Numerical Simulation of Dynamic Stall

Debojyoti Ghosh* James Baeder†(Adviser)

AMSC 663 - Advanced Scientific Computation I
(Mid-Year Project Report)
December 2007

Abstract

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 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 spanwise 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.

A short and simplified description of the flow around an airfoil, the generation of aerodynamic forces and process of stalling (including dynamic stalling) has been attempted in [2]. For a more complete understanding of this phenomenon, the reader is encouraged to refer to [1] and references therein. The various flow features that make up the process of

*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

dynamic stalling require that the full Navier-Stokes equations of fluid dynamics [3] be solved to obtain relevant results. The Navier-Stokes in their complete form describe the flow of a compressible, viscous fluid. It should be noted that while the rotorcraft itself may not operate at the same high forward velocities as fixed-wing aircrafts, the incoming flow velocities seen by the advancing blade can be high enough to justify the assumption of compressible flow. On the other hand, the flow seen by the retreating blade will be low-speed and at high angles of attack, it will be dominated by viscous effects. Therefore, the solution of the full Navier-Stokes equations is required to study the unsteady aerodynamics and dynamic stalling of a rotor blade cross-section.

As a first step towards the numerical solution of the Navier-Stokes equations, the inviscid Euler equations [3, 4] are solved. The Euler equations can be derived from the Navier-Stokes equations by assuming an inviscid, perfectly (heat) insulating fluid. 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) while the accurate prediction of drag requires the inclusion of viscous terms to account for the shear forces on the body. Flow separation and stalling are viscous phenomena and cannot be modeled by the Euler equations. However, an algorithm to solve the Euler equations provides the foundation which can then be extended to the Navier-Stokes equations.

The present report concentrates on the development of a 2D Euler solver through various stages, starting with 1D computations to flow computations over an airfoil. A finite volume approach [5] 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 [6] are used for flux reconstruction. These aspects of the algorithm are described more elaborately in subsequent sections. The report is outlined in the following manner. Section 2 provides the governing equations as well a brief explanation. Section 3 briefly describes the Essentially Non-Oscillatory technique of interpolation which is used in the numerical treatment of the governing equations. Section 4 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 Governing 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, eq. (3), relate the total energy to the flow variables.

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. The Navier-Stokes equations also have the same hyperbolic terms (in addition to the parabolic viscous terms) which can be handled in exactly the same way.

3 Essentially Non-Oscillatory Schemes

The Essentially Non-Oscillatory (ENO) class of schemes for spatial reconstruction has proven to be highly successful in finding the numerical solution of the Euler equations [6]. 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 [4]. 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 [6] 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

$$p_{N+1}(x) = f[x_0] + \sum_{i=1}^{N} f[x_0, ..., x_i]\Pi_{j=0}^{i-1}(x - x_j) \tag{6}$$

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 [6]. 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.

4 Numerical Scheme

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

$$\frac{d\mathbf{u}_i}{dt} = \mathbf{Res}(\mathbf{u}_i);\ \mathbf{Res}(\mathbf{u}_i) = -\frac{1}{\Delta x}(\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2}) \tag{7}$$

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:

$$\mathbf{F}_{i+1/2} = \sum_{k=1}^{m} f^k_{i+1/2}\mathbf{R}^k_{i+1/2} \tag{8}$$

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,

$$f^k_{i+1/2} = Rec(f^k_j;\ j \in S) \tag{9}$$

where Rec is the reconstruction procedure, dependent on the scheme used. In the present study, the Roe-Fixed (RF) formulation [6] is used to evaluate the characteristic flux in an upwinded fashion. The RF formulation is given as

$$\begin{aligned} f^k_{i+1/2} &= f^k_L,\ \text{if } \lambda^k_i, \lambda^k_{i+1/2}, \lambda^k_{i+1} > 0 \\ &= f^k_R,\ \text{if } \lambda^k_i, \lambda^k_{i+1/2}, \lambda^k_{i+1} < 0 \\ &= \frac{1}{2}[f^k_L + f^k_R + \alpha_{i+1/2}(u^k_R - u^k_L)],\ \text{otherwise} \end{aligned} \tag{10}$$

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 [6] 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. (7), 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 have been used in the present study in conjunction with the high order ENO spatial discretization.

