University of California
Los Angeles

Final Project

2D Supersonic Jet Inlet

Author:
David Vasko

Professor:
Xiaolin Zhong

March 18, 2015

Contents

List of Figures

1 Abstract

The purpose of this report is to develop a fast and accurate two dimensional computational fluid dynamic solver utilizing MacCormacks method with artificial dissipation. The final product will be demonstrated by solving for the oblique shocks that form in a supersonic jet inlet. Accuracy will be tested by comparing the numerical results to the exact solution, which was solved for using analytical formulas presented in the following section. Speed will be shown by comparing the time complexity of my algorithm to that of a baseline solution which relies on nested loops. My algorithm is highly adjustable and solving parameters such as grid resolution and artificial dissipation can be easily set by the user. Speed is one of the most important traits of any analytical software because it enables companies to iterate designs quickly in order to find an optimal product. In this case, my algorithm could be used to appropriately dimension a supersonic jet inlet.

2 Problem Statement

When a aircraft is going faster than Mach 1, the supersonic air must be slowed down to subsonic speeds, relative to the jet engine, in order for the engine to function properly. Jet engines cannot operate in supersonic flows, because supersonic flows would result in shocks on the compressor blades. Such shocks would cause widely varying pressures and high drags, which would destroy efficiency and possibly damage the engine. Furthermore, the air needs to be slowed down in order for the fuel in the combustion chamber to be used completely. In a supersonic jet inlet, air that enters the inlet at high Mach numbers must be reduced through oblique shocks before continuing on into the jet engine itself. The following grid represents the basic shape of the jet inlet to be analyzed. The top and bottom rows of the grid represent physical boundaries, and the flow is constrained on these two sides.The superimposed blue lines represent the shocks that will be solved for

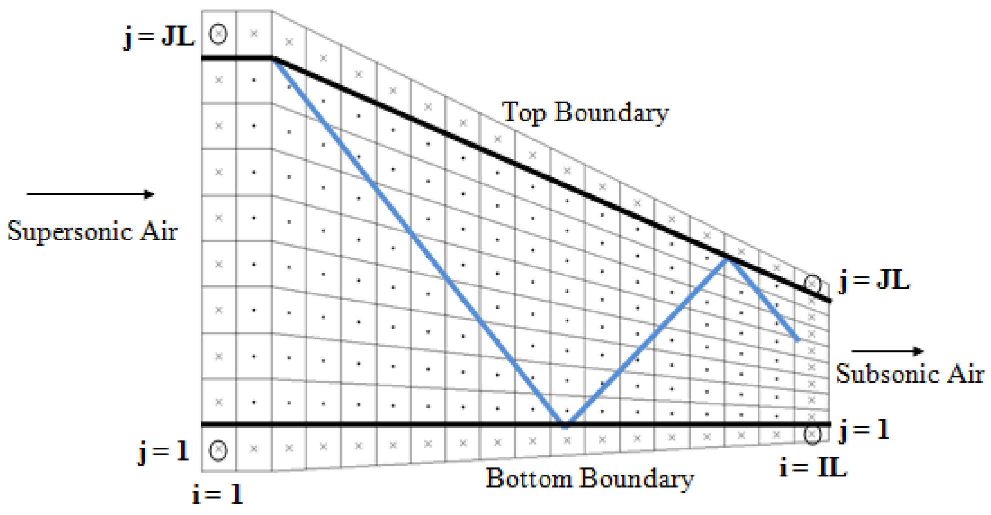


Figure 1: Simple grid showing general shape of the problem. Incoming air enters on the left at prescribed conditions. The numerical grid has JL rows and IL columns where the bottom left node is given index (1,1) and the top right node is given index (IL,JL). Tentative shocks are shown in blue

The incoming air will enter from the left side of the figure, and leave from the right side. The numerical grid has JL rows and IL columns, where the bottom left node is given index (1,1) and the top right node is given index (IL,JL). The properties of the incoming air are prescribed by known flow conditions and will be constant throughout this analysis

$$P = 10^5 N/m^2$$
$$\rho = 1 kg/m^3$$
$$M = 2.9$$
$$\theta = 10.95^\circ$$

Where P is the pressure, ρ is the density, and M is the Mach number. The vertical component of velocity, v, is zero here. The fourth parameter θ is the steepness of the perturbation measured from the horizontal. The purpose of analyzing this problem is to find out the steady state behavior of the supersonic jet inlet. The boundary and initial conditions will be simulated numerically and will be discussed in Numerical Methods.

3 Theory

The theory needed to understand the analytical solution to the supersonic jet inlet includes the normal shock relations and the θ, β, M relation.

3.1 Normal Shock Relations

The normal shock relations are analytical formulas that tell you exactly how a given parameter such a pressure, Mach number or density will change across a normal shock. Written out these equations are

$$\frac{P_2}{P_1} = \frac{2\gamma {M_1}^2 - (\gamma - 1)}{\gamma - 1}$$
$$\frac{\rho_2}{\rho_1} = \frac{(\gamma + 1){M_1}^2}{(\gamma - 1){M_1}^2 + 2}$$
$${M_2}^2 = \frac{(\gamma - 1){M_1}^2 + 2}{2\gamma {M_1}^2 - (\gamma - 1)}$$

Where the subscript two represents parameters on the upstream side of the shock wave and the subscript 2 represents downstream parameters. The parameter γ is the specific heat ratio, and since we are dealing with air, treated as an ideal gas,

$$\gamma = 1.4$$

To use the normal shock relations you have to use the geometry of the problem to find the components of of the flow that are perpendicular to the shock itself. These do not work for arbitrary shocks, you must find the angle of the shock so you can calculate normal and tangential parts of the Mach number.

3.2 θ, β, M relation

In order to solve for the angle of the shock (β), measured from the horizontal, you have to use the θ, β, M relation. This is a fairly complicated relationship and in practice one often uses tables to look up values. The θ, β, M relation shown analytically is

$$\tan(\theta) = 2\cot(\beta)\frac{{M_1}^2 \sin^2\beta - 1}{{M_1}^2(\gamma + \cot(2\beta)) + 2}$$

3.3 Newton-Rhapson

To implement this relation in my code i used the Newton-Rhapson method. The graph of this relationship is very smooth as it approaches the $\theta = 0$ axis very nicely which makes it a good candidate for Newton-Rhapson. The Newton-Rhapson method of solving for zeros of an equation is described as follows.

1. Define a function $f(\beta)$ such that it equals zero. In the θ, β, M relation this would be

$$f(\beta) = 0 = 2\cot(\beta)\frac{{M_1}^2 \sin^2\beta - 1}{{M_1}^2(\gamma + \cot(2\beta)) + 2} - \tan(\theta)$$

2. Compute the derivative of $f(x)$ with respect to beta. Since this is very complicated in the θ, β, M relationship, I used a three point central difference scheme to approximate the derivative.
$$f'(\beta) \approx \frac{f(\beta+h) - f(\beta-h)}{2h}$$
This difference scheme will produce a second order error,and will an appropriately chosen h this approximation will close enough to the analytical solution for our purposes.

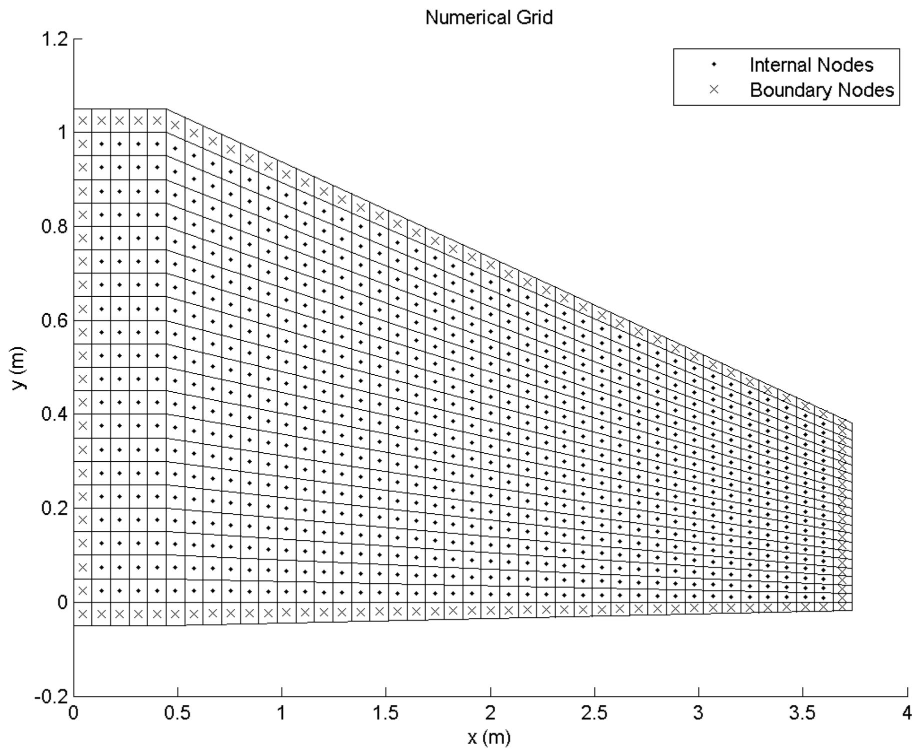
3. Use the Newton update equation to iteratively solve for the zeros of the function. By setting a tolerance for convergence you can get within machine precision of the true value. The Newton update equation is shown below.
$$\beta_{n+1} = \beta_n - \frac{f(\beta_n)}{f'(\beta_n)}$$
Where the subscript n just represents what incremental step it is in.

Once β is found using this method the normal components of Mach number can be found and the normal shock relations can be used. This whole process is done across the whole length of the supersonic jet inlet to propagate the oblique shocks down the length.

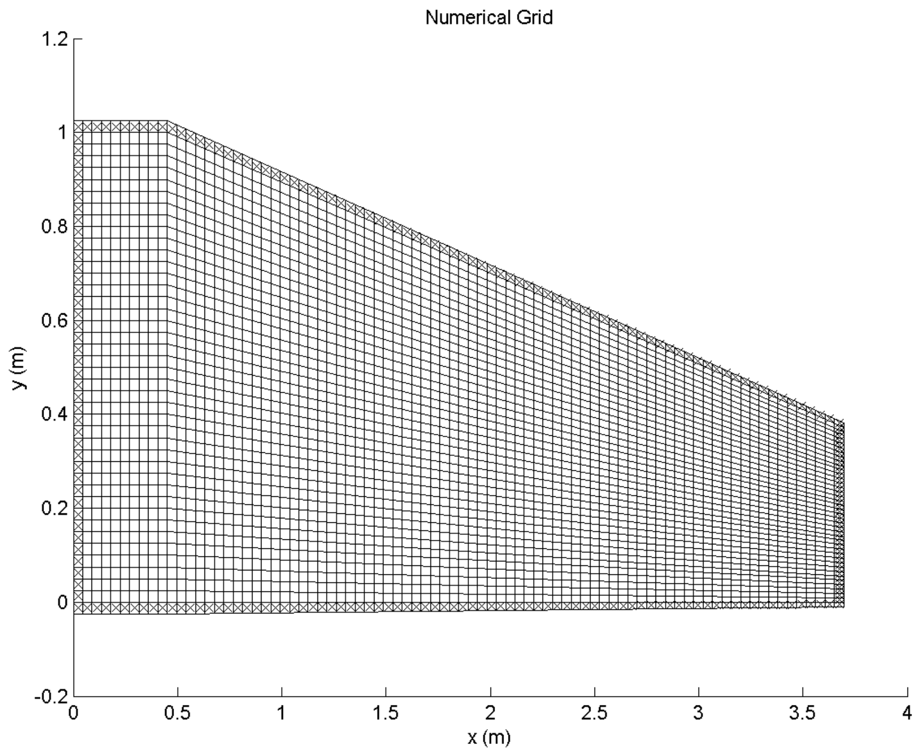
4 Numerical Methods

4.1 Grid Generation

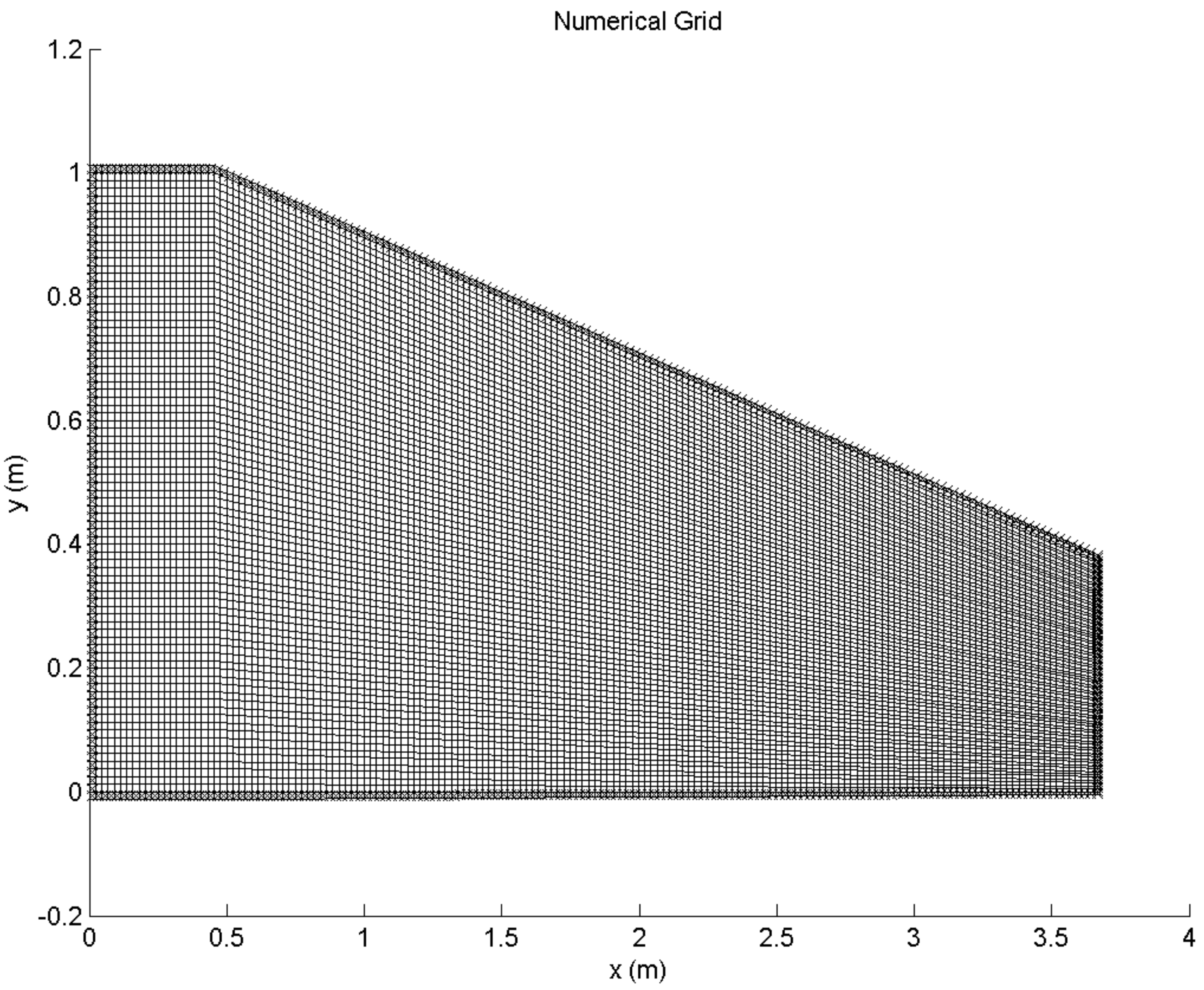
The grids used for this simulation are all algebraic. The computational domain is divided into multiple columns and rows where the number depends on the grid resolution. The width(dx) for each cell is constant throughout the domain, however the height(dy) of each cell decreases as the grid moves in the x direction. This decrease is simply because of the perturbation that angles downward from the top of the inlet. An important note on grid generation is that two separate grids are generated. The first set of these grids represents the discrete nodes of which we divided the fluid into. The second set represents the boundaries for each node. So the value for each nodes is representative of that whole area or finite volume. In finite volume analysis we are interested in looking at fluxes across boundaries, therefore we need both the values at a particular nodes and the boundaries themselves. The following figures show the three different grid resolutions used in this analysis. The figures will begin on the subsequent page.



(a) Original grid with 42 nodes in the x-direction and 22 nodes in the y-direction. Internal nods are shown with a dot in the middle while boundary nodes are shown with an x.



(b) Doubling the number of elements in both the x and y directions. The result is a grid with four times the number of elements. The dots in the internal nodes are dropped to give a better visual of what the grid looks like.



(c) Quadrupling the number of elements in both the x and y directions with respect to the original grid. The result is a grid with sixteen times the number of elements. The dots in the internal nodes are dropped to give a better visual of what the grid looks like.

Figure 2: Sub figures (a), (b), and (c) show the grids with in order of increasing resolution. The 4x grid has sixteen times the number of grid points as the original grid. Using such a fine meshing will give a good sense of what the accuracy of the algorithm is

4.2 Finite Volume Formulation

In the finite volume formulation each node is associated with the area encapsulated by the adjacent borders. This makes up a representative unit or cell that has constant properties across it, and fluxes across its four borders. Each cell has several important geometric properties. Each cell an area (2D equivalent to volume), four edge lengths (2D equivalent of surface area) and four normal vectors which by definition point outward from the center of the cell. The choice of normal definition has consequences on the signs of the MacCormack predictor and corrector. The following figure gives a visual for what a cell is and the geometric properties it possesses.

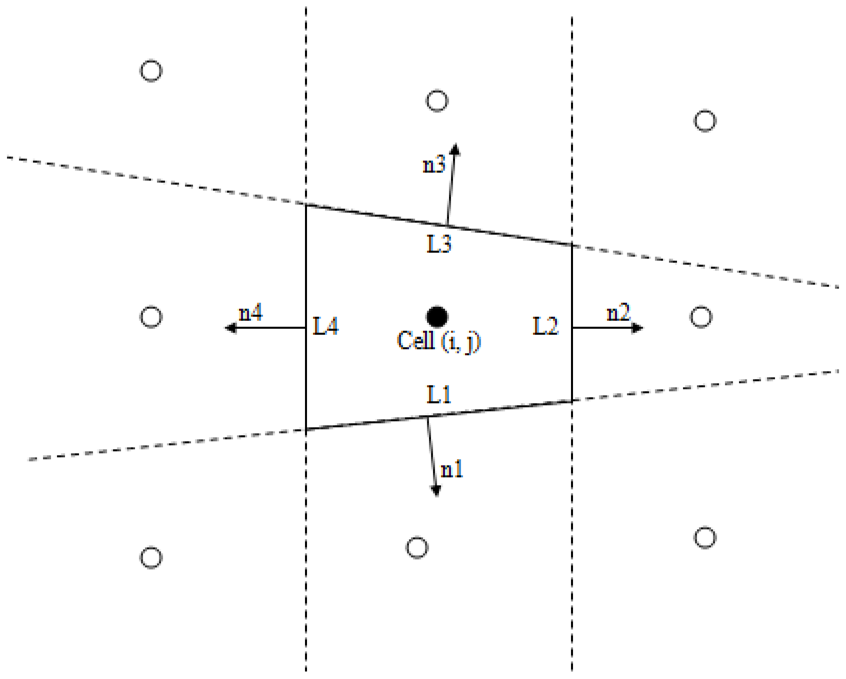


Figure 3: This figure visually shows the definitions of the geometry used. Area is not shown explicitly on this figure however, it is simply the area enclosed by the four solid black lines.

These geometric properties are calculated based on the x and y coordinates of the four corners of the cell. Since the grid is not moving over time these properties can be calculated once and used for the whole simulation.

