Parallel Compressible Navier Stokes Solver: Shock Boundary Layer Interaction

Advanced Computational Fluid Dynamics, Final Project,
University of Michigan • Ann Arbor, Michigan • USA
Masters of Sciences (MSE) - Final Project

Gandharv Kashinath[1] and Matthias Ihme[2]
University of Michigan, Ann Arbor, MI, 48109*, USA*

Abstract: A parallel compressible Navier Stokes solver has been developed using a finite-volume algorithm. The robustness and accuracy of the resulting solution procedure have been assessed by performing many calculations in four different areas: shock-tube flows; regular shock reflection; supersonic boundary layer; and shock-boundary layer interactions. These numerical results have been validated and compared with corresponding exact solutions or experimental data.

I. Introduction

ACCURATE computations of shock-boundary-layer interaction are needed in many aerospace and aeronautical applications to predict, for example, flows around transonic airfoils, supersonic air intakes, or deflected control surfaces of vehicles at transonic or supersonic speed. Because of the limitations of Reynolds-averaged Navier–Stokes equations (RANS), large-eddy simulation (LES) is becoming more and more popular to simulate numerically these kinds of flows (1). In pursuit of developing a three dimensional LES solver it is essential to develop a laminar shock boundary layer interaction code as the first building block before incorporating the turbulence modeling essential in studying such flows. In this project an unsteady second order accurate finite-volume, explicit, numerical algorithm was developed, tested and validated with published results to study the laminar shock boundary layer interaction problem. In developing a three dimensional codes capable of doing turbulence modeling it is essential to enhance the speed of computation and in order to accomplish this goal, the numerical algorithm thus developed was parallelized using MPI in FORTRAN.

The governing equations are introduced first, followed by the presentation of the numerical implementation of the finite volume algorithm, Riemann solvers, boundary conditions and the parallelization technique. Finally the results from various test cases are presented and compared with published data.

II. Governing Equations

The flow is described by the compressible Navier-Stokes equations and the integral form of the equations can be expressed as (2);

$$\frac{\partial}{\partial t}\iiint_V \boldsymbol{U}\,dV + \oiint_S (\overline{\boldsymbol{H}}.\hat{\boldsymbol{n}})\,dS = 0$$

where, $\boldsymbol{U}$ is the state vector of the conserved quantities and the finite volume (V) is bounded by the surface S and the tensor $\overline{H}$ in Cartesian coordinates is given as;

$$\overline{\boldsymbol{H}} = \left((\boldsymbol{E_{inv}} + \boldsymbol{E_{vis}})\hat{\imath} + (\boldsymbol{F_{inv}} + \boldsymbol{F_{vis}})\hat{\jmath} + (\boldsymbol{G_{inv}} + \boldsymbol{G_{vis}})\hat{k}\right)$$

[1] Graduate Student, Aerospace Engineering, University of Michigan, gandharv@umich.edu.
[2] Assistant Professor, Aerospace Engineering, University of Michigan, mihme@umich.edu.

for the finite volume represented by a cube;

$$\overline{\boldsymbol{H}}.\hat{\boldsymbol{n}}\, dS = (\widetilde{\boldsymbol{E}}dydz - \widetilde{\boldsymbol{F}}dxdz + \widetilde{\boldsymbol{G}}dydx)$$

where,

$$\widetilde{\boldsymbol{E}} = \boldsymbol{E}_{inv} + \boldsymbol{E}_{vis}$$

similarly for $\widetilde{\boldsymbol{F}}$ and$\widetilde{\boldsymbol{G}}$. Finally assuming the finite volume, V is constant; we can express the above integral equation as follows;

$$\frac{\partial}{\partial t}(\boldsymbol{U})_{ijk} = -\frac{1}{V_{ijk}}\left[\sum_{l=1}^{6}(\widetilde{\boldsymbol{E}}dydz - \widetilde{\boldsymbol{F}}dxdz + \widetilde{\boldsymbol{G}}dydx)_l\right] \qquad (1)$$

where $(\boldsymbol{U})_{ijk}$ is the value of $\boldsymbol{U}$ associated with the finite volume i, j, k and the summation is applied to the fluxes at all exterior sides (faces) of the finite volume.

The state vector of the conservative variables is given by $\boldsymbol{U}$ = (ρ, ρu_1, ρu_2, ρu_3, Et), where the inviscid fluxes are defined as;

$$\boldsymbol{E_{inv}} = \begin{Bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ \rho u_1 u_3 \\ (E_t + p)u_1 \end{Bmatrix} \quad \boldsymbol{F_{inv}} = \begin{Bmatrix} \rho u_2 \\ \rho u_2 u_1 \\ \rho u_2^2 + p \\ \rho u_2 u_3 \\ (E_t + p)u_2 \end{Bmatrix} \quad \boldsymbol{G_{inv}} = \begin{Bmatrix} \rho u_3 \\ \rho u_3 u_1 \\ \rho u_3 u_2 \\ \rho u_3^2 + p \\ (E_t + p)u_3 \end{Bmatrix}$$

and the viscous fluxes are;

$$\boldsymbol{E_{vis}} = \begin{Bmatrix} 0 \\ \tau_{11} \\ \tau_{12} \\ \tau_{13} \\ (\tau \boldsymbol{u})_1 - q_1 \end{Bmatrix} \quad \boldsymbol{F_{vis}} = \begin{Bmatrix} 0 \\ \tau_{21} \\ \tau_{22} \\ \tau_{23} \\ (\tau \boldsymbol{u})_2 - q_2 \end{Bmatrix} \quad \boldsymbol{G_{vis}} = \begin{Bmatrix} 0 \\ \tau_{31} \\ \tau_{32} \\ \tau_{33} \\ (\tau \boldsymbol{u})_3 - q_3 \end{Bmatrix}$$

The shear-stress tensor τ_{ij} is given by (summation rule applies and δ_{ij} = 1 for i = j and 0 otherwise, i, j = 1, 2, 3);

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right)$$

The viscous dissipation in the energy equation is calculated from;

$$(\tau \boldsymbol{u})_l = \tau_{l1}u_1 + \tau_{l2}u_2 + \tau_{l3}u_3$$

and the heat flux caused by conduction is given by;

$$q_l = -k\frac{\partial T}{\partial x_l}$$

The total energy is given by;

$$E_t = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u_1^2 + u_2^2 + u_3^2)$$

and finally the temperature is calculated using the equation of state for a perfect gas;

$$T = \frac{p}{\rho R_g}$$

Since a Cartesian coordinate system with an orthogonal grid was used for calculations the velocities, $u_1 = u$, $u_2 = v$, $u_3 = w$ in all the above equations. The kinematic viscosity, $\nu = \mu/\rho$ was prescribed as an input instead of the dynamic viscosity, μ.

