A Dissertation

entitled

Fourth order Multi-Time-Stepping Adams-Bashforth (MTSAB) scheme for NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulation (BASS) Code

by

Vasanth Allampalli

Submitted to the Graduate Faculty as partial fulfillment of
the requirements for the Doctor of Philosophy Degree in Engineering

_____
Dr. Ray Hixon, Committee Chair

_____
Dr. A.A.Afjeh, Committee Member

_____
Dr. Ng, Tsun-Ming (Terry), Committee Member

_____
Dr. Masiulaniec, K Cyril, Committee Member

_____
Dr. Ashok Kumar, Committee Member

_____
Dr. Patricia Komuniecki, Dean
College of Graduate Studies

The University of Toledo

May 2010

An Abstract of

Fourth Order Explicit Multi-Time-Stepping Adams-Bashforth (MTSAB) scheme for NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulation (BASS) Code

By

Vasanth Allampalli

Submitted to the Graduate faculty as a partial fulfillment of the requirements for the Doctors of Philosophy Degree in Engineering

The University of Toledo

May 2010

By using a Multi-Time-Stepping Adams-Bashforth (MTSAB) scheme, different regions of the computational grid can march with different time steps based on their local stable time steps and on local minimum length and time scales. This can save computational time for problems with large range of length and time scales. In this work, a fourth order, automated MTSAB scheme was developed for NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulation Code (BASS) code. BASS code solves the Navier-Stokes equations on structured multi-block grids. The automated MTSAB scheme, during the run assigns time steps to grid blocks, based on local stability and accuracy. The scheme automatically changes the time steps during the run as required. In this work, two automatic block cutting algorithms were also developed. The first

block cutting algorithm cuts the existing grid blocks during the run, to minimize the number of points at the smallest time steps. The second block cutting algorithm cuts the grid blocks to maximize the parallel efficiency of the scheme. This scheme was tested on CAA workshop problems, highly nonlinear flows (Transonic flows) with grid motion, and viscous flow cases. Results from these cases and the speed gains from using the MTSAB scheme are presented.

To my wife and my family.

Acknowledgements

I would like to thank Dr. Hixon for his great support and advice throughout my pursuit of the degree. This work would not have been possible without his guidance. I would also like to thank him for supporting me financially.

This work was supported by the Subsonic Fixed Wing Project of the NASA Fundamental Aeronautics Program, under Task NNC07E125T-0. I would like to thank Dr. Edmane Envia, the technical monitor for this effort.

I express my gratitude to my other committee members, Dr. A.A.Afjeh, Dr. Ng, Tsun-Ming (Terry), Dr. Masiulaniec, K Cyril and Dr. Ashok Kumar for their comments and suggestions.

I would like to thank my lab mates Adrian Sescu, Daniel Ingraham and Nima Mansouri, for their support and friendship. Adrian deserves special thanks for helping me with running the BASS code on the clusters. I wish a great future for them.

I am indebted to my wife, Prachi, our dog (Jack) and my family for providing me motivation and encouragement at every step during this pursuit and being with me through thick and thin.

Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Introduction

The field of Computational Aeroacoustics (CAA) is focused on the accurate simulation of unsteady flow and noise [1, 2]. To achieve this goal, highly accurate spatial differencing schemes have been developed (e.g., Refs. [3-7]), along with optimized time marching schemes (e.g., Refs. [8-10]). To validate these schemes, a range of benchmark validation problems have been specified, and solutions made available [11-14].

As the CAA schemes have increased in capability, the validation problems have increased in complexity, incorporating realistic nonlinear flows about complex geometries. For example the optimized spatial differencing and time marching incorporated in CAA methods are designed to propagate accurately unsteady flow phenomena through non-uniform grids wrapped about complex geometries such as the stators in a turbojet engine. Experience has shown that the increased accuracy from the spatial differencing schemes is always beneficial, even near flow discontinuities.

In a time marching scheme for an unsteady flow problem, there are two time steps of interest. The first is the largest time step that can be taken while retaining an accurate,

unsteady solution (the accuracy limit). The second time step of interest is the largest time step that can be taken while retaining a stable calculation (the stability limit).

In general, there are two types of time marching schemes used: explicit time marching schemes and implicit time marching schemes. A good basic understanding of these approaches is given in [15]. For an explicit time marching scheme, the inviscid stability limit is directly related to the minimum time required for the fastest propagating wave to move from one grid point to the next (the CFL condition). If the time step taken is larger than the limit imposed by the stability limit, the time marching scheme will quickly go unstable. The use of explicit time marching schemes can thus result in an excessive number of time steps resolving the solution due to stability limit imposed on the time step size. The upside of an explicit time marching is that, for a given time step, the amount of computational work required is less as compared to implicit schemes. The implicit schemes may require multiple iterations for each time step.

For most implicit schemes, stability can be maintained for large values of time steps. Some implicit schemes are unconditionally stable. The large time steps are accompanied with large errors and hence large time steps cannot be used to achieve high accuracy solution for an unsteady problem. In other words, the desired accuracy puts a limit on the time step size for an unsteady problem [15]. An example of application of an implicit scheme for a time accurate problem can be found in [16].

The classification of time marching schemes is shown in Figure 1-1. This figure does not show different kinds of implicit time marching schemes as the current work is focused on the use of explicit time marching schemes. From this point onwards, explicit time marching schemes will be just referred to as time marching schemes.

1.2 Motivation

In a time marching scheme using a single time step size throughout the computational domain, such as a Runge-Kutta time marching scheme or an Adams-Bashforth scheme, the smallest grid spacing determines the time step for the entire calculation, regardless of whether the resulting temporal resolution is necessary for solution accuracy. The explanation of the workings of Runge-Kutta scheme and Adams-Bashforth scheme are explained later in this chapter.

A popular explicit Runge-Kutta scheme is the classical four-stage fourth-order scheme, which has an inviscid stability limit of CFL = 2.83. In the past, several researchers have optimized Runge-Kutta schemes to increase their stability and/or accuracy. For steady-state calculations, Jameson [17] developed a five-stage second-order scheme, which has a stability limit of CFL = 4.0 but low accuracy for time marching. Hu et al. [8] introduced the low-dispersion and dissipation Runge-Kutta (LDDRK) schemes, which were optimized for accuracy to the stability limit. The most popular of these schemes is the two-step fourth-order RK56 scheme, which has a stability limit of CFL = 2.85. Recently, Calvo et al. [10] introduced a six-stage fourth-order Runge-Kutta scheme that is partially optimized for accuracy and partially optimized for a large stability limit. Allampalli et al. developed new High-Accuracy Large-Step Explicit Runge-Kutta (HALE-RK) schemes [18], which have an inviscid stability limit of CFL = 4.9-5.7 while obtaining higher accuracy than the classical Runge-Kutta fourth-order scheme.

An inherent assumption in the optimization of Runge-Kutta schemes for stability is that the unsteady flow dynamics at the length and time scales associated with the smallest

grid spacing is not important to the accuracy of the overall solution. But for problems of practical interest such as Large Eddy Simulation and Broadband Noise calculations, the length and time scales associated with smallest grid spacing is important in the study of flow physics. Because of this, the high CFL offered by some of the optimized time marching scheme cannot be made use of.

One way to save computational time, while retaining time accuracy is to use different time steps in different regions on the computational domain. Tam et al. [19] developed optimized Adams-Bashforth schemes to use multi-time-stepping in the domain. In these methods the change in the time step from one region of the grid to the neighboring region is hardwired to a factor of two. Shen et al. [20] used multi-time-stepping for solving CAA problems, including the numerical simulation of the jet screech phenomena.

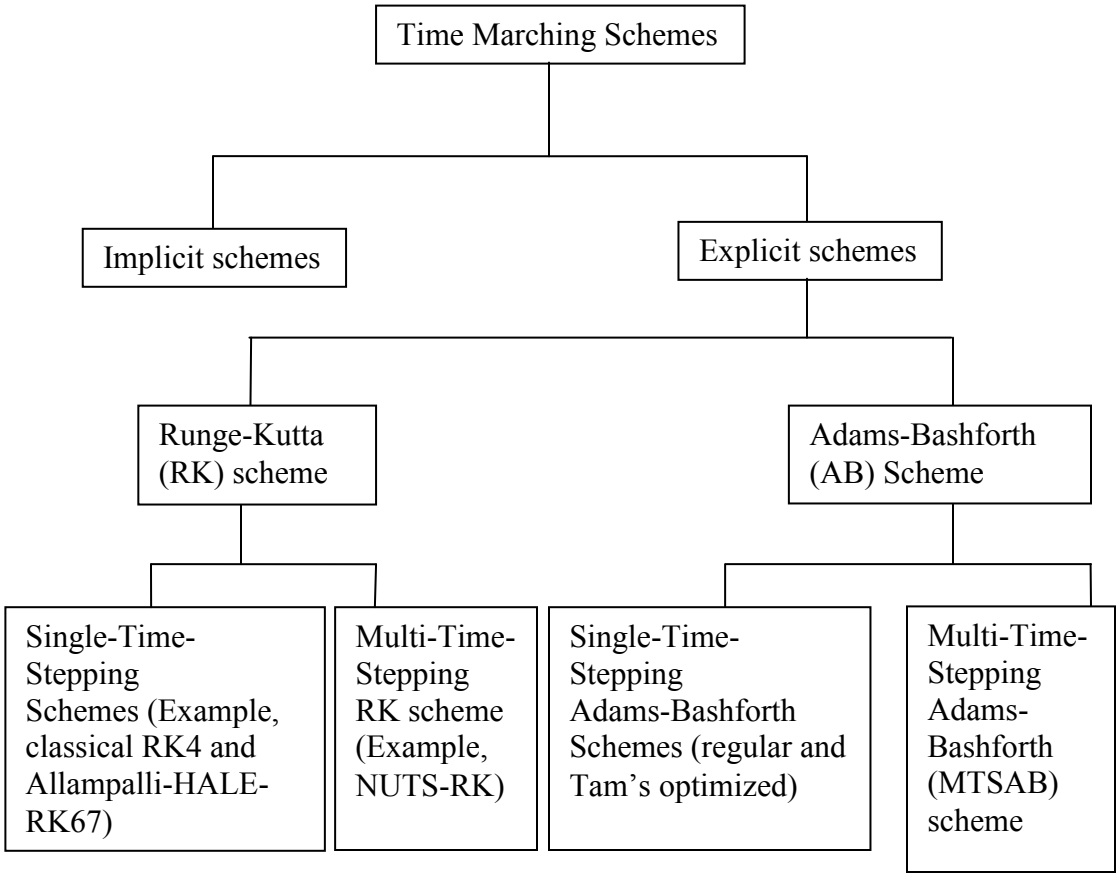


Figure 1-1 Classification of the time marching schemes

Recently Liu et al. [21] developed Non-Uniform Time-Step Runge-Kutta schemes for CAA.

Multi-time stepping can be more generally applicable for complex geometries and grids, if the implementation is automated. The motivation behind current work was to automate the implementation of the new Multi-Time-Stepping Adams-Bashforth scheme and its optimization, to achieve maximum speed, while retaining fourth-order accuracy.

1.3 Objective

The objective of this work was to develop a new fourth order Multi-Time-Stepping Adams-Bashforth (MTSAB) scheme, and to implement and validate this scheme for NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulation (BASS) code. BASS code is 3-D, curvilinear, Navier-Stokes solver [22].

The MTSAB scheme uses different time steps in different regions of the grid, based on local stability and accuracy. Two block cutting algorithms were also developed to achieve maximum speed.

After implementing the scheme in BASS code, it was tested on CAA Benchmark problems, transonic flows over airfoils and viscous flow problems. The scheme was also extended to solve problems with grid motion. A plunging airfoil case was solved for validation.

1.4 Linear Analysis of Numerical Schemes

In order to analyze the performance of a time-marching numerical method, a linear model equation for inviscid wave propagation is used:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{1.1}$$

A simple-harmonic solution to the equation is assumed:

$$u = e^{i(kx-\omega t)} \tag{1.2}$$

where,

$$\omega = kc \tag{1.3}$$

This solution represents a simple-harmonic wave propagating at a speed of *c* in the positive x-direction. In applying a numerical scheme to solve Eq. (1.1), the spatial and temporal derivatives are replaced by numerical approximations, each of which impacts the accuracy of the solution. In the case of a propagating wave, the errors can be classed as *dispersion* (a change in the propagation speed of the wave) and *dissipation* (a change in the amplitude of the wave). The spatial derivative is considered first. The analytic result for the spatial derivative is:

$$\left.\frac{\partial u}{\partial x}\right|_{j,analytical} = iku_j = \frac{i(k\Delta x)}{\Delta x}u_j \tag{1.4}$$

A finite-difference spatial derivative at grid point j on a uniform grid can be written as:

$$\left.\frac{\partial u}{\partial x}\right|_{j,numerical} = \frac{1}{\Delta x}\sum_{n=-M}^{N} a_n u_{j+n} \tag{1.5}$$

If the spatial derivative is formulated as a central difference, the dissipation errors can be eliminated. The resulting spatial derivative can be written as:

$$\left.\frac{\partial u}{\partial x}\right|_{j,numerical} = \frac{1}{\Delta x}\sum_{n=1}^{N} a_n(u_{j+n-} u_{j-n}) \tag{1.6}$$

Substituting the assumed solution into the central difference approximation, the numerical spatial derivative is obtained:

$$\left.\frac{\partial u}{\partial x}\right|_{j,numerical} = \frac{i}{\Delta x}\left[\sum_{n=1}^{N} a_n \sin(nk\Delta x)\right] u_j \tag{1.7}$$

Eq. (1.7) can be written as

$$\left.\frac{\partial u}{\partial x}\right|_{j,numerical} = \frac{i(k\Delta x)}{\Delta x}\left[\frac{\sum_{n=1}^{N} a_n \sin(nk\Delta x)}{k\Delta x}\right] u_j \tag{1.8}$$

Defining the *numerical wavenumber* as:

$$(k\Delta x)^* = \sum_{n=1}^{N} a_n \sin\,(nk\Delta x) \tag{1.9}$$

Eq. (1.8) becomes:

$$\left.\frac{\partial u}{\partial x}\right|_{j,numerical} = \frac{i(k\Delta x)}{\Delta x}\left[\frac{(k\Delta x)^*}{(k\Delta x)}\right] u_j = \left[\frac{(k\Delta x)^*}{(k\Delta x)}\right]\left.\frac{\partial u}{\partial x}\right|_{j,analytic} \tag{1.10}$$

Due to the Nyquist limit, the highest-wavenumber wave that can be resolved on a uniform grid has two grid points per wavelength. This limits the maximum value of the physical wavenumber that may be resolved in a numerical calculation to:

$$k_{max} = \frac{2\pi}{\lambda_{min}} = \frac{2\pi}{2\Delta x} = \frac{\pi}{\Delta x} \tag{1.11}$$

Figure 1-2 shows the numerical wavenumber of several finite-differencing schemes, and are compared to the exact result. The differences in accuracy of the various schemes can be seen; in particular, it can be seen that each scheme has a different maximum value for the numerical wavenumber. Table 1.1 lists the maximum numerical wavenumber from each scheme. Using the development thus far, the numerical equivalent of Eq. (1.1) can be written for this assumed solution as:

$$\frac{\partial u}{\partial t} = F(u) = -\frac{i}{\Delta x}[c(k\Delta x)^*]u \tag{1.12}$$

Using Eq. (1.3) to define a *numerical frequency*:

$$\omega^* = \frac{c(k\Delta x)^*}{\Delta x} \tag{1.13}$$

Equation (1.12) can be rewritten as:

$$\frac{\partial u}{\partial t} = F(u) = -i\omega^* u \tag{1.14}$$

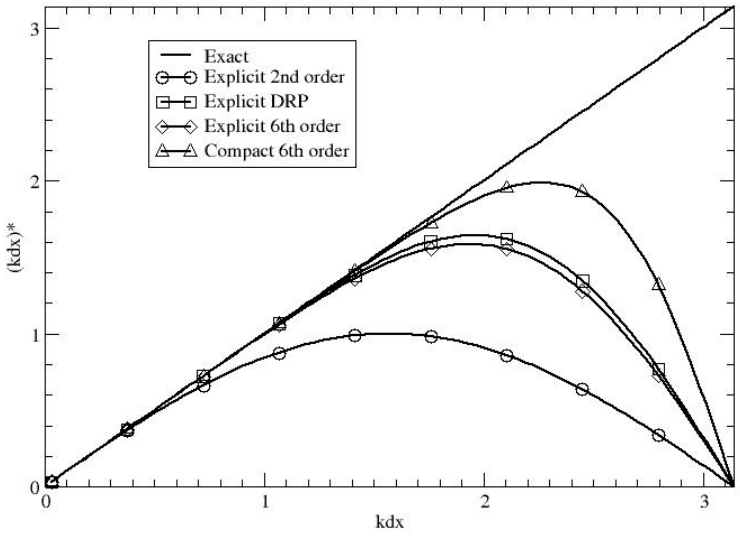


Figure 1-2 Numerical Wavenumbers for Spatial Differencing Methods

Differencing Scheme	Maximum (kΔx)*
Explicit 2nd order	1.0
Explicit DRP	1.644
Explicit 6th order	1.586
Compact 6th order	1.989

Table 1.1: Maximum Value of Numerical Wavenumber for Spatial Differencing Methods

To integrate Eq. (1.14) in time, a Runge-Kutta scheme or an Adams-Bashforth time marching scheme can be used. These two time marching schemes are discussed in the next section.

1.2.1 Runge-Kutta Scheme

For an unknown vector u a classical four-stage fourth-order Runge-Kutta scheme can be written as:

$$
\begin{aligned}
u_0 &= u^n \\
k_1 &= \Delta t\, F(u_0, t) \\
k_2 &= \Delta t\, F(u_0 + \tfrac{1}{2}k_1, t + \tfrac{\Delta t}{2}) \\
k_3 &= \Delta t\, F(u_0 + \tfrac{1}{2}k_2, t + \tfrac{\Delta t}{2}) \\
k_4 &= \Delta t\, F(u_0 + k_3, t + \Delta t) \\
u^{n+1} &= u^n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
\tag{1.15}
$$

The scheme requires a solution (u^n) at the current time. Intermediate solutions are calculated at different stages and final solution (u^{n+1}) at the next time level is obtained at the end of the last stage. The intermediate solutions obtained are non-physical and

contain errors. The errors from all the stages cancel out to give a final fourth order solution at the next time level. Substituting in the function from Eq. (1.14), the fourth-order Runge-Kutta scheme gives:

$$u^{n+1} = \left(1 + (i\omega^*\Delta t) + \frac{1}{2}(i\omega^*\Delta t)^2 + \frac{1}{6}(i\omega^*\Delta t)^3 + \frac{1}{24}(i\omega^*\Delta t)^4\right)u^n \quad (1.16)$$

The relative error magnitude of the time marching can be evaluated as:

$$Error = \left|\left(\frac{u^{n+1}}{u^n}\right) - e^{-i(\omega^*\Delta t)}\right| \quad (1.17)$$

For stability, the amplitude of the propagating wave should not increase:

$$\left|\frac{u^{n+1}}{u^n}\right| \leq 1 \quad (1.18)$$

Figure 1-3 and Figure 1-4 show the error magnitude and amplitude results for the fourth order Runge-Kutta scheme.

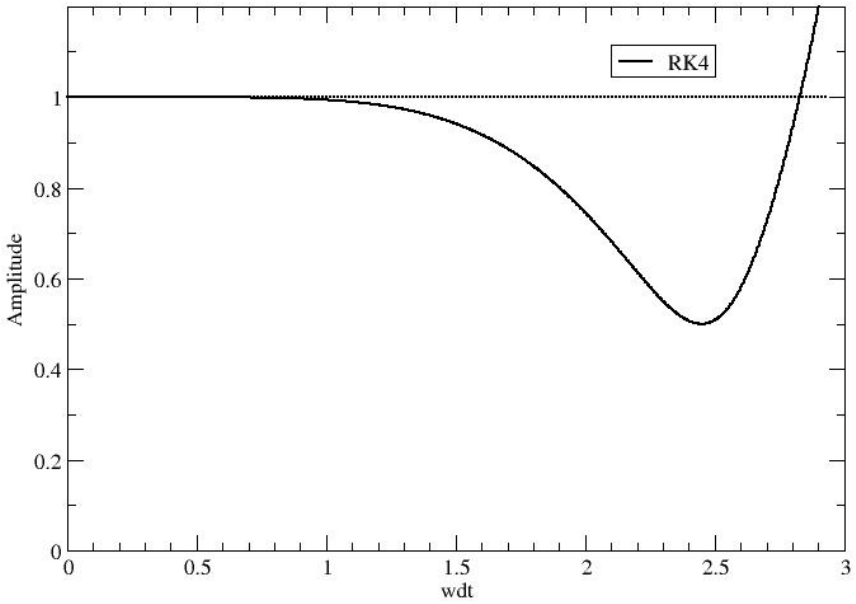


Figure 1-3: Amplitude Performance of Classical Runge-Kutta Scheme

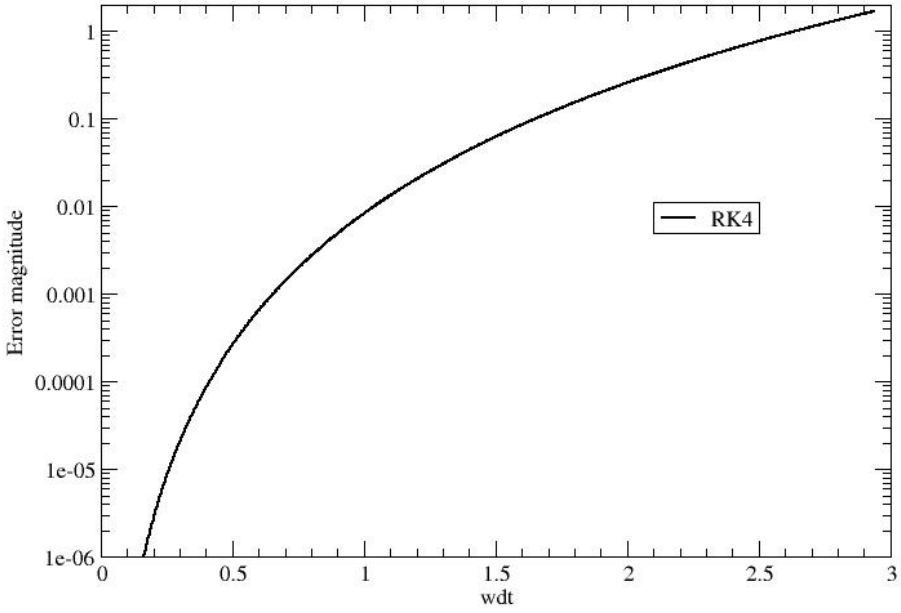


Figure 1-4: Error Magnitude of Classical Runge-Kutta Scheme.

1.2.2 Standard Adams-Bashforth scheme

The fourth order Adams-Bashforth scheme is a four level scheme. Suppose u is the unknown vector. The time axis is divided into a uniform grid with time step Δt. It is assumed that the values of u and $\frac{\partial u}{\partial t}$ are known at time level *n, n -1, n – 2, and n – 3*.

To advance to the next time level, 4-level, fourth order finite difference approximation given by Eq. (1.19) is used.

$$u^{n+1} = u^n + \Delta t \sum_{j=0}^{3} b_j \frac{\partial u^{n-j}}{\partial t} \tag{1.19}$$

The last term on the right side of Eq. (1.19) may be regarded as a weighted average of the time derivatives at the last 4 mesh points in time. There are four coefficients, namely, b_0, b_1, b_2, and b_3. On applying Laplace transform to Eq. (1.19), we need to generalize the equation to one with a continuous variable. The result is

$$u(t + \Delta t) = u(t) + \Delta t \sum_{j=0}^{3} b_j \frac{\partial u(t - j\,\Delta t)}{\partial t} \tag{1.20}$$

Eq. (1.20) reduces to Eq. (1.19) by using $t = n\Delta t$. On applying the shifting theorem for Laplace transform to (1.20), Eq. (1.21) is obtained.

$$\tilde{u} e^{-i\omega\Delta t} = \tilde{u} + \Delta t \left(\sum_{j=0}^{3} b_j \; e^{-ij\omega\Delta t} \right) \frac{\partial \tilde{u}}{\partial t} \tag{1.21}$$

Thus,

$$-i \frac{i(e^{-i\omega\Delta t} - 1)}{\Delta t \left(\sum_{j=0}^{3} b_j \; e^{-ij\omega\Delta t} \right)} \tilde{u} = \frac{\partial \tilde{u}}{\partial t} \tag{1.22}$$

The Laplace transform of the time derivative of u is $i\omega u$. Thus by comparing the two sides of equation (1.22), the quantity,

$$\omega^*\Delta t = \frac{i(e^{-i\omega\Delta t} - 1)}{\Delta t\left(\sum_{j=0}^{3} b_j \; e^{-ij\omega\Delta t}\right)} \tag{1.23}$$

It is easily seen from Eq. (1.23) that the relationship between and $\omega^*\Delta t$ and $\omega\Delta t$ is not one to one. In other words, there are multiple solutions for $\omega^*\Delta t$. One of these solutions is physical (gives actual solution) and the remaining solutions are spurious. These spurious solutions could cause numerical instability and introduce errors in the solution. So it is necessary to find out about their behavior. Let us rewrite Eq. (1.23) in the form below:

$$b_3 z^4 + b_2 z^3 + b_1 z^2 + \left(b_0 + \frac{i}{\omega^*\Delta t}\right) z - \frac{i}{\omega^*\Delta t} = 0 \tag{1.24}$$

In Eq. (1.24) $z = e^{i\omega\Delta t}$. Thus, given $\omega^*\Delta t$ there are four roots of $e^{i\omega\Delta t}$ and hence four values of $\omega\Delta t$. The values of the coefficients of a fourth order Adams –Bashforth scheme are:

$$b_0 = \frac{55}{24} \;, b_1 = \frac{-59}{24} \;, b_2 = \frac{37}{24}, b_3 = \frac{-9}{24} \tag{1.25}$$

For the values of the coefficients specified in Eq. (1.25), the values of the four roots of $\omega\Delta t$ as functions of $\omega^*\Delta t$ are found. The real and imaginary part of these roots, over the range $0 \leq \omega^*\Delta t \leq \pi$ are plotted in Figures 1-5 and 1-6. In the range $\omega^*\Delta t \leq 0.42$, the imaginary parts of all the four roots are negative. Recall that the solution has time dependence of the form $e^{\omega\Delta t}$. If the imaginary part of the root is negative, the solution is damped in time. However, for $\omega^*\Delta t > 0.42$, one spurious root has positive imaginary

part. The solution corresponding to this root will grow in time leading to numerical instability. On examining the real parts of the four roots, it is seen that one of the roots gives $\omega^*\Delta t \sim \omega\Delta t$ over the range of $\omega^*\Delta t \leq 0.42$. This is the desired root. The magnitude of the imaginary component of this root is very small and the spurious roots are damped below $\omega^*\Delta t = 0.42$.

Since the numerical frequency is the product of the physical wave speed and the numerical wavenumber of the spatial differencing method, the maximum value of the numerical frequency depends on maximum value of the numerical wavenumber of the spatial differencing scheme. The maximum values of numerical wavenumber for different spatial differencing scheme are given in Table 1.1. Thus, spatial differencing schemes that have a higher maximum numerical wavenumbers result in higher maximum values of the numerical frequency. Because of this, maximum allowable time step for stability is lower for spatial differencing schemes with higher maximum numerical wavenumbers. For example, from Table 1.1 it is expected that the use of DRP scheme instead of a second order central scheme will result in a time step that is approximately 1.644 times lower due to a higher value of maximum numerical wavenumber.

In this work, $\omega^*\Delta t = 0.31$ or less was used. For this value of $\omega^*\Delta t$, $Im(\omega\Delta t)$ of the desired root is 4.3×10^{-4}. This automatically guarantees numerical stability and negligible numerical damping.

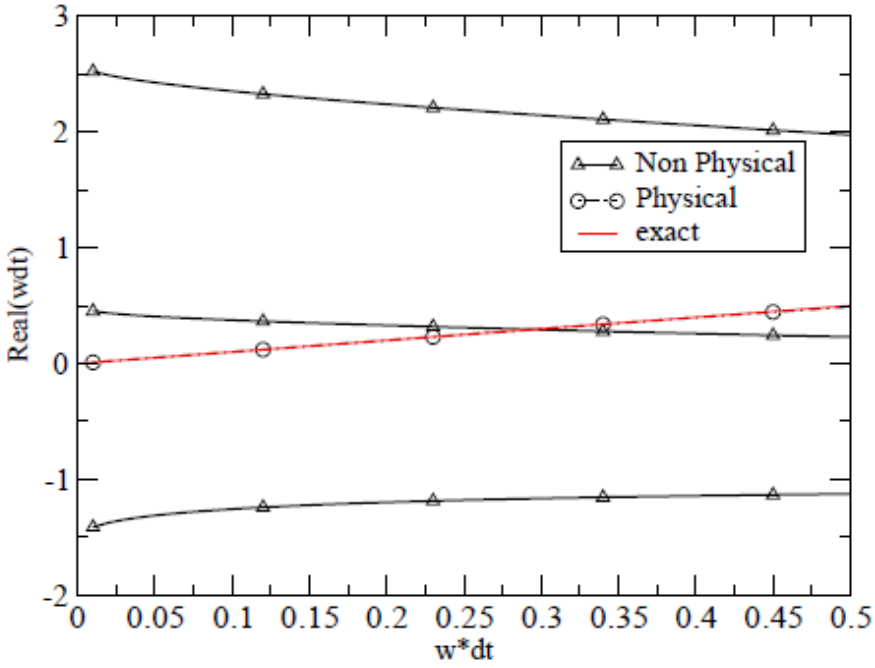


Figure 1-5: Real $\omega\Delta t$ $vs.$ $\omega^*\Delta t$ for Adams-Bashforth scheme

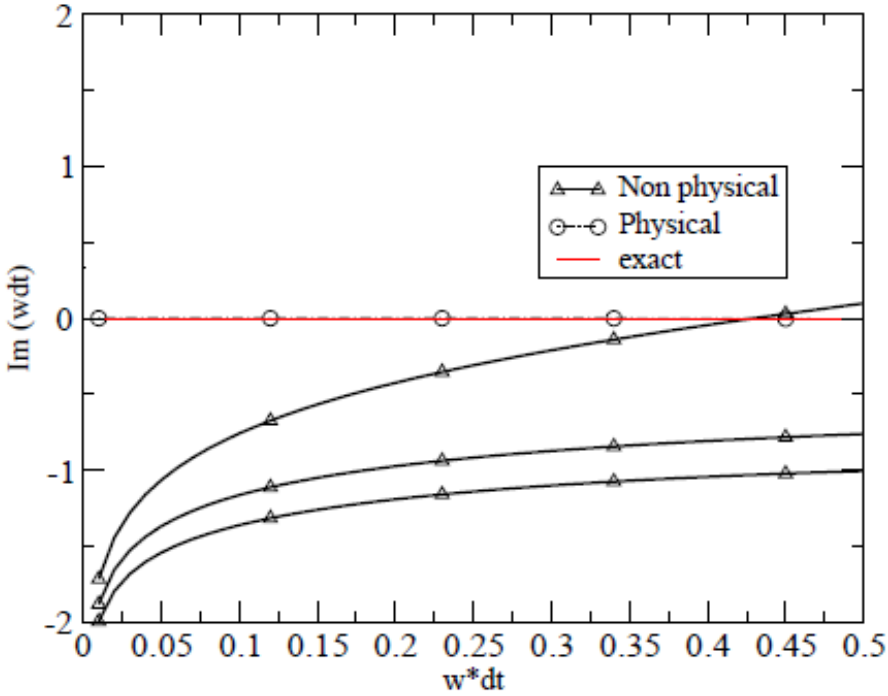


Figure 1-6: Real $\omega\Delta t$ $vs.$ $\omega^*\Delta t$ for Adams-Bashforth scheme

1.2.3 Generalized form of the Adams Bashforth scheme

Before moving to MTSAB scheme it is important to discuss the generalized form of Adams-Bashforth scheme. The Adams-Bashforth scheme given by Eq. (1.19) is the standard form of the Adams-Bashforth scheme. In this form of the Adams-Bashforth scheme,

$$\Delta t_0 = \Delta t_{-1} = \Delta t_{-2} = \Delta t_{-3} = \Delta t \tag{1.26}$$

When,

$$\Delta t_0 \neq \Delta t_{-1} \neq \Delta t_{-2} \neq \Delta t_{-3} \tag{1.27}$$

the generalized form of the Adams-Bashforth scheme in Eq. (1.28) is obtained.

$$u^{n+1} = u^n + \sum_{j=0}^{3} \Delta t_{-j} b_j \frac{\partial u^{n-j}}{\partial t} \tag{1.28}$$

The coefficients b_j can again be calculated from Taylor series for fourth order accuracy and can be written as:

$$\begin{aligned} b_0 &= f(\Delta t_0, \Delta t_{-1}, \Delta t_{-2}, \Delta t_{-3}) \\ b_1 &= f(\Delta t_0, \Delta t_{-1}, \Delta t_{-2}, \Delta t_{-3}) \\ b_2 &= f(\Delta t_0, \Delta t_{-1}, \Delta t_{-2}, \Delta t_{-3}) \\ b_3 &= f(\Delta t_0, \Delta t_{-1}, \Delta t_{-2}, \Delta t_{-3}) \end{aligned} \tag{1.29}$$

Figure 1-7 gives an illustration of the standard and the generalized Adams-Bashforth scheme. The generalized form of the Adams-Bashforth is the key in the implementation of the Multi-Time-Stepping of the Adams-Bashforth scheme. Let the current time be t. By varying Δt_0 in the generalized Adams-Bashforth scheme, the data in the right hand side of the Eq. (1.28) can be used to calculate u^{n+1} for any time between t and $t + \Delta t_0$,

by just recalculating the coefficients. This cannot be done for a Runge-Kutta scheme because all the time derivatives (Eq. (1.15)) at the intermediate stages have to be recalculated again to get u^{n+1} for a different Δt_0. The solutions at intermediate times (stages), if available, cannot be made use of because they are non-physical or spurious.

When two adjacent grid blocks are marching in time with different time steps, by only recalculating the values of coefficients in Eq. (1.28) (for any Δt_0), data that a neighboring block needs at the block interface calculated. The complete method of implementation of the Multi-Time Stepping Adams-Bashforth scheme is explained in detail in the next chapter.

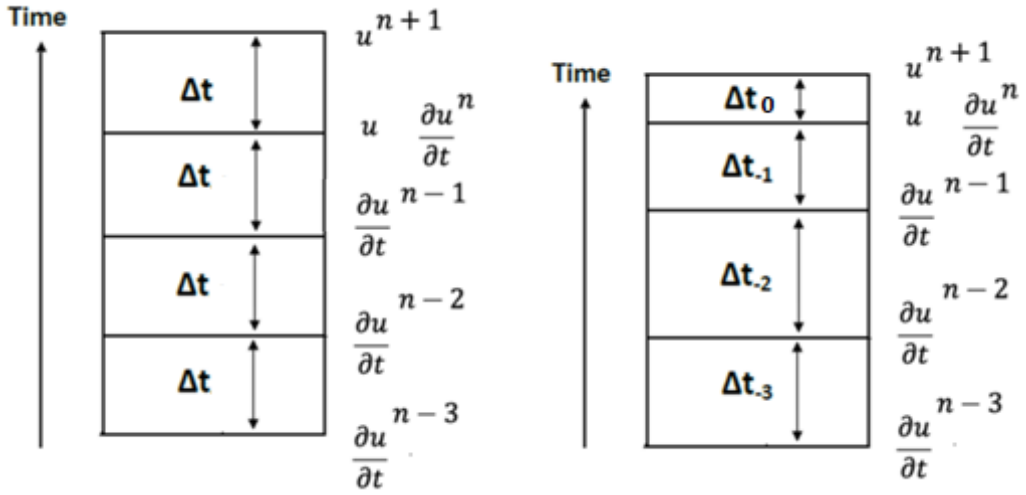


Figure 1-7 Illustration of the standard (left) and the generalized (right) Adams-Bashforth Scheme

Chapter 2

Multi-Time-Stepping Adams Bashforth (MTSAB) Scheme

2.1 Introduction

As already mentioned in the previous chapter, using a Multi-Time-Stepping Adams Bashforth (MTSAB) Scheme, different regions or grid blocks of a multi-block grid can march in time with different time steps. The criteria for choosing the time steps in different regions of the computational domain and the implementation of the scheme are explained in this chapter, with the help of a 1-D grid.

Figure 2-1 shows a 1-D grid, with two grid blocks. The smallest grid spacing is in Grid Block 1 and the grid gradually stretches into Grid Block 2. The first step in using the MTSAB scheme is to calculate the stable time steps that can be used in each of the two grid blocks. The CFL condition is used to calculate the stable time steps. Let the smallest grid spacing in Grid Blocks 1 and 2 be $\Delta x_{\min 1}$ and $\Delta x_{\min 2}$ respectively. The CFL condition given by Eq. (2.1) is used to calculate the stable time step for any Grid block *n* in the 1-D grid example.

$$c\frac{\Delta t_{stable\,n}}{\Delta x_{\min n}} \leq 0.19 \tag{2.1}$$

In Eq. (2.1), c is the fastest wave speed in the calculation. Let the stable time steps calculated for the Grid blocks of the 1-D grid be $\Delta t_{stable\,1}$ and $\Delta t_{stable\,2}$ respectively. It can be seen from Eq. (2.1) that since $\Delta x_{\min 2}$ is greater than $\Delta x_{\min 1}$, $\Delta t_{stable\,2}$ is greater than $\Delta t_{stable\,1}$.

When Runge-Kutta scheme or a single step Adams-Bashforth scheme is used, a single global time step (Δt_{global}) is used for all the regions or blocks of the computational domain. The global time step is the largest stable time step that can be used for the entire domain. It is calculated as the minimum of the stable time steps calculated for all grid blocks in the computational domain. For the two block 1-D grid example, the global stable time step is calculated as:

$$\Delta t_{global} = Min.(\Delta t_{stable\,1}, \Delta t_{stable\,2}) \tag{2.2}$$

Since, $\Delta t_{stable\,2} > \Delta t_{stable\,1}$,

$$\Delta t_{global} = \Delta t_{stable\,1} \tag{2.3}$$

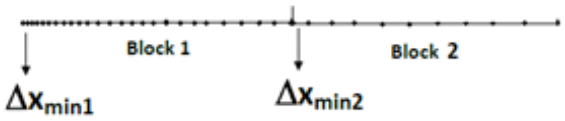


Figure 2-1 1D grid with two grid blocks

Once the global time step is calculated, the grid blocks are assigned different time steps for multi-time stepping. In this work, for any Grid block n, the time step (Δt_n) is calculated as,

$$\Delta t_n = (2^{m-1})\Delta t_{global} \tag{2.4}$$

The maximum value of *m* in Eq. (2.4) is chosen such that,

$$\Delta t_n \leq \Delta t_{stable\ n} \tag{2.5}$$

For the 1-D grid example, the time steps for the Grid blocks are calculated as,

$$\Delta t_1 = \Delta t_{global}$$

$$\Delta t_2 = (2^{m-1})\Delta t_{global} \tag{2.6}$$

The value *m* is chosen such that,

$$\Delta t_2 \leq \Delta t_{stable\ 2} \tag{2.7}$$

In the implementation of the MTSAB scheme, the grid blocks of multi-block grid are assigned different "level" values. The level for a block is the value *m* chosen for that block. In the 1-D grid example, Grid block 1 is at level 1. If the value of *m* chosen for block 2 was 3, then Grid block 2 is at level 3 and so on.

For a computational grid with 6 levels, the time steps for the levels are given as:

$$\begin{aligned}
\Delta t_{level\ 1} &= \Delta t_{global} \\
\Delta t_{level\ 2} &= 2 \times \Delta t_{global} \\
\Delta t_{level\ 3} &= 4 \times \Delta t_{global} \\
\Delta t_{level\ 4} &= 8 \times \Delta t_{global} \\
\Delta t_{level\ 5} &= 16 \times \Delta t_{global} \\
\Delta t_{level\ 6} &= 32 \times \Delta t_{global}
\end{aligned} \tag{2.8}$$

Once the time steps (or levels) are chosen for different grid blocks in the domain, MTSAB scheme can be implemented.

2.2 Starting problem with the Adams-Bashforth scheme

It can be seen from Eq. (1.19), the fourth order Adams-Bashforth scheme need the values of time derivatives from three previous time steps. Therefore generally, when using an Adams-Bashforth scheme the problem cannot be started with just an initial condition for the flow variables.

A Runge-Kutta scheme as shown in Eq. (1.15) can be just started with an initial condition. Therefore, for the first three steps Runge-Kutta scheme is used. The time derivatives from the first three steps are stored after which the Adams-Bashforth scheme can be used.

2.3 The Concept of Buffer Blocks

In this work DRP (Dispersion Relation Preserving) scheme [4] was used for spatial differentiation. The DRP scheme is a seven point stencil. To calculate spatial derivative using the DRP scheme at an interior point requires three data points on each side of that point. The scheme uses one sided stencils at inflow, outflow and at wall boundaries. Consider a point j located in the interior of the computational domain (away from inflow, outflow and wall boundaries) and at the interface of two grid blocks. The data required to calculate for spatial derivative at a given point j, is shown in Figure 2-2. Both the grid blocks share this point and therefore, the spatial derivative is calculated for point j in both the grid blocks.

In a multi-block grid, like in this example, the grid points near the block interfaces (number of points depends on how big the scheme stencil is) need data from the neighboring blocks to calculate spatial derivative. For each of the Grid blocks, blocks

defined as “Buffer blocks” are used to store data from the neighboring block, to calculate the spatial derivative at the block interface. In Figure 2-2, Buffer blocks shown by red lines, store data obtained from the neighboring blocks and enable the calculation of spatial differentiation at point. In BASS code, buffer blocks are generated between the blocks when the grid blocks are on different processors. For the implementation of the MTSAB scheme, the buffer blocks are also generated between adjacent blocks marching with different time steps. Buffer blocks enable the synchronization of the blocks marching with different time steps.

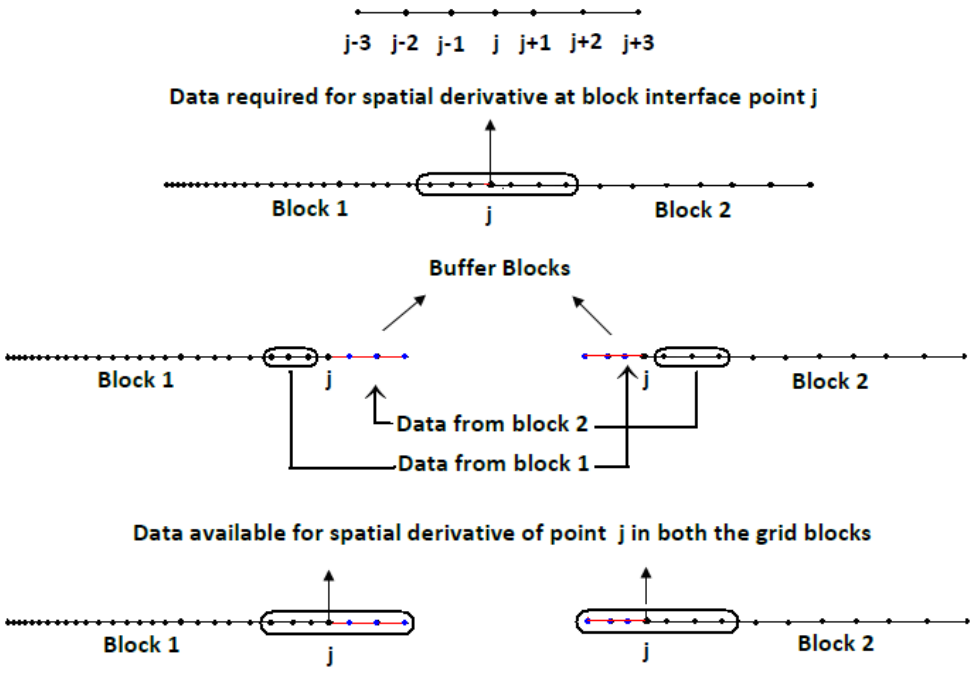


Figure 2-2 Illustrations of Buffer Blocks

2.4 Method of Implementation of the MTSAB Scheme

In this section, the implementation of the Multi Time Step Adams-Bashforth (MTSAB) scheme is explained for same the 1-D grid with two grid blocks. Let the Global time step be Δt. The time step for Grid block 1 is equal to Δt (level 1) and the time step for Grid block 2 is equal to $2 \times \Delta t$ (level 2).

As explained earlier, Buffer blocks are used to store data needed from the neighboring blocks. Figures 2-3a shows the data that is stored in grid blocks and the buffer blocks. Data is sent from Grid blocks to Buffer blocks when the Grid blocks that send and receive data from these buffer blocks are at the same time level.

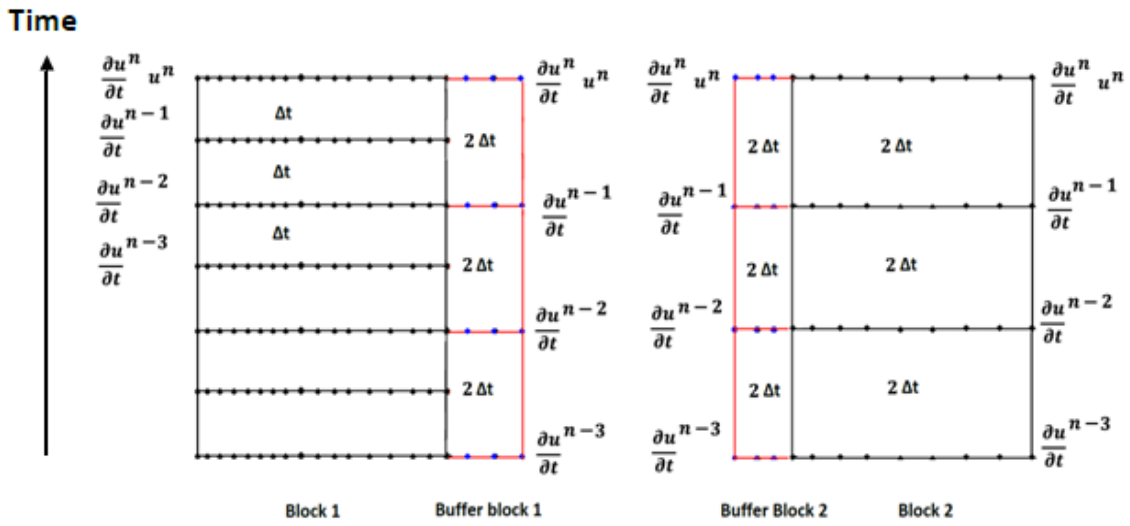
Since Grid block 1 is marching with a time step Δt, it needs buffer data at every increment of Δt to calculate the spatial derivative at block interface. But its buffer block (Buffer block1) receives data from Grid block 2 (which is marching with $2 \times \Delta t$), only at every other Δt. Only at every other time step, grid blocks (Grid block 1 and Grid block 2) sending and receiving data from this buffer block are at the same time level.