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

$$\frac{d\mathbf{u_{ij}}}{dt} V_{ij} + \sum_{faces} \mathbf{F}.\hat{\mathbf{n}} dS = 0 \Rightarrow \frac{d\mathbf{u_{ij}}}{dt} = \mathbf{Res}(i,j) \tag{11}$$

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

$$\mathbf{Res}(i,j) = \frac{-1}{V_{ij}} [\sum_{l=1}^{4} \mathbf{F}.\hat{\mathbf{n}}_l dS_l \tag{12}$$

dS_l is the length of the cell interfaces. The semi-discrete equation, as given by eq. (11) 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 Boundary Conditions

The numerical treatment of the boundary is very important to ensure the correct conditions are satisfied while computing the flow at bondary 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.

- Flat wall: 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.

- Curved wall: This differs from the flat wall since the flow is turning and thus there is a centrifugal force involved. Thus, the pressure gradient normal to the wall has to counter this centrifugal force to prevent the fluid from separating from the surface and "flying off". The normal momentum equation has to be solved to find the pressure gradient in the direction normal to the surface

$$\frac{\partial P}{\partial \hat{\mathbf{n}}}_{wall} = -\rho \frac{\mathbf{u}^2}{R} \tag{13}$$

  where R is the radius of curvature of the surface and $\hat{\mathbf{n}}$ is the normal. Similar to the flat wall boundary condition, the normal velocity is reflected while the tangential velocity is kept the same at the ghost cell. The pressure at the ghost cell is found from the pressure at the interior point and the normal pressure gradient and the density is found assuming isentropic flow. The details of this treatment are outlined in [7] and [11].

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

6 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 [4, 5] 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 [8, 9] 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 [10], which is another benchmark case. Following these, the algorithm is tested for flow through a compression ramp [10], which involves a non-Cartesian grid. Finally, the 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 [11, 12]. The performance of the code is compared with the TURNS code [13] 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 [4] approach for spatial reconstruction.

1D Euler validation

The 1D Riemann problems [4] 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-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:

$$\begin{aligned} \rho_L &= 1 \quad ; \quad \rho_R = 0.125 \\ u_L &= 0 \quad ; \quad u_R = 0 \\ P_L &= 1 \quad ; \quad P_R = 0.1 \end{aligned} \tag{14}$$

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:

$$\begin{aligned} \rho_L &= 0.445 \quad ; \quad \rho_R = 0.5 \\ u_L &= 0.698 \quad ; \quad u_R = 0 \\ P_L &= 3.528 \quad ; \quad P_R = 0.571 \end{aligned} \tag{15}$$

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 [8]. 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 initial conditions are given as follows:

$$\begin{aligned} \rho_{NW} = 2.0 \quad &; \quad \rho_{NE} = 1.0 \\ u_{NW} = 0.75 \quad &; \quad u_{NE} = 0.75 \\ v_{NW} = 0.5 \quad &; \quad v_{NE} = -0.5 \\ P_{NW} = 1.0 \quad &; \quad P_{NE} = 1.0 \\ \\ \rho_{SW} = 1.0 \quad &; \quad \rho_{SE} = 3.0 \\ u_{SW} = -0.75 \quad &; \quad u_{SE} = -0.75 \\ v_{SW} = 0.5 \quad &; \quad v_{SE} = -0.5 \\ P_{SW} = 1.0 \quad &; \quad P_{SE} = 1.0 \end{aligned} \tag{16}$$

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 [8] and [9]. To make the comparison meaningful, the contours plotted in 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

This problem involves the reflection of an oblique shock at 30^o 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 [3]) 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.

Flow through a Compression Ramp

The flow through a channel with a 15^o 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^o 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 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 [3]. 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 (9) show the pressure variation and the streamlines of the flow while figure (10) show the pressure contours. The oblique shock and the expansion fan can be clearly distinguished in these two figures.

Inviscid Flow over Airfoil

Following the validation of the algorithm over simple problems in 2D, it is used to solve the flow over the NACA0012 airfoil. The domain is discretized using a body-fitted curvilinear mesh. The mesh topology is C-type and figure (11) 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 (12) shows a magnified view of the grid around the airfoil. Since the algorithm uses a finite volume formulation, it is expected that no additional modifications are needed to handle a body-fitted grid as this one.

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 condtions 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

$$C_p(x,y) = \frac{P(x,y) - P_\infty}{\frac{1}{2}\rho_\infty u_\infty^2} \tag{17}$$

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

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^o angle of attack. Figure (13) 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 (15) 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. As a comparison, figure (16) shows the coefficient of pressure obtained in [11]. The negative of the pressure coefficient is plotted in the reference and thus the y-axis is the negative of that in figure (15). It can be seen that the computed results agree very well with the results in [11].