III. Numerical Approach

In order to solve the above finite volume formulation of the Navier-Stokes equations, a second order accurate spatial discretization was employed and the operations on each term were performed using the conserved quantities instead of the primitive variables. The time-marching method was decoupled from the spatial discretization and a three stage marching method as used for integrating ODE's (Runge-Kutta) was employed. The full time-space method's accuracy is third-order if the equations are linear i.e. in the absence of shocks. Since this is computationally intensive to achieve the method is considered to be formally second-order accurate.

A. Computing Interface States

A quadratic interpolation in each cell was employed for better spatial resolution. The spatial interpolation is done according to the κ-scheme with $\kappa = 1/3$. The original equation of the κ-scheme is cast into the form used by Pete Sweby in one dimension as (3);

$$\boldsymbol{U}_{i+\frac{1}{2},L} = \boldsymbol{U}_i + \frac{\varphi(r_i)}{2}(\boldsymbol{U}_i - \boldsymbol{U}_{i-1})$$

$$\boldsymbol{U}_{i-\frac{1}{2},R} = \boldsymbol{U}_i - \frac{r_i\varphi\left(\frac{1}{r_i}\right)}{2}(\boldsymbol{U}_i - \boldsymbol{U}_{i-1})$$

$$r_i = \frac{\boldsymbol{U}_{i+1} - \boldsymbol{U}_i}{\boldsymbol{U}_i - \boldsymbol{U}_{i-1}}$$

where φ is the Koren Limiter given by (3);

$$\varphi(r) = \frac{2r^2 + r}{2r^2 - r + 2}$$

$$r\varphi\left(\frac{1}{r}\right) = \frac{r^2 + 2r}{2r^2 - r + 2}$$

Similar interpolations were carried out in the remaining two directions namely the *j* and *k* directions.

B. Riemann Solvers - Inviscid Flux Computation

In order to calculate the Inviscid fluxes at each face the exact (Gudonov Type) (4) or the approximate Riemann (Roe's approximate) (5) solvers were employed. Since an orthogonal grid was used to do the computation, the interpolated conserved quantities on either side of each face were sent to the Riemann solver along with the direction in which the Riemann problem needed to be solved, thus determining the appropriate Flux vector, $\boldsymbol{E_{inv}}$, $\boldsymbol{F_{inv}}$ or $\boldsymbol{G_{inv}}$. The following are the inputs were sent to the Riemann solver;

$$\boldsymbol{U_{stateL}} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E_t \end{Bmatrix}_L \qquad \boldsymbol{U_{stateR}} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E_t \end{Bmatrix}_R$$

and the direction x, y or z to pick the appropriate Flux vector. These conserved quantities were converted to the primitives; [ρ u v w p] and using the direction identifier (switch) the Riemann Solver was applied to the appropriate variable, for instance if u was the identified velocity then the Riemann solver was used to solve for; [u^* ρ_L^* ρ_R^* p^*]. The other two velocities v and w are passively convected and depending on the direction of the contact discontinuity $v^* = v_L$ or v_R and $w^* = w_L$ or w_R was chosen. Once these quantities were determined the time slicing method was used and the appropriate (*) quantities were chosen and the Invsicid Flux Vector was assembled. This computation was carried out while running a loop across all faces in the domain.

C.Viscous Flux Computation - Finite Difference Approximation

In order to compute the viscous fluxes a second order finite difference discretization was employed. Presented below is one of several such gradient operations needed at each face of the control volume. Using the Taylor Series expansion for the quantity $U'_{i-\frac{1}{2}} = f(U_i, U_{i-1}) = \frac{\partial U}{\partial x}$ can be written as (6);

$$\left(U_{i-1} = U_{i-\frac{1}{2}} - U'_{i-\frac{1}{2}}(\Delta x/2) + U''_{i-\frac{1}{2}}\left(\frac{(\Delta x/2)^2}{2!}\right) + O(\Delta x)^3\right) \times a_0$$

$$\left(U_i = U_{i-\frac{1}{2}} + U'_{i-\frac{1}{2}}(\Delta x/2) + U''_{i-\frac{1}{2}}\left(\frac{(\Delta x/2)^2}{2!}\right) + O(\Delta x)^3\right) \times a_1$$

Adding the above two equation gives;

$$a_0 U_i + a_1 U_{i-1} = U_{i-\frac{1}{2}}[a_0 + a_1] + U'_{i-\frac{1}{2}}(\Delta x/2)[-a_0 + a_1] + U''_{i-\frac{1}{2}}\left(\frac{(\Delta x/2)^2}{2!}\right)[a_0 + a_1] + O(\Delta x)^3$$

constraints to the above equation are;

$$a_0 + a_1 = 0, \qquad -a_0 + a_1 = 1$$

Solving for a_0, a_1 and writing an expression for $U'_{i-\frac{1}{2}}$ yields;

$$U'_{i-\frac{1}{2}} = \frac{U_i - U_{i-1}}{\Delta x} + O(\Delta x)^2$$

which is nothing but the central difference operator at face, $i - \frac{1}{2}$.

Further, the derivatives at $i - \frac{1}{2}$ which involve $\partial U/\partial y$ and $\partial U/\partial z$ needed special treatment as these quantities were offset from the face, $i - \frac{1}{2}$. In order to compute these derivatives, the quantity U was interpolated using second order interpolation to the face, $i - \frac{1}{2}$ and then the central difference operator was applied on these quantities keeping in mind the distance separating the interpolated quantities. This was done for all other operators needed in the viscous flux computation in all three directions.

The heat flux, q, involves a gradient calculation involving the temperature and this computation was also done similarly using a second order central difference scheme along with the equation of state for a perfect gas. The density term in the kinematic viscosity expression was also computed using second order interpolation scheme.
Finally the viscous flux at each face was assembled keeping in mind the appropriate flux vector, $\boldsymbol{E_{vis}}$, $\boldsymbol{F_{vis}}$ and $\boldsymbol{G_{vis}}$ in the respective directions.

Presented below is a pictorial representation of the gradient calculation of various quantities. This can be further extended to a three dimensional setting.

