6 Pseudocode

```
while error> $\epsilon$ do
    if Level=0 (fine) then
        solve for $\nabla^2 P = S_{ij}$
        ie smoothen pressure using red-black SOR(GPU)
    end if
    for Level=1 to Level<maxm do
        Restrict residual from $R_{fine}$ to $R_{coarse}$ ie from Level-1 to Level
        solve for $\nabla^2(\delta P) = -R$ at each level using restricted R above
        ie smoothen pressure corrections using red-black SOR(GPU)
    end for
    for Level=maxm to Level=1 do
        Prolongate correction from $\delta P_{coarse}$ to $\delta P_{fine}$ ie from Level to Level-1
        solve for $\nabla^2(\delta P) = -R$ at each level using corrected initial guesses
        ie smoothen pressure corrections using red-black SOR(GPU) using corrected
initial guesses
    end for
    solve for $\nabla^2 P = S_{ij}$ using corrected initial guess
    ie smoothen pressure using red-black SOR(GPU) using corrected initial guess
end while
```

Chapter 3

Governing Equation and Numerical methodology

3.1 Governing Equation

The basic governing equation for incompressible laminar flow are as follows

Continuity Equation

$$\nabla \cdot u = 0 \tag{3.1}$$

Momentum Equation

x-momentum equation

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2)}{\partial x} + \frac{\partial (uv)}{\partial y} + \frac{\partial (uw)}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + \rho f x \tag{3.2}$$

y-momentum equation

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (v^2)}{\partial y} + \frac{\partial (vw)}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + \rho f y \tag{3.3}$$

z-momentum equation

$$\frac{\partial w}{\partial t} + \frac{\partial (uw)}{\partial x} + \frac{\partial (vw)}{\partial y} + \frac{\partial (w^2)}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + \rho f z \tag{3.4}$$

3.2 Numerical Methodology for Solving Navier-Stokes Equation

The governing equations have been solved numerically by discretizing them temporally and spatially. Spatial discretization has been performed using the finite volume method with a structured orthogonal mesh. The Finite Volume approach has been used since it is easy to extend on non orthogonal grid. Presently non-staggered (collocated) grid arrangement has been used, where the dependent variables are calculated at the centroid of the finite volume. But this arrangement can produce non-physical oscillations in the pressure field, the so-called checker-board pressure distribution. When central differencing is used to represent both the pressure gradient term in the momentum equations and the cell-face velocity in the continuity equation, it then happens that the velocities depend on pressure at alternate nodes and not on adjacent ones and the pressure too depends on velocities at alternate nodes. This behaviour is called velocity-pressure decoupling, Patankar[14]. To avoid this decoupling, the momentum interpolation method, first proposed by Rhie and Chow [15] has been used.

The collocated grid arrangement where all the dependent variables are defined at the centroid of the cell P has been used. The four neighboring control volume centers have been defined by E, W, N and S, (for the east, west, north and south neighbours respectively). The face center points e, w, n and s are located at the corresponding face centroid of control volume P. We need to calculate surface vectors for each of the faces of the finite volume and also its volume.

Descretization in Finite Volume

Based on the control volume formulation of analytical fluid dynamics, the first step in the Finite Volume Method is to divide the domain into a number of control volumes. The next step is to integrate the differential form of the governing equations over each control volume. Interpolation profiles are then assumed in order to describe the variation of the concerned variable between cell centroids. The resulting equation is called the discretized or discretization equation. In this manner, the discretization equation expresses the conservation principle for the variable inside the control volume.

Continuity

Taking an integral over a finite volume for the continuity equation

$$\int_V (\rho\nabla\cdot\mathbf{u})\,dV = \int_S \rho\mathbf{u}\cdot dS \tag{3.5}$$

$$\int_S \rho\mathbf{u}\cdot dS \approx \sum_{f=e,w,n,s} \rho(\mathbf{u}\cdot S)_f = \sum_f \rho\mathbf{u_f}\cdot S_f$$

where S_f is the surface vector representing the area of the f^{th} cell face and u_f is the velocity defined at the face center *f*. Based on mass conservation the discretized form of the continuity equation can be derived as

$$\sum_f F_f = F_e + F_w + F_n + F_s = 0 \tag{3.6}$$

where the F_f is the outward mass flux through face f,defined by

$$F_f = \rho\mathbf{u_f}\cdot S_f \tag{3.7}$$

Momentum Equation

Unsteady Term

The unsteady term is descretized after integrating over finite volume

$$\frac{\partial}{\partial t}\int_V \rho\mathbf{u}dV \approx \frac{(\rho\mathbf{u}V)_P^{n+1} - (\rho\mathbf{u}V)_P^n}{\Delta t} \approx \rho V_P \frac{(\mathbf{u})_P^{n+1} - (\mathbf{u})_P^n}{\Delta t} \tag{3.8}$$

Convective Term

$$\int_V (\rho\mathbf{u}\nabla\cdot\mathbf{u})\,dV = \int_S \rho\mathbf{u}\,(\mathbf{u}\cdot\mathbf{dS}) \tag{3.9}$$

Summing over all faces

$$\int_S \rho\mathbf{u}\,(\mathbf{u}\cdot\mathbf{dS}) \approx \sum_f \rho\mathbf{u_f}\,(\mathbf{u}\cdot\mathbf{S})_f = \sum_f F_f u_f \tag{3.10}$$

Where $F_f = u_f \cdot S_f$ is the convective flux and u_f is the value of u at the center of the face.

$$\int_S \rho \mathbf{u} \left(\mathbf{u} \cdot \mathbf{dS} \right) \approx F_e u_e + F_w u_w + F_n u_n + F_s u_s \tag{3.11}$$

The value of u at the faces can be determined using various convective schemes.

Convective scheme

First order UPWIND scheme

It has been found that the first-order upwind scheme unconditionally satisfies the boundedness criteria and never yields an oscillatory solution, but it has been found to introduce excessive numerical diffusion. Hence second order upwind scheme has been used in the present study. According to first order UPWIND

$$\phi_f = \begin{cases} \phi_P & \text{if} F_f \geq 0 \\ \\ \phi_{nb} & \text{if} F_f < 0 \end{cases} \tag{3.12}$$

Second order UPWIND scheme

In this scheme the values at the face are mainly calculated from the linear interpolation of the values of the neighbouring points. Calculating for east face for $F_e \geq 0$

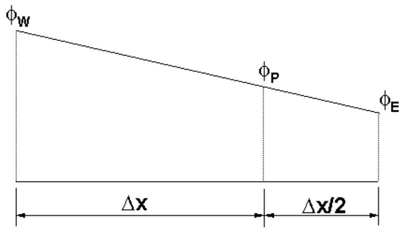

Figure 3.1: 2nd order UPWIND

$$\frac{\phi_e - \phi_P}{\frac{\Delta x}{2}} = \frac{\phi_p - \phi_W}{\Delta x}$$

$$\phi_e = \frac{3}{2}\phi_P - \frac{\phi_W}{2}$$

$$\phi_e = \begin{cases} \frac{3}{2}\phi_P - \frac{\phi_W}{2} & \text{if} F_f \geq 0 \\ \\ \frac{3}{2}\phi_E - \frac{\phi_{EE}}{2} & \text{if} F_f < 0 \end{cases} \tag{3.13}$$

Similarly it can be calculated at all other faces.