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^o. Figure (14) 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 (14) 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 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. Figure (18) shows the results obtained in [11] (where once again, the negative of the pressure coefficient is plotted) and excellent agreement can be observed. Similar results are also presented in [12]

7 Conclusions

A 2D algorithm has been developed for the inviscid Euler equations and has been validated for the a variety of test problems, starting with 1D shock tube problems to inviscid airfoil

computations. While the solution to the Euler equations do not provide results relevant to stalling, this algorithm provides the foundation for a 2D Navier Stokes solver. The Euler equations and the Navier Stokes equations share the same hyperbolic flux terms while the Navier Stokes equations has additional dissipative terms which are absent in the Euler equations. Thus, the present code can be easily extended to the Navier Stokes equations and this is the next stage of the present study. Following the extension to and validation of the Navier Stokes solver, an appropriate turbulence model will be chosen to complete the model which will be used to simulate dynamic stalling. For further details, the timeline provided in [2] should be referred to.

References

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

[2] Ghosh D., *Numerical Simulation of Dynamic Stall*, Project Proposal, October 2007

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

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

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

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

[7] Dadone A., *Symmetry Techniques for the Numerical Solution of the 2D Euler Equations at Impermeable Boundaries*, International Journal for Numerical Methods in Fluids, **28**, 1998, 1093 - 1108