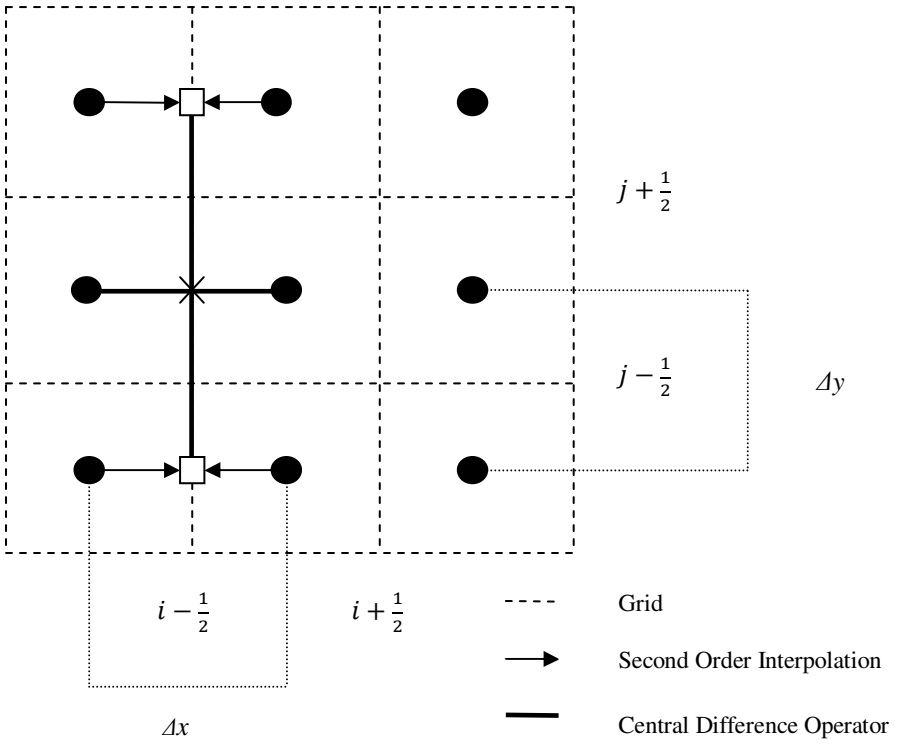


Figure 1: Pictorial representation of Gradient Calculation at face $i-\frac{1}{2}$ for Viscous Flux Calculation.

D.Boundary Conditions

Several problem specific boundary conditions were used and presented below is an overall general description of these boundary condition. It must be noted that all the boundary procedures are implemented on the conserved quantities.

1. Euler Solver – Inflow

A soft, "floating" boundary (Neumann) procedure is implemented in which two rows of virtual cells are given exactly the same values as the first and second real row of cells. This did the best job in absorbing the reflected shock when it arrived at the inlet, especially if flow reversal occurred behind the shock. The two virtual cell procedure was implemented to retain the second order accurate scheme at the boundaries.

2. Euler Solver – Outflow

Similar to the Inflow boundary conditions a simple Neumann outflow boundary condition was employed where the internal cell values were extrapolated using second order extrapolation scheme, to the virtual cells.

3. Euler Solver – Hard (Wall) Boundary Condition

Across the wall boundary, a couple of virtual cells that are the mirror image of its real neighbor were introduced. The reflection procedure was formulated in the frame of the wall face and its normal, since it is the normal velocity that must be set to zero. This condition is given by;

$$\boldsymbol{v}.\hat{n} = 0$$

4. General Neumann Boundary Condition

In order to implement the general Neumann boundary conditions similar to the inflow and outflow boundary conditions of Euler solver, the following conditions were enforced;

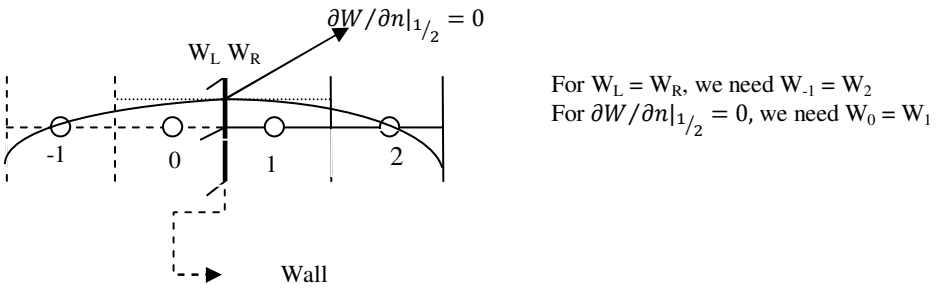


Figure 2: Neumann Boundary Condition Implementation at a Wall.

This boundary condition ensured that we enforced the no jump and known flux conditions at the wall.

5. General Dirichlet Boundary Condition

The General Dirichlet Boundary condition was important in simulating an inflow profile for the compressible boundary layer and the shock boundary layer problems. Since the entire south boundary was considered to be a wall, an inflow profile similar to the analytical solution of a boundary layer needed to be prescribed. This was done by maintain the virtual cells with the analytical solution obtained by solving for the boundary layer profile. Prescribed profiles used to simulate this inflow are presented in the results section.

6. General Wall Boundary Condition

In order to implement the No-Slip boundary conditions the velocity components at the wall were set to zero. Since the conserved quantities were used in the implementation this translated to setting the quantities, (ρu, ρv, ρw) to zero. Besides this velocity implementation, the energy equation needed special treatment at no slip walls. The temperature at the wall was prescribed and using the equation of state the energy terms in the virtual cells were computed. Care was taken to interpolate the wall temperature to the cell centers of the virtual cells.

E. Time – Marching Scheme

The Runge-Kutta three stage time marching scheme as presented below is employed to do the time marching. Since the time integration is uncoupled from the spatial operations a simple ODE type integration scheme can be employed (2) ;

$$\begin{aligned}
\boldsymbol{U}^{(0)} &= \boldsymbol{U}^{(n)}, \\
\boldsymbol{U}^{(1)} &= \boldsymbol{U}^{(0)} + \frac{\Delta t}{3} RHS\left(\boldsymbol{U}^{(0)}\right), \\
\boldsymbol{U}^{(2)} &= \boldsymbol{U}^{(0)} + \frac{\Delta t}{2} RHS\left(\boldsymbol{U}^{(1)}\right), \\
\boldsymbol{U}^{(3)} &= \boldsymbol{U}^{(0)} + \Delta t RHS\left(\boldsymbol{U}^{(2)}\right), \\
\boldsymbol{U}^{(n+1)} &= \boldsymbol{U}^{(3)}
\end{aligned}$$

where RHS is the right hand side of Eqn. (1). The scheme is third order accurate in time when applied to a linear system of conservation laws, but only second-order accurate when applied to a non-linear system. The third stage is needed to insure the scheme's linear stability when combined with a third-order accurate spatial discretization, i.e., with κ=1/3 in the spatial operator. The stability condition for this scheme when combined with κ=1/3 is (7);

$$\upsilon_{i,j} \leq 1.35$$

where the CFL condition is defined as (7);