4.3 Governing Equations

The finite volume formulation has a simple physical meaning. Each cell has a particular value or ”state” at any given time step which is uniform across the entire cell. Between time steps there is a transfer of this quantity across the boundaries between adjacent cells. This is called the flux and if you account for the flux in every direction you can tell how a a cell is going to change over time. Any given cell is only directly affected by its adjacent neighbors, but ultimately the steady state of each cell is dictated by the boundary conditions of the problem. In physical problems the quantities we are interested in are mass, momentum and energy. Mass, momentum and energy must be conserved and by conserving them at a nodal scale with state vectors and flux vectors, they will also be conserved on a global scale. The state vector $\vec{U}$ is given as

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}$$

And the two flux vectors $\vec{E}$ and $\vec{F}$ corresponding to the x and y directions respectively are

$$\vec{E} = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ \rho u v \\ (e+P)u \end{bmatrix}, \vec{F} = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + P \\ (e+P)v \end{bmatrix}$$

Where ρ is the density, u is the x velocity, v is the y velocity, e is the volumetric energy and P is the pressure. Pressure and energy are related to the other terms using the ideal gas law which reduces to

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

$$P = \left[e - \frac{\rho(u^2+v^2)}{2}\right](\gamma - 1)$$

Note that the total flux vector, can be expressed as a sum of the two individual flux vectors. This will be used later when calculating the dot products of fluxes and normals.

$$\vec{F_{total}} = \vec{E} + \vec{F}$$

The numerical method used is developed from a coordinate change and discretization of the Navier-Stokes equations in conservation law form. Written explicitly the governing equation for this problem is

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = 0$$

After working through a rigorous coordinate transform from (x,y) into (ξ, η) you can arrive at the strong form of the conservation laws

$$\frac{\partial U'}{\partial t} + \frac{\partial E'}{\partial x} + \frac{\partial F'}{\partial y} = 0$$

$$\begin{cases} U' = UJ \\ E' = J\xi_x E + J\xi_y F \\ F' = J\eta_x E + J\eta_y F \\ J = \frac{\partial(x,y)}{\partial(\xi,\eta)} = X_\xi Y_\eta - X_\eta Y_\xi \end{cases}$$

These equation become much simpler because we are using a stationary algebraic grid. J is simply ratio of areas from the (x,y) frame to the (ξ, η) frame, which is just the area in the physical domain since the transformed domain has area 1 by definition. The partial derivatives also simplify. They become normals in the specified direction scaled by the appropriate edge length. At this point we will move on to introduce the actual numeral method used.

4.4 MacCormack with Artificial Dissipation

The numerical method used in this simulation is MacCormack's Method with artificial dissipation. This method is a second order difference scheme and is essentially a two step variation of the Lax-Wendroff scheme. It utilizes a predictor term which looks at the node one forward in the x-direction (i + 1), and one forward in the y direction (j + 1). It combines this with a corrector term which looks at the node one backward in the x-direction (i - 1) and one backward in the y-direction (j - 1). On top of the standard MacCormack predictor-corrector algorithm, artificial dissipation is applied to reduce high frequency errors that appear at discontinuities. The dissipative term is proportional to the change between one node and the next. Therefore if two adjacent nodes have, dramatically different values (a sharp corner) the dissipative term is greater and has a flattening effect on the high frequency error. However if adjacent nodes are similar the dissipative term will be small and you will see basically the original MacCormack. So The MacCormack method with dissipation takes the best parts form the original MacCormack (simplicity and ease of implementation) but reduces the errors that appear at sharp corners.

4.4.1 Predictor

The equation for the predictor is

$$U_{ij}\overline{^{n+1}} = U_{ij}{}^{n} - \frac{\Delta t}{A_{ij}}\left(\frac{\partial \dot{E}^{*}_{i+\frac{1}{2},j}}{\partial \xi}|L_{i\pm\frac{1}{2},j}| + \frac{\partial \dot{F}^{*}_{i,j+\frac{1}{2}}}{\partial \eta}|L_{i,j\pm\frac{1}{2}}|\right)$$

Where $U_{ij}{}^{n}$ is the state variable at that particular cell and time step, $U_{ij}\overline{^{n+1}}$ is the prediction for the next time step at that particular cell, A_{ij} is the area of the cell (would volume C in 3D), and the partial terms will be expanded below.

$$\frac{\partial \dot{E}^{*}_{i+\frac{1}{2}}}{\partial \xi}|L_{i\pm\frac{1}{2}}| = \left(\dot{E}_{i+1} - D_{i+1}(U_{i+1} - U_{i})\right)|L_2| + \left(\dot{E}_{i} + D_{i}(U_{i} - U_{i-1})\right)|L_4|$$

Note that the j and n subscripts have been dropped entirely from the above equation simply because every term is evaluated at exactly j and n. Also note that by definition $\Delta\xi$ and $\Delta\eta$ are equal to one which means the partial terms become a simple difference in the ξ or η directions. In this equation,$\dot{E}$ is the total flux vector $\vec{F_{total}}$ dotted with the normal on the edge of interest, D is the artificial dissipation term, U is the state vector at a particular point, and L with the subscript represents the edge length (would be surface area S in 3D). My definition for edge length order is given by Figure 3 in the Finite Volume Formation. Further expanding the flux vectors and dissipative terms using the convention in Figure 3 again to express my normals,

$$\begin{aligned}
\dot{E}_{i+1} &= \vec{E_{i+1}} n_{2x} + \vec{F_{i+1}} n_{2y} \\
\dot{E}_{i} &= \vec{E}_{i} n_{4x} + \vec{F}_{i} n_{4y} \\
D_{i+1} &= \epsilon(\vec{u}_i n_{2x} + \vec{v}_i n_{2y} + c_i)\frac{|P_{i+1} - 2P_i + P_{i-1}|}{P_{i+1} + 2P_i + P_{i-1}} \\
D_{i} &= \epsilon(\vec{u}_i n_{4x} + \vec{v}_i n_{4y} + c_i)\frac{|P_{i} - 2P_{i-1} + P_{i-2}|}{P_{i} + 2P_{i-1} + P_{i-2}}
\end{aligned}$$

Where the only new terms introduced are c, which is the speed of sound as normal, and ϵ which is coefficient for artificial dissipation. Now moving on to the j direction

$$\frac{\partial \dot{F}^*_{j+\frac{1}{2}}}{\partial \xi}|L_{j\pm\frac{1}{2}}| = \left(\dot{F}_{j+1} - D_{j+1}(U_{j+1} - U_j)\right)|L_3| + \left(\dot{F}_j + D_j(U_j - U_{j-1})\right)|L_1|$$

Note that the i and n subscripts have been dropped entirely from the above equation simply because every term is evaluated at exactly i and n. Further expanding the flux vectors and dissipative terms using the convention in Figure 3 again to express my normals,

$$\begin{aligned}
\dot{F}_{j+1} &= \vec{E_{j+1}} n_{3x} + \vec{F_{j+1}} n_{3y} \\
\dot{F}_j &= \vec{E_j} n_{1x} + \vec{F_j} n_{1y} \\
D_{j+1} &= \epsilon(\vec{u}_j n_{3x} + \vec{v}_j n_{3y} + c_j)\frac{|P_{j+1} - 2P_j + P_{j-1}|}{P_{j+1} + 2P_j + P_{j-1}} \\
D_j &= \epsilon(\vec{u}_j n_{1x} + \vec{v}_j n_{1y} + c_j)\frac{|P_j - 2P_{j-1} + P_{j-2}|}{P_j + 2P_{j-1} + P_{j-2}}
\end{aligned}$$

Which concludes the formulation of the predictor. Combining all the above equation represents the full predictor in all of its glory.

4.4.2 Corrector

The equation for the corrector is

$$U_{ij}{}^{n+1} = \frac{1}{2}\left[U_{ij}{}^{n} + U_{ij}{}^{\overline{n+1}} - \frac{\Delta t}{A_{ij}}\left(\frac{\partial \dot{E}^{**}_{i-\frac{1}{2},j}}{\partial \xi}|L_{i\pm\frac{1}{2},j}| + \frac{\partial \dot{F}^{**}_{i,j-\frac{1}{2}}}{\partial \eta}|L_{i,j\pm\frac{1}{2}}|\right)\right]$$

Where $U_{ij}{}^n$ is the state variable at that particular cell and time step, $U_{ij}{}^{\overline{n+1}}$ is the prediction for the next time step at that particular cell, $U_{ij}{}^{n+1}$ is the actual value of that cell at the next time step, A_{ij} is the area of the cell, and the partial terms will be expanded below.

$$\frac{\partial \dot{E}^{**}_{i-\frac{1}{2}}}{\partial \xi}|L_{i\pm\frac{1}{2}}| = \left(\dot{E}_i - D_i(U_{i+1} - U_i)\right)|L_2| + \left(\dot{E}_{i-1} + D_{i-1}(U_i - U_{i-1})\right)|L_4|$$

Note that the j and n subscripts have been dropped entirely from the above equation simply because every term is evaluated at exactly j and n. Also note all the other conventions including normals used in the previous formulation for the predictor apply here as well.

$$\begin{aligned}
\dot{E}_i &= \vec{E_i} n_{2x} + \vec{F_i} n_{2y} \\
\dot{E}_{i-1} &= \vec{E_{i-1}} n_{4x} + \vec{F_{i-1}} n_{4y} \\
D_i &= \epsilon(\vec{u}_i n_{2x} + \vec{v}_i n_{2y} + c_i)\frac{|P_i - 2P_{i-1} + P_{i-2}|}{P_i + 2P_{i-1} + P_{i-2}} \\
D_{i-1} &= \epsilon(\vec{u}_i n_{4x} + \vec{v}_i n_{4y} + c_i)\frac{|P_{i-1} - 2P_{i-2} + P_{i-3}|}{P_{i-1} + 2P_{i-2} + P_{i-3}}
\end{aligned}$$

Moving on to the j direction

$$\frac{\partial \dot{F}^*_{j-\frac{1}{2}}}{\partial \xi}|L_{j\pm\frac{1}{2}}| = \left(\dot{F}_j - D_j(U_{j+1} - U_j)\right)|L_3| + \left(\dot{F}_{j-1} + D_{j-1}(U_j - U_{j-1})\right)|L_1|$$

Note that the i and n subscripts have been dropped entirely from the above equation simply because every term is evaluated at exactly i and n. Further expanding the flux vectors and dissipative terms

$$\begin{aligned}
\dot{F}_j &= \vec{E}_j n_{3x} + \vec{F}_j n_{3y} \\
\dot{F}_{j-1} &= \vec{E_{j-1}} n_{1x} + \vec{F_{j-1}} n_{1y} \\
D_j &= \epsilon(\vec{u}_j n_{3x} + \vec{v}_j n_{3y} + c_j)\frac{|P_j - 2P_{j-1} + P_{j-2}|}{P_j + 2P_{j-1} + P_{j-2}} \\
D_{j-1} &= \epsilon(\vec{u}_j n_{1x} + \vec{v}_j n_{1y} + c_j)\frac{|P_{j-1} - 2P_{j-2} + P_{j-3}|}{P_{j-1} + 2P_{j-2} + P_{j-3}}
\end{aligned}$$

Which concludes the formulation of the corrector. Combining predictor and corrector will give you the MacCormack methods approximation of the state vector one time step in the future. Applying this same formula over and over again until convergence is reached yields numerical solution.

4.5 Boundary Conditions

There are four separate boundary conditions for this problem, the entrance, the exit, the top boundary and the bottom boundary.

Entrance Boundary Condition

This is the simplest boundary condition. Since this computational problem represents a supersonic jet inlet where the incoming flow is assumed to be constant, the boundary condition will be constant as well. Every time step of the solution you simply make sure that the far left nodes are all equal to the prescribed flow conditions described in the Problem Statement. Those conditions are

$$\begin{aligned}
P &= 10^5 N/m^2 \\
\rho &= 1 kg/m^3 \\
M &= 2.9 \\
v &= 0
\end{aligned}$$

For all cells in the first column (i=1,j) set their state vector equal to this.

$$\vec{U}(i = 1, j) = \vec{U}_{prescribed}$$

where the notation (i = 1 , j) simply means that we are looking at all j rows but only the first column where i is equal to 1.

Exit Boundary Condition

In the physical domain flow that leaves the supersonic jet inlet continues on into the jet engine itself. Since the computational problem is only worried about the inlet we must impose a simple end condition where we want to end computation. This is done using interpolation for the far right column of our computational domain. This means we will not apply MacCormacks method for the far right slave cells. This is a convenient way to impose the exit boundary condition because we do not have to apply a special backward difference scheme for the far right slave cells. Written mathematically this boundary condition looks like

$$\vec{U}(i = IL, j) = 2\vec{U}(i = IL - 1, j) - \vec{U}(i = IL - 2, j)$$

where the notation (i = IL , j) simply means that we are looking at all j rows but only the last column where i is equal to IL. Remember as stated in the problem statement IL is the number of columns in the computational domain.

Top and Bottom Boundaries

These top and bottom boundary conditions are essentially the same thing. You do not want any flux to cross from internal nodes to slave cells. You solve this problem in an interesting way; set all the properties of the state vector, except for normal velocity, in the slave cells equal to their adjacent internal node, then set the normal velocity to point in the opposite direction. The effect this has is that the velocity flux is "reflected" back into the internal node while the other state properties remain unchanged. The tricky part of this is converting the Cartesian velocities into a normal an tangential frame. You can do this using trigonometry and the perturbation angle θ.

$$\vec{v}_n = \vec{u}_x \sin(\theta) + \vec{v}_y \cos(\theta)$$
$$\vec{v}_t = \vec{u}_x \cos(\theta) - \vec{v}_y \sin(\theta)$$

Where the subscript n is to reference the normal direction and the subscript t is to reference tangential direction. The subscripts x and y are added to emphasize that the current $\vec{u}$ and $\vec{v}$ are in the Cartesian frame. To set the boundary condition correctly the normal component has to be reversed while the tangent component must remain unchanged. After reversing the normal component the velocity must be transformed back into the Cartesian frame. This is done in a similar manner. All in all the top boundary condition is applied mathematically in the following way

$$\vec{U}(i, JL) = \begin{bmatrix} \rho(i, JL-1) \\ \rho(\vec{v}_t \cos(\theta) - \vec{v}_n \sin(\theta)) \\ -\rho(\vec{v}_t \sin(\theta) + \vec{v}_n \cos(\theta)) \\ e(i, JL-1) \end{bmatrix}$$

Where $\vec{v}_n$ and $\vec{v}_t$ are calculated based on the shown equations using the (i,JL-1) nodes. For the bottom boundary the Cartesian frame is the normal frame so no extra work has to be done. The bottom boundary condition is written as

$$\vec{U}(i, 1) = \begin{bmatrix} \rho(i, 2) \\ \rho u(i, 2) \\ -\rho v(i, 2) \\ e(i, 2) \end{bmatrix}$$

These boundary conditions must be enforced after every single time step as well as between the predictor and corrector.

4.6 Initial Conditions

Any initial condition should ultimately converge to the steady state solution but for ease of implementation the initial condition is chosen as setting every node equal to the prescribed entrance conditions. This essentially has the effect of suddenly adding the perturbation to a previously unperturbed straight supersonic flow. Written mathematically the initial condition is

$$\vec{U}(i, j) = \vec{U}_{prescribed}$$

where the prescribed conditions are

$$\begin{aligned}P &= 10^5 N/m^2\\ \rho &= 1kg/m^3\\ M &= 2.9\\ v &= 0\end{aligned}$$

4.7 Convergence

Steady state is simply defined as the state a system reaches that is no longer changing with respect to time. In a numerical simulator this is checked by seeing how much the solution changes each time step. Numerical steady state is reached when the change between one state and the next becomes small enough to assume it is zero. The Δ used in my algorithm is

$$\Delta = max\left(\frac{|\vec{U}^{n+1}(i,j) - \vec{U}^{n}(i,j)|}{\vec{U}^{n}(i,j)}\right)$$

Once the value of Δ falls below a preset tolerance we conclude that the system has converged. The tolerance can theoretically be set as low as machine precision, but good accuracy can be obtained by setting tolerance equal to 10^-10. Using this convergence criteria the solution converges in as low as 1000 iterations for the 1X grid and as many as 4500 iterations for the 4x grid.

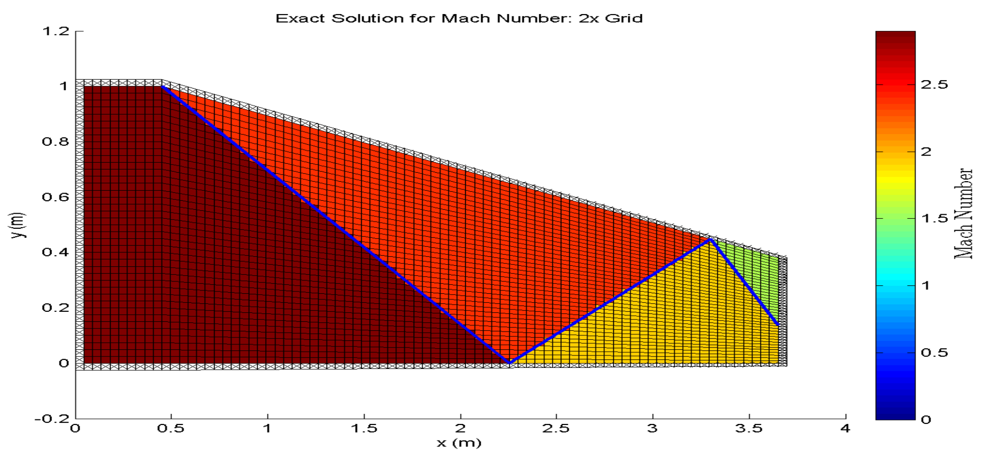
5 Results and Discussions

5.1 Overview

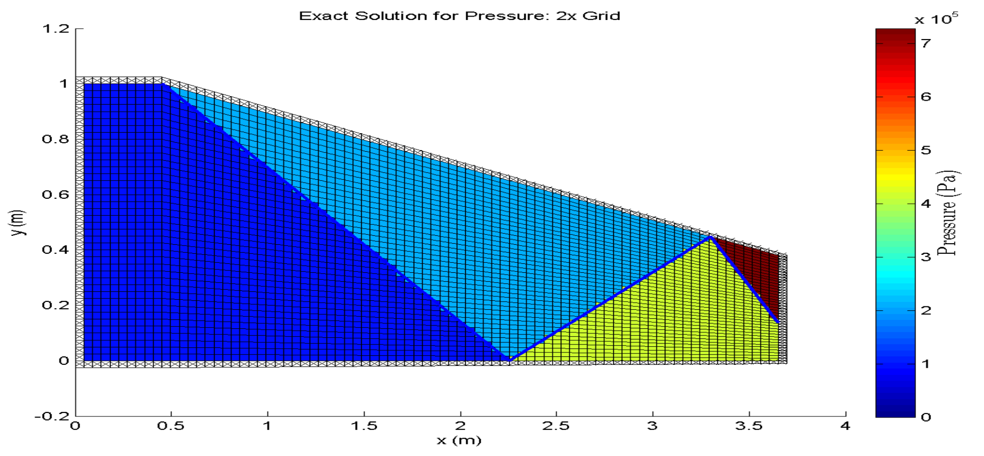
Even after correctly implementing the MacCormack algorithm, in order to optimize its accuracy there are several important steps take. These include determining the optimal dissipation factor ϵ and Courant number. The goal of the numerical solver is to approximate the exact solution as closely as possible, so optimal parameters are determined by iterating over a wide range of those parameters and seeing which results in the least error. The first part of this section will show the exact solution, then explain convergence and show teh convergence history and then will go on to explain how optimal parameters were calculated. After optimal parameters are determined multiple graphs will be presented to showcase the accuracy of my numerical solution as well as the improvement of MacCormacks method with artificial dissipation over the original MacCormack.

5.2 Exact Solution

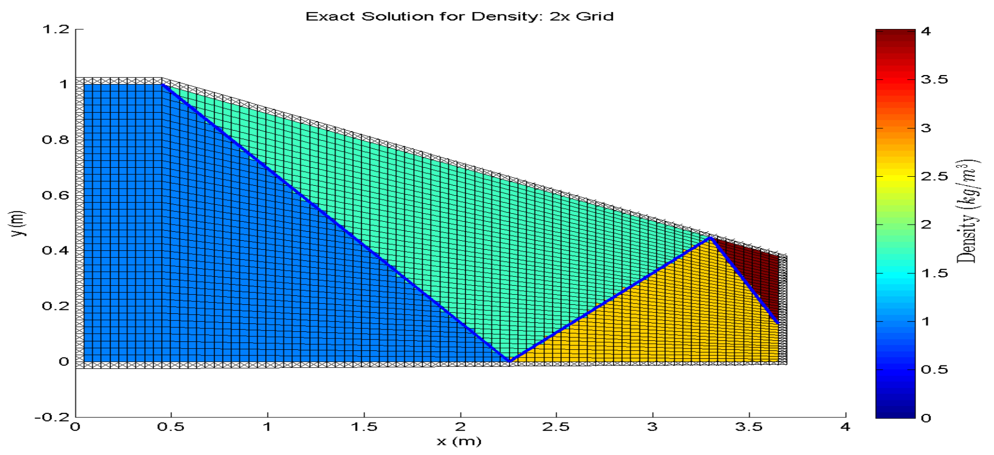
Before delving into the results I obtained it is first important to understand the exact solutions for which all comparison is based. The following figure presents the exact solution for Mach Number, Density and Pressure as color plots. In a color plot, the x and y dimensions show the grid while the third dimension is color. The color represents the magnitude of the quantity being shown and a scale is shown to the right. Exact values will be presented after the graphs. The blue lines superimposed on the figure represent the exact shocks determined based on the analytical formulas expressed in the Theory section.