Figures 2-3b and 2-3c show how Grid block 1 uses it buffer block to calculate data needed at the block interface, at times when its Buffer block does not receive data from Grid block 2.

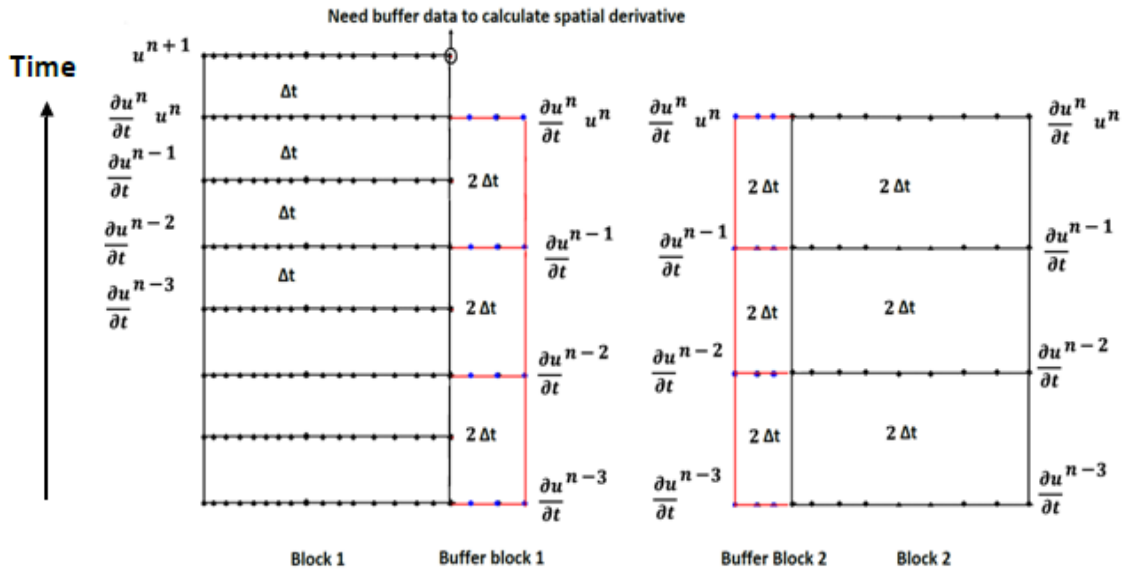
The horizontal lines are the Grid blocks and the Buffer blocks at different times. The vertical distance between the lines is a measure of time steps. In figure 2-3a all the blocks have sufficient data to march to the next time using the Adams-Bashforth formula. In Figure 2-3b Grid block 1 marches in time with a time step Δt, using the Adams-Bashforth scheme. At this new time level, Grid Block 1 needs data (three data points on the right side) at the interface to calculate the spatial derivative at that point. This data is

obtained, when Buffer block 1 marches in time with a time step Δt, using the generalized Adams-Bashforth scheme (because the time steps used in the formula are not equal). This is shown in Figure 2-3c. In other words, Buffer block 1, which stores data from Grid Block 2, calculates data required by Grid block 1 (at the block interface) when this data is not directly calculated and sent by Grid block 2.

(a)



(b)



(c)

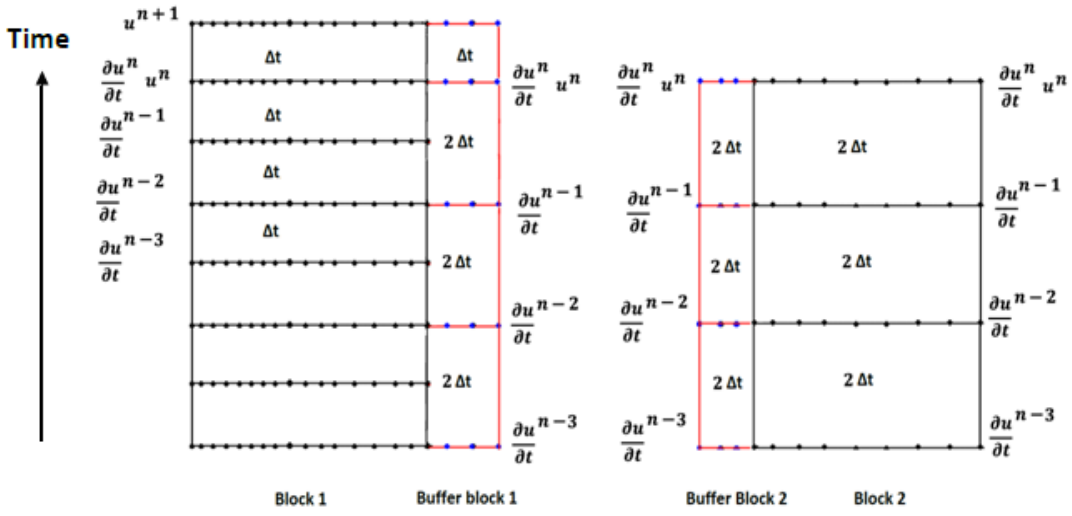


Figure 2-3: Pictures show the synchronization of blocks (marching with different time steps), using buffer blocks.

Chapter 3

Automated MTSAB scheme for BASS Code

3.1 Cartesian Coordinate Equations

BASS code is NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulation code [22]. The code solves the Navier-Stokes equation, which is written in Cartesian coordinates as:

$$\frac{\partial Q}{\partial t}+\frac{\partial (E-E_V)}{\partial x}+\frac{\partial (F-F_V)}{\partial y}+\frac{\partial (G-G_V)}{\partial z}=D(Q) \tag{3.1}$$

where,

$$Q=\begin{Bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ E_{tot}\end{Bmatrix} \tag{3.2}$$

The inviscid fluxes are:

$$E=\begin{Bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ u(E_{tot}+p)\end{Bmatrix} \tag{3.3}$$

$$F = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho uw \\ v(E_{tot} + p) \end{Bmatrix} \tag{3.4}$$

$$G = \begin{Bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E_{tot} + p) \end{Bmatrix} \tag{3.5}$$

The viscous flux terms are:

$$E_v = \begin{Bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{Bmatrix} \tag{3.6}$$

$$F_v = \begin{Bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{xy} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y \end{Bmatrix} \tag{3.7}$$

$$G_v = \begin{Bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \end{Bmatrix} \tag{3.8}$$

The stresses are defined as Newtonian, using Stokes hypothesis:

$$\tau_{xx} = \mu\left(2\frac{\partial u}{\partial x}\right) - \frac{2\mu}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)$$

$$\tau_{xz} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)$$

$$\tau_{yy} = \mu\left(2\frac{\partial v}{\partial y}\right) - \frac{2\mu}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)$$

$$\tau_{yz} = \mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)$$

$$\tau_{zz} = \mu\left(2\frac{\partial w}{\partial z}\right) - \frac{2\mu}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) \tag{3.9}$$

$$q_x = -\kappa\frac{\partial T}{\partial x}$$

$$q_y = -\kappa\frac{\partial T}{\partial y}$$

$$q_z = -\kappa\frac{\partial T}{\partial z}$$

The Prandtl number is used to define κ in terms of μ:

$$\kappa = \frac{\mu C_p}{P_r} = \frac{\mu_p}{(\gamma - 1)P_r} \tag{3.10}$$

$D(Q)$ is the artificial dissipation added to the equations to damp unresolved wavenumbers.

3.2 Chain Rule Curvilinear Coordinate Equations

The equations are then transformed into generalized curvilinear co-ordinates:

$$\xi = \xi(x, y, z, t)$$

$$\eta = \eta(x, y, z, t) \tag{3.11}$$

$$\zeta = \zeta(x, y, z, t)$$

$$\tau = \tau(t)$$

The Cartesian derivatives can be written in terms of curvilinear co-ordinates as:

$$\frac{\partial}{\partial x} = \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} + \zeta_x \frac{\partial}{\partial \zeta}$$

$$\frac{\partial}{\partial y} = \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} + \zeta_y \frac{\partial}{\partial \zeta} \tag{3.12}$$

$$\frac{\partial}{\partial z} = \xi_z \frac{\partial}{\partial \xi} + \eta_z \frac{\partial}{\partial \eta} + \zeta_z \frac{\partial}{\partial \zeta}$$

$$\frac{\partial}{\partial t} = \tau_t \frac{\partial}{\partial \tau} + \xi_t \frac{\partial}{\partial \xi} + \eta_t \frac{\partial}{\partial \eta} + \zeta_t \frac{\partial}{\partial \zeta}$$

In this transformation, the τ curvilinear time like coordinate is set equal to the t Cartesian time coordinate.

$$\tau = t$$

$$\tau_t = 1 \tag{3.13}$$

$$\frac{\partial}{\partial \tau} = \frac{\partial}{\partial t}$$

Using this transformation, the Navier-Stokes equations are written as:

$$Q_t + \begin{pmatrix} \xi_t Q_\xi + \xi_x (E - E_V)_\xi + \xi_y (F - F)_\xi + \xi_z (G - G)_\xi \\ \eta_t Q_\eta + \eta_x (E - E_V)_\eta + \eta_y (F - F)_\eta + \eta_z (G - G)_\eta \\ \zeta_t Q_\zeta + \zeta_x (E - E_V)_\zeta + \zeta_y (F - F)_\zeta + \zeta_z (G - G)_\zeta \end{pmatrix} = D(Q) \tag{3.14}$$

The curvilinear coordinate transformations are also applied to the calculation of the viscous stresses:

$$\tau_{xx} = 2\mu(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \Pi)$$

$$\tau_{xy} = \mu \begin{pmatrix} \xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta + \\ \xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta \end{pmatrix}$$

$$\tau_{xz} = \mu \begin{pmatrix} \xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta + \\ \xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta \end{pmatrix}$$

$$\tau_{yy} = 2\mu(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta + \Pi) \tag{3.15}$$

$$\tau_{xz} = \mu \begin{pmatrix} \xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta + \\ \xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta \end{pmatrix}$$

$$\tau_{zz} = 2\mu(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta + \Pi)$$

$$q_x = -\kappa(\xi_x T_\xi + \eta_x T_\eta + \zeta_x T_\zeta)$$

$$q_y = -\kappa(\xi_y T_\xi + \eta_y T_\eta + \zeta_y T_\zeta)$$

$$q_z = -\kappa(\xi_z T_z + \eta_z T_\eta + \zeta_z T_\zeta)$$

$$\tau_{yy} = 2\mu(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta + \Pi)$$

$$\Pi = -\frac{1}{3}\begin{pmatrix} \xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \\ \xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta + \\ \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta \end{pmatrix}$$

3.2.1 Grid Metrics for Chain Rule Equations

The grid metrics can be related to curvilinear derivatives of the Cartesian grid point locations:

$$\frac{1}{J} = \begin{pmatrix} x_\xi\, y_\eta\, z_\zeta + x_\eta\, y_\zeta\, z_\xi + x_\zeta\, y_\xi\, z_\eta \\ -x_\xi\, y_\xi\, z_\zeta - x_\zeta\, y_\eta\, z_\xi - x_\xi\, y_\zeta\, z_\eta \end{pmatrix}$$

$$\xi_x = J(y_\eta z_\zeta - y_\zeta z_\eta)$$

$$\xi_y = J(z_\eta x_\zeta - z_\zeta x_\eta)$$

$$\xi_z = J(x_\eta y_\zeta + x_\zeta y_\eta) \tag{3.16}$$

$$\xi_t = J\begin{pmatrix} y_\zeta\, z_\eta\, x_t + z_\zeta\, x_\eta y_t + x_\zeta\, y_\eta\, z_t \\ -y_\eta\, z_\zeta\, x_t - z_\eta\, x_\zeta\, y_t - x_\eta\, y_\zeta\, z_t \end{pmatrix}$$

$$\eta_x = J(y_\zeta z_\xi - y_\xi z_\zeta)$$

$$\eta_y = J(z_\zeta x_\xi - z_\xi x_\zeta)$$

$$\eta_z = J(x_\zeta y_\xi + x_\xi y_\zeta)$$

$$\eta_t = J\begin{pmatrix} y_\xi\, z_\zeta\, x_t + z_\xi\, x_\zeta\, y_t + x_\xi\, y_\zeta\, z_t \\ -y_\zeta\, z_\xi\, x_t - z_\zeta\, x_\xi\, y_t - x_\zeta\, y_\xi\, z_t \end{pmatrix}$$

$$\zeta_x = J(y_\xi z_\eta - y_\eta z_\xi)$$

$$\zeta_y = J(z_\xi x_\eta - z_\eta x_\xi)$$

$$\zeta_z = J(x_\xi y_\eta - x_\eta y_\xi)$$

$$\zeta_t = J\begin{pmatrix} y_\eta\, z_\xi\, x_t + z_\eta\, x_\xi\, y_t + x_\eta\, y_\xi\, z_t \\ -y_\xi\, z_\eta\, x_t - z_\xi\, x_\eta\, y_t - x_\xi\, y_\eta\, z_t \end{pmatrix}$$

3.3 Numerical Schemes used in BASS Code

In the formulation described in the previous sections, the spatial differencing in BASS code is performed using either a second-order explicit, sixth order explicit, 7-point explicit DRP or sixth order prefactored compact differences. The time marching scheme that is currently used in BASS is Runge-Kutta schemes in a 2N storage format and the MTSAB scheme. The artificial dissipation scheme used in BASS code is a blended dissipation scheme and combines explicit high accuracy background dissipation with a second order shock capturing dissipation [23].

3.4 Structure of BASS Code

The spatial grid used by the code is block-structured. Each block can have arbitrary surface patches on each block face. These surface patches can be connectivity patches (where two computational blocks have interface in the interior of the computational domain) or boundary condition patches (where the boundary of the computational domain lie, and inflow, outflow or wall boundary conditions must be specified).The main computational work of the code for a given block can be split into these several areas:

a) Grid metrics calculation (at every step for moving grids)

b) Flux calculation

c) Artificial dissipation calculation

d) Flux derivative calculation

e) Boundary condition calculation

f) Flow time update

g) Block communications

The first five steps require spatial derivative noting that the flux calculation requires spatial derivative only for a viscous problem.

The different volumes used in the BASS code are shown in Figure 3-1. The basic object in the code is a „spatial derivative volume', in which the values of spatial derivatives are calculated. To define fully a spatial derivative volume there are three volumes. The first volume is the interior volume. In this volume the spatial derivatives will be used by the code. The second volume, which encompasses first as its subset, is the derivative volume which is defined as the volume in which the derivative is computed. The third and the largest volume is the „total volume' which contains all the variable data that is necessary for spatial derivative to be computed in the „derivative volume'. The next level of objects are „boundary condition volume' which contain „spatial derivative volumes' and any other information such as mean flow data or source data that are required to define the boundary conditions. The next higher object is the „block volume' which is a grid block that may contain a number of „boundary condition volumes'. The highest object is the „node volumes' dimensioned to the number of processors in the calculation.

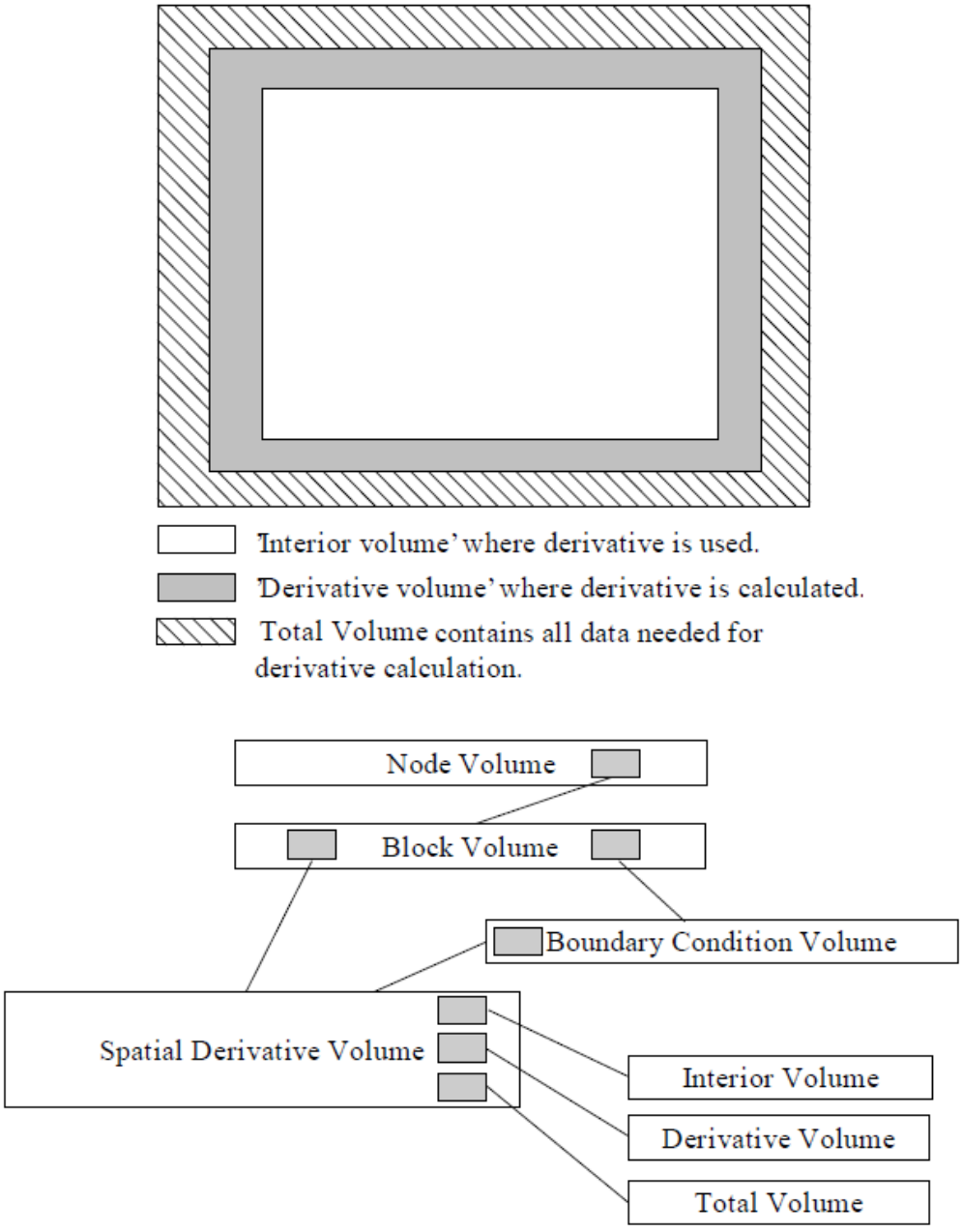


Figure 3-1 Data Structure in BASS Code

3.5 Data transfer Mechanism in BASS Code

Whenever a grid block (block 1) has a boundary that connects to another grid block (block 2), the total volume of the grid block 1 must contain data from the neighboring grid block 2 in order to compute the spatial derivative at the boundary of grid block 1.

There are three data transfers that may occur during the single time derivative calculation:

a) Grid data transfer (to compute the grid metrics)

b) Flow data transfer (to compute the artificial dissipation and the viscous fluxes)

c) Flux data transfer (to compute the flux derivatives, and thus the time derivatives)

These data transfers between the blocks can be local or non local depending on whether the blocks are on the same processor. If the blocks are both on the same processor, the data can be transferred directly between the blocks as needed during the time derivative calculation process.

However, if the blocks are on different processors, messages that contain the data must be passed between the processors.

The way in which these messages are passed determines how efficient the parallel performance of the code. The message passing consists of two stages: message initiation and data transfer. Message initiation requires a fixed amount of CPU time, and data transfer is a function of the amount of data (message length) in the message. The data transfer rate increases with the message length until the network maximum data transfer is reached, where upon the data transfer rate remains at the maximum. Thus for parallel efficiency, it is preferred to have as few messages as possible (minimum number of initiation) that are as long necessary (maximum data transfer).

Each block requires data from the neighboring blocks to compute spatial derivatives. This data is taken from the neighboring block and placed in the total volume of the block. The data that is required in the total volume of the local block becomes available on the neighboring block at different times during the derivative calculation process.

As explained in Chapter 2, buffer blocks contain data from the neighboring blocks. For the MTSAB scheme to work, buffer blocks also march in time (as explained in chapter 2), when the neighboring block is at a different MTSAB level.

These buffer blocks may themselves have buffer blocks associated with them; this occurs in the case of viscous flow calculation. The code defines the main grid block as Type 0 blocks; the first level of the buffer blocks as Type 1, the next level Type 2, etc. Figure 3-2 shows the grid blocks and buffer blocks in two dimensions.

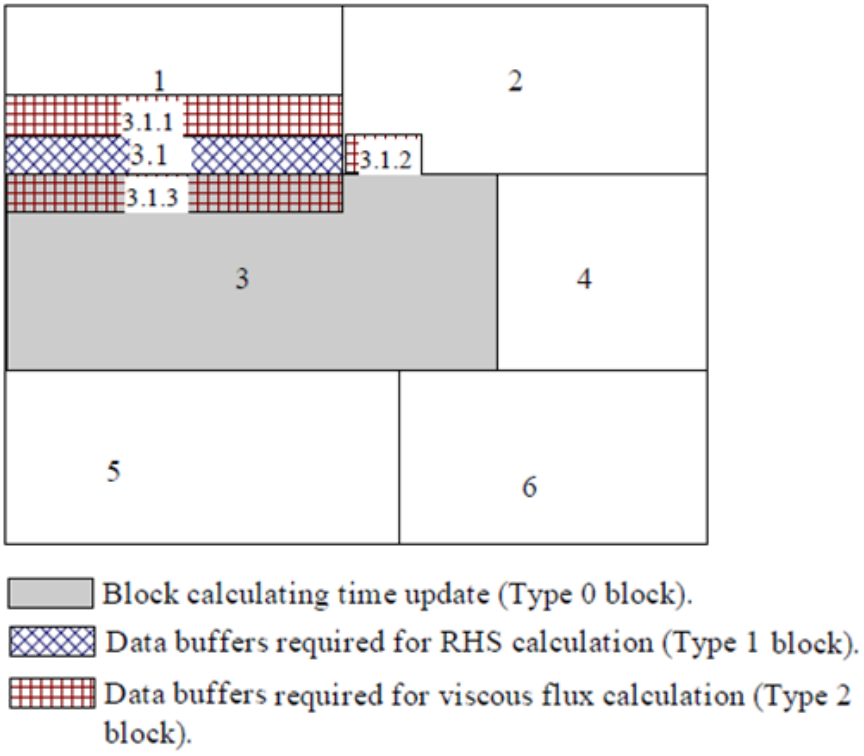


Figure 3-2 An example buffer blocks in BASS code

3.6 Automated MTSAB for BASS Code

BASS code is used to solve realistic flows over complex geometries (for e.g. (24)). The grids generated for these geometries can also be complex. It is difficult for a user to assign different values of time steps to different blocks of the multi-block grid (method explained in the previous chapter). Even if the user was able to do that, it is very difficult to keep track of the changing stable time steps in cases such as moving grid calculations or flows with large disturbances. In the implementation of the MTSAB scheme in BASS code, the optimization is performed during the run, to achieve maximum speed-ups. The optimization algorithms are explained later in the chapter. It is difficult for the user to perform this optimization manually. A major part of this work was to automate the MTSAB scheme in BASS code to overcome these hurdles. All the features of the automation are explained below:

3.6.1 Assigning the Time Steps to Grid Blocks

MTSAB scheme in BASS code first starts with the single step time marching scheme (as explained in chapter 2). Based on the CFL condition the code calculates the global minimum time step (Δt_{global}) for the complete grid, and local minimum time step (or local stable time step) for each grid block of the multi block grid (Δt_{stable}). After the first few single steps the code dynamically groups the blocks into different levels and are assigned time steps based on their level values.

To quantify the speed increase from using the MTSAB scheme, Actual Speedup and Theoretical Speed-up can be calculated. The formulae for the Actual and Theoretical speed ups are given by Eq. (3.17) and Eq. (3.18).

The Actual Speed-up includes overhead (calculations in the buffer blocks). Theoretical speed up does not include this overhead. Therefore the Theoretical Speed up is the maximum possible speed up for a given distribution of grid points in each level. It can be thought of as Carnot efficiency in a thermodynamic cycle.

$$Actual\ Speed-up = \frac{Run\ time\ for\ a\ \ Single\ step\ AB\ scheme}{Run\ time\ for\ the\ MTSAB\ scheme} \tag{3.17}$$

$$Theoretical\ Speed-up = \frac{Total\ number\ of\ \ grid\ points}{Effective\ number\ of\ \ grid\ points} \tag{3.18}$$

In Eq. (3.18), the total number of grid points and the effective number of grid points are:

$$Total\ number\ of\ \ grid\ points = \sum_{n=1}^{max.level} Grid\ points\ in\ level\ n \tag{3.19}$$

$$Effective\ num.of\ \ grid\ points = \sum_{n=1}^{max.level} \frac{Grid\ points\ in\ level\ n}{2^{n-1}} \tag{3.20}$$

3.6.2 Dynamic level change

Once the grid blocks are marching with different time steps, condition given by Eq. (2.5) is checked at regular intervals. The interval at which this check is performed can be set by the user in the BASS Input File. If at any time during the run this condition fails, the

code dynamically (or during the run) re-assigns the level values to the grid blocks, such that condition in Eq. (2.5) is satisfied again. This is a very useful automation for simulations with grid motion and flows with large disturbances, where the local stable time step can change.

3.6.3 Optimization using Block cutting Algorithms

In this work two block cutting algorithms were designed to optimize the implementation of the MTSAB scheme. These algorithms are discussed below:

a) Algorithm to increase MTSAB Speed-ups

Within a given grid block, every point can have a different level (based on the CFL at that point). The level value assigned to the block is smallest level value of all the points within the block (for stability). This algorithm checks for point to point variation of the level value within the grid block and cuts the blocks based on this variation. As a result, new blocks are generated at higher levels. This shifts more grid points to a higher level and therefore increases the speed-up as shown by Eq. (3.17) and Eq. (3.18).

This is explained with an example. Consider an O-grid with a single grid block (Figure 3-3) around a 2-D cylinder. The grid is clustered near the cylinder and is stretched away from it. Let the global time step for the grid be (Δt_{global}). It is based on the smallest grid spacing (near the surface of the cylinder). Since the grid has one block, for numerical stability, ($\Delta t_{stable\ 1}$) is equal to (Δt_{global}) for the block.

The level value for this grid block is 1 and there is no speed increase on applying the MTSAB scheme. Since the grid spacing on the cylinder is very small as compared to the

grid spacing at the boundaries, there is a grid point to grid point variation of levels or stable time steps within the grid block.

The block cutting algorithm cuts the original block during the run. It uses the point to point variation of levels within the grid block as a guiding tool to generate new blocks at higher levels. On using the automated block cutting algorithm, 6 blocks are generated at 6 levels from the initial grid block. These blocks are shown in Figure 3-4. The block boundaries are shown by black lines. The new blocks, which are at higher levels, can increase the speed of the run MTSAB scheme.

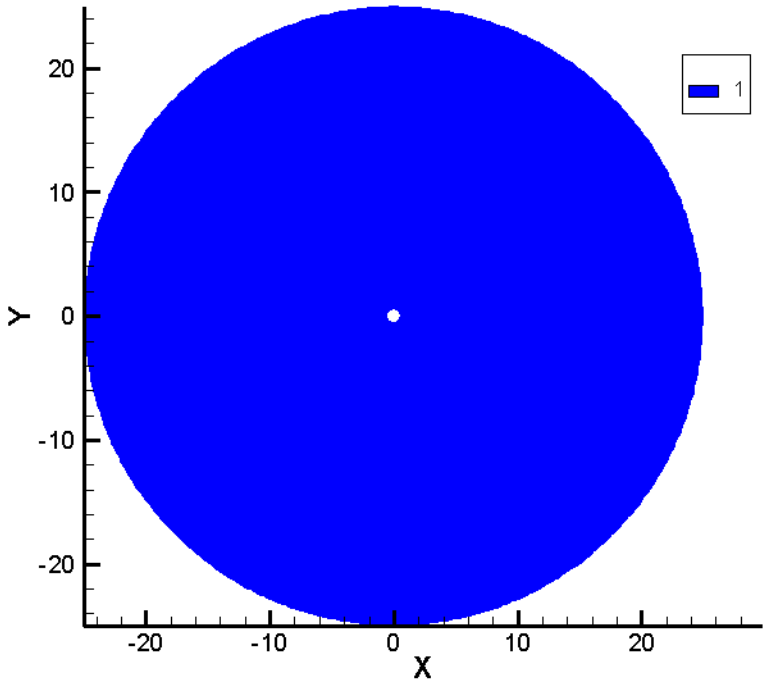


Figure 3-3: Single block O-grid around a 2D cylinder has one MTSAB level

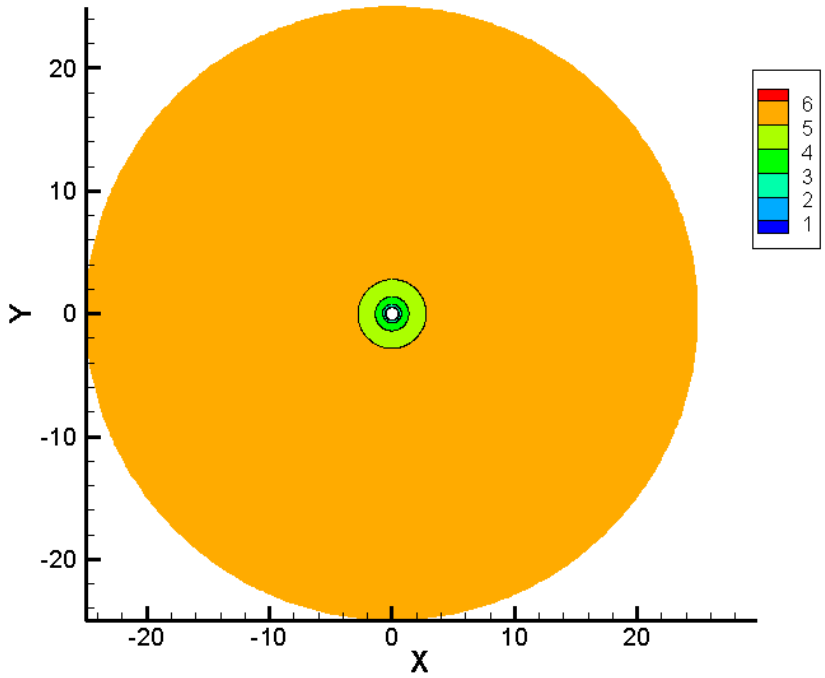


Figure 3-4: Six blocks generated at six MTSAB levels after cutting the original block, based on the point to point variation of levels within the block

The method used by this algorithm to detect the variation of levels within a given grid block and to cut the block is explained here with an example. Consider a block shown in Figure (3-5a). Figure 3.5a shows the point to point variation of the levels within the block. There are two levels within the block. Level 1 is shown by the red region and the rest of the block is at level 2.

Before the block is cut, the block is at level 1 (minimum level value within the block). The following steps are performed by the algorithm for one cut in i-direction. The same method is used for all the cuts the code makes, in all the directions.

1) The block cutting algorithm uses the level variation data in the block and first constructs a histogram for the number of points in level 1, for every grid plane perpendicular to *i*-direction. This histogram is shown in Figure 3-5b.

2) The algorithm then detects the *i*-location with maximum value in the histogram. This is shown as *ihist* line in Figure 3-5c.

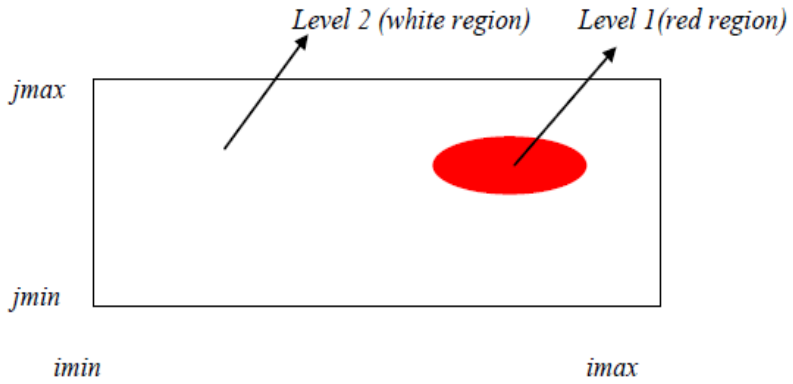
3) Once the *ihist* line is located, the block cutting algorithm calculates the location of two other lines: *ihist (+) and ihist (-).* These lines completely cover the level 1 region. These lines are shown in Figure (3-5d).

4) The algorithm then decides whether to make the cut on the *ihist (+)* line or the *hist (-)* line. For this, which ever of these two lines are closer to the nearest boundary in the *i*-direction snaps to that boundary and the cut is made on the other line.

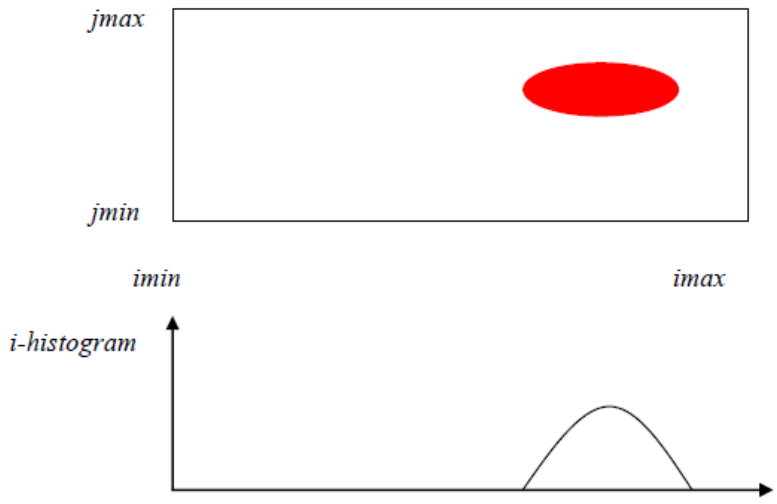
In the example, *ihist (+)* is closer to the *imax* boundary than *ihist (-)* is from the *imin* boundary. Therefore, *ihist (+)* snaps with the *imax* boundary and the cut is made on the *ihist (-) line*.

4) The above steps are repeated in all the remaining cuts, in all the directions (including current direction). Figure 3-5g shows the two blocks after the cut is made. The new block generated as a result of the cut is at level 2.

(a)


Level 2 (white region)
Level 1(red region)
jmax
jmin
imin
imax

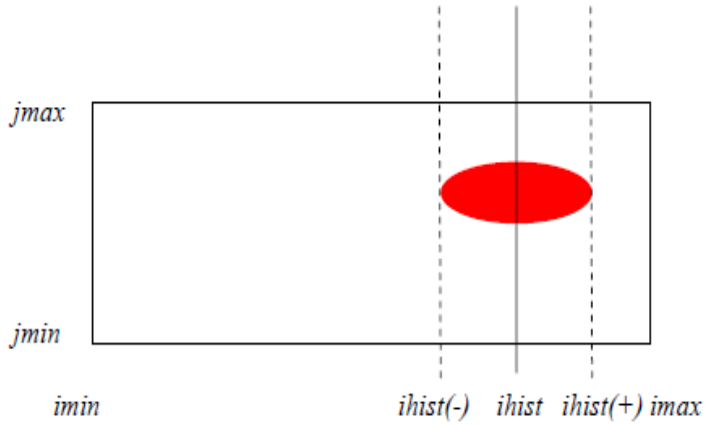
(b)


jmax
jmin
imin
imax
i-histogram

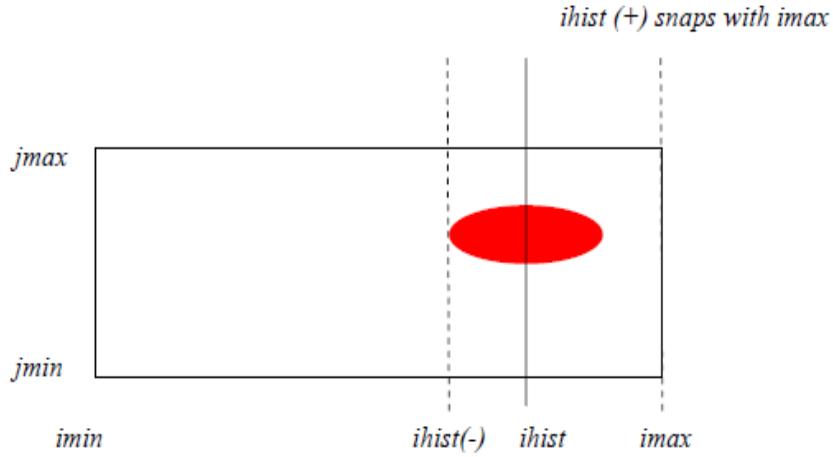
(c)


jmax
jmin
imin
ihist
imax

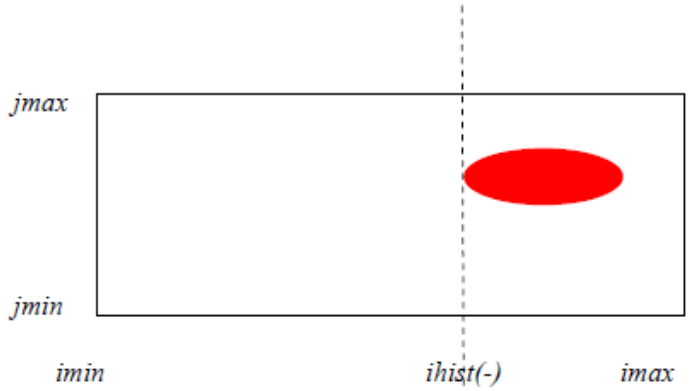
(d)


jmax
jmin
imin
ihist(-)
ihist
ihist(+)
imax

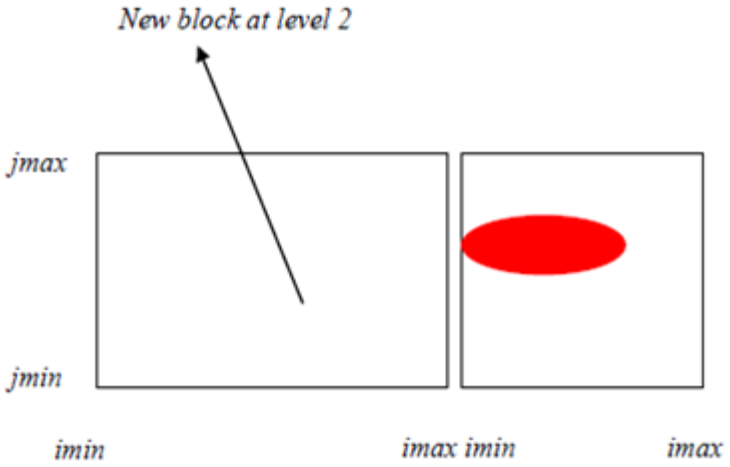
(e)


ihist (+) snaps with imax
jmax
jmin
imin
ihist(-)
ihist
imax

(f)


jmax
jmin
imin
ihist(-)
imax

(g)



Figures 3-5: Pictures *(a)* to *(g)* show how one cut is made using the block cutting algorithm

b) Algorithm for Load Balancing

Once the blocks are cut to maximize speed of the MTSAB scheme, a second block cutting algorithm is used for getting an efficient load balancing of the blocks in each level, for parallel computing.

Consider the grid over the cylinder again. If this same grid is run on four processors (of equal efficiency), the second block cutting algorithm cuts the block at each level into four equal blocks, and a total of 24 blocks are generated. Figure 3-6 shows all the new blocks and their levels after using both the block cutting algorithms. This algorithm uses METIS [25] recursively for the load balancing. Input that is given to METIS is a matrix that contains the number of points in each block (diagonal elements) and the amount of communication between the blocks (non-diagonal elements). A second input contains the number of processors and their corresponding weights.

The following are the steps performed in the block cutting algorithm at each MTSAB level.

1) Calculate the inputs matrices for METIS and call METIS.

3) Calculate the over loading or under loading for each processor using the distribution (output) given by METIS. The underloading or overloading is given as,

$$Overloading\ or\ Undeloading = \left| \frac{(Actual\ points\ on\ processor - Ideal\ points\ on\ processor)}{Ideal\ points\ on\ processor} \right|$$

where, the actual points on the processor is the number of grid point currently assigned to the processor by METIS and the ideal points on the processor is the processor weight multiplied by the total number if grid points in the domain.

4) $If\ [\ (Max(overloading) + Max(Underloading)) > 0.2]$, the largest block into two equal parts. The cut is made on the plane perpendicular to the direction with the largest number of points. Steps *1-4* are repeated again.

$If\ [\ (Max(overloading) + Max(Underloading)) \leq 0.2]$, or if the blocks cannot be cut any more (the minimum number of points in every direction for any block is 10) exit the algorithm.

The minimum number of points in every direction was hardwired 10 for this work. For values below 10, too many blocks were produced for some cases (from both the algorithms). For these cases, for values below 10, the algorithm used for increasing the MTSAB speed, tried to follow the point to point variations of levels too closely and in

turn produced too many connected blocks at different levels. This in turn produced huge number of buffer blocks. The MTSAB calculations in these buffer blocks caused an increase in the overhead. This also increased the overhead due to local data transfers between the blocks.

As a part of this work, sections in BASS code performing the local transfer of data was improved and overhead from this was reduced. At present, the overhead is dominated by the buffer block calculations required for the MTSAB scheme.

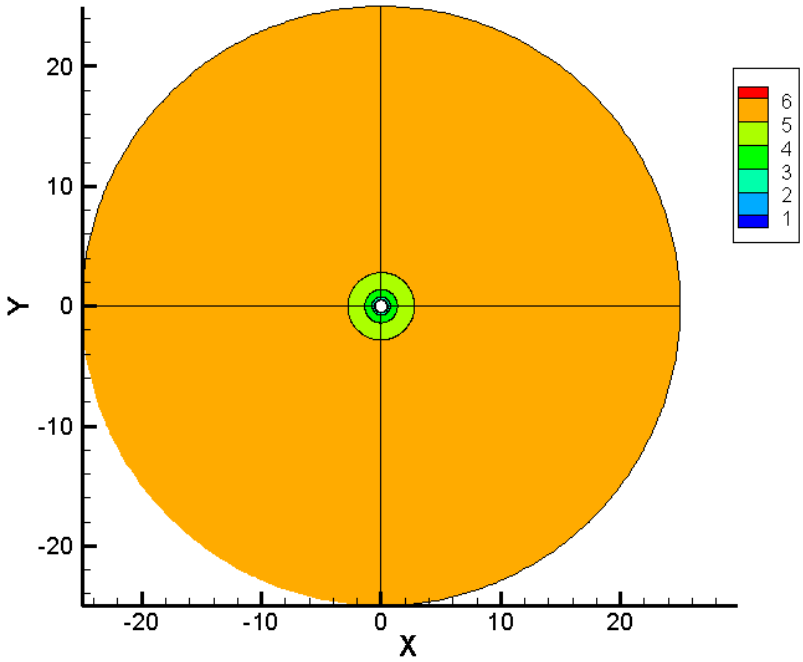


Figure 3-6: A total of 24 blocks are generated after using both the block cutting algorithms

Chapter Four

Validation of the Multi-Time Stepping Adams- Bashforth (MTSAB) scheme for Computational Aeroacoustics (CAA) Problems

4.1 Introduction

Calculation of unsteady flow and noise is more demanding than calculation of steady flow solutions [1]. The field of Computational Aeroacoustics (CAA) is focused on the application of computational methods for the purpose of understanding the physics noise generation and propagation. To validate the numerical schemes developed for CAA, a range of benchmark validation problems have been specified, and solutions made available in the CAA workshops [11-14].

Three of the CAA workshop problems were used to test the accuracy of the automated MTSAB scheme in BASS Code. The results from using the MTSAB scheme are presented in this chapter.

4.2 Problem-1: Acoustic Scattering

This test case is from the 2nd CAA Workshop [12]. This test case is an initial value problem, and simulates the propagation of an acoustic pulse and its reflection from a 2D-cylinder. In Figure 4-1, point O is the center of the cylinder and point S is the location of the initial acoustic pulse. As the pulse travels, it is scattered off by the cylinder. The problem is to find the pressure histories at points A ($r = 5, \theta = 90^{\circ}$), B ($r = 5, \theta = 135^{\circ}$) and C ($r = 5, \theta = 180^{\circ}$)

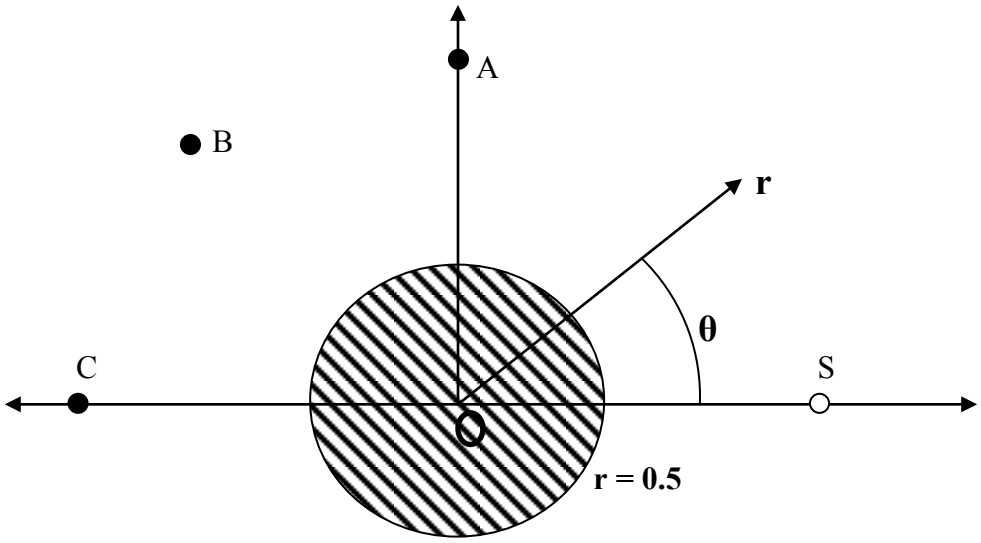


Figure 4-1: Configuration of the Acoustic-Scattering problem

4.2.1 Equations

The equations specified in the workshop, for this problem are the Linearized Euler Equations. As mentioned previously, BASS code is a Navier-Stokes solver. The equations in BASS Code can be reduced to Euler Equations (non-linear) by switching off the viscous terms in the Navier-Stokes equation. The idea, that for small amplitudes of

disturbance, (in this case, the acoustic pulse) the Euler equations behave like the Linearized Euler Equation, was used for this solving the problem [26].