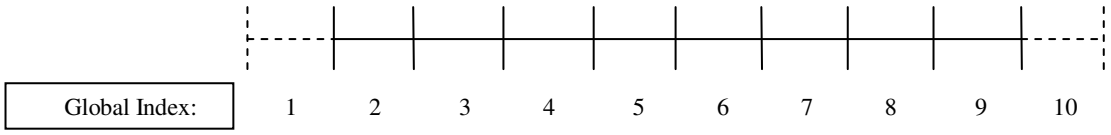
$$v_{i,j} = \frac{(\Delta t)_{ijk}}{2V_{ijk}} \left[\sum_{l=1}^{6} (q_l + a_l)(\Delta s)_l \right]_{ijk}$$

where, q_l is normal velocity and a_l is the speed of sound associated with the face *l*. The local CFL was computed for each cell and monitored for any violation of the CFL condition.

IV. Parallel Algorithm

In order to parallelize the code for speed, an algorithm employed by a code developed at Stanford called NGA was used. This code is a finite-difference parallel algorithm written to solve various incompressible flows. The data structure from this code was extracted along with the parallel algorithm established to parallelize the solver in two dimensions namely x and y. The third dimension namely, the z-direction is always treated to be periodic. The parallel communication in this algorithm is carried out using Message Passing Interface (MPI) routines. Presented below is a pictorial demonstrating a brief overview of the indexing and communication between each sub-domains which formed the essential components of the parallel structure;

- **Global Mesh with virtual cells:** N_x =



- **Local Mesh with virtual cells:** N_x = 8, nproc = 2

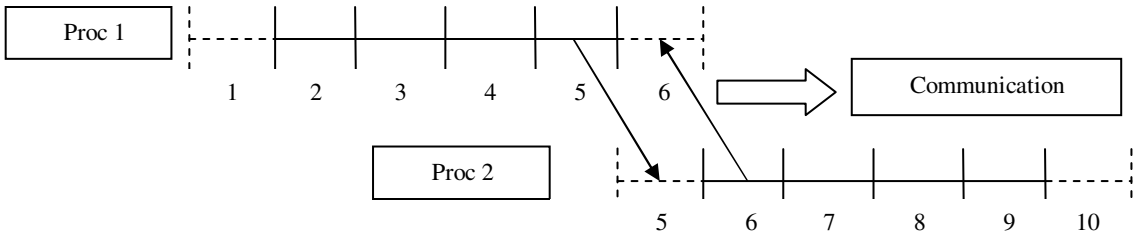


Figure 3: Indexing and Communication in Global and Local domains.

V. Results and Discussion

The Finite-Volume Parallel algorithm thus developed was used to run several cases and was validated with published data and experimental results. The results accompanied by a brief discussion are presented below.

A. Unsteady Shock-Tube Problem

Numerical results for the standard Sod's shock-tube problem have been obtained by solving the 3-D Euler equations and this has been compared with the exact 1-D solutions (8). This test problem was used to prove that the code was directionally independent by setting up the shock-tube problem in either of the two directions namely x and y and obtaining the same solution. Since the code was periodic in z, this direction was ignored. The boundary conditions for this problem were similar to the boundary conditions employed in Ref. (9) where all the boundaries are considered to be Neumann type. The computational domain is covered by 100 x 20 uniformly distributed cells ($\Delta x = \Delta y = 0.01$) and the following initialization was used;

Table 1: Normalized Initial Condition for Unsteady Shock-Tube Problem

	$0 < x < 0.5$			$0.5 \leq x \leq 1$		
	ρ	u	p	ρ	u	p
Sod's Problem	1	0	1	0.125	0	0.1

The same test case was again run with the Navier-Stokes solver with a Reynolds number, Re = 2000 and this was compared with the Euler solutions due to lack of exact solution for these equations. The results were plotted at t=0.15s and is presented in Fig. 5. From these plots it can be concluded that the solution is as predicted and shows a consistent density and pressure profiles. The Navier-Stokes solution shows a smeared out shock and expansion waves thus indicating the influence of the viscous terms. The results are unchanged on changing the direction of operation thus confirming the directional independence of the solution.

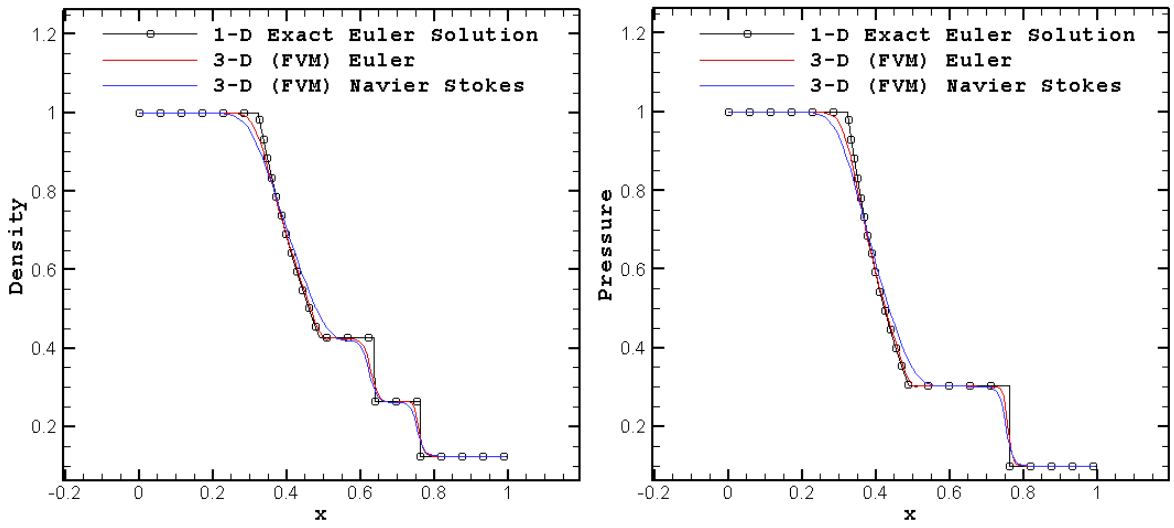


Figure 4: Sod's problem: (a) density; (b) pressure.