Diffusion term

$$-\frac{\mu}{\rho}\int_V \nabla\cdot[\nabla\mathbf{U}] \approx -\frac{\mu}{\rho}\int_S [\nabla\mathbf{U}]\cdot\mathbf{dS} \tag{3.14}$$

$$\approx -\frac{1}{\rho}_P \sum_f [\mu\nabla U]_f \cdot S_f$$

$$\approx -\frac{1}{\rho_P}\sum_f F_{dUf}^{n+1}$$

Pressure term

$$\int_V -\frac{1}{\rho}\frac{\partial P}{\partial x}dV \approx -\frac{1}{\rho_P}\int_V \frac{\partial P}{\partial x} \tag{3.15}$$

$$\approx -\frac{1}{\rho_P}\int_V (\nabla P)\cdot n_x dV$$

$$\approx -\frac{1}{\rho_P}\sum_f P_f^{n+1} S_{fx}$$

The solution of Momentum Equation

A two step strategy is adopted to solve Navier Stokes equation. In the first step the pressure term is dropped and the equation is solved explicitly for fictitious velocity field called mass velocities u^* and v^*. In the second step, the corrected pressure field is found out by using the mass velocity obtained from the first step and by solving the pressure Poisson equation. By enforcing the divergence free condition on the velocity field, the Pressure Poisson equation has been derived from the momentum equations as per Julien et al [10]. This corrected pressure field is used to obtain corrected velocity field. The Descretized form of Navier Stokes equation thus obtained is as follows

$$V_p\frac{U_p^{n+1}-U_p^n}{\Delta t} + \sum_f F_f^n U_f^n + \frac{1}{\rho}\sum_f F_{dUf}^n = -\frac{1}{\rho}\sum_f p_f^{n+1} S_{fx} \tag{3.16}$$

$$V_p\frac{V_p^{n+1}-V_p^n}{\Delta t} + \sum_f F_f^n V_f^n + \frac{1}{\rho}\sum_f F_{dVf}^n = -\frac{1}{\rho}\sum_f p_f^{n+1} S_{fy} \tag{3.17}$$

The Predictor step

Dropping the pressure terms and solving explicitly for mass velocities u^* and v^* as per equations 3.18 and 3.19 given below.

$$V_p \frac{U_p^* - U_p^n}{\Delta t} + \sum_f F_f^n U_f^n + \frac{1}{\rho} \sum_f F_{dUf}^n = 0 \tag{3.18}$$

$$V_p \frac{V_p^* - V_p^n}{\Delta t} + \sum_f F_f^n V_f^n + \frac{1}{\rho} \sum_f F_{dVf}^n = 0 \tag{3.19}$$

The Corrector step

By definition correction

$$U_P' = U_P^{n+1} - U_P^* \tag{3.20}$$

Subtracting equation 3.16 and 3.18 , also equation 3.17 and 3.19

$$V_p \frac{U_p'}{\Delta t} = -\frac{1}{\rho} \sum_f p_f^{n+1} S_{fx} \tag{3.21}$$

$$V_p \frac{V_p'}{\Delta t} = -\frac{1}{\rho} \sum_f p_f^{n+1} S_{fx} \tag{3.22}$$

Here the R.H.S of equation can be better understood by simplifying as per equation 3.23 below

$$-\sum_f p_f S_f \approx -\int_S pdS = -\int_V \nabla p dV \approx -V_P \left(\nabla p\right)_p \tag{3.23}$$

From the above two equations

$$U_P^{'} = -\frac{\Delta t}{\rho} \left(\nabla p^{n+1}\right)_p \tag{3.24}$$

At the face the equation can be written as

$$U_f^{'} = -\frac{\Delta t}{\rho} \left(\nabla p^{n+1}\right)_f \tag{3.25}$$

Also we know that

$$F_f^{n+1} = F_f^n + F_f^{'} \tag{3.26}$$

Imposing continuity equation $\sum_f F_f^{n+1} = 0$ and solving equation 3.26 yields

$$\sum_f F_f^{'} = -\sum_f F_f^* \tag{3.27}$$

substituting equation 3.27 in equation 3.25 we get the Pressure Poisson equation

$$\sum_f \left(\nabla p^{n+1}\right)_f \cdot S_f = \frac{\rho}{\Delta t} \sum_f F_f^* \tag{3.28}$$

$$\frac{\Delta t}{\rho} \nabla^2 P = \nabla \cdot U^* \tag{3.29}$$

3.2.1 Solution Algorithm

As per equation 3.18 and 3.19

$$V_p \frac{U_p^* - U_p^n}{\Delta t} + \sum_f F_f^n U_f^n + \frac{1}{\rho} \sum_f F_{dUf}^n = 0$$

$$V_p \frac{V_p^* - V_p^n}{\Delta t} + \sum_f F_f^n V_f^n + \frac{1}{\rho} \sum_f F_{dVf}^n = 0$$

Solve explicitly for u^* and v^*
Solve pressure poisson equation 3.28

$$\sum_f \left(\nabla p^{n+1}\right)_f \cdot S_f = \frac{\rho}{\Delta t} \sum_f F_f^* \tag{3.30}$$

The pressure poisson equation has been solved implicitly using Gauss Siedel on GPU(Red black) with multigrid . The corrected pressure field is obtained after convergence . Calculating Velocity corrections from corrected pressure field using

$$V_p \frac{U_p'}{\Delta t} = -\frac{1}{\rho} \sum_f p_f^{n+1} S_{fx} \tag{3.31}$$

$$V_p \frac{V_p'}{\Delta t} = -\frac{1}{\rho} \sum_f p_f^{n+1} S_{fx} \tag{3.32}$$

The corrected velocity field is updated

$$U^{n+1} = U^* + U' \tag{3.33}$$

$$V^{n+1} = V^* + V' \tag{3.34}$$

3.3 Reynolds Averaged Navier-Stokes (RANS) Equations

Incompressible Navier Stokes in tensor notation is as follows

$$\rho\frac{\partial u_i}{\partial t} + \rho\frac{\partial(u_i u_j)}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(\tau_{ij}) \tag{3.35}$$

where τ_{ij} is shear stress

$$\tau_{ij} = \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] \tag{3.36}$$

RANS equations for incompressible flows can be derived using the following steps.

- Substitute $u_i = U_i + u_i'$ and $p = P_i + p_i'$
- Take the time average over the resulting equations.
- Solve for the averaged quantities ($\overline{U}$) and simplify using the statistical properties of turbulent fluctuating quantities ($\overline{u'} = 0$) to simplify the resulting equations.
- Use continuity equation to simplify the equations further.

Taking time average of each term mentioned above as follows

Unsteady term

$$\rho\frac{\partial u_i}{\partial t} = \rho\frac{\partial\left(U_i + u_i'\right)}{\partial t} \tag{3.37}$$

By taking time average and using the property of fluctuating components of variables we get

$$\overline{\frac{\partial\left(\rho u_i\right)}{\partial t}} = \overline{\frac{\partial\left(\rho U_i + u_i'\right)}{\partial t}} \tag{3.38}$$

$$= \frac{\partial\left(\rho\overline{U_i} + \overline{u_i}'\right)}{\partial t}$$

$$\overline{\frac{\partial\left(\rho u_i\right)}{\partial t}} = \frac{\partial\left(\rho U_i\right)}{\partial t} \tag{3.39}$$

Convective term

$$\frac{\partial \rho u_j u_i}{\partial x_j} = \frac{\partial \left[\rho \left(U_j + u'_j\right)\left(U_i + u'_i\right)\right]}{\partial x_j} \tag{3.40}$$

Taking time average and simplifying

$$\overline{\frac{\partial \rho u_j u_i}{\partial x_j}} = \overline{\frac{\partial \left[\rho \left(U_j + u'_j\right)\left(U_i + u'_i\right)\right]}{\partial x_j}} \tag{3.41}$$

$$= \overline{\frac{\partial \left[\rho \left(U_j U_i + U_j u'_i + U_i u'_j + u'_j u'_i\right)\right]}{\partial x_j}}$$

$$\overline{\frac{\partial \left(\rho u_j u_i\right)}{\partial x_j}} = \frac{\partial \left(\rho U_j U_i + \rho \overline{u'_j u'_i}\right)}{\partial x_j} \tag{3.42}$$

Pressure Term

$$\frac{\partial \overline{p}}{\partial x_i} = \frac{\partial \overline{(P + p')}}{\partial x_i} \tag{3.43}$$

$$\frac{\partial \overline{p}}{\partial x_i} = \frac{\partial P}{\partial x_i} \tag{3.44}$$

Stress Term

$$\frac{\partial}{\partial x_j}\left[\overline{\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_j}\right)}\right] = \frac{\partial}{\partial x_j}\left[\overline{\mu\left(\frac{\partial \left(U_i + u'_i\right)}{\partial x_j} + \frac{\partial \left(U_j + u'_j\right)}{\partial x_i}\right)}\right] \tag{3.45}$$

$$= \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial \left(\overline{U_i} + \overline{u'_i}\right)}{\partial x_j} + \frac{\partial \left(\overline{U_j} + \overline{u'_j}\right)}{\partial x_i}\right)\right]$$

$$= \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right]$$

$$\frac{\partial}{\partial x_j}\left[\overline{\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_j}\right)}\right] = \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right] \tag{3.46}$$

Hence RANS can be now written in the form

$$\rho\frac{\partial u_i}{\partial t} + \rho\frac{\partial (u_i u_j)}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\left(\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right) - \rho\overline{u'_i u'_j}\right]$$

The term $\rho\overline{u'_i u'_j}$ is known as the Reynolds stress tensor comprising three normal stress components and six shear stress components which are not known as a priori and need to be modeled for the closure of the set of governing equations. It is symmetric, hence the number of unknowns are reduced to six (three normal stress components and three shear stress components).