[8] 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

[9] 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

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

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

[12] 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

[13] 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

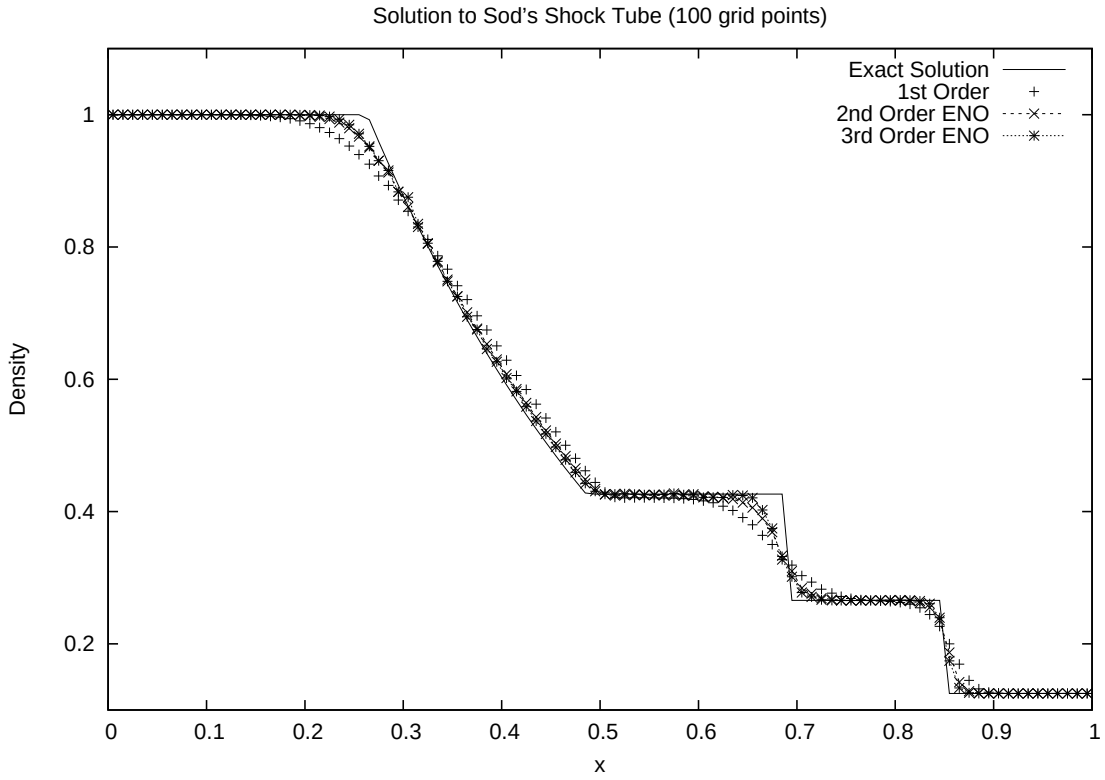