B. Inviscid Oblique Shock Reflection on a Plane Wall

Numerical solutions of the regular shock reflection with an incident shock angle β of 29° and a free stream Mach number M_∞ of 2.9 have been obtained by solving the Euler equations. The calculations are started by assuming uniform flow with $M_\infty = 2.9$ except for the top boundary, where the variables are consistently overspecified from the jump conditions. The computational domain extends from x = 0 to 4 and from y = 0 to 1. Uniform grids are used to discretize the domain. A "Hard Boundary" condition is employed for the lower plane wall and Neumann boundary conditions is applied to all other boundaries. The computational domain is covered by 121 x 41 uniformly distributed cells and a converged solution was obtained. The pressure contours followed by the comparison of computed and exact solutions along y = 0.4878 is illustrated in Fig. 6 are presented. The results show a strong correlation along y=0.4878 and also the pressure contour is consistent with the results presented in Ref (9).

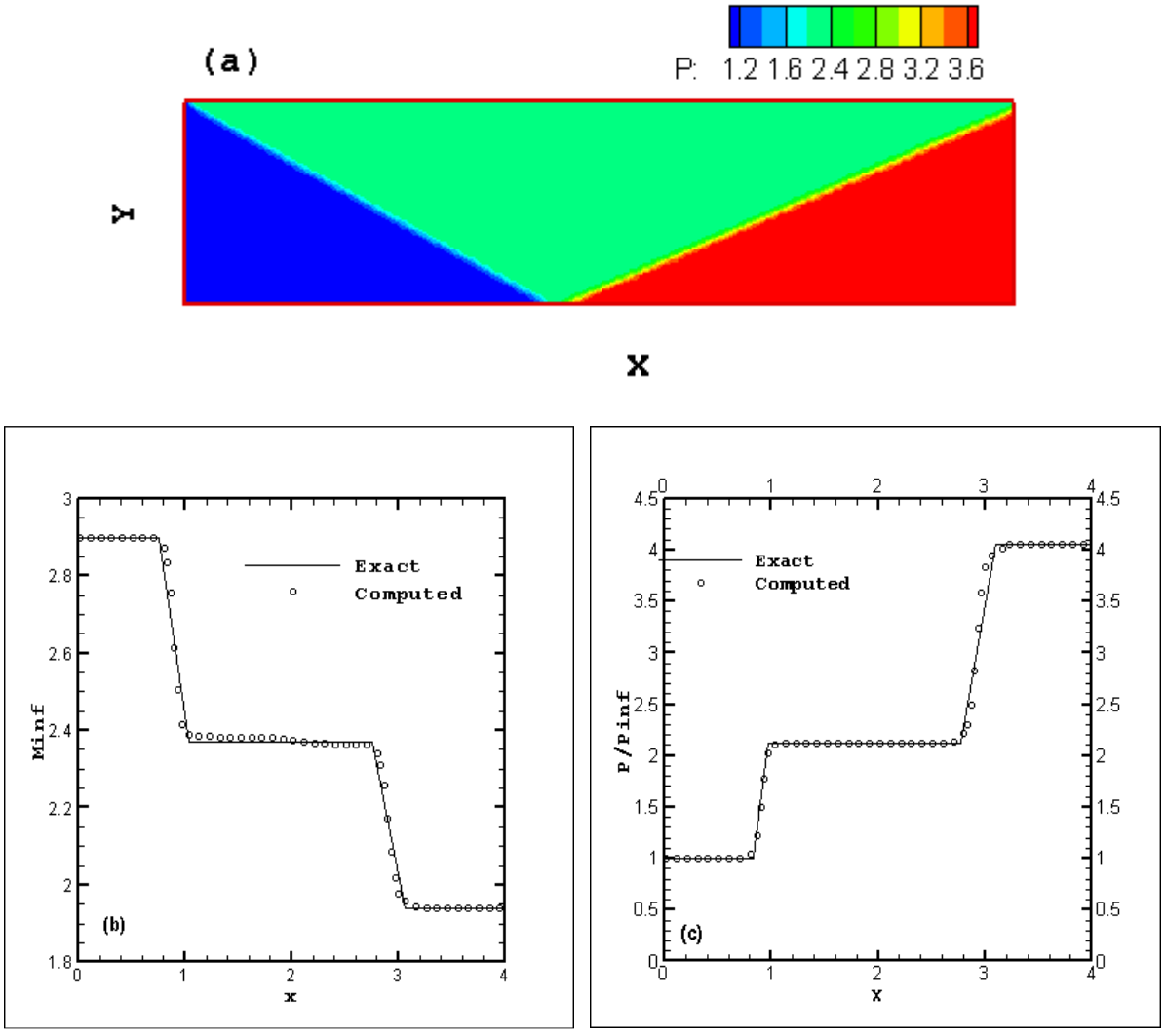


Figure 5: Oblique shock reflection in inviscid flow ($M_\infty = 2.9$, β = 29°): (a) pressure contours; (b) M_∞ distribution at y = 0.4878; (c) pressure distribution at y = 0.4878.

C.Supersonic Boundary Layer

A Mach 2.2 Laminar Flow over an adiabatic flat plate is calculated by solving the Navier-Stokes equations. The Reynolds number based on the free stream condition and a reference length of 0.08m is, Re=9.8645 × 10^4. In the streamwise direction, $0.0 \leq x \leq 4.0$, 74 grid points with a constant $\Delta x = 0.054$ are used. In the direction normal to the wall, $0.0 \leq y \leq 2$, 62 grid points with the following distribution are used (9);

$$\begin{aligned} \Delta y_j &= 1.5625 \times 10^{-4}, & 1 \leq j \leq 4, \\ \Delta y_j &= 1.1868\, \Delta y_{j-1}, & 5 \leq j \leq 33, \\ \Delta y_j &= 3.75 \times 10^{-2}, & 33 \leq j \leq 62. \end{aligned}$$