In the problem, the initial acoustic pulse is given as:

$$p' = Ae^{\left[-ln2\left(\frac{(x-4)^2+y^2}{0.2^2}\right)\right]} \tag{4.1}$$

where the amplitude of the pulse,

$$\text{A} = 1.0$$

To solve the problem with Euler equations, the amplitude was reduced to,

$$\text{A} = 0.001$$

The initial conditions used for this problem are:

$$\rho = 1.0$$

$$p = \frac{1}{\gamma} \tag{4.2}$$

$$u = 0.0$$

$$v = 0.0$$

In Eq. (4.2), $\gamma = 1.4$.

After computing the results with the reduced amplitude, the values in the pressure perturbation history at points A, B and C were scaled back, by multiplying the values by a factor of $1/0.001$. This was done to compare the results from the current simulation to the exact solution.

4.2.2 Grid and Numerical Details

The circular cylinder has a radius $r = 0.5$ and is located at the center of the computational domain. The numerical simulation was carried out in a domain that extends from $r = 0.5$ to $r = 10.5$. Figure 4-2 shows the computational grid. The computational grid consists of two grid blocks with 201 grid points in the radial direction and 201 grid points in the azimuthal direction in each grid block. Both the grid blocks extend from the cylinder surface to the outflow boundary. The grid has a uniform spacing in the *r* and θ direction.

The 4^{th}-order DRP scheme was used for spatial differentiation. BASS code was run twice for this case. For the first run, the single step Adams-Bashforth scheme was used. The automated MTSAB scheme was used for the second run. The CFL set to 0.28 for both the cases.

Wall boundary condition [27] on the cylinder sets the normal velocity on the wall to zero. Acoustic Radiation (ACRAD) boundary condition [4] was used at the outflow boundary. The stable CFL for ACRAD boundary condition was 0.14-0.16 approximately, for the value of background dissipation (10^{th} order) coefficient ranging between 0.05-0.1. This may be attributed to the fact that some boundary schemes may have a lower stability limit [29]. The exact reason for this not currently known. The CFL was therefore reduced to 0.15 at the outflow boundary.

4.2.3 MTSAB Performance

The size of the grid cell is a function of the radial distance from the cylinder, with the smallest cell on the cylinder surface and the largest at the outflow boundary. The

MTSAB scheme first calculates the level value (explained in chapter 2) at each grid point in the grid. This point by point distribution of the MTSAB levels in the grid is shown in Figure 4-4. It can seen in this figure, that the point to point distribution of the levels varies from a value of 1 to 3.The level drops to a value of 2 at the outflow boundary.(due to the lower stability limit of ACRAD boundary condition).

Without using the block cutting algorithm during the run, the stable time step for both the grid blocks is based on the smallest cell size in them, which is located on the cylinder surface. Therefore, both the grid blocks are at level one as shown in Figure 4-3. On using the block cutting algorithm during the run, the two original grid blocks are cut based on the point by point distribution on levels. The distribution of levels in the grid after the original blocks are cut, is shown in Figure 4-5

Table 4.1 shows the speed-up data from using the MTSAB scheme. The table shows the actual, theoretical and ideal speed ups. The table also gives the number of points in each level for: point to point distribution, with block cutting and without cutting the blocks. The ideal speed up, which is based on the point to point distribution of the levels in the grid, is 3.03. There is no speed up without cutting the blocks because all the blocks (or grid points) are at level 1. The theoretical speed up after cutting the blocks during the run is 2.92 and the actual speed up is 2.6. The difference between the theoretical speed up after block cutting and the ideal speed up is because of the fact that, the minimum number of points for any block in both the directions was set to 10 points.

After the blocks are cut, there are 8 blocks and the difference between the actual and the theoretical speed up is the overhead from the buffer block calculations.

4.2.4 Results

Figures 4-6 to 4-9 show the snapshots of the pressure contours at different times. The initial pulse is located in the level 3 blocks. At around T=4 the large wave front reaches the cylinder surface and a small reflected wave (second wave front) can be seen near the cylinder. At this time the large wave front has already crossed through all the level change interfaces.

At T=7 a third wave front is observed near the cylinder which is generated as the initial wave propagates over the cylinder. At T=10 the initial pulse has already reached the outflow boundary and a smooth transition towards outside is seen. At this time, the second and the third wave fronts have crossed through all the level change interfaces.

Figures 4.10-4.12 compares the time history of the pressure perturbations at points A, B, and C. These points are shown in Figure 4-1. The numerical solutions are compared to the exact (analytical) solution given by Kurbatskii in [12].

There is an excellent match, in amplitude and phase in the pressure histories at these locations for the single step Adam-Bashforth scheme and the MTSAB scheme. Small differences, between the exact and the computed results can be seen.

In Figure 4-10, the percentage of difference between the exact data and the data from using the MTSAB scheme for point A, at the maximum peak location is 0.41%. At the minimum peak location the difference is 1.38%.

In Figure 4-11, the percentage of difference between the exact data and the data from using the MTSAB scheme for point B, at the maximum peak location is 0.29%. At the minimum peak location the difference is 0.25%.

In Figure 4-12, the percentage of difference between the exact data and the data from using the MTSAB scheme for point C, at the maximum peak location is 0.19%. At the minimum peak location the difference is 1.8%.

	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Theoretical Speed up	Actual Speed up	Number of blocks
Point by Point Distribution	4422	12462	63918	3.03	-	-
Distribution Without Block Cutting	80802	0	0	None	None	2
Distribution With Block Cutting	4422	16482	59898	2.92	2.6	8

Table 4.1: Speed up data for the MTSAB scheme

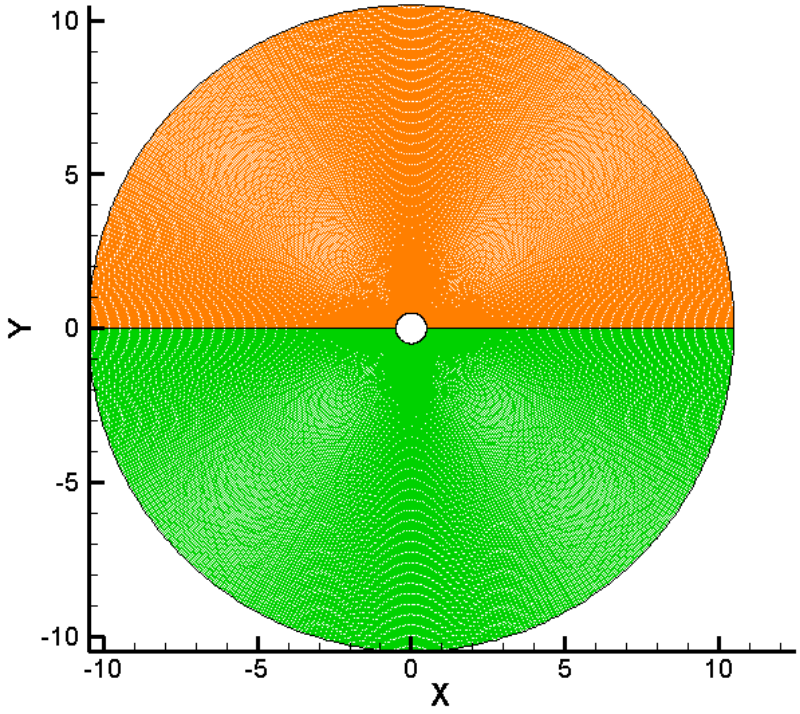


Figure 4-2: Grid for the Acoustic-Scattering Benchmark problem

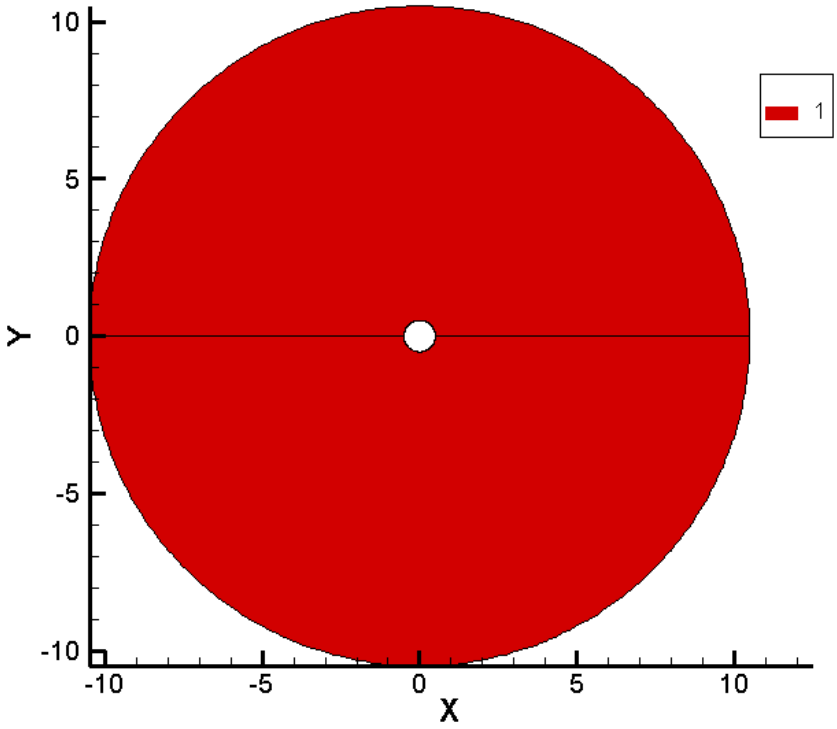


Figure 4-3: MTSAB level in the grid without block cutting

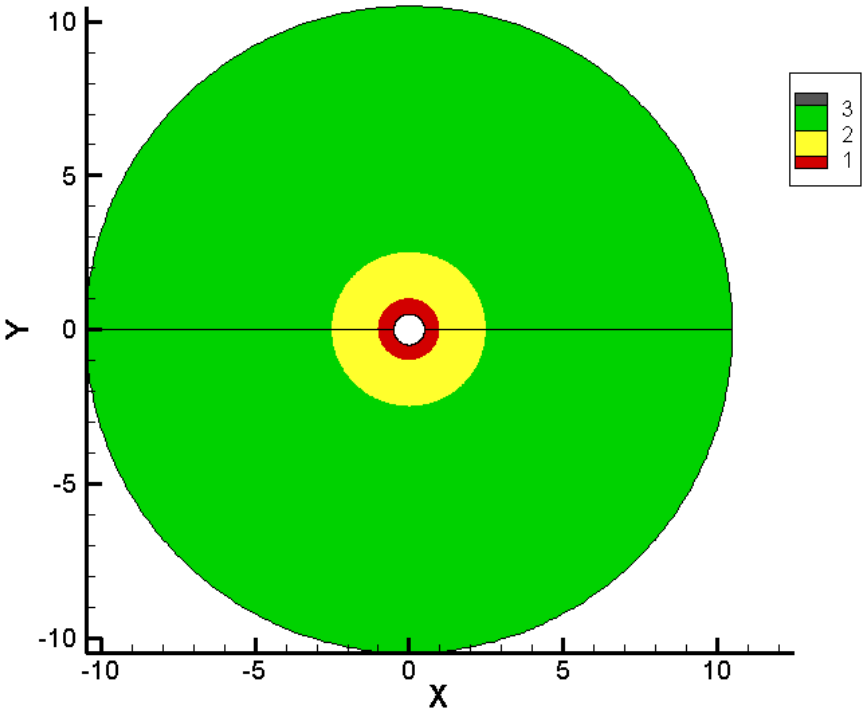


Figure 4-4: Point to Point Distribution of the MTSAB levels in the grid

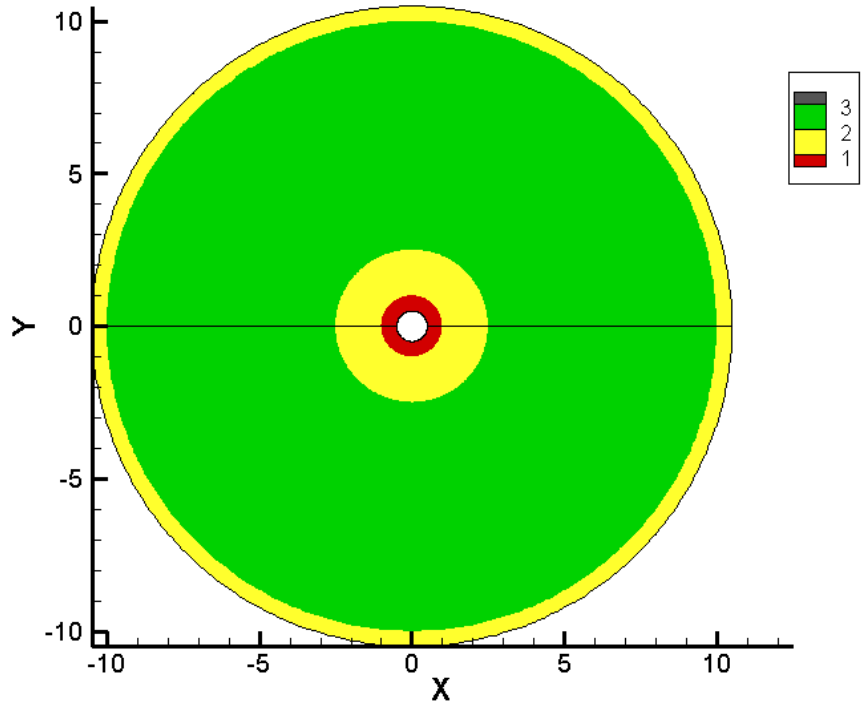


Figure 4-5: New grid block and their levels, after the blocks are cut during the run

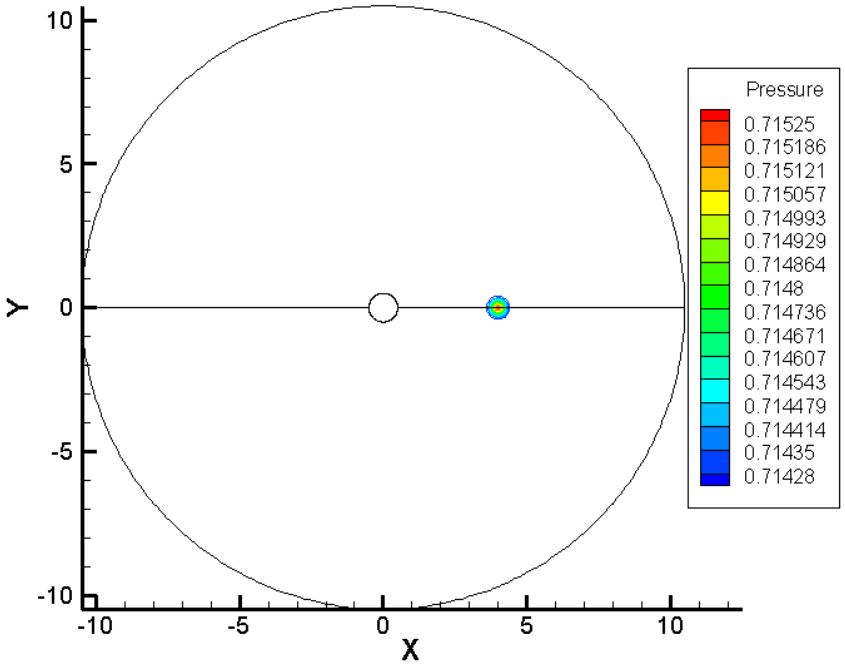


Figure 4-6: Acoustic pulses at T=0

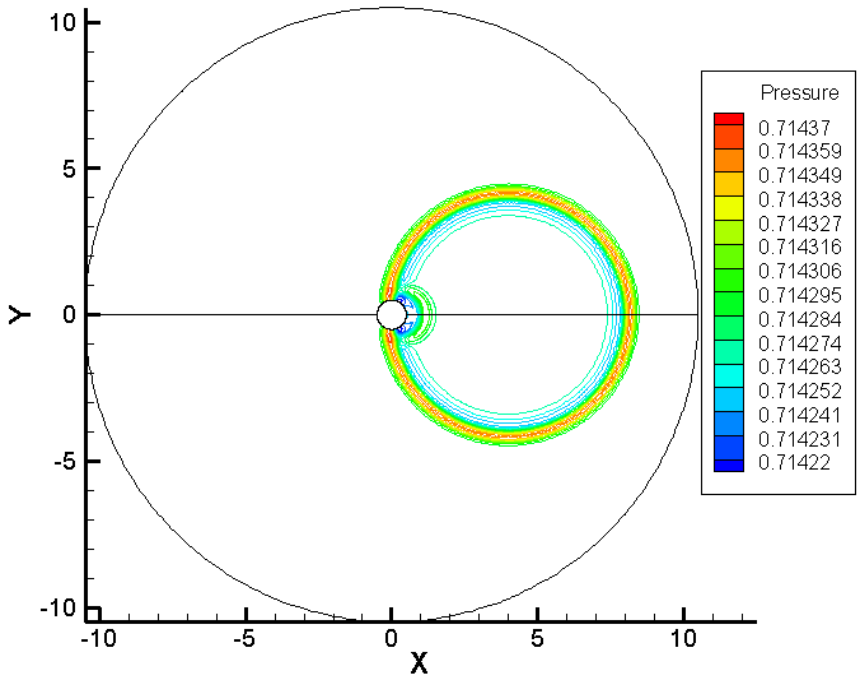


Figure 4-7: Acoustic pulse at T=4

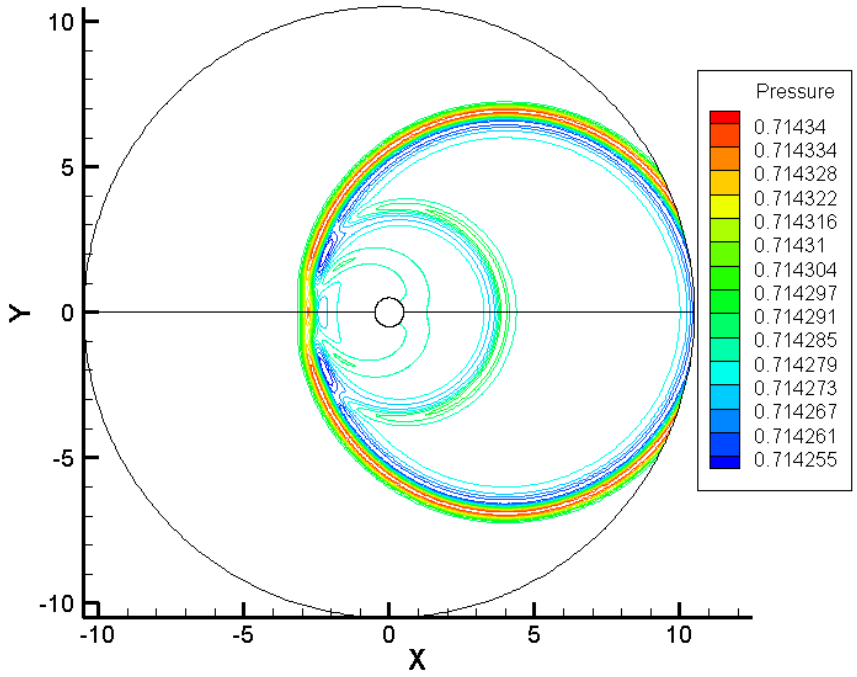


Figure 4-8: Acoustic pulse wave at T=6.8

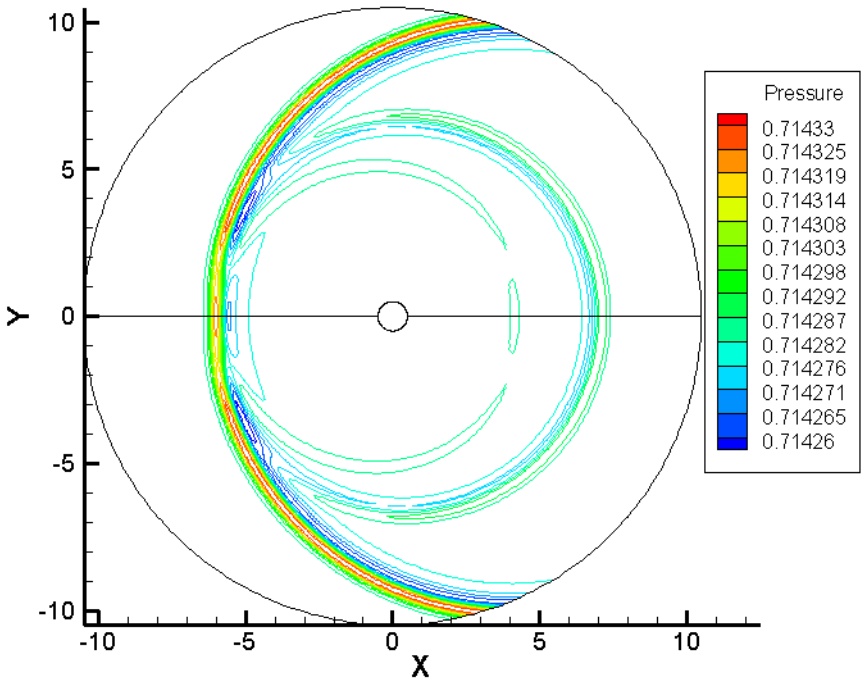


Figure 4-9: Acoustic pulse wave at T=10

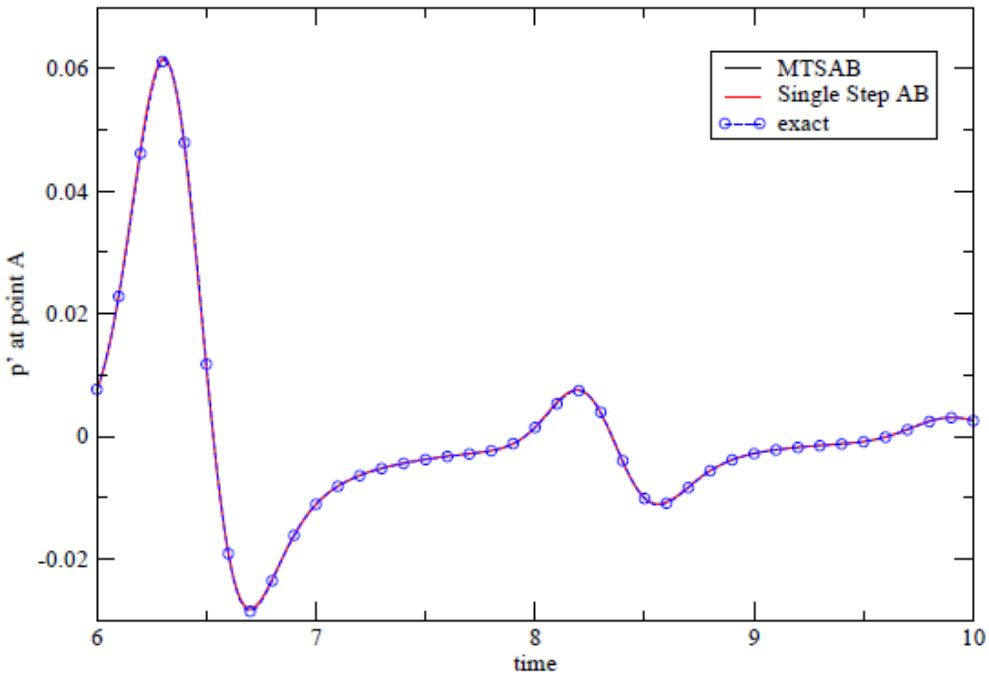


Figure 4-10: Pressure disturbance history at point A

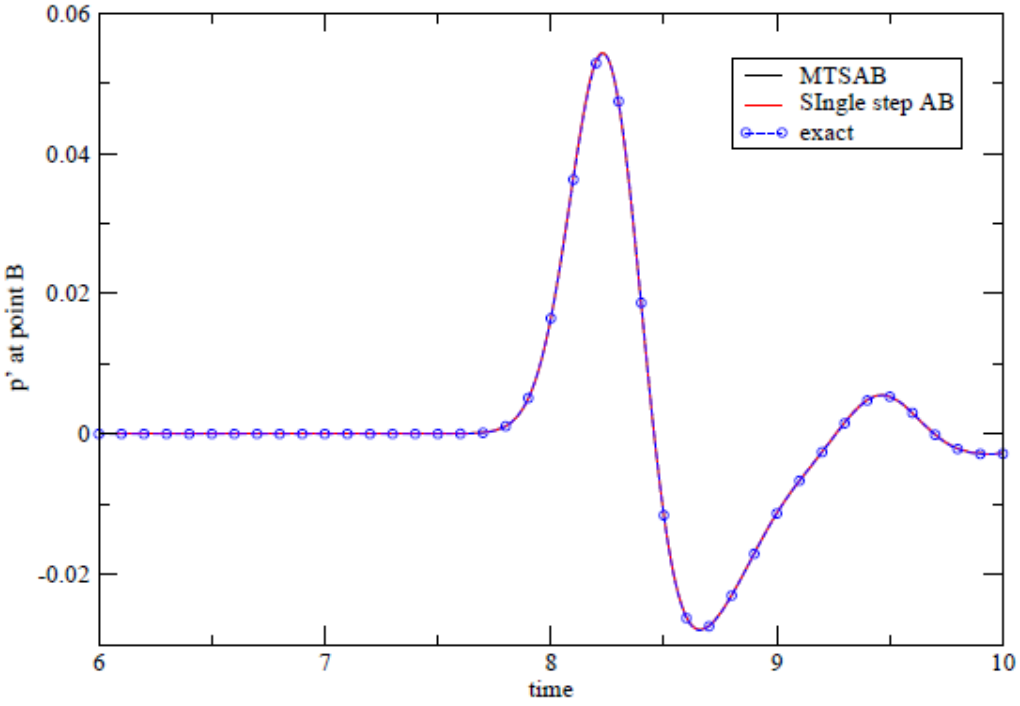


Figure 4-11: Pressure Disturbance History at point B

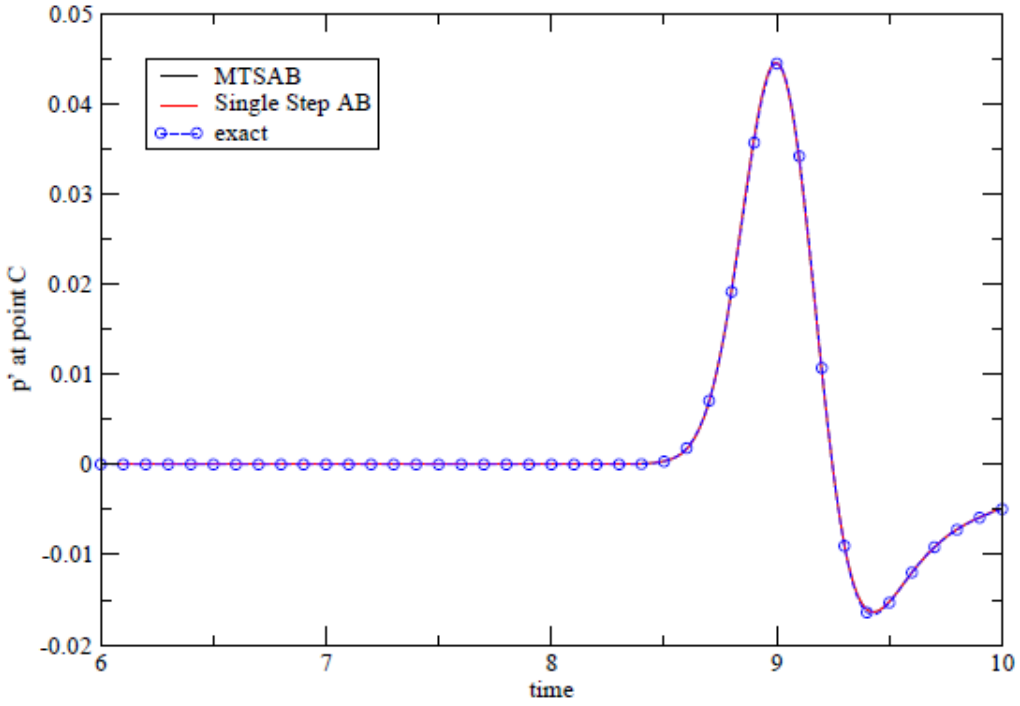


Figure 4-12: Pressure Disturbance history at point C

4.3 Gust-Airfoil Problem

The MTSAB scheme then was tested on a realistic 2D benchmark gust-airfoil interaction problem from the 4th CAA Workshop [14]. In this test case, the nonlinear Euler equations are solved for the unsteady flow about a cambered Joukowski airfoil. The free stream Mach number for this case is 0.5. A two dimensional vortical gust is introduced at the inflow boundary and convects with the flow. When the gust impinges the airfoil, noise is radiated. The BASS code has been extensively tested for this problem; details of these calculations are given in [30].

4.3.1 Grid and Numerical Details

The grid used was C-H topology, extending at least 10 chord lengths away in each direction. The grid was generated using the commercial package GridPro [31]. The reduced frequency of the vortical gust, is introduced at the inflow boundary is defined as,

$$k = \frac{\omega c}{2M_\infty a_\infty} \tag{4.3}$$

In Eq. (4.3), c is the chord length and a_∞ is the speed of sound. In this current work, the MTSAB scheme is validated for the values of the reduced frequencies, $k = 1.0$ and $k = 2.0$. The vortical gust is defined as,

$$u_{gust} = -\frac{\varepsilon\beta M_\infty}{\sqrt{\alpha^2+\beta^2}}\cos\left(\alpha x + \beta y - \omega t\right) \tag{4.4}$$

$$v_{gust} = -\frac{\varepsilon\alpha M_\infty}{\sqrt{\alpha^2+\beta^2}}\cos\left(\alpha x + \beta y - \omega t\right) \tag{4.5}$$

$$\epsilon = 0.001 \; and \; M_\infty = 0.5$$

$$\alpha = 2k \quad \beta = 2k \quad \omega = 2kM_\infty \tag{4.6}$$

The grid was designed to have a minimum of 15 grid points per wavelength at the highest reduced frequency of inflow disturbance; to minimize the effects of grid variations on the solution.

Figure 4-13a and 4-13b show a close-up of the body-fitted grid used about the Joukowski airfoil. The grid is clustered near the leading edge of the airfoil in order to resolve the sharp flow gradients in this region.

The spatial derivative scheme used for this problem is the DRP scheme. The time marching scheme used are: single step Adams-Bashforth scheme, the MTSAB scheme with block cutting algorithm switched off and MTSAB scheme with block cutting. The results are compared to previously validated results from this code using Stanescu and Habashi's [9] low-storage extension of Hu's LDDRK56 method [8].

The smallest time step for this problem is based on stability restriction from the clustered grid at the leading edge of the airfoil. To provide damping for the inviscid nonlinear calculation, an explicit constant-coefficient 10th-order artificial dissipation using Kennedy and Carpenter's boundary stencils [32] was added at each stage of the RK56 scheme and each step of the Adams-Bashforth scheme.

In this calculation, the physical wavelength of the unsteady gust for $k = 2.0$ is approximately 1.5 times the chord length of the airfoil; thus, the time step obtained from the CFL is very small as compared to the time step necessary for resolving the unsteady gust dynamics. Therefore, using an MTSAB scheme, small time steps are only used

where the time step is restricted by grid clustering at the leading edge of the airfoil. In the rest of the region, as stable time step size increases, larger time steps can be taken while still resolving the unsteady vortical gust.

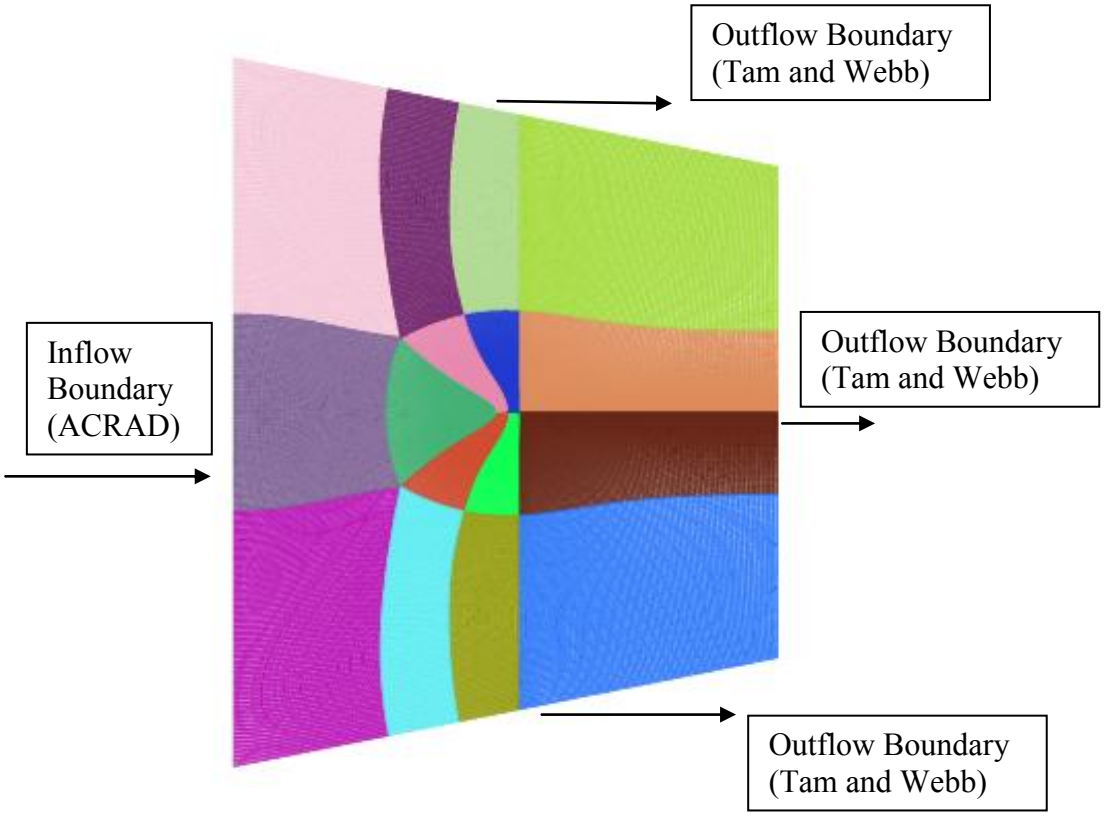


Figure 4-13a: Grid used for the Joukowski airfoil case

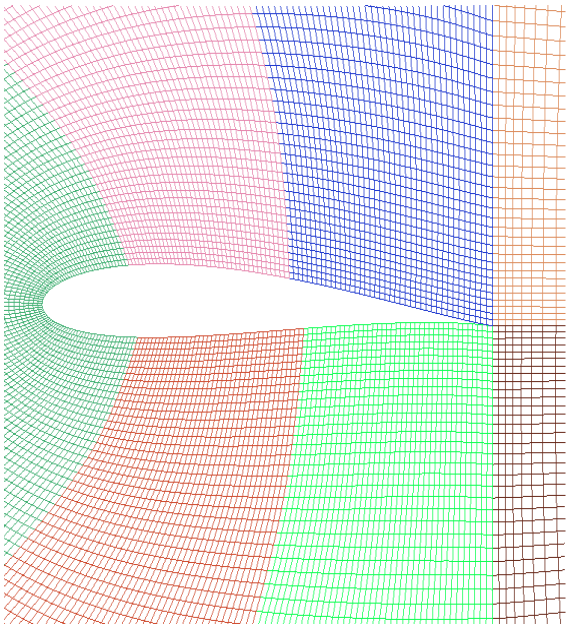


Figure 4-13b: Grid used for the Joukowski airfoil case

As an initial condition for the runs, the airfoil is impulsively started in a uniform flow field. The gust is introduced at the inflow boundary and convects into the domain.

As shown in Figure 4-13b, Acoustic Radiation (ACRAD) boundary condition [4] was used at the inflow and Tam and Webb boundary condition [4] was used at the three outflow boundaries.

The CFL of 0.30 was used for the Adams-Bashforth scheme. The stable local CFL at the ACRAD boundary was 0.15. For the RK56 scheme a CFL of 1.25 was used.

4.3.2 MTSAB Performance

Figure 4-14 shows the MTSAB level distribution in the grid when the blocks are not cut. It can be seen from this figure that all the blocks surrounding the airfoil are at levels 1 and 2. The rest of the blocks are at level 4.

Figure 4-15 show the point to point variation of level in the complete grid and Figure 4-16 shows the point to point distribution of levels near the airfoil surface. It can be seen from these figures that, only near the leading edge of the airfoil, where the grid is clustered to resolve the mean flow gradient, the grid points are at level 1. As the grid stretched away from the stagnation point, the levels of the points increase to a maximum value of 5. It can be seen in these figures that the original grids blocks can be cut, to generated new blocks at higher levels.

Figure 4-17 and figure 4-18 show the distribution of levels in the grid after the blocks are cut. It can be seen from these figures that, after the blocks are cut, there are 5 levels in the domain as opposed to 4 (without cutting the blocks). At the inflow boundary there is a drop in the level to a value of 4. This is because of the lower stable CFL at the ACRAD boundary.

By comparing Figures 4-15, 4-16, 4-17, 4-18 it can be seen that the block cutting algorithm is cutting the blocks efficiently. The difference in the point to point distribution and the distribution after the blocks are cut is due to the fact that the minimum number of points in all the directions, for any block is fixed at 10. Two small patches of level 4 points can be seen on the upper and lower outflow boundaries, in Figure 4-15. These patches are present due to skewness of the grid lines in these regions; which caused a drop in level at these points. The block cutting algorithm is able to detect this drop in the level in this region and performs cuts around this region very efficiently.

Table 4.2 shows the speed- up data from using the MTSAB scheme. The table shows the actual, theoretical and ideal speed ups. The table also gives the number of points in each level for: point to point distribution, with block cutting and without cutting

the blocks. The ideal speed up, which is based on the point to point distribution of the levels in the grid, is 6.85. The theoretical speed up without cutting the blocks is 2.46.

The number of points at level 1 has the greatest effect on reducing the theoretical speed up. And the effect decreases as the level of the point decreases. This is show by the Eqns. (3.18, 3.19, and 3.20) in Chapter 3. It can be seen from Table 4.2 that in the ideal distribution, the number of points at levels 1 and 2 are 669 and 7872 respectively. The number of points in level 1 and 2 without cutting the blocks is 16463 and 28684 respectively. This has a major impact in the difference between the ideal speed up and the theoretical speed up without cutting the blocks. The actual run time speed up without cutting the blocks in 2.46. The difference between the actual and the theoretical speed up is around 1.6%.The difference is due to the overhead from the buffer block calculations.

From table 4.2 it can be seen that after the blocks are cut, the theoretical speed up is 6.27. Looking at the distribution of points at different levels it can be seen that the block cutting algorithm tries to achieve the ideal distribution for this case. The restriction of 10 points in any direction for a block is the reason for the difference in these distributions. The actual speed up after cutting the blocks is 4.81. The difference between the actual and the theoretical has increase to 23%. This is because, after cutting the blocks the number of blocks has increased from 16 to 67. This increased the overhead from the buffer block calculations.

Without breaking blocks, the MTSAB scheme was 2.5 times faster than the RK56 scheme for this run.

At the time of calculation of the results using the MTSAB scheme, the block cutting algorithm was still under development. The blocks were cut manually, or in other words

the cut locations were hardwired into the BASS code. The grid blocks and their levels after the blocks are cut is shown in Figure 4-19. The block cutting that was done manually was not as efficient as the automated block cutting and theoretical speed up was around 4.0 and the actual speed up was around 3.0. The results presented for the MTSAB scheme with block breaking are for the manual block breaking. Currently the input gust section of the ACRAD boundary is under work in BASS code. This is the reason for not running this particular case again with the automated block cutting.

	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Grid Points in Level 4	Grid Points in Level 5	Grid Points in Level 6	Theoretical Speed up	Actual Speed up	Number of blocks
Ideal Distribution	669	7872	14295	17920	53720	0	6.85	-	-
Distribution Without Block Cutting	16463	28684	0	49329	0	0	2.5	2.46	16
Distribution With Block Cutting	1133	8281	14868	26726	43468	0	6.27	4.81	67

Table 4.2 Speed up data from using the MTSAB scheme for the Joukowski airfoil case

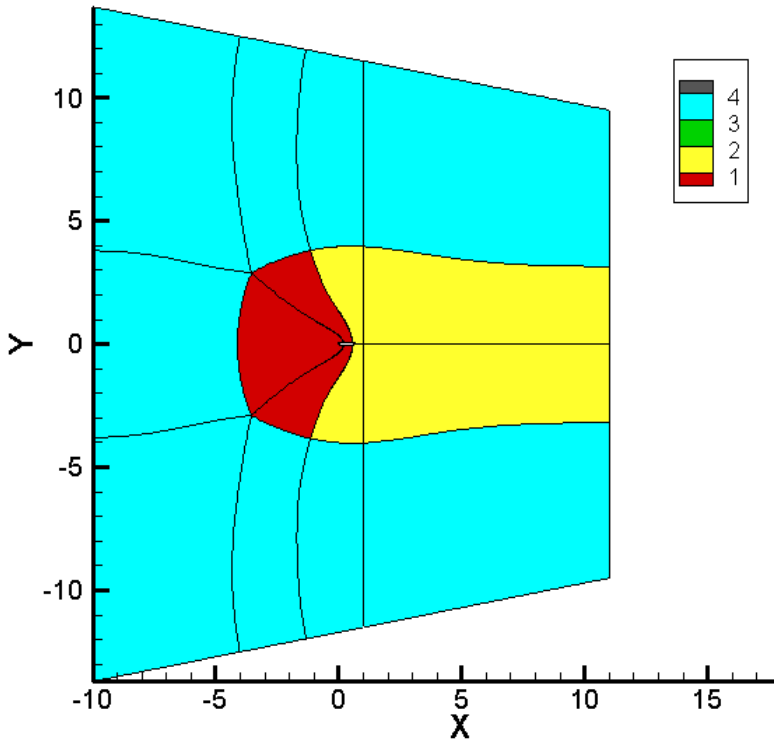


Figure 4-14: Distribution of levels without block cutting

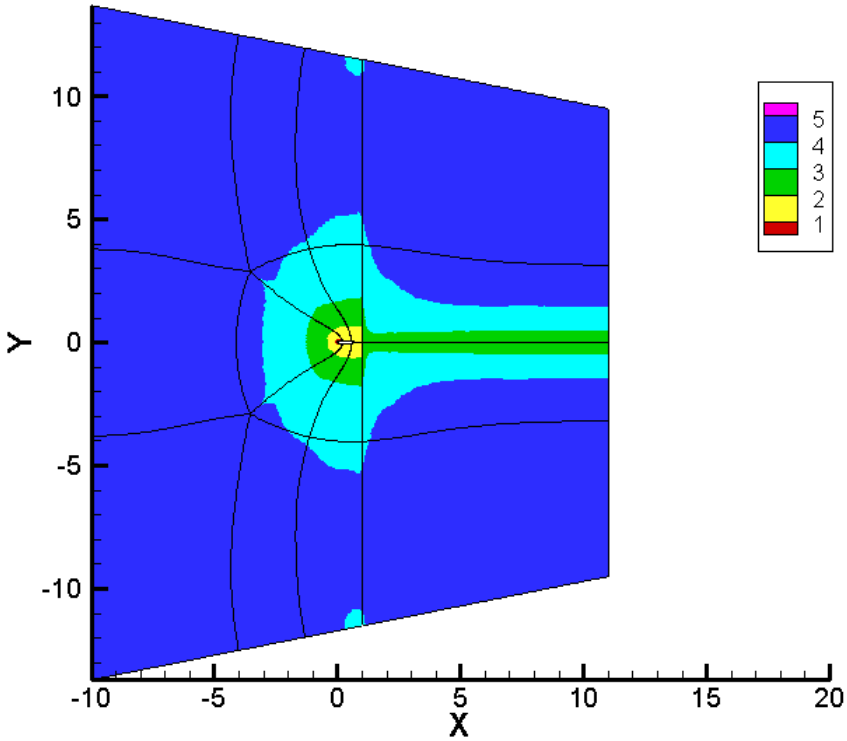


Figure 4-15: Point by point distribution of levels in the grid (Ideal)

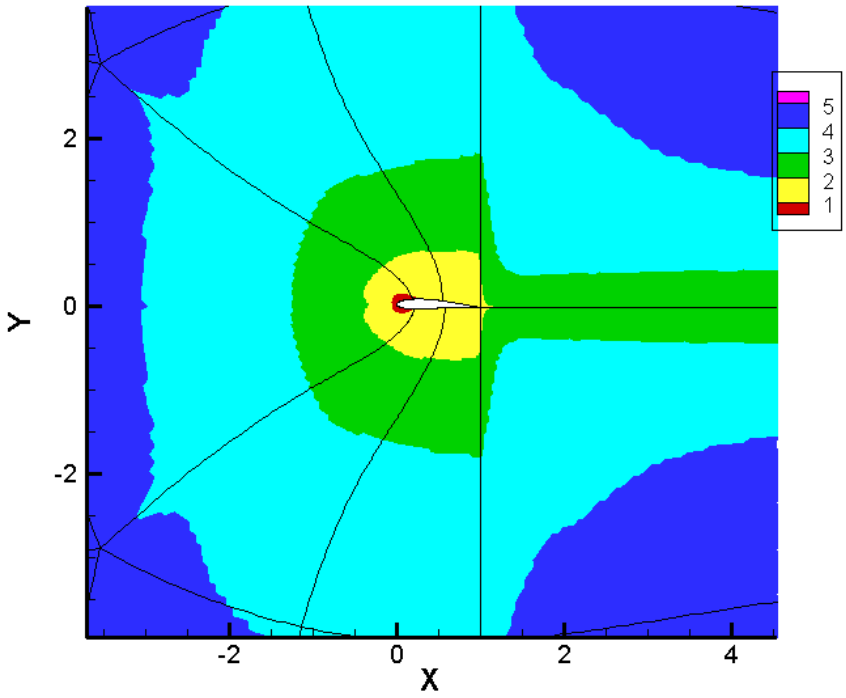


Figure 4-16: Point by point distribution of levels near the airfoil surface

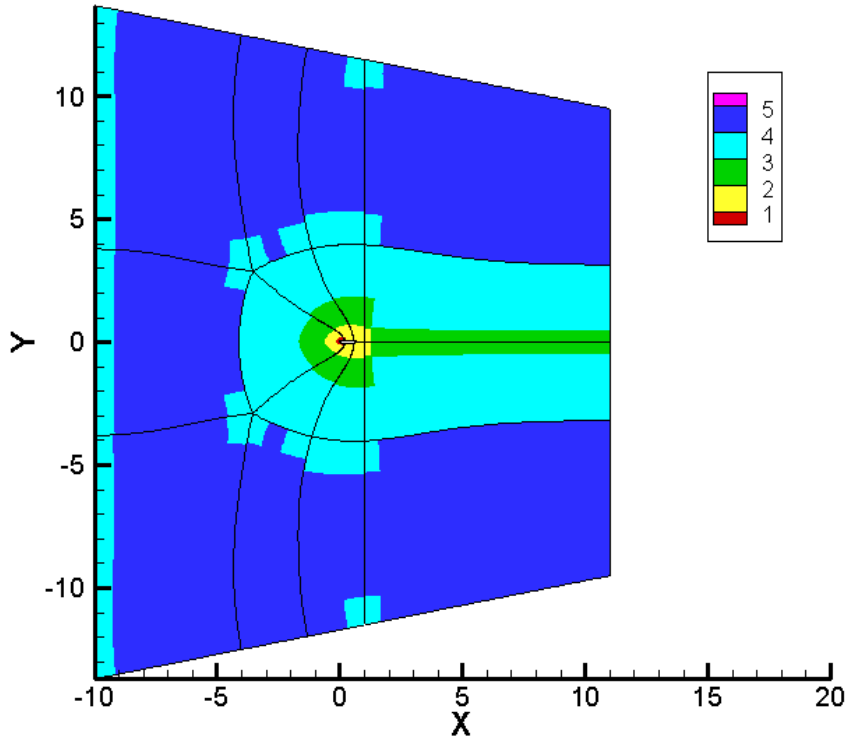


Figure 4-17: New blocks and their levels after the blocks are automatically cut

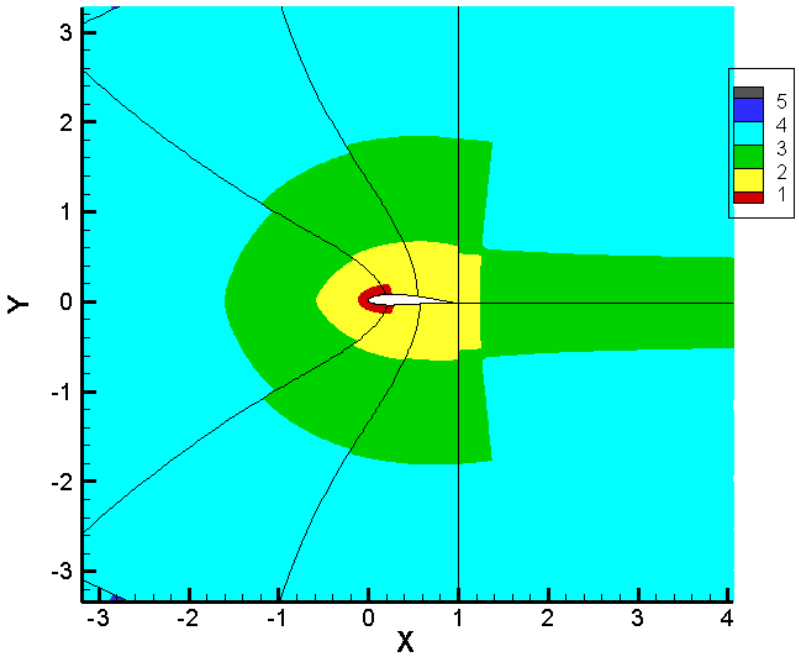


Figure 4-18: Levels of the new grid blocks near the airfoil surface after the original blocks are cut during the run.

