Towards the Numerical Simulation of Dynamic Stall

Debojyoti Ghosh*

Adviser: Dr. James Baeder†

AMSC 663/664 - Advanced Scientific Computation I/II
Project Report (May 2008)

Abstract

A two-dimensional flow solver is developed and applied to flow around airfoils. It is intended to be a first step in the numerical solution of dynamic stalling of rotor blades. The inviscid Euler equations are solved and the algorithm is validated on benchmark problems, on Cartesian as well as curvilinear meshes. The Essentially Non-Oscillatory class of schemes are used in conjunction with Local Lax-Friedrich's upwinding for high order spatial accuracy. Explicit time integration is carried out using the Runge-Kutta family of ODE solvers. The algorithm is extended to the Navier Stokes equations. Implicit time integration based on the Backward Euler scheme is used because of the overly restrictive time step restriction for explicit time integration. The algorithm is validated on simple test problems as well as viscous flows around airfoils.

1 Introduction

Dynamic stalling of the rotor blade [1] is one of the major factors limiting the performance of rotorcrafts, especially when they are operating at high angles of attack and/or high forward flight velocities. It is essentially an unsteady phenomenon where the time-varying velocity of the incoming flow as well as time-varying angle of attack of the rotor blade have a substantial effect on the aerodynamic nature of the flow. Dynamic stalling causes the aerodynamic loads on the blade to change drastically but the nature and timing of this change is different from that seen for steady flow around airfoils. There is a sudden loss in lift as well as a sudden increase in the torsional forces on the blade, both of which are undesirable from aerodynamic and structural considerations. Thus, the accurate computation of the unsteady flow around a rotor blade and the prediction of dynamic stall is of utmost importance in the field of rotorcraft aerodynamics.

The goal for this project was to develop a two-dimensional viscous flow solver, with specific application for flow around airfoils. This is required for the broader objective of numerically simulating the dynamic stalling of rotor blades. Initially the inviscid Euler equations were to be solved and the algorithm validated for several 1D and 2D benchmark problems.

*Department of Mathematics, University of Maryland, College Park, Email: ghosh [at] umd.edu

†Department of Aerospace Engineering, University of Maryland, College Park, Email: baeder [at] eng.umd.edu

These problems would test the accuracy and robustness of the algorithm, especially with respect to shock capturing. The Euler solver would then be extended to the Navier Stokes equations by including the dissipative terms. It was also planned to include a turbulence model for more accurate computations in the boundary layer. The algorithm would then be validated on problems already solved in literature and subsequently used to generate results pertinent to dynamic stalling.

The airfoil is a basic 2D lifting body and can be thought of as a cross-section of a typical wing or rotor blade. While flow around wings and rotors are three-dimensional by nature, involving cross-flows, a 2D computation of the flow, assuming uniform flow along the span of the wing/blade gives a preliminary idea of the flow and the aerodynamic forces. While the 3D effects are important near the root and tip of the wing/blade, flow over large parts in the interior can be modeled as a two-dimensional flow with flow properties assumed constant in the span-wise direction. Thus, computation and analysis of the flow around an airfoil is a starting point in the modeling of flows past wings and blades. The present study is aimed at simulating and analyzing the 2D flow around an airfoil when subjected to conditions seen by rotorcraft blades.

The inviscid Euler equations [2, 3] can be derived from the Navier-Stokes equations by assuming an inviscid fluid with no heat conduction. They are considerably simpler and easier to solve. While the Euler equations model an ideal fluid, they are still very important in high speed flows. A typical solid body in high speed flow is immersed in a boundary layer in which the velocity of the fluid varies drastically as one moves away from the surface. This is because a viscous fluid sticks to the body at the surface (hence a zero velocity) while it flows at a certain velocity away from the surface. Typically, in high speed flows, the thickness of this boundary layer is very small compared to the characteristic dimensions of the flow. Therefore, it is a reasonable approximation to model the flow outside the boundary layer by inviscid flow equations (the Euler equations) and then make corrections for the viscous effects inside the boundary layer. While computing flows around airfoils, inviscid computations provide a very accurate idea of the lifting forces (which are primarily caused by the pressure differences).

The Euler solver can then be extended to solve the Navier Stokes equations. While the convective part remains the same in these two systems of equations, the Navier Stokes contains dissipative terms which account for the effect of viscous forces and heat conduction on the momentum and energy conservation equations. The accurate prediction of drag requires the inclusion of viscous terms to account for the shear forces on the body and thus, the solution of the Navier Stokes equations provides a more complete picture of the flow near solid surfaces (boundary layers). Additionally, some important features of flows around airfoils like flow separation and stalling are viscous phenomena and thus can be modeled with a Navier Stokes solver.

The present report concentrates on the development of a 2D Navier Stokes solver through various stages, starting with 1D Euler computations to viscous flow computations over an airfoil. For the Euler equations, a finite volume approach [4] is used to discretize the governing equations. Since the governing equations are hyperbolic in nature, the solution is composed of waves traveling in various directions and a characteristic based algorithm has been used to model this. Additionally, the solution to a hyperbolic system need not be smooth and to prevent the numerical scheme from exhibiting oscillations around discontinuities for higher order computations, the Essentially Non-Oscillatory (ENO) schemes [5] are used for flux

reconstruction. While inviscid computations have been carried out using explicit time integration, the geometric stiffness associated with stretched meshes for viscous solutions has necessitated the use of implicit time integration. The extension of the Euler solver to Navier Stokes involved the adoption of the curvilinear form of the governing equations and Euler backward time stepping while retaining the finite volume based computation of convective terms. These aspects of the algorithm are described more elaborately in subsequent sections.

The report is outlined in the following manner. Section 2 describes the phenomenon of dynamic stalling in brief. Section 3 provides the governing equations as well a brief explanation. Section 4 briefly describes the Essentially Non-Oscillatory technique of interpolation which is used in the numerical treatment of the governing equations. Section 5 describes the numerical scheme used to discretize and solve these equations while Section 5 gives an overview of the boundary conditions required. Subsequent sections deal with the validation of the algorithm over a number of 1D and 2D problems which are considered as benchmarks.

2 Flow around Airfoils - Brief Description

In a steady flow around an airfoil, the flow, and thus the pressure distribution on the surface, is dependent on the angle of attack, i.e, the angle between the free-stream velocity and the airfoil chord (line joining leading and trailing edges). The higher the angle, the greater is the perturbation to the flow, causing higher velocities and lower pressures over the upper surface. The pressure minimum (suction peak) occurs towards the leading edge of the airfoil and the flow over the upper surface downstream of the suction peak experiences an adverse pressure gradient. Under usual conditions, the momentum of the flow helps overcome the pressure gradient. However at high angles of attack, the adverse pressure gradient become too high, causing the flow to separate from the surface of the airfoil. The region of flow right next to the upper surface sees reverse flow and the pressure distribution over the upper surface causing the lifting force is destroyed. This phenomena is termed "stalling" and is characterized by a sudden loss of lift at a high angle of attack. The design of a fixed wing aircraft involves making sure that the maximum angle of attack is never exceeded in any flight condition.

The flow around a rotor blade is complicated due to its unsteady nature. While the flow around a fixed wing can be modeled using steady state flows, modeling flows around rotors has to take into account the time-varying angles of attack as well as free-stream velocities. The rotor is constructed such that during one full rotation, the angle of attack changes sinusoidally. It is at a minimum when the blade is advancing and at a maximum when the blade is retreating. The free-stream velocity that a given cross-section of the blade sees is also a function of time. While the blade is advancing, the free-stream velocity is at a maximum since it is a sum of the forward flight speed as well as the linear blade speed. On the other hand, while the blade is retreating, the free-stream velocity is at a minimum since the blade movement and the rotorcraft motion are in opposite directions. In addition to these unsteady effects, the blade is also subject to the wake from the preceding blade. However, this will not be modeled in the present study.

The unsteady nature of the flow around the rotor blade makes it susceptible to dynamic stalling. The general outline of the process can be describes as follows. As the angle of attack increases from minimum, the lift generated increases as the pressures over the upper surface decrease. As the stall-point is reached, the flow starts to separate from the trailing edge. As

the angle of attack increases further, beyond the static stall limit, the flow separation moves upstream to the leading edge. By this time, the flow over the upper surface can separated completely and a vortex is formed in the separated region. The vortex creates a region of low pressure and thus augments the lift, beyond its static stall limit. However, as the vortex starts convecting downstream, it causes a high nose-down pitching moment. This stage is referred to as the "moment stall" and causes high torsional loads on the blade. As the angle of attack increases further, the vortex convects downstream and is shed off the trailing edge, thus causing a sudden loss of lifting force. This stage is called as the "lift stall". By this time, the airfoil is at its highest angle of attack and starts pitching downwards. As the angle of attack decreases, the flow starts reattaching and by the time the airfoil is at its minimum angle of attack, the flow is attached and well-behaved.

3 Governing Equations

3.1 Inviscid Euler Equations

The Euler equations are the governing equations for an inviscid, compressible fluid. They consist of the conservation of mass, momentum and energy, when applied to a fluid element in the flow. In the present study, the ID Euler equations are solved initially to validate the algorithm and its performance for various orders of accuracy. Simple benchmark 1D problems provide good validation cases which may not be possible in 2D because of the more complicated nature of flow. Subsequent to validating the algorithm in 1D, it is extended to 2D and validated. The 1D and 2D Euler equations are presented below in their differential forms.