In order to compute this solution, an inflow profile was prescribed and maintained using Dirichlet type boundary condition. Since the entire south boundary was considered to be a flat plate, an inflow profile which was obtained using a flat plate boundary layer analytical solution as prescribed in Ref. (10) was chosen as follows;

$$\begin{aligned} u_j &= U_\infty \sin\left(\frac{\pi}{2}\frac{y_j}{l}\right), & 0 \leq y_j \leq 0.6 \\ u_j &= U_\infty, & y_j \geq 0.6. \end{aligned}$$

where;

$$l = \sqrt{\frac{2\pi^2}{4-\pi}}\sqrt{\frac{7.22E3\,\nu}{U_\infty}}$$

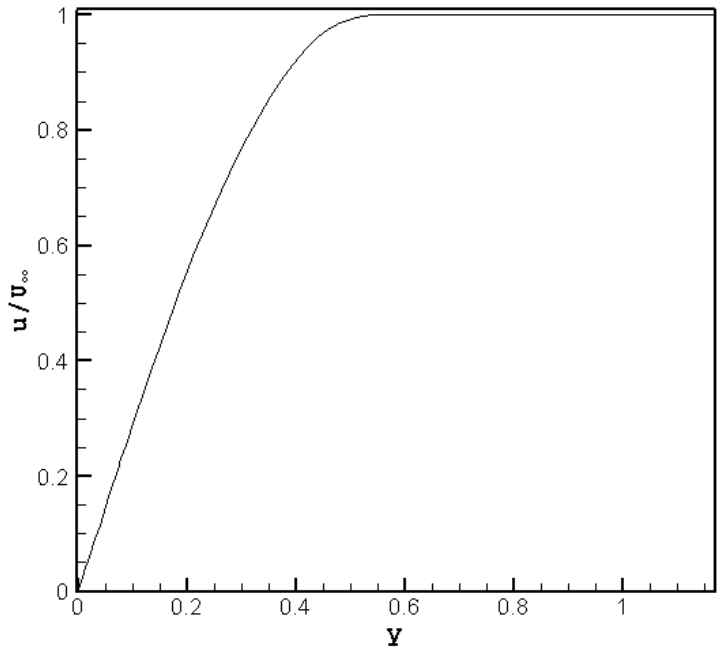


Figure 6: Inflow profile used to simulate a flat plate boundary layer.

The simulation was run using a $\Delta t = 0.001$ and the velocity and temperature profiles thus obtained were normalized with free stream velocity and temperature, respectively. They are then plotted against y/θ, where θ denotes the local momentum thickness. In order to compute the local momentum thickness, the boundary layer edge is assumed to be at a point where the u-velocity is 99.5% of the free stream value. Presented below, Fig. 7, is the temperature and velocity profiles which were compared with experimental results presented in Ref (9). The results match closely with the experimental results presented in Ref. (9).

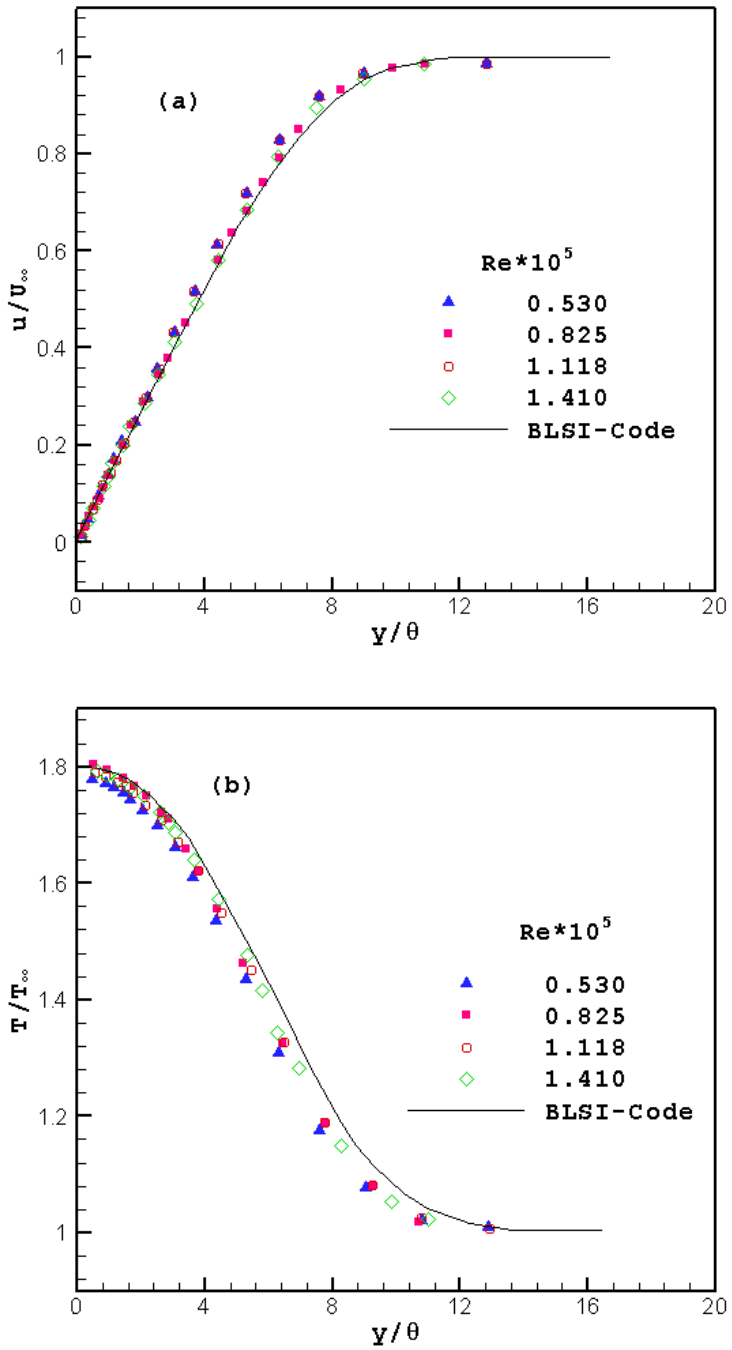


Figure 7: Solutions for the supersonic laminar boundary layer over a flat plate: (a) normalized u-velocity profiles; (b) normalized temperature profiles.

D.Shock Boundary Layer Interaction

The main features of the oblique shock wave and a laminar boundary layer interaction are well understood. Since this configuration involves shocks and separated flow, and carries no uncertainty associated with turbulence modeling, it is chosen here as a test case. The numerical solutions at steady state are compared with experimental data reported in Ref. (11).