(a)



(b)



(c)

Figure 4: The exact solution for Mach Number (a), Pressure (b) and Density (c)

There are three total shock that form in the exact solution. The analytically determined shock angles (β) are as follows:

$$\begin{aligned} \beta_1 &= 29.0^\circ \\ \beta_2 &= 34.2^\circ \\ \beta_3 &= 41.6^\circ \end{aligned}$$

The exact numbers for the exact solution are broken down into four sections. Section 1 is the incoming air before it hits the first shock; section two is between shocks 1 and 2; section three is between shocks 2 and 3; and section four is to the right of shock 3.

Section One

$$\begin{aligned} M_1 &= 2.9 \\ P_1 &= 1.0 \times 10^5 Pa \\ \rho_1 &= 1.0\, kg/m^3 \end{aligned}$$

Section Two

$$\begin{aligned} M_2 &= 2.2871 \\ P_2 &= 2.1395 \times 10^5 Pa \\ \rho_2 &= 1.699\, kg/m^3 \end{aligned}$$

Section Three

$$\begin{aligned} M_3 &= 1.9424 \\ P_3 &= 4.1074 \times 10^5 Pa \\ \rho_3 &= 2.6872\, kg/m^3 \end{aligned}$$

Section Four

$$\begin{aligned} M_4 &= 1.5516 \\ P_4 &= 7.2855 \times 10^5 Pa \\ \rho_4 &= 4.0245\, kg/m^3 \end{aligned}$$

5.3 Convergence History

In general the numerical solution should tend toward steady state. Obviously there will be some fluctuations up or down, but eventually the numerical method should reach a essentially constant value. The below figure shows the convergence history of the MacCormack Method with artificial dissipations on the 4x grid. The solution is considered converged when the relative change in energy from one state to the next is less that a specified tolerance of 10^-10.

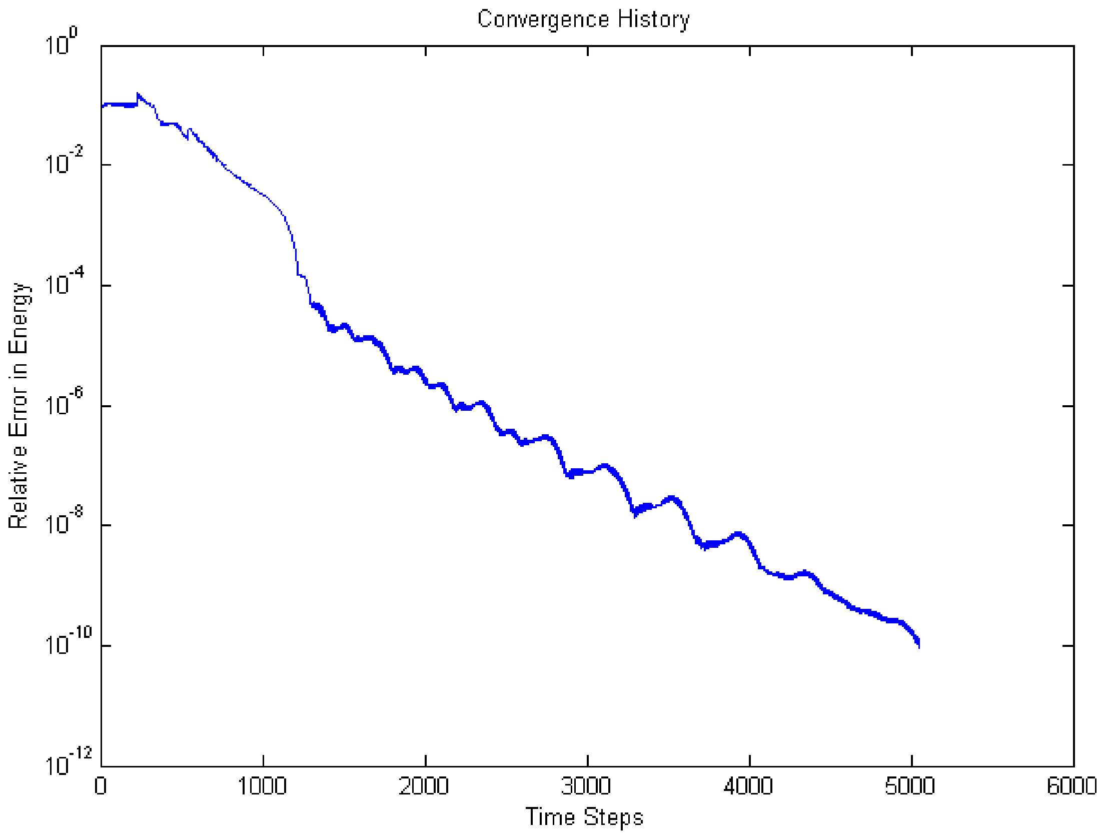


Figure 5: This figure shows the convergence history for a 4x grid with $\epsilon = 0.85$*. Notice that this is a semilog plot because the relative error decreases very rapidly. Also notice that in general the error is deceasing as the time steps go up.*

This figure represents exactly what we would expect for numerical convergence. While there are some iterations that result in a slightly larger error than the previous state, the overall trend is a decreasing error. It decreases very rapidly and once it reaches the tolerance which was set at 10^-10 it is safe to say it has converged.

5.4 Optimal Dissipation

The optimal dissipation was determined by iterating over a wide range of dissipation values, $0.1 \leq \epsilon \leq 3$. For each value of ϵ the bulk error between the exact and numerical solutions were calculated. Bulk error is defined as the sum of all errors for all point between the exact and numerical solutions. I chose to define my bulk error based on Mach number, and thus the following figure shows the bulk error in terms of Mach. This test was done with a grid size of 2x, and a Courant number of 1.

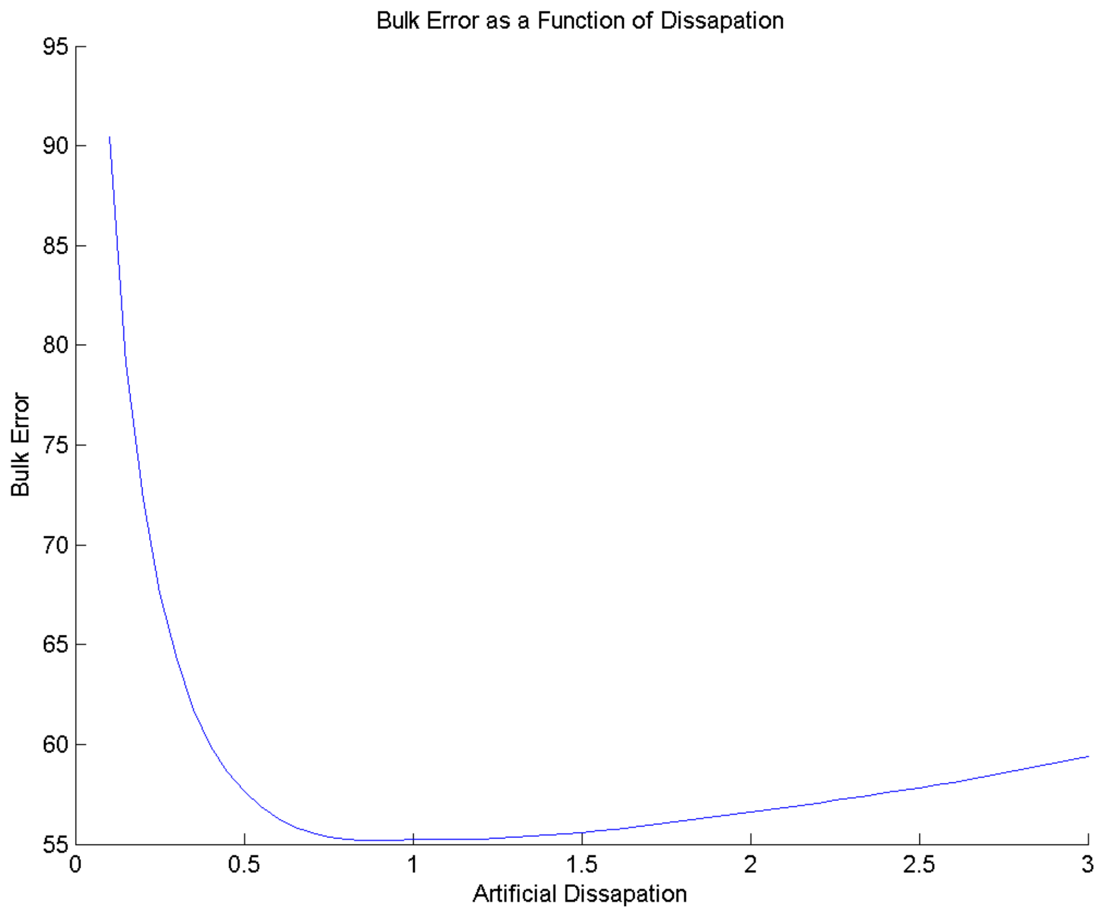


Figure 6: This figure shows that the optimal dissipation was determined by iterating over a wide range of dissipation values. Courant was set to 1 and grid resolution at 2x. There is a clear minimum that occurs at, $\epsilon = 0.85$.

From this figure it is obvious that the optimal dissipation is $\epsilon = 0.85$. While a bulk error in the range of 60 may seem rather large, one must remember that this bulk error is shared across all $JL \times IL$ nodes. In the 2x case, that is 3444 nodes which means each node is on average off by only 0.016 in terms of Mach number. This works out to approximately an average percent error of just over 1 percent per node, which is considered great consistency. Additionally the curve looks as iff it asymptotically approaches the bulk error axis which implies that the error associated with teh zero dissapation case would be enormous. This tells us the the artificial dissipation makes a huge difference in terms of accuracy. Unless otherwise stated assume that this is the dissipation used for the rest of the report.

5.5 Optimal Courant Number

Similar to finding the optimal dissipation, the optimal Courant number was determined by solving for bulk error in Mach number over a wide range of Cournat numbers. The grid size for this simulation was 2x, and the dissipation was set at 0.85. The range of courant numbers was $0.1 \leq v \leq 1.1$. The simulation was run for Courant numbers above 1 to see what would happen if we ventured into the unstable regime. The following figure shows the bulk error of Mach number as a function of Courant.

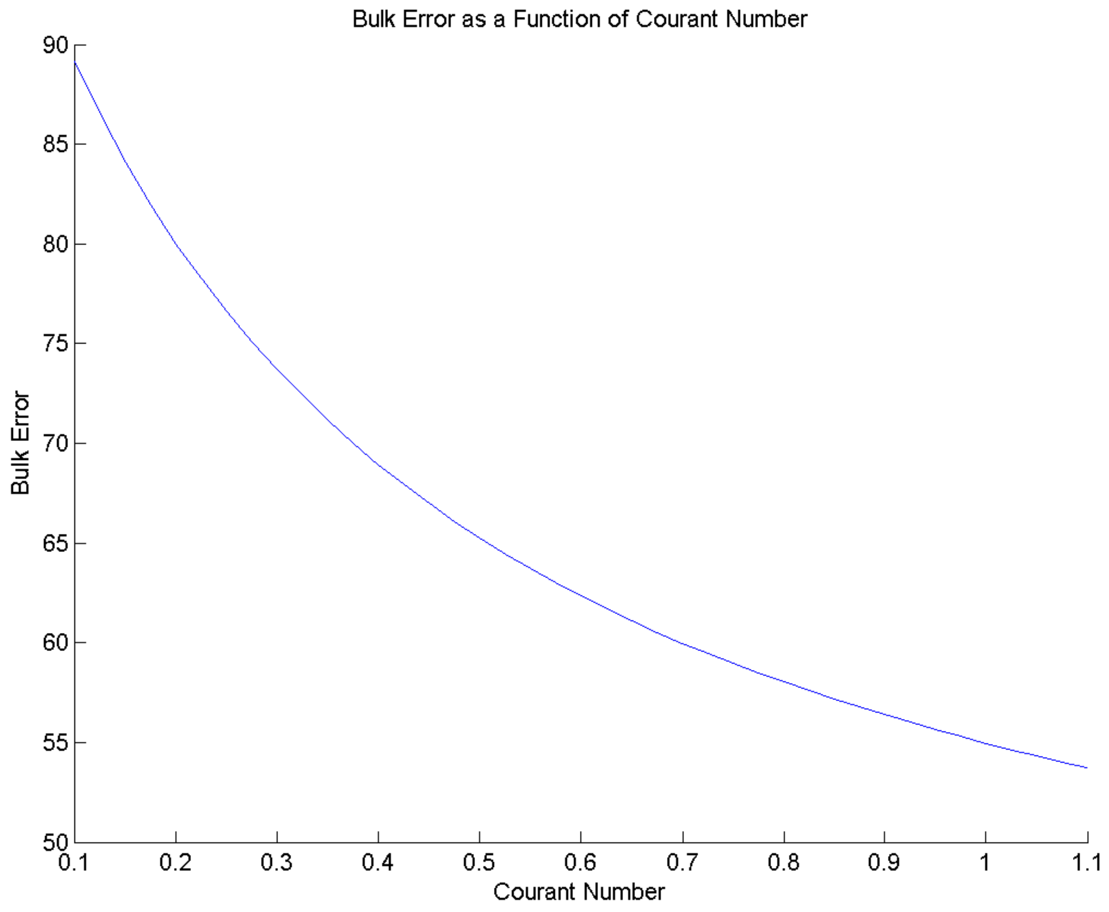


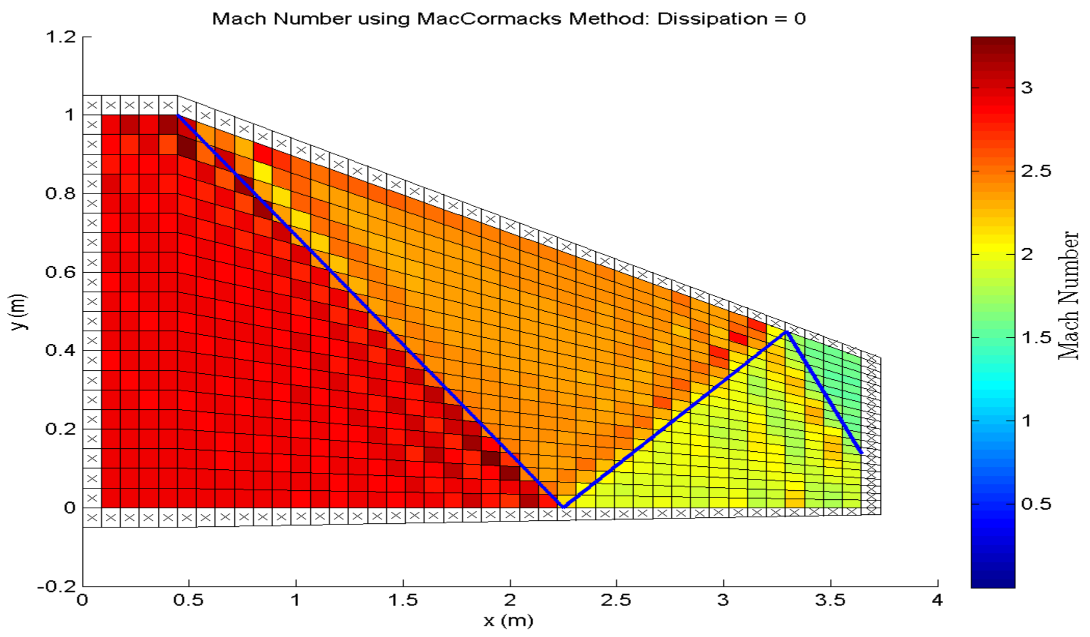
Figure 7: This figure shows that the bulk error decreases as Courant increases. Dissipation was set to 0.35 and grid resolution at 2x.

The numerical method becomes unstable above $v = 1$ but the above figure shows that better accuracy is actually achieved when slightly in the unstable regime. This can be explained by the fact that only a few nodes inside the grid will actually have a Mach number high enough to actually propagate further than 1 node in a single time step. Since most of the nodes will still be stable they compensate for the few instabilities. That being said, the Courant number used for the rest of the report is set to 1. Although $v = 1.1$ technically gave the lowest bulk error, it is not good to be in the unstable regime. Even if the entire solution is not blowing up, there are some nodes that exhibit unstable behavior.

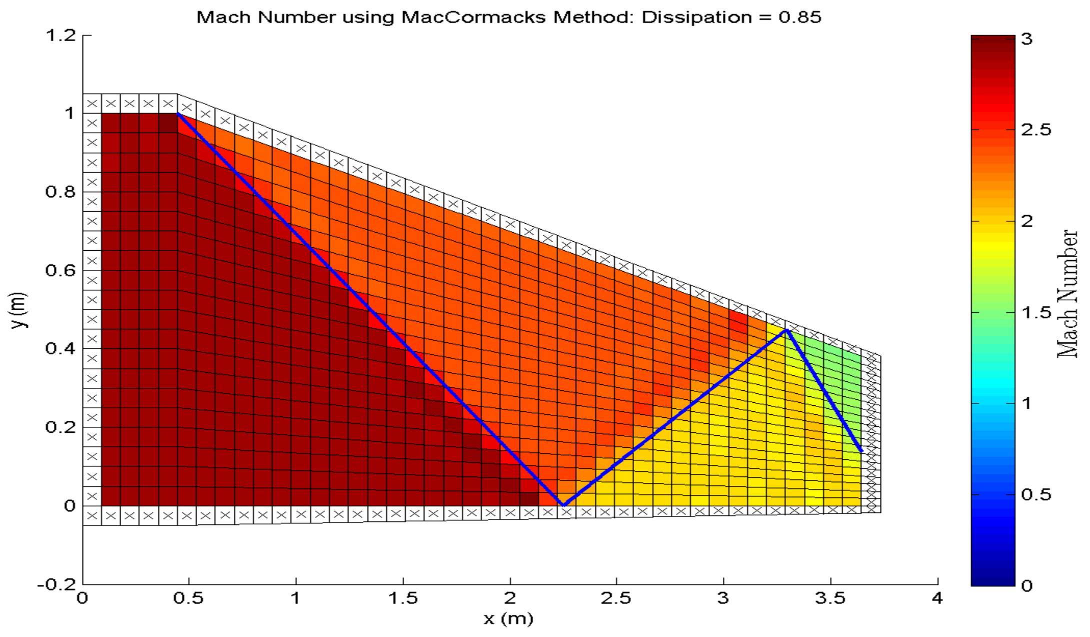
5.6 Zero Dissipation vs Optimal Dissipation

In order to explicitly show how much MacCormack with artificial dissipation improves upon the original MacCormack, a series of graphs at both the standard grid and the 2x grid will be presented. Comparison of color plots will be shown for Mach Number, Pressure and Density. A comparison of these same three quantities as a function of x along one characteristic row will also be shown. The characteristic row chosen is the middle row (j = JL/2). By looking at this row, it will give a good sense of how the high frequency errors look at each shock. In general the MacCormack Method with artificial dissipation is a massive improvement over the original MacCormack. While the high frequency errors that appear as a result of the abrupt change in properties across a shock do not disappear completely, they are dramatically reduced. Additionally, as the grid size gets smaller, dispersion becomes more serious and without dissipation the solution may explode even when using a stable Courant number. This will all be shown more explicitly on the plots which will begin on the next page.