Boussinesq approximation

$$-\rho\overline{u_i'u_j'} = \mu_t\left[\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right] - \frac{2}{3}\rho\delta_{ij}k \tag{3.47}$$

Substituting above equation provides the closure model for RANS equations. Thus the modeled RANS equations in tensor notation are

$$\frac{\partial}{\partial x_j}\left[\overline{\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_j}\right)}\right] = \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\rho\delta_{ij}k\right] \tag{3.48}$$

There are various closure models that have been developed so far. In the present study the $k - \epsilon$ model which is a two equation model has been used.

3.4 The $k - \epsilon$ model

The two equation models are found to be complete since they can be used to predict properties of a given turbulent flow with no prior knowledge of turbulent structures. The turbulent viscosity μ_t is calculated on the basis of turbulent kinetic energy k and dissipation rate ϵ. Two separate transport equation are used to solve for k and ϵ

k equation

$$\frac{\partial k}{\partial t} + U_j\frac{\partial k}{\partial x_j} = \tau_{ij}\frac{\partial U_i}{\partial x_j} - \epsilon + \frac{1}{\rho}\frac{\partial}{\partial x_j}\left[(\mu + \mu_t/\sigma_k)\frac{\partial k}{\partial x_j}\right] \tag{3.49}$$

ϵ equation

$$\frac{\partial \epsilon}{\partial t} + U_j\frac{\partial \epsilon}{\partial x_j} = C_{\epsilon 1}\frac{\epsilon}{k}\tau_{ij}\frac{\partial U_i}{\partial x_j} - C_{\epsilon 2}\frac{\epsilon^2}{k} + \frac{1}{\rho}\frac{\partial}{\partial x_j}\left[(\mu + \mu_t/\sigma_\epsilon)\frac{\partial \epsilon}{\partial x_j}\right] \tag{3.50}$$

$$\mu_t = \rho C_\mu k^2/\epsilon \tag{3.51}$$

where

$$\tau_{ij} = \frac{\partial}{\partial x_j}\left[\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right] \tag{3.52}$$

Closer coefficients as per Wilcox[16]

$C_{\epsilon 1} = 1.44$ $C_{\epsilon 2} = 1.92$ $C_\mu = 0.09$ $\sigma_k = 1.0$ $\sigma_\epsilon = 1.3$

3.5 The solution algorithm for RANS based turbulence models

- The velocity and pressure field is initialized to zero.
- The turbulence parameters are specified through turbulent intensity and viscosity ratio.
- The finite volume descretization of RANS is similar to that of momentum equation ,following is the descretized form of RANS.

$$V_p \frac{U^* - U_p^n}{\Delta t} + \Sigma F_f^n U_f^n + \frac{1}{\rho} \Sigma F_{duf}^n = V_p \frac{1}{\rho} \left[\frac{\partial(\mu + \mu_t)}{\partial y} \frac{\partial V}{\partial x} - \frac{\partial(\mu + \mu_t)}{\partial x} \frac{\partial V}{\partial y} \right] \tag{3.53}$$

$$V_p \frac{V^* - V_p^n}{\Delta t} + \sum F_f^n V_f^n + \frac{1}{\rho} \sum F_{dvf}^n = V_p \frac{1}{\rho} \left[\frac{\partial(\mu + \mu_t)}{\partial y} \frac{\partial U}{\partial y} - \frac{\partial(\mu + \mu_t)}{\partial x} \frac{\partial U}{\partial x} \right] \tag{3.54}$$

- The term $-\frac{2}{3} \sum k_f S_{fx}$ can be included in the pressure term to get a modified pressure field.
- The extra source terms obtained due to the spatial variation of $(\mu + \mu_t)$ denoted as S_x and S_y are expressed as follows

$$S_x = V_p \frac{1}{\rho} \left[\frac{\partial(\mu + \mu_t)}{\partial y} \frac{\partial V}{\partial x} - \frac{\partial(\mu + \mu_t)}{\partial x} \frac{\partial V}{\partial y} \right] \tag{3.55}$$

$$S_y = V_p \frac{1}{\rho} \left[\frac{\partial(\mu + \mu_t)}{\partial y} \frac{\partial U}{\partial y} - \frac{\partial(\mu + \mu_t)}{\partial x} \frac{\partial U}{\partial x} \right] \tag{3.56}$$

- It has been found that for the current study the contribution of S_x and S_y is negligible.Hence it is has been neglected.
- The equations 4.8 and 4.9 are solved explicitly to find U^* and V^*
- Solve the pressure equation as per 3.28

$$\sum_f \left(\nabla p^{n+1}\right)_f \cdot S_f = \frac{\rho}{\Delta t} \sum_f F_f^* \tag{3.57}$$

- The pressure poisson equation has been solved implicitly using Gauss Siedel on GPU(Red black) with multigrid.
- The corrected pressure field is obtained after convergence.
- Calculating Velocity corrections from corrected pressure field using

$$V_p \frac{U'_p}{\Delta t} = -\frac{1}{\rho} \sum_f p_f^{n+1} S_{fx} \tag{3.58}$$

$$V_p \frac{V'_p}{\Delta t} = -\frac{1}{\rho} \sum_f p_f^{n+1} S_{fx} \tag{3.59}$$

- The corrected velocity field is updated.

$$U^{n+1} = U^* + U' \tag{3.60}$$

$$V^{n+1} = V^* + V' \tag{3.61}$$

- The value of k and ϵ are calculated from their respective transport equation.

Descretized form of k equation

$$V_p \frac{k_p^{n+1} - k_p^n}{\Delta t} + \Sigma F_f^{n+1} k_f^{n+1} + \frac{1}{\rho} \Sigma F_{dkf}^{n+1} + D_k^{n+1} = Pr^{n+1} \tag{3.62}$$

Descretized form of ϵ equation

$$V_p \frac{\epsilon_p^{n+1} - \epsilon_p^n}{\Delta t} + \Sigma F_f^{n+1} \epsilon_f^{n+1} + \frac{1}{\rho} \Sigma F_{d\epsilon f}^{n+1} + D_\epsilon^{n+1} = C_{\epsilon 1} \left(\frac{\epsilon}{k}\right)^n Pr^{n+1} \tag{3.63}$$

Production term

$$Pr = V_p \frac{\mu_t^n}{\rho} \left[\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \frac{\partial U_i}{\partial x_j} \right]$$

D_k is dissipation rate of turbulent K.E

$D_k = \epsilon = C_D \frac{k^{3/2}}{l}$

D_ϵ is destruction rate of ϵ

$D_\epsilon = C_{\epsilon 2} \frac{\epsilon^2}{k}$

- Above equations 3.62 and 3.63 are solved using Gauss siedel and iterating till convergence to find updated values k^{n+1} and ϵ^{n+1}.
- The eddy viscosity is updated using the following equation

$$\mu_t^{n+1} = \rho C_\mu (k^{n+1})^2/\epsilon^{n+1} \tag{3.64}$$

- This updated value of μ_t is used in RANS equations 4.8 and 4.9 at next timestep.
- The problem is solved for steady state.