1D Euler Equations

$$\mathbf{u}_t + \mathbf{f}_x = 0 \tag{1}$$

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \ \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ (E+P)u \end{bmatrix}, \tag{2}$$

Here, ρ is the fluid density, u is the velocity, P is the pressure and E is the total energy. In addition to these three equations, the equation of state relates the total energy to the flow variables.

$$E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho u^2 \tag{3}$$

where γ is the ratio of specific heats.

2D Euler Equations

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla . \mathcal{F} = 0 \tag{4}$$

where the flux is $\mathcal{F} = \mathbf{f}\hat{\mathbf{i}} + \mathbf{g}\hat{\mathbf{j}}$, and

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \ \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ (E+P)u \end{bmatrix}, \ \mathbf{g}(\mathbf{u}) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ (E+P)v \end{bmatrix} \tag{5}$$

The variables have the same meaning as in 1D and u, v are the Cartesian components of the fluid velocity. The equation of state for 2D flow is

$$E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2) \tag{6}$$

The Euler equations, both in 1D and 2D, form a hyperbolic system of partial differential equations. The flux Jacobian matrix yields a complete set of eigenvalues (three for 1D and four for 2D) and eigenvectors, representing the waves and their directions in the solution. The eigenvalues are a function of the fluid velocity and the speed of sound in the medium. From a numerical standpoint, this enables the use of characteristic based algorithm where the flux function is reconstructed in a biased way which models the direction of information flow. The details of this are presented in subsequent sections.

3.2 Navier Stokes Equations

The Navier Stokes equations are an extension of the Euler equations to include the dissipative terms. While the conservation of mass remains unchanged, the equations for the momentum and energy conservation are modified to account for the viscous forces and heat conduction. The 2D Navier Stokes equations are

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} + \frac{\mathbf{g}(\mathbf{u})}{\partial y} = \frac{\mathbf{f_v}(\mathbf{u})}{\partial x} + \frac{\mathbf{g_v}(\mathbf{u})}{\partial y} \tag{7}$$

The terms on the left hand side have the same definitions as in the case of the 2D Euler equations. The terms on the right hand side account for viscosity and heat conduction and are defined as

$$\mathbf{f_v}(\mathbf{u}) = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ k\partial_x T + u\tau_{xx} + v\tau_{xy} \end{bmatrix}, \ \mathbf{g_v}(\mathbf{u}) = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ k\partial_y T + u\tau_{yx} + v\tau_{yy} \end{bmatrix} \tag{8}$$

where

$$\begin{aligned} \tau_{xx} &= (\lambda + 2\mu)\frac{\partial u}{\partial x} + \lambda\frac{\partial v}{\partial y} \\ \tau_{xy} &= \tau_{yx} = \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) \\ \tau_{yy} &= \lambda\frac{\partial u}{\partial x} + (\lambda + 2\mu)\frac{\partial v}{\partial y} \end{aligned} \tag{9}$$

The equation of state, eqn. (6), relates the energy to the pressure and flow velocity while the ideal gas law relates the temperature to the pressure and the density.

$$P = \rho R T \tag{10}$$

The fluid properties in the above equations are μ (fluid viscosity coefficient), $\lambda = -2\mu/3$ (bulk viscosity coefficient), k (thermal conductivity) and R (universal gas constant). The two relevant similarity parameters for viscous flows are the Mach number and the Reynold's number.

$$M = \sqrt{\frac{(u^2 + v^2)}{\gamma R T}}; \quad Re = \frac{\rho u c}{\mu} \tag{11}$$

where c is a characteristic length of the flow.

4 Essentially Non-Oscillatory Schemes

The Essentially Non-Oscillatory (ENO) class of schemes for spatial reconstruction has proved to be highly successful in finding the numerical solution of the Euler equations [5]. As mentioned in the previous section, the Euler equations constitute a set of hyperbolic partial differential equations whose solutions consist of smooth regions as well as discontinuities (shocks and contact discontinuities). Along a given direction, the system consists of three characteristic fields, two of which are acoustic waves and the third is an entropy wave [3]. For the numerical solution, the spatial reconstruction of the flux at the interfaces from the flow data given at discrete points is an important step. The ENO schemes [5] provide a tool for high order interpolation in smooth regions while avoiding oscillations near discontinuities. This is achieved by using piecewise continuous polynomials and adaptive stenciling for the interpolation process. For a given order, the smoothest stencil (based on the divided differences) is chosen.

The ENO technique is essentially applicable to a scalar function and can be extended to a system of equations in two ways. One is to apply the scalar interpolation to each component of the system, as expressed in primitive or conservative form. Another way to extend this concept is by decomposing the conservation laws into its constituent characteristic fields and applying the ENO procedure along each characteristic. The latter method has the advantage of following the physics of the problem, i.e, the wave nature of the solution.

The implementation of the ENO schemes can be described by the following iterative algorithm, based on polynomial interpolation in the Newton form. For a set of data given at discrete points $x_i, f_{i i=1}^{n}$, the $(N+1)$th order Newton polynomial is given by

$$p_{N+1}(x) = f[x_0] + \sum_{i=1}^{N} f[x_0, ..., x_i] \Pi_{j=0}^{i-1}(x - x_j) \tag{12}$$

where x_0 is the starting point (arbitrarily chosen) and $x_i,\ i = 1, ..., N$ are successively chosen points in the stencil (without any assumption on their order). Square brackets denote divided differences and $f[x_0] = f(x_0)$. Based on this, the ENO implementation for a left-biased reconstruction can be expressed as follows:

Initialize $f_{i+1/2} = f(x_i)$ (Starting value)
Initialize $X = (x_{i+1/2} - x_i)$
$low = i,\ up = i$ (lower and upper boundaries of stencil)
Till desired order is reached
    $c1 = f[x_{low-1}, x_{up}],\ c2 = f[x_{low}, x_{up+1}]$ (Creating two candidate stencils by adding points)
    if $(|c1| < |c2|)$ (Choosing the left stencil)
        $f_{i+1/2} = f_{i+1/2} + c1 * X$
        $low = low - 1$
        $X = X * (x_{i+1/2} - x_{low})$
    else, (choosing the right stencil)
        $f_{i+1/2} = f_{i+1/2} + c2 * X$
        $up = up + 1$
        $X = X * (x_{i+1/2} - x_{up})$
End Loop

For an uniform grid, the ENO schemes can be simplified to yield relatively simple reconstruction formulae for the interface flux. Choice of the stencil can be made using undivided differences and the coefficients of each point in the stencil can be determined, thus simplifying the implementation of the ENO procedure. The coefficients for different orders as well as the simplified implementation has been described in [5]. While these simplifications have been derived for an uniform grid, they have often been used to implement the ENO/WENO scheme for non-uniform meshes for algorithms reported in literature. The assumption is that for a fine enough mesh, the non-uniformity over the stencil will be very small and thus the resultant error will be negligible. However, using these simplified schemes results in a loss of the order of interpolation. While the errors may be negligible for meshes with slight non-uniformity, they have the potential to spoil the solution for highly twisted or deformed meshes. This is especially pronounced for higher orders where the stencil size is large. Along with a loss of accuracy, the reconstructed values may be incorrect if the geometry of the mesh is not taken into account. In the present study, the ENO schemes in their original form (as expressed in the iterative procedure above) are used for the computations involving non-uniform meshes.

5 Numerical Scheme

5.1 1D Euler Equations

The semi-discrete form of eq. (1), using finite volume formulation is

$$\frac{d\mathbf{u}_i}{dt} = \mathbf{Res}(\mathbf{u}_i);\ \mathbf{Res}(\mathbf{u}_i) = -\frac{1}{\Delta x}(\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2}) \tag{13}$$

where i is the cell index and δx is the cell width. $\mathbf{F}_{i-1/2}$ and $\mathbf{F}_{i+1/2}$ are the numerical fluxes evaluated at the left and right interfaces of the ith cell. A characteristic-based scheme is used where the flux is reconstructed by decoupling along the characteristic directions. The eigenvalues, the left and right eigenvectors at the interface ($\lambda^k_{i+1/2}$, $\mathbf{L}^k_{i+1/2}$ and $\mathbf{R}^k_{i+1/2}$ respectively for $k = 1, ..., m$ where m is the number of characteristic directions of the system) used for decoupling, upwinding and re-coupling the fluxes are evaluated at an arithmetically averaged state. The flux is evaluated as:

$$\mathbf{F}_{i+1/2} = \sum_{k=1}^{m} f^k_{i+1/2}\mathbf{R}^k_{i+1/2} \tag{14}$$

where $f^k_{i+1/2}$ is the component of the flux vector along the kth characteristic direction, evaluated numerically. For a scheme using a stencil S, characteristic flux at the interface is a function of those evaluated at cell centers lying in the stencil,

$$f^k_{i+1/2} = Rec(f^k_j;\ j \in S) \tag{15}$$

where Rec is the reconstruction procedure, dependent on the scheme used. In the present study, the Roe-Fixed (RF) formulation [5] is used to evaluate the characteristic flux in an upwinded fashion. The RF formulation is given as

$$\begin{aligned} f^k_{i+1/2} &= f^k_L,\ \text{if } \lambda^k_i, \lambda^k_{i+1/2}, \lambda^k_{i+1} > 0 \\ &= f^k_R,\ \text{if } \lambda^k_i, \lambda^k_{i+1/2}, \lambda^k_{i+1} < 0 \\ &= \frac{1}{2}[f^k_L + f^k_R + \alpha_{i+1/2}(u^k_R - u^k_L)],\ \text{otherwise} \end{aligned} \tag{16}$$

where $\alpha_{i+1/2} = max(|\lambda_i^k|, |\lambda_{i+1/2}^k|, |\lambda_{i+1}^k|)$. The RF formulation uses the LLF flux formulation [5] as an entropy fix to the Roe's scheme by introducing extra dissipation and thus breaking up non-physical expansive shocks. Using the RF formulation is also computationally cheaper than the LLF flux formulation since reconstruction of the state vector is required only in cases where entropy fix is required. The interpolated values of the decoupled fluxes $f_{L,R}^k$ at the interface are found (from the cell-centered values) using the ENO reconstruction technique described in the previous section. The procedure outlined in the previous section is for the left-biased term f_L^k and the corresponding procedure for the right-biased term f_R^k can be easily derived. The semi-discrete equation, eq. (13), is advanced in time using the Runge-Kutta (RK) family of schemes like in the case of the scalar hyperbolic equation. The 1st order (Forward Euler), 2nd and 3rd order accurate Total Variation Diminishing (TVD) RK and 4th order RK schemes [5] have been used in the present study in conjunction with the high order ENO spatial discretization.