The flow parameters are such that $M_\infty = 2.2$, $Re_{sh} = u_\infty X_{sh}/\nu = 9.8645 \times 10^4$, where $X_{sh} = 0.08$m is the distance between the leading edge and the shock impingement point in the experiment and the incident shock angle $\beta=29°$. The total number of grid points used is 74×62, and they are distributed similar to the compressible boundary layer case and is shown below;

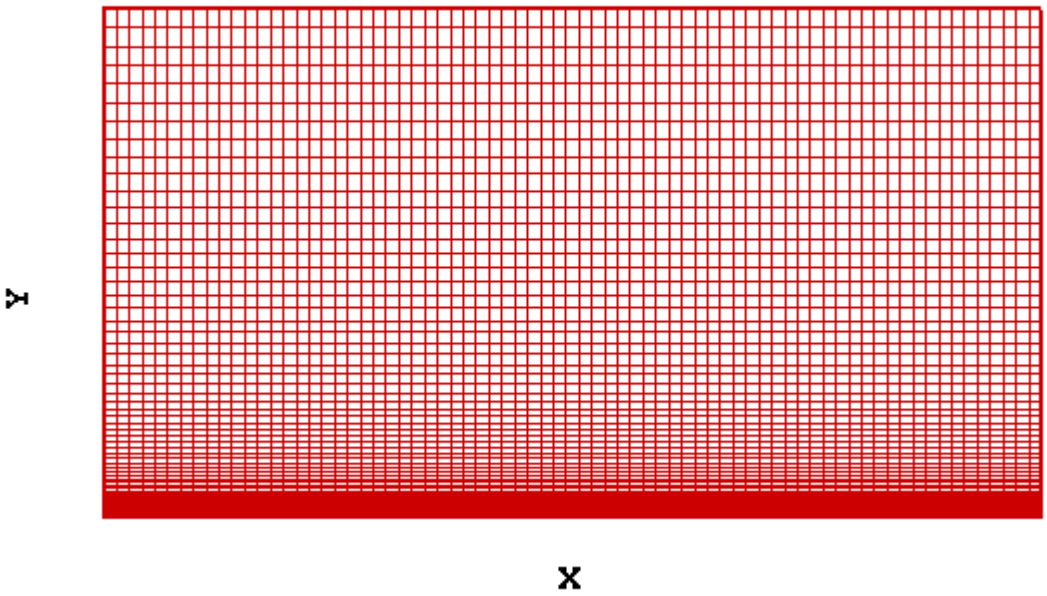


Figure 8: Non-uniform grid used to study Compressible Boundary Layer and the Shock-Boundary Layer Interaction problem.

In order to study the Shock-Boundary Layer Interaction, a steady-state boundary layer flow is established as in the compressible boundary layer solution. Then an oblique shock wave is imposed at the upper left corner of the inflow boundary and along the top boundary, where the variables are consistently overspecified using jump relations. This solution is left to evolve in time until a steady-state solution is reached.

In order to compare the results with Ref. (11) the wall pressure and the velocity distribution is considered. The pressure was normalized by P_0 i.e. the minimum pressure just upstream of the interaction and the result is presented in Fig. 9. The velocity distribution was considered at the point of separation, i.e. $X/X_{sh} = 1.0$ and the local velocity was normalized using the free stream velocity U_∞ and the result is presented in Fig. 10. From the two plots it can be concluded that the pressure and velocity distribution are comparable with the experimental results. A mesh refinement was considered to see if the solution converges and is also presented in Fig. 9. Finally a contour plot of pressure and Mach number is presented in Fig. 10 and Fig. 11 respectively, to illustrate the flow field during a Laminar Shock Boundary Layer interaction. It must be noted that the experimental results presented in Ref. (11) were not completely accurate due to the uncertainty associated with the seed particles as mention in the paper.

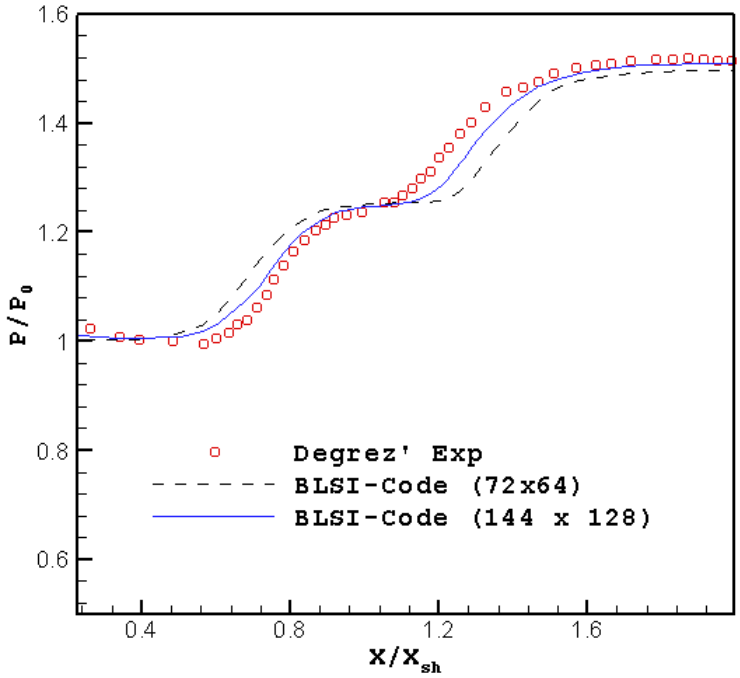


Figure 9: Shock-Boundary Layer Interaction, pressure plot.

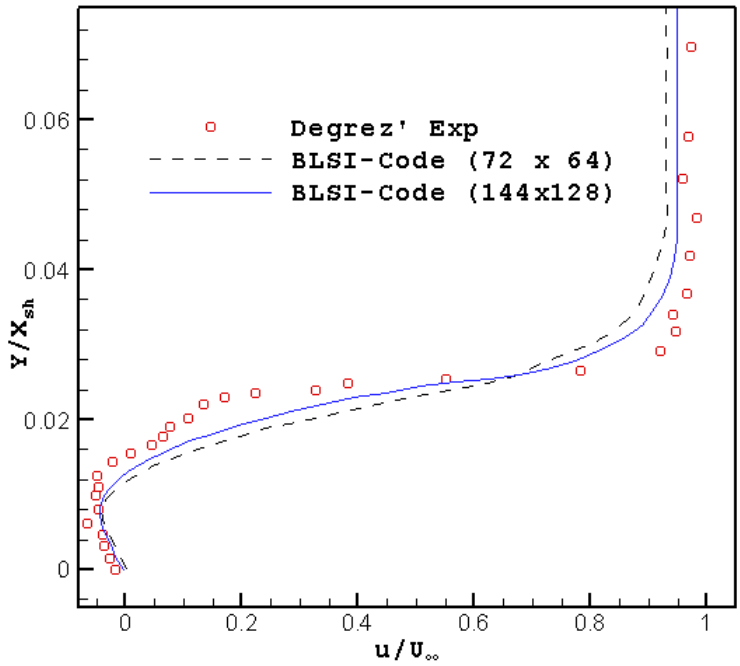


Figure 10: Shock Boundary Layer Interaction, velocity profile at the separation point X/X_{sh} = 1.00.