Source term linearization

- The source term is treated as per Patankar[14].
- For the destruction rate ϵ

$$D_k^{n+1} \approx D_k^n + \left(\frac{\partial D_k}{\partial k}\right)^n \left(k^{n+1} - k^n\right) \tag{3.65}$$

$$\epsilon^{n+1} \approx \epsilon^n + \left(\frac{\partial \epsilon}{\partial k}\right)^n \left(k^{n+1} - k^n\right)$$

$$\epsilon^{n+1} \approx \epsilon^n + \left(\frac{\epsilon}{k}\right)^n \left(k^{n+1} - k^n\right)$$

- Similarly for dissipation term.

$$D_\epsilon^{n+1} \approx D_\epsilon^n + \left(\frac{\partial D_\epsilon}{\partial k}\right)^n \left(\epsilon^{n+1} - \epsilon^n\right) \tag{3.66}$$

$$\approx C_{\epsilon 2} f_2 \left(\frac{\epsilon}{k}\right)^n \epsilon^n + 2C_{\epsilon 2} f_2 \left(\frac{\epsilon}{k}\right)^n \left(\epsilon^{n+1} - \epsilon^n\right)$$

$$\approx C_{\epsilon 2} f_2 \left(\frac{\epsilon}{k}\right)^n \left(2\epsilon^{n+1} - \epsilon^n\right)$$

Chapter 4

Large Eddy Simulation

4.1 Spectral analysis

Turbulence consists of a continous spectrum of scales ranging from the largest scale to the smallest scale. The transfer of energy from one scale to other is called energy cascading. As already mentioned large scale fluctuations have a larger dependence on the flow as compared to the small scale fluctuations and these in turn are mostly independent of Large scale statistics, LES mainly exploits this behaviour. Therefore in LES the large scale fluctuations are resolved and small scale fluctuations are modeled. The figure as per [16]shows the energy spectrum for turbulent flows.

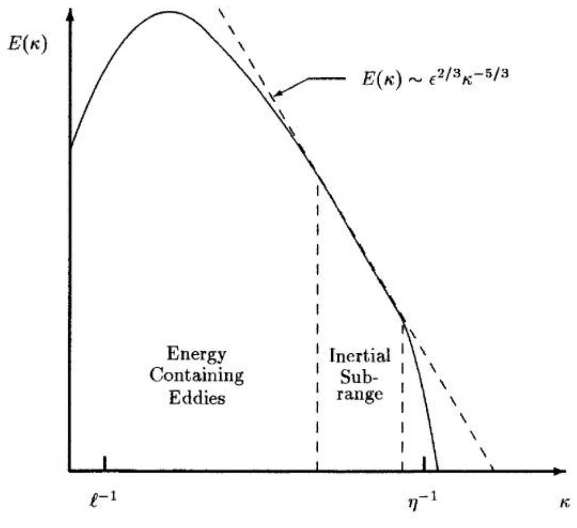

Figure 4.1: Energy spectrum for a turbulent flow

The energy spectrum shows that there is range of eddy sizes between largest and smallest scales for which the energy cascade is independent of the energy containing eddies and direct effects of molecular viscosity. Hence for this range the energy transferred by inertial effects dominates and is called as inertial subrange. Since the largest eddies are directly affected by the boundary conditions and carry most of the Reynolds stress they must be computed while the small scale fluctuations are weaker and contribute less to Reynolds stresses hence they can be modeled. Therefore the fluctuations upto the inertial subrange are resolved and fluctuations below that are modeled.

4.2 Filtering

In order to filter the small scale fluctuations from the flow filters are used. Using the Gaussian filter function through convolution such that the filtered quantity is given by [17]

$$\overline{u}(x,t) = \int \vec{u}(y,t) G_{\Delta x}(x-y)dy \tag{4.1}$$

Any variable $\vec{u}$ can be described as follows

$$\vec{u} = \overline{u} + u' \tag{4.2}$$

where $\overline{u}$ indicates the filtered quantity which is resolved and $u\prime$ indicates the subgrid scaled fluctuation which is modeled. A filter width Δ is defined such that the fluctuations falling above this length scale are termed as large scaled fluctuations and fluctuations below this length scale are termed as subgrid scaled fluctuations. The filter width is generally taken to be $(\Delta_x \Delta_y \Delta_z)^{1/3}$.

Filtered Navier Stokes Equation

After applying the Gaussian filter the Navier Stokes equation takes the form

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{\partial (\overline{u_i}\,\overline{u_j})}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) + T_{ij}\right) \tag{4.3}$$

where T_{ij} is defined as the sub-grid scale tensor which is given by

$$T_{ij} = \overline{u_i}\,\overline{u_j} - \overline{u_i u_j} \tag{4.4}$$

The sub-grid scale tensor is modeled using suitable sub-grid scale model. Out of the various models suggested so far, Smagorinsky-Lilly model [5, 6] has been implelmented in the current study.

4.3 Smagorinsky-lilly model

Smagorinsky-lilly model is the most widely used eddy viscosity model. It was first proposed by Smagorinsky in 1963. Using Smagorinsky-lilly model the subgrid scale term is modeled as follows

$$T_{ij} = (Cs\Delta)^2 |S_{ij}| S_{ij} \tag{4.5}$$

The Smagorinsky's model is a sort of mixing-length assumption in which the eddy viscosity is assumed to be proportional to the subgrid scale characteristic length scale Δ_x . The eddy viscosity μ_t is obtained by assuming that the small scales are in equilibrium, so that energy production and dissipation are in balance. The eddy viscosity is calculated from filtered velocity field as follows

$$\mu_t = \rho(Cs\Delta)^2 |S_{ij}| \tag{4.6}$$

Here $|S_{ij}| = \sqrt{2S_{ij}S_{ij}}$ Where S_{ij} is calculated as follows

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) \tag{4.7}$$

C_s is defined as smagorinsky constant
$\Delta = (\Delta_x \Delta_y \Delta_z)^{1/3}$.

4.4 Brief discussion on Smagorinsky's constant

For accurate solution of the flow it is necessory to ensure that the subgrid model dissipates correct amount of energy from resolved scale to subgrid scale. The Smagorinsky model is known to be excessively dissipative as per Germano et.al.[18]. Also Large-eddy simulations of transition to turbulence in boundary layers done by Elhady et.al [19] show that during the early stages of transition the Smagorinsky model predicts excessive damping of the resolved structures, leading to incorrect growth rates of the initial perturbations. The Smagorinsky's constant (Cs) plays a vital role in accurate modeling of the subgrid term. Cs does not have a universal value, it is found to change from problem to problem. As per Eswaran and Biswas it can range from 0.07 to 0.2. For a turbulent mixing layer a value of Cs=0.17 or Cs=0.1 has been suggested by Vreman et al.[20]. A higher value of Cs is found to cause excessive damping of large-scale fluctuations in the presence of mean shear [21]. Hence in the present work investigations have been done to find the suitable value.

4.5 Inlet boundary conditions

In Reynolds Averaged Navier-Stokes turbulence model the turbulent fluctuations are expressed through the Reynolds stress tensor which is linked to the mean quantities through some turbulence model ($k - \epsilon$). However for LES the goal is to explicitly resolve the turbulent fluctuations and inlet conditions cannot be derived from experimental results due to unsteady and pseudo random nature of the flow. This problem is more important for spatially developing turbulent flows where the boundary or shear layer thickness changes rapidly. Even for stationary turbulent flows, if realistic initial conditions are not prescribed, the establishment of a fully developed turbulence takes unreasonably long execution time (Smirnov et al. [22]). Hence it is necessary to specify realistic turbulent fluctuations at the inlet as the flow downstream is highly dependent on these conditions. The inlet perturbation propagates throughout the domain and helps trigger the turbulence that is to be captured. The fluctuations have been applied as described in Smirnov et al.[22]. The fluctuations are generated by using turbulence intensity I, a pseudo-random number α, and the mean streamwise velocity.

$$U' = V' = I\alpha U_{mean}$$

$$U_{inlet} = U_{mean} + I\alpha U_{mean}$$

The random number α is chosen using normal distribution. The mean is specified as zero whereas the variance is specified as unity. The random number hence generated is between -1 and 1. The turbulent intensity is taken around 10%.