5.2 2D Euler Equations

For the 2D Euler equations, the semi-discrete form of eq. (4) using the finite volume formulation is given as:

$$\frac{d\mathbf{u_{ij}}}{dt}V_{ij} + \sum_{faces} \mathbf{F}.\hat{\mathbf{n}}dS = 0 \Rightarrow \frac{d\mathbf{u_{ij}}}{dt} = \mathbf{Res}(i,j) \tag{17}$$

Here, V_{ij} is the area of the cell. The residual is given by (for a quadrilateral cell)

$$\mathbf{Res}(i,j) = \frac{-1}{V_{ij}}[\sum_{l=1}^{4} \mathbf{F}.\hat{\mathbf{n}}_l dS_l \tag{18}$$

dS_l is the length of the cell interfaces. The semi-discrete equation, as given by eq. (17) is marched in time using the multi-stage Runge-Kutta (RK) algorithm (similar to the 1D case). It is to be noted that $\mathbf{F}.\hat{\mathbf{n}} = n_x\mathbf{f} + n_y\mathbf{g}$ is a vector representing the normal flux at a given interface. Thus it can be reconstructed in the same way as described for the 1D Euler equations, using characteristic decoupling based on the eigenstructure evaluated at the cell interface, normal to it.

5.3 2D Navier Stokes Equations

In typical computations for flow around airfoils, a highly stretched mesh is used with very small cells near the surface to capture the boundary layer. The time step size in explicit time integration is bounded by the CFL stability condition [3, 4, 5] which is very restrictive for the meshes typically used for airfoil computations. This leads to very slow convergence to the steady state solution. Implicit time integration, on the other hand, has unconditional stability and the time step size can be decided based on accuracy rather than stability. While implicit time stepping is computationally more intensive per time step, the larger time step sizes allows for faster convergence rates. In the present study, the Euler solver described previously is extended to the Navier Stokes equations by using implicit time integration. While the finite volume formulation of the governing equations in the Cartesian coordinates can handle curvilinear meshes, parts of the implicit time algorithm as well as the viscous terms are treated using finite differences and thus, the governing equations in curvilinear coordinates are used.

The Navier Stokes equations in curvilinear coordinates are given as

$$\frac{\partial \hat{\mathbf{u}}}{\partial t} + \frac{\partial \hat{\mathbf{f}}(\hat{\mathbf{u}})}{\partial \xi} + \frac{\hat{\mathbf{g}}(\hat{\mathbf{u}})}{\partial \eta} = \frac{\hat{\mathbf{f}}_{\mathbf{v}}(\hat{\mathbf{u}})}{\partial \xi} + \frac{\hat{\mathbf{g}}_{\mathbf{v}}(\hat{\mathbf{u}})}{\partial \eta} \tag{19}$$

where

$$\hat{\mathbf{u}} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix},\ \hat{\mathbf{f}}(\hat{\mathbf{u}}) = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x P \\ \rho v U + \xi_y P \\ (E+P)U - \xi_t P \end{bmatrix},\ \hat{\mathbf{g}}(\hat{\mathbf{u}}) = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x P \\ \rho v V + \eta_y P \\ (E+P)V - \eta_t P \end{bmatrix} \tag{20}$$

with

$$U = \xi_t + \xi_x u + \xi_y v, \quad V = \eta_t + \eta_x u + \eta_y v \tag{21}$$

The viscous terms are

$$\hat{\mathbf{f}_{\mathbf{v}}}(\hat{\mathbf{u}}) = J^{-1}(\xi_x \mathbf{f}_{\mathbf{v}} + \xi_y \mathbf{g}_{\mathbf{v}})\hat{\mathbf{g}_{\mathbf{v}}}(\hat{\mathbf{u}}) = J^{-1}(\eta_x \mathbf{f}_{\mathbf{v}} + \eta_y \mathbf{g}_{\mathbf{v}}) \tag{22}$$

The terms on the left hand side have the same definitions as in the case of the 2D Euler equations. The terms on the right hand side account for viscosity and heat conduction and are defined as

$$\mathbf{f}_{\mathbf{v}}(\mathbf{u}) = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ k\partial_x T + u\tau_{xx} + v\tau_{xy} \end{bmatrix},\ \mathbf{g}_{\mathbf{v}}(\mathbf{u}) = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ k\partial_y T + u\tau_{yx} + v\tau_{yy} \end{bmatrix} \tag{23}$$

where

$$\begin{aligned} \tau_{xx} &= (\lambda + 2\mu)\frac{\partial u}{\partial x} + \lambda \frac{\partial v}{\partial y} \\ \tau_{xy} &= \tau_{yx} = \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) \\ \tau_{yy} &= \lambda \frac{\partial u}{\partial x} + (\lambda + 2\mu)\frac{\partial v}{\partial y} \end{aligned} \tag{24}$$

The stress and heat conduction terms are transformed into the curvilinear coordinates as

$$\begin{aligned} \tau_{xx} &= \mu(4(\xi_x u_\xi + \eta_x u_\eta) - 2(\xi_y v_\xi + \eta_y v_\eta))/3 \\ \tau_{xy} &= \tau_{yx} = \mu(\xi_y u_\xi + \eta_y u_\eta + \xi_x v_\xi + \eta_x v_\eta) \\ \tau_{yy} &= \mu(-2(\xi_x u_\xi + \eta_x u_e ta) + 4(\xi_y v_\xi + \eta_y v_\eta))/3 \\ \partial_x T &= \xi_x T_\xi + \eta_x T_\eta \\ \partial_y T &= \xi_y T_\xi + \eta_y T_\eta \end{aligned} \tag{25}$$

A family of implicit schemes were proposed in [6] for the inviscid Euler equations and extended to the Navier Stokes equations in [7] for Cartesian meshes. Their implementation on a curvilinear mesh is described in [8] with several examples of airfoil computations. While these schemes used finite differences with second order central stencils, an upwinded scheme with mixed finite-difference and finite volume discretization is implemented in the present study. The implicit integration for a differential equation in time is

$$u^{n+1} = u^n + \delta t[\frac{\partial u}{\partial t}^{n+1}] \tag{26}$$

Considering the inviscid form of eq. (19) (with zero right hand side), the implicit time integration can be expressed as (dropping the hats for convenience)

$$u^{n+1} = u^n - \delta t[(\frac{\partial f}{\partial \xi} + \frac{\partial g}{\partial \eta})^{n+1}] \tag{27}$$

Since the fluxes f, g are non-linear, their evaluation at the next time level requires their linearization with respect to time,

$$\begin{aligned} f^{n+1} &= f^n + A^n(\Delta u) + O(\delta t^2) \\ g^{n+1} &= g^n + B^n(\Delta u) + O(\delta t^2) \end{aligned} \tag{28}$$

where $\Delta u = u^{n+1} - u^n$. $A = \partial f/\partial u$ and $B = \partial g/\partial u$ are the Jacobian matrices. Using this linearization, the implicit algorithm for the inviscid equation is

$$[I + \delta t(\partial_\xi A^n + \partial_\eta B^n)]\Delta u = -\delta t(\partial_\xi f + \partial_\eta g)^n \tag{29}$$

Note that the right hand side of the above equation is equivalent to that for an explicit time integration and thus, can be evaluated in exactly the same manner as described in the section for 2D Euler equations. In the computation of the steady state solutions, the accuracy of the right hand side is important while the factor of Δu on the left hand side is to provide stability. Therefore, a simple extension to the viscous equation is

$$[I + \delta t(\partial_\xi A^n + \partial_\eta B^n)]\Delta u = -\delta t(\partial_\xi f - \partial_\xi f_v + \partial_\eta g - \partial_\eta g_v)^n \tag{30}$$

A more accurate scheme for the viscous equation is

$$[I + \delta t(\partial_\xi A^n - \partial_\xi^2 R^n + \partial_\eta B^n - \partial_\eta^2 S^n)]\Delta u = -\delta t(\partial_\xi f - \partial_\xi f_v + \partial_\eta g - \partial_\eta g_v)^n \tag{31}$$

where R and S are as defined in [7]. Either of eqns. (30) and (31) can be used for steady state computations while the latter is expected to yield better time-accurate results. For a 2D structured mesh, the above two equations result in a block pentadiagonal system of size $O(n^2 \times n^2) = O(n^4)$ where n is the number of cells along one dimension. To reduce the order of the system, the Alternating-Direction-Implicit (ADI) factorization is used, which results in solutions of decoupled block tridiagonal linear systems of $O(n^2)$ size along each direction. The ADI factored scheme is

$$[I + \delta t \partial_\xi A^n][I + \delta t \partial_\eta B^n]\Delta u = -\delta t(\partial_\xi f - \partial_\xi f_v + \partial_\eta g - \partial_\eta g_v)^n \tag{32}$$

which is solved as

$$\begin{aligned} (I + \delta t \partial_\xi A^n)\Delta u^* &= -\delta t(\partial_\xi f - \partial_\xi f_v + \partial_\eta g - \partial_\eta g_v)^n \\ (I + \delta t \partial_\eta B^n)\Delta u &= \Delta u^* \end{aligned} \tag{33}$$

This approximate factorization introduces an error of $O(\delta t^2)$ into the scheme which is of the same order as the error introduced by the linearization in time. Finally, the solution is updated as $u^{n+1} = u^n + \Delta u$.