5.6.1 1x Grid

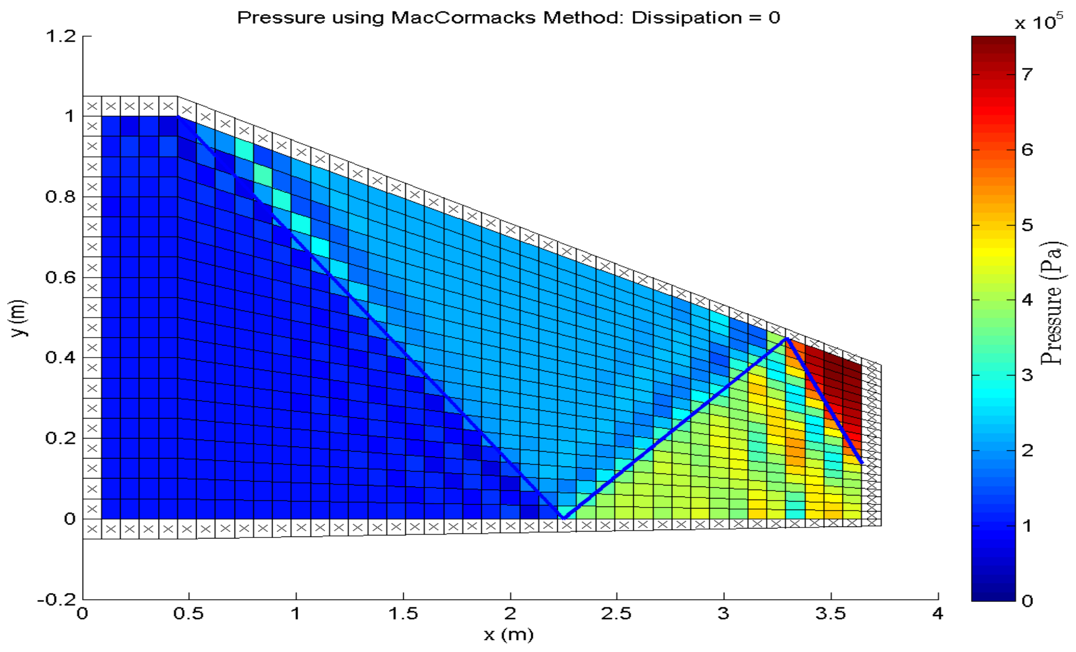


(a)

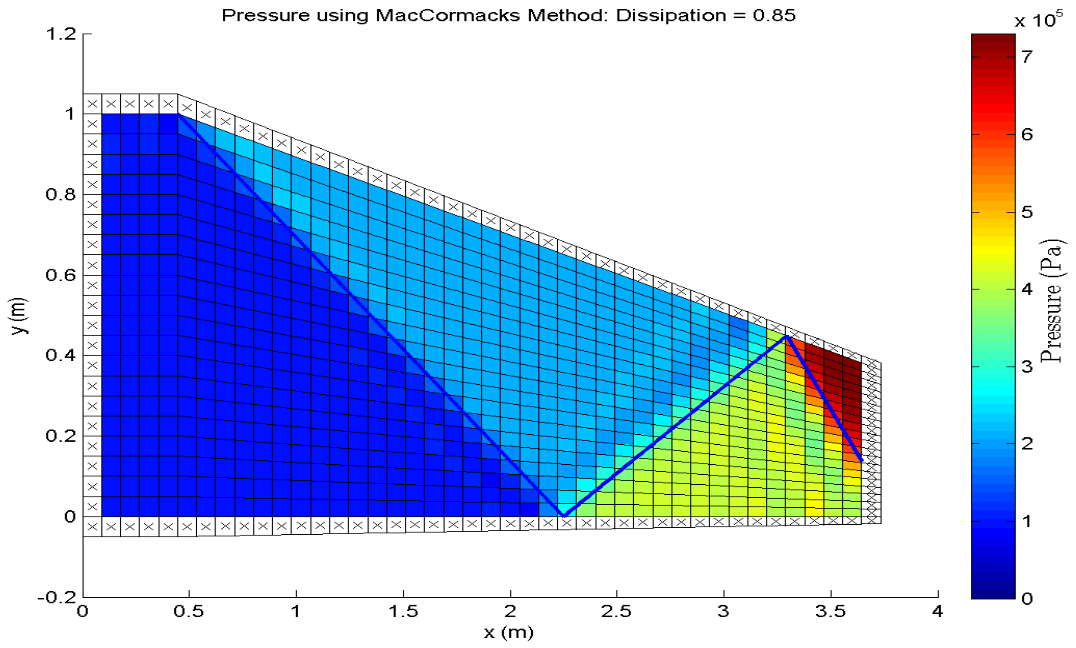


(b)

Figure 8: Figure (a) shows the Mach Number when $\epsilon = 0$. Figure (b) shows the Mach Number when $\epsilon = 0.85$. The exact shocks are superimposed as blue lines. As you can see, both solutions match up with the exact shock, however when there is dissipation the high frequency errors are significantly reduced.

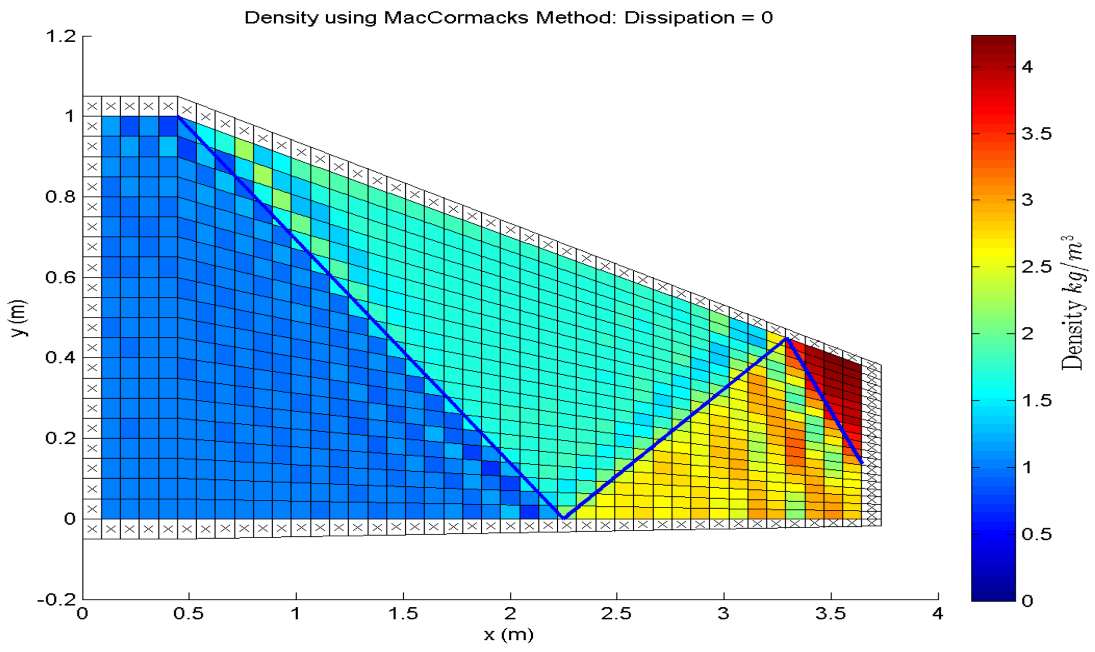


(a)

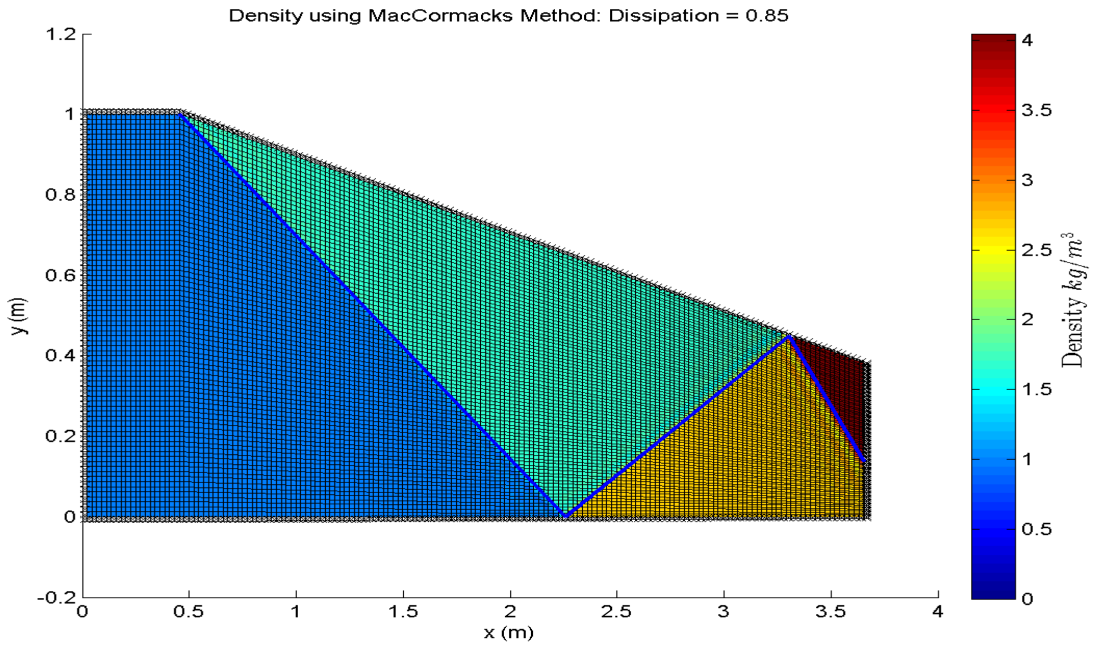


(b)

Figure 9: Figure (a) shows the Pressure when $\epsilon = 0$. Figure (b) shows the Pressure when $\epsilon = 0.85$. The exact shocks are superimposed as blue lines. As you can see, both solutions match up with the exact shock, however when there is dissipation the high frequency errors are significantly reduced. There are still obvious errors even with optimal dissipation, but the amplitude was definitely reduced.

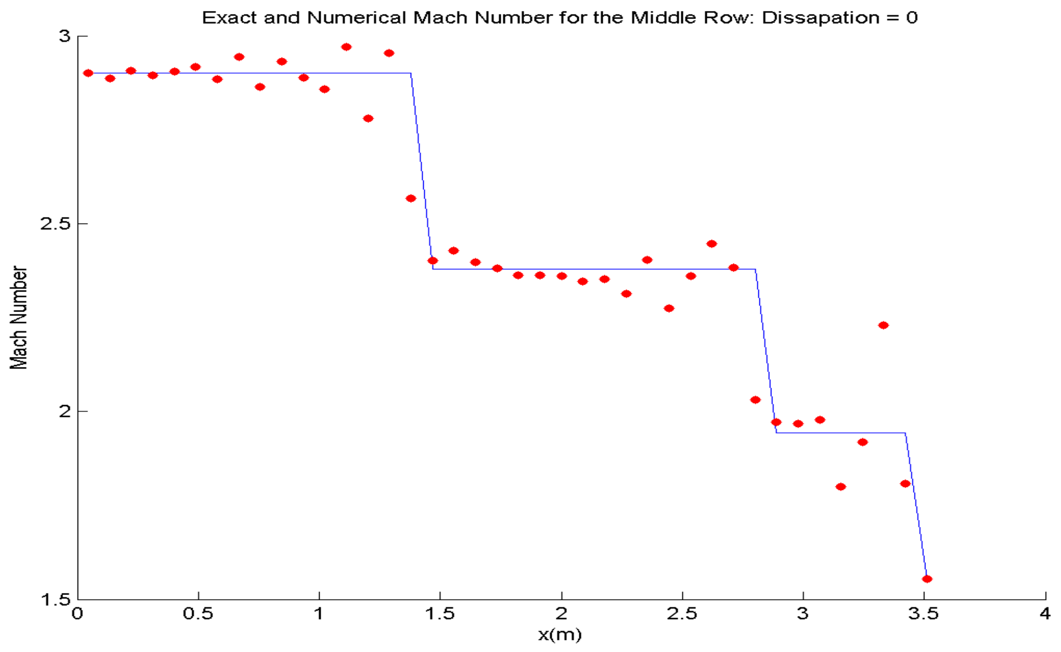


(a)

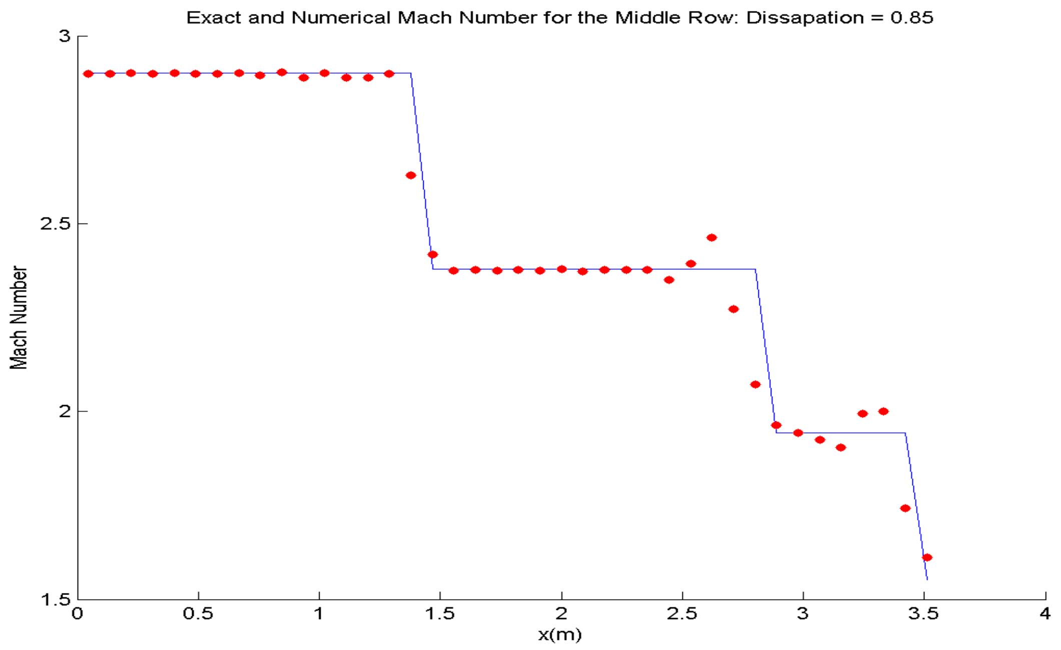


(b)

Figure 10: Figure (a) shows the Density when $\epsilon = 0$. Figure (b) shows the Density when $\epsilon = 0.85$. The exact shocks are superimposed as blue lines. As you can see, both solutions match up with the exact shock, however when there is dissipation the high frequency errors are significantly reduced. While there are still some errors this figure shows the most significant decease in the high frequency errors at the shocks.

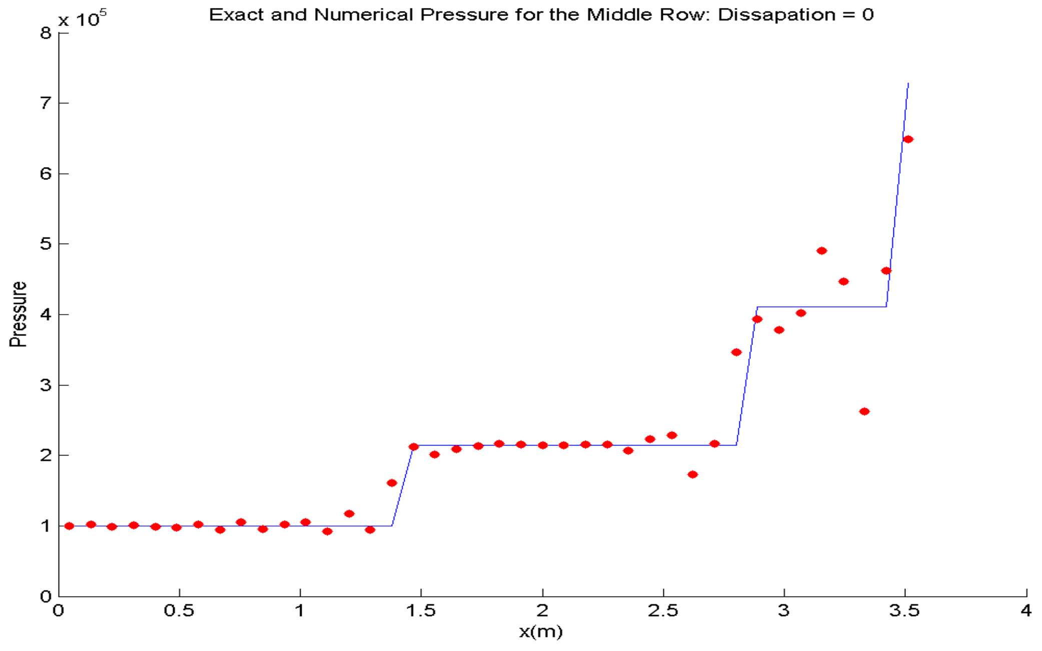


(a)

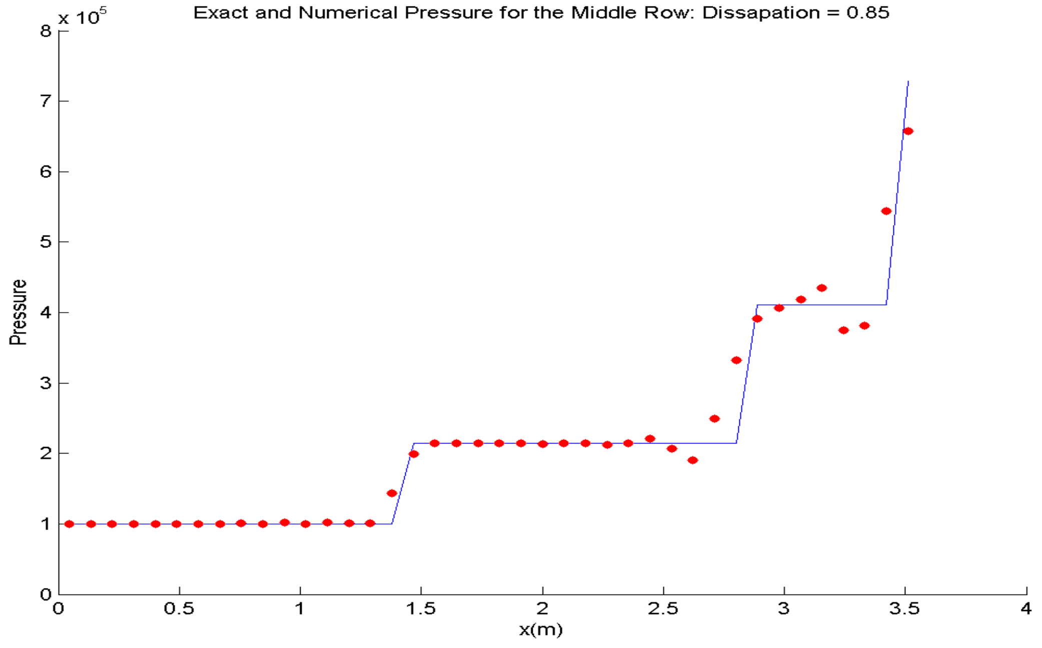


(b)

Figure 11: Figure (a) shows both the exact and the numerical solution for Mach Number when $\epsilon = 0$. *The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. Figure (b) shows both the exact and the numerical solution Mach Number when* $\epsilon = 0.85$. *The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. While the is still some dispersion at the discontinuities, the artificial dissipation help make the solution more accurate. In fact it appears to capture the first shock exactly. It slightly degrades in accuracy as it goes across successive shocks.*

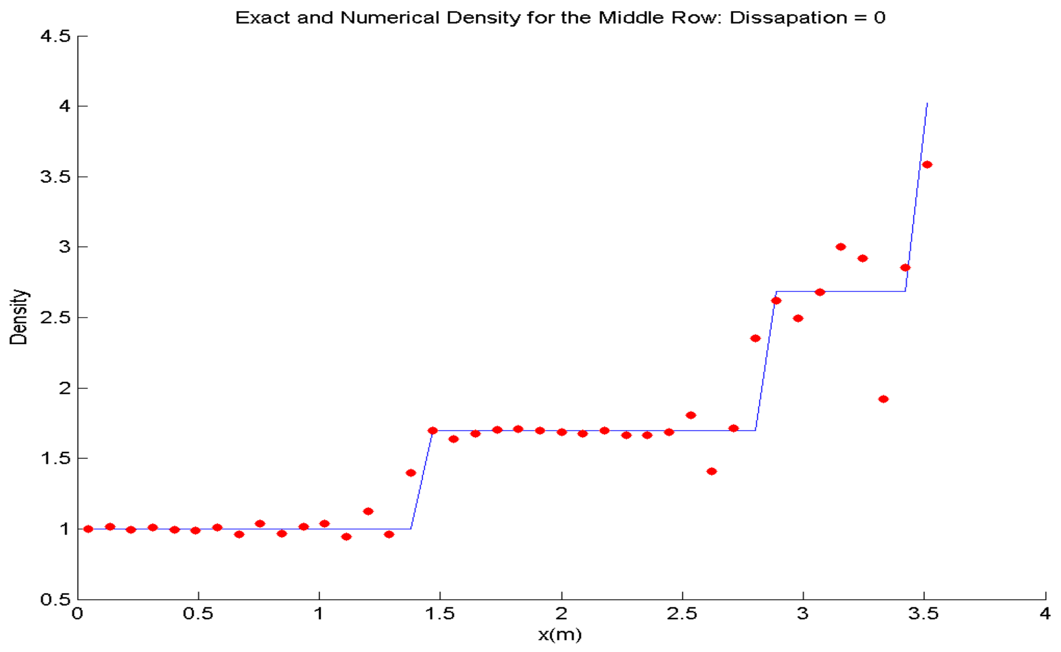


(a)

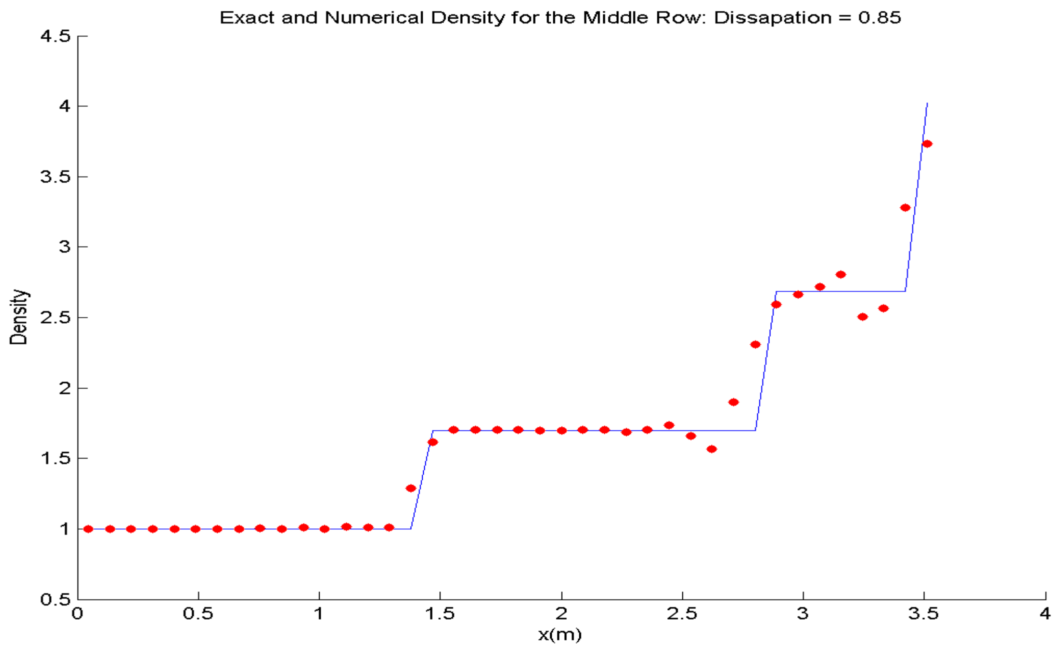


(b)

Figure 12: Figure (a) shows both the exact and the numerical solution for pressure when $\epsilon = 0$. The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. Figure (b) shows both the exact and the numerical solution pressure when $\epsilon = 0.85$. The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. While the is still some dispersion at the later shocks, the artificial dissipation help make the solution much more accurate.