Figure 1: Solution of Sod's shock tube

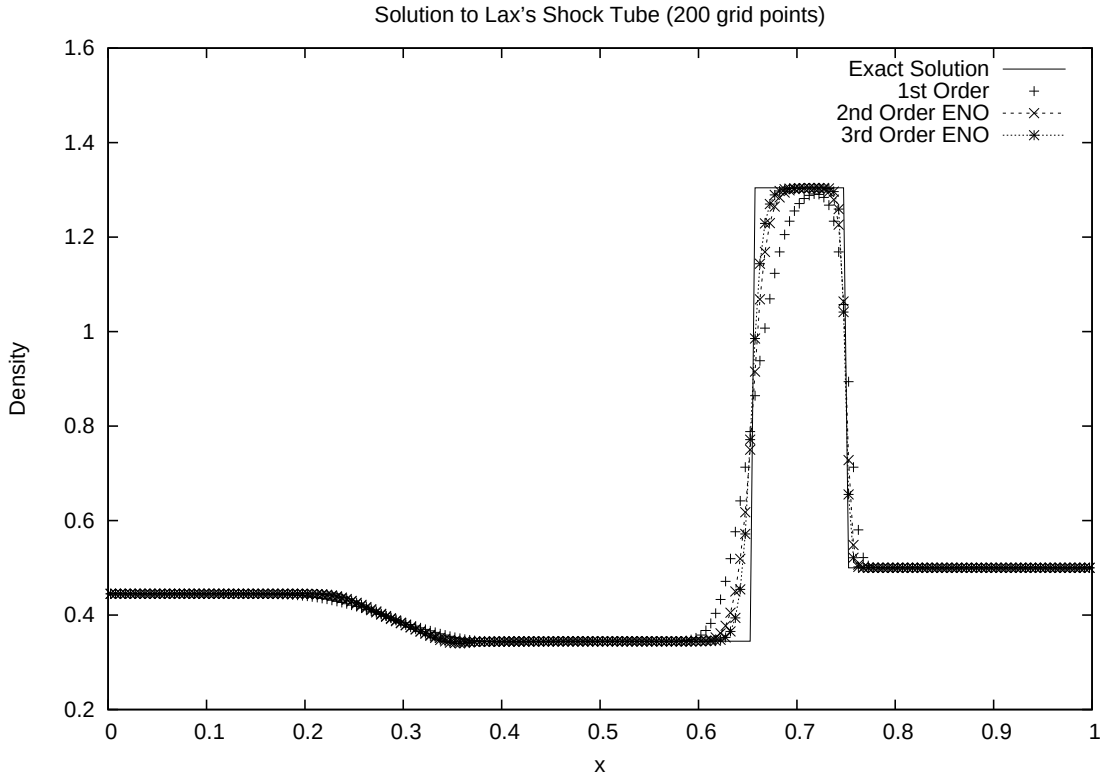

Figure 2: Solution of Lax's shock tube

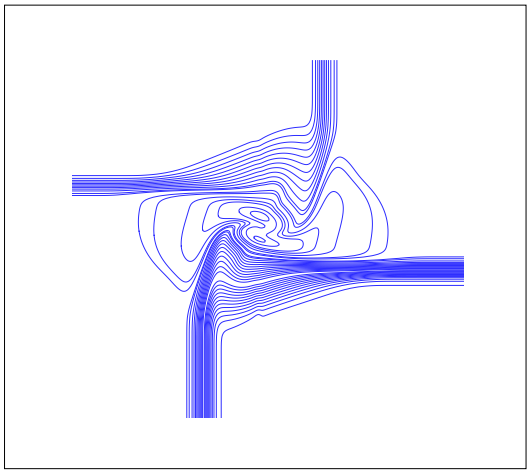

Figure 3: Density - 1st order

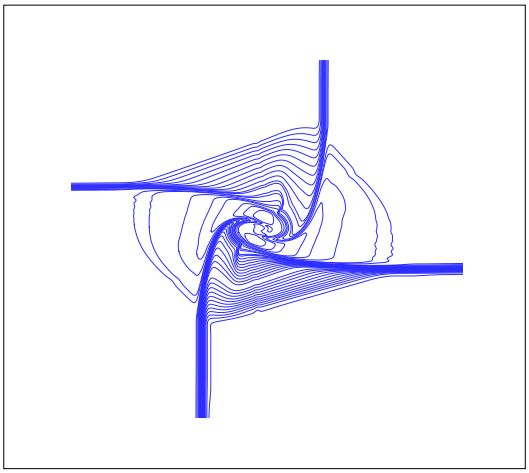

Figure 4: Density - 3rd order

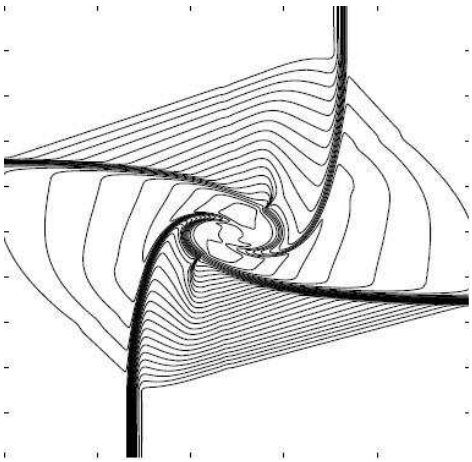

Figure 5: Density - from [8]

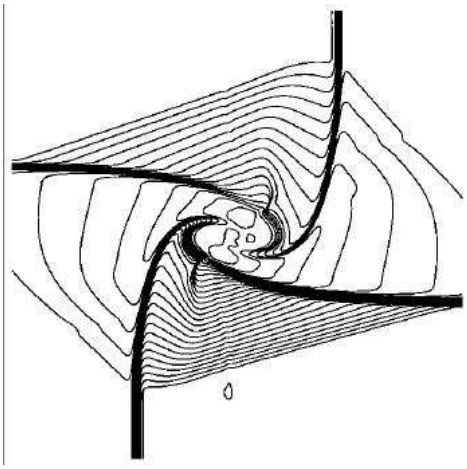

Figure 6: Density - from [9]