4.6 Solution Algoritm for LES

Solve the problem using $k - \epsilon$ model till steady state is achieved
Take the same velocity and pressure field as initial value for LES

$$V_p \frac{U^* - U_p^n}{\Delta t} \quad + \quad \Sigma F_f^n U_f^n \quad + \quad \frac{1}{\rho} \Sigma F_{duf}^n \quad = \quad 0 \tag{4.8}$$

$$V_p \frac{V^* - V_p^n}{\Delta t} \quad + \quad \sum F_f^n V_f^n \quad + \quad \frac{1}{\rho} \sum F_{dvf}^n \quad = \tag{4.9}$$

The equations 4.8 and 4.9 are solved explicitly to find U^* and V^*
Solve the pressure equation as per 3.28

$$\sum_f \left(\nabla p^{n+1}\right)_f \cdot S_f = \frac{\rho}{\Delta t}\sum_f F_f^* \tag{4.10}$$

The pressure poisson equation has been solved implicitly using Gauss Siedel on GPU(Red black) with multigrid . The corrected pressure field is obtained after convergence . Calculating velocity corrections from corrected pressure field using

$$V_p \frac{U_p'}{\Delta t} = -\frac{1}{\rho}\sum_f p_f^{n+1} S_{fx} \tag{4.11}$$

$$V_p \frac{V_p'}{\Delta t} = -\frac{1}{\rho}\sum_f p_f^{n+1} S_{fx} \tag{4.12}$$

The corrected velocity field is updated

$$U^{n+1} = U^* + U' \tag{4.13}$$

$$V^{n+1} = V^* + V' \tag{4.14}$$

The eddy viscosity is calculated based on Smagorinsky-lilly model

$$\mu_t = \rho (Cs\Delta)^2 |S_{ij}| \tag{4.15}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) \tag{4.16}$$

Here S_{ij} is calculated based on the corrected velocity field
The eddy viscosity calculated is used in the next time step.

Chapter 5

Results and Conclusion

5.1 Laminar Code validation

The laminar code has been validated for lid driven cavity problem. The speed up has been tested separately for GPU and multigrid.

Problem Definition

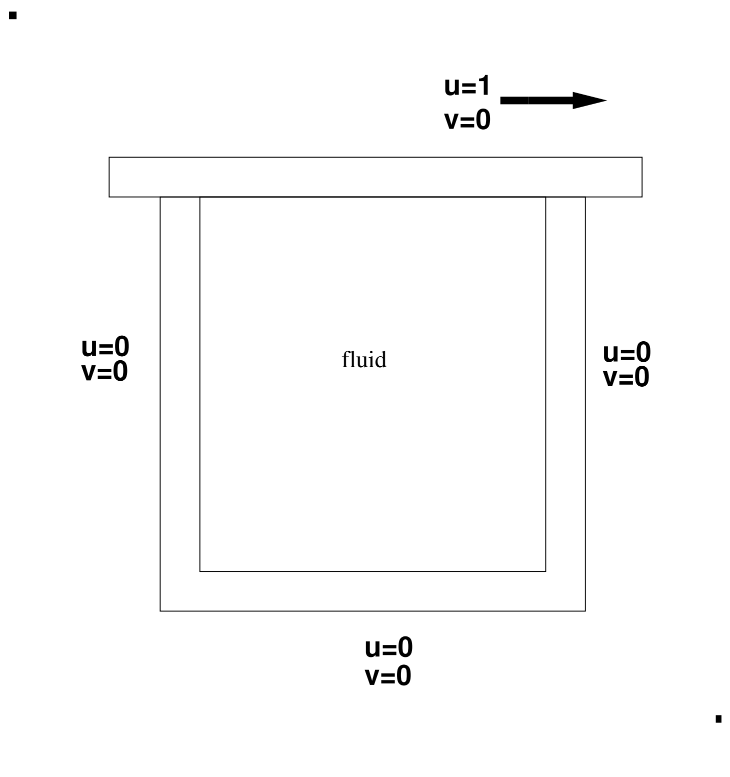

Figure 5.1: Lid driven cavity Problem

The lid driven cavity is a standard benchmark problem for Navier-stokes equations. It has three stationary wall with a moving lid. The lid has a constant velocity and is assumed to be of infinite length.

Boundary conditions

Wall:

$u = 0 \ v = 0 \ \frac{\partial p}{\partial n} = 0$

Lid:

$u = 1 \ v = 0 \ \frac{\partial p}{\partial y} = 0$

Re=100
Convergence criteria $\epsilon = 10^{-7}$
The problem was solved on GPU using red black Gauss siedel algorithm on single grid. The results were tabulated and the speed up was observed.

Grid size	Regular code	GPU	Speedup
$1025X1025$	1.41X 10^{-2}	1.78X 10^{-4}	78X
$513X513$	51.24X 10^{-5}	1.44X 10^{-5}	35X
$257X257$	1.44X 10^{-5}	3.76X 10^{-6}	15 X
$129X129$	7.91X 10^{-6}	1.05X 10^{-6}	7.5 X

Table 5.1: Runtime GPU vs serial code

Code Validation

The centerline velocities of components in horizontal and vertical direction were plotted and compared with results obtained from Young et al.[23]

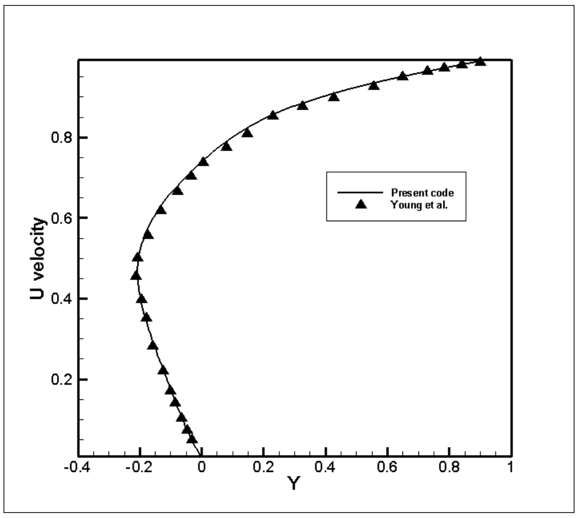

Figure 5.2: U v/s Y Red Black GS

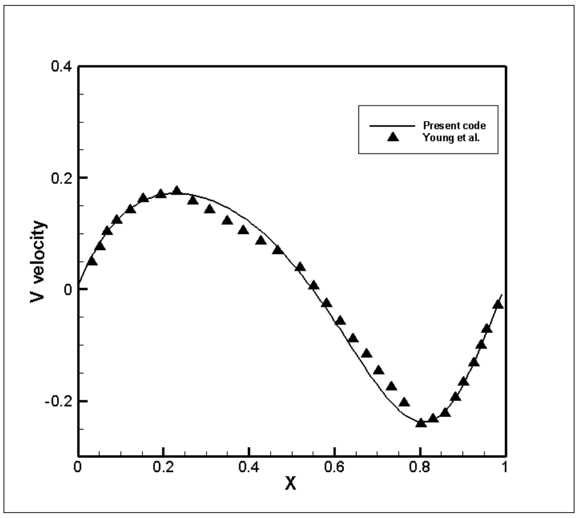

Figure 5.3: V v/s X Red Black GS

The code was extended on multigrid. The results were tabulated and computational effort in terms of work units was compared. For a 512X512 grid the following table shows the variation of work units with levels.

Levels	workunits
1	16234321
2	9532673
3	2324576
4	338936
5	92744
6	33088
7	19496
8	17384
9	17312

Table 5.2: Variation of workunits with levels

Grid size	GPU single grid code	GPU multigrid	Speedup
$1025X1025$	66982312	17656	3793.742X
$257X257$	16232341	17312	937.63 X
$129X129$	5309132	15464	343.322 X
$65X65$	1416243	12232	115.781X

Table 5.3: Comparison GPU single vs Multigrid

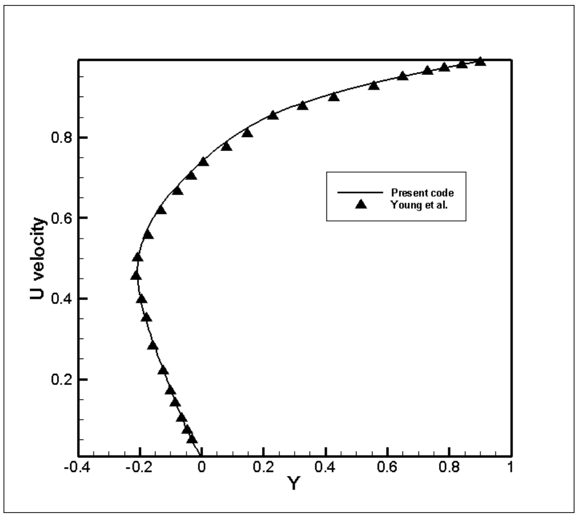

Figure 5.4: U v/s Y Red Black GS on multigrid

5.2 Turbulence code validation

The turbulent code has been validated for 2D mixing layer problem.