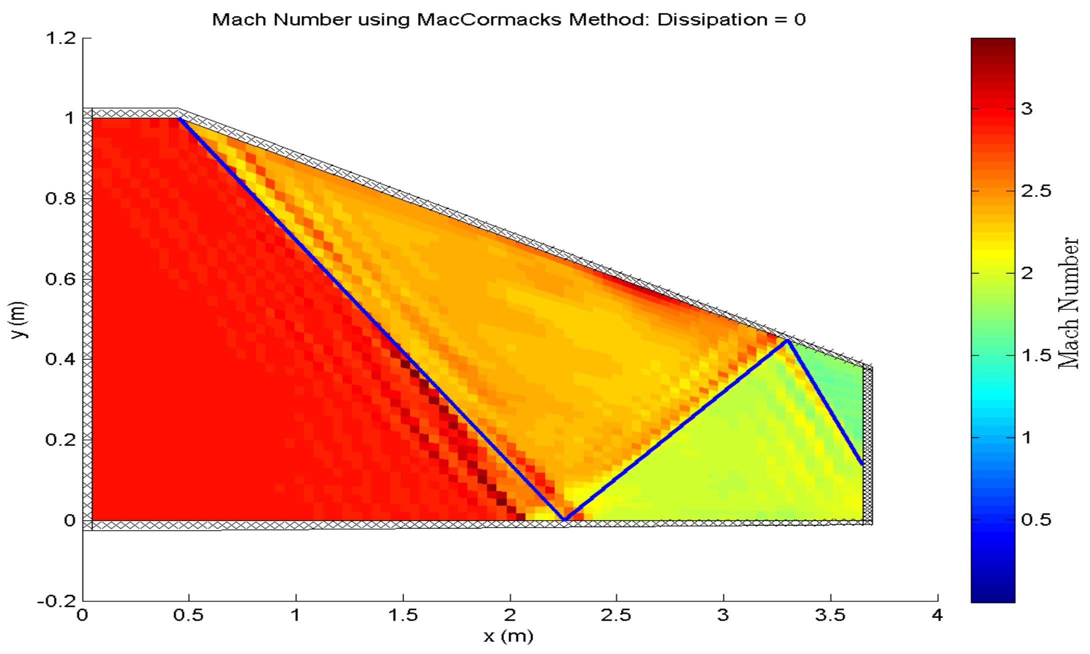
(a)



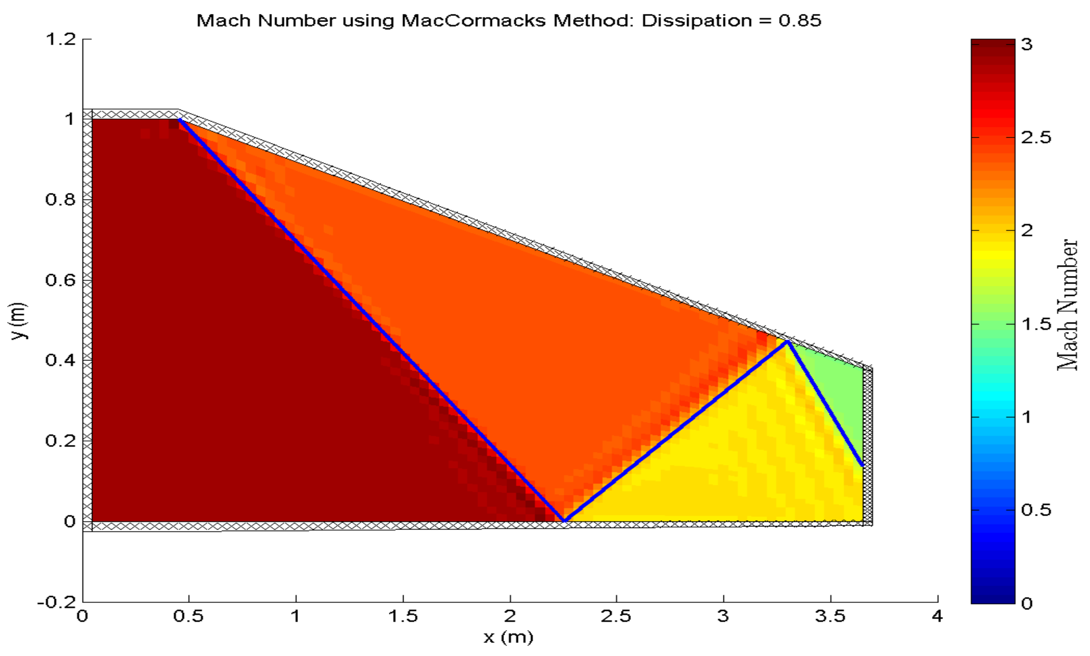
(b)

Figure 13: Figure (a) shows both the exact and the numerical solution for density when $\epsilon = 0$. *The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. Figure (b) shows both the exact and the numerical solution density when* $\epsilon = 0.85$. *The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. While the is still some dispersion at the later shocks, the artificial dissipation help make the solution much more accurate.*

5.6.2 2x Grid

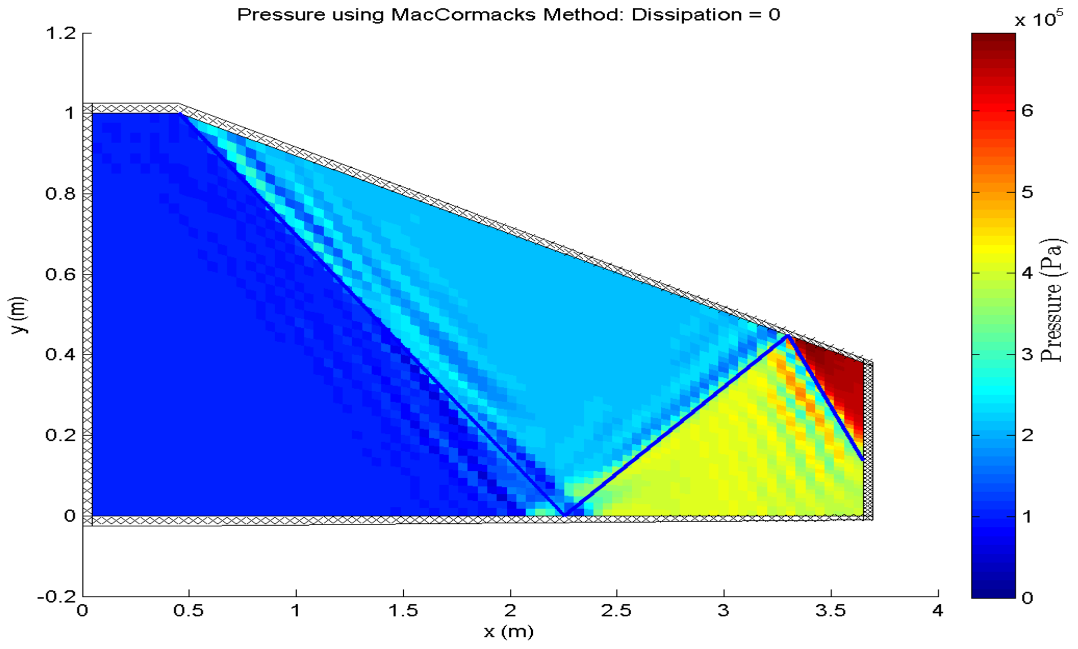


(a)

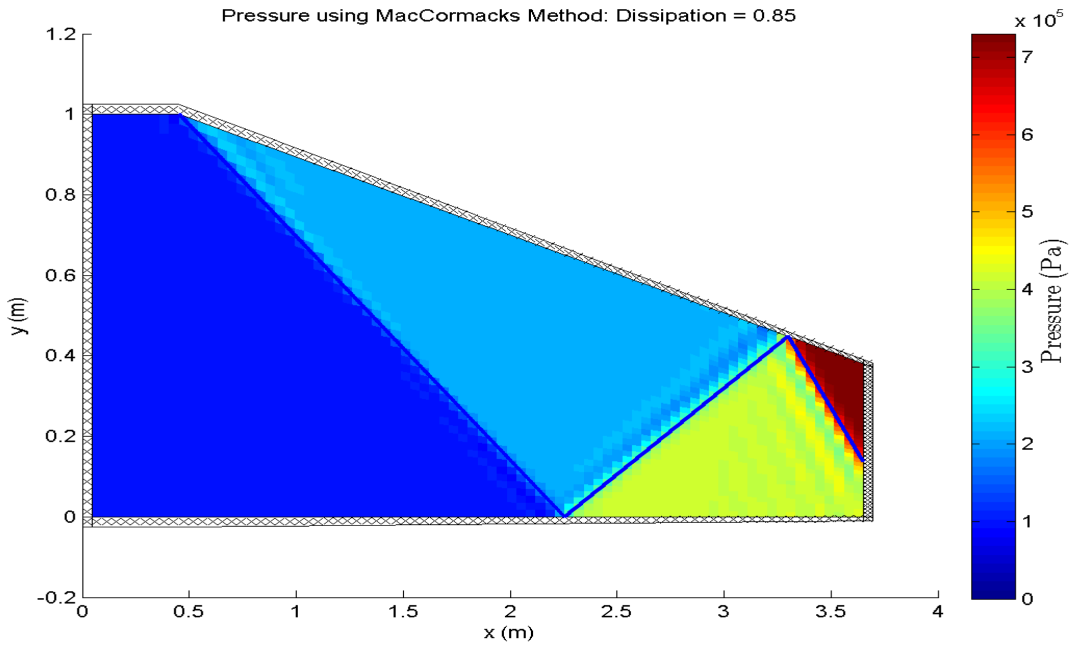


(b)

Figure 14: Figure (a) shows the Mach Number when $\epsilon = 0$. *Figure (b) shows the Mach Number when* $\epsilon = 0.85$. *The exact shocks are superimposed as blue lines. Note that the internal grid borders were dropped to show the solution more clearly. As you can see, both solutions match up with the exact shock, however when there is dissipation the high frequency errors are significantly reduced. Also notice the irregularities that are beginning to form on the top and bottom of the zero dissipation solution.*

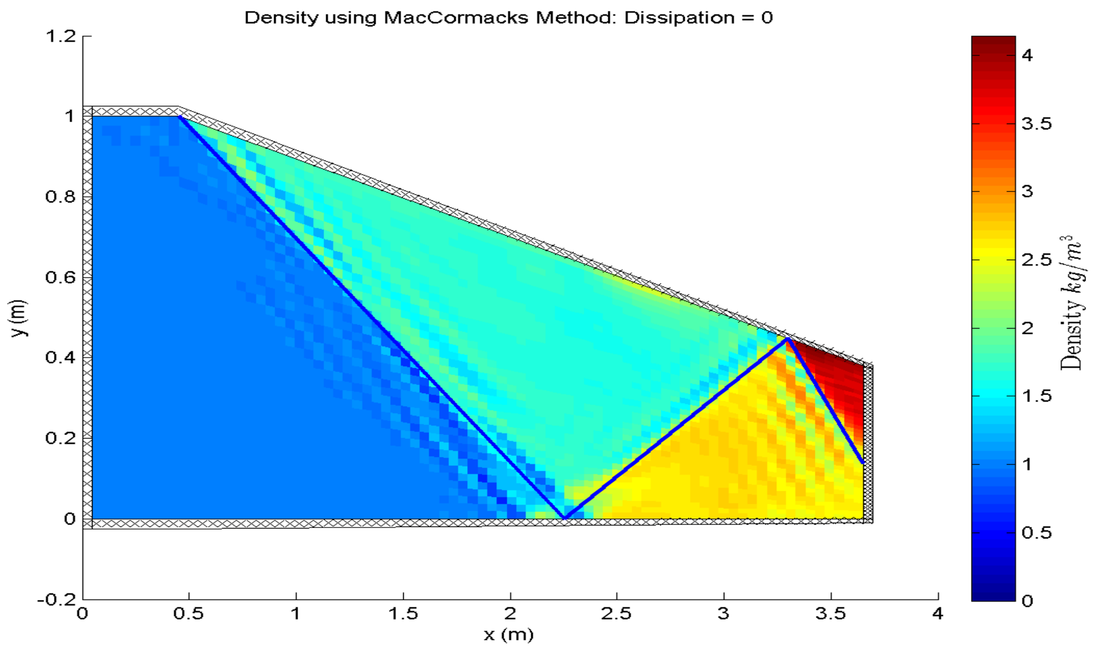


(a)

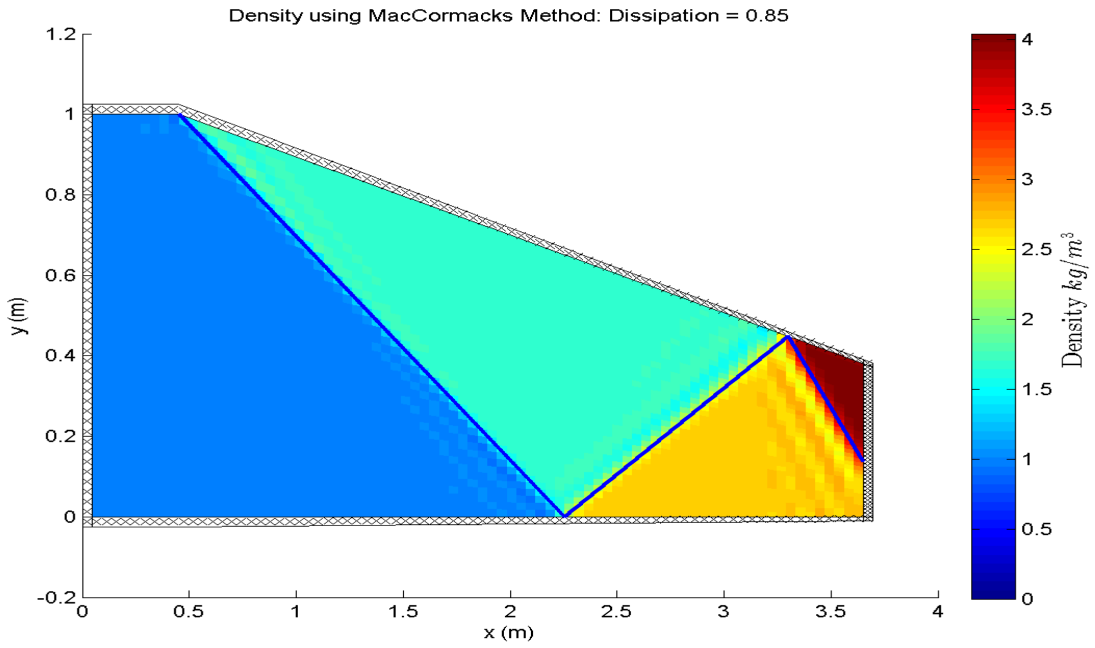


(b)

Figure 15: Figure (a) shows the Pressure when $\epsilon = 0$. Figure (b) shows the Pressure when $\epsilon = 0.85$. The exact shocks are superimposed as blue lines. As you can see, both solutions match up with the exact shock, however when there is dissipation the high frequency errors are significantly reduced.

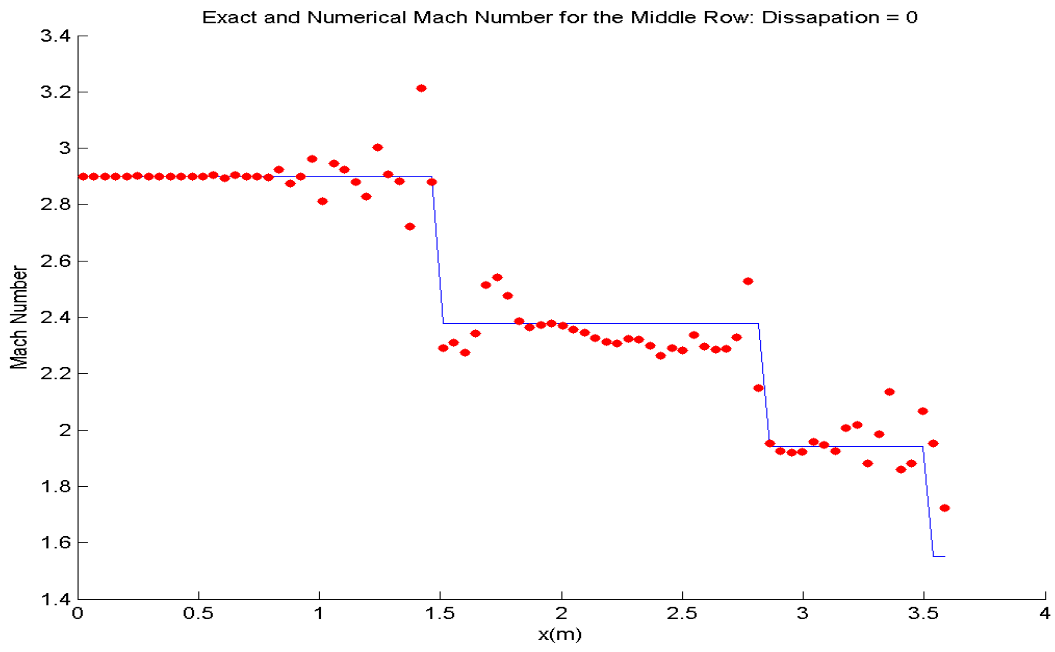


(a)

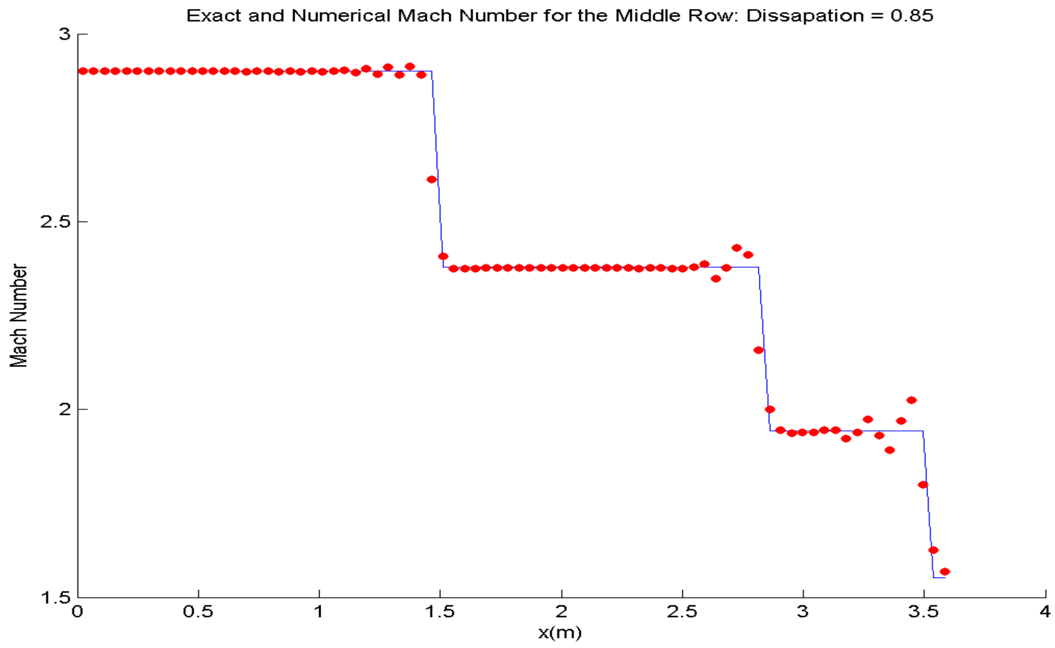


(b)

Figure 16: Figure (a) shows the Pressure when $\epsilon = 0$. Figure (b) shows the Pressure when $\epsilon = 0.85$. The exact shocks are superimposed as blue lines. As you can see, both solutions match up with the exact shock, however when there is dissipation the high frequency errors are significantly reduced.Also you can see the irregularities that are beginning to form in the top middle of the second section as well as the intersection of shocks 1 and 2. These anomalies will eventually explode making the zero dissipation solution useless.