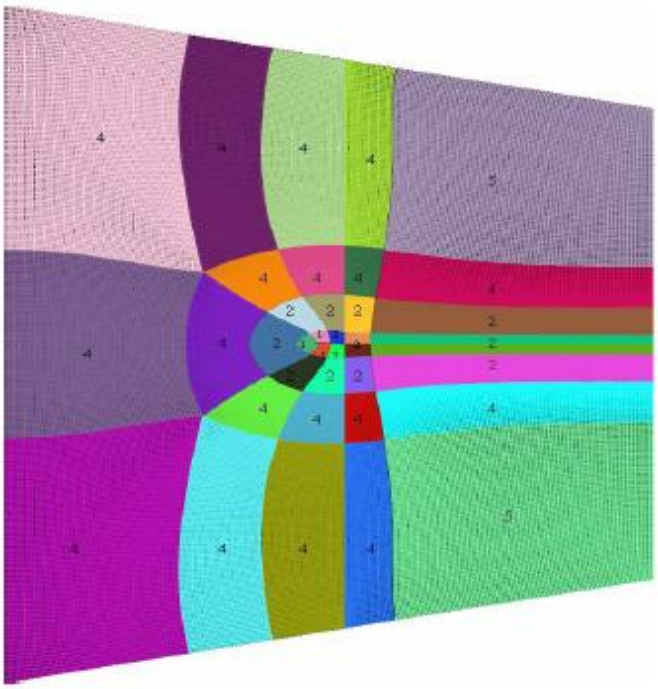


Figure 4-19: Levels of the new grid blocks after the blocks were cut manually

4.3.3 Results

Figures 4-20 to 4-25 compare the results for the single Adams-Bashforth scheme, MTSAB scheme (with manual block breaking and without breaking the blocks) and RK56 scheme, for both the gust frequencies.

Figures 4-20 and 4-23 show the mean pressure distribution on the airfoil surface for both the values of the reduced frequencies. It can be seen in these figures that, there is an excellent match in the mean pressure for all the schemes used, for both the gust frequencies.

Figures 4-21 and 4-24 show the RMS pressure distribution on the airfoil surface for both the values of the reduced frequencies. It can be seen in these figures that there is a good match between the single step Adams-Bashforth scheme and the RK56 scheme, for the RMS pressure distribution. The RMS pressure distribution from using the MTSAB schemes (with manual block breaking and without block cutting) varies slightly from the single step Adams-Bashforth scheme. This can be seen for both the values of the gust frequencies. The difference is more for the case with block cutting than without cutting the blocks.

Figures 4-22 and 4-25 show the plots of acoustic intensity in the near field on a circle of radius R=1, 2, 3, and 4 times the chord length of the airfoil, for both the values of the gust frequencies. There is an excellent match in the acoustic intensities for both; single step Adams-Bashforth scheme and the RK56 scheme. The MTSAB schemes, with and without cutting the blocks vary slightly from the single time stepping schemes.

The differences can be attributed to fact that the grid points at a bigger time steps when using a MTSAB scheme.

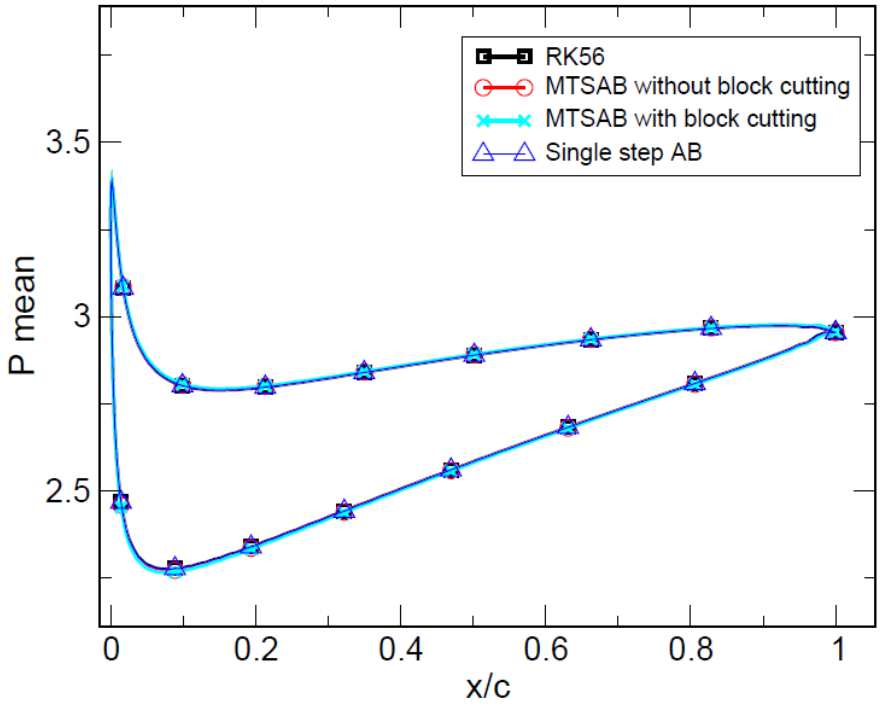


Figure 4-20: Mean pressure distribution on airfoil surface for k=1.0

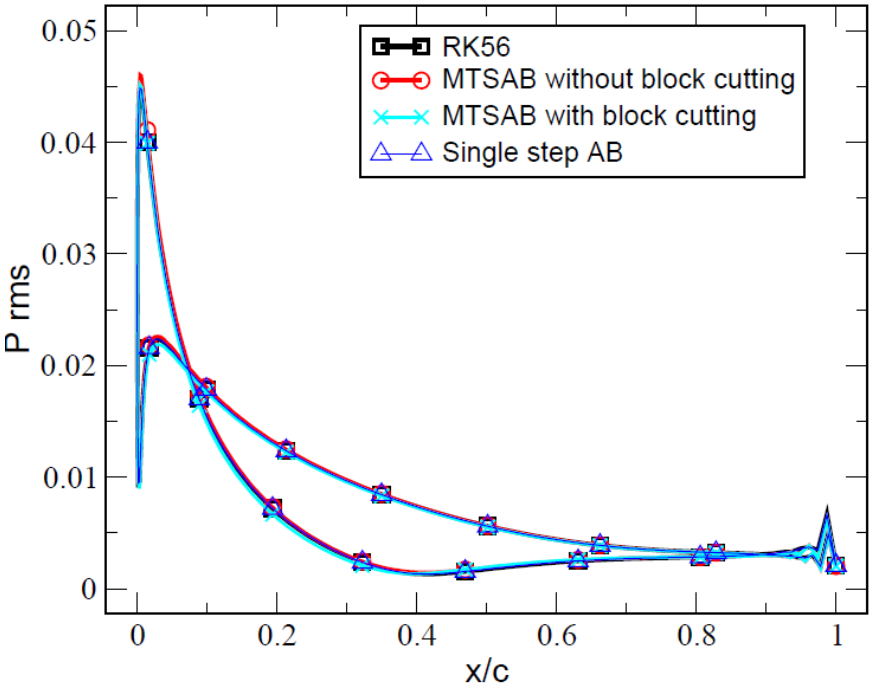


Figure 4-21: RMS pressure distribution on airfoil surface k=1.0

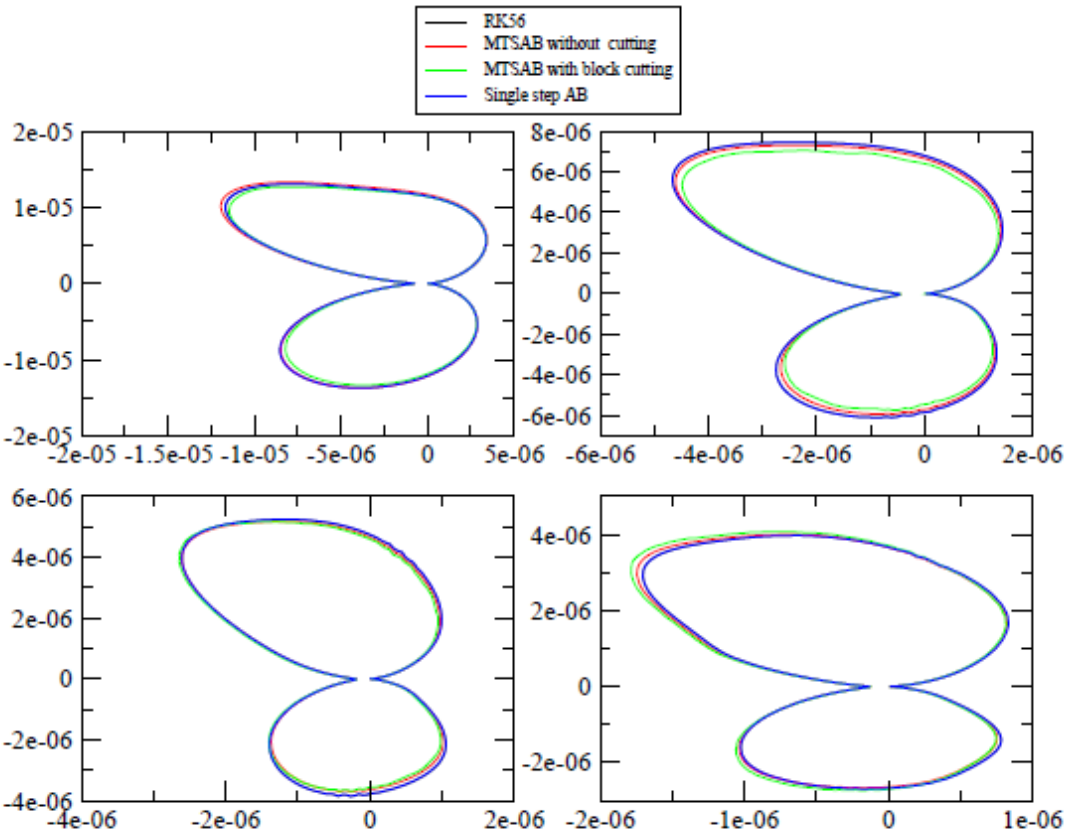


Figure 4-22: Near Field Acoustic intensities for k=1.0

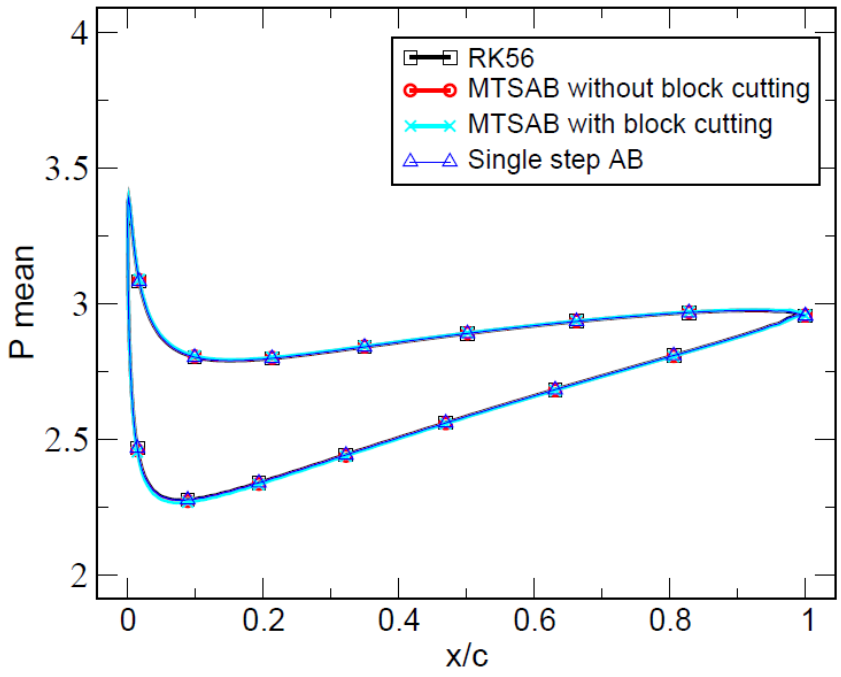


Figure 4-23: Mean pressure distribution on airfoil surface for k=2.0

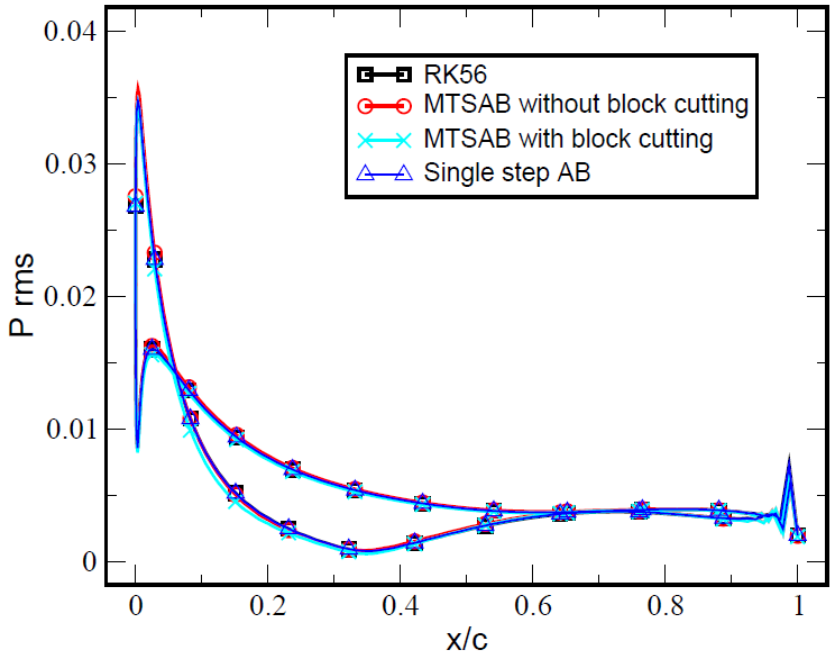


Figure 4-24: RMS pressure distribution on airfoil surface for k=2.0

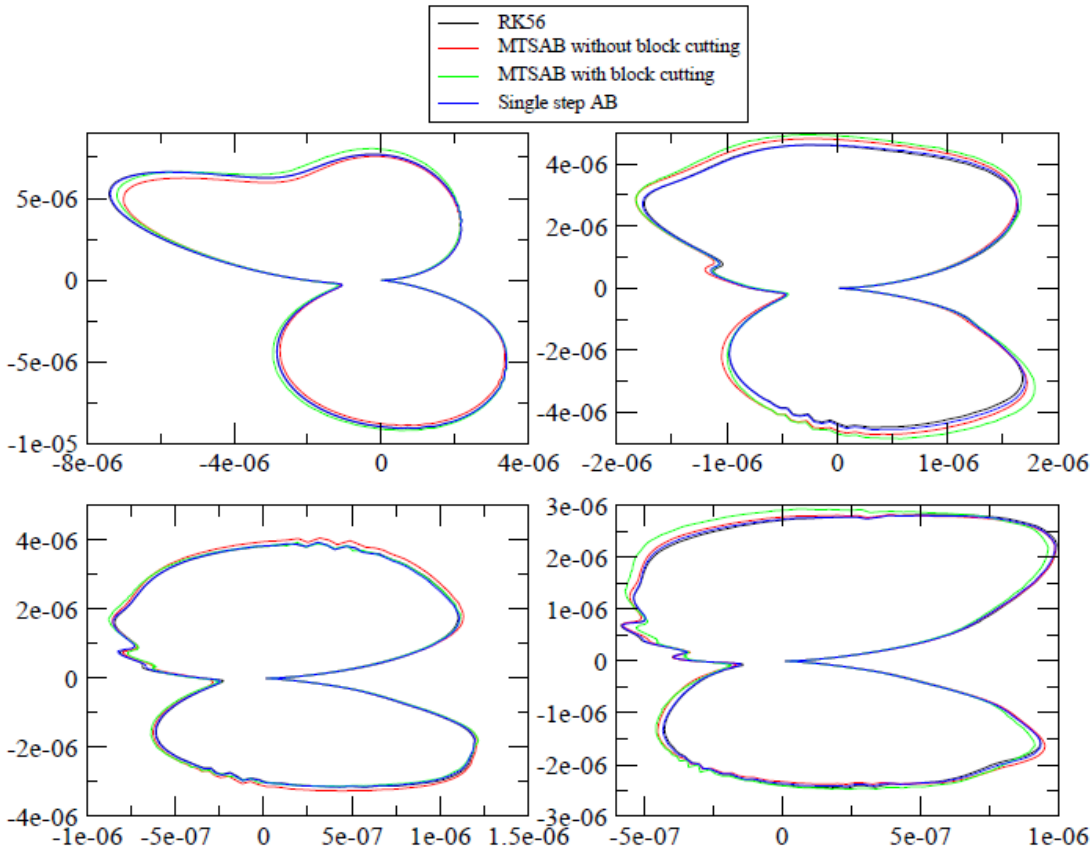


Figure 4-25: Near Field Acoustic intensities for k=2.0

4.5 Fan-Stator with Harmonic Excitation by Rotor Wake

The Category 4 problem from the Third CAA Workshop [13] was chosen to test the MTSAB for three dimensional problems in BASS code. In this problem, a stator row consisting of 24 infinitely thin flat plates are mounted in a constant radius annulus (Figure 4.26). A uniform axial mean flow convects the wakes from an upstream 16-blade rotor through the stator row, and noise is generated when these wakes impinge on the stators.

In cylindrical co-ordinates, the wake is given as,

$$v(r,\Phi,x,t) = U\sum_{n=0}^{\infty} V_n\, e^{inB/(\frac{\Omega x}{U}+\Phi-\theta(r)-\Omega t)} \tag{4.7}$$

Where U is the axial flow speed, $a_o M_\infty$ and Ω is the rotor angular velocity. Only the blade passing frequency is considered (n=1) with up wash amplitude equal to 0.1. Following (13),

$$(V_{nx}, V_{n\theta}, V_{nr}) = a_n U(-\frac{U}{\Omega r}, 1, 0) \tag{4.8}$$

where a_n is in general complex. The function giving the radial dependence is:

$$\theta(r) = \frac{2\pi q}{B}\frac{r-h}{R-h} \tag{4.9}$$

In Eq. (4.9), q is the wake parameter; for $q = 0$ the excitation is in phase from root to the tip of the stator. When $q = 3$, there are *3* wakes intersecting each stator vane. All calculations were performed at tip Mach number $M_T = 0.783$. BASS Code was tested by Sescu. et.al [24] for q values ranging from *0.0* to *3.0*.The value of $q = 3.0$ was chosen to test the MTSAB scheme.

4.5.1 Grid and Numerical Details

The computational grid was generated using the GridPro [31] package. A single passage grid was generated, and then „stacked' in the azimuthal direction. The grid has a minimum of 10 points per wavelength in all the three co-ordinate directions.

Axially, the grid begins 1.5 chord lengths upstream of the stator leading edge and extends 6 chord lengths downstream of the stator trailing edge. The sponge layer begins from, 2 chord lengths from trailing edge of the flat plate to the outflow boundary. The flat plate stators have no thickness in the grid, which gives rise to grid singularities at the leading and the trailing edge points of the stators. The grid is clustered about these points to resolve the sharp flow gradients which are generated at these locations. Due to the periodicity of the test problem, the computational domain included only three of the 24 flat plate stators (1/8 of the full annulus).Periodic boundary conditions were specified in the azimuthal direction. The grid has a total of 526,752 grid points.

The wake given by Eq. (4.16) is imposed at the inflow boundary. Details of the gust imposition can be found in [24]. Giles boundary condition [34] is used at the inflow and outflow. The equations are designed such that the outgoing waves are absorbed with no reflection, but this is not always true.

At the outflow boundary, a sponge layer combined with grid stretching to absorb the unwanted spurious waves that could contaminate the flow domain [35, 36].

4.5.2 MTSAB performance

Based on the stability limit, CFL of the MTSAB scheme was set to 0.31. Figure 4-27 shows the complete grid and the grid near the flat plate. Figure 4-28 shows the distribution of the levels in the grid before using the blocks cutting algorithm.

It can be seen from this figure that there are four levels in this grid. As expected, the blocks which are at the leading and the trailing edges are at level 1 and the levels of the blocks increase as they are located away from the leading and the trailing edges.

Figure 4-29 shows the point to point distribution of the levels in the grid. It can be seen that most of the points in the grid are at level 5. The points at the inner radius of the annulus are at level 4 and the levels of the points increase to 5 away from the inner cylinder. The points clustered near the plate are at a lower level as expected.

Figure 4-30 shows the distribution of the grid points after the blocks are cut. As already discussed, the minimum number of points for any given block, in every direction is 10. The block cutting for this case was restricted by an additional constrain for this case. The blocks could not be successfully cut on the Giles inflow and outflow faces due to a technical problem in the code. Currently, work is being done to resolve this issue. Looking at the levels near the plate in Figure 4-30 it can be seen that blocks are getting cut successfully at the interior of the computational domain.

Table 4.3 shows the speed up data from using the MTSAB scheme. The ideal speed up, which is based on the point to point distributions of the levels in the grid, is 6.85. The theoretical speed up without cutting the blocks was 2.66 and the actual speed up was 2.25.

After the blocks are cut during the run, the theoretical increased to 3.02 and the actual speed up increased slightly to 2.28. The difference in the actual and the theoretical speed up is due to the overhead from the buffer block calculations. The number of blocks after using the block cutting algorithm, increased from 156 to 234. This added to existing overhead from the buffer block calculations. The big difference in the theoretical speed up after the blocks are cut and the ideal speed up is due to the restrictions imposed on the block cutting (as already discussed).

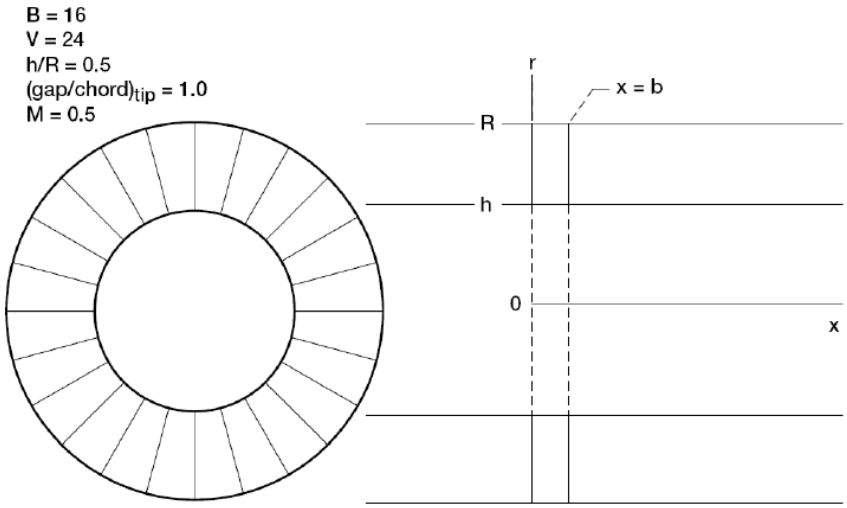


Figure 4-26: Configuration of the blades in category 4 problem from the 3rd CAA Workshop

	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Grid Points in Level 4	Grid Points in Level 5	Grid Points in Level 6	Theoretical Speed up	Actual Speed up	Number of blocks
Ideal Distribution	669	7872	14295	17920	53720	0	6.85	-	-
Distribution Without Block Cutting	54684	196602	82026	193440	0	0	2.66	2.25	156
Distribution With Block Cutting	54684	140988	76998	225282	28800	0	3.02	2.28	234

Table 4.3 Speed-up data for the MTSAB scheme

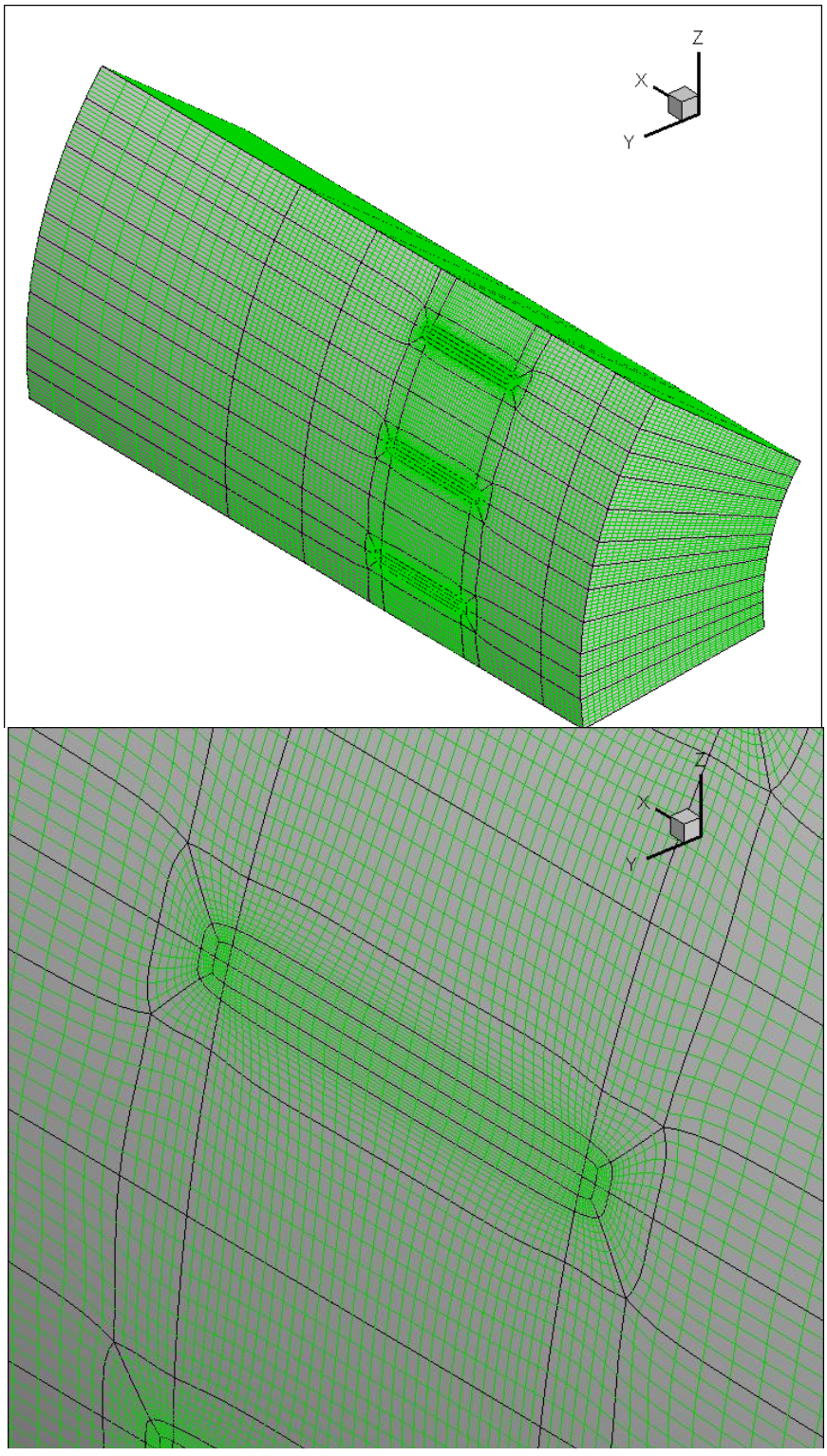


Figure 4-27: Grid for the 3D Workshop Problem

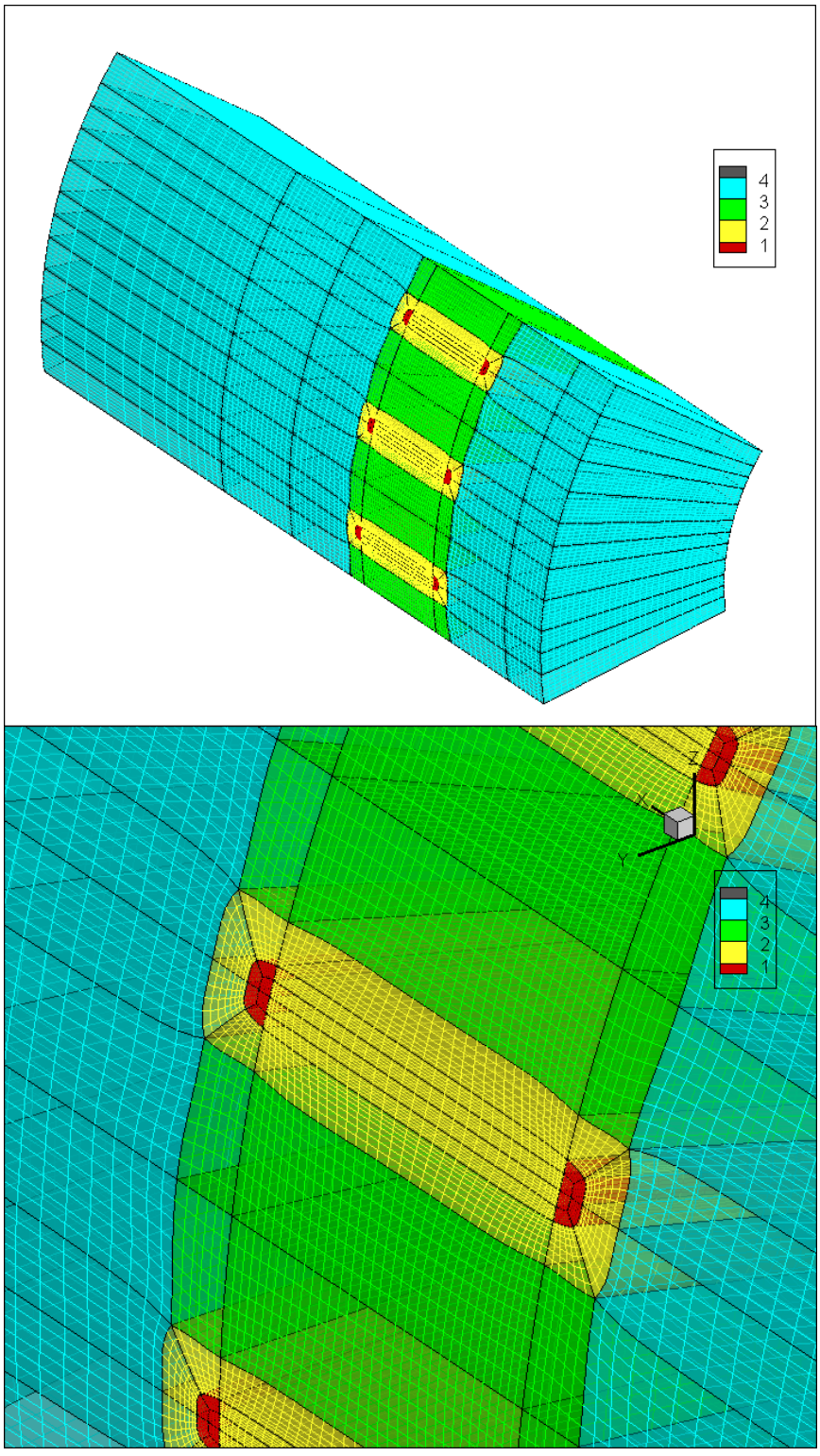


Figure 4-28: Distribution of levels before cutting blocks

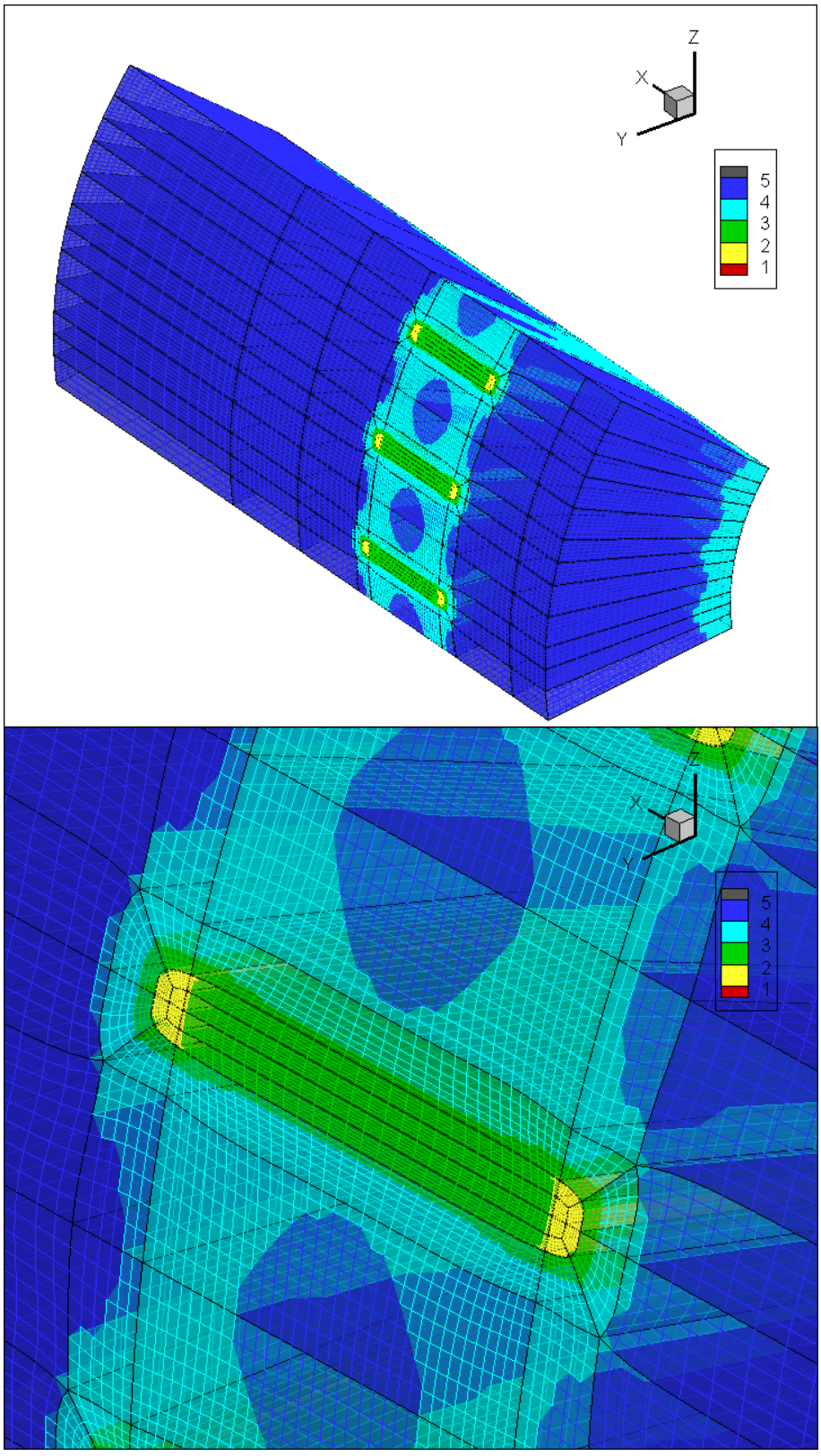


Figure 4-29: Point by point distribution of levels in the grid

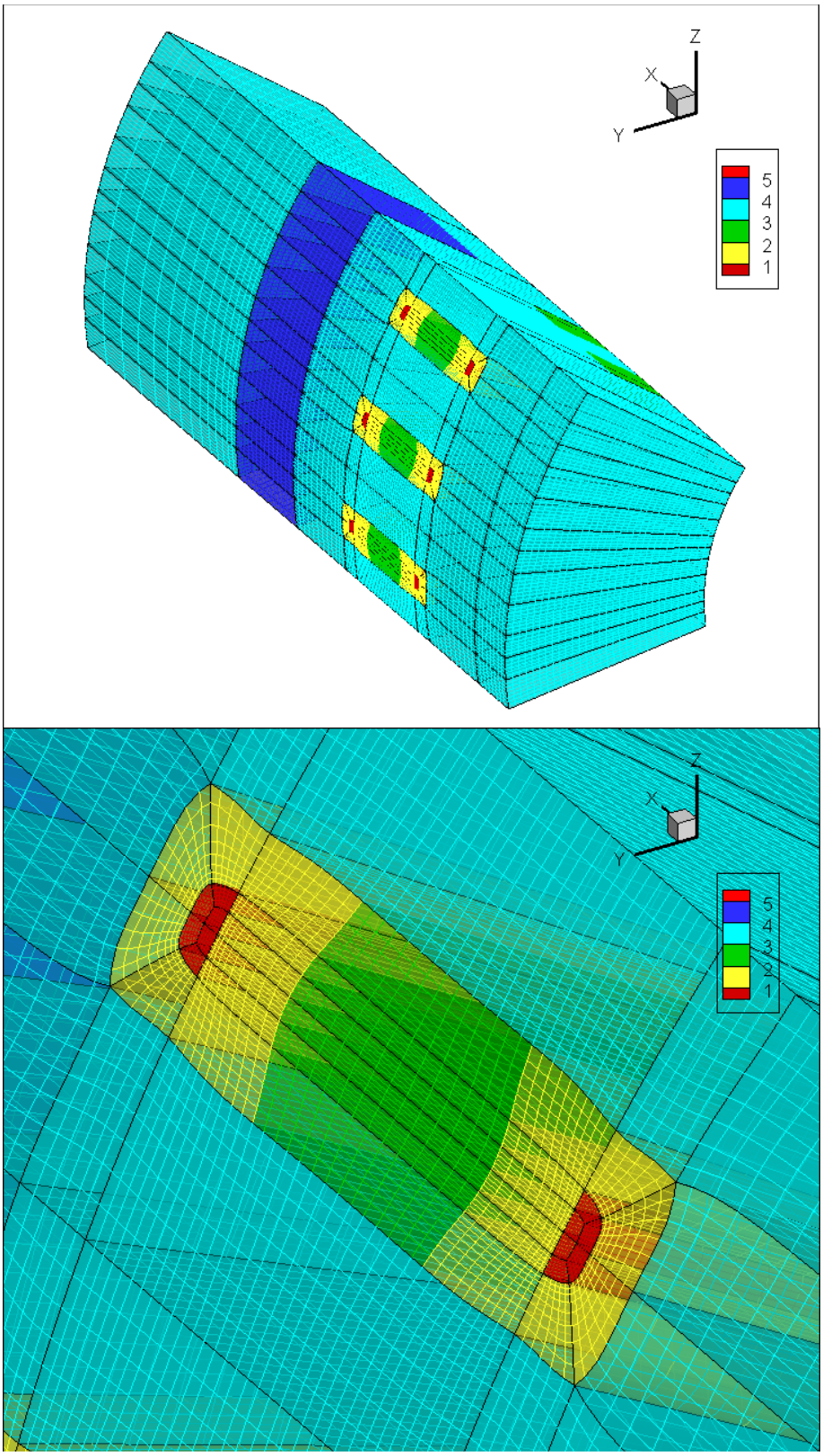


Figure 4-30: Distribution of levels after the block cutting algorithm is used

4.5.3 Results

The results are compared for the MTSAB scheme with automatic block cutting and the optimized HALE-RK67 scheme (Allampalli et. al. [18]). The CFL 1.5 was used for the RK67 scheme. The MTSAB scheme after block cutting was 1.8 times faster than the HALE-RK67 scheme.

Figure 4-31 show contours of instantaneous velocity magnitude. The effect of wake phase parameter can be seen in this figure. For *q=3.0*, the gust excitation at the hub leads that of the tip, as in the real fan wakes.

A pressure field modal expansion given by Tyler and Sofrin [37] is given as,

$$p_n(r,\Phi,x,t) = p_o \sum_{k=\infty}^{\infty} \sum_{\mu=0}^{\infty} A_{nm\mu}(x)\Psi_{\mathrm{m\mu}}(\mathrm{r})\mathrm{e}^{\mathrm{i(m\Phi-nB\Omega t)}} \qquad (4.10)$$

In Eq. (4.10) $A_{nm\mu}$ are the complex amplitudes and $\Psi_{\mathrm{m\mu}}(\mathrm{r})$ is the radial mode shape and $m=nB-Kv$, with k=0, ±1, ±2.

Figures 4-32 to 4-39 compare the real and imaginary parts of the complex pressure amplitude of the radial modes for *m*= -8 and *m*=16 and for $\mu = 0,1$ and 2. Comparison is made for both the schemes used, at two axial locations. The axial upstream location is at x=-b and the downstream location is x=2b, where b is the chord length of the plate.

In these Figures it can be seen that at the upstream location, for m=16 real part of the complex amplitude computed from the MTSAB scheme differs from the values calculated by the RK67 scheme. The difference is in the order of 10^{-6}. This can be attributed to the fact that the data computed for the MTSAB scheme has double precision.

The RK67 data had single precision. This is also true at the downstream location for the value of m=16.

For m=-8, at the upstream location, the MTSAB scheme slightly underpredicts the values of the real and the imaginary parts for $\mu = 0$ and slightly over predicts the values for $\mu = 1$

It can be seen from the results that at the downstream location, for m=-8 and $\mu = 1$ the MTSAB scheme predicts a slightly higher value than the RK67 scheme for both, real and imaginary parts of the complex amplitude.