Problem Definition

The mixing layer problem consists of two layers of the same fluid with different velocities respectively.

$k - \epsilon$ model

The k-ϵ model has been used to solve the problem.

Boundary conditions

Boundary Conditions
Inlet1:
$u = 1 \qquad v = 0 \qquad k = ko \qquad \epsilon = \epsilon_o \qquad \frac{\partial P}{\partial x} = 0$

Inlet2:
$u = 0.25 \qquad v = 0 \qquad k = ko \qquad \epsilon = \epsilon_o \qquad \frac{\partial P}{\partial x} = 0$

Exit: $\frac{\partial \phi}{\partial x} = 0$ $(\phi = u, v, k, \epsilon)$ $P = 0$

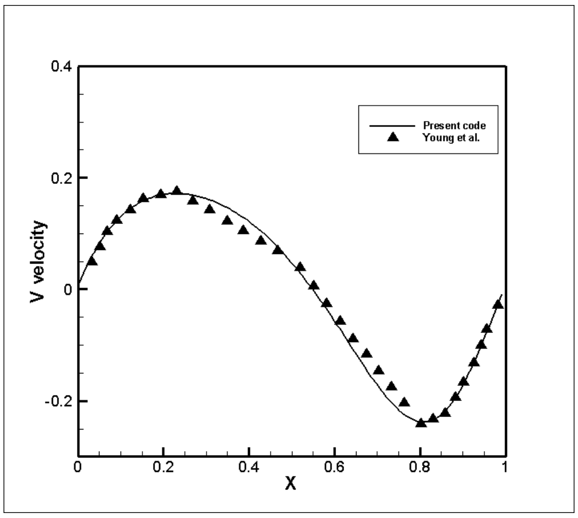

Figure 5.5: V v/s X Red Black GS on multigrid

Top & bottom boundary :

$\frac{\partial \phi}{\partial y} = 0 \qquad (\phi = u, k, \epsilon, P) \qquad v = 0$

k_o and ϵ_o are calculated from I and μ_r

Turbulence parameters

The turbulence constants are specified as per Wilcox[16]

$C_{\epsilon 1} = 1.44 \qquad C_{\epsilon 2} = 1.92 \qquad C_{\mu} = 0.09 \qquad \sigma_k = 1.0 \qquad \sigma_{\epsilon} = 1.3$

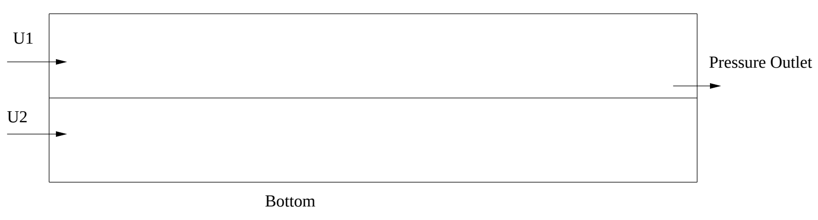

Figure 5.6: Mixing layer problem

The turbulent kinetic energy k and dissipation rate ϵ are calculated from turbulent intensity I and viscosity ratio. For current problem turbulent intensity I=10% and viscosity ratio $\mu_r = 10$.

Code Validation

The mid plane velocity was compared with results obtained by Liepmann and Lauffer. The same problem was solved in ANSYS FLUENT and the results were compared.

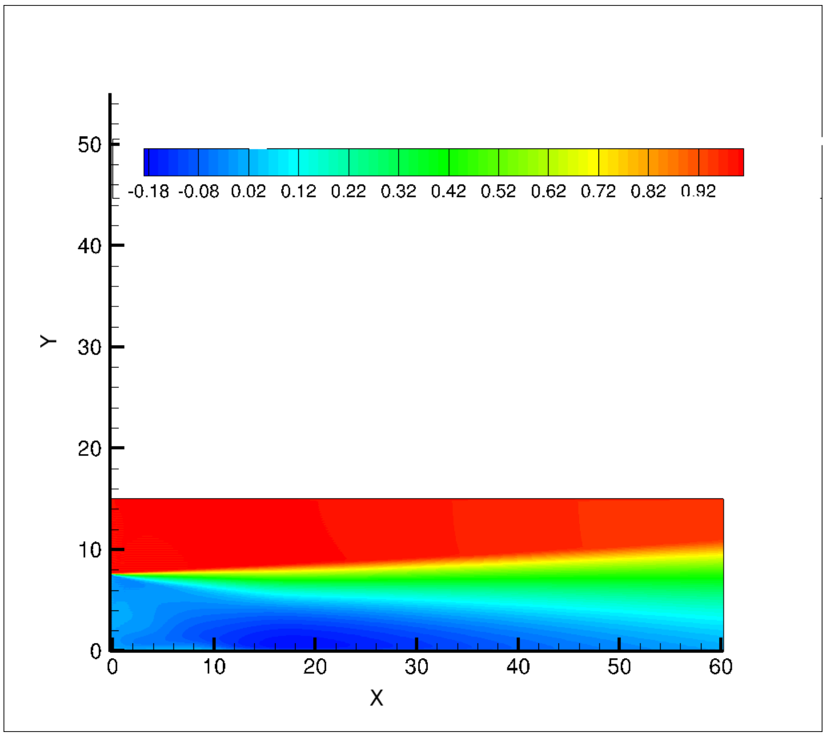

Figure 5.7: Steady state Mixing layer profile obtained

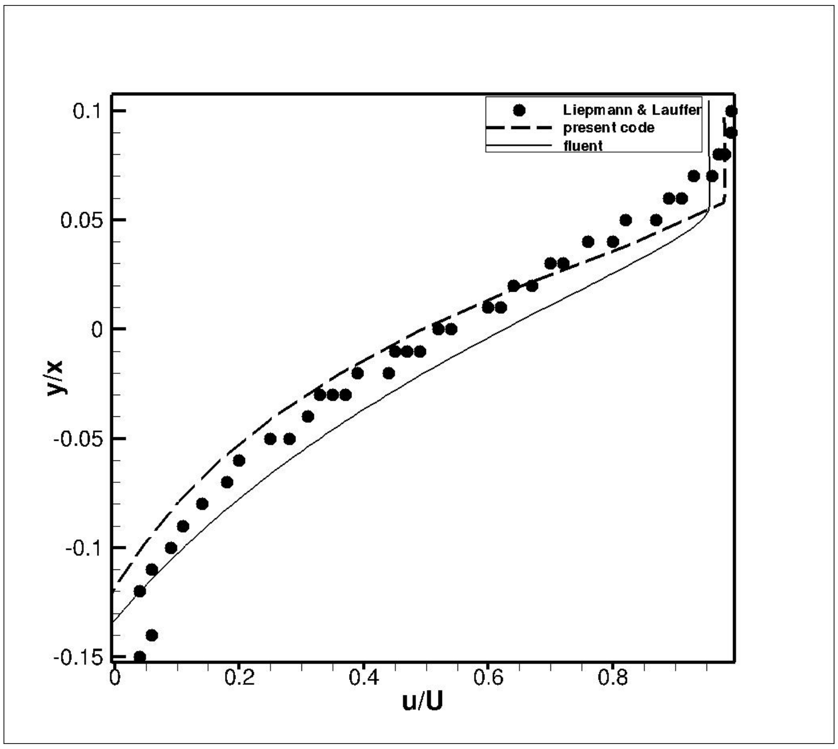

Figure 5.8: Comparing mid plane velocity profile

Grid size	Regular code	GPU	Speedup
$1025X1025$	4.5X 10^{-2}	5X 10^{-4}	100X
$513X513$	2.07X 10^{-3}	4.32X 10^{-5}	48X
$257X257$	2.5X 10^{-4}	1.128X 10^{-5}	22 X
$129X129$	1.9X 10^{-5}	2.15X 10^{-6}	9 X

Table 5.4: GPU vs serial comparison k-ϵ

Grid size	GPU single grid code	GPU multigrid	Speedup
$1025X1025$	113942720	27656	4120X
$257X257$	32849752	27352	1201 X
$129X129$	10448440	25484	410 X
$65X65$	2779032	22232	125X

Table 5.5: GPU single vs multigrid k-ϵ

5.3 LES model

The mixing layer problem was solved using LES.The steady state solution obtained for k-ϵ is given as initial guess for LES. The filter width Δ_x & Δ_y is taken equal to the grid size.