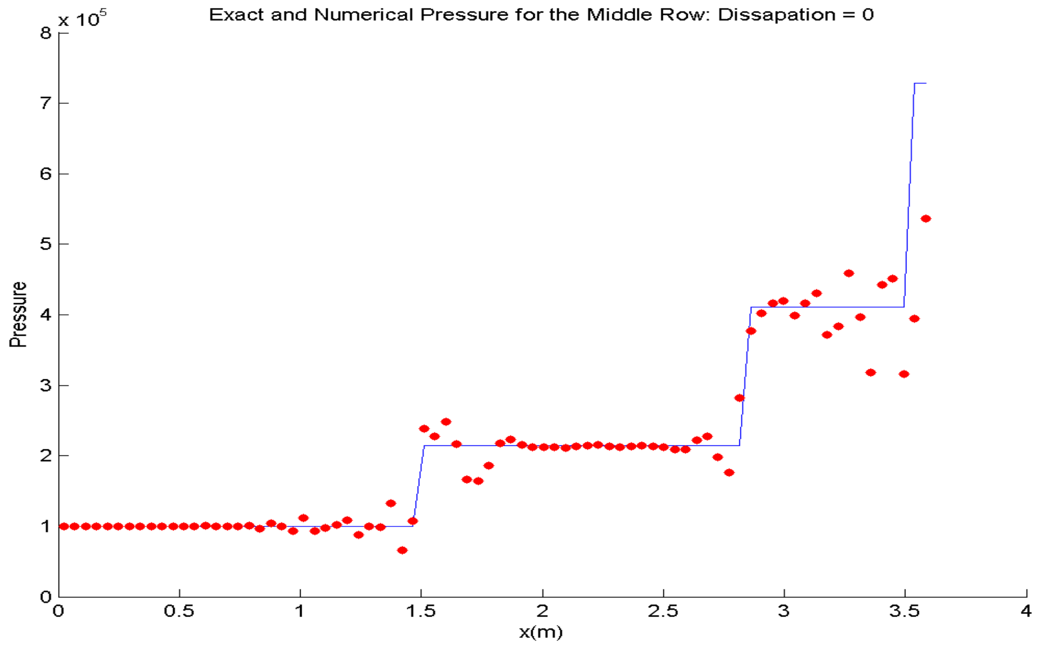


(a)

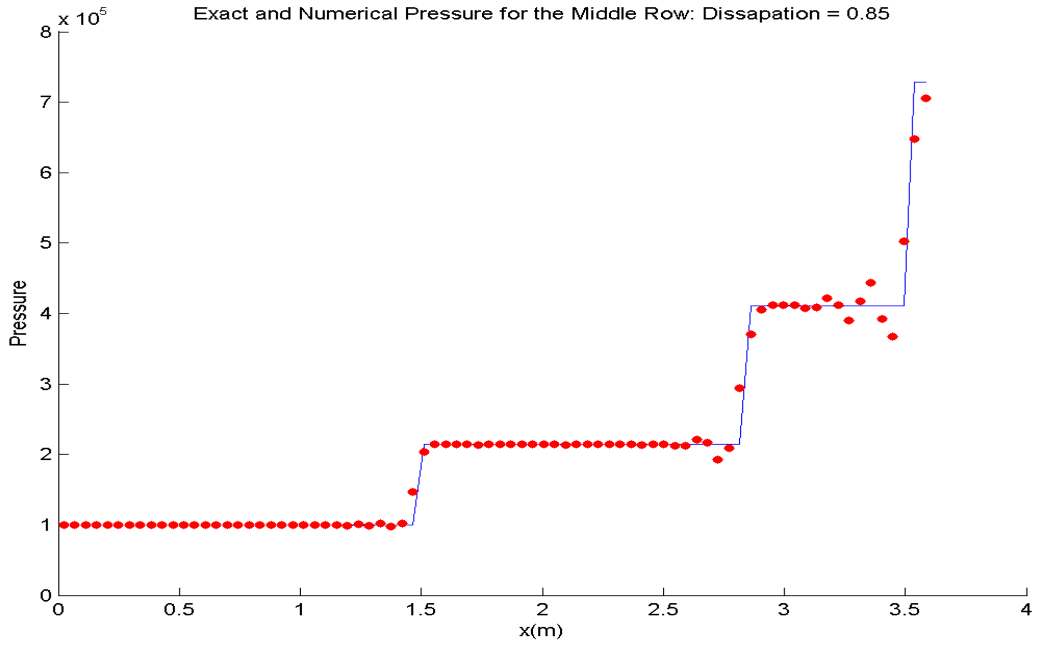


(b)

Figure 17: Figure (a) shows both the exact and the numerical solution Mach Number when $\epsilon = 0$. *The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. Figure (b) shows both the exact and the numerical solution Mach Number when* $\epsilon = 0.85$. *The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. While the is still some dispersion at the discontinuities, the artificial dissipation help immensely. It virtually perfectly captures the first shock, and is still very accurate for the enxt two. The only significant dispersion is on the third shock.*

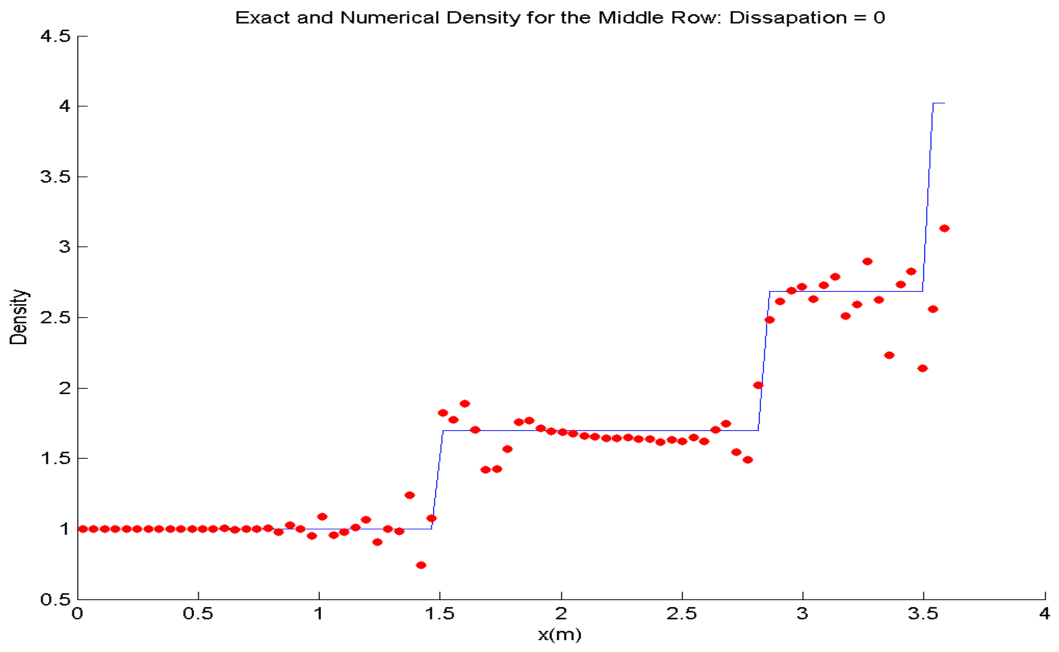


(a)

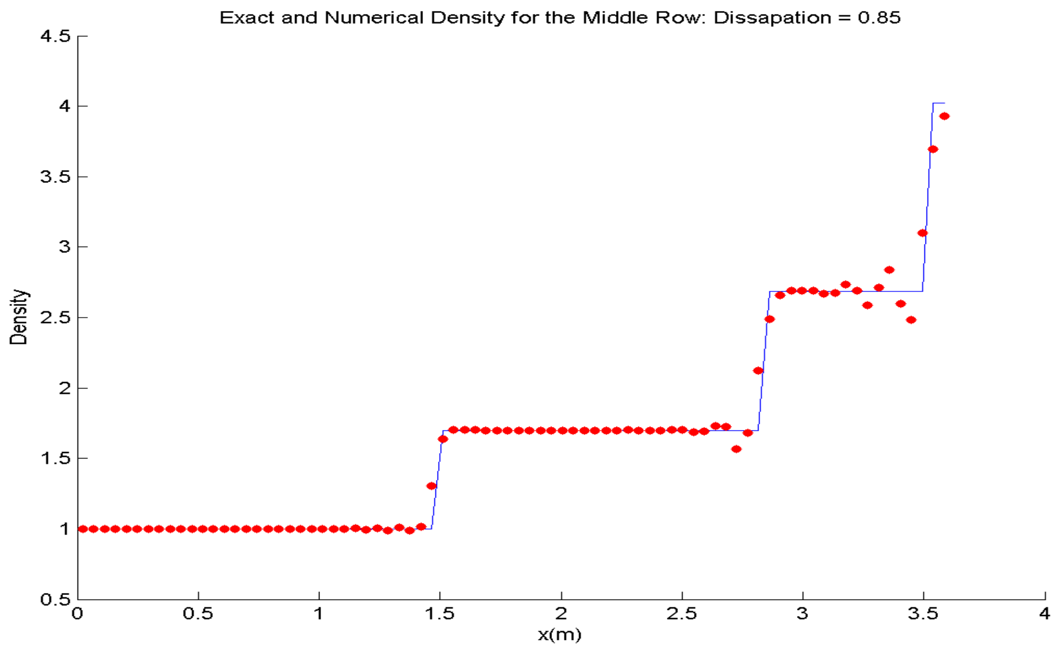


(b)

Figure 18: Figure (a) shows both the exact and the numerical solution for pressure when $\epsilon = 0$. The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. Figure (b) shows both the exact and the numerical solution pressure when $\epsilon = 0.85$. The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. While the is still some dispersion at the discontinuities, the artificial dissipation help make the solution dramatically more accurate.



(a)

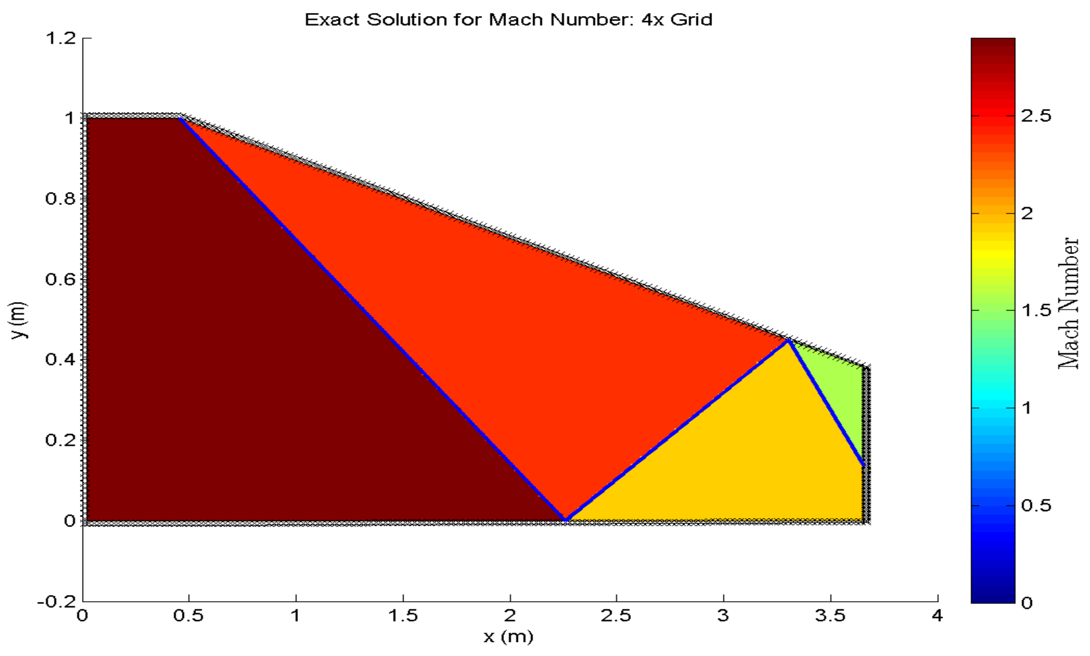


(b)

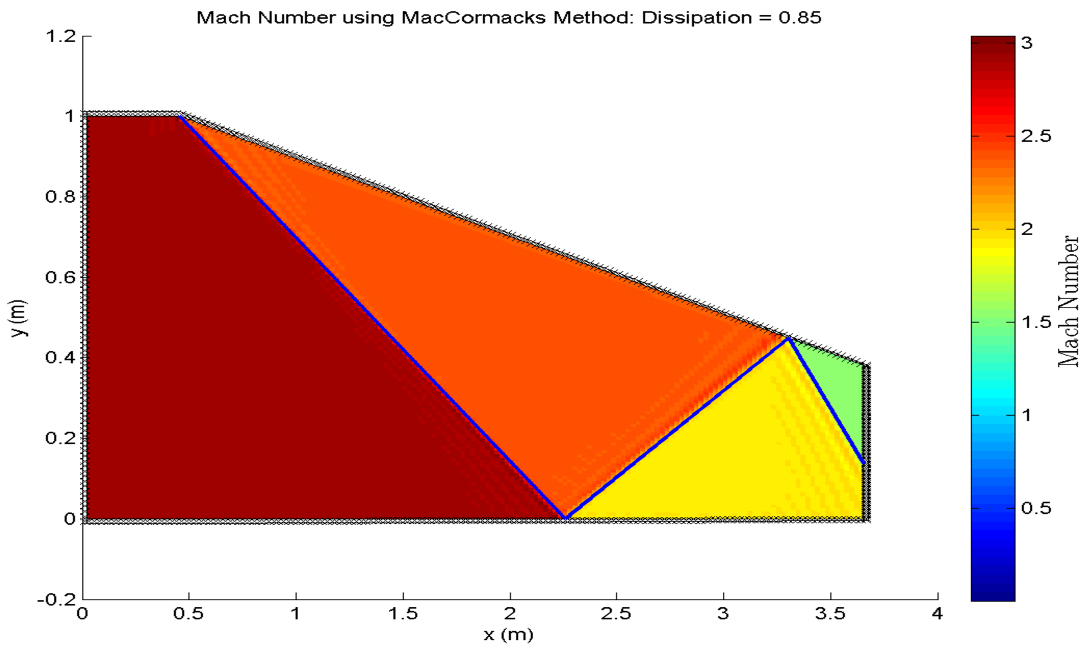
Figure 19: Figure (a) shows both the exact and the numerical solution for density when $\epsilon = 0$. The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. Figure (b) shows both the exact and the numerical solution density when $\epsilon = 0.85$. The exact solution is plotted as a blue line while the numerical solution is plotted as red dots. While the is still some dispersion at the discontinuities, the artificial dissipation help make the solution dramatically more accurate.

5.7 Final Results Using 4x Grid

In this section we will present all plots for MacCormacks Method with Artificial Dissipation on the 4x grid to showcase the power of this algorithm. The exact and numerical solutions will be plotted on the same page in order to make the comparison easy. The consistency between the two is great. Full page figures of the numerical solutions with 8x grid will be provided in the appendix. Resolution beyond 4x does not add much to the accuracy, although it does showcase the efficiency of my algorithm.

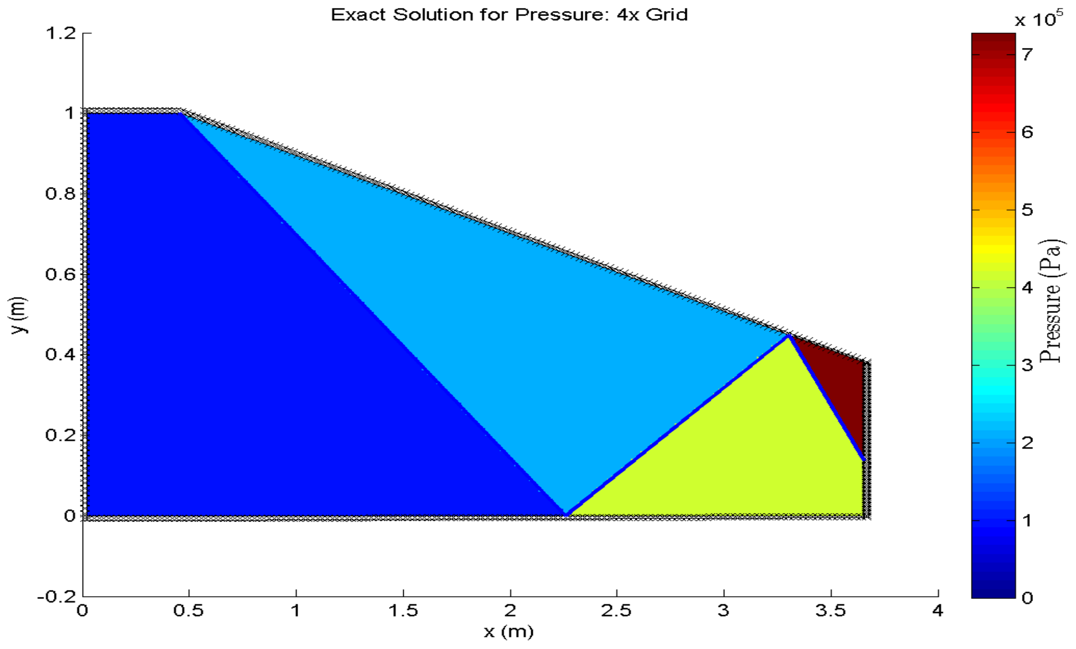


(a)

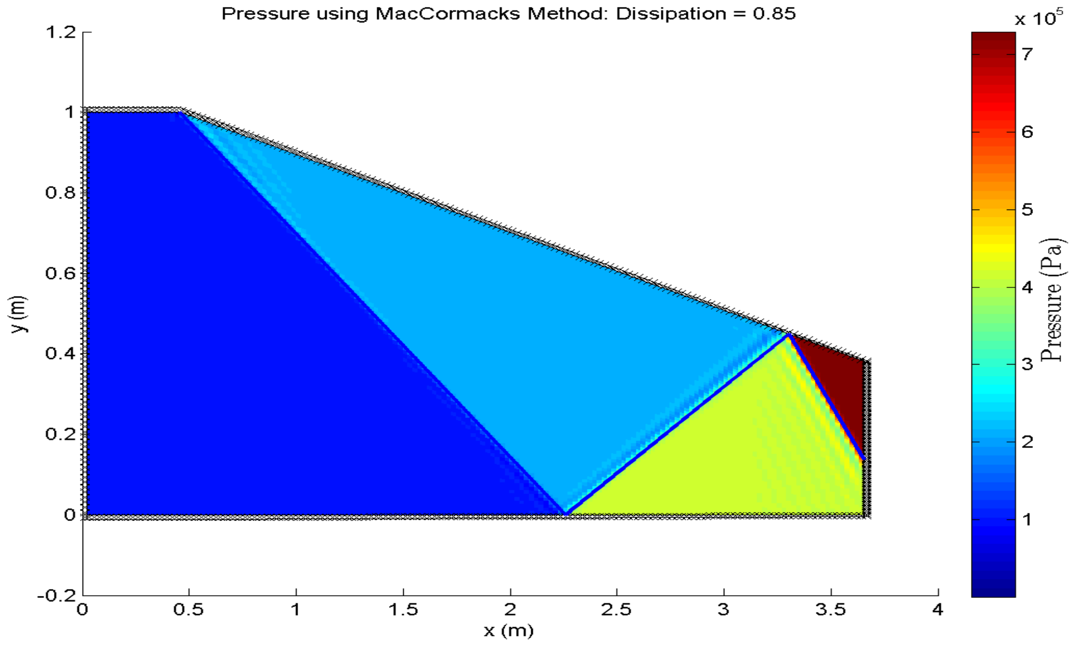


(b)