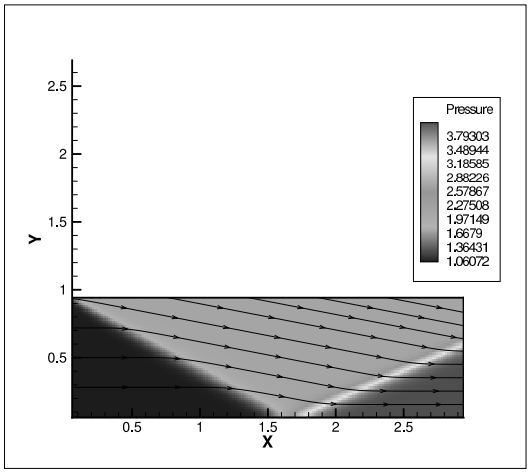

Figure 7: Oblique Shock Reflection (Pressure) - 2nd order

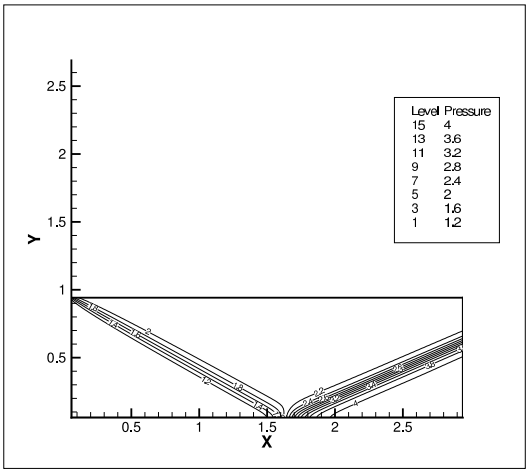

Figure 8: Oblique Shock Reflection (Pressure) - 3rd order

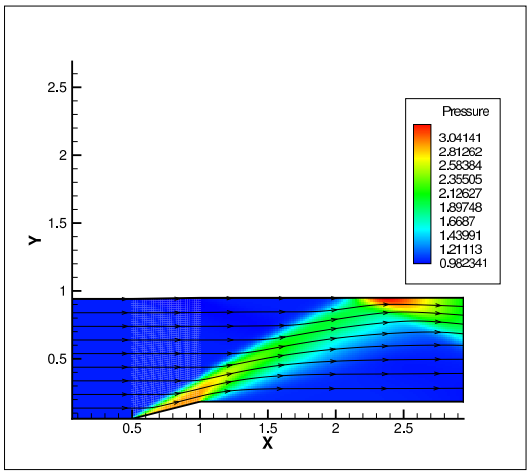

Figure 9: Compression Ramp (Pressure) - 2nd order

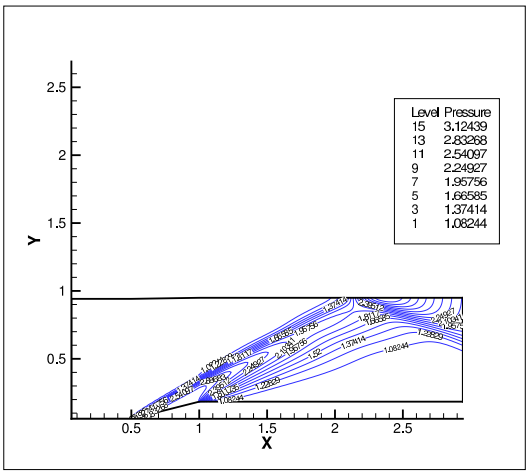

Figure 10: Compression Ramp (Pressure) - 3rd order

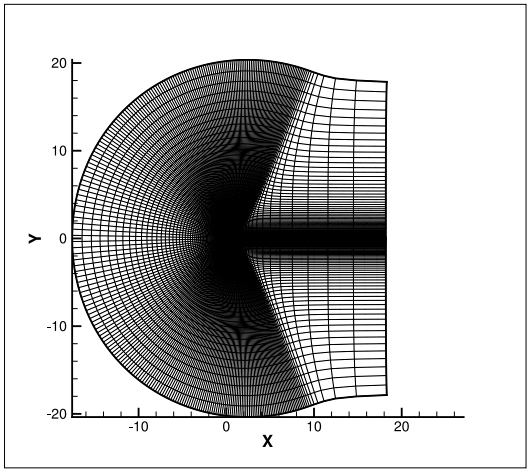

Figure 11: The computational domain for airfoil computations

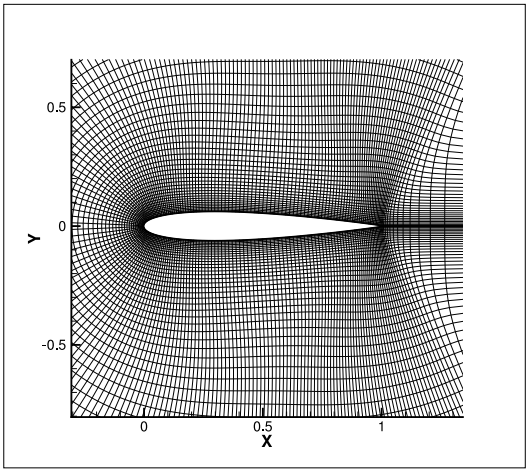