Boundary conditions

LES is sensitive to inlet boundary conditions,hence a suitable fluctuating boundary condition has to be given. As already explained in section 4.5 the velocity inlet profile is superimposition of the fluctuation on the mean profile as shown in figure given below.

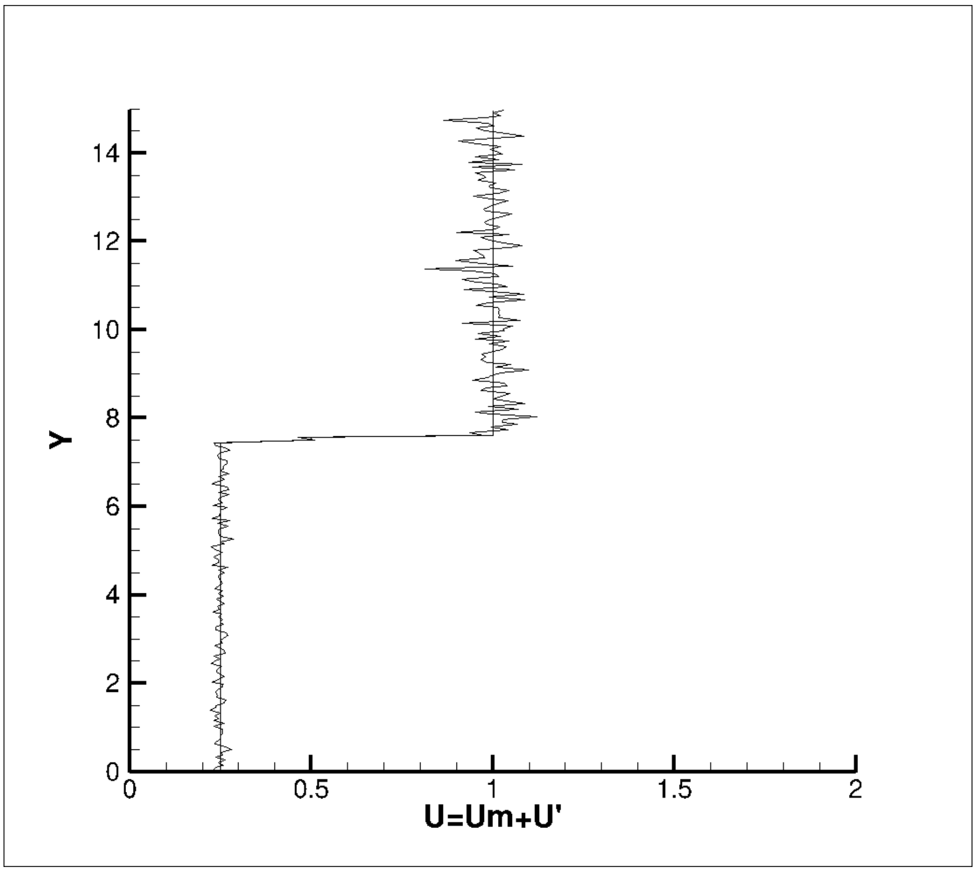

Figure 5.9: Inlet Profile for LES

Determining the length & time scales

Deciding grid size

From the k-ϵ model solving RANS equation $\epsilon_{max} = 0.0570$
For air $\nu = 0.000198$

Kolmogorov length scale is given by

$$\eta = \left(\frac{\nu^3}{\epsilon_{max}}\right)^{1/4}$$

$$\eta = 3.416 \times 10^{-3}$$

For the energy spectrum to fall beyond sub-inertial range,length scale for LES $\approx$ $10\times$ kolmogorov length scale as per Wilcox [16].

$\Delta x = 3.416 \times 10^{-2}$

$$\begin{aligned} 3.416 \times 10^{-2} &= L/N_x \\ N_x &= L/3.416 \times 10^{-2} \\ N_x = 10/3.416 \times 10^{-2} &= 3 \times 10^2 \end{aligned} \tag{5.1}$$

The grid size suitable for the problem is 512×512

.

Deciding timestep

Kolmogorov time scale is calculated

$$\tau = (\nu/\epsilon_{max})^{1/2} \tag{5.2}$$

From the above equation the Kolmogorov time scale is calculated to be of the order of 10^{-3}. Due to the CFL constraint on explicit scheme the timestep is taken as 10^{-4} for the current problem.

LES results

The code was first tested for Cs=0.17. The code was found to relaminarize as shown in figure below

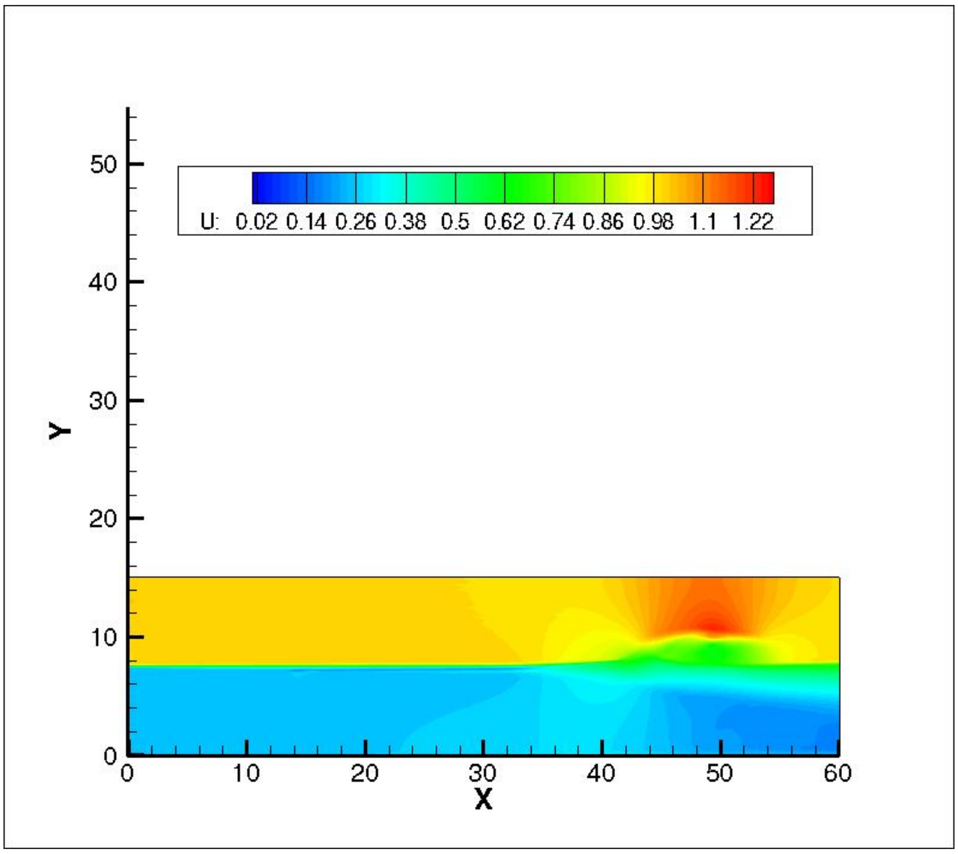

Figure 5.10: X-Velocity contour for Cs=0.17

The code was tested for Cs=0.08 . The fluctuations hence obtained at t=40 sec are as per figure below