Figure 20: Figure (a) shows the Mach Number for the exact solution. Figure (b) shows the Mach Number for the MacCormack Method when $\epsilon = 0.85$ *and 4x grid.*

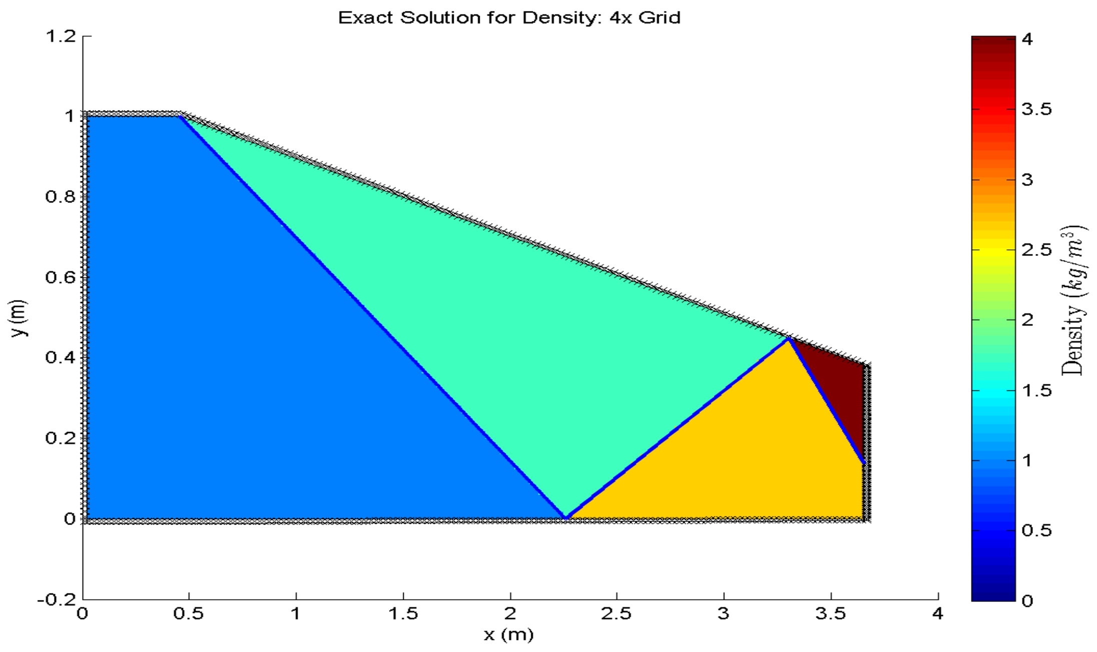


(a)

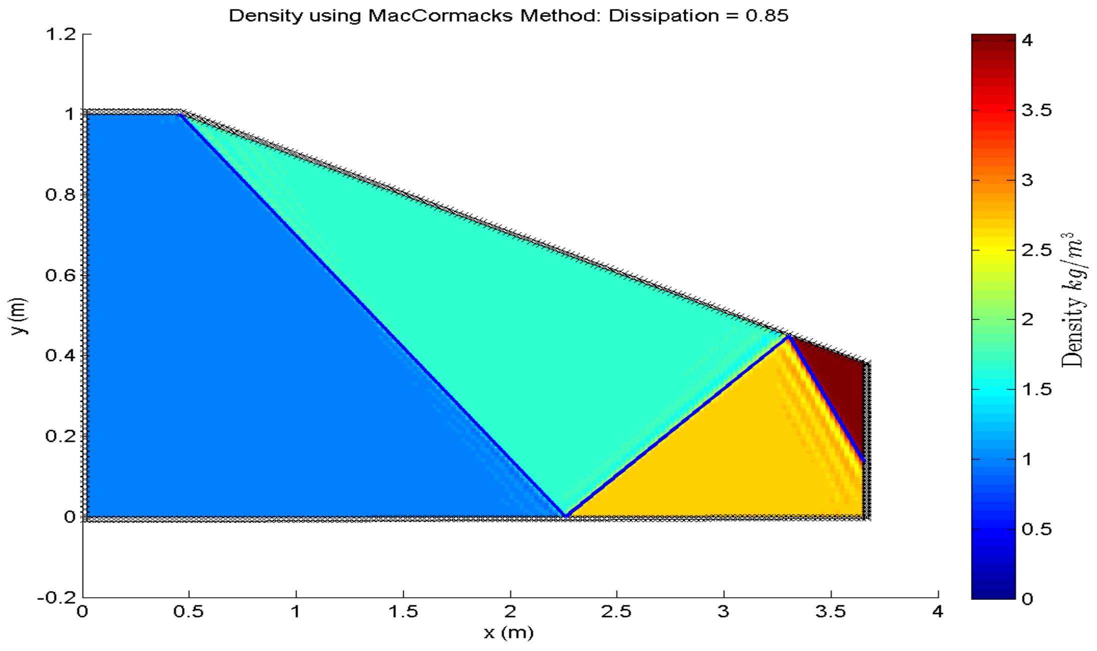


(b)

Figure 21: Figure (a) shows the Pressure for the exact solution. Figure (b) shows the Pressure for the MacCormack Method when $\epsilon = 0.85$ adn 4x grid.



(a)



(b)

Figure 22: Figure (a) shows the Density for the exact solution. Figure (b) shows the Density for the MacCormack Method when $\epsilon = 0.85$ and 4x grid.

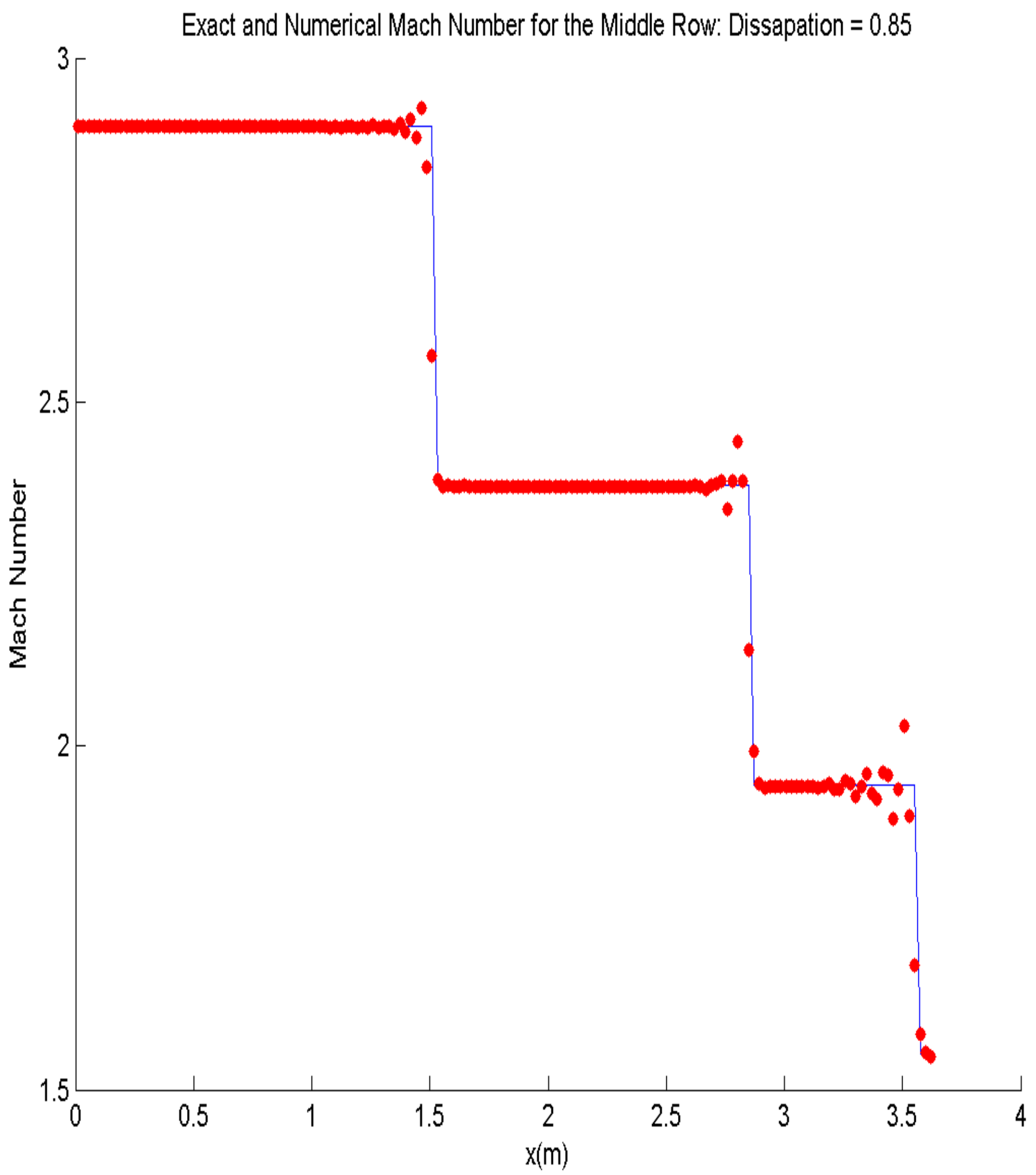


Figure 23: This figure shows the numerical solution for Mach Number in the middle row of the 4x grid. The exact solution is shown in blue. This figure makes it apparent that my solution accurately captures the 3 shocks that form in the supersonic jet inlet. You can clearly see the small dispersion that occurs at the shocks, but overall consistency is great.

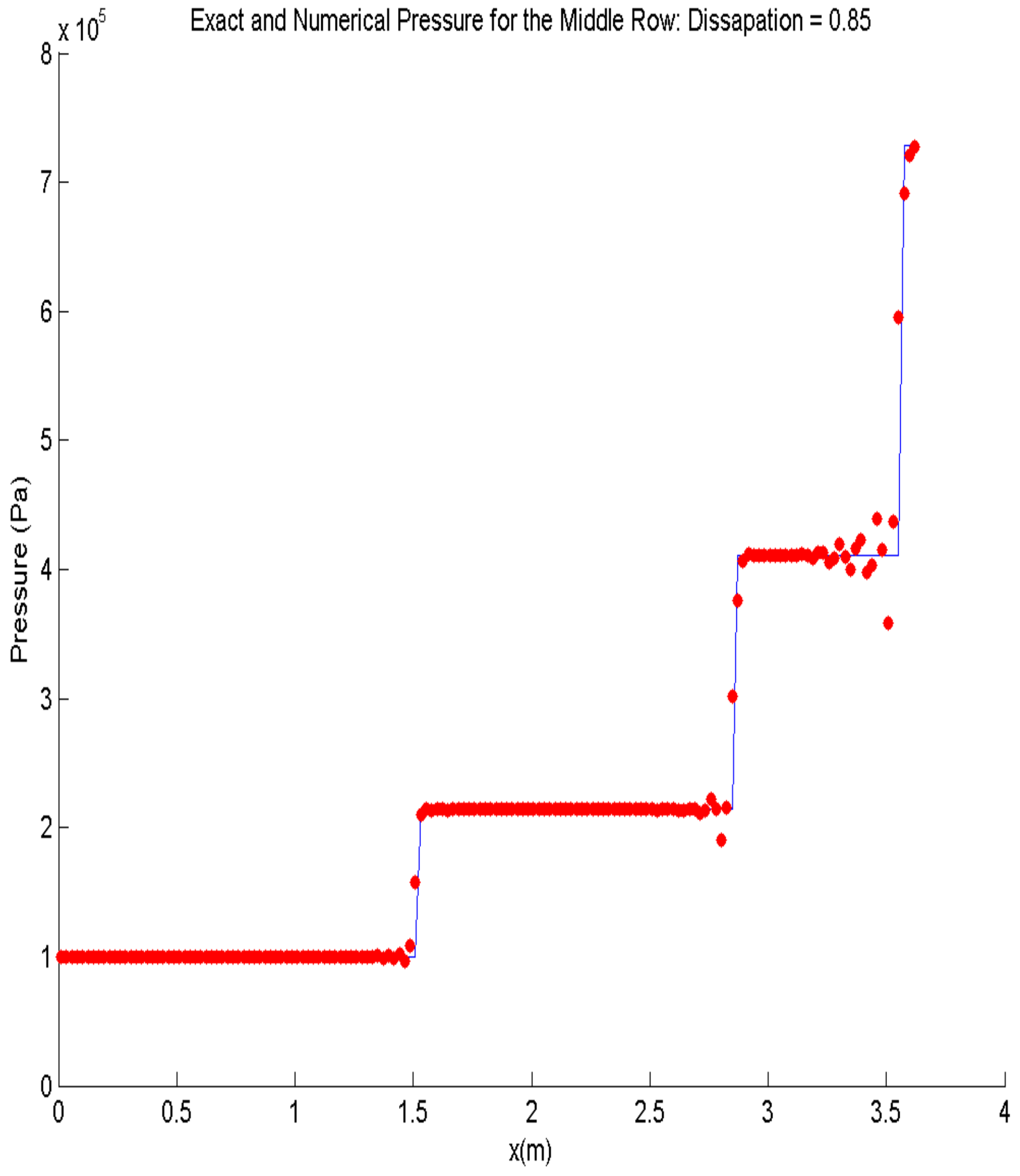


Figure 24: This figure shows the numerical solution for pressure in the middle row of the 4x grid. The exact solution is shown in blue. This figure makes it apparent that my solution accurately captures the 3 shocks that form in the supersonic jet inlet. You can clearly see the small dispersion that occurs at the shocks, but overall consistency is great.

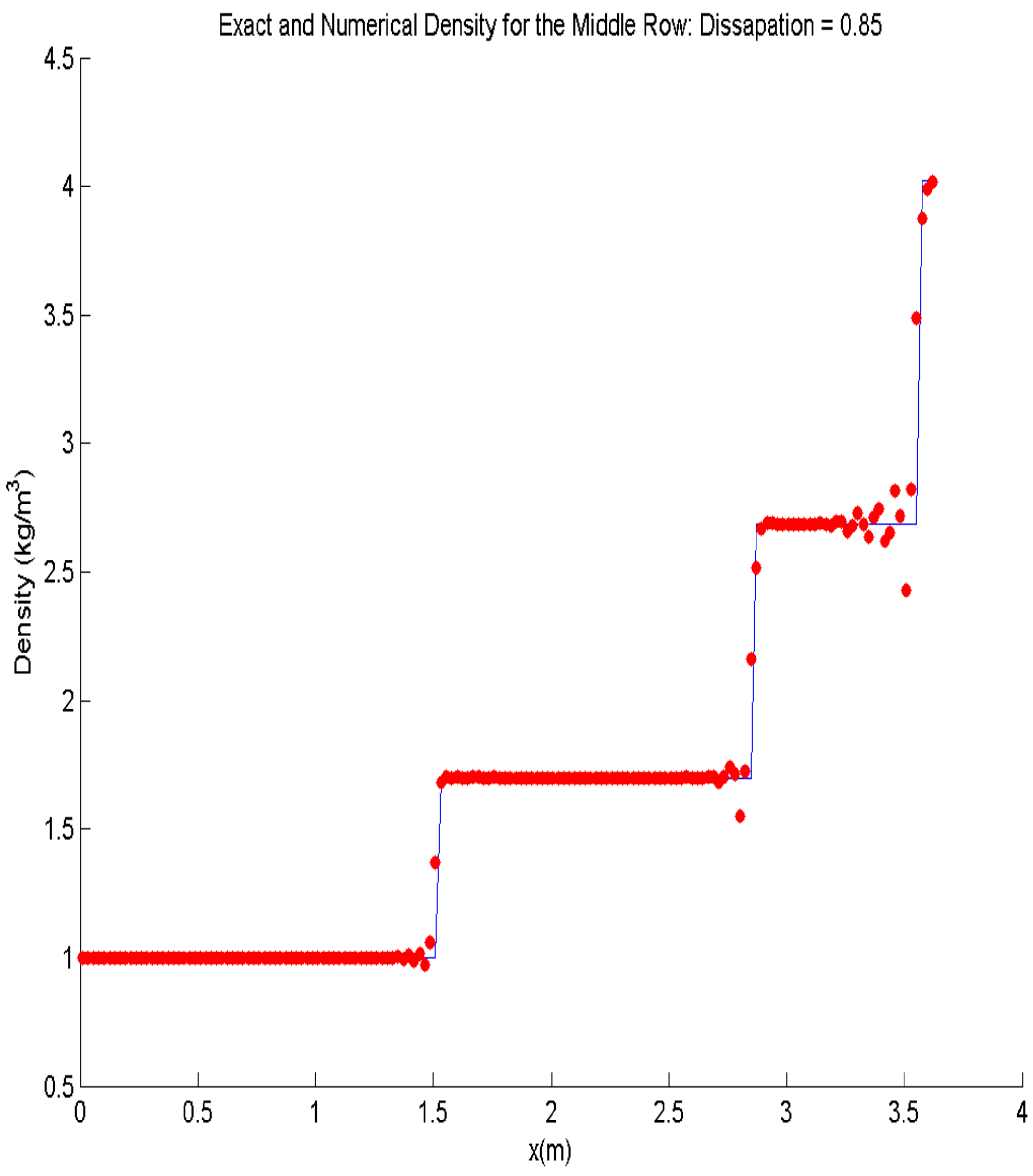


Figure 25: This figure shows the numerical solution for density in the middle row of the 4x grid. The exact solution is shown in blue. This figure makes it apparent that my solution accurately captures the 3 shocks that form in the supersonic jet inlet. You can clearly see the small dispersion that occurs at the shocks, but overall consistency is great.

Overall we see great consistency between the exact an numerical solutions. While there is still some dispersion across the shocks, the accuracy of the algorithm as a whole is great. The first shock is captured almost perfectly and the dispersion increases for the next two shock. As a design tool, trying to numerically determine where shocks where occur, and the bulk properties of the exit flow, this algorithm is great.

6 Time Complexity

In the computational world one of the most important traits of an algorithm or solver is how long it takes. While accuracy of solution is definitely the most important, solve time is a close second, and a certain mount of accuracy is often sacrificed for speed. In the engineering world profit is about throughput and if your simulations are slow, your design iterations will be slow which ultimately results in less profit. That being said my algorithm for calculating the oblique shocks that form in a supersonic jet inlet has vastly improved upon the baseline solution, boasting time coefficient improvements of two orders of magnitude. Before showing how the algorithms compare i will give a brief description of the algorithms.

Baseline

The baseline solution is using two nested for loops to iterate over the grid. This is a very easy and natural way to implement this problem since it allows you to explicitly see what is happening each iteration at the node level. Unfortunately nested for loops are an inherently slow algorithm.

Optimized Algorithm

The optimized algorithm works using smart indexing. Since the same computation is being computed each cell, you can do the computations on the matrix as a whole. This means when you want to look at the (i+1,j) elements you would shift the whole matrix in the i direction by 1. Remember that internal nodes span $2 \leq i \leq IL-1$ and $2 \leq j \leq JL-1$ and appropriate boundary conditions are set on the slave cells where i=1 or i = IL or j = 1 or j = JL. Therefore when we want access to the (i+1,j) elements with regard to the (i,j) nodes, you can just use the range $3 \leq i \leq IL$ and $2 \leq j \leq JL-1$. Similarly you can shift the matrix any which way.

Having explained the basics of the two algorithms figures showing the difference in run time will be presented. The first figure shows the relationship between run time and time steps.