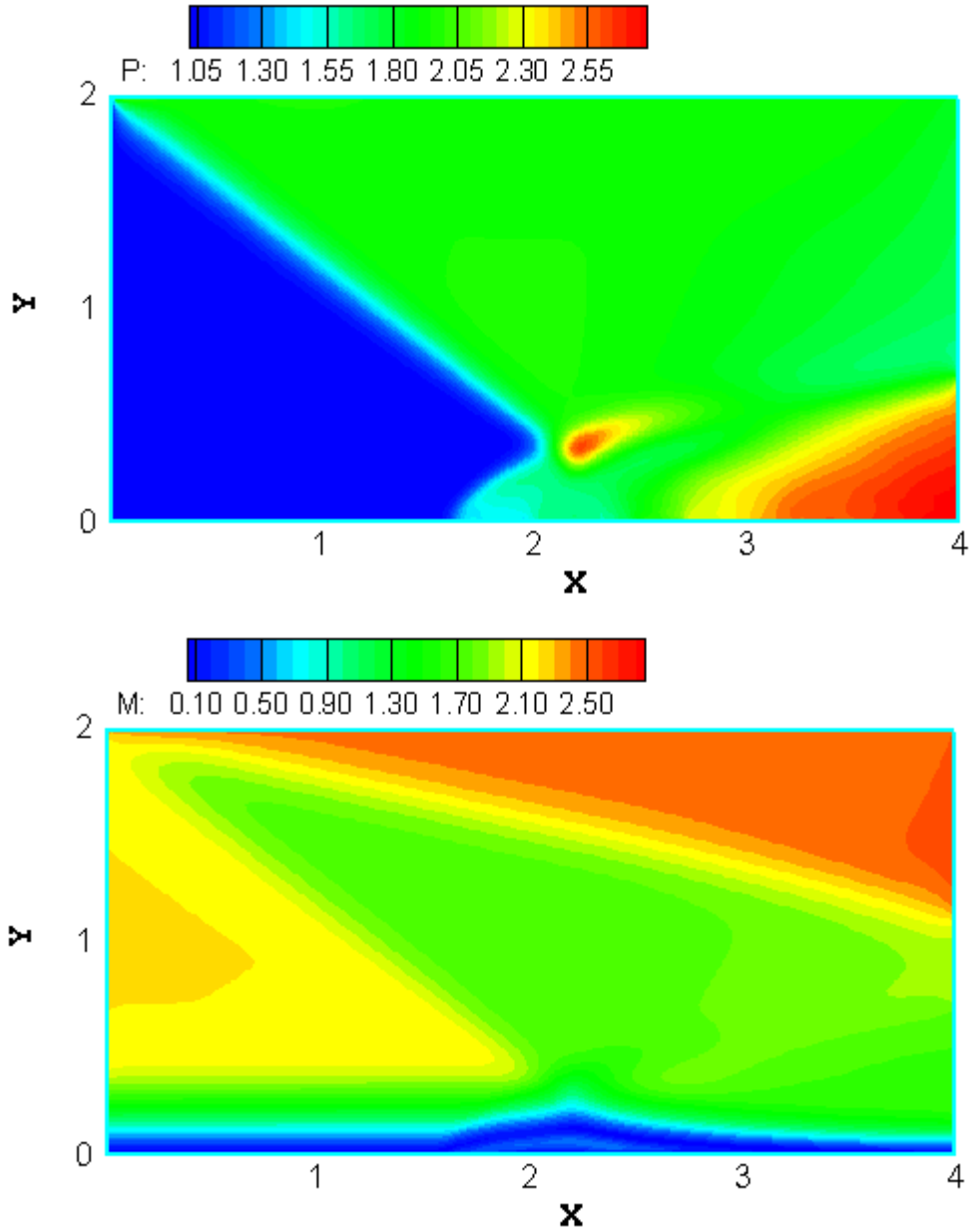


Figure 11: Shock Boundary Layer Interaction; a) Pressure Contour, b) Mach Number Contour.

VI. Conclusion

Under the present effort a parallel compressible Navier-Stokes solver has been successfully developed and validated. This code can now be used to incorporate turbulence modeling for further applications in studying Shock-Boundary Layer Interactions. The several cases studied during the course of this project gives ample confidence in the laminar code. The Laminar Shock-Boundary Layer Interaction gave a good comparison with experimental results and thus gave a good insight into this classic problem.

References

1. **Kaenel, R. von, Kleiser, L., Adams, N. A. and Vos, J. B.** *Large-Eddy Simulation of Shock-Turbulence Interaction.*, AIAA, Vol. 42, p. 2516, 2004.

2. **Tannehill, J. C., Anderson, D. A. and Pletcher, R.H.** *Computational Fluid Mechanics and Heat Transfer.* U.S.A : Taylor & Francis, 1997.

3. **Leer, Bram van.** *Diffraction of a shock in a channel bend: A 2-D Euler code from 1-D building blocks.* Advanced CFD Course Notes. Ann Arbor, MI : Winter 2009.

4. **Gudonov, S.K.** *A Finite Difference Method for the Computation of Discontinuous Solutions of the Equation of Fluid Dynamics*, Mat. Sb. , pp. 357-393, 1959.

5. **Roe, P.L.** *Approximate Riemann solvers, parameter vectors, and difference schemes.* J. Comput. Phys., Vol. 43, p. 357., 1981.

6. **Hirsch, C.** *Numerical Computation of Internal & External Flow* : John Wiley & Sons, Ltd., 2007.

7. **Leer, Bram van.** Computer Problem, Writing a Riemann Solver, AE 520 - Compressible flow Notes. Ann Arbor, MI, USA : Fall 2008.

8. **Toro, E.F.** *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction.* Berlin, Germany : Springer, 1999.

9. **Kim, H. D. and Liu, N. S.** *A Time-Accurate High-Resolution TVD Scheme for solving the Navier-Stokes Equations.* Great Britain : Computers Fluids, Vol. 22, p. 517, 1993.

10. **Schlichting, H. and Gersten, K.** *Boundary Layer Theory.* 8th Ed. Germany : Springer, 2000.

11. **Leer, Bram van.** *Building your own Shock Tube.* Advanced CFD Course Notes. Ann Arbor, MI : Winter 2009.

12. **Degrez, G., Boecadoro, C. H. and Wendt, J. F.** *The interaction of an oblique shock wave with a laminar boundary layer revisited. An experimental and numerical study.* J. Fluid Mech., Vol. 177, p. 247, 1987.