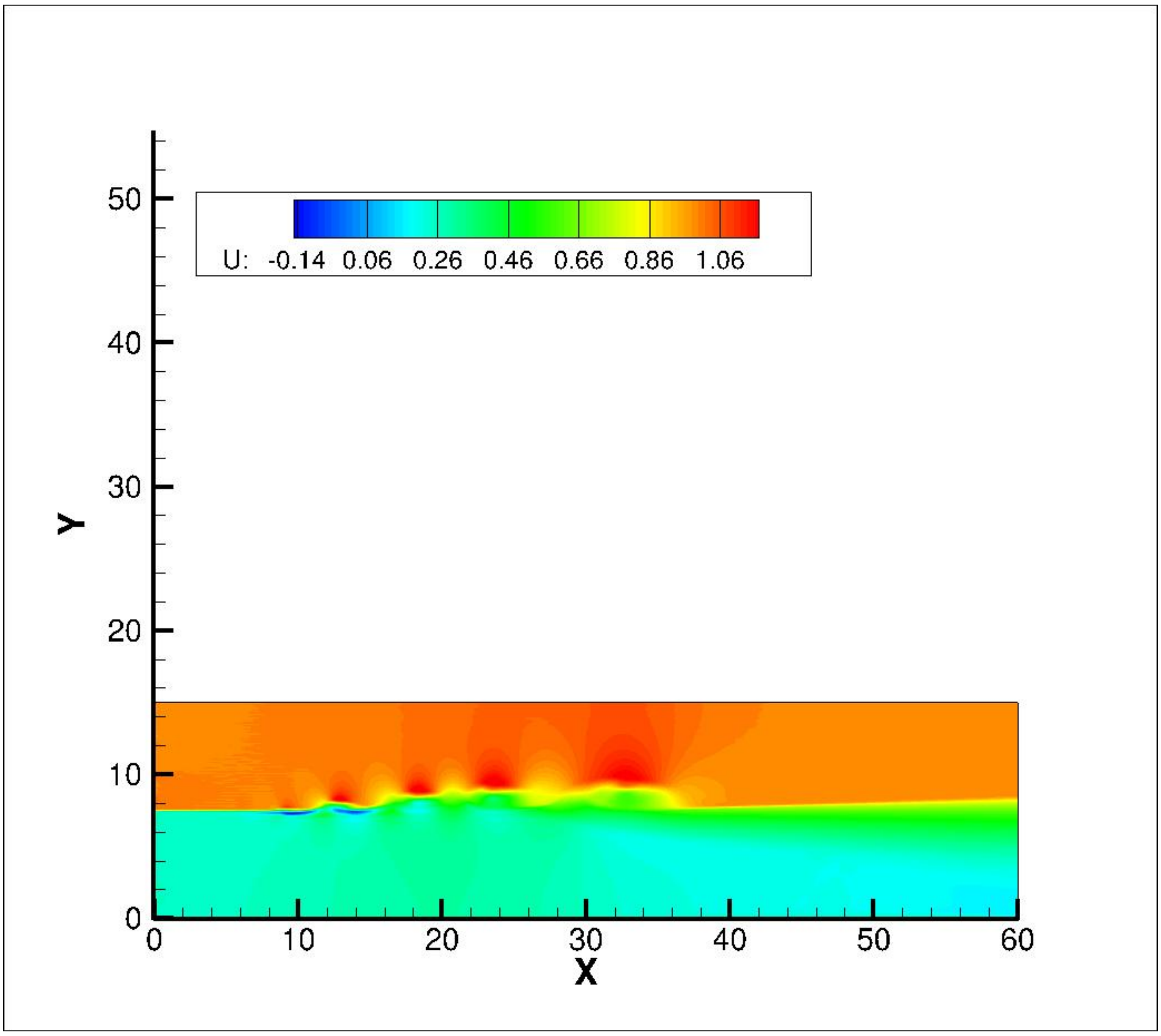

Figure 5.11: X-Velocity contour for Cs=0.08

5.4 Conclusion

The first part of this study mainly consists of developing a GPU based solver which uses multigrid algorithm and which is faster than regular solver by several orders of magnitude. Firstly the solver was used to solve the Navier Stokes equation for laminar case and was successfully validated. The solver was then extended to turbulent case. It was first tested for RANS based model, since RANS based models are computationally less expensive as compared to LES . Also from the modeling perspective there is a striking similarity between RANS based model like k-ϵ and LES. The k-ϵ code was then successfully validated with experimental results obtained by Liepmann & Lauffer.

In LES it was found that the flow downstream is sensitive to inlet conditions hence realistic fluctuating inlet boundary condition is necessory. Also it was found that the Smagorinsky's constant does not have a universal value and a high value results in damping of large scale fluctuations Suitable value of C_s for current 2D mixing layer is 0.08.

5.5 Future Scope

The effect of filter width Δ needs to be tested. The code can be extended to 3D to get better insight into vortex shedding. A general purpose solver based on unstructured grid can be developed. In order to reduce the computational time the code can solved on multiGPU clusters.

References

[1] Baldwin, B.S. and Lomax H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows", *AIAA Paper 78-257.*, (1978)

[2] Cebeci, T. and Smith, "Analysis of Turbulent Boundary Layers", *A.M.O., Academic Press.*, (1984)

[3] Baldwin, B.S. and Bath, T.J., "A One Equation Turbulence Transport Model for High Reynolds Number Wall Bounded Flows", *NASA TM 102847.*, (1990)

[4] Spalart P.R. and Allmaras S.R., "A One-Equation Turbulence Model for Aerodynamic Flows", *30th Aerospace Sciences Meeting and Exhibit, Reno,AIAA-92-0439* (January 1992).

[5] Smagorinsky, Joseph, "General Circulation Experiments with the Primitive Equations", *Monthly Weather Review 91 (3): 99164.* (March 1963).

[6] Lilly,D. K., "A proposed modification of the Germano subgrid-scale closure method", *Physics of Fluids A 4 (3): 633636.* (1992).

[7] S. Pratap Vanka,Aaron F. Shinn,Kirti C . Sahu, "Computational fluid dynamics using graphics processing units:challenges and opportunities", *ASME,IMECE20 11-65260.*(2011).

[8] David B Kirk,Wen-mei W.Hwu, "Programming massively Parallel Processors ", *NVIDIA*(2011).

[9] S. R. Reddy, J. Sebastian, S.M. Miyyadad, R. Banerjee, N. Sivadasan, "VOF based two-phase flow solver on GPU architecture", *ICCMS*(December 2012).

[10] Thibault JC, Senocak I, “CUDA Implementation of a Navier-Stokes solver on Multi- GPU desktop platforms for Incompressible Flows”, *47th AIAA Aerospace Sciences Meeting ,Orlando, Florida*(Jan 2009).

[11] William Briggs,Ven Henson,Steve F. McCormick, “A Multigrid Tutorial”, *Society for Industrial and Applied Mathematics.*

[12] John Tannehill,Dale A.Anderson,Richard H.Pletcher, “Computational Fluid Mechanics and Heat Transfer”,

[13] Pieter Wesseling, “An Introduction to Multigrid Methods”,

[14] Patankar S.V., “Numerical Heat Transfer and Fluid Flow”, *Taylor and Francis*(2004)

[15] Rhie C.M. and Chow W.L., “Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation”, *AIAA Journal,Vol. 21, No. 11, pp. 1525-1532*(1983)

[16] David.C.Wilcox, “Turbulence modeling for CFD”,

[17] Marcel Lesieur, “Turbulence in fluids”

[18] Massimo Germano,Ugo Piomelli,Pervez Moin,William H Cabol, “A dynamic subgrid-scale eddy viscosity model”, *Physics of fluids A 3, 1760*(1991)

[19] Nabil M. ElHady, Thomas A. Zang, and Ugo Piomelli, “Application of the dynamic subgridscale model to axisymmetric transitional boundary layer at high speed”, *Physics of fluids 6, 1299*(1994)

[20] Bert Vreman,Bernard Geurts,Hans Kuerten, “Large Eddy simulation of the turbulent mixing layer”, *J. Fluid Mech 339, 357-390*(1997)

[21] , “Fluent User guide 6.3”,

[22] A. Smirnov, S.Shi, I. Celik, “Random Flow Generation Technique for Large Eddy Simulations and Particle-Dynamics Modeling”, *Transactions of the ASME-I-Journal of Fluids Engineering 123.2 359-371.*(2001)

[23] D. L. Young, Y. H. Liu, T. I. Eldho, “A combined BEM-FEM model for the velocity- vorticity formulation of the Navier-Stokes equations in three dimensions”, *Engineering Analysis with Boundary Elements 24.*(2000)

[24] A. F. Shinn, S. P. Vanka, and W. W. Hwu., “Direct numerical simulation of turbulent ow in a square duct using a graphics processing unit (GPU)”, *In 40th AIAA Fluid Dynamics Conference.*(2010)

[25] Launder, Brian Edward, and Dudley Brian Spalding, “Lectures in Mathematical Models of Turbulence”, *Academic press*(1972)