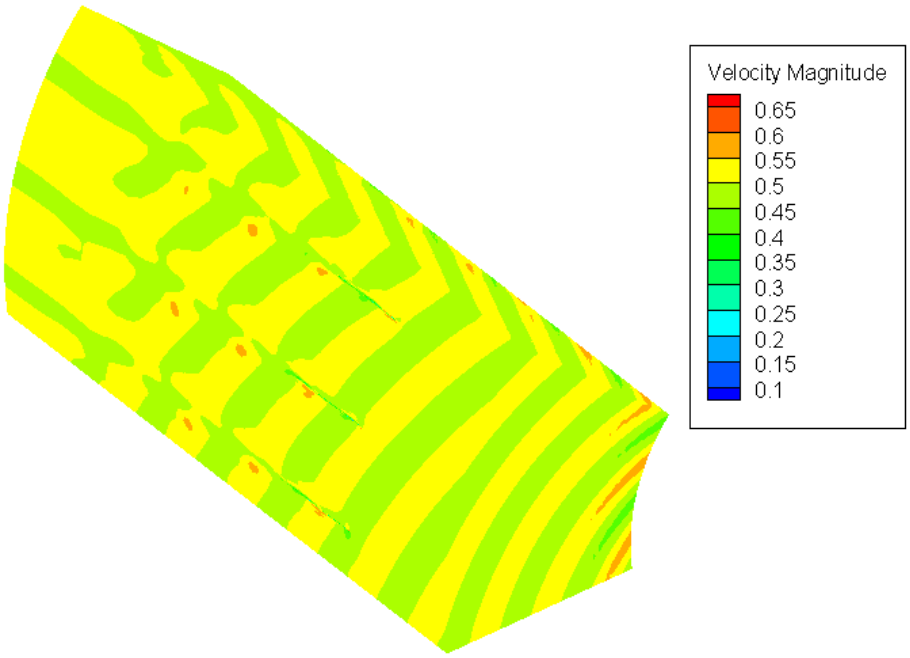


Figure 4-31: Figure shows the Instantaneous Velocity magnitude

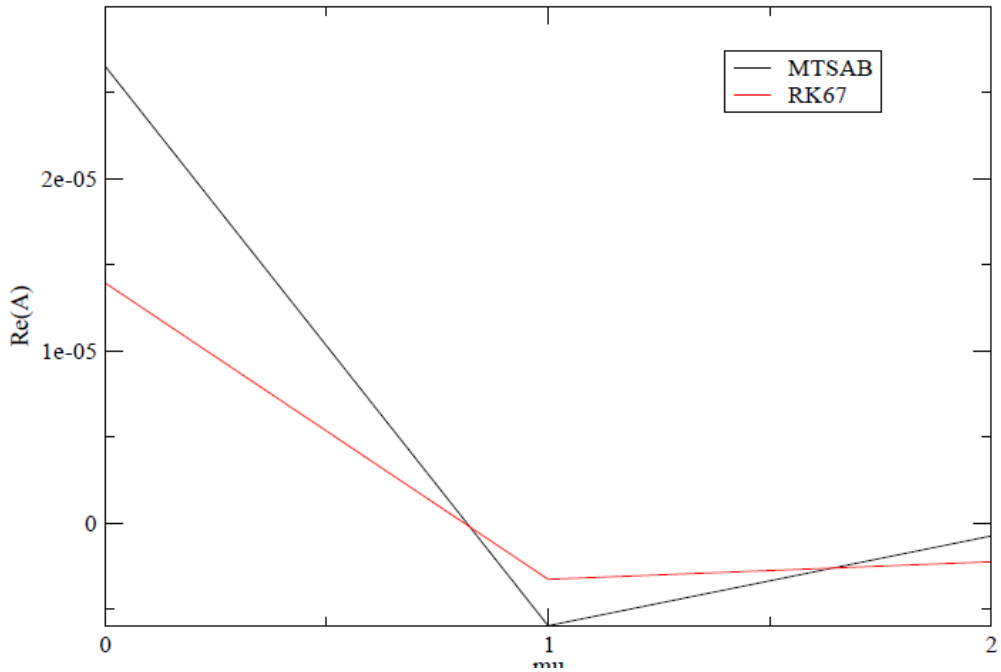


Figure 4-32: Real part of the Complex pressure at the upstream location; m=16

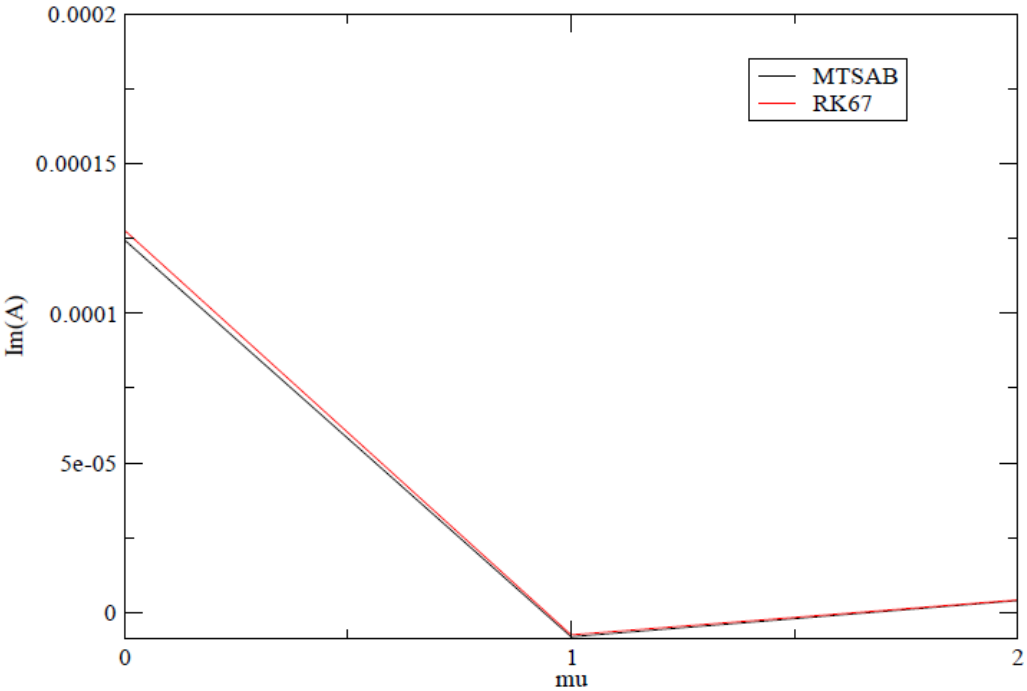


Figure 4-33: Imaginary part of the complex pressure amplitude at the upstream location; m=16

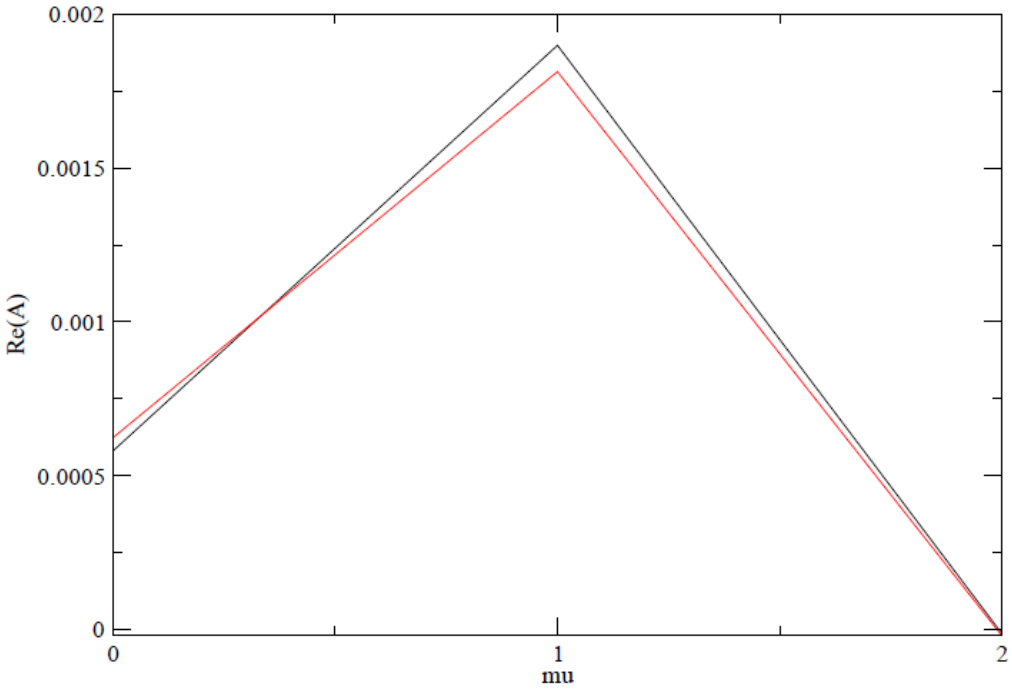


Figure 4-34: **Real part of the Complex pressure at the upstream location; m=-8**

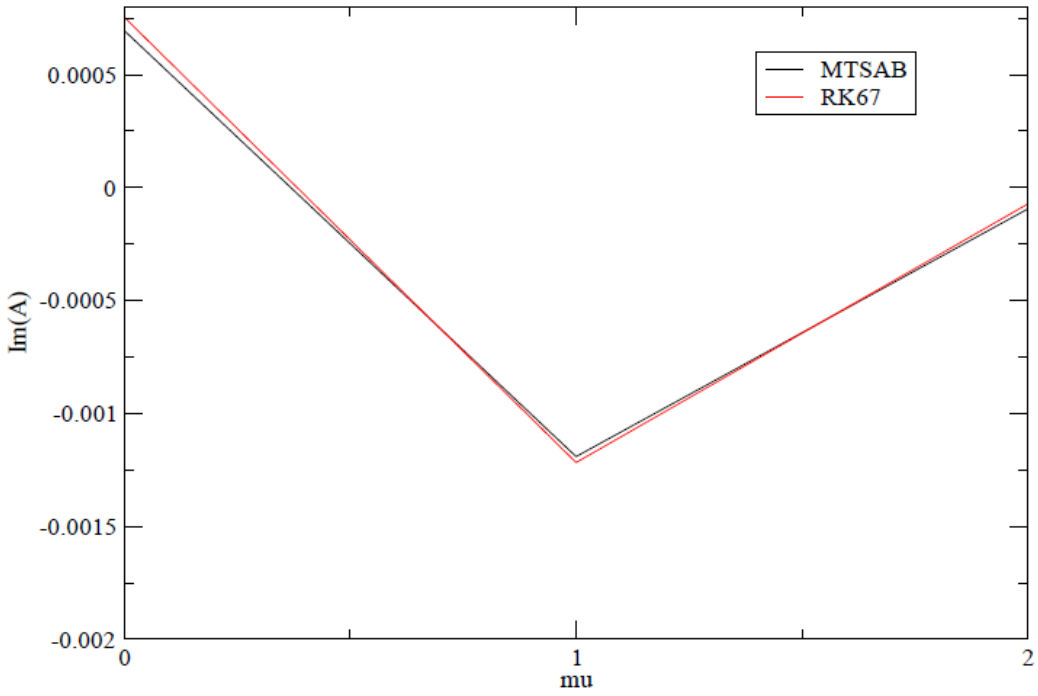


Figure 4-35: Imaginary part of the complex pressure amplitude at the upstream location; m=-8

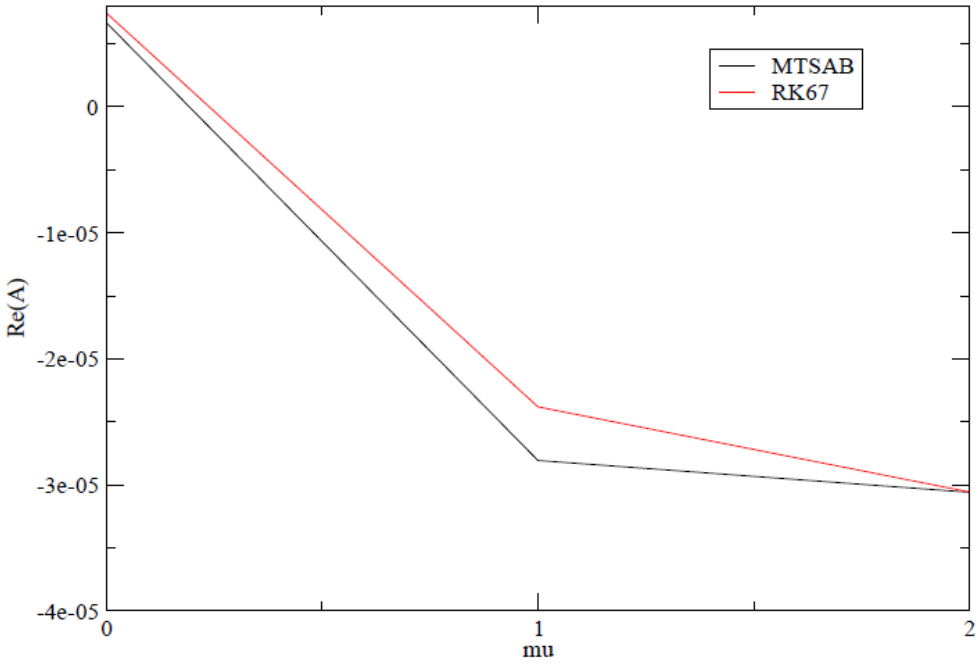


Figure 4-36: Real part of the Complex pressure amplitude at the downstream location; m=16

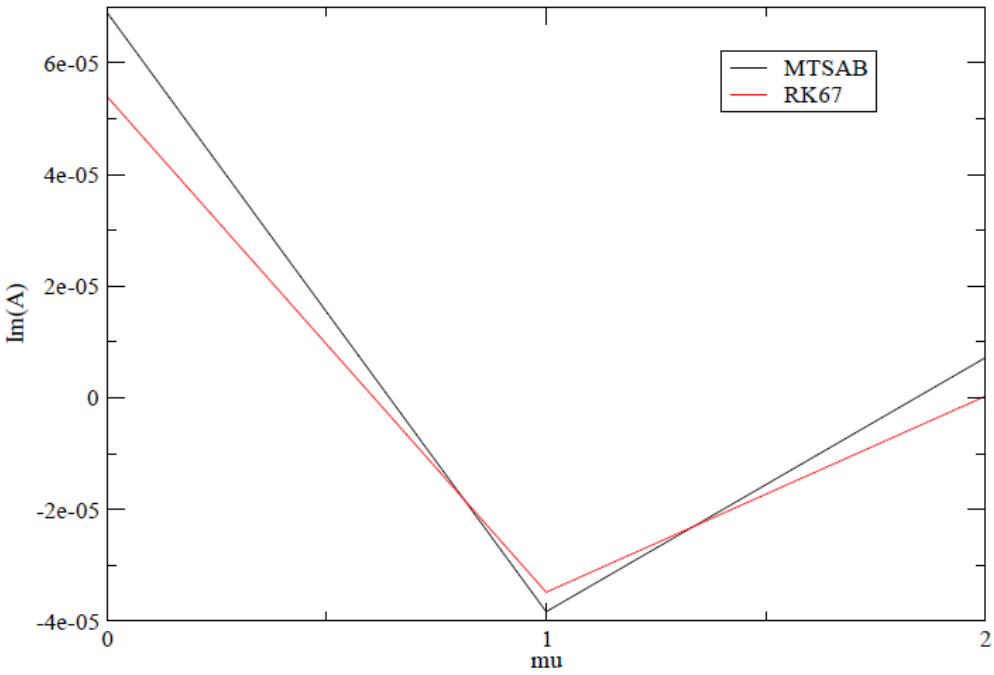


Figure 4-37: Imaginary part of the complex pressure amplitude at the downstream location; m=16

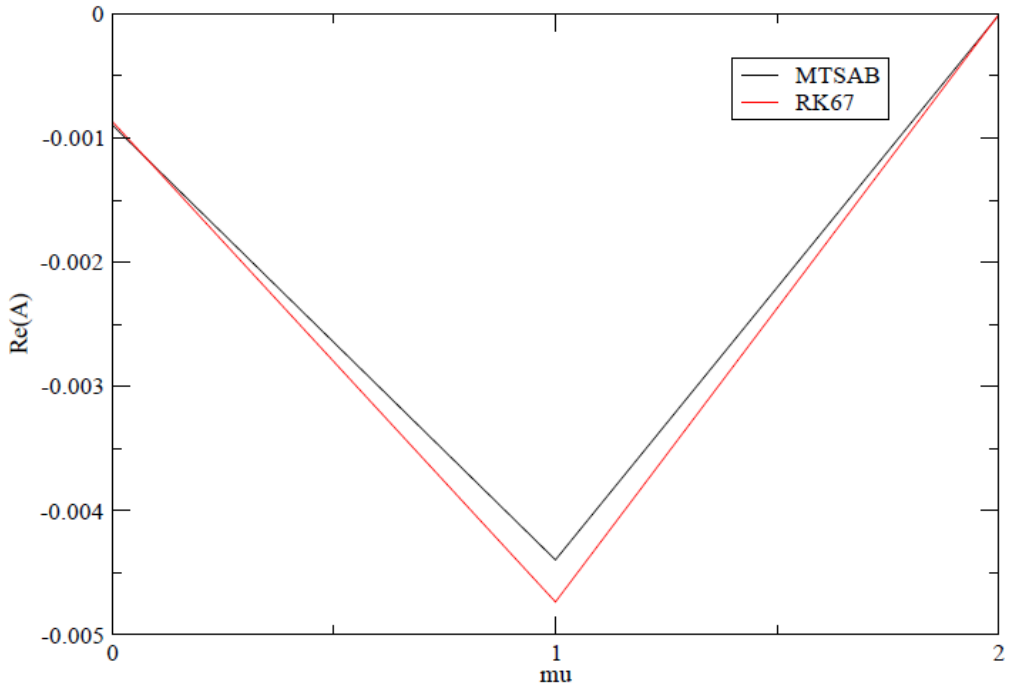


Figure 4-38: Real part of the Complex pressure amplitude at the downstream location; m=-8

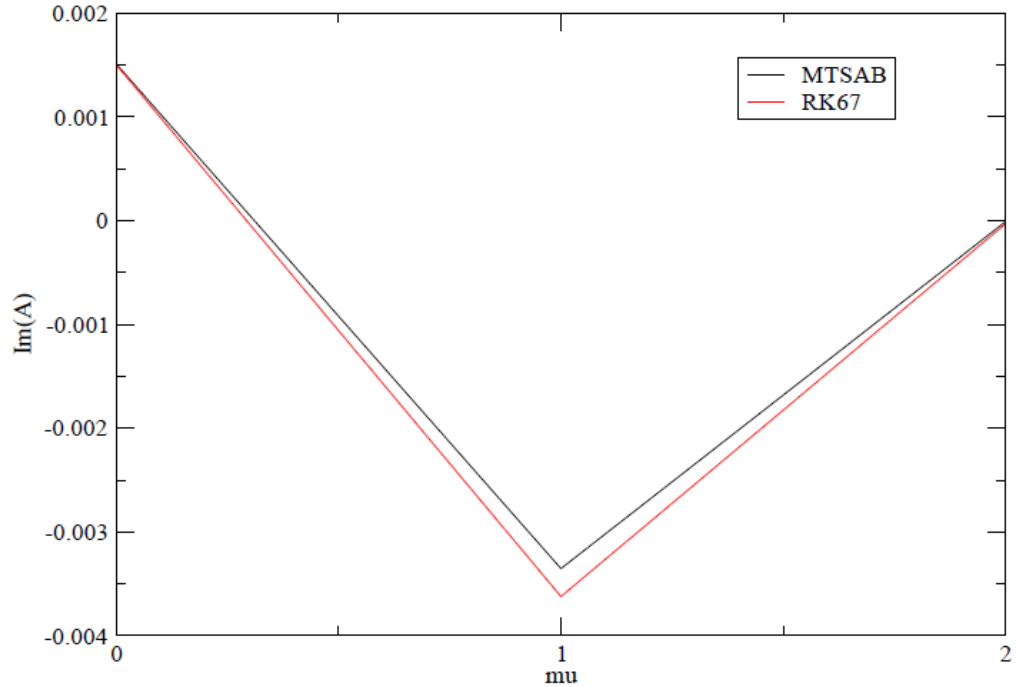


Figure 4-39: Imaginary part of the complex pressure amplitude at the downstream location; m=-8

Chapter 5

Application of Multi-Time-Stepping Adams-Bashforth Scheme to Transonic Flows

5.1 Introduction

Transonic flow occurs when there is mixed subsonic and supersonic local flow in the same flow field. Usually the supersonic flow region is terminated by a shock wave, allowing the flow to slow down to subsonic speed. The transonic problem is difficult because it is inherently nonlinear.

Multi-Time-Stepping Adams-Bashforth scheme in BASS code was tested for inviscid, transonic flows. The scheme was extended to be used with grid motion. The code, as discussed already, during the run calculates different time steps it chooses for different regions of the grid and changes this distribution automatically, as need arises. This automation is very useful, particularly for cases with grid motion. As the grid changes shape, there may be a change in local stable time step for some blocks. The MTSAB scheme automatically detects this and performs a Dynamic Level Redistribution, where it

reassigns the levels of the blocks during the run.

NACA0012 and NACA64A010 airfoils were used for the tests. Steady calculations were obtained for NACA0012 airfoil at Mach 0.63 and Mach 0.75, for an angle of attack of 2 degrees. Unsteady flow was calculated for the plunging NACA64A010 airfoil at Mach 0.8.

5.2 Shock Capturing

Shocks are characterized by discontinuities in the flow variables. In an inviscid flow, the shocks have no thickness. The very sharp flow gradients at such discontinuities may not be resolved by the spatial differencing scheme and the grid spacing used, causing non physical oscillations to be generated in the flow properties at the nearby grid points [23]. These oscillations may be large enough to cause the flow solver to predict negative values for the flow density, pressure or temperature. When this occurs, the flow calculation fails.

Several researchers studied this issue and proposed switchable dissipation schemes by which shocks and discontinuities could be directly computed as part of the flow solution (e.g., Refs. 38-43). Generally, these researchers defined a background dissipation scheme for use in the smooth regions of the flow, a more aggressive dissipation scheme for use near flow discontinuities, and a shock detection method to determine the presence and location of flow discontinuities. An extension of this method was developed for use in CAA schemes [23].This method combines explicit high accuracy background dissipation with a second order shock capturing dissipation. The cases presented in this work used

this particular shock capturing method to calculate steady and unsteady, inviscid, transonic flow over airfoils.

5.3 Grid Generation for Steady Flow Cases

MTSAB scheme was first tested for subsonic flow over the NACA0012 airfoil at Mach 0.63. The Mach number was then increased to 0.75 at which the flow is transonic. Two grids were generated for the steady flow over the NACA0012 airfoil with a rounded trailing edge. The grids for these cases were generated using the GridPro package [31].

Two grids were generated. The first grid (Grid 1) was used to calculate the steady flow solutions at Mach 0.63 and Mach 0.75. Figures 5-1 and 5-2 show the complete grid and the grid near the airfoil surface. Grid 1 has 40334 grid points.

For the Mach 0.75 case, an approximate shock location on the airfoil surface was obtained from running the case with Grid 1. A second grid (Grid 2) was generated with grid clustering at this approximate shock location, near the airfoil surface. This served two purposes. The first was to see the effect on shock resolution from a greater number of points on the airfoil surface, at the shock location. The second was to see the effect of changing the time step at the shock location.

Figure 5-3 shows the grid clustering on the airfoil surface for Grid 2. Grid 2 has a total of 44525 points. Both the grids generated extend to 24 chord lengths in the downstream direction and 10 chord lengths in remaining directions.

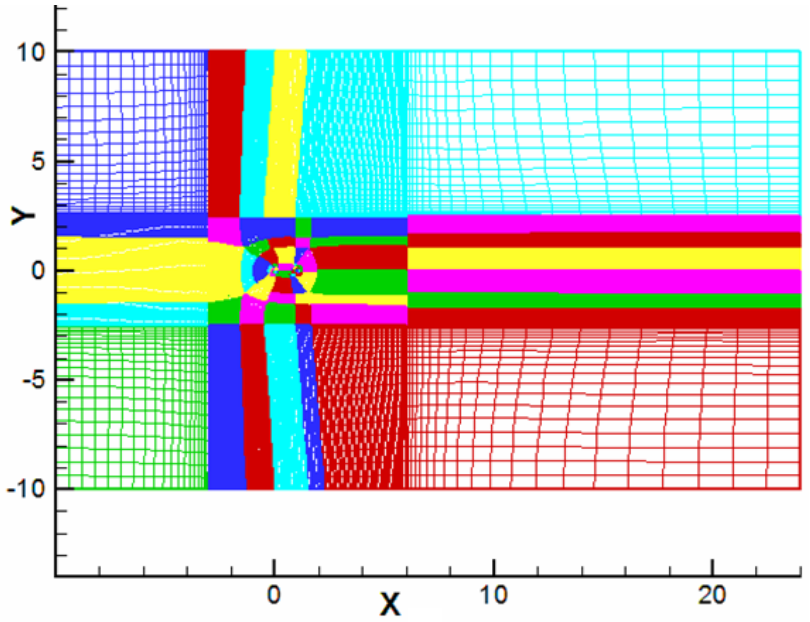


Figure 5-1: Complete grid for NACA0012 airfoil

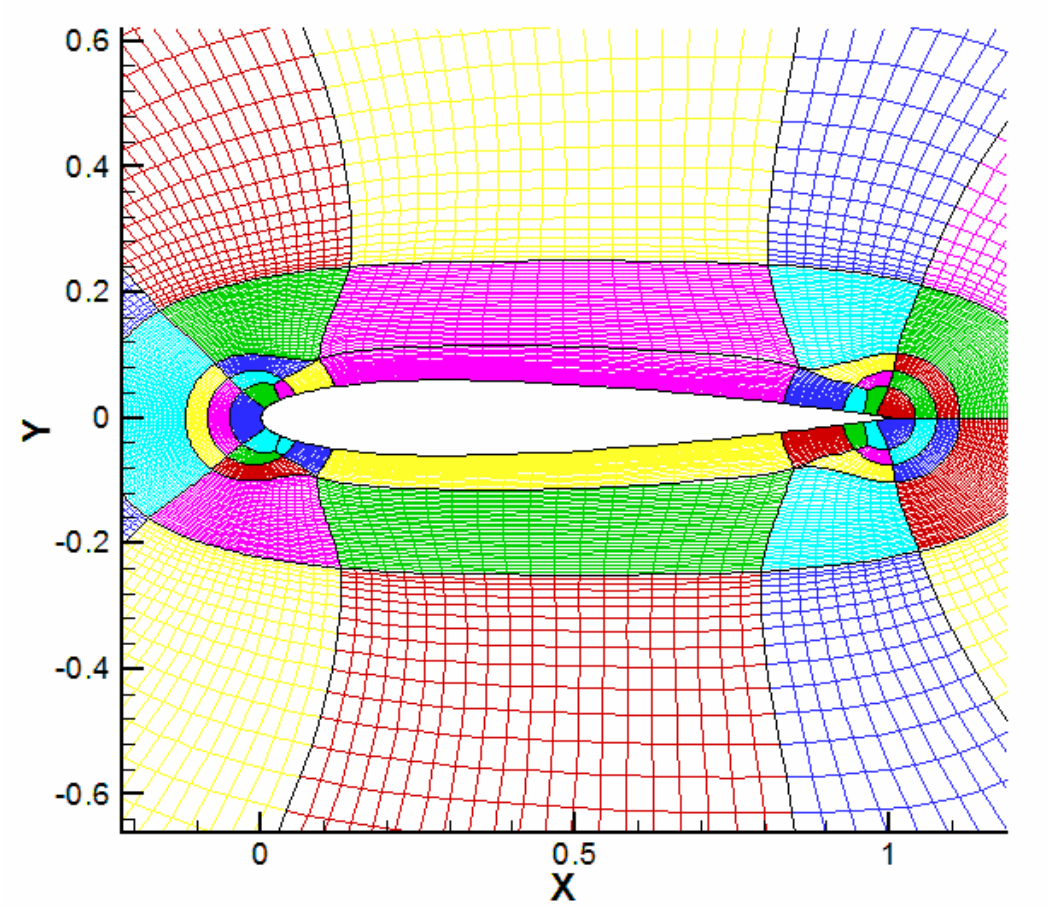


Figure 5-2: Grid near the airfoil surface (Grid 1)

Grid 1	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Grid Points in Level 4	Grid Points in Level 5	Grid Points in Level 6	Theoretical Speed up	Actual Speed up	Number of blocks
Ideal Distribution	131	2730	5522	6132	1626	24193	8.62	NA	-
Distribution Without Block Cutting	4715	1155	7056	5953	0	21455	4.76	3.87	94
Distribution With Block Cutting at Level 1	2365	3285	7276	5953	0	21455	5.57	3.48	114
Distribution With Block Cutting at Levels 1 and 2	2365	2955	7606	5953	0	21455	5.63	3.53	117
Distribution With Block Cutting at Level 1,2 and 3	2365	2955	4776	8783	0	21455	5.93	3.57	126
Distribution With Block Cutting at Level 1,2,3 and 4	2365	2955	4776	6673	420	23145	6.09	3.61	137
Distribution With Block Cutting at Levels 1,2,3,4 and 5	2365	2955	4776	6673	220	23345	5.99	3.62	139

Table 5.1 MTSAB Speed up Data for Grid 1

Grid 2	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Grid Points in Level 4	Grid Points in Level 5	Grid Points in Level 6	Theoretical Speed up	Actual Speed up	Number of blocks
Ideal Distribution	84	3189	6266	6738	2644	25604	8.81	-	-
Distribution Without Block Cutting	5881	1155	6479	7548	903	22559	4.55	3.67	120
Distribution With Block Cutting at Level 1	2750	3636	7129	7548	903	22559	5.52	3.46	143
Distribution With Block Cutting at Levels 1 and 2	2750	3206	7559	7548	903	22599	5.6	3.47	147
Distribution With Block Cutting at Level 1,2 and 3	2750	3206	5249	9858	903	22559	5.81	3.57	155
Distribution With Block Cutting at Level 1,2,3 and 4	2750	3206	5249	8168	903	24249	5.93	3.59	166
Distribution With Block Cutting at Levels 1,2,3,4 and 5	2750	3206	5249	8168	473	24679	5.94	3.6	168

Table 5.2: MTSAB Speed up Data for Grid 2

5.4 Numerical Details

The equations solved were the Euler equations. Optimized 4^{th} order DRP scheme of Tam and Webb was used for spatial differentiation. The shock capturing dissipation uses 2^{nd} order explicit dissipation, while the background dissipation uses the 10^{th} background dissipation [23]. The 2^{nd} order dissipation coefficient used for the MTSAB scheme was 0.08 and the 10^{th} order dissipation coefficient was 0.2. Values, greater than 0.2, for the

background dissipation coefficient makes the solver unstable. Thompson boundary conditions were used for the inflow and the outflow boundaries [46].

5.5 MTSAB Performance for Steady Flow Cases

The CFL of the MTSAB scheme was set to 0.31. Figure 5-4 shows the MTSAB levels in Grid 1 when the blocks are not cut. It can be seen from this figure that most of the region away from the airfoil is at level 6. Figure 5-5 shows the MTSAB levels near the airfoil surface for Grid 1, when the blocks are not cut. It can be seen from this figure that, blocks on the trailing and the leading edges of the airfoil, where the grid is clustered to resolve the stagnation points, are at level 1. The levels of the blocks away from the stagnation points increase as the grid is stretched out from the stagnation points.

Figure 5-6 shows the point to point distribution of levels near the airfoil surface for Grid 1. By comparing Figures 5-5 and 5-6 it can be seen that the blocks near the airfoil surface can be cut to generate new blocks at higher levels.

Figure 5-7 shows the distribution of levels in the grid after the block cutting algorithms was used during the run. It can be seen from this figure that, the blocks near the airfoil surface are cut during the run to get blocks at higher levels. The effect of the singularities on the levels can also be seen in this figure. There are two singularity points slightly away from the airfoil surface near the leading and the trailing edges of the airfoil. The level drops back to 1 in the regions surrounding the singularity points.

Figure 5-8 shows the levels of the blocks in Grid 2, before the blocks are cut. In Grid 2, as discussed earlier, the grid is clustered not only at the leading and the trailing edges to resolve the stagnation point but also at the approximate shock location (to improve the

shock resolution). Therefore the blocks, in the vicinity of the shock location are at levels 1 and 2.

Figure 5-9 shows the point to point distribution of levels near the airfoil surface for Grid 2 and Figure 5-10 shows the levels of the blocks after the block cutting algorithm cuts the original grid blocks during the run. By comparing these two figures it can be seen that block cutting algorithm cuts the blocks efficiently. The difference between these distributions is again from the fact that minimum number of points in every direction is fixed at 10.

In Grid 2, an interesting observation can again be made about singularity points again. Two singularity points were created in the process of clustering the points near the shock location (just like for the leading and the trailing edges). It can be seen in Figure 5-10 that, at the shock location, the region attached to the surface of the airfoil is at level 2.The level drops from 2 to 1 slightly above the airfoil surface and goes back to 2 again. The drop in the level is due to the presence of the singularity points.

Table 5.2 shows the distribution of the grid points at different levels for Grid 1. The table also shows the actual speed up, theoretical speed up and ideal speed. The ideal speed up, which is based on point to point distribution of levels, is 8.62. The number of blocks in the grid is 94. The theoretical speed up without cutting the blocks is 4.76 and actual run time speed up without cutting the blocks was 3.87. The difference between the actual and the theoretical speed ups is due to the overheard from the buffer block calculations.

Table 5.2 also shows the actual and theoretical speed ups after the block cutting was used at different maximum levels. It can be seen that when the blocks cut only at level 1, although the theoretical speed up increases to 5.57 the actual speed up drops to a value of 3.48. The number of blocks increased from 94 to 114. This drop in the actual speed up is due to the fact that the speed loss from the increased overhead from buffer block calculations overwhelms the speed gain from cutting the blocks at level 1. As the maximum levels at which the blocks are cut increases to 5, the theoretical speed up reaches 5.99. The actual speed up, after the initial drop, after block cutting at level 1, increases to a value of 3.26. It never crosses the value of the 3.87, which was the actual speed up when no block cutting was performed. The total number of block generated, after the blocks were cut at 5 levels was 139.

The drop in the actual speed up when the blocks were cut only at level 1 can be explained by a combination of two possible reasons. The first reason is that, the overhead from the buffer blocks can depend on what level the buffer block is at. For example, buffer blocks generated for a level 1 block performs more work that the buffer blocks that are generated for a level 6 block, for same number of points. This is because; buffer blocks generated for level 1 block performs calculations at every single time step. The second reason might be the number of points at level 1 interacting with other levels. In other words, for a given number of points at level 1, the number of grid points that communicate with other levels can dictate the amount of overhead from the buffer block calculations. Comparing Figures 5-5 and 5-7 it can be seen that the points at level 1 communicating with other levels has increased after the blocks are cut at that level.

Table 5-3 shows the speed up data for Grid 2. It can be seen from this table that the actual and the theoretical speed ups exhibit similar behavior for Grid 2 as it was for Grid 1.

Although there was a drop in the actual speed up from using the block cutting algorithm in both the grids, block cutting was used at all the levels to compute the results. This was done to test the accuracy scheme for a maximum value of the theoretical speed up.

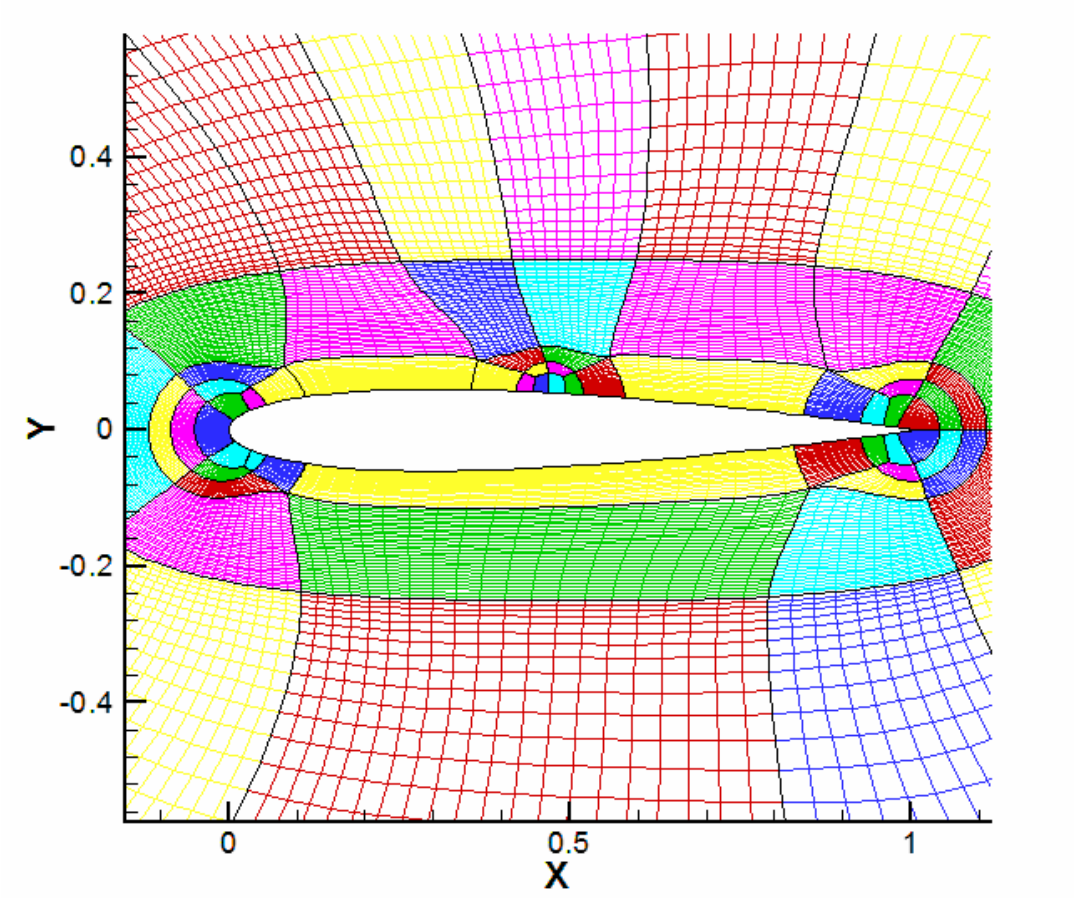


Figure 5-3: Grid clustering at approximate shock location (Grid 2)

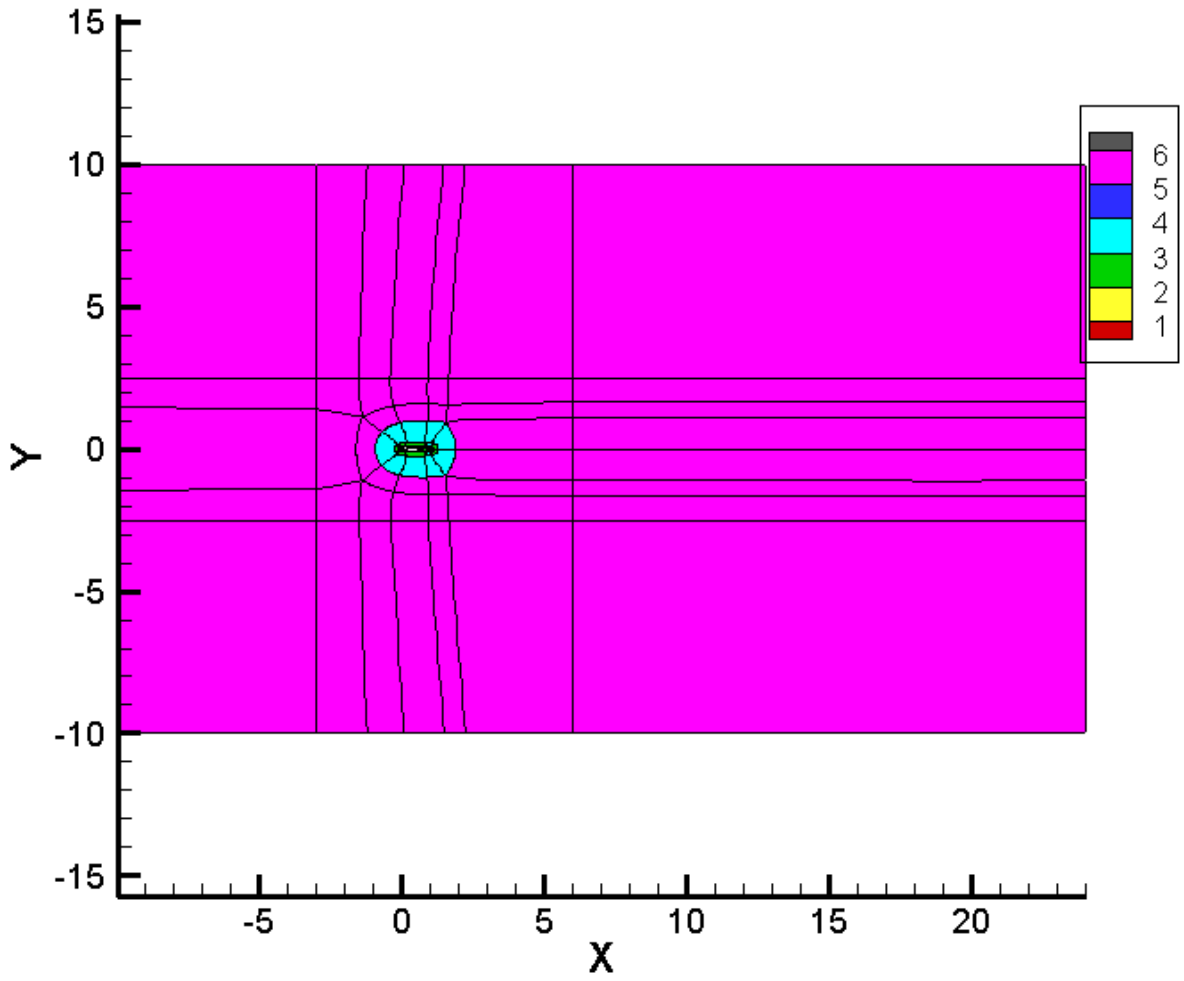


Figure 5-4: Distribution of the MTSAB levels in Grid 1 without block cutting

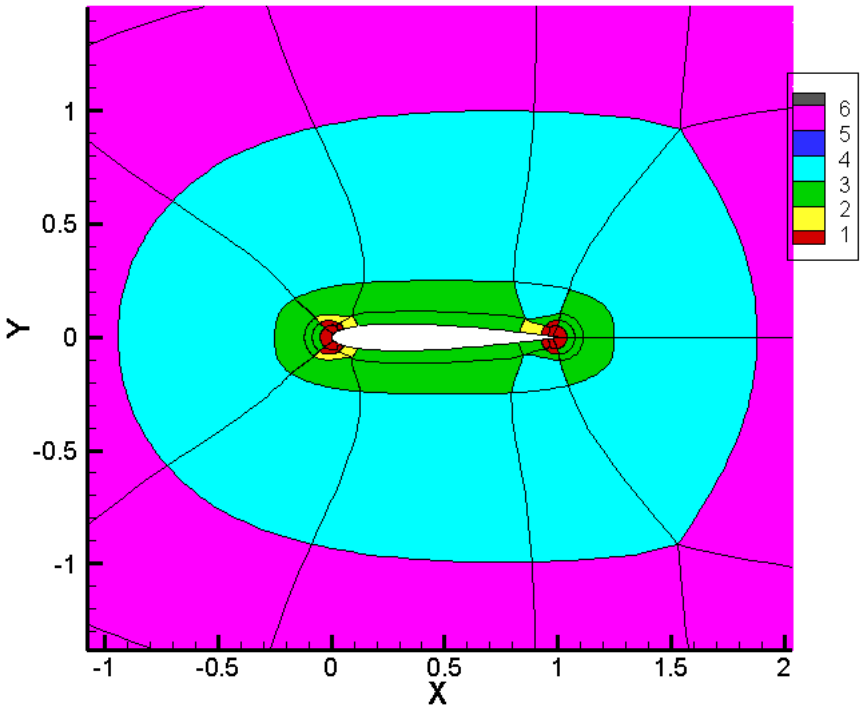


Figure 5-5: Distribution of the MTSAB levels near the airfoil (in Grid 1) without block cutting.

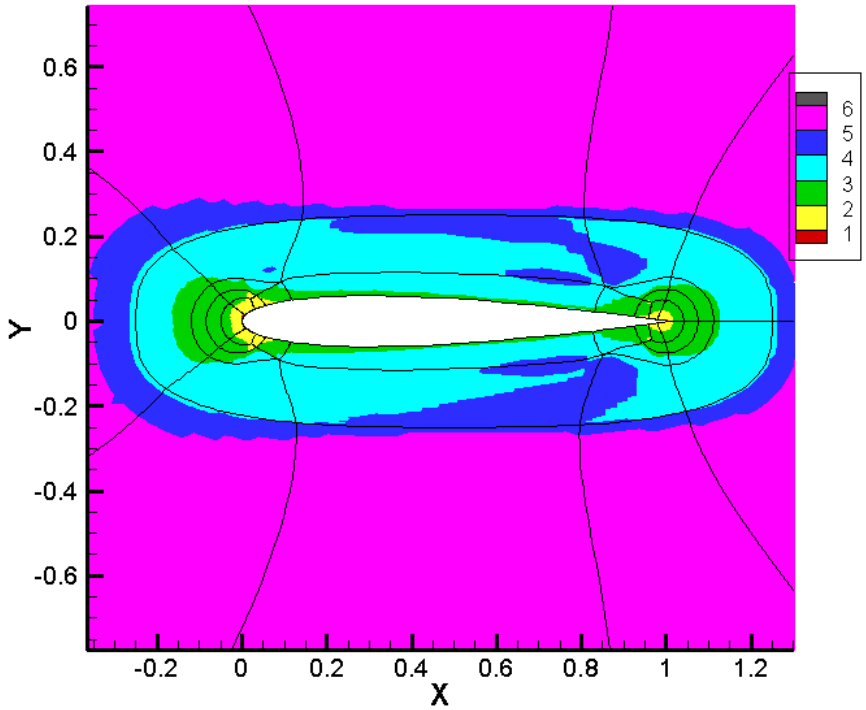


Figure 5-6: Point by point distribution of the MTSAB levels in Grid 1 near the airfoil

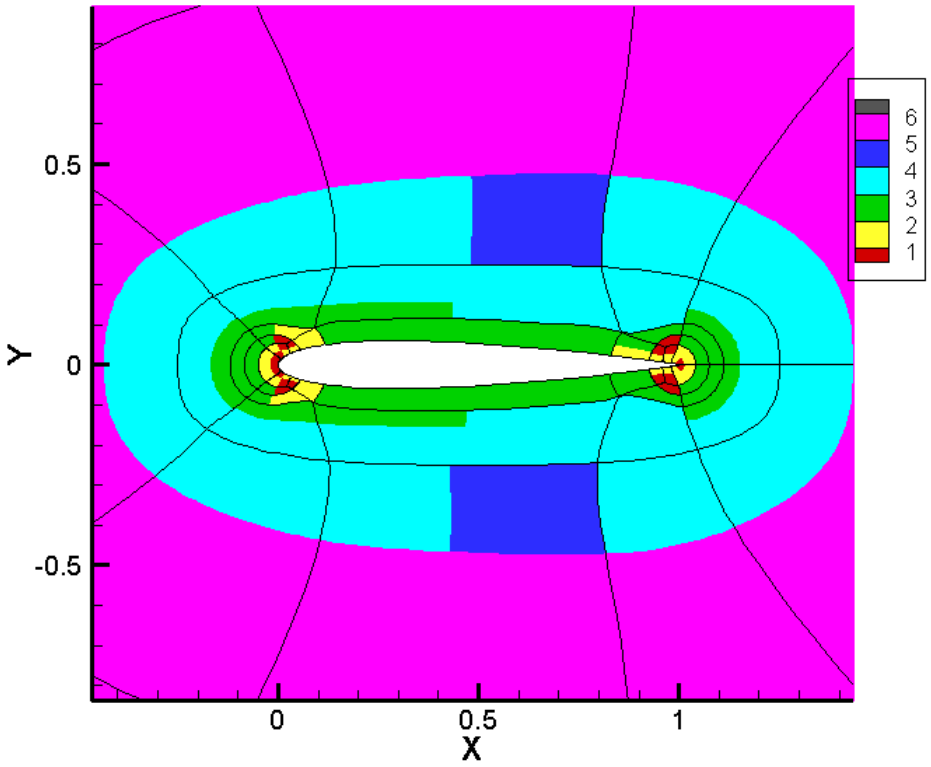


Figure 5-7: Distribution of levels in Grid 1, after the block cutting algorithm cuts the blocks during the run

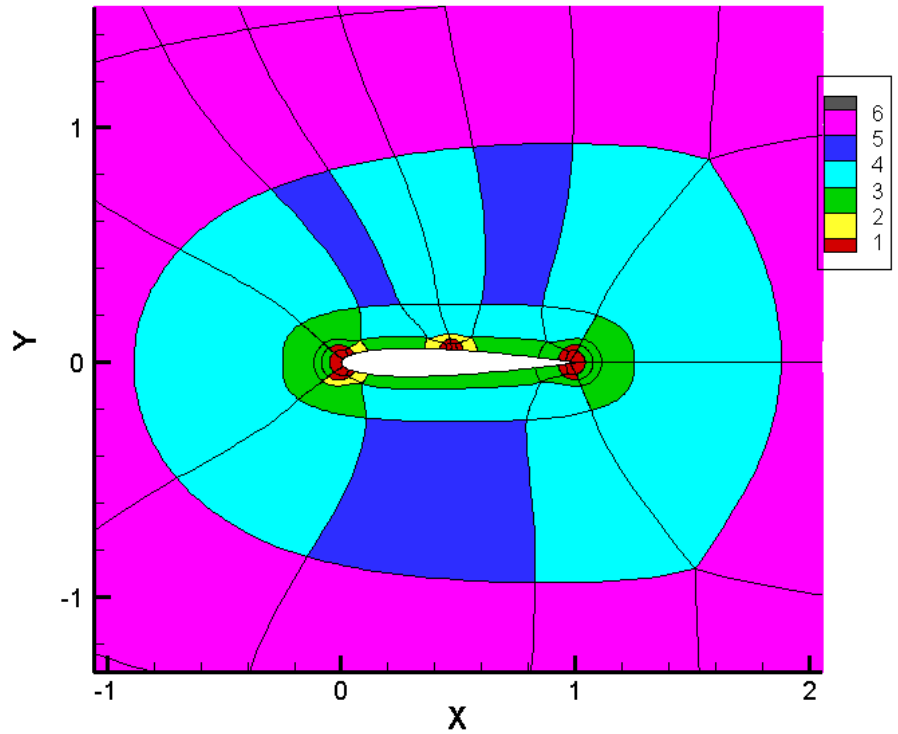


Figure 5-8: Distribution of the MTSAB levels near the airfoil (in Grid 2) without block cutting

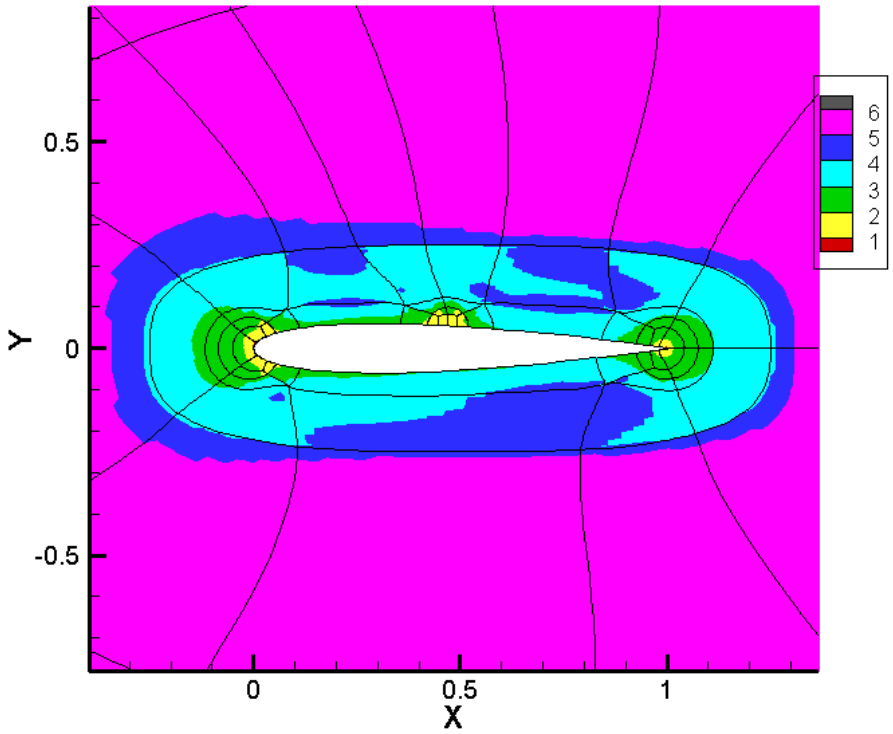


Figure 5-9: Point by point distribution of the MTSAB levels in Grid 2

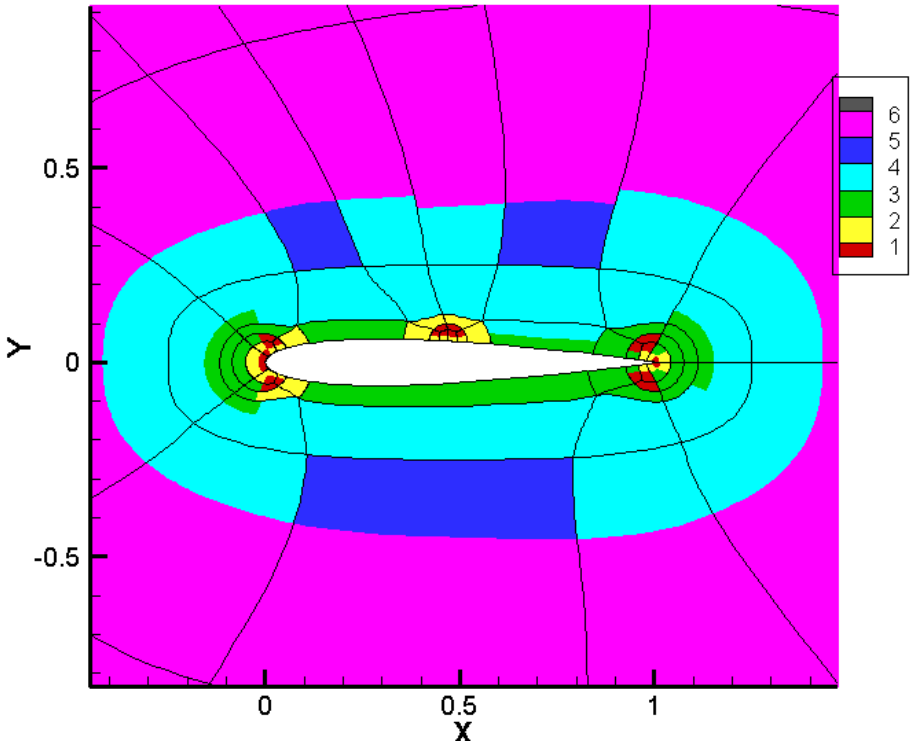


Figure 5-10: Distribution of levels in Grid 2 after the block cutting algorithm cuts the blocks during the run

5.6 Steady Flow Results

The steady flow was calculated for the rounded trailing edge NACA0012 airfoil at Mach 0.63 and 0.75, for an angle of attack of 2 degrees. For the cases, the non-dimensional free stream quantities at the inflow boundary are:

$$\bar{\rho} = 1.0$$

$$\bar{p} = 0.7142857$$

For Mach 0.63:

$$\bar{u} = 0.63 \; at \; \alpha = 2^{\circ} \tag{5.1}$$

For Mach 0.75:

$$\bar{u} = 0.75 \; at \; \alpha = 2^{\circ}$$

Figures 5-11, 5-12 and 5-13 show the pressure contours around the airfoil. It can be seen from these pictures that using the MTSAB scheme, is able to capture the flow features in both subsonic and the transonic cases.