The right hand side of the first step of the ADI scheme consists of convective as well as dissipative fluxes. The convective fluxes are computed as described in the section for 2D Euler equations, using the finite volume formulation and characteristic based upwinding.

The dissipative flux terms f_v and g_v are computed using 2nd order central finite differences to evaluate the stress and heat conduction terms. These fluxes are computed at the cell interfaces and their contributions are added to the convective fluxes to get the total fluxes at the interfaces. The left hand side involves the computation of the $\partial_\xi(A\Delta u)$ and $\partial_\eta(B\Delta u)$ which are computed using upwinded first order finite differencing.

$$\begin{aligned}
\partial_\xi(A\Delta u)_{i,j} &= A^+_{i-1/2,j}\frac{(\Delta u_{i,j}-\Delta u_{i-1,j})}{\Delta\xi} + A^-_{i+1/2,j}\frac{(\Delta u_{i+1,j}-\Delta u_{i,j})}{\Delta\xi} \\
\partial_\xi(B\Delta u)_{i,j} &= B^+_{i,j-1/2}\frac{(\Delta u_{i,j}-\Delta u_{i,j-1})}{\Delta\eta} + B^-_{i,j+1/2}\frac{(\Delta u_{i,j+1}-\Delta u_{i,j})}{\Delta\eta}
\end{aligned} \tag{34}$$

where

$$\begin{aligned}
A^\pm &= R_\xi \Lambda^\pm_\xi R^{-1}_\xi \\
B^\pm &= R_\eta \Lambda^\pm_\eta R^{-1}_\eta
\end{aligned} \tag{35}$$

where $\Lambda^{+,-}$ denote diagonal matrices with positive and negative eigenvalues respectively. Finally, the block tridiagonal system is solved using Gauss elimination but there is scope for making the algorithm faster by using block tridiagonal solvers and/or iterative Gauss Siedel solvers. This completes the description of the numerical scheme.

6 Boundary Conditions

The numerical treatment of the boundary is very important to ensure the correct conditions are satisfied while computing the flow at boundary cells. In the present study, the boundary conditions have been imposed by using "ghost cells" which is a very simple yet elegant way of applying the required the boundary conditions. Depending on the order of the scheme, a certain number of neighboring cells are required in the application of the numerical scheme at a given cell. While these neighboring cells are present for cells in the interior of the domain, the cells at the boundary need to be treated specially, either by modifying the numerical scheme itself at boundary cells or by imagining the existence of ghost cells which lie outside the formal domain.

The ghost cells approach requires the definition of the flow variables at these imaginary points outside the domain. This is done such that the physical boundary conditions are satisfied. The following are the different boundary conditions encountered in the present study and their implementation using ghost cells:

- Freestream boundary: Freestream (i.e. specified) conditions are imposed on the boundary. The flow variables in the ghost cells take the specified value and the numerical scheme ensures that the correct information travel in to or out of the domain.

- Outgoing boundary: This is used for supersonic outflow. Since such a flow involves no information moving in to the domain, it suffices to set the ghost cell flow variables with the same values as the last cell in the interior. This ensures a non-reflective boundary condition with information flowing out. However, for the present numerical scheme, this is not really necessary as a special case. If freestream conditions are set in the ghost cell such that it represents a supersonic outflow, the characteristic decomposition and flux computation will model the one-way flow of information.

- Periodic boundary: This is often required to simulate periodic flows over an infinite domain. The ghost cells at one boundary are given the same values as the interior cells of the opposite boundary, this simulating a continuity in the domain.

- Wall boundary: At a solid wall, for inviscid flow, the flow satisfies the tangency condition, i.e. the direction of the velocity vector has to be parallel to the surface. This is imposed by decomposing the velocity into its normal and tangential components at the first interior cell next to the solid wall. The conditions in the ghost cell on the other side of the wall are set by reflecting the normal velocity while keeping every other variable the same. This ensures zero normal velocity at the wall itself. For viscous flow, the no-slip condition requires that the flow adjacent to the wall be at the same velocity as the wall itself. Typically, the wall is static and thus, the flow velocity in the ghost cells are set as the negative of that inside the domain to ensure zero velocity at the interface.

These boundary conditions have been used in conjunction with the numerical technique to solve the problems in subsequent sections.

7 Validation

The numerical scheme described above is validated as described in this section. Initially, the 1D Euler equations are solved to test the performance of the various orders of the algorithm. Two standard Riemann problems [3, 4] are solved which consist of an initial discontinuity which evolves into a shock wave, a rarefaction wave and a contact discontinuity. Both these problems are benchmark problems used in the validation of 1D Euler algorithms in literature. The computed results are compared with the exact solution. The algorithm is then used to solve 2D Cartesian problems. An example of a 2D Riemann problem [9, 10] is chosen which consists of the evolution of a discontinuous initial solution over a square domain. The algorithm is also used to solve the Mach 2.9 oblique shock reflection problem [11], which is another benchmark case. To test the performance of the Navier Stokes solver on a Cartesian mesh, the Couette flow problem is solved. It involved incompressible flow between two infinite flat plates moving at a given velocity relative to each other [7]. Following these, the algorithm is tested for flow through a compression ramp [11], which involves a non-Cartesian grid. Finally, the inviscid flow around the NACA0012 airfoil is considered. Two cases, subsonic and transonic flow, are considered and the flow solved for. As a part of post-processing, the coefficient of pressure on the airfoil surface is computed and compared to results in literature [12, 13]. The performance of the code is compared with the TURNS code [14] which is a validated and published code developed at the Alfred Gessow Rotorcraft Center, University of Maryland. The TURNS code uses an implicit time stepping scheme along with a MUSCL-type [3] approach for spatial reconstruction. The Navier Stokes solver is used to compute the viscous flow around an RAE2822 airfoil in the transonic regime and the computed pressure coefficients compared with results in literature [15, 16].

1D Euler validation

The 1D Riemann problems [3] consist of a domain $[0, 1]$ with a discontinuity located at the center $x = 0.5$. The solution involves the evolution of this initial discontinuity into a left-

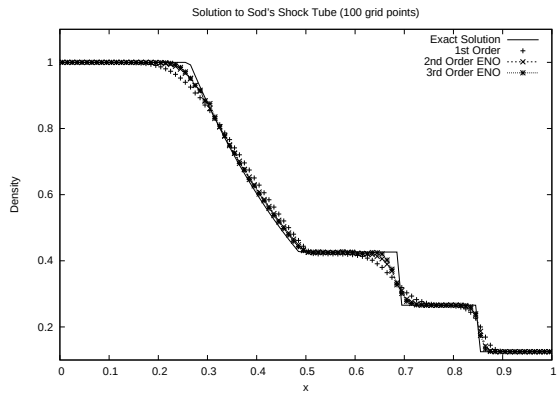

Figure 1: Solution of Sod's shock tube

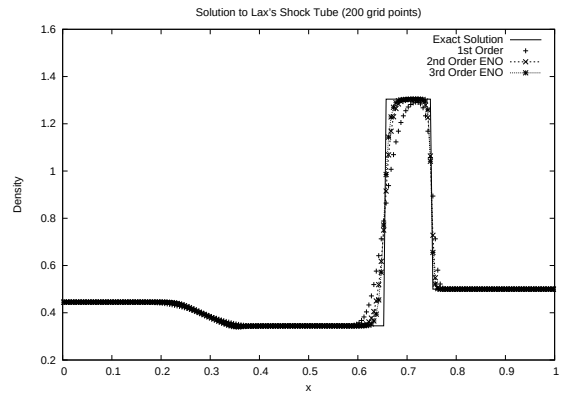

Figure 2: Solution of Lax's shock tube

running rarefaction wave and a right-running contact discontinuity and a shock wave. Two such cases are considered. For the first case, referred to as Sod's shock tube problem, the initial conditions are given as follows:

$$\begin{aligned} \rho_L &= 1 \quad ; \quad \rho_R = 0.125 \\ u_L &= 0 \quad ; \quad u_R = 0 \\ P_L &= 1 \quad ; \quad P_R = 0.1 \end{aligned} \tag{36}$$

where the subscripts L and R denote the left and right side of the initial discontinuity. The second case, referred to as Lax's shock tube problem, has the following initial conditions:

$$\begin{aligned} \rho_L &= 0.445 \quad ; \quad \rho_R = 0.5 \\ u_L &= 0.698 \quad ; \quad u_R = 0 \\ P_L &= 3.528 \quad ; \quad P_R = 0.571 \end{aligned} \tag{37}$$

In both cases, the ratio of specific heats γ is taken as 1.4

Figure (1) shows the results for the first test case, solved on a 100-point grid. The density is plotted and the main features of the solution (a left-running rarefaction, a right running shock wave and a right running contact discontinuity) can be easily distinguished. The computed solutions for various orders are compared with the exact solution. The solutions are computed using 1st order in space with explicit Euler in time, 2nd order ENO in space with 2nd order TVD RK in time and 3rd order ENO in space with 3rd order TVD RK in time. It can be seen that the 2nd and 3rd order schemes show much better resolution across discontinuities than the 1st order scheme as expected. However, across the discontinuities, the higher order ENO schemes do not suffer from the problem of spurious oscillations like naive higher order schemes. Figure (2) shows the density variation for the second test case, solved on a 200-point grid. Similar conclusions can be drawn.