Figure 12: Magnified view of the grid near the airfoil

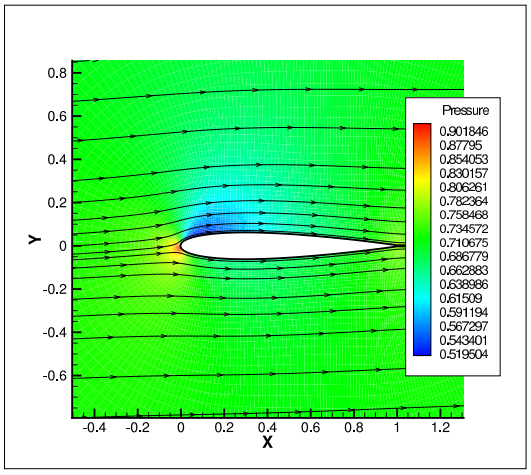

Figure 13: Pressure variation for the subsonic airfoil case

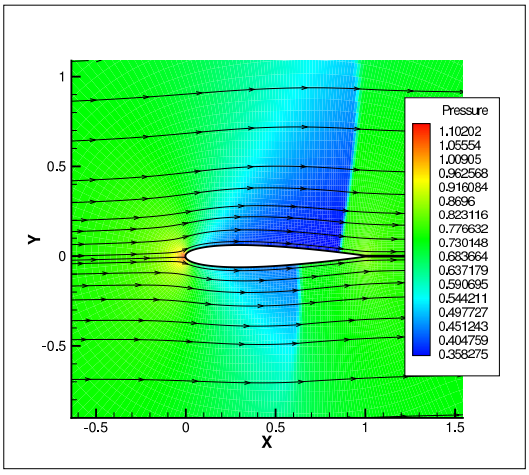

Figure 14: Pressure variation for the transonic airfoil case

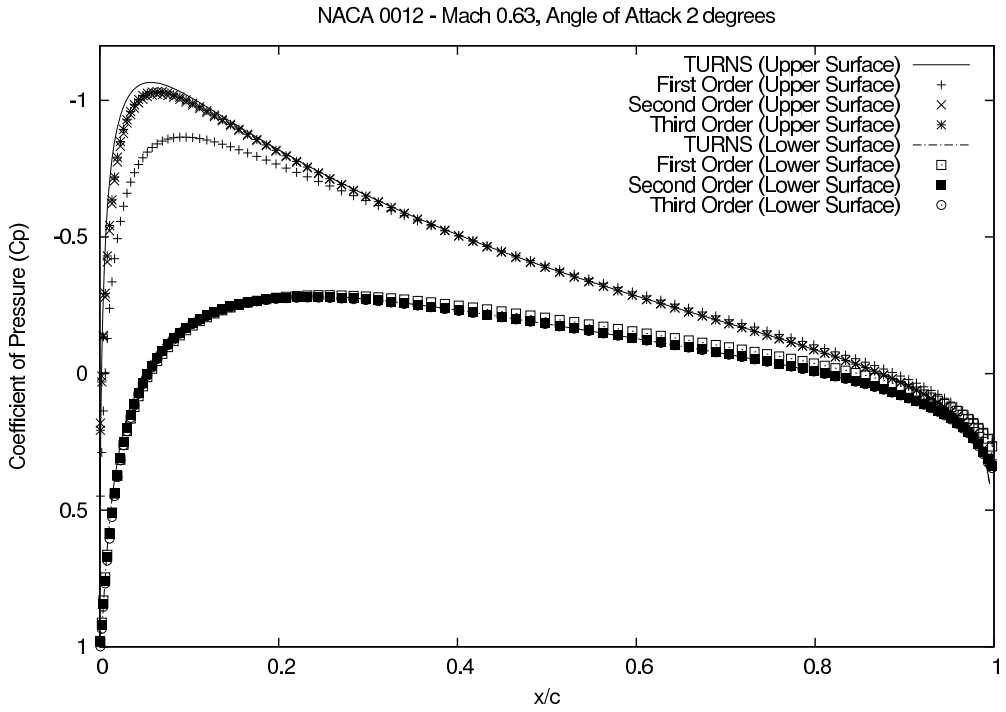

Figure 15: Surface coefficient of pressure variation for the subsonic airfoil case - Computed

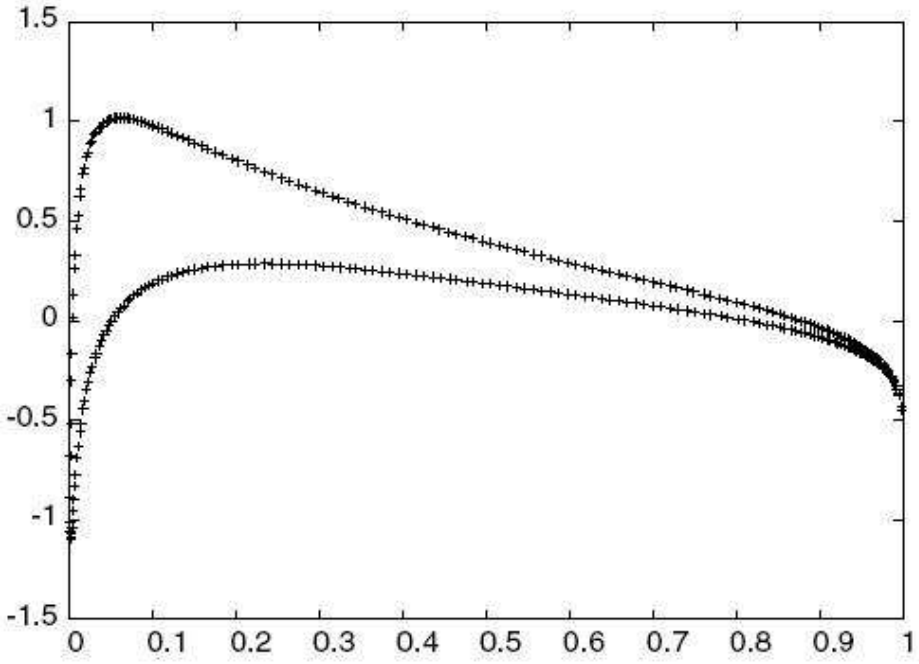

Figure 16: Surface coefficient of pressure variation for the subsonic airfoil case - from reference [11]