In figure 5-12, the effect of clustering of grid points in Grid 2 at the approximate shock location can be seen for the Mach 0.75 case. The shock in Figure 5-12 is less smeared as compared to the shock in Figure 5-11.

The distribution of coefficient of pressure on the airfoil surface for the cases run is shown in Figures 5-14 and 5-15. Figure 5-14 compares the compares the coefficient of pressure from the current simulation to that given by Steger in [44], for the Mach 0.63 case. A good comparison in the lift can be seen for the computed results.

Figure 5-15 shows the distribution of the coefficient of pressure on the airfoil surface for Mach 0.75 case. The results from using Grids 1 and 2 are compared to that given by Steger [44] for the same flow conditions. A good match in the shock location and lift can be seen between the computed results. The effect of clustering of grid points in Grid 2 can also be seen in this figure. The shock is better resolved when the number of points across the shock is increases. Figure 5-16 shows the actual location of points on the airfoil surface, at the shock location. Grid 1 has around 8 points across the shock and Grid 2 has around 40 points across the shock.

These results are a validation for the MTSAB scheme to solve highly nonlinear steady flows.

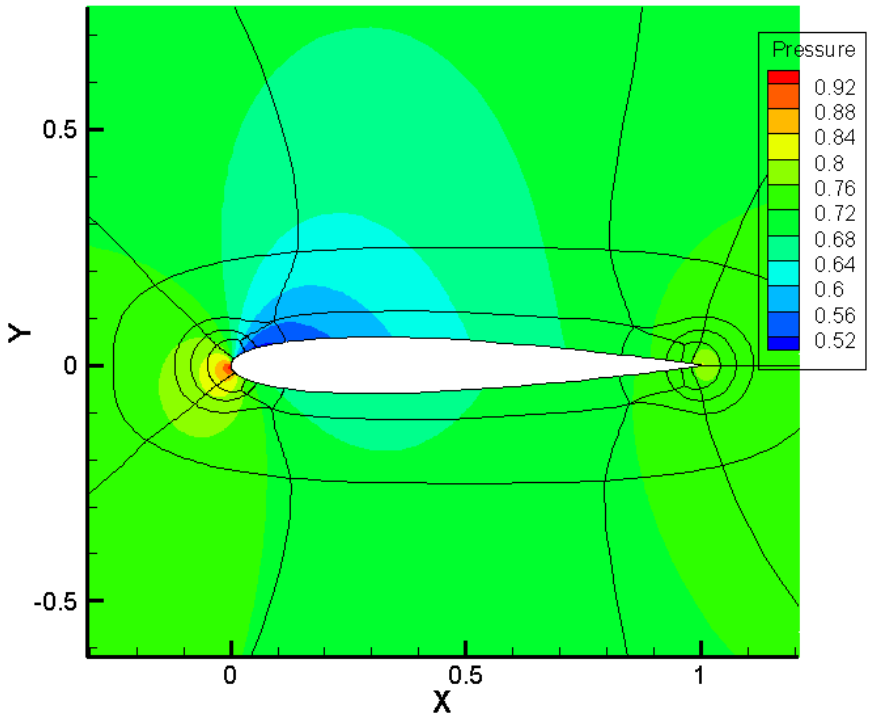


Figure 5-11: Pressure Distribution for Mach 0.63 and 2 deg. AOA case

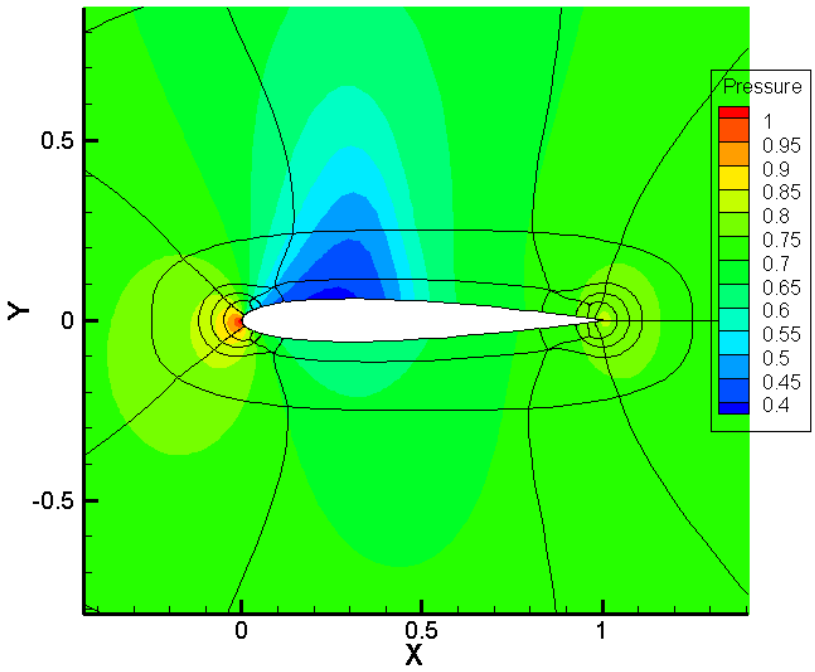


Figure 5-12: Pressure Distribution for Mach 0.75 and 2 deg. AOA case without grid clustering at shock

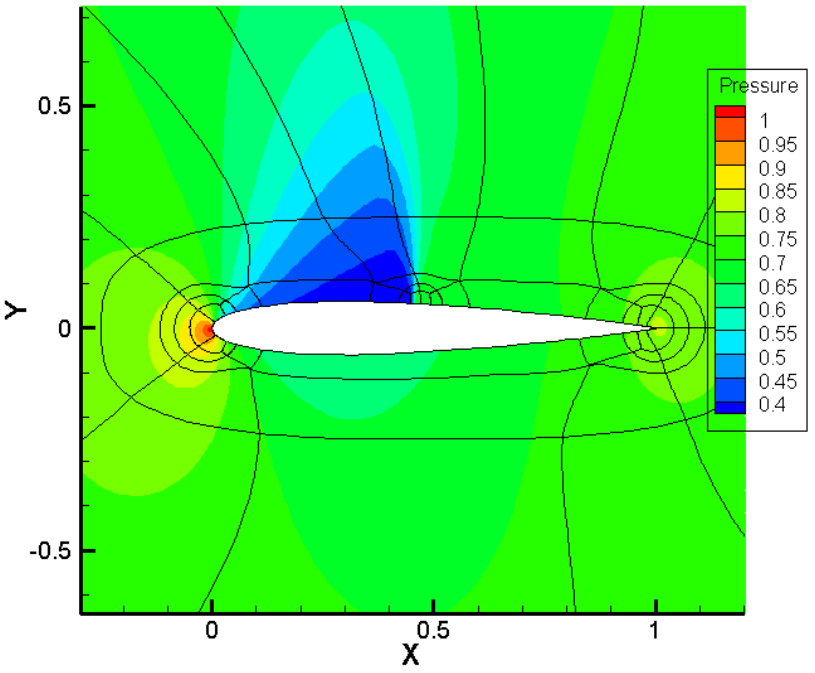


Figure 5-13: Pressure Distribution for Mach 0.75 and 2 deg. AOA case with grid clustering at shock

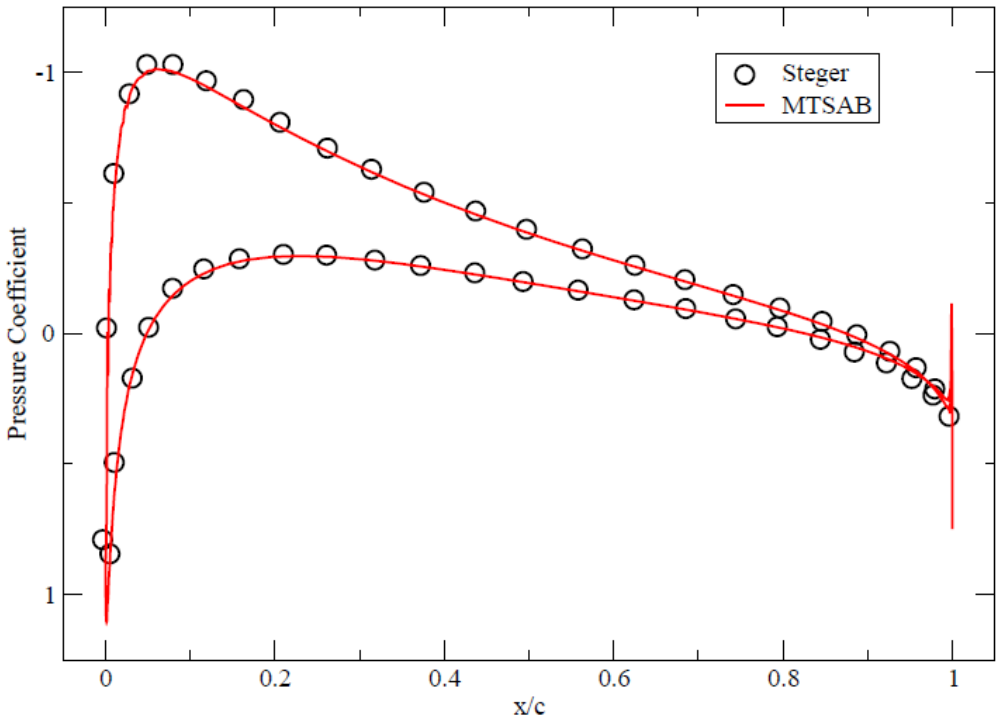


Figure 5-14: Distribution of coefficient of pressure on the NACA0012 airfoil at Mach 0.63 and 2 AOA

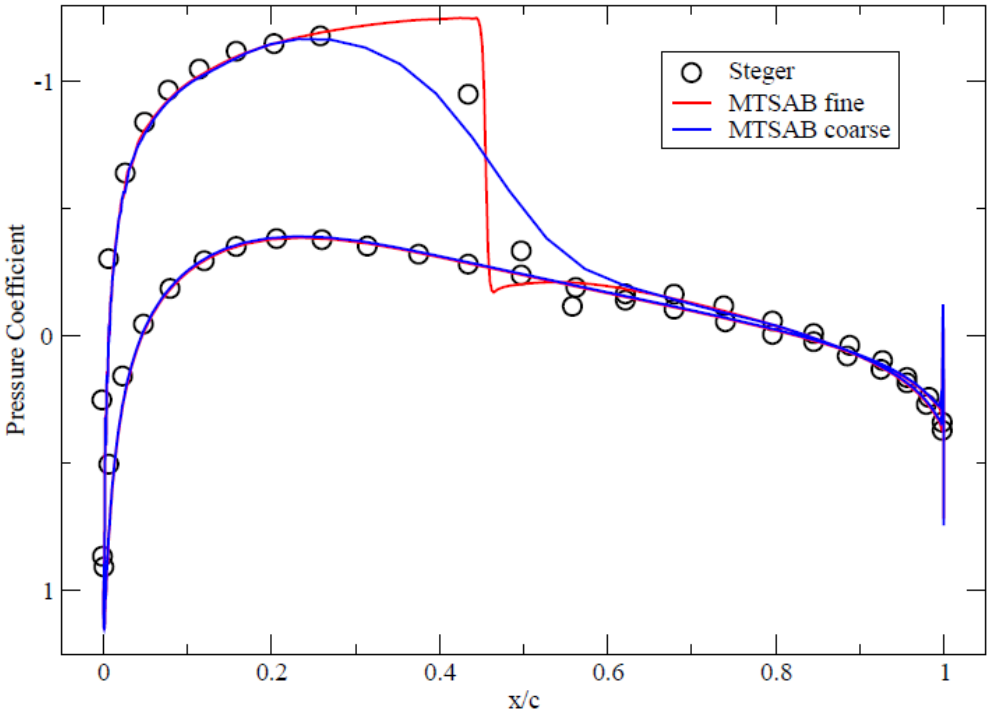


Figure 5-15: Distribution of coefficient of pressure on the NACA0012 airfoil at Mach 0.75 and 2 AOA

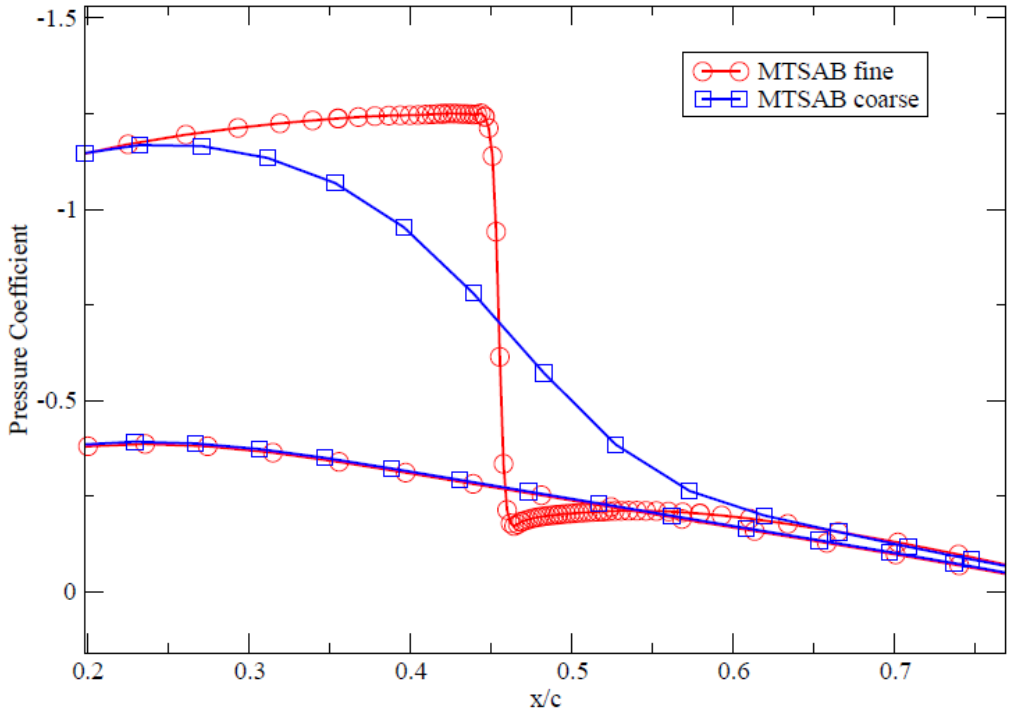


Figure 5-16 Plot showing the number of points resolving the NACA0012 airfoil at Mach 0.75 and 2 AOA

5.7 Unsteady Flow over NACA64A010 Airfoil

MTSAB scheme in the BASS code was extended to work with grid motion. To test the scheme, it was used solve for, flow over a plunging NACA64A010 airfoil. The airfoil is plunging at the rate given by,

$$y_\tau = -a_\infty M_\infty sin\left(\frac{\pi}{180}\right) sin(\emptyset) \tag{5.2}$$

For this case, the freestream Mach number was $M_\infty = 0.8$ and the speed of sound $a_\infty = 1$. The airfoil is plunging between ±1 degrees AOA. In Eq. (5.2), the plunge angle at any non dimensional time is given as,

$$\emptyset = \frac{2\pi}{\bar{t_p}} t_{non\ dimensional} \tag{5.3}$$

Where, $\overline{t_p}$ is the non dimensional time period of the plunge and is defined as,

$$\overline{t_p} = \frac{t_p a_\infty}{c} \tag{5.4}$$

The chord length c in Eq. (5.4) is equal to 1. Therefore,

$$\overline{t_p} = t_p \tag{5.5}$$

To calculate $\overline{t_p}$ a reduced frequency of the plunging motion is defined as,

$$k = \frac{2\pi}{\overline{t_p} M_\infty} \tag{5.6}$$

The reduced frequency for this case was 0.4.

Using Eq. (5.3) and Eq.(5.7), the plunge angle $\emptyset$, is calculated as,

$$\emptyset = kM_\infty t_{non\ dimensional} \tag{5.7}$$

The plunging motion for this case was hardwired into the BASS code.

5.8 Grid Generation and Numerical Details

The grid (Grid 3) was again generated using the GridPro software. Grid 3 has a total of 51465 points. The number of grid blocks in this grid was 95. Figure 5-17 shows the complete grid and Figure 5-18 shows the grid near the airfoil surface. The grid was again clustered at the trailing and the leading edges to resolve the stagnation points. Since for this case, the shocks are moving on the surface of the airfoil, the shock location is not fixed. Therefore, no grid clustering was used to resolve the shocks. Instead, more grid points were used on the airfoil surface for Grid 3, as compared to Grid 1. The same numerical schemes, used for the steady cases were used for this case. BASS code uses a

high accuracy moving wall boundary condition developed by Hixon [45]. Thompson boundary condition was used at the inflow and the outflow boundaries [46].

Grid 3	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Grid Points in Level 4	Grid Points in Level 5	Grid Points in Level 6	Theoretical Speed up	Actual Speed up	Number of blocks
Ideal Distribution	472	2541	6754	9863	13686	18149	8.45	-	-
Distribution Without Block Cutting	6279	451	9513	6069	16802	12351	4.64	3.99	95
Distribution With Block Cutting at Level 1	2849	3001	10393	6069	16802	12351	5.62	3.71	122
Distribution With Block Cutting at Levels 1 and 2	2849	2601	10793	6069	16802	12351	5.69	3.72	126
Distribution With Block Cutting at Level 1,2 and 3	2849	2601	7466	9396	16802	12351	5.96	3.68	141
Distribution With Block Cutting at Level 1,2,3 and 4	2849	2601	7466	8566	17632	12351	6.00	3.66	144
Distribution With Block Cutting at Levels 1,2,3,4 and 5	2849	2601	7466	8566	13897	16068	6.084	3.66	163

Table 5.3 MTSAB Speed up Data for Grid 3

5.9 MTSAB Performance for Unsteady Flow cases

The CFL of the MTSAB scheme was set to 0.31. Figures 5-4 shows the MTSAB levels in Grid 3 when the blocks are not cut. It can be seen from this figure that most of the region away from the airfoil is at level 5 and 6. Figure 5-5 shows the MTSAB levels near the

airfoil surface for Grid 1, when the blocks are not cut. It can be seen from this figure that blocks on the trailing and the leading edges of the airfoil, where the grid is clustered to resolve the stagnation points are at level 1 and the levels of the blocks increase as the grid is stretched away from the stagnation points. The level distributions shown in the figures are at the start of the plunging motion and can vary during the plunge cycle.

As the grid deforms from the plunging motion the, the point to point level distribution in the domain, can change. If the stable time step (or level) in a grid block drops below it existing value during the run, and the grid block continues to march with its originally assigned time step or level, the solution can quickly become unstable. The MTSAB scheme, on detecting this level change performs a Dynamic Level Redistribution. During this redistribution, the blocks are reassigned new level values to stay within the stability limit (as explained in Chapter 2).

Figure 5-23 shows the coefficient of lift during plunging cycle. The post-processor was programmed to write the coefficient of lift to an output file, at regular intervals. It was also programmed to write the lift coefficient whenever a Dynamic Level Redistribution was performed by the MTSAB scheme. In this figure, there are two locations where the data points are located very close to each other, than the rest of the data points. These are the locations in the cycle where the MTSAB scheme detects a level change and performs a Dynamic Level Redistribution.

Table 5.3 shows the speed up data of the MTSAB scheme at the start of the plunging motion. The ideal speed up, which is based on point to point variation of the levels in the grid, is 8.45.

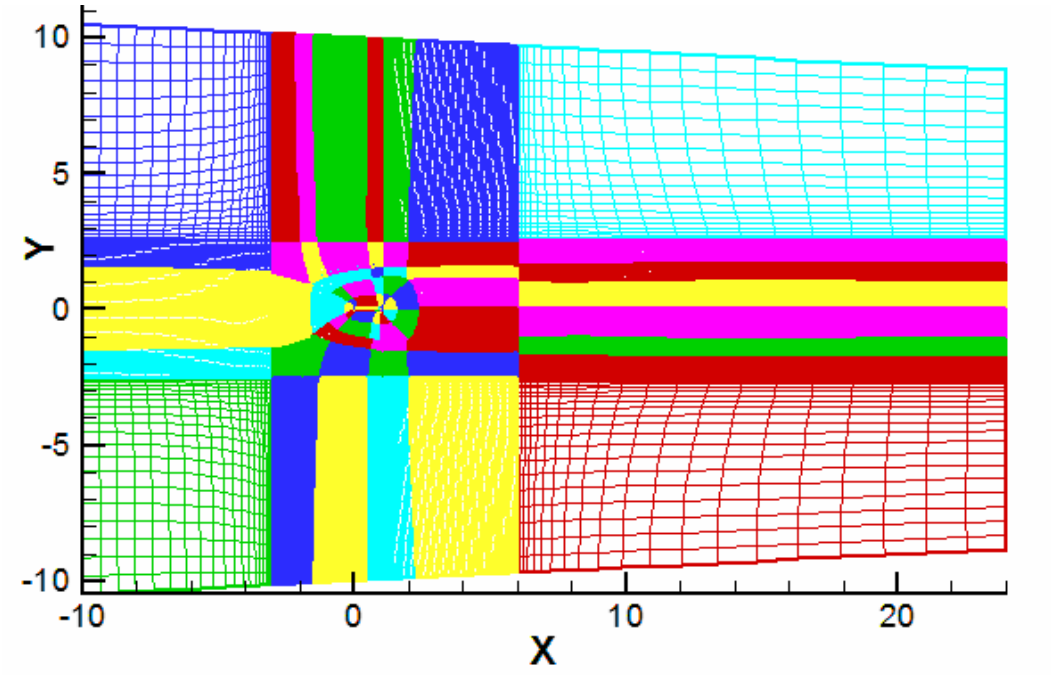


Figure 5-17 Complete grid for NACA64A010 airfoil

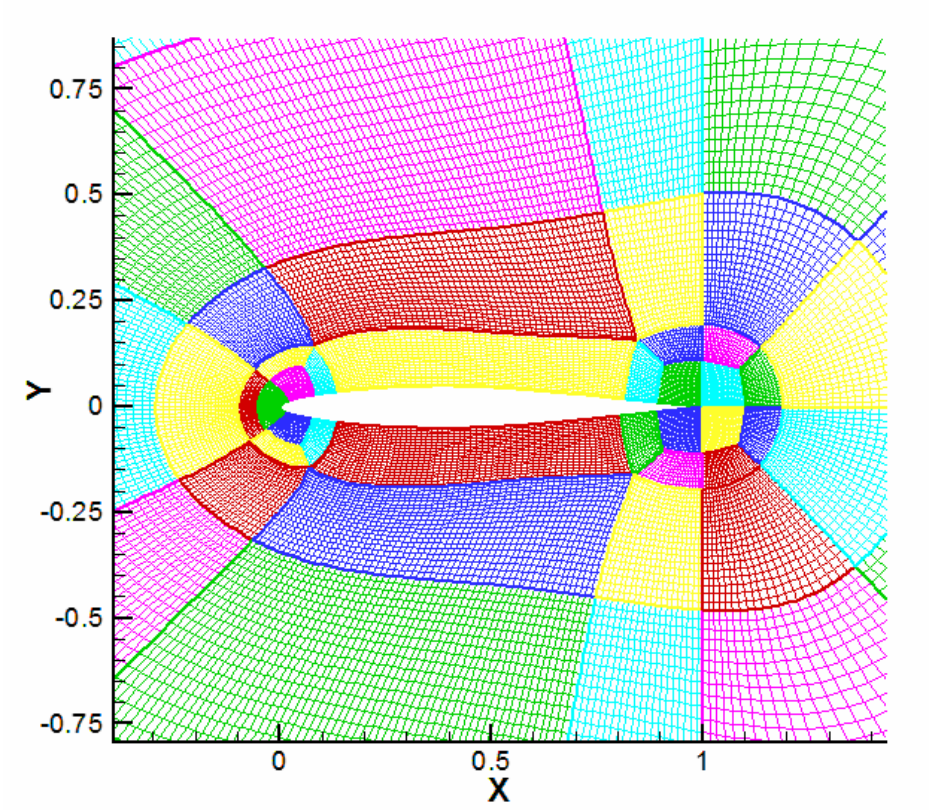


Figure 5-18: Grid near the NACA64A010 airfoil surface (Grid 3)

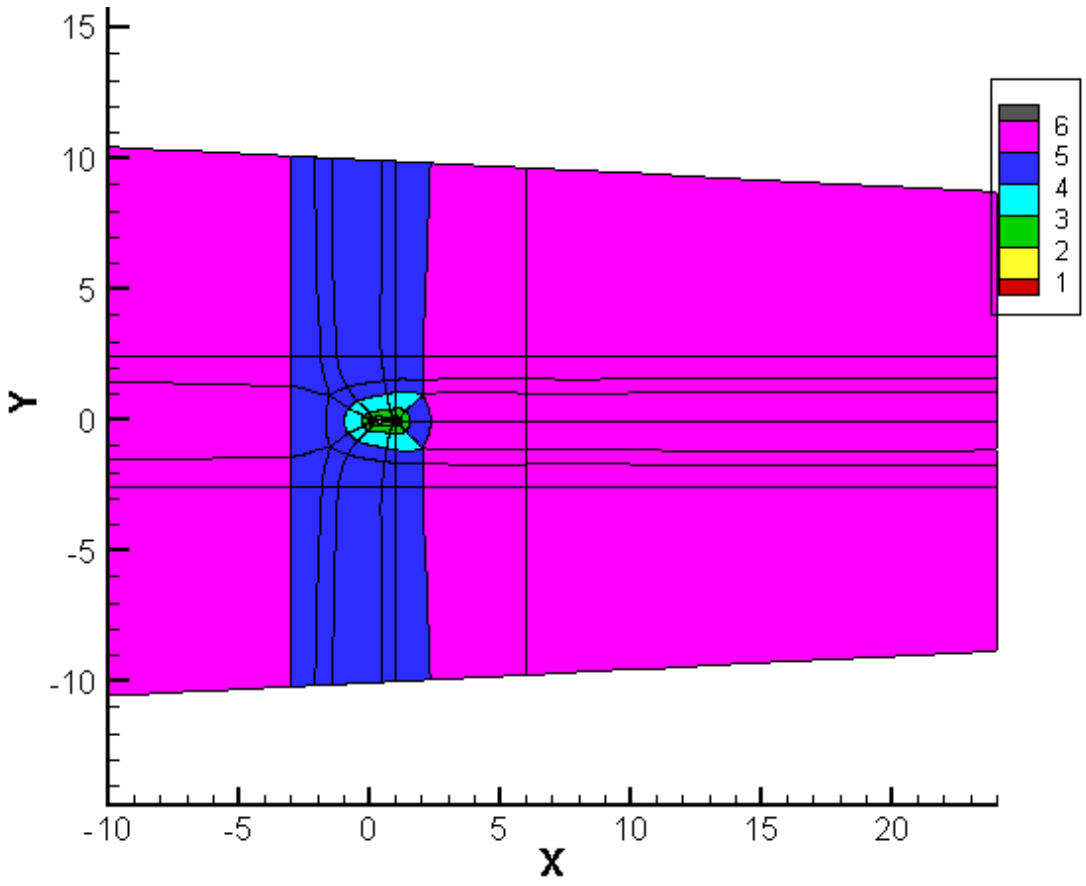


Figure 5-19 Distribution of the MTSAB levels in without block cutting

The theoretical speed up without cutting any block is 4.64 and the actual speed up without using the block cutting algorithm is 3.99. The difference between them is due the overhead from buffer block calculation. Table 5.3 also shows the theoretical speed ups and the actual speed ups for different maximum levels at which the blocks are cut. Again, as it was seen for the steady cases, cutting the blocks just at level 1 reduced actual speed up of the run to 3.71, although the theoretical speed up increases from 4.64 to 5.62. 27 new blocks were generated from cutting blocks at level 1. This adds to the overhead from buffer block calculations. The theoretical speed-up increases to a value of 6.084, when

blocks are cut at 5 levels. It can be seen that block cutting beyond 3 levels does not add much to the theoretical speed-up.

Although 22 blocks were added after block cutting at levels 4 and 5, the buffer block overhead at these levels hardly affects the actual speed up.

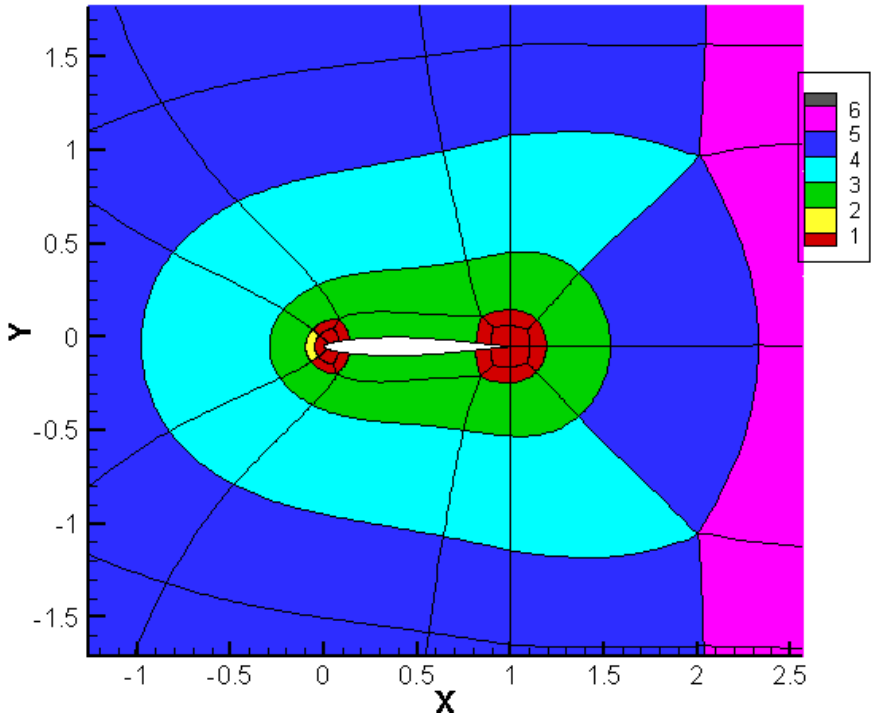


Figure 5-20: Distribution of the MTSAB levels near the NACA64A101 airfoil without block cutting

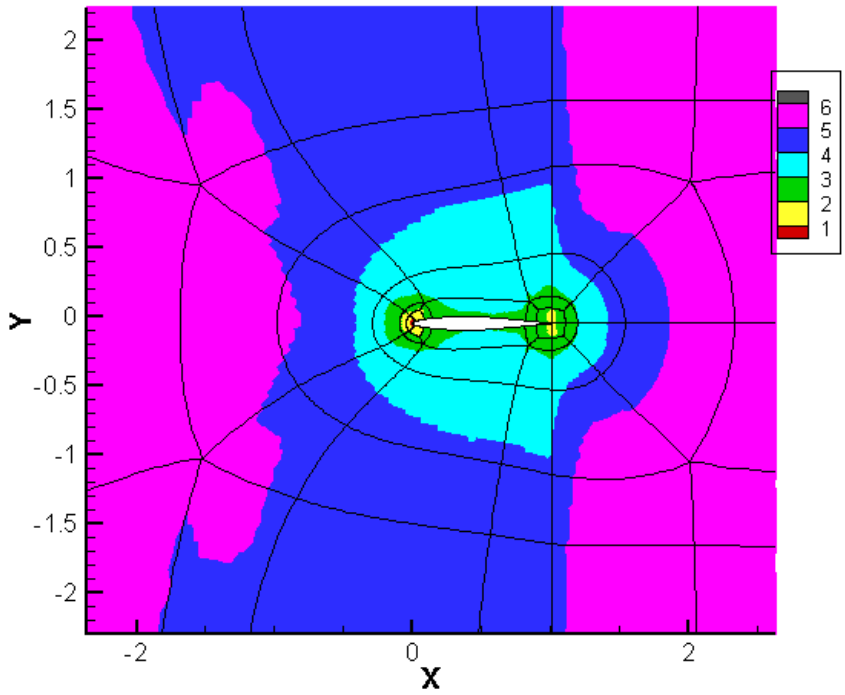


Figure 5-21: Point by point distribution of the MTSAB levels in Grid 1 near the airfoil

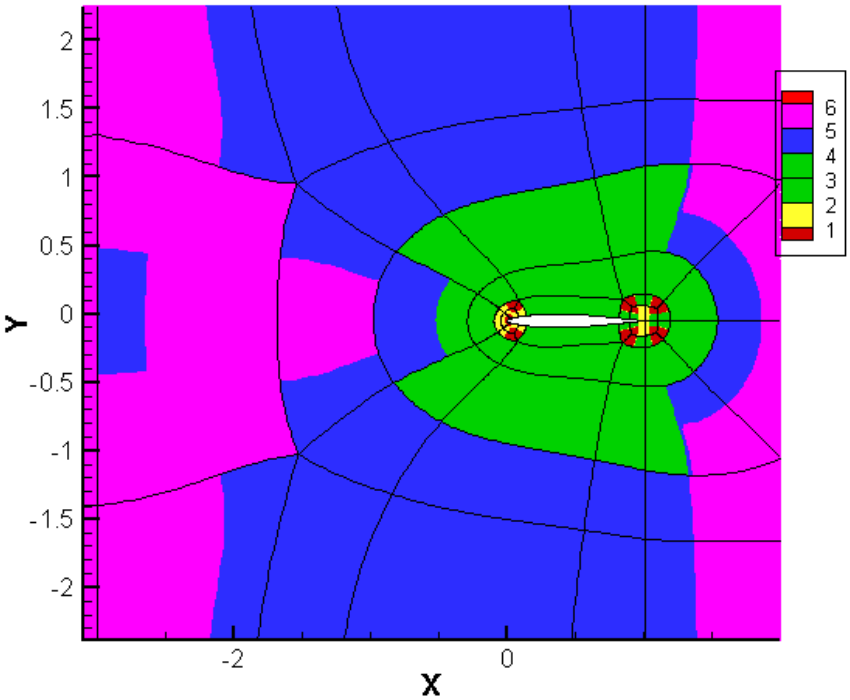


Figure 5-22: Distribution of MTSAB levels after the block cutting algorithm cuts the blocks during the run

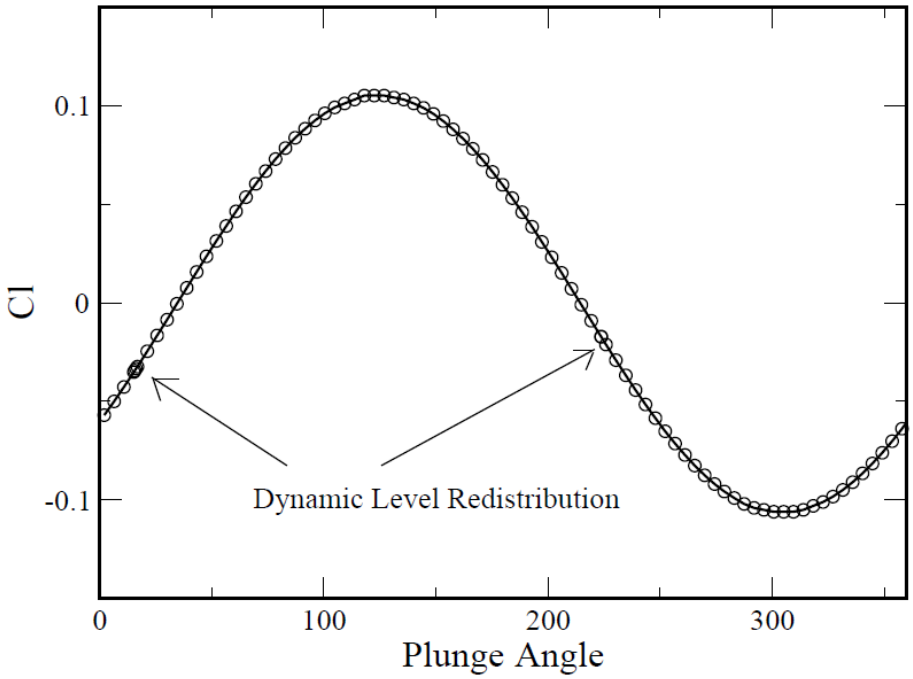


Figure 5-23: The Coefficient of lift vs. Plunge angle plot showing the locations of Dynamic Level Redistribution

5.10 Unsteady Flow Results

Figures 5-24, 5-25 and 5-26 show the instantaneous pressure contours near the airfoil surface, for the maximum, almost mean and minimum values of the lift coefficient during the plunge cycle. The change in the shock locations during the plunge cycle can be seen in these figures.

In Figures 5-27 to 5-28, the coefficients of moment and lift are plotted against the plunge angle for the fourth cycle of the plunging motion. In these figures, the data from the current simulation is compared to the data given by Steger in [44].

These figures also show the phase of the airfoil motion during the plunge cycle. The amplitude of the coefficient of lift computed from the present simulation differs from Steger's data, between 2-6%.

The coefficient of moment, calculated at the quarter chord, differs slightly in amplitude and phase for all the three sets of data. This difference can be attributed to the fact that, the moment sensitive to the difference in shock resolution.

The NACA64A010 airfoil is a symmetric airfoil and there is no mechanism for there to be a difference in the minimum and the maximum values of the coefficients of lift during the plunging motion. In Table 5.4, the minimum, maximum and average values of the lift and the moment coefficients from the present calculation are compared to the minimum, maximum and average values in Steger's and Magnus's data. It can be seen from the table that the data from the present calculation almost has the same magnitude for positive and negative amplitudes of the coefficient of lift.

The moment can be slightly asymmetric, depending on the location of the grid points on the top and the bottom surface of the airfoil.