2D Cartesian validation

A class of 2D Riemann problems have been presented in [9]. They consist of a square domain and initial conditions as constant states in the four quadrants of the domain. The solution involves evolving them till a given time. In the present study, the 6th case is chosen, whose

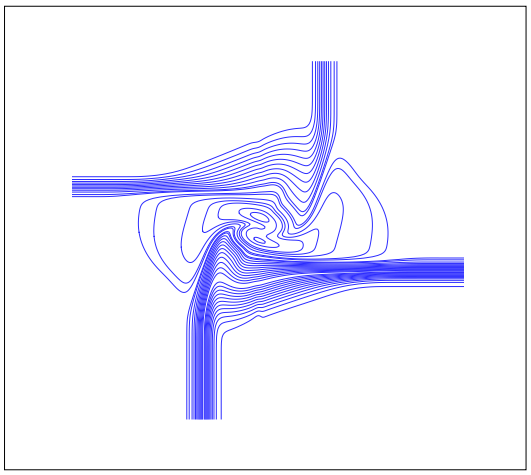

Figure 3: Density - 1st order

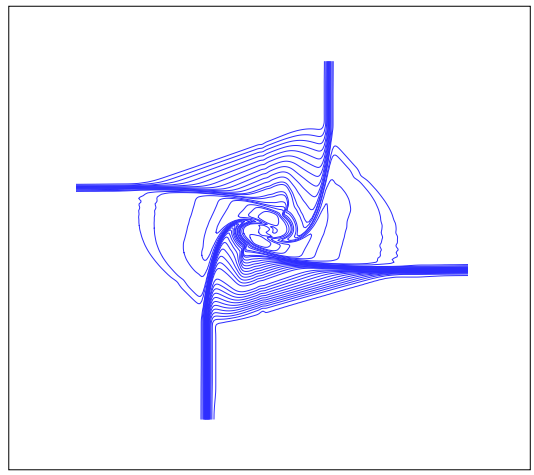

Figure 4: Density - 3rd order

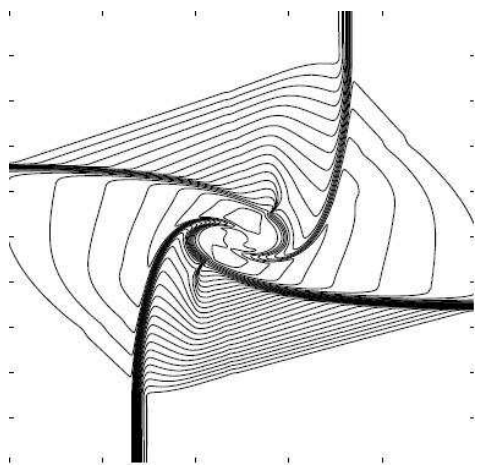

Figure 5: Density - from [9]

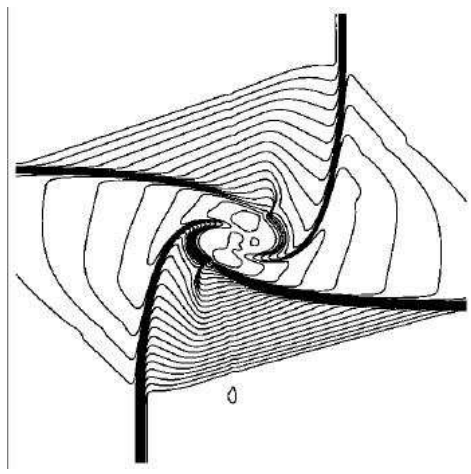

Figure 6: Density - from [10]

initial conditions are given as follows:

$$
\begin{aligned}
\rho_{NW} &= 2.0 \quad ; \quad \rho_{NE} = 1.0 \\
u_{NW} &= 0.75 \quad ; \quad u_{NE} = 0.75 \\
v_{NW} &= 0.5 \quad ; \quad v_{NE} = -0.5 \\
P_{NW} &= 1.0 \quad ; \quad P_{NE} = 1.0 \\
\\
\rho_{SW} &= 1.0 \quad ; \quad \rho_{SE} = 3.0 \\
u_{SW} &= -0.75 \quad ; \quad u_{SE} = -0.75 \\
v_{SW} &= 0.5 \quad ; \quad v_{SE} = -0.5 \\
P_{SW} &= 1.0 \quad ; \quad P_{SE} = 1.0
\end{aligned}
\tag{38}
$$

where the subscripts NW, NE, SW and SE refer to the north-west, north-east, southwest and southeast quadrants respectively. The ratio of specific heats γ is taken as 1.4, as usual. The four boundaries are outgoing and the solution is evolved to time $t = 0.3$.

Figures (3) and (4) show the density contours for the 1st and 3rd order computed results. It can be seen using the same contour levels that the first order scheme is quite diffuse and

the shocks are smeared over a large area. On the other hand, the 3rd order computations capture the shocks much better as well as the flow features around the center of the domain. As a comparison, figures (5) and (6) show the results presented in [9] and [10]. To make the comparison meaningful, the contours plotted in all these figures (3) and (4) use the same minimum, maximum and delta as used in the references. All the computations were carried out on a 400×400 grid and the domain was a unit square.

Oblique Shock Reflection

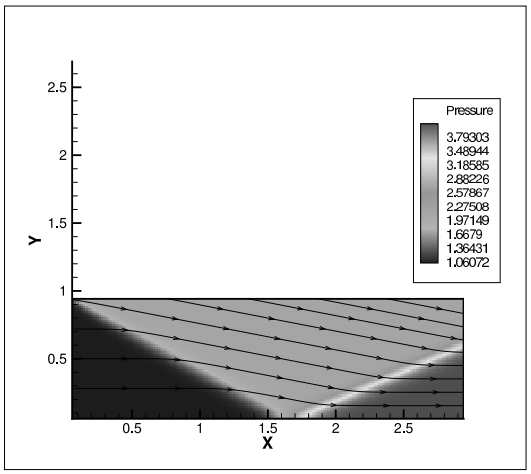

Figure 7: Oblique Shock Reflection (Pressure) - 2nd order

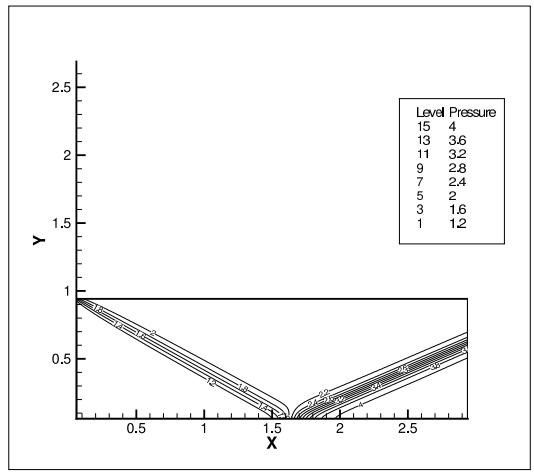

Figure 8: Oblique Shock Reflection (Pressure) - 3rd order

This problem involves the reflection of an oblique shock at 30° on a solid surface. The domain is rectangular with the length three times the height. The boundary conditions involve a supersonic inflow at Mach 2.9 on the left boundary. Exact post-shock conditions (computed using oblique shock relations [2]) are imposed on the top boundary, thus simulating an oblique shock entering the domain from the north-west corner. The right boundary is set to supersonic outflow and the bottom boundary is set as a solid wall. The domain is initialized to a physically relevant state (in the present study, it was initialized to $\rho, u, v, P = 1, 0, 0, 1$) and the solution is marched in time till it reaches steady state. For first order computations, the steady state is attained when the residual norms for time marching fall to machine zero. For higher order computations using ENO schemes, the residuals do not fall to machine zero due to shock oscillations over a cell but this has negligible effect on the solution. Figure (7) shows the pressure variation and streamlines over the domain for 2nd order computations while figure (8) show the pressure contours for 3rd order computations. It should be noted that since the solid wall is flat, the shocks become normal to it near the surface to maintain zero pressure gradient in the normal (to the wall) direction, as is physically required.