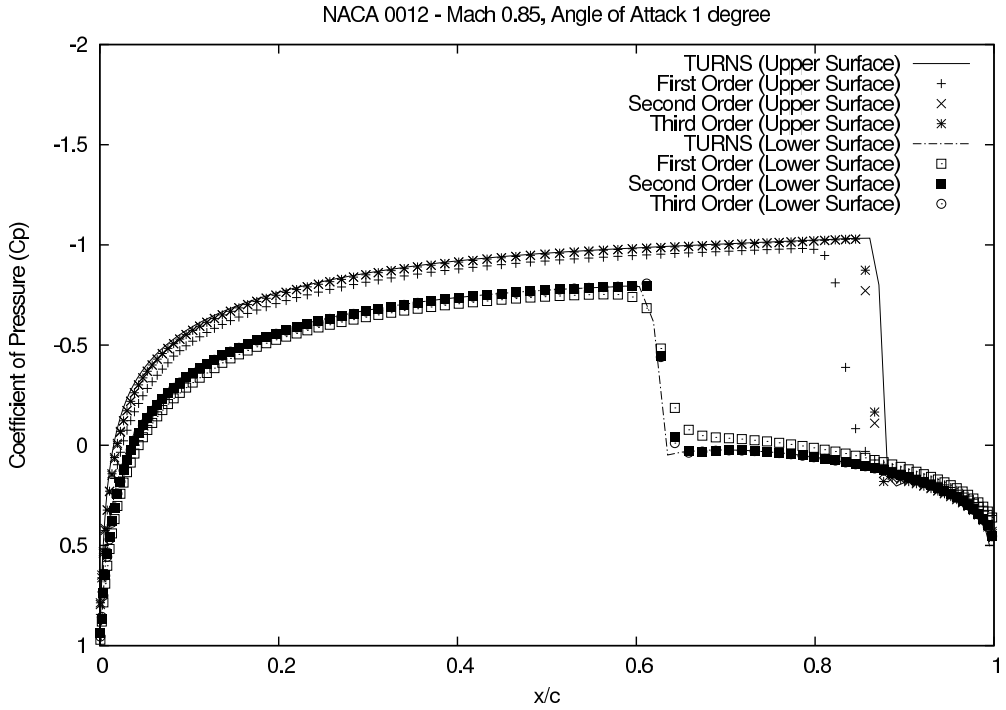

Figure 17: Surface coefficient of pressure variation for the transonic airfoil case - Computed

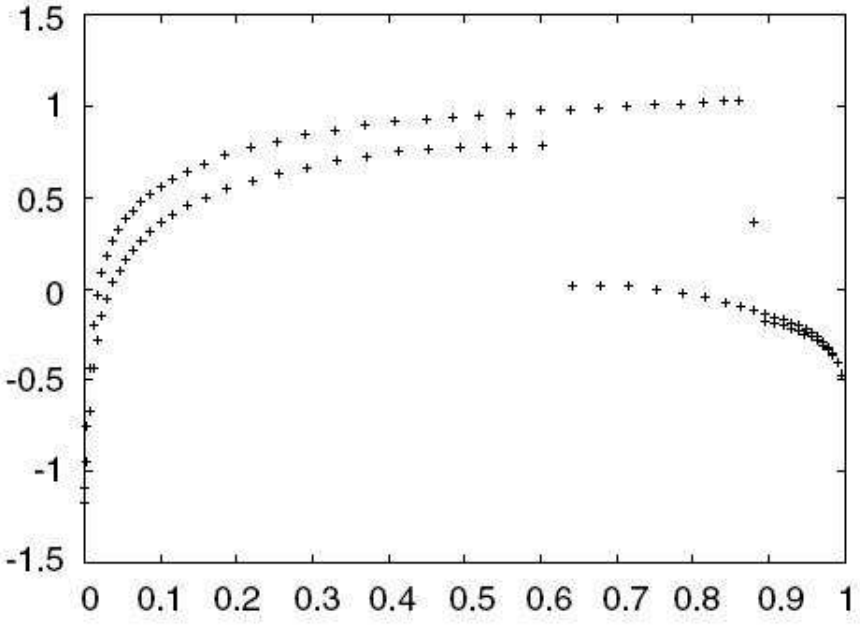

Figure 18: Surface coefficient of pressure variation for the transonic airfoil case - from reference [11]