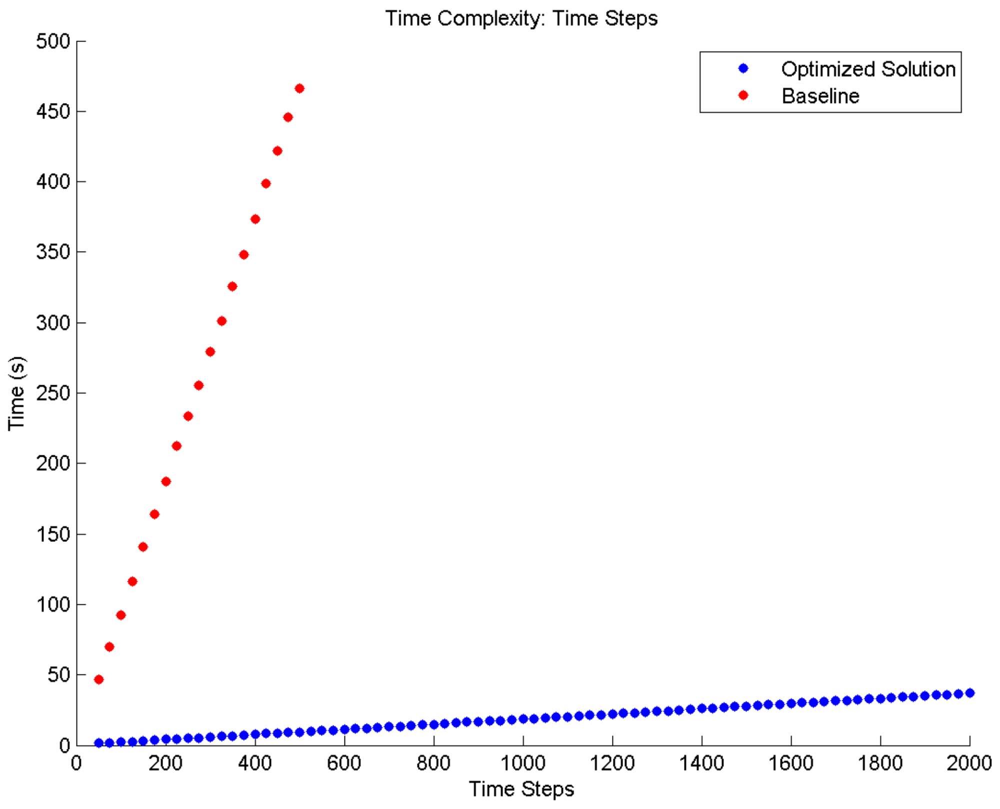


Figure 26: This figure shows the runtime comparison between my solution and the baseline solution with respect to the number of time steps. The relationship is linear in both cases but the coefficients are much different.

This shows a dramatic improvement over the baseline solution. The two trend lines are

$$t_b(TS) \approx 0.933x$$

$$t_o(TS) \approx 0.025x$$

where t_b is the baseline runtime as a function of time steps(TS) and t_o is the optimized runtime as a function of time steps(TS). Note that the difference in coefficients is about 1.4 orders of magnitude!! That is a gigantic difference especially when runs may take upwards of 10000 iterations to converge. These are both linear relationships, which in the grand scheme of programing is actually good. The next figure shows how runtime varies as a function of grid scale.

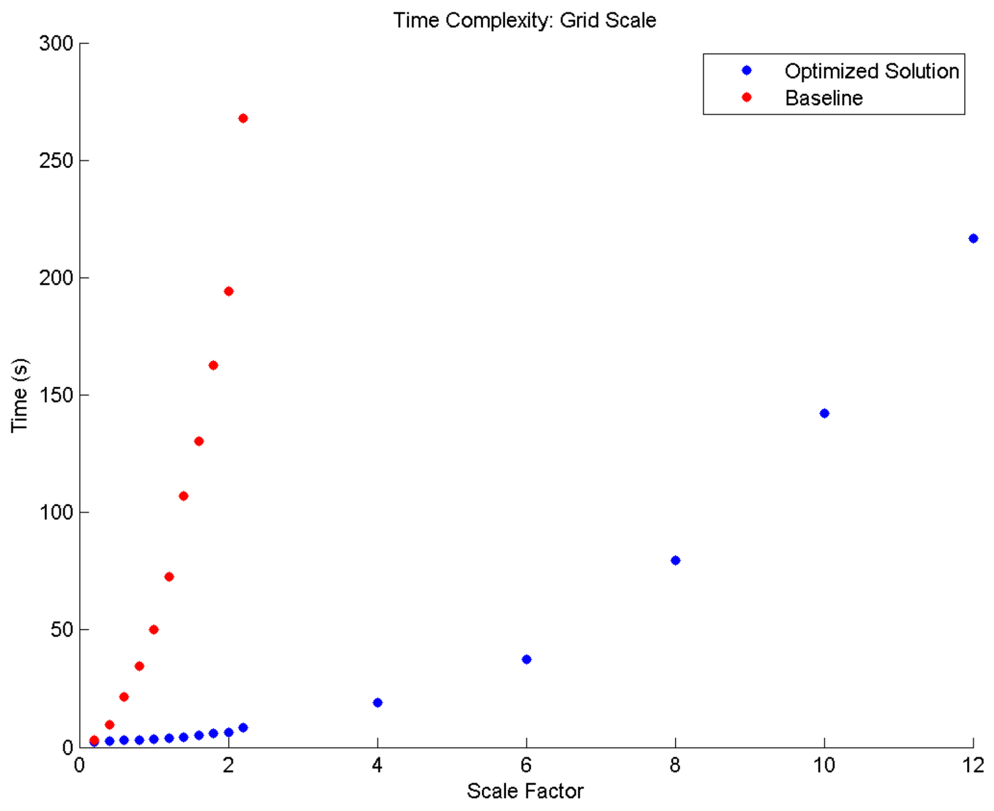


Figure 27: This figure shows the runtime comparison between my algorithm and the baseline solution with respect to the grid size. The relationship is order n^2 and the optimized solution is dramatically better then the baseline

Now that we have an $O(n^2)$ relationship the coefficient of the trend equation is even more important. The two trend lines are

$$t_b(GS) \approx 53.4x^2$$
$$t_o(GS) \approx 1.64x^2$$

Where the runtime is a function of the grid scale (GS). This relationship shows an almost 1.4 order of magnitude difference between the two coefficients. This has enormous consequences since the relationship is n^2. Combined with the linear relationship of time steps, my algorithm is over 2 orders of magnitude faster than the baseline solution.

7 Conclusion

While no numerical method will ever totally recreate the exact analytical solution, numerical solutions can be applied in cases where the analytical solution is either impossible to solve, or incredibly difficult to solve. The algorithm i have developed over the last several weeks accurately captures the shocks that form inside a supersonic jet inlet and does so much quicker than the baseline solution. The average accuracy per node was found to be roughly one percent, and the speed was found to be about 2 orders of magnitude faster than the baseline. If my algorithms were expanded to encapsulate general geometries rather than the prescribed jet engine inlet, it could be a valuable asset in quickly designing geometries that are required to produce desired flow conditions.

8 Source Code Listing

main.m
The main routine which sets up a series of boolean switches to decide which plots the user desires to create. The parameters for grid size and dissipative terms cal also be set here.

cell_normals.m
This routine takes in the stationary grid and calculates the volume, the four edge lengths and the four normal vectors for each node. Volume is stored as a single number, the lengths are stored as a 1x4 array where each elements is a length, and the normals are stored in a 4x2 array where the x and y components of the normal are stored in different rows.

compareToExact.m
This routine takes in the exact and numerical solutions and plots them along three characteristic rows. The top row, the bottom row and the middle row. This is so you can see exactly how far off we are. Its easier to see in these types of plots rather than color plots. It also returns the bulk error of the numerical solution with respect to the exact.

disterms.m
This routine is used exclusively by the baseline solution. I calculates the dissipative terms for a single node.

disterms2.m
This routine is used by the optimized algorithm. It calculates the dissipative terms for the predictor step for the entire matrix.

disterms4.m
This routine is used by the optimized algorithm. It calculates the dissipative terms for the corrector step for the entire matrix.

exact_sol.m
This routine takes in the grid and solves the analytical formulas presented in the Theory section. It gives the exact solution of the problem at internal node.

gen_grid.m
This routine creates the grid. It takes a single parameter which is the scaling factor of the grid.

macdis.m
This is the baseline solution. It uses for loops to iterate through the entire grid while performing the steps of MacComracks Method.

macFAST.m
This routine is the optimized solution that relies on smart indexing to compute whole matrices at a time.

normal_shock.m
This routine is used by the exact solution solver to solve for the property change across a shock. It take in the upstream Mach Pressure and Density and gives you the same vales on the downstream side.

plot_grid.m
This routine plots the grid that was made by gen_grid.m. It has various optional arguments to decide how you want to plot the grid. Ex if you want grid lines on or off.

plot_results.m
This routine takes in the gird and the solution array and plots it as a color plot.

recalc_EF.m
This routine takes in the state vector U and computes the flux vectors E and F. This is used internally inside macFAST.m and macdis.m Since the flux vectors need to be recalculated ever step it is nice to have a separate function handle that.

set_slave.m
A very important function that sets the slave cells as described in the Numerical Methods section. Was made a seperate function because it is called every step of the MacCormack solution.

tantheta.m
This function is simply the theta beta Mach relationship, but the returned value is $tan(\theta)$ as a function of β and M.

tbM.m
This is the function that performs the Newton-Rhapson iteration to solve for beta. It is used in the exact solution to solve for the oblique shocks analytically.

9 References

[1] Zhong, Xiaolin. "MAE250D-Notes." UCLA. Winter 2015.

[2] "Normal Shock Wave." NASA. Web. Jun. 12, 2014. "http://www.grc.nasa.gov/WWW/k-12/airplane/normal.html." Accessed Mar. 14, 2015.

10 Appendix

Additional figure begin on the next page.

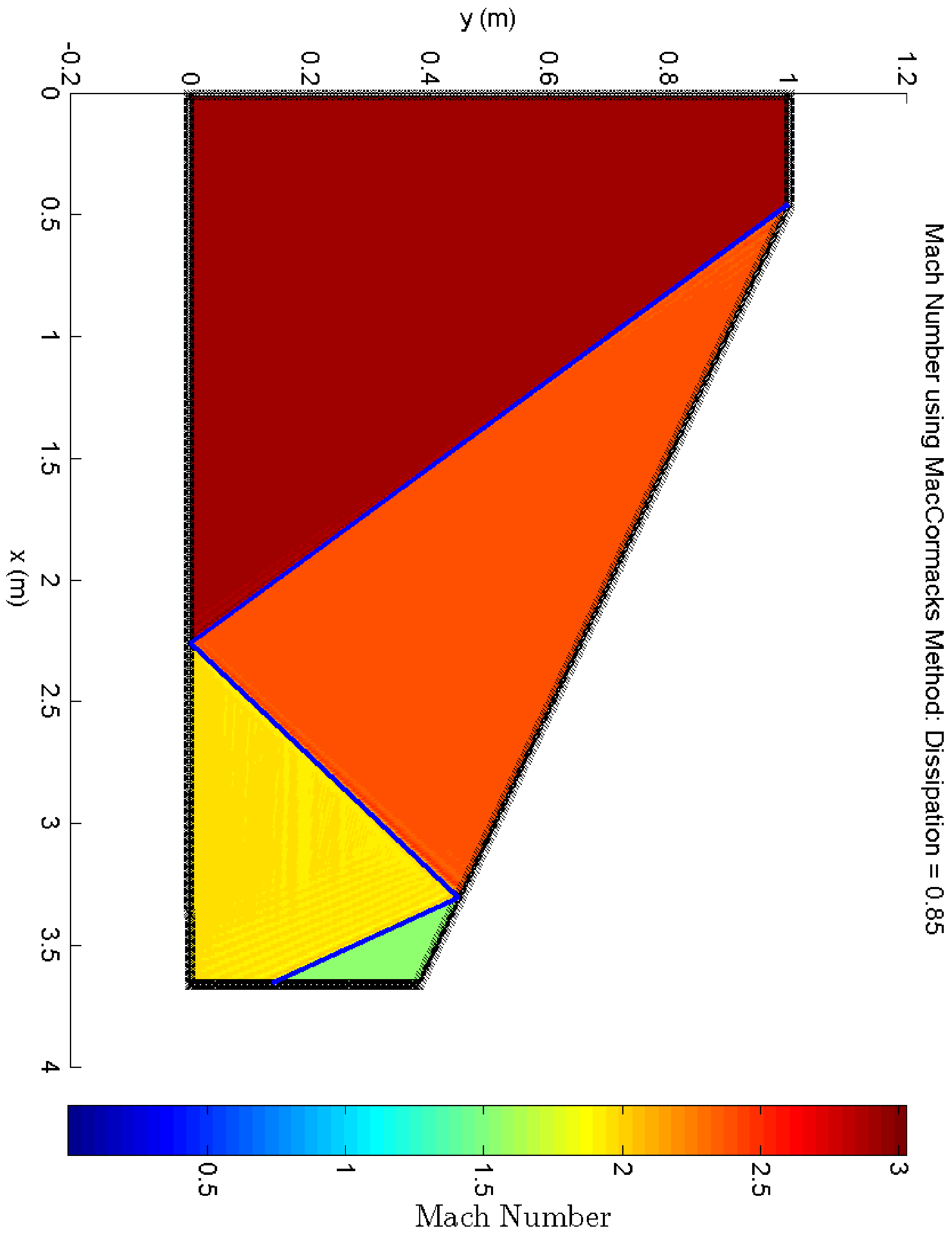


Figure 28: This figure shows the numerical solution for Mach number on the 8x grid. The exact shocks are shown as superimposed blue lines.

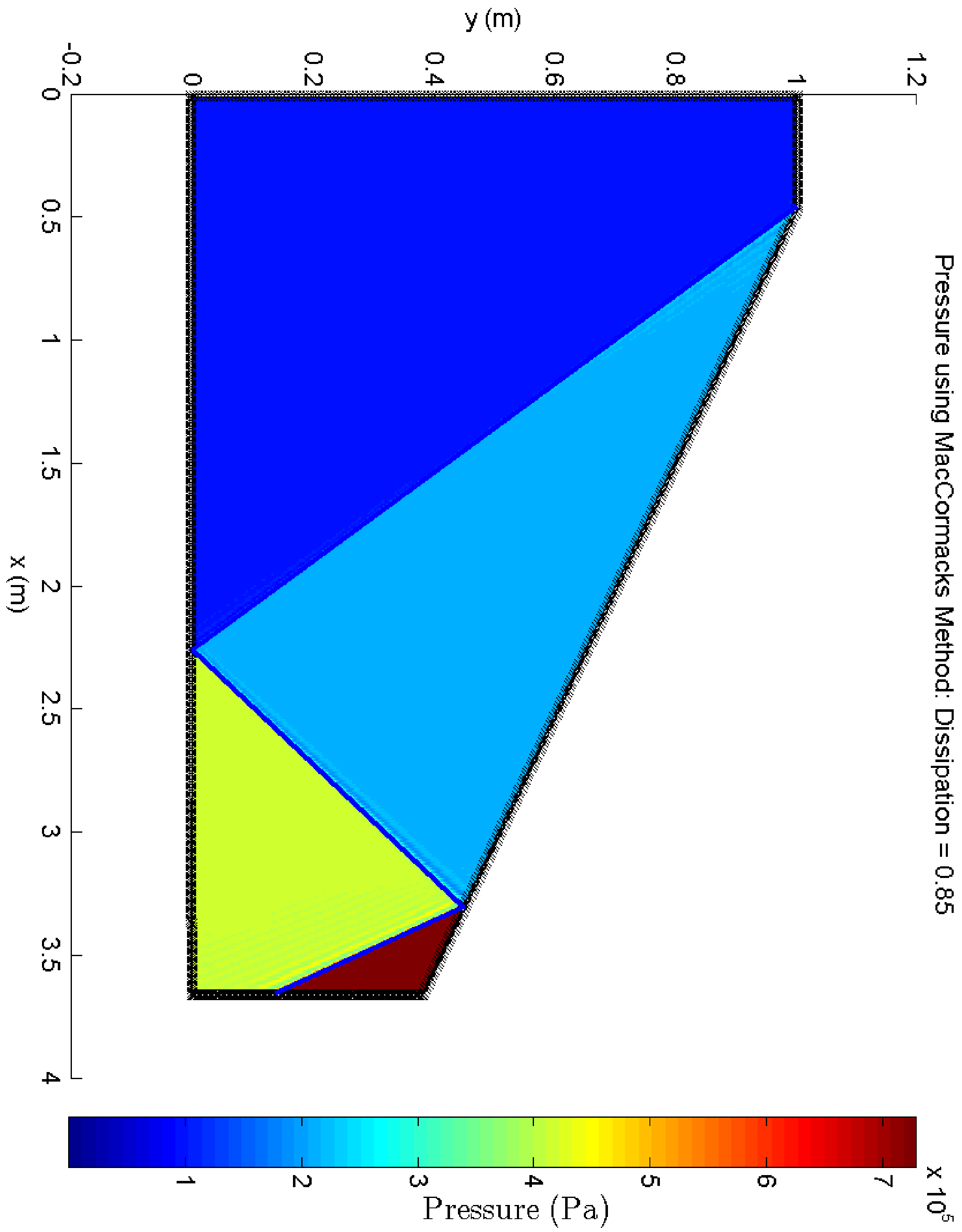


Figure 29: This figure shows the numerical solution for pressure on the 8x grid. The exact shocks are shown as superimposed blue lines.

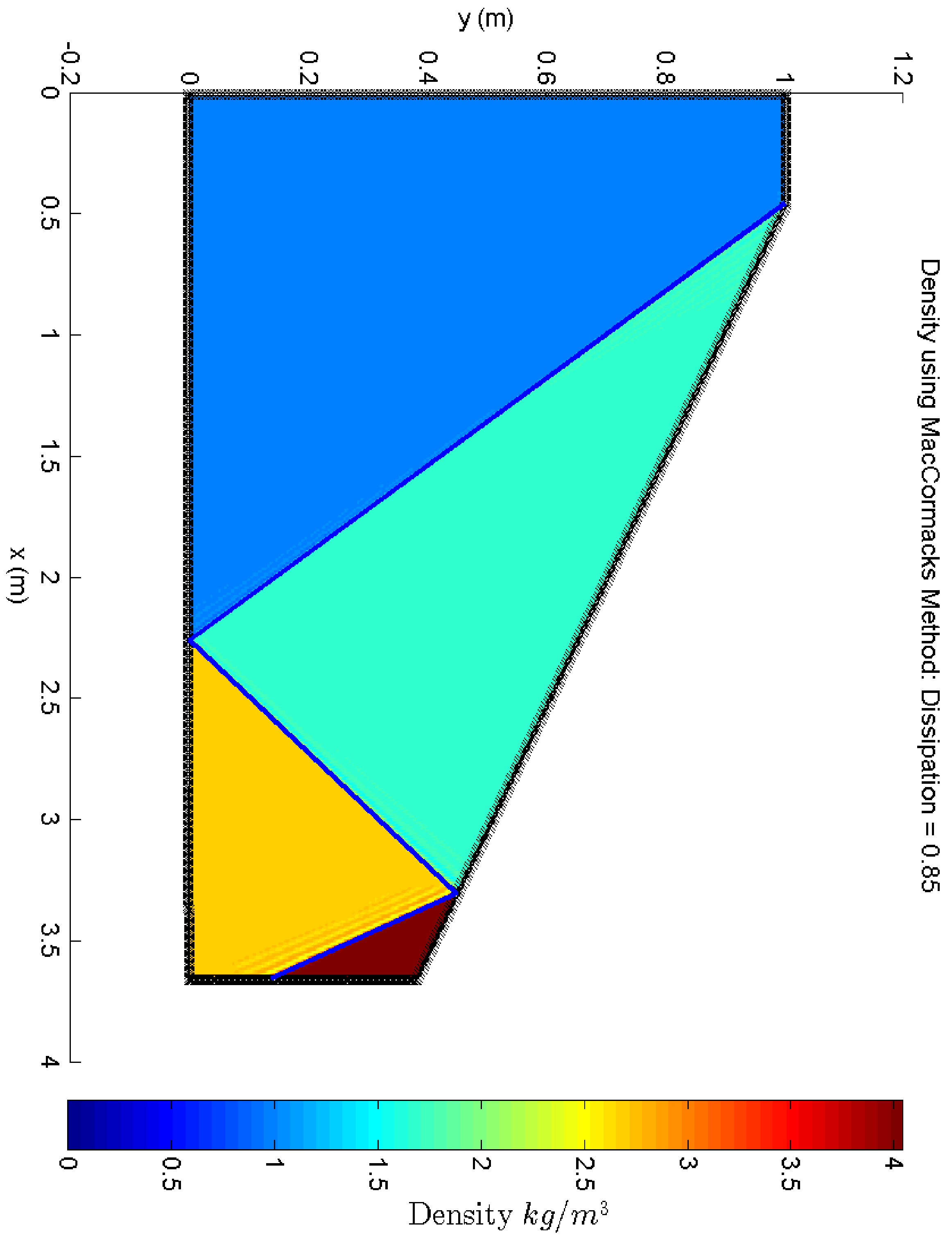


Figure 30: This figure shows the numerical solution for density on the 8x grid. The exact shocks are shown as superimposed blue lines.