	Min C_L	**Max C_L**	**Avg. C_L**
Current	-0.106	0.105	-0.0005
Steger	-0.113	0.102	-0.0055
Magnus	-0.114	0.104	-0.005
	Min C_m	**Max C_m**	**Avg. C_m**
Current	-0.0106	0.011	0.0002
Steger	-0.0124	0.014	0.0008
Magnus	-0.0117	0.0105	-0.0006

Table 5.4: Comparison of Min, Max and Avg. values of C_L and C_m for all three sets of data

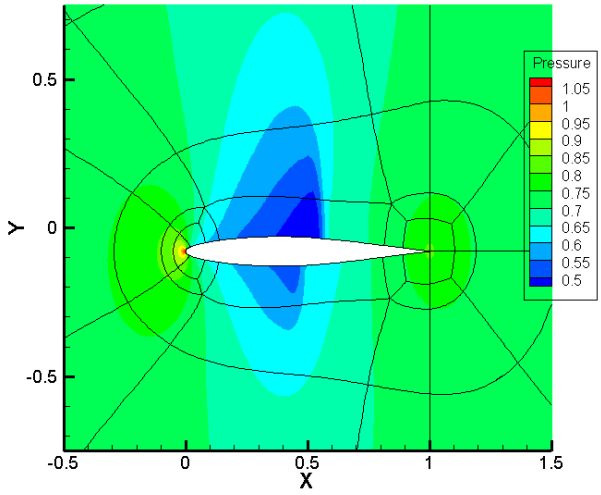


Figure 5-24: Instantaneous Pressure distribution on NACA64A010 (Coeff. of lift=0.105)

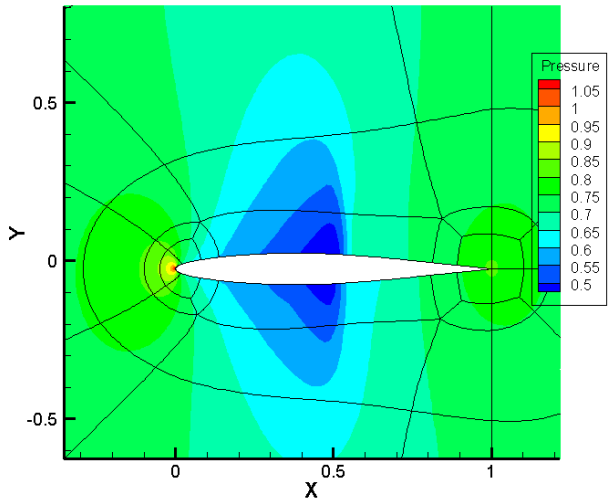


Figure 5-25: Instantaneous Pressure distribution on the NACA64A010 (Coeff. of Lift=-0.004)

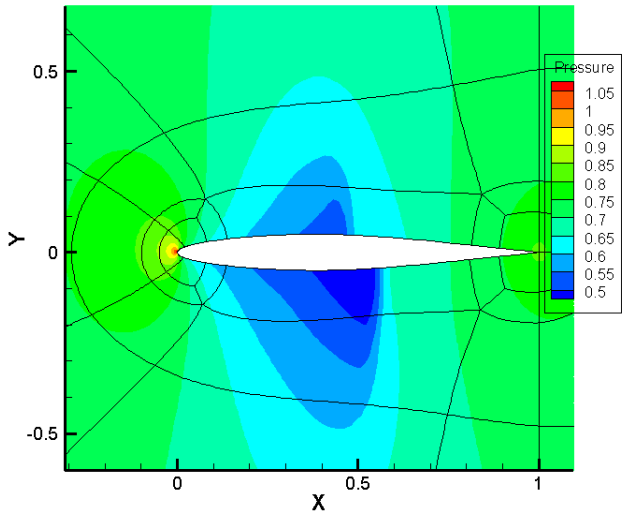


Figure 5-26: Instantaneous Pressure distribution on the NACA64A010 (Coeff. of Lift=-0.106)

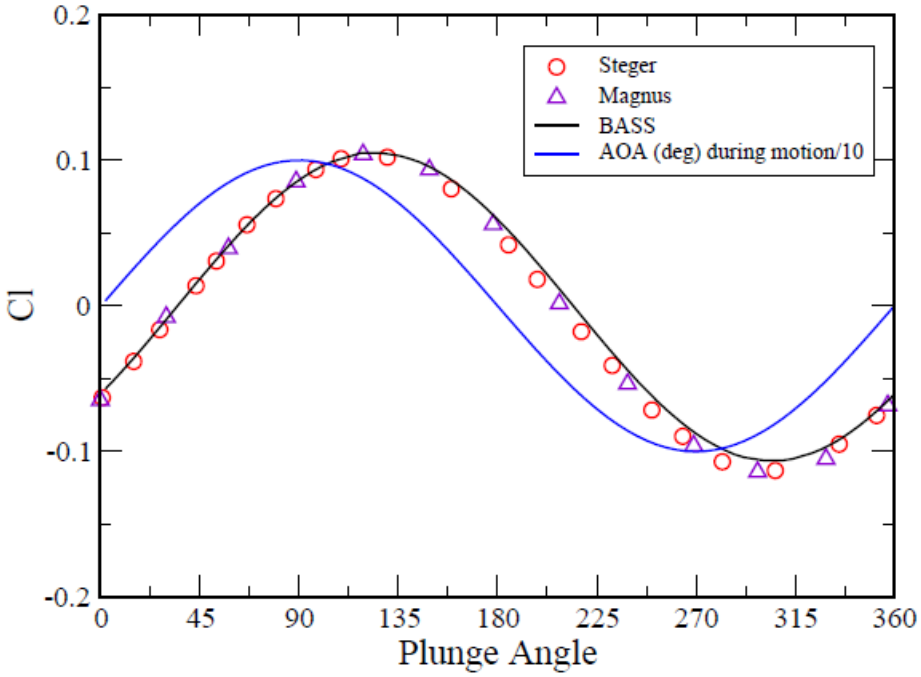


Figure 5-27: C_L variation during the Plunging motion

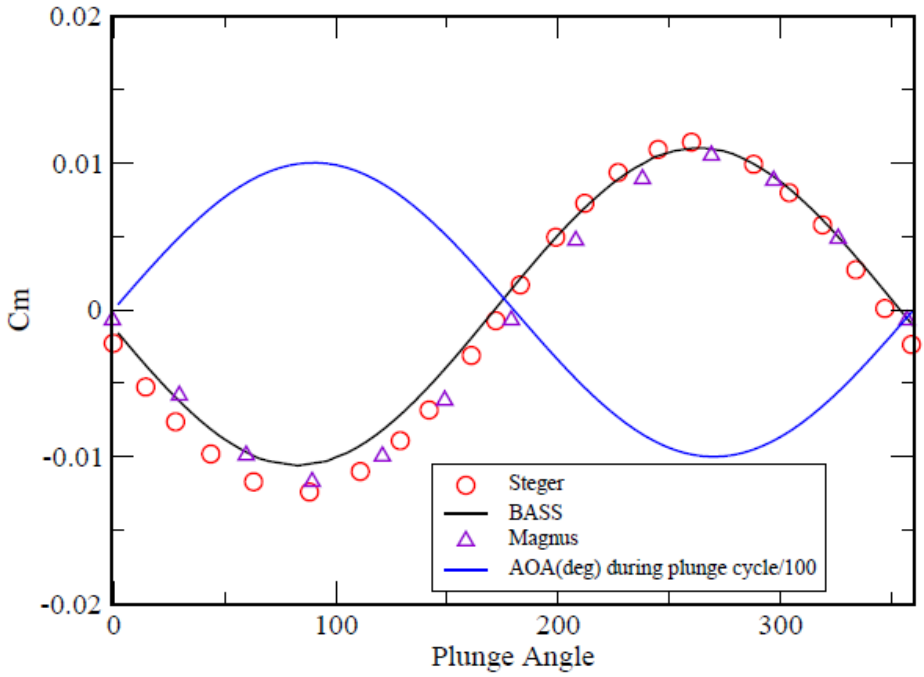


Figure 5-28: C_M variation during the plunging motion

Chapter Six

Multi-Time-Stepping Adams-Bashforth scheme for Viscous Flow Calculations

6.1 Introduction

As a solid body moves through the fluid or as the fluid moves past a solid body, the effect of fluid viscosity causes a thin boundary layer to be formed near the surface of the solid body. The thickness of the boundary layer depends on the Reynolds number of the flow. Boundary layer is a region with large velocity gradients. Inside the boundary layer the velocity changes from being zero on the surface of the solid body (no-slip condition) to free stream velocity. According to Newton's shear stress law (Eq. (6.1)), which states that the shear stress is proportional to velocity gradient, the local shear stress can be very large inside a boundary layer. As a result, there is a skin friction drag exerted on the surface.

$$\tau_0 = \mu \left.\frac{\partial u}{\partial y}\right|_{y=0} \tag{6.1}$$

In computational fluid dynamics, to resolve the huge velocity gradient, the grid spacing in the boundary layer is very small. The grid is clustered near the solid surface

and is stretched outside the boundary layer. For a solver using an explicit time marching scheme, the time step (calculated from the CFL condition) is driven by the very small grid spacing in the boundary layer. Also, the viscous stability limit is much stricter than the inviscid stability limit. This CFL condition for viscous flow calculations is shown in the next section. This results in very long run times. In this scenario, a MTSAB scheme can be a very useful tool. In the BASS code, the MTSAB scheme was extended to perform viscous flow calculation. In this chapter, viscous flow simulations performed using the MTSAB scheme in BASS code are presented.

6.2 Stability Restrictions for Viscous Flow Calculations

The viscous terms in the governing equations add an additional restriction on the stable time step for explicit time marching schemes. This restriction is related to the time scales associated with viscosity.

The exact value of the stable time step is a function of both the spatial differencing method and the time marching scheme used.

Stability analysis for a model 1-D convective-diffusive equation is shown here. Given the Linearized viscous Burger's equation:

$$u_t + cu_x - \vartheta u_{xx} = 0 \tag{6.2}$$

where, ϑ is the kinematic viscosity a numerical method is used to solve the equation, which is marching in the time direction:

$$u_i^{n+1} = u_i^n + F(\Delta t u_t)_i^n \tag{6.3}$$

In Eq. (6.3), $F(\Delta t u_t)_i^n$ represents the time marching scheme and Δt is the time step used.

The governing equations relate the time derivatives of u with the spatial derivatives. Thus using the spatial differencing scheme for the first derivative *D*, the time derivatives can be obtained as:

$$(\Delta t u_t)_i^n = -c\Delta t D(u)_i^n + \vartheta \Delta t D(D(u)_i^n)_i^n \tag{6.4}$$

The combined equation is written as:

$$u_i^{n+1} = u_i^n + F(-c\Delta t D(u)_i^n + \vartheta \Delta t D(D(u)_i^n)_i^n)_i^n \tag{6.5}$$

The discrete solution S is defined as the exact solution to this equation:

$$S_i^{n+1} - (\, S_i^n + F(-c\Delta t D(S)_i^n + \vartheta \Delta t D(D(S)_i^n)_i^n)_i^n \,) = 0 \tag{6.6}$$

However, due to the limited precision of the digital computers, the solution that is actually obtained is defines as the computed solution C:

$$C = S + \varepsilon \tag{6.7}$$

where, ε is the round off error due to the limited presicion of the digital computers. Thus the actual solution that is obtained is given as:

$$C_i^{n+1} = C_i^n + F(-c\Delta t D(C)_i^n + \vartheta \Delta t D(D(C)_i^n)_i^n)_i^n = 0 \tag{6.8}$$

Substituting the definition of the computed solution into the previous equation, we obtain:

$$(S+\varepsilon)_i^{n+1} = (S+\varepsilon)_i^n + F(-c\Delta t D(S+\varepsilon)_i^n + \vartheta \Delta t D(D(S+\varepsilon)_i^n)_i^n)_i^n \tag{6.9}$$

$$\varepsilon_i^{n+1} = \varepsilon_i^n + F(-c\Delta t D(\varepsilon)_i^n + \vartheta \Delta t D(D(\varepsilon)_i^n)_i^n)_i^n \tag{6.10}$$

The error is transformed into Fourier components in space:

$$\varepsilon(x,t) = A(t)\sum_{j=1}^{N} e^{ik_j x} \tag{6.11}$$

Since the error is linear, the equations for each error wavenumber can be examined separately. The resulting equation for a wavenumber k is:

$$A^{n+1}e^{ikx} = A^n\left(e^{ikx} + F\left(-c\Delta t D\left(e^{ikx}\right) + \vartheta \Delta t D\left(D\left(e^{ikx}\right)\right)\right)\right) \tag{6.12}$$

The stability analysis is decomposed into three parts. First, the stability of the time marching scheme is analyzed. Second, the performance of the spatial differencing method is obtained. Third, these results are combined with the governing equation to determine the stability of the actual scheme.

6.2.1 Effect of the Time Marching Scheme

Initially, is assumed that „perfect' spatial differencing scheme is used:

$$D\left(e^{ikx}\right) = ie^{ikx} = i\frac{k\Delta x}{\Delta x}\, e^{ikx} \tag{6.13}$$

$$D\left(D\left(e^{ikx}\right)\right) = -\delta e^{ikx} = -\frac{\delta \Delta x^2}{\Delta x^2}\, e^{ikx} \tag{6.14}$$

In the above equation, $\delta = k^2$ and Δx is the grid spacing. Substituting in, Eq. (6.12) becomes,

$$A^{n+1}e^{ikx} = A^n\left(e^{ikx} + F\left(-i\left(\frac{c}{\Delta x}\right)(k\Delta x)\Delta t\left(e^{ikx}\right) - \left(\frac{\vartheta}{\Delta x^2}\right)(\delta \Delta x^2)\Delta t\left(e^{ikx}\right)\right)\right) \tag{6.15}$$

Defining the parameters:

$$\gamma = \frac{c}{\Delta x} \tag{6.16}$$

$$\beta = \frac{\vartheta}{\Delta x^2} \tag{6.17}$$

gives

$$A^{n+1}e^{ikx} = A^n\left(1 + F(-i\gamma(k\Delta x)\Delta t - \beta(\delta\Delta x^2)\Delta t)\right) \tag{6.18}$$

In order for the time marching scheme to be stable, the magnitude of the error at each wavenumber cannot increase from one time step to the next:

$$\left|\frac{A^{n+1}}{A^n}\right| \leq 1 \tag{6.19}$$

or, for the particular time marching scheme,

$$|1 + F(-i\gamma(k\Delta x)\Delta t - \beta(\delta\Delta x^2)\Delta t)| \leq 1 \tag{6.20}$$

Defining,

$$\omega = (-i\gamma(k\Delta x) - \beta(\delta\Delta x^2))$$

$$\omega^*\Delta t = F(\omega\Delta t) \tag{6.21}$$

gives the stability equation for the time marching scheme:

$$|1 + (\omega^*\Delta t)| \leq 1 \tag{6.22}$$

In the governing equation, the real and the imaginary components of ω depend on the magnitude of c,δ, grid spacing Δx, and the wavenumber k. Physically, the possible dynamics in the equation can range from the convection- dominated (corresponding to the imaginary value of ω), to diffusion dominated (corresponding to the real value of ω)

or anywhere in between. Note that the grid spacing can also change the dynamics in the numerical equation.

Thus in practical analysis, the worst case ratio of the imaginary and the real components are determined along with the lowest magnitude of the $\omega\Delta t$, which will cause instability. This value is defined as $(\omega\Delta t)_{max}$. With the knowledge of $(\omega\Delta t)_{max}$, a maximum frequency ω_{max} must be defined for the problem. Combining the two parameters results in the maximum stable time step for the calculation:

$$(\Delta t)_{max} = \frac{|(\omega\Delta t)_{max}|}{|\omega_{max}|} \tag{6.23}$$

6.2.2 Effect of the Spatial Differencing Scheme

The spatial differencing method that is used and the grid chosen combine to define the value of ω_{max} which can occur in the calculation. To accomplish this, the spatial differencing scheme is applied to the calculation of the derivative of a simple harmonic function on a grid with uniform spacing Δx. This results in the definition:

$$D\left(e^{ikx}\right) = i\frac{(k\Delta x)^*}{\Delta x}\, e^{ikx} \tag{6.24}$$

$$D\left(D\left(e^{ikx}\right)\right) = -\frac{(\delta\Delta x^2)^*}{\Delta x^2}\, e^{ikx} \tag{6.25}$$

Due to the Nyquist limit, there is a maximum wavenumber that can be resolved on the grid. The maximum wavenumbers correspond to the simple harmonic function with two

grid points per wavelength. Thus, the permissible values of the wavenumber must lie in the range:

$$\frac{-\pi}{\Delta x} \leq k \leq \frac{\pi}{\Delta x} \tag{6.26}$$

or

$$-\pi \leq k\Delta x \leq \pi \tag{6.27}$$

In this range of the wavenumbers, the maximum values of $(k\Delta x)^*_{max}$ and $(\delta\Delta x^2)^*_{max}$ can be determined. As an example, for the Tam and Webb DRP scheme, the maximum values are:

$$(k\Delta x)^*_{max} = 1.644 \tag{6.28}$$

$$(\delta\Delta x^2)^*_{max} = 2.7034 \tag{6.29}$$

These maximum values can be substituted in to the analysis for the time marching scheme to obtain:

$$\omega_{max} = (-i\gamma(k\Delta x)^*_{max} - \beta(\delta\Delta x^2)^*_{max}) \tag{6.30}$$

$$|\omega_{max}| = \sqrt{(\gamma^2(k\Delta x)^{*2}_{max} + \beta^2(\delta\Delta x^2)^{*2}_{max})} \tag{6.31}$$

$$(\Delta t)_{max} = \frac{|(\omega\Delta t)_{max}|}{|\omega_{max}|} \tag{6.32}$$

6.3 Laminar Flow over a Flat Plate

A laminar flow over a two dimensional flat plate, with zero pressure gradient was used to verify the accuracy of the MTSAB scheme in the BASS code. In this test, a flat plate with

a 0.12% thickness was used. The leading and the trailing edges of the flat plate were semicircular, to avoid the numerical singularities associated with infinitely thin flat plate. The results from the BASS code simulations are compared to the exact solution of Blasius, as given by Schlicting [47]. The non dimensional length of the plate is 1.0. As the fluid flows along the plate, a boundary layer is formed and the thickness of the boundary layer increases along the length of the plate

6.3.1 Blasius solution

A similarity solution for the boundary layer for a flow over a flat plate was presented by H. Blasius. Let the leading edge of the flat plate be at x=0, the plate being parallel to the x-axis and infinitely long. Let the free stream velocity of the flow parallel to this flat pate flat plate is U_∞.The velocity of the potential flow is constant in this case and therefore,

$$\frac{\partial p}{\partial x} = 0 \tag{6.33}$$

Using the above, the boundary layer equations for the flow over flat plate become,

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial x} = 0 \tag{6.34}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial x} = \vartheta\frac{\partial^2 u}{\partial y^2} \tag{6.35}$$

$$y = 0:\ u = v = 0 \tag{6.36}$$

$$y = \infty:\ u = U_\infty \tag{6.37}$$

Since the flat plate is infinitely long, it is assumed that the velocity profiles at varying distances from the leading edge are similar to each other. The velocity curves $u(y)$ for

varying x distances are made identical by selecting the free stream velocity U_∞ and the boundary layer thickness $\delta(x)$ at any given x location, as scale factors for u and y. The similarity of the velocity profiles in the boundary layer is given as,

$$\frac{u}{U_\infty} = \phi\left(\frac{y}{\delta}\right) \tag{6.38}$$

From the order of magnitude analysis of the Navier-Stokes, the boundary layer thickness at any x location is given as,

$$\delta(x) = \left(\frac{\vartheta x}{U_\infty}\right)^{\frac{1}{2}} \tag{6.39}$$

At this point a non-dimensional co-ordinate η is used as similarity variable. It is given as,

$$\eta = \left(\frac{y}{\delta(x)}\right) = y\left(\frac{U_\infty}{\vartheta x}\right)^{\frac{1}{2}} \tag{6.40}$$

At this point stream function ψ is introduced and the u and the v components of the velocity can be expressed in terms of stream functions as,

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x} \tag{6.41}$$

Using a stream function defined above, Eq. (6.11) is obtained.

$$\psi = \sqrt{\vartheta x U_\infty} f(\eta) \tag{6.42}$$

the u velocity component becomes,

$$u = U_\infty f'(\eta) \tag{6.43}$$

Using the stream function and substituting it into the boundary layer equations an ordinary differential equation is obtained.

$$ff'' + 2f''' = 0 \tag{6.44}$$

The boundary conditions for this differential equation are:

$$\eta = 0:\ f = 0, f' = 0 \tag{6.45}$$

$$\eta = \infty:\ f' = 1$$

Numerical Solution to this nonlinear ODE is given in Table 6.2.Using the Blasius solution, the surface shear stress is given as:

$$\tau_0 = \mu \frac{\partial u}{\partial y}\bigg|_{y=0} = \mu U_\infty \left(\frac{U_\infty}{\vartheta x}\right)^{\frac{1}{2}} f''(0) = 0.332 \mu U_\infty \left(\frac{U_\infty}{\vartheta x}\right)^{\frac{1}{2}} \tag{6.46}$$

Then the Blasius, local skin friction coefficient is:

$$C_{fx} = \frac{0.664}{\sqrt{Re_x}} \tag{6.47}$$

where, the plate Reynolds number Re_x is,

$$Re_x = \frac{U_\infty x}{\vartheta} \tag{6.48}$$

6.3.2 Grid Generation and Numerical Details

The grid for this case was generated using GridPro [31]. The grid is a multi-block structured grid. The grid has 86 blocks and a total of 43808 grid points. Figure 6-1 and Figure 6-2 show the close up and body fitted views of the grid. Thompson boundary condition [46] was used at the inflow and the outflow boundaries and wall boundary condition [27, 48] was used on the plate. The non dimensional mean quantities at the inflow boundary are,

$$\bar{\rho} = 1.0$$

$$\bar{p} = 0.714285 \tag{6.49}$$

$$\bar{u} = 0.2$$

$$\bar{v} = 0.0$$

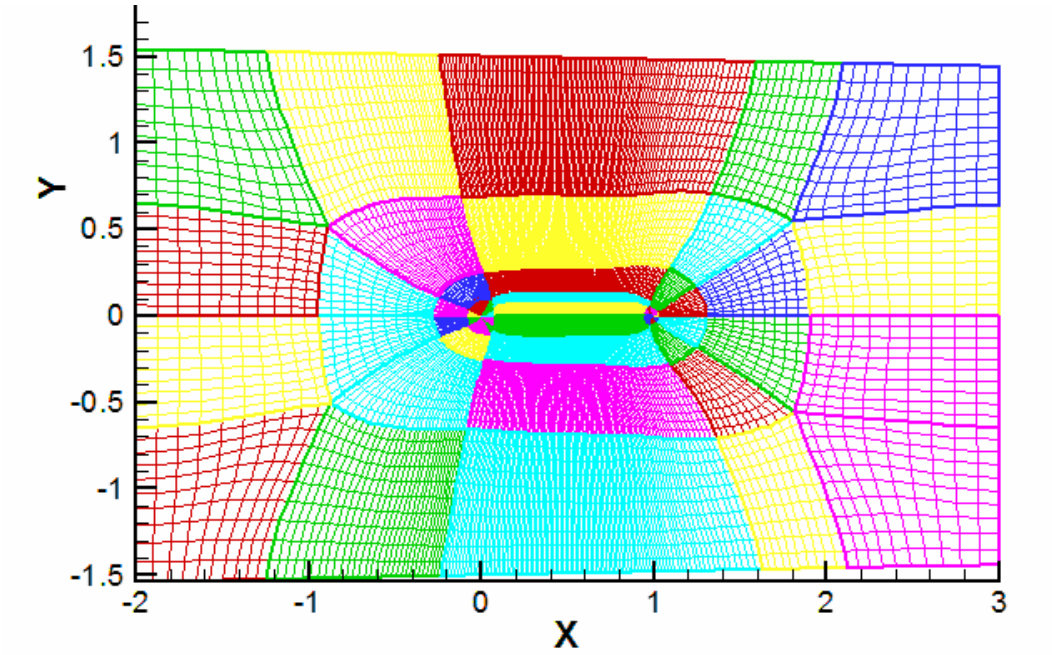


Figure 6-1: Complete grid for the flow over a flat plate

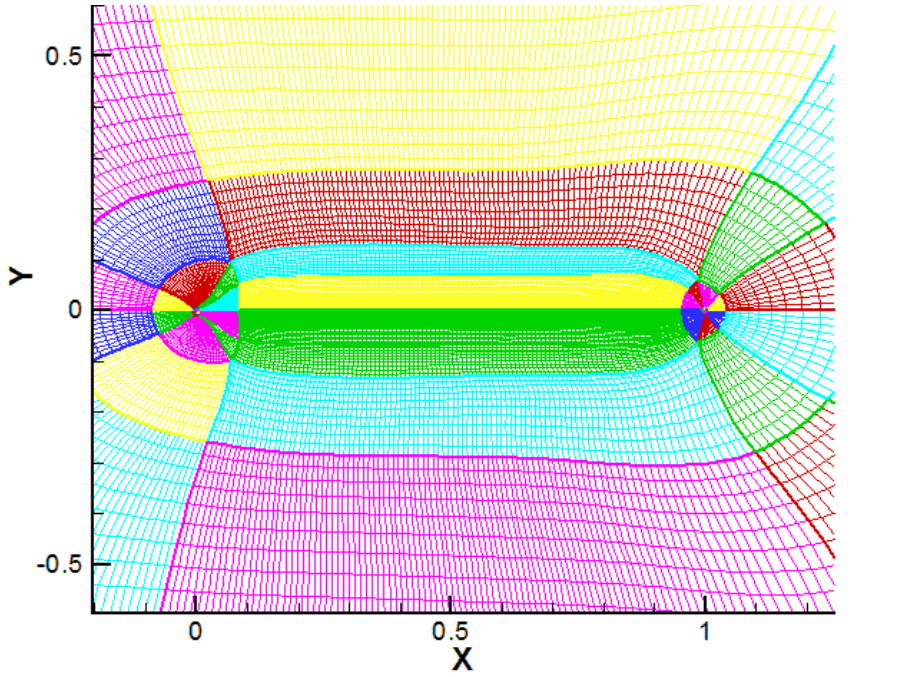


Figure 6-2: Grid near the flat plate

Optimized 4th order DRP scheme of Tam and Webb was used for spatial differentiation. Hixon's [29] optimized finite difference boundary stencils were used at the boundaries for improved stability.10th order background dissipation was used. Two cases with different length scales were run. For Case 1 the length scale was set to 0.01 and for Case 2 the length scale was increased to 0.1. Based on the stability limit, CFL for the MTSAB scheme was set to 0.2.

6.3.3 MTSAB Performance

Figure 6-3 shows the grid near the leading edge of the flat plate. It can be seen from this figure that the blocks that are located on the plate have very small grid spacing near the plate (to resolve the boundary layer on the plate) and stretch rapidly away from the plate.

Figure 6-4 shows the distribution of levels near the leading edge when the block cutting algorithm was not used. The MTSAB speed-up data is given in Table 6-1.The actual speed up without the use of the block cutting algorithm was 2.2 and the theoretical speed up was 2.3. The difference between the actual and theoretical speed-ups is 0.05%.

Figure 6-5 shows the point by point variation of levels near the leading edge. It can be seen from this figure that the blocks located on or near the plate if cut, can result in shifting grid points to higher levels. The ideal speed up, which is based on the point to point distribution, is 9.32.

Figure 6-6 shows the distribution of the levels in the blocks after the blocks are cut during the run. The theoretical speed up after the blocks are cut at all levels is 6.07. Comparing Figures 6-6 and 6-5, it can be seen that the block cutting algorithm cuts the blocks efficiently. As mentioned before, the minimum number of points in every direction is restricted to 10 and this caused the difference between the point to point variations in the levels and the distribution of levels obtained from the block cutting algorithm. The actual run time speed up after cutting the blocks was 3.27.

It can be seen from Table 6.1 that as the blocks are cut at higher levels, the actual speed up does not keep in pace with the theoretical speed up. Table 6.1 also shows the number of blocks generated after cutting the blocks, for different maximum levels for which the blocks are cut. The number of blocks increase from 86, without cutting any block to 156 after the blocks are cut at all the levels. The increased number of blocks caused an increase in the overhead from the buffer block calculation. This overhead is greater for viscous flow calculations than it is for inviscid, as mentioned in chapter 3.

	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Grid Points in Level 4	Grid Points in Level 5	Grid Points in Level 6	Theoretical Speed up	Actual Speed up	Number of blocks
Ideal Distribution	1120	2306	3018	3654	5150	28560	9.32	NA	86
Distribution Without Block Cutting	12800	10100	1606	726	3572	15004	2.3	2.2	86
Distribution With Block Cutting at Level 1	2827	15939	5740	726	3572	15004	3.36	2.54	102
Distribution With Block Cutting at Levels 1 and 2	2827	3938	9863	726	11450	15004	5.13	3.06	114
Distribution With Block Cutting at Level 1,2 and 3	2827	3938	3177	5916	12946	15004	5.75	3.18	136
Distribution With Block Cutting at Level 1,2,3 and 4	2827	3938	3177	4048	14814	15004	5.84	3.2	142
Distribution With Block Cutting at Levels 1,2,3,4 and 5	2827	3938	3177	4048	5984	23834	6.07	3.27	156

Table 6.1 Speed up data from using the MTSAB scheme

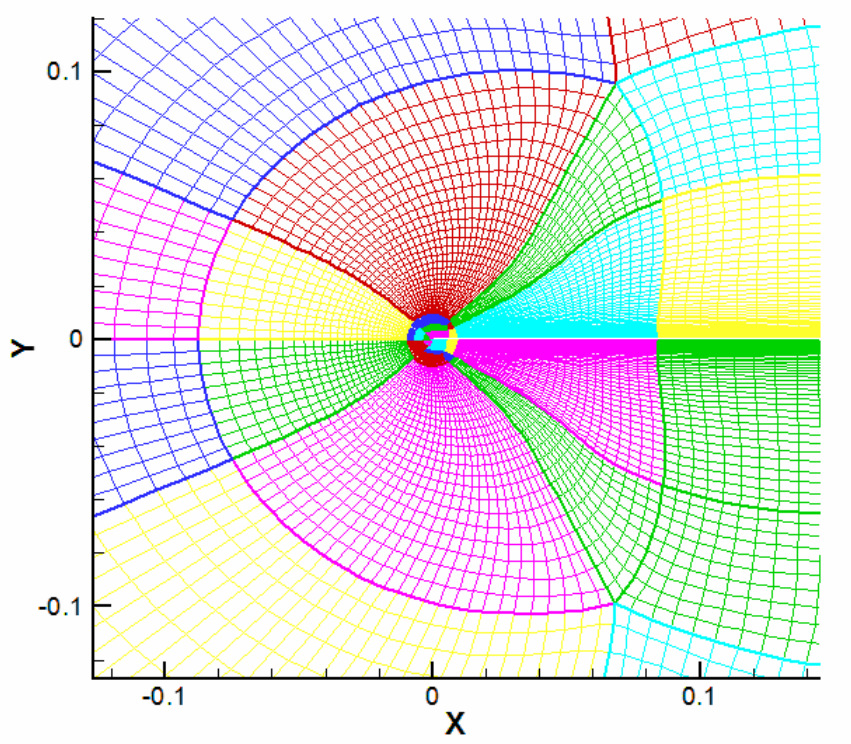


Figure 6-3 Grid near the leading edge of the flat plate

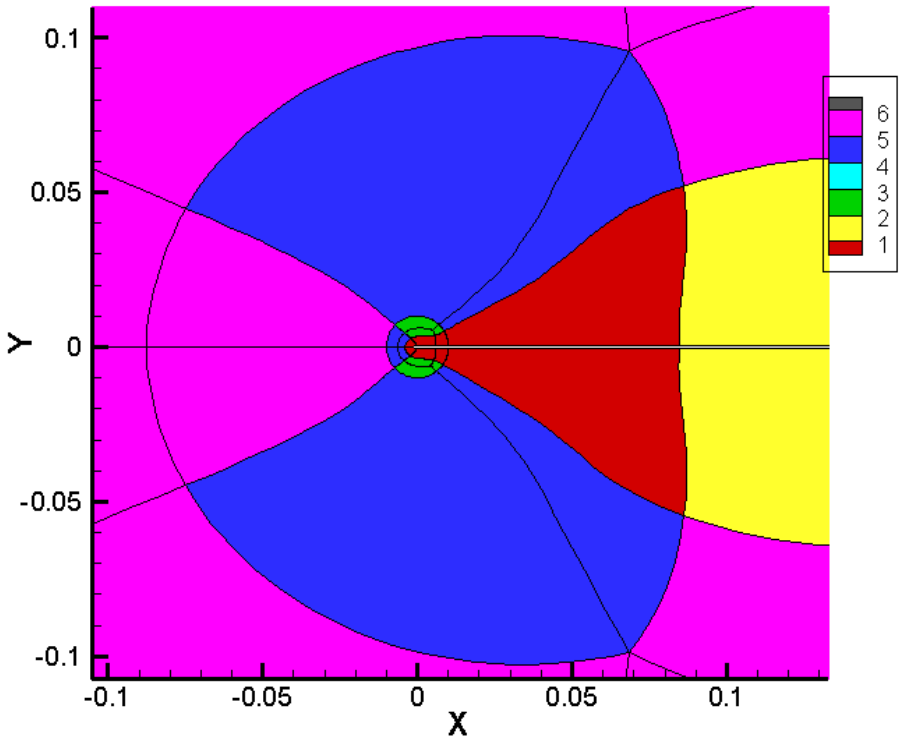


Figure 6-4 Distribution of the MTSAB levels near the leading edge, without block cutting

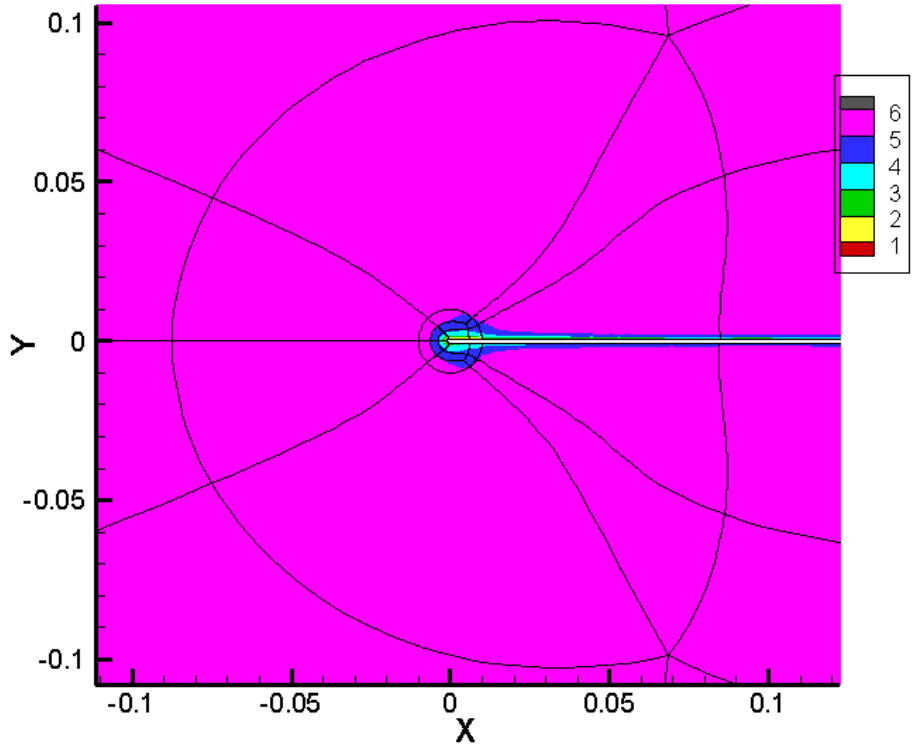


Figure 6-5 Point by point distribution of MTSAB levels near the leading edge

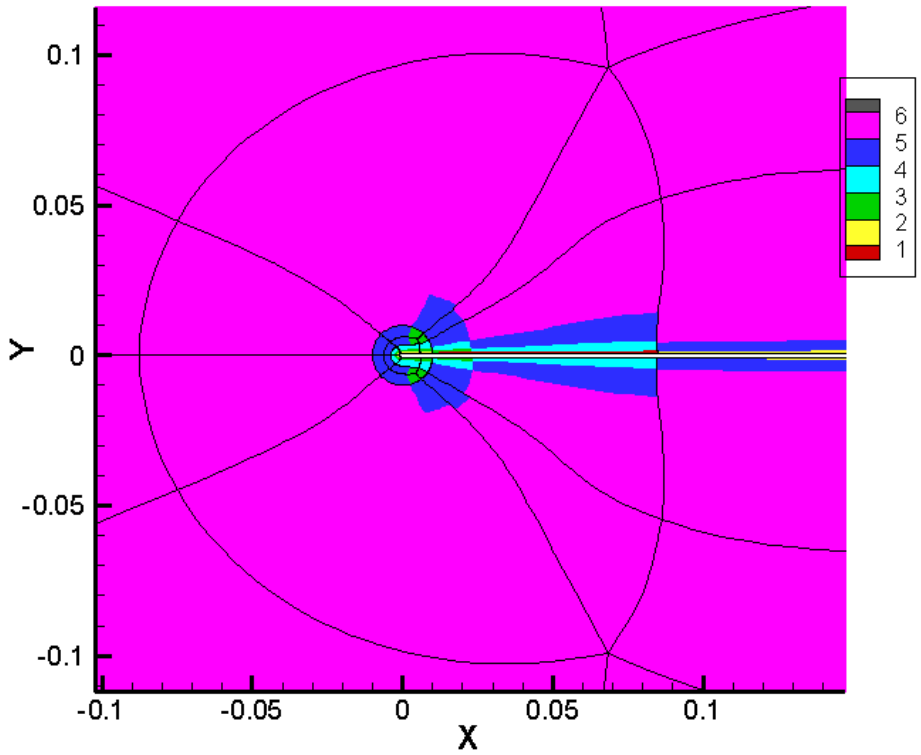


Figure 6-6 Distribution of the MTSAB levels near the leading edge, with block cutting

6.3.4 Results

The Reynolds number based on the length of the plate is 46,600 for case 1 and 466,000 for case 2. Figures 6.7-6.8 show velocity vectors and temperature contours near the stagnation point and the leading edge of the flat plate for both the length scales used. The grid resolution is clearly seen in these figures along with the growth of boundary layer as the flow moves downstream. Figures 6-9 and 6-11 show the unscaled boundary layer velocity profiles at nine streamwise locations along the upper surface of the flat plate. In these plots, the growth rate and the thickness of the boundary layers along the flat plate can be clearly seen for both the length scales used. Figures 6-10 and 6-12 show the boundary layer velocity profiles scaled using the Blasius similarity parameter η, given by Eq. (6.40). The profiles nearly collapse onto a single curve, which matches very well with Blasius similarity profile. Figures 6-13 and 6-14 compare the skin friction coefficient on the top surface of the plate with the skin friction coefficient given by Blasius (Eq. 6.47), for both the cases run. A good agreement in the slopes can be seen in these figures.

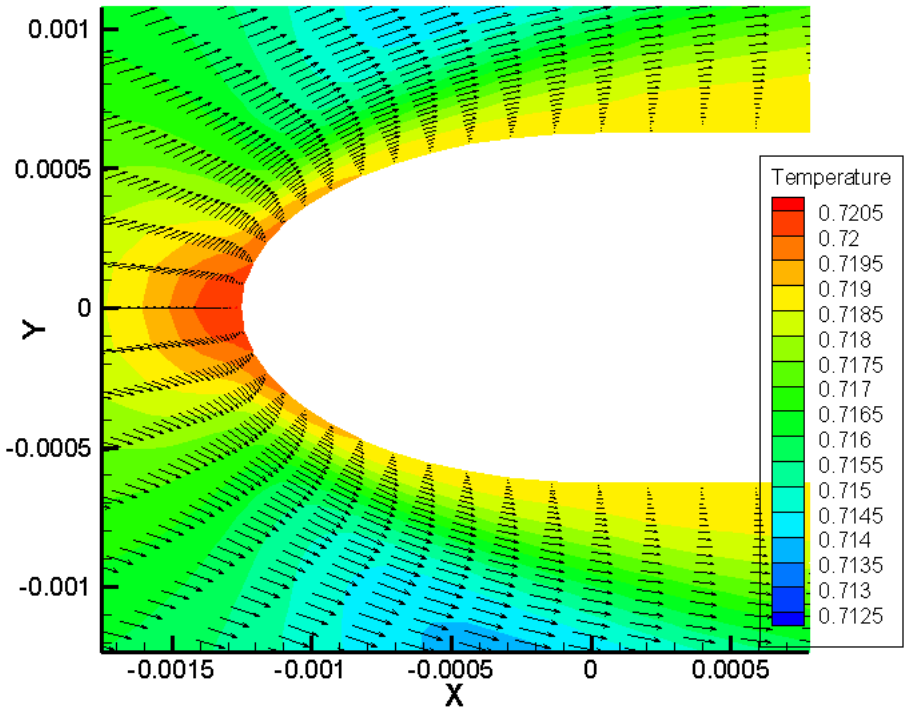


Figure 6-7: Flow development near the leading edge of the plate for length scale=0.01

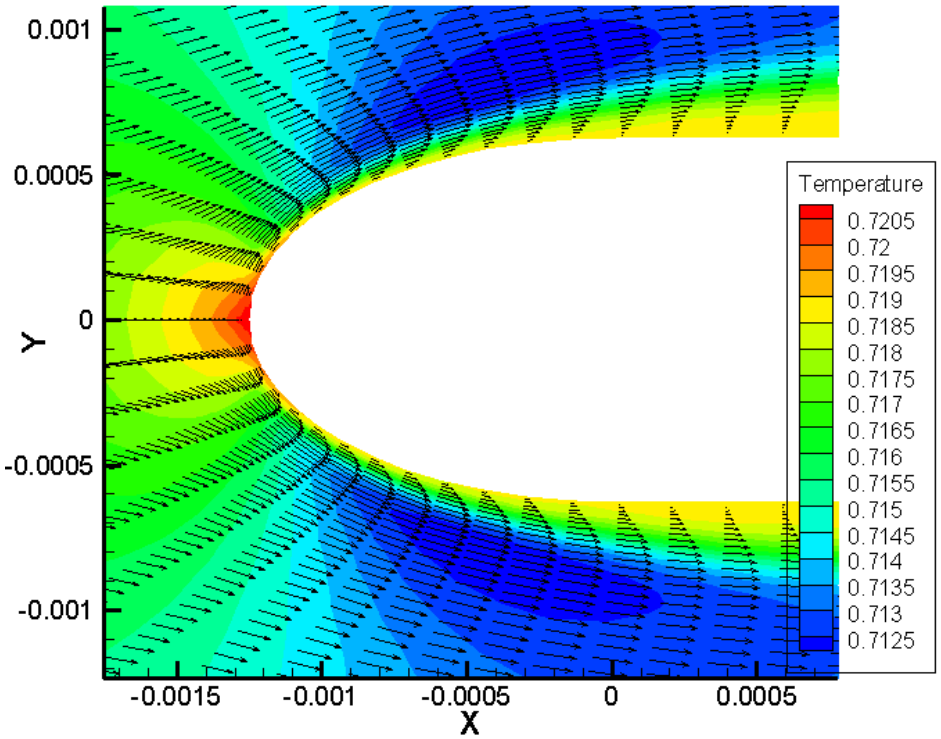


Figure 6-8: Flow development near the leading edge of the plate for length scale=0.1

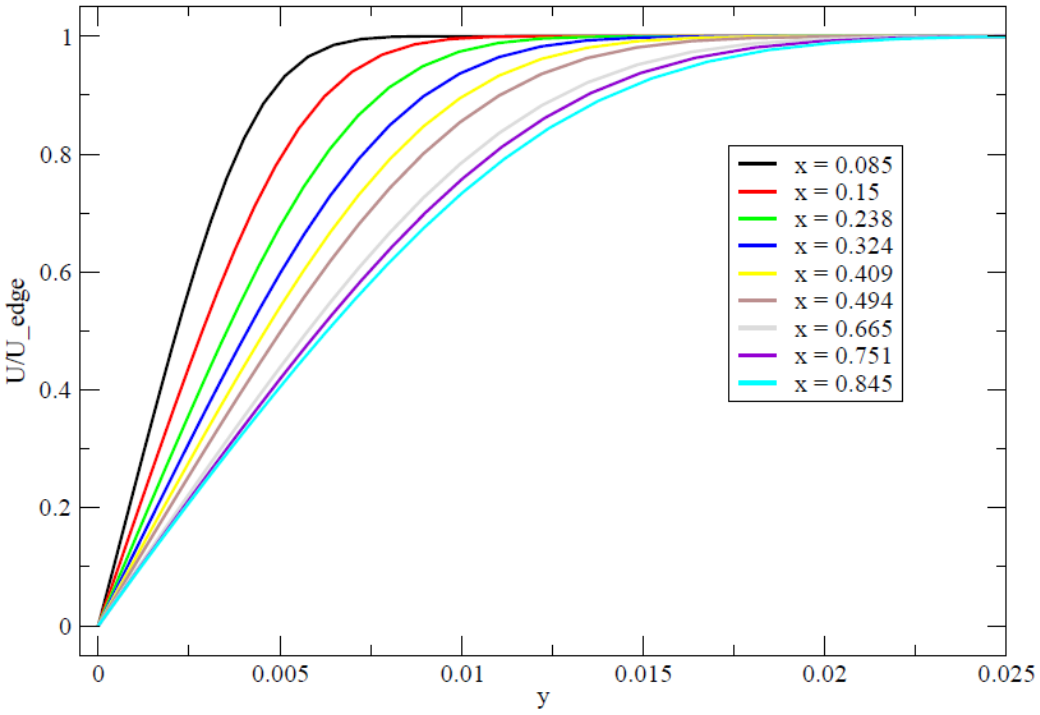


Figure 6-9: U/U_edge versus y at different x locations for the case with length scale=0.01

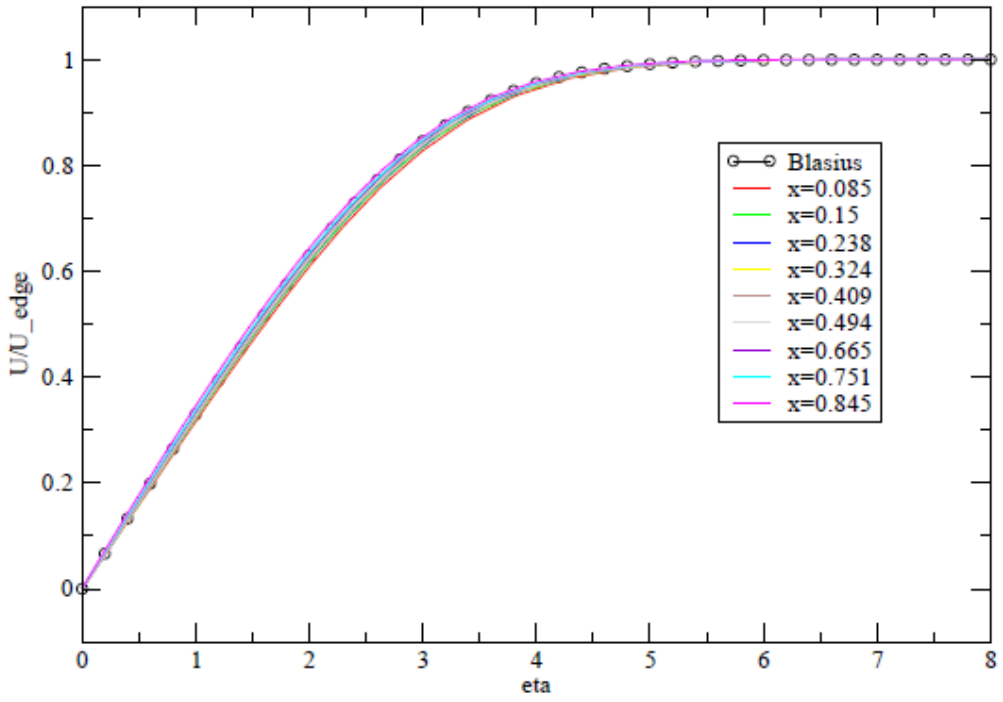


Figure 6-10: U/U_edge versus η at different x locations for case with length scale 0.01

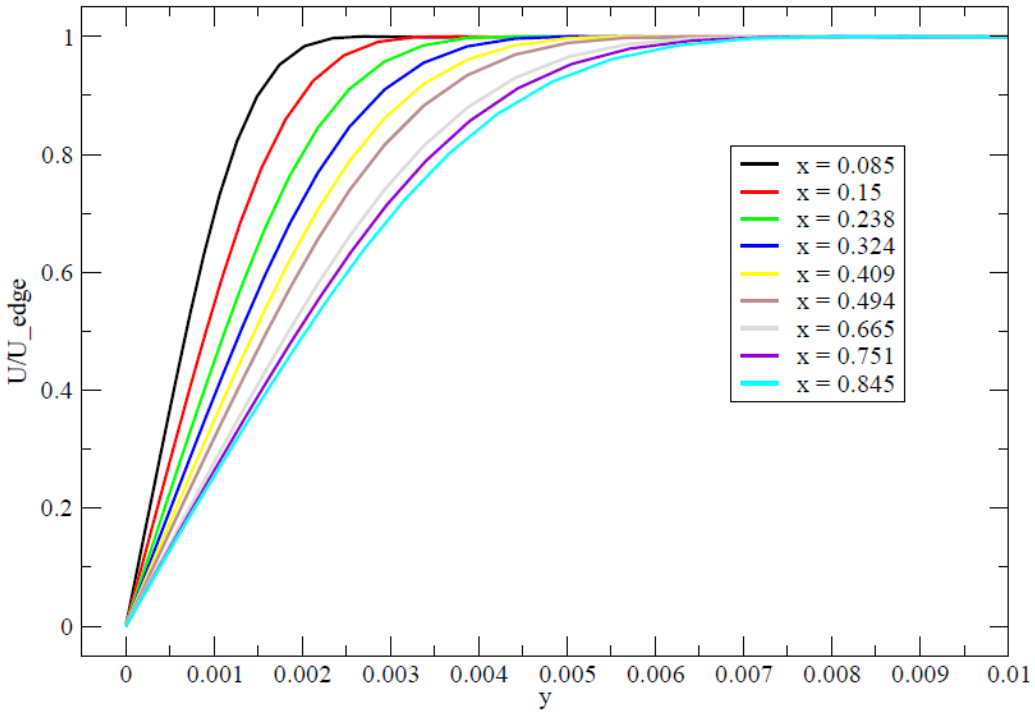


Figure 6.11: U/U_edge versus y at different x locations for the case with Length scale=0.1