Couette Flow

The Couette flow is one of the simplest illustrations of viscous flow with the no-slip condition at the solid wall. The problem consists of two infinite flat plates moving at a constant velocity relative to each other with an incompressible viscous fluid in between. The steady state solution is a linear velocity profile between the two walls with the velocity of the fluid equal

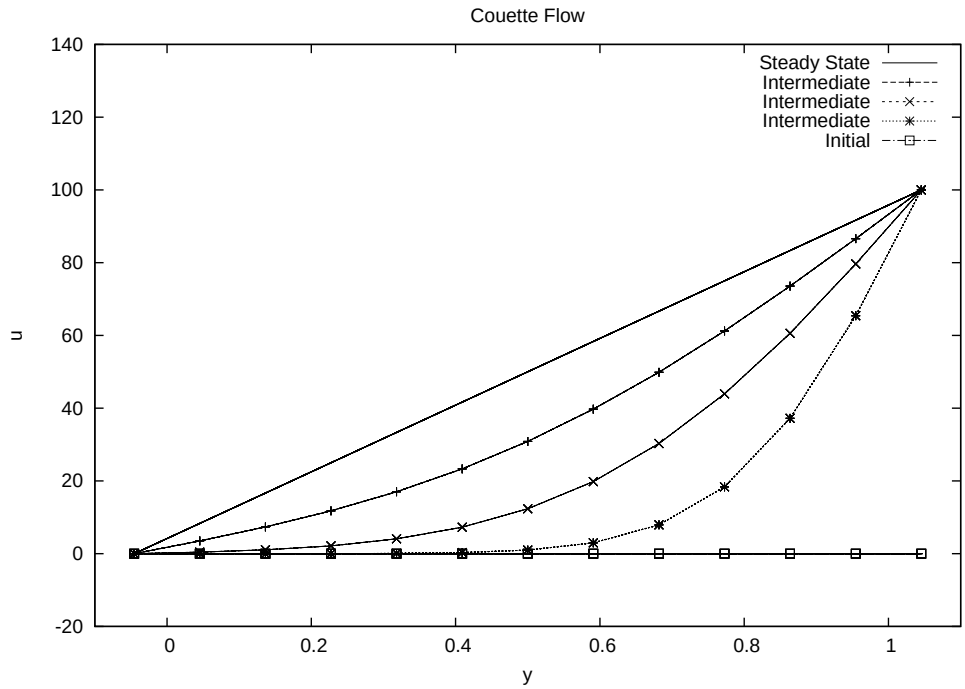

Figure 9: Initial, intermediate and final velocity profiles for the Couette flow problem

to the velocity of the walls at the boundaries. In the present study, the domain consisted of a 6×13 grid with six cells in x-direction and thirteen cells in the y-direction (excluding the ghost cells). Periodic boundary conditions were set along the x-direction to simulate an infinite domain in this dimension. No-slip boundary conditions were set along the top and bottom boundaries which are the solid plates. The relative velocity of the upper plate with respect to the lower one was taken as 100 m/s. The initial conditions were set as atmospheric density and pressure with velocity as zero all through the domain. The solution was marched through till steady state till the linear profile was obtained.

Figure (9) shows the initial, intermediate and final velocity profiles for the solution. The algorithm was seen to converge pretty rapidly for moderate CFL numbers. Convergence degenerated at larger time steps due to errors caused by linearization and ADI factorization. However, it was verified that the implicit algorithm allowed much larger time steps than was possible by using explicit time stepping.

Flow through a Compression Ramp

The flow through a channel with a 15° compression ramp is studied. The domain involves a rectangular channel with the length three times the height. The bottom surface of the channel as a 15° ramp starting at one-sixth the length of the channel and continuing till one-third the length of the channel and then flattening out into a flat surface. The flow comes in at supersonic speeds from the left boundary and forms an oblique shock when it encounters the ramp, thus going through a compression. At the end of the ramp, the flow at the surface turns outwards, this causing an expansion fan. The flow then exits the channel at the right boundary at a supersonic speed. The angle of the initial oblique shock depends on the speed of the incoming flow and therefore, it reflects from some location on the upper

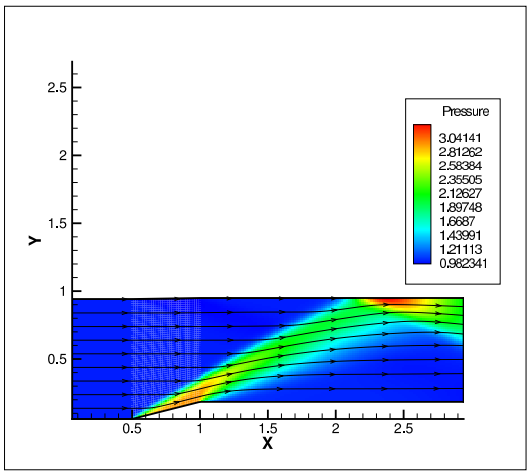

Figure 10: Compression Ramp (Pressure) - 2nd order

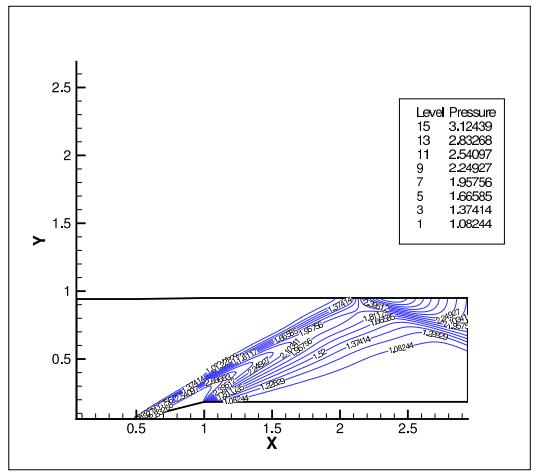

Figure 11: Compression Ramp (Pressure) - 3rd order

wall of the channel depending on the speed of the incoming flow. Once again, the reflected shock may or may not reflect from the bottom surface, depending on the incoming flow forming a reflecting shock system. In the present study, a Mach 3.3 inflow and the computed results can be validated with the exact solutions obtained through oblique shock relations and Prandtl-Meyer expansion fan relations for compressible flow [2]. In the present case, a Mach 3.3 inflow causes a 30.2^o oblique shock to form at the beginning of the ramp, which reflects from the upper wall before exiting the domain. An expansion wave forms at the top of the ramp. The exact solution across these shock and expansion waves are calculated and compared with the computed results.

Figure (10) show the pressure variation and the streamlines of the flow while figure (11) show the pressure contours. The oblique shock and the expansion fan can be clearly distinguished in these two figures.

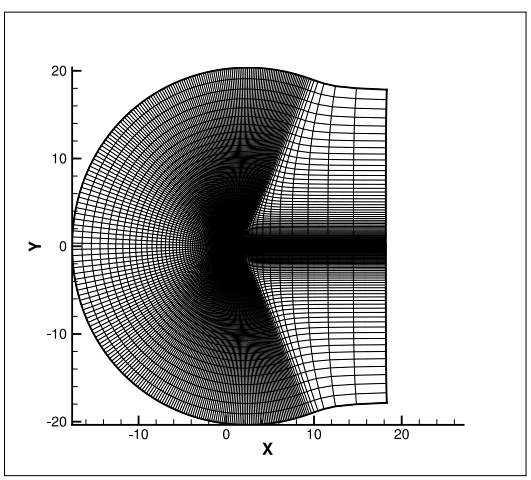

Figure 12: The computational domain for airfoil computations

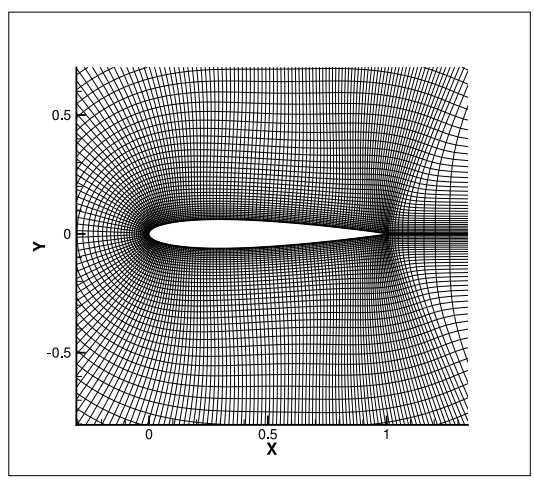

Figure 13: Magnified view of the grid near the airfoil

Flow around Airfoils

Following the validation of the algorithm over simple problems in 2D, it is used to solve the flow over airfoils. The domain is discretized using a body-fitted curvilinear mesh. The mesh topology is C-type and figure (12) shows the whole domain. The airfoil is assumed to have a unit chord length and the far field boundary of the domain is taken at twenty chords away from the airfoil, which is sufficiently far away. Figure (13) shows a magnified view of the grid around the airfoil.

Since this is a C-type mesh, the domain can be mapped into a quadrilateral domain with four boundaries. The "top" boundary of this quadrilateral consists of the whole far field boundary except the outflow behind the airfoil (far right edge of the domain) (see figure (11)). The "left" boundary consists of the bottom half of the outflow behind the airfoil while the "right" boundary consists of the top half of the outflow. The "bottom" boundary consists of the airfoil surface itself and a cut through the domain from the airfoil trailing edge to the far right end of the domain. In accordance with this, freestream boundary conditions are imposed on the "top", "left" and "right" boundaries of the domain, since it is assumed that the far field boundary is far enough for the airfoil to have any effect on the flow there. For the "bottom" boundary, the part of it covering the airfoil surface is treated by curved wall boundary conditions while re-entrant conditions are imposed in the wake region to ensure continuity of the flow across the wake.

The flow around the airfoil is computed and one of main quantities used in comparing the flow is the coefficient of pressure. It is a non-dimensional pressure given by

$$C_p(x,y) = \frac{P(x,y) - P_\infty}{\frac{1}{2}\rho_\infty u_\infty^2} \tag{39}$$

where the subscript ∞ denotes freestream flow conditions. The term in the denominator is termed as the freestream "dynamic pressure".