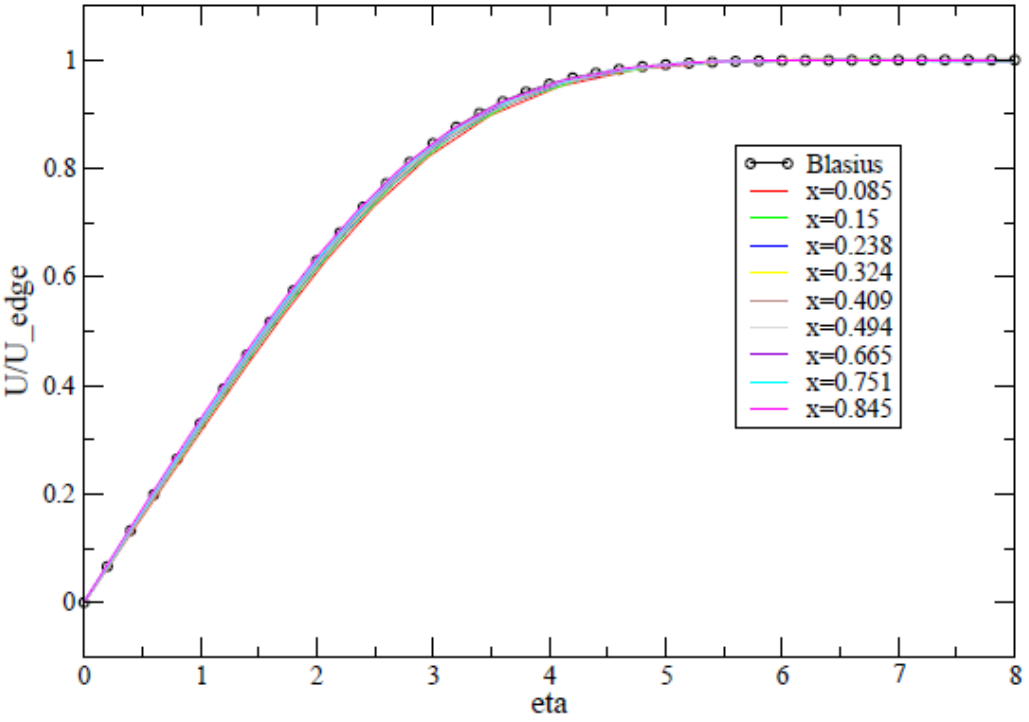


Figure 6-12: U/U_edge versus η at different x locations for the case with length scale=0.1

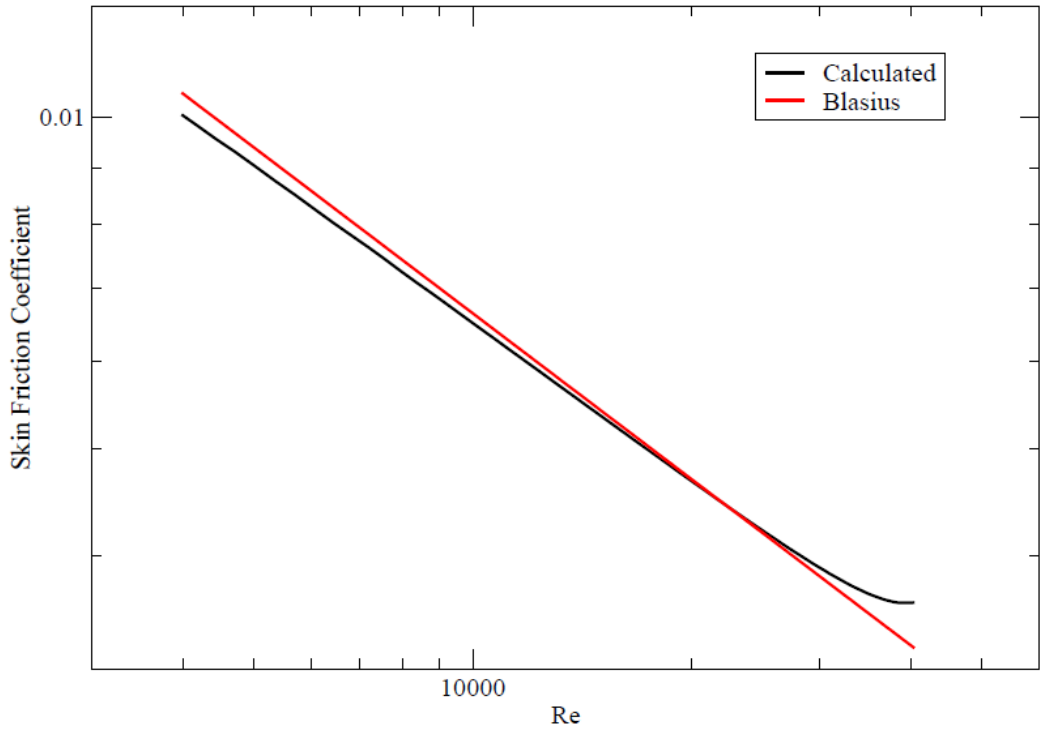


Figure 6-13: Skin friction coefficients along of the surface of the plate (for length scale=0.01)

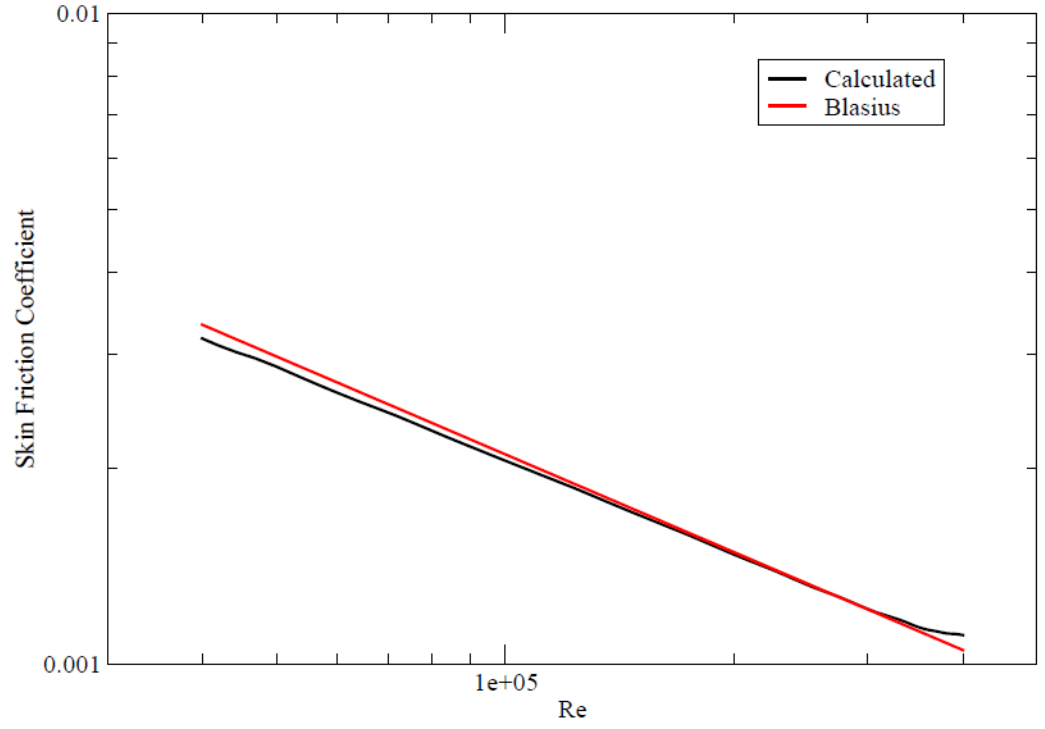


Figure 6-14: Skin friction coefficients along of the surface of the plate (for Length scale 0.1)

$\eta = y\sqrt{\frac{U_\infty}{vx}}$	F	$f' = \frac{u}{U_\infty}$	f''
0.0	0	0	0.33206
0.2	0.00664	0.6641	0.33199
0.4	0.02656	0.13277	0.33147
0.6	0.05974	0.19894	0.33008
0.8	0.10611	0.26471	0.32739
1.0	0.16557	0.32979	0.32301
1.2	0.23795	0.39378	0.31659
1.4	0.32298	0.45627	0.30787
1.6	0.42032	0.51676	0.29667
1.8	0.52952	0.57477	0.28293
2.0	0.65003	0.62977	0.26675
2.2	0.78120	0.67132	0.24835
2.4	0.92230	0.72899	0.22809
2.6	1.07252	0.77246	0.20646
2.8	1.23099	0.81152	0.18401
3.0	1.39682	0.84605	0.16136
3.2	1.56911	0.87609	0.13913
3.4	1.74696	0.90199	0.11788
3.6	1.92954	0.92333	0.09809
3.8	2.11605	0.94112	0.08013
4.0	2.30576	0.95552	0.06424
4.2	2.49806	0.96696	0.05052
4.4	2.69238	0.97587	0.03897
4.6	2.88826	0.98269	0.02948
4.8	3.08534	0.98779	0.02187
5.0	3.28329	0.99155	0.01591
5.2	3.48189	0.99425	0.01134
5.4	3.68094	0.99616	0.00793
5.6	3.88031	0.99748	0.00543
5.8	4.07990	0.99838	0.00365
6.0	4.27964	0.99898	0.00240
6.2	4.47948	0.99937	0.00155
6.4	4.67938	0.99961	0.00098
6.6	4.87931	0.99977	0.00061
6.8	5.07928	0.99987	0.00037
7.0	5.27926	0.99992	0.00022
7.2	5.47925	0.99996	0.00013
7.4	5.67924	0.99998	0.00007
7.6	5.87924	0.99999	0.00004
7.8	6.07923	1.00000	0.00002
8.0	6.27923	1.0000	0.00001
8.2	6.47923	1.00000	0.00001
8.4	6.67923	1.00000	0.00000
8.6	6.87923	1.00000	0.00000

Table 6.2 Function $f(\eta)$ for the boundary layer along the flat plate, after L. Howrath [51]

6.4 Flow over Cylinder at Re=150

External flows past a blunt body, such as circular cylinder, usually experiences a boundary layer separation and flow oscillations in the wake region behind the body. The pressure is maximum at the stagnation point and gradually decreases along the front half of the cylinder. The flow stays attached in the region with favorable pressure gradient. However the pressure starts to increase in the rear half of the cylinder resulting in an adverse pressure gradient. This adverse pressure gradient causes the flow to separate. For Re>45, the flow becomes unsteady and an alternate vortex shedding appears behind the circular cylinder, even though the imposed conditions are held steady [49]. This regular pattern of vortices in the wake is called Von-Karman Vortex Street. Flow over a two dimensional cylinder at Reynolds number of 150 was chosen to test the MTSAB scheme. At this Reynolds number the flow is essentially two dimensional and laminar. The results are compared with those of the existing experimental and numerical studies.

6.4.1 Grid and Numerical Details

An O-grid is used for this test case is generated using GridPro [31]. The center of the cylinder is at the origin (x=0, y=0). The non dimensional diameter of the cylinder is D=1.0. The boundaries extend to 25 diameters from the center of the circle. The complete and the close-up grid are shown by Figures 6-15 and 6-16. The grid has four blocks and a total of 404 grid points in the circumferential direction and 143 grid points in the radial details. On the cylinder surface the grid spacing in the radial direction is 0.001. The grid was stretched from the cylinder surface at the rate of 1.08. The incoming freestream flow is uniform with a Mach number 0f 0.2. Thompson boundary condition was used at the

inflow and the outflow boundaries .No-slip and adiabatic conditions were used on the cylinder surface. The reference variables for this test case are:

$$L_{ref} = 0.0000321834m$$

$$T_{ref} = 288.15K$$

$$P_{ref} = 101325.15N/m^2 \tag{6.50}$$

$$C_{ref} = 340.2625m/s$$

$$\rho_{ref} = 1.225235kg/m^3$$

$$\mu_{ref} = 0.00001789\ kg/ms$$

The initial flow and free stream quantities are:

$$\bar{\rho} = 1.0$$

$$p = 0.714285 \tag{6.51}$$

$$\bar{u} = 0.2$$

$$\bar{v} = 0.0$$

The Reynolds number is calculated as:

$$Re = \frac{(\rho \times \rho_{ref})(u \times C_{ref})(D \times L_{ref})}{\mu_{ref}} = 150 \tag{6.52}$$

6.4.2 MTSAB Performance

Without the use of the block cutting algorithm all the blocks are at level 1. Figures (6-17) and Figure (6-18) show the distribution of levels in the domain after the blocks are cut. The use of the block cutting algorithm breaks the existing grid blocks during the run

to get 6 levels shown by Figures 6-17 and 6-18. The number of points in each level is given by Table 6.3. It can be seen in this table that there is a close match between the speed up from the ideal distribution and the theoretical speed up after the blocks are cut. The small difference between them is from the fact the minimum length block length was set at 10 points. The theoretical speed after the blocks are cut is 5.25. The actual run time speed up for this case was 4.04.

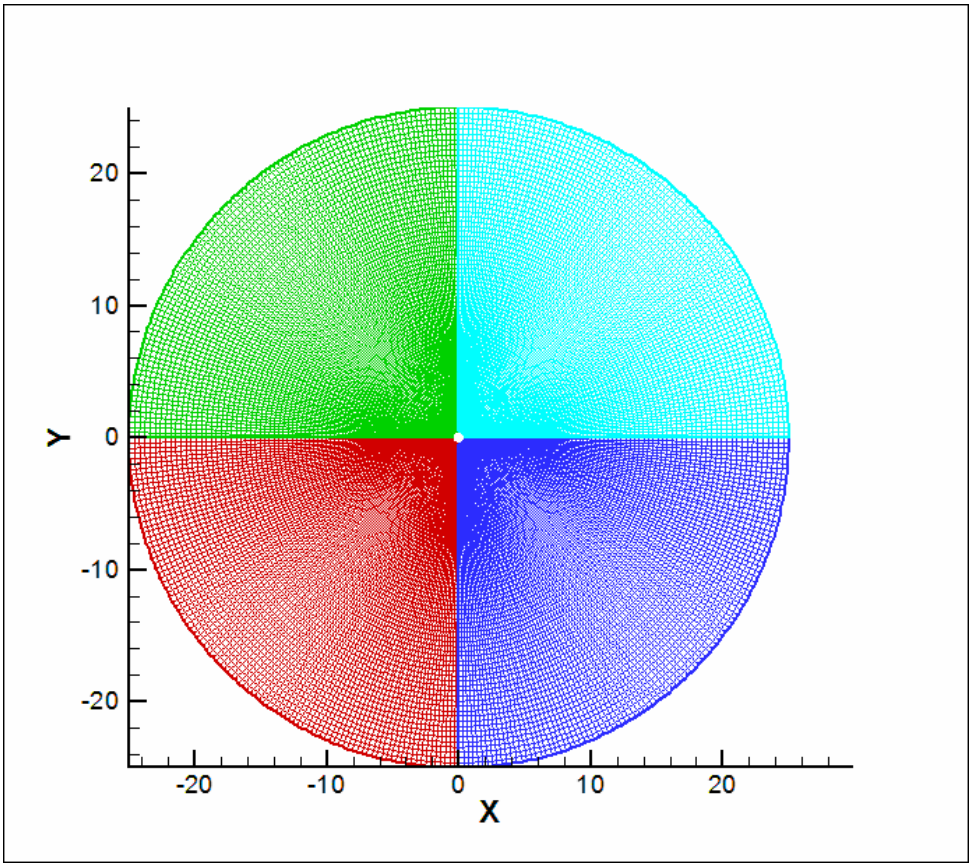


Figure 6-15: Complete grid for the 2D cylinder case

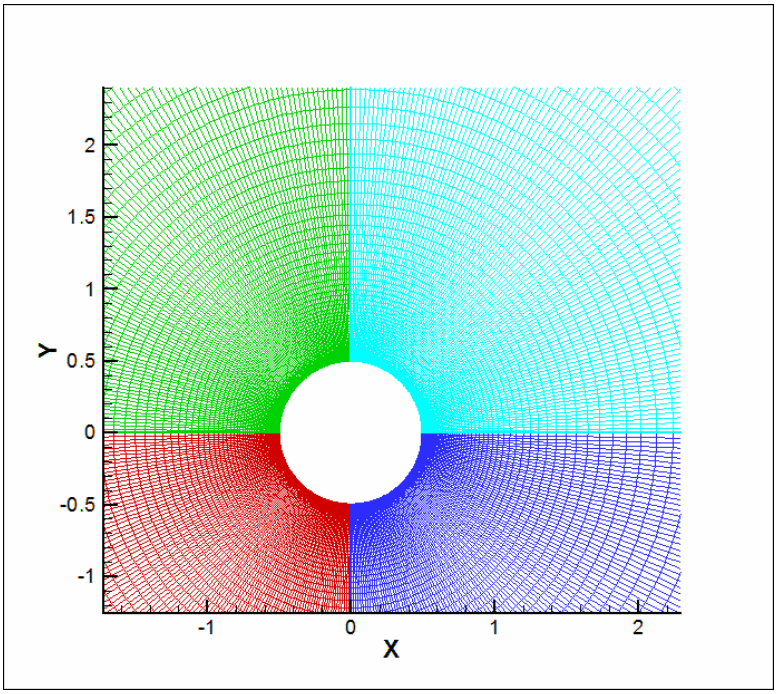


Figure 6-16: Close-up view of the grid near the cylinder

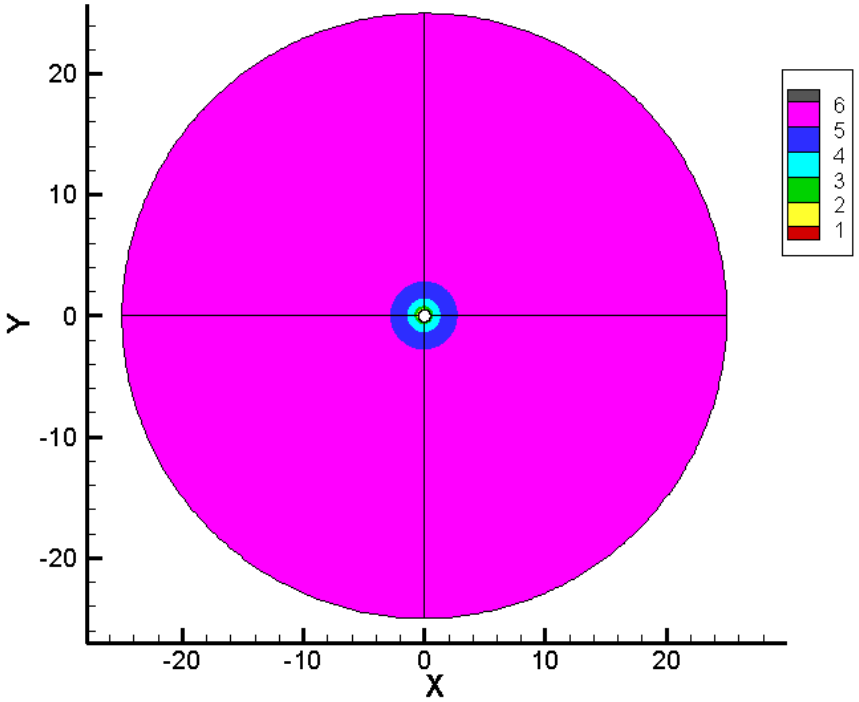


Figure 6-17: Levels for the blocks after cutting the blocks

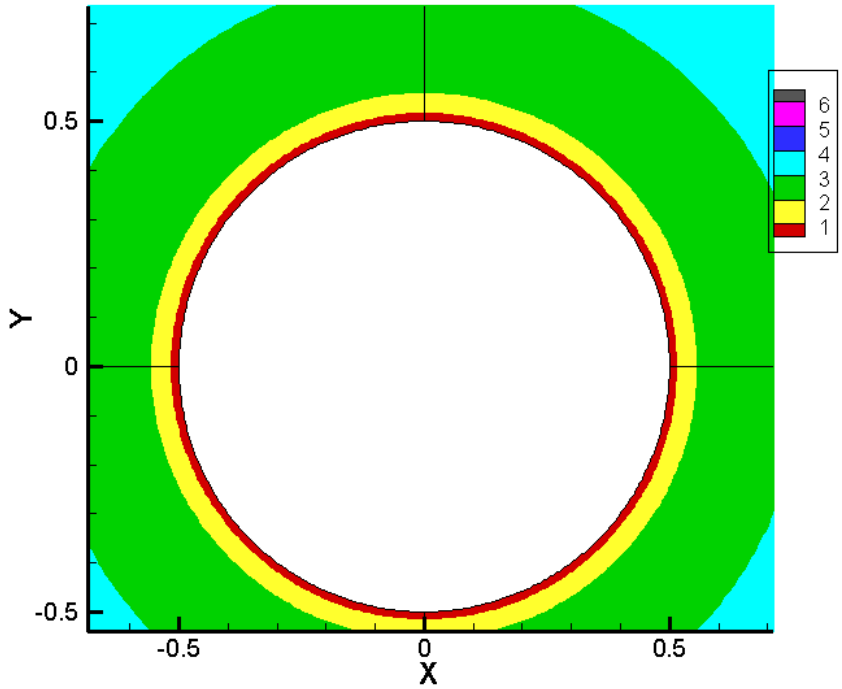


Figure 6-18: Distribution of levels near the cylinder after cutting the blocks

6.4.3 Results

Figure 6-19 show the instantaneous u-velocity in the computational domain. The asymmetric vortex shedding and resulting vortex street can be clearly seen downstream of the cylinder. Figure 6-20 shows the instantaneous velocity magnitude near the surface. In this figure, the boundary layer, separation and vortex formation near the surface is clearly seen. Figure 6-21 shows the instantaneous temperature distribution near the cylinder surface. The figure has contour lines showing the distribution of the temperature gradients. In this figure, the effect of the zero heat flux wall boundary condition is clearly evident; the contour lines are all normal to the wall, indicating a zero normal derivative of the temperature at the wall. The dimensionless frequency of this vortex shedding is represented by the Strouhal number. The Strouhal number is given as,

$$St = \frac{fD}{u} \tag{6.53}$$

where, D is the non dimensional diameter and u is the non dimensional free stream velocity. The parameter f is the frequency of the vortex shedding. To measure this quantity, u-velocity history at a point downstream of the cylinder and far away from the center of the wake is used. Figure 6-22 shows the history of the u-velocity for the point chosen. The initial conditions were symmetric and no attempt was made to force the asymmetric vortex shedding. The solution transitioned to asymmetric shedding as time went by. This transition can be seen for the chosen point in Figure 6-22. The value f is the inverse of the time between the peaks in the u-velocity history. The value Strouhal number calculated using this method is 0.185. This value compares well with the data given in [49] and [50]. Next, the lift and drag coefficients were calculated. Figure 6-23 shows plot of coefficient of lift and drag plotted against the non-dimensional time. Table 6.4 compares the data obtained from this simulation to the data given in [49] and [50]. The values of Strouhal number, fluctuation in the coefficient of lift and fluctuation in the coefficient of drag from the current simulation fit the trend of the values from [49] and [50]. The mean drag coefficient is 2-3% higher than the values from [49] and [50].

	Grid Points in Level 1	Grid Points in Level 2	Grid Points in Level 3	Gris Points in Level 4	Grid Points in Level 5	Grid Points in Level 6	Theoretical Speed up	Actual Speed up	Number of blocks
Ideal Distribution	4444	4516	6630	7188	5643	29351	5.48	-	-
Distribution Without Block Cutting	57772	0	0	0	0	0	None	None	4
Distribution With Block Cutting	4444	4848	8282	6666	5656	27876	5.25	4.04	24

Table 6.3 Speed up data from using the MTSAB scheme

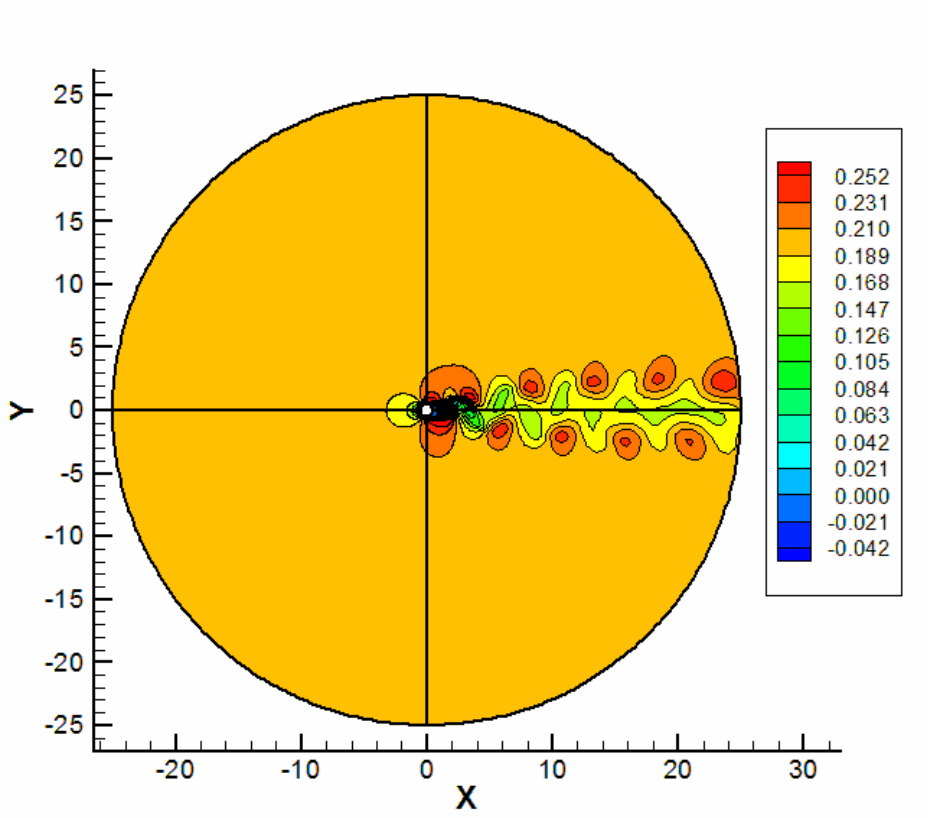


Figure 6-19: Instantaneous u-velocity contours

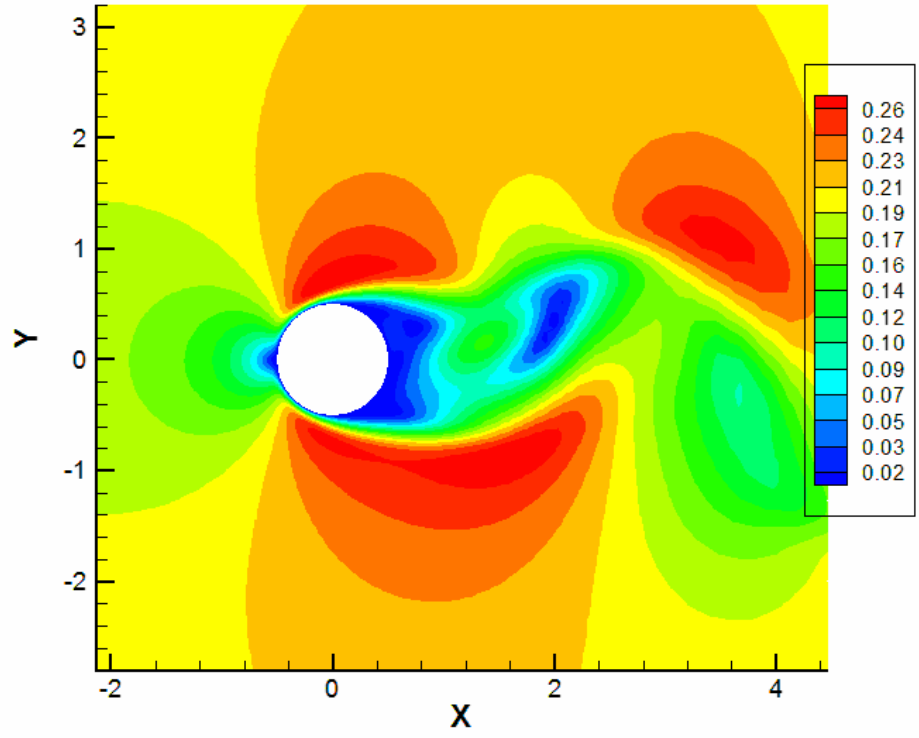


Figure 6-20: Instantaneous velocity magnitude contours

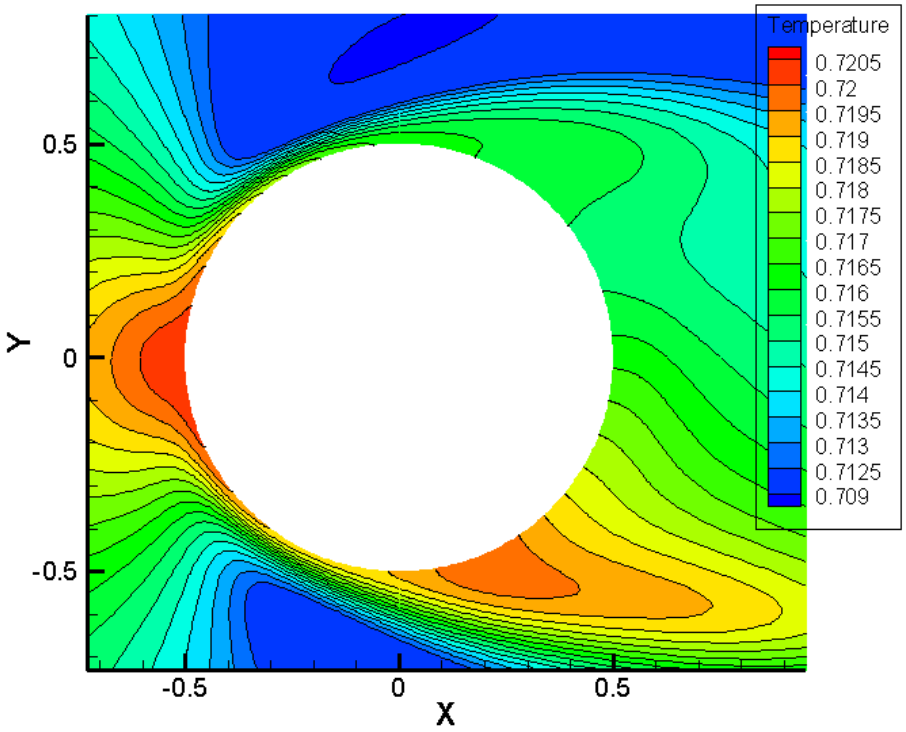


Figure 6-21: Instantaneous Temperature contours

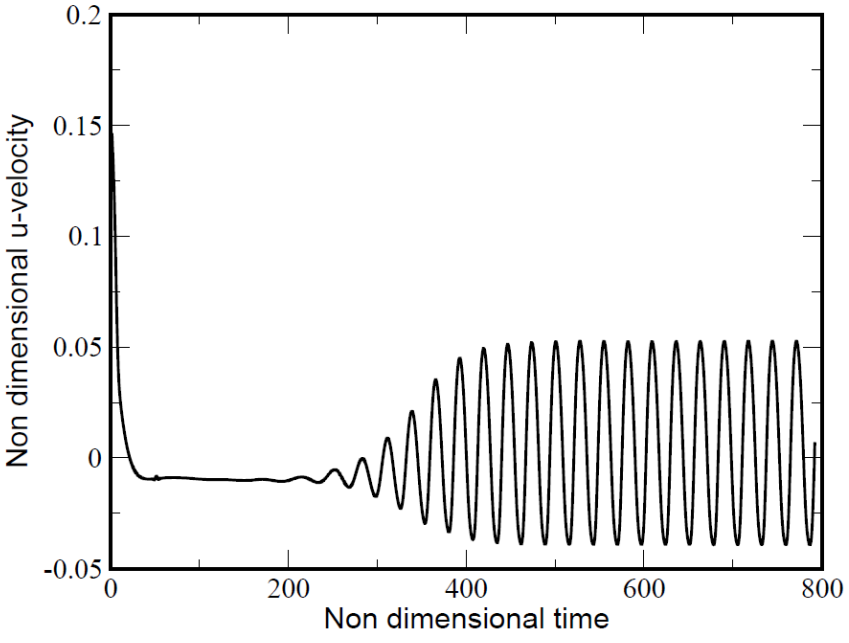


Figure 6-22: u-velocity history at a downstream location(x=0.87, y=0.31)

Re	Strouhal Number	C_d, mean	C_d, fluctuating	C_l, fluctuating
100 [50]	0.1569	1.3353	-	0.2534
140 [49]	0.182	1.32	0.0224	0.4823
150[Current]	0.185	1.3607	0.0242	0.5272
160[49]	0.188	1.32	0.0293	0.5501
200 [50]	0.1957	1.3365	-	0.6002

Table 6.4: Comparison of calculated data with references

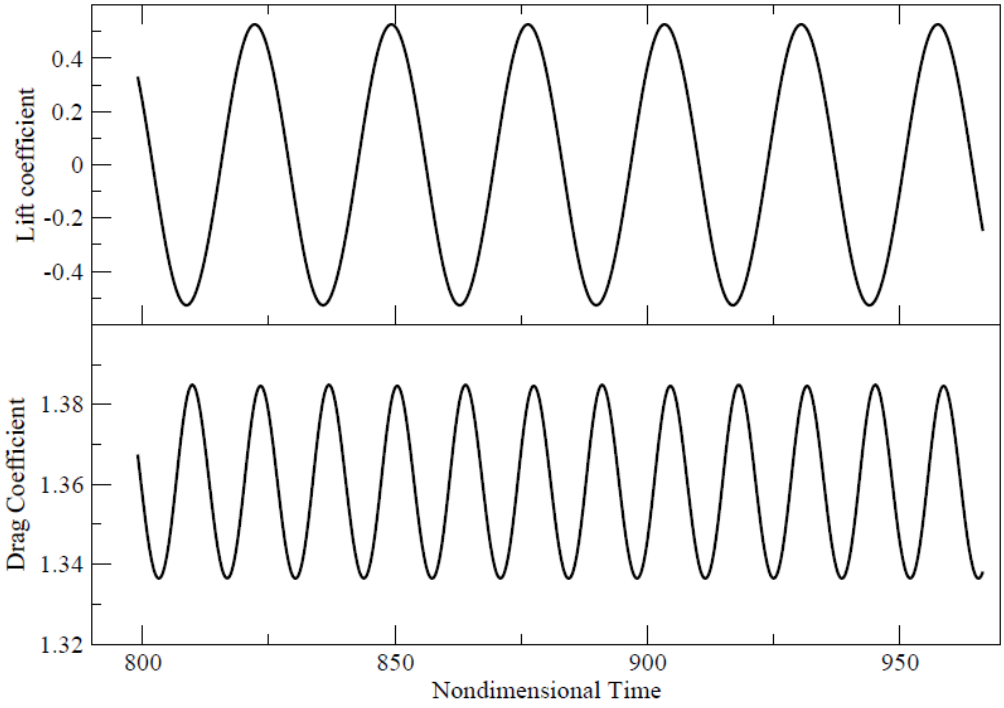


Figure 6-23: Total drag and lift coefficients

Chapter Seven

Conclusions and Future Work

A new Multi-Time-Stepping Adams-Bashforth (MTSAB) scheme was developed to increase the speed of a code using an explicit time marching scheme, which performs time accurate calculations. The scheme was then implemented in NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulation Code.

It can be difficult to use a Multi-Time Stepping scheme for complex problems without the automation methods that were developed as a part of this work. Two automated block cutting algorithms were developed.

The first algorithm was aimed at increasing speed the MTSAB scheme by increasing the number of grid points marching close to their stable time steps. This algorithm efficiently cut the blocks, based on the point to point distribution of local stable time steps (or levels). Manually cutting the blocks can be difficult, as it was seen for the gust airfoil problem in Chapter 4.

The point to point distributions of stable time steps can change during the run for cases with grid motion and flows with large disturbances. The automation developed for the MTSAB scheme takes this change into consideration and assigns new levels (Dynamic

Level Redistribution) to the blocks during the run. During this redistribution process, the block cutting algorithm cuts new blocks at different levels, based on the new point to point distribution. The second automated block cutting algorithm was developed to increase the parallel efficiency of the MTSAB scheme. This was implemented at every MTSAB level to achieve a good parallel efficiency of the scheme.

The automated MTSAB scheme was also extended to work with grid motion. A plunging airfoil case was shown to work with the scheme. The Dynamic Redistribution of Levels was shown to be particularly useful for this case.

The automated MTSAB scheme was also extended to work for viscous flow calculations. Several validation cases were tested to validate the automated MTSAB scheme for Computational Aero Acoustics (CAA) Workshop problems, steady and unsteady flows with shocks and viscous flows. The results based on these tests show a good performance of the scheme for the problems tested.

One of the major challenges faced in this work was to reduce the overhead from the buffer block calculations. Although, the first block cutting algorithm cut the blocks efficiently, as the number of blocks increased, so did the number of buffer blocks.
For some cases, the speed increase from cutting the blocks was less than the speed reduction from the increased buffer block overhead. More efficient ways of implementing buffer blocks will be investigated in the near future.

As the development of the BASS code continues, the goal is to have a multi-size grid implementation in the BASS code. This is analogous to multi-time steeping, but is space (while still resolving all the unsteady dynamics). This, in conjunction with the MTSAB scheme can further reduce the run times. Coarsening of an existing grid block will

basically increase the stable time step in the block and hence increasing the MTSAB level of the block.

References

[1] Tam, C. K. W., "Computational Aeroacoustics: Issues and Methods," AIAA Journal, Vol. 33, No. 10, 1995, pp. 1788-96.

[2] Lele, S. K., "Computational Aeroacoustics: A Review", AIAA Paper 97-0018, Reno, NV, 1997.

[3] Lele, S. K., "Compact Finite-Difference Schemes with Spectral-Like Resolution," Journal of Computational Physics, Vol. 103, 1992, pp. 16-42.

[4] Tam, C. K. W. and Webb, J. C., "Dispersion-Relation-Preserving Finite-Difference Schemes for Computational Acoustics," Journal of Computational Physics, Vol. 107, 1993, pp. 262-281.

[5] Kim, J. W. and Lee, D. J., "Optimized Compact Finite-Difference Schemes with Maximum Resolution," AIAA Journal, Vol. 34, 1996, pp. 887-893.

[6] Hixon, R., "Prefactored Small-Stencil Compact Schemes," Journal of Computational Physics, Vol. 165, 2000, pp. 522-541.

[7] Visbal, M. R., and Gaitonde, D. V., "Very High Order Spatially Implicit Schemes for Computational Acoustics on Curvilinear Meshes," Journal of Computational Acoustics, Vol. 9, 2001, pp. 1259-1286.

[8] Hu, F. Q., Hussaini, M. Y., and Manthey, J., "Low Dissipation and Dispersion Runge-Kutta Schemes for Computational Acoustics," Journal of Computational Physics, Vol. 124, 1996, pp. 177-191.

[9] Stanescu, D., and Habashi, W. G., "2N-Storage Low Dissipation and Dispersion Runge-Kutta Schemes for Computational Acoustics," Journal of Computational Physics, Vol. 143, 1998, pp. 674-681.

[10] Calvo, M., Franco, J. M., and Randez, L, "A New Minimum Storage Runge-Kutta Scheme for Computational Acoustics," Journal of Computational Physics, Vol. 201, 2004, pp. 1-12.

[11] Hardin, J. C., Ristorcelli, J. R., and Tam, C. K. W. (eds), ICASE/LaRC Workshop on Benchmark Problems in Computational Aeroacoustics (CAA), NASA CP-3300, Hampton, VA, 1994.

[12] Tam, C. K. W., and Hardin, J. C. (eds), Second Computational Aeroacoustics (CAA) Workshop on Benchmark Problems, NASA CP-3352, Hampton, VA, 1996.

[13] Dahl, M. (ed), Third Computational Aeroacoustics (CAA) Workshop on Benchmark Problems, NASA CP-2000-209790, Cleveland, OH, 2000.

[14] Dahl, M. (ed), Fourth Computational Aeroacoustics (CAA) Workshop on Benchmark Problems, NASA CP-2004-212954, Cleveland, OH, 2004

[15] John, D. Anderson, Jr. Computational Fluid Dynamics: The Basics with Application McGraw Hill, 1995

[16] Rizzetta, D., and Visbal, M. and G.A. Blaisdell, "A time-implicit high-order compact differencing and filtering scheme for large eddy simulation," International Journal of Numerical Methods in Fluids 2003, Volume 42, Issue 6, pp. 665-693

[17] Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," AIAA Paper 91-1596, Honolulu, HI, June 1991.

[18] Allampalli, V., Hixon, R., Nallasamy, M., & Sawyer, S., "High-Accuracy Large-Step Explicit Runge-Kutta (HALE-RK) Schemes for Computational Aeroacoustics," Journal of Computational Physics, Vol. 228, No. 10, 2009, pp 3837-3850.

[19] Tam, C. K. W and, K. A., Kurbastkii, "Multi-size-mesh Multi-time-step Dispersion-relation preserving scheme for Multi-scales Aeroacoustics Problems," International Journal of Computational Fluid Dynamics, Vol. 17(2), 2003, pp. 119-132.

[20] Shen, H., and Tam, C. K. W., "Three Dimensional Numerical Simulation of the Jet Screech Phenomenon," AIAA Journal, Vol. 40, 2002, pp. 33-41.

[21] Liu, L., Li, X., Hu, Q. Fang., "Nonuniform Time-step Runge-Kutta Discontinuous Galerkin Method for Computational Aeroacoustics", AIAA Paper 2009-3114, 15th AIAA/CEAS Aeroacoustics Conference, Miami, Florida", May 2009

[22] Hixon, R., Nallasamy, M., Sawyer, S.D, "Parallelization Strategy for an Explicit Computational Aeroacoustic Code," 8th AIAA Aeroacoustics Conference, 2002

[23] Hixon, R., Bhate, D. L, "Shock-Capturing Dissipation Schemes for High-Accuracy Computational Aeroacoustics (CAA) Codes," AIAA Paper 2006-2413, 2006.

[24] Sescu, A., Hixon, R., "Validation of a CAA Code Using a Benchmark Wake-Stator Interaction Problem", AIAA Paper 2009-3340, 15th AIAA/CEAS Aeroacoustics Conference, Miami, Florida", May 2009

[25] http://www.cs.umn.edu/~metis

[26] Allampalli, V., "Validation a Linearized Euler Equation Code for the Prediction of Jet Noise", M.S Thesis Mechanical, Industrial and Manufacturing Engineering Department, The University of Toledo, December 2004.

[27] Hixon, R., 'Curvilinear Wall Boundary Conditions for Computational Aeroacoustics', AIAA Paper 99-2395, June 1999.

[28] Hixon,R.,"Radiation and Wall Boundary Conditions for Computational Aeroacoustics": A Review", International Journal of Computational Fluid Dynamics, Vol. 18, Number 6, August 2004, pp. 523-531.

[29] Hixon, R., Allampalli, V. "Optimization of Finite-Difference Boundary Stencils for Improved Viscous Stability," AIAA Paper 2010-837, 48th American Institute of Aeronautics and Astronautics (AIAA) Aerospace Sciences Meeting Including the New Horizons Forums and Aerospace Exposition, Orlando, FL, 2010-0837, January 2010.

[30] Hixon, R., Golubev, V., Mankbadi, R.R., Scott, J.R, Sawyer, S., Nallasamy, M., Application of a nonlinear computational aeroacoustics code to the gust-airfoil problem, AIAA J. 44 (2006) 323–328.

[31] P.R.Eiseman, et al., GridPro/az 3000, "User's Guide and Reference Manual", 111 pp., Program Development Corporation of Scarsdale, NY.

[32] Kennedy, C. A., and Carpenter, M. H., "Several New Numerical Methods for Compressible Shear-Layer Simulations," *Applied Numerical Simulations*, Vol. 14, 1994, pp. 397-433

[33] Hixon, R., Shih, R.R., Mankbadi, "Evaluation of the boundary conditions for Computational Aeroacoustics" AIAA Journal, Vol. 33 (1995), 2012

[34] Giles, M., "Nonreflecting boundary conditions for Euler equation calculations", AIAA Journal, Vol. 28 (1990) 2050–8.

[35] Romenski, E., Titarev, V. A., and Toro, E. F., 'Perfectly Matched Layers with High Rate Damping for Hyperbolic Systems', CFD J., Vol. 15, No. 2, 2006, pp. 240-246.

[36] G Goodrich, J. A Comparison of Three PML Treatments for CAA (and CFD), AIAA Paper 2008-2922, May 2008.

[37] Tyler, J.M., Sofrin, T.G., Axial Flow Compressor Noise Studies, SAE Trans., Vol. 70, 1962, pp. 397-433.

[38] Jameson, A., Schmidt, W., and Turkel, E., „Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes', AIAA Paper 81-1259,1981.

[39] Swanson, R. C. and Turkel, E., 'Artificial Dissipation and Central Difference Schemes for the Euler and Navier-Stokes Equations', AIAA Computational Fluid Dynamics Conference, AIAA-87-1107-CP

[40] Tam, C. K. W. and Shen, Hao, „Direct Computation of Nonlinear Acoustic Pulses using High-Order Finite Difference Schemes', AIAA Paper 1993-4325.

[41] Lockard, D. P. and Morris, P. J., „A Parallel Implementation of a Computational Aeroacoustic Algorithm for Airfoil Noise', *Journal of Computational Acoustics*, Vol. 5, No. 4, 1997, pp. 337-353.

[42] Visbal, M. R. and Gaitonde, D. V., „Shock Capturing Using Compact Differencing Methods', AIAA Paper 2005-1265,2005

[43] Bhate, D. L., „Shock-Capturing Artificial Dissipation Model for Higher-Order Schemes', M. S. Thesis, Mechanical, Industrial, and Manufacturing Engineering Department, The University of Toledo, December 2005.

[44] A Steger, J.L, „Implicit Finite Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries', *AIAA Journal*, Vol. 16, No. 7, 1978, pp. 679-686.

[45] A Hixon, R., 'High-Accuracy Moving Wall Boundary Conditions for Computational Aeroacoustics', AIAA Paper 2008-0030, Jan. 2008.

[46] Thompson, K.W., "Time dependent boundary conditions for hyperbolic systems," Journal of Computational Physics, Vol. 68, 1987, pp. 1-24

[47] Schlicting, H., "Boundary Layer Theory." McGraw Hill, Seventh Edition, 1979.

[48] Hixon, R., Nallasamy, M., and Sawyer, S., 'A Method for the Implementation of Boundary Conditions in High-Accuracy Finite-Difference Schemes', AIAA Paper 2005-0608, Jan. 2005.

[49] Park, J., Kwon, K., Choi, H., "Numerical Solutions of Flow Past a Circular Cylinder at Reynolds Number up to 160", KSME International Journal, Volume 12, No. 6, pp 1200-1205,1998

[50] Rajani, B. N., Kandasamy, Majumdar, S., "Numerical Simulation of Laminar Flow Pasta Circular Cylinder", Applied Mathematical Modeling, Volume 33, pp. 1228-1247, 2009

[51] Howrath, L. “On the solution of the Laminar Boundary Layer Equations”, Proceeding of The Royal Society of London, A, 164, 547-579 (1938)