Inviscid Flow

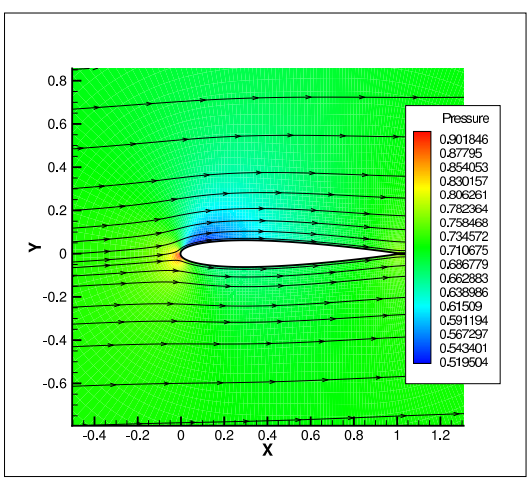

Figure 14: Pressure variation for the subsonic airfoil case

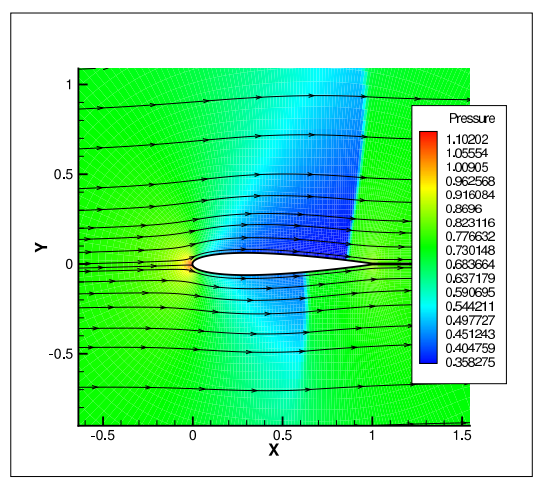

Figure 15: Pressure variation for the transonic airfoil case

Two cases are considered for the flow around the airfoil. The first case involves a fully subsonic flow around it. The freestream Mach number is 0.63 and the airfoil has a 2° angle of

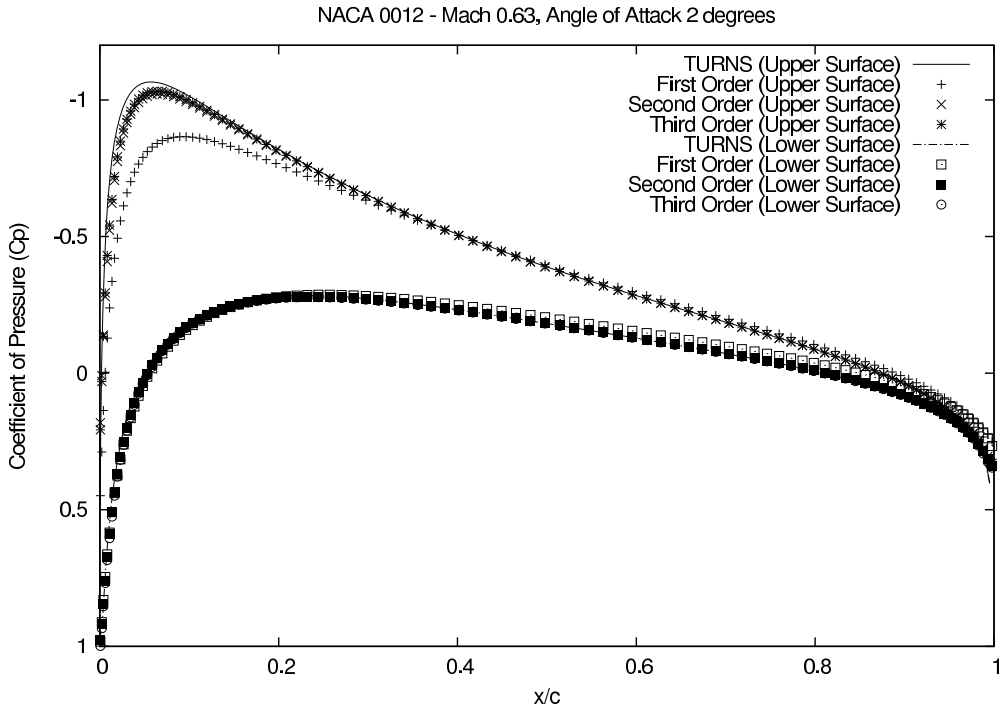

Figure 16: Surface coefficient of pressure variation for the subsonic airfoil case - Computed

attack. Figure (14) shows the pressure variation around the airfoil and the flow streamlines. The high pressure region at the leading edge of the airfoil is the stagnation point while the low pressure region on top of the airfoil, towards the leading edge is the suction peak which is the main lift generator. Figure (16) shows the coefficient of pressure computed using the algorithm and computed using the TURNS code for the upper and lower surfaces. It can be seen that the higher order schemes capture the suction peak much better than the 1st order scheme and thus, are likely to yield much more accurate values of the lifting force. At other parts of the flow, the various schemes agree very well with the results of the TURNS code. It can be seen that the computed results agree very well with the results in [12].

The second case involves the transonic flow around an airfoil. When the freestream Mach number is high enough (but subsonic), the flow can accelerate and reach supersonic speeds around the airfoil. Shocks on the upper and lower surfaces form to bring the flow back to its subsonic condition before leaving the airfoil. Such a situation is not desirable in flight since it increases the aerodynamic loading of the wings drastically and also increases the pressure drag by a large amount. In the present computations, the freestream Mach number is 0.85 while the angle of attack is 1°. Figure (15) shows the pressure variation around the airfoil. As in the subsonic case, a stagnation region is formed at the leading edge of the airfoil and the flow accelerates to supersonic speeds don both the upper and lower surface. However, the speeds attained on the upper surface are higher. A normal shock is formed at around 0.9 times the chord on the upper surface while the lower surface shock is positioned at around 0.6 times the chord. Downstream of the shock, the flow is subsonic. The blue regions in figure (15) show the pockets of supersonic flow. Figure (17) show the pressure coefficient variation for the computed results and those obtained by the TURNS code, for both the upper and lower surfaces. It is seen that the 1st order scheme is not only dissipative around the shock, it does not capture the upper surface shock position correctly either. In comparison, the 2nd

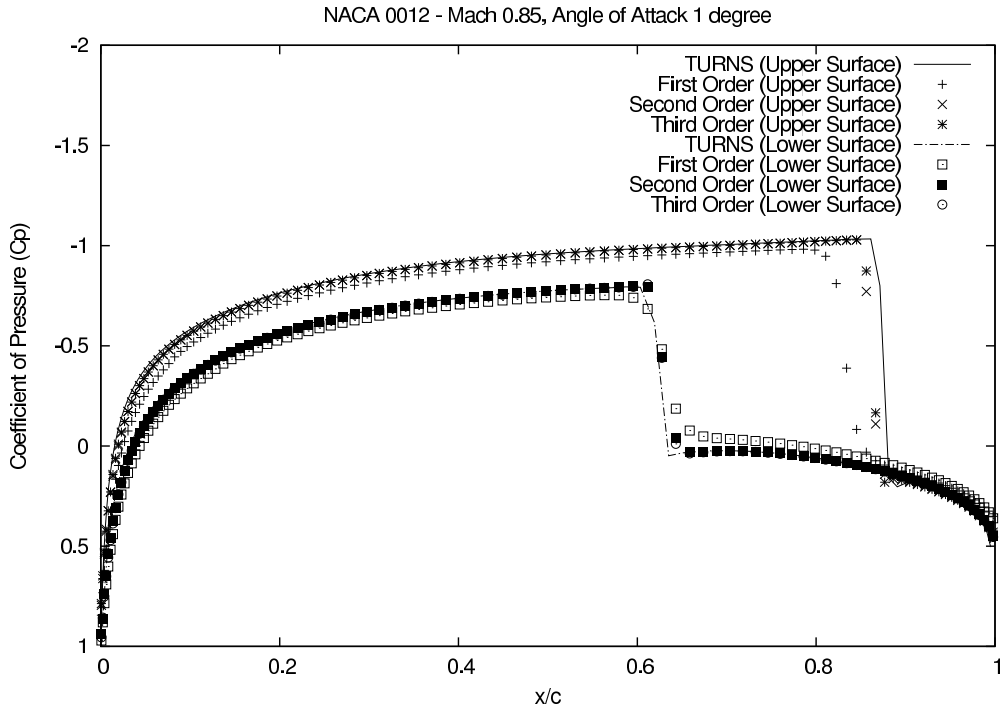

Figure 17: Surface coefficient of pressure variation for the transonic airfoil case - Computed

and 3rd order schemes show much sharper resolution of the shocks and captures the correct positions. At other parts of the flow, the solutions from all the schemes agree well with the TURNS results. Excellent agreement can be observed with results in [12]. Similar results are also presented in [13]

Viscous Flow around Airfoil

The Navier Stokes solver is used to solve for transonic flow around the RAE2822 airfoil. Viscous flow around this airfoil has been well documented in literature [15, 16] and provides a good validation case. The same mesh topology is used as shown in figures (12) and (13) but for viscous computations, the mesh spacing is much smaller near the body surface to accurately resolve the boundary layer. In the present study the cells adjacent to the surface were taken as a tenth of what they were in the inviscid computations. Other details of the domain were the same while the appropriate no-slip boundary conditions were applied to the airfoil surface.

The freestream Mach number is 0.75 while the angle of attack is 2.8^o. The flow is similar in nature to the transonic inviscid case described in the preceding section. However, the shape of the airfoil is such that a bottom surface shock does not form and the flow is smooth. On the top surface, a shock forms which decelerates the locally supersonic flow to subsonic speeds. Figure (18) shows the pressure variation around the airfoil and flow features such as the stagnation point and the shock wave can be easily seen. While there is no apparent difference with inviscid flow, a magnified view near the airfoil surface in figure (19) shows the velocity profile in the boundary layer. It is only in this thin region near the surface that viscous forces have significant influence on the flow. Figure (20) shows the coefficient of pressure on the upper and lower surface. The results shown here are obtained using a

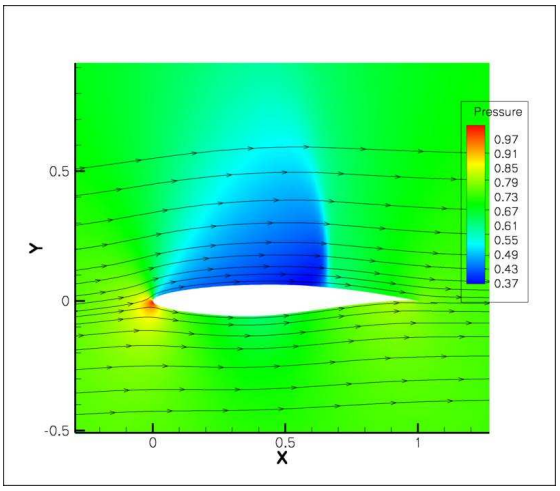

Figure 18: Pressure variation for the RAE2822 airfoil case

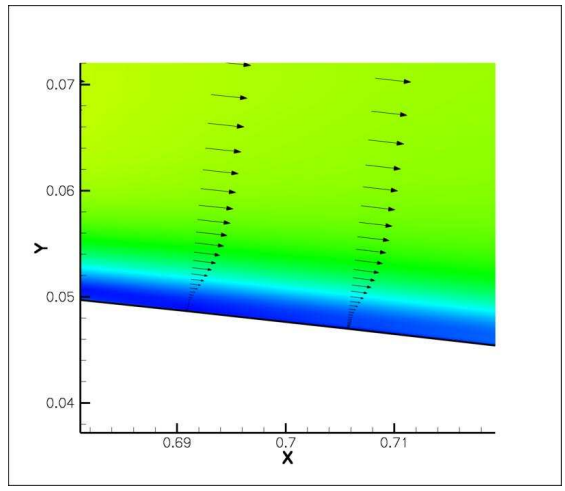

Figure 19: Magnified view of the boundary layer velocity profile

first order Roe scheme and thus, substantial diffusion can be seen across the shock. The coefficient of pressure on the airfoil surfaces is compared to those in [15] and good agreement is observed. Similar results are also shown in [16].

8 Conclusions and Future Work

An algorithm is developed to solve for laminar, viscous flows around airfoils and validated on several test problems. Initially, inviscid flow is considered which required the solution of the Euler equations of gasdynamics. A characteristic based reconstruction is used for the convective flux computations which exploits the wave nature of the solution. The Essentially Non-Oscillatory scheme is used for higher order spatial accuracy while the Local Lax-Friedrich's flux splitting is used for upwinding. Time integration is carried out using explicit Runge Kutta family of schemes. The solver is validated on several 1D and 2D problems exhibiting various flow features. Inviscid flow around airfoils is also computed to test the solver's performance on curvilinear meshes.

The inviscid flow solver is subsequently extended to solve for viscous flows. The Navier Stokes equations in their curvilinear coordinates form are solved using implicit time stepping based on the Backward Euler scheme. While the convective fluxes are reconstructed using the same techniques as in the inviscid solver, the dissipative terms are approximated using second order central differences. Linearization in time is carried out for the evaluation of non-linear fluxes in the implicit formulation. The Alternating-Direction-Implicit factorization is used to reduce the computational complexity of the system. The viscous flow solver is tested on relevant problems, using Cartesian as well as curvilinear meshes and representative results are shown.

Several improvements are required before the algorithm described in this report can be said to be comparable to existing flow solvers. The solution of the linear system for each time step in the implicit formulation is presently carried out using a naive LU decomposition. Since the matrices involved are block tri-diagonal, an efficient solver for such systems will reduce the computational cost. While the present algorithm is designed for steady state

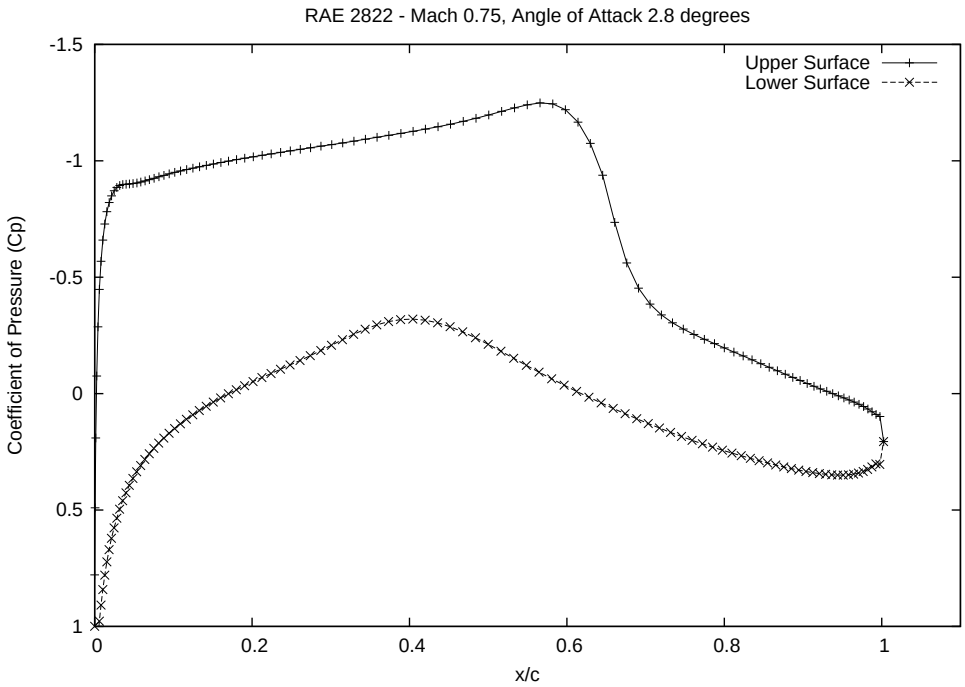

Figure 20: Surface coefficient of pressure variation for the RAE2822 airfoil case - Computed

computations, it's application to unsteady problems may not yield very accurate solutions due to several simplifications. Time accuracy can be improved by performing several sub-iterations for each time step and this needs to be investigated before the application of this solver to problems of unsteady aerodynamics.

The present algorithm solves for laminar flow since it lacks a turbulence model. To get physically relevant solutions for viscous flows, a turbulence model is required to account for the changes in the viscosity coefficient and thermal conductivity. Several turbulence models have been proposed in literature and are usually algebraic models or involve solution of an additional partial differential equation. Further studies are needed for the selection and incorporation of a turbulence model.

While most of the initial goals stated have been met, further work needs to be done before the problem of dynamic stalling can be studied. However, the algorithm, in its present state, provides a firm foundation on which the necessary additions can be built.

References

[1] Leishman J.G., *Principles of Helicopter Aerodynamics*, Cambridge University Press, 2006

[2] Anderson J.D., *Modern Compressible Flow: With Historical Perspective*, McGraw-Hill Professional, 2003

[3] Laney C.B., *Computational Gasdynamics*, Cambridge University Press, Cambridge, 1998

[4] Leveque R.J., *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, Cambridge, 2002

[5] Shu C.W., Osher S., *Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws*, ICASE report 97-65, 1997

[6] Beam R.M., Warming R.F., *An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation-Law Form*, Journal of Computational Physics, **22**, 1976, 87 - 110

[7] Beam R.M., Warming R.F., *An Implicit Factored Scheme for the Compressible Navier-Stokes Equations*, AIAA Journal, **16** (4), 1978, 393 - 402

[8] Steger J.L., *Implicit Finite-Differences Simulation of Flow about Arbitrary Two-Dimensional Geometries*, AIAA Journal, **16** (7),1978, 679 - 686

[9] Lax P.D., Liu X.D., *Solution of Two Dimensional Riemann Problems of Gas Dynamics by Positive Schemes*, SIAM Journal of Scientific Computing, **19** (2), 1998, 319 - 340

[10] Kurganov A., Tadmor E., *Solution of Two Dimensional Riemann Problems for Gas Dynamics without Riemann Problem Solvers*, Numerical Methods for Partial Differential Equations, **18** (5), 2002, 584 - 608

[11] Jaisankar S., Raghurama Rao S.V., *Diffusion regulation for Euler Solvers*, Journal of Computational Physics, **221**, 2007, 577 - 599

[12] Krivodonova L., Berger M., *High-order accurate implementation of solid wall boundary conditions in curved geometries*, Journal of Computational Physics, **211**, 2006, 492 - 512

[13] Marshall D.D., Ruffin S.M., *A New Inviscid Wall Boundary Condition Treatment for Embedded Boundary Cartesian Grid Schemes*, 42nd AIAA Aerospace Sciences Meeting and Exhibit, January 5-8, 2004, Reno NV

[14] Srinivasan G.R., Baeder J.D., *TURNS - A free-wake Euler Navier-Stokes numerical method for helicopter rotors*, AIAA Journal, **31** (5), 1993, 959 - 962

[15] Pulliam, T.H., *Efficient Solution Methods for The Navier-Stokes Equations*, Lecture Notes for the von Krmn Institute For Fluid Dynamics Lecture Series : Numerical Techniques for Viscous Flow Computation In Turbomachinery Bladings, von Krmn Institute, Rhode-St-Genese, Belgium , 1985

[16] Rossow C-C., *Efficient computation of compressible and incompressible flow*, Journal of Computational Physics, **220**, 2007, 879 - 899