INDEX

Introduction

The aim of this project is to simulate propagation of a shock inside a shock tube and find steady state solutions a converging-diverging laval nozzle for subsonic and transonic cases. These 2 problems are well researched problems and their exact solutions have been well tabulated in various literatures. Shock tube finds its application in replicating conditions in turbine sections and jet engines and these tubes when connected with an added nozzle at the end can be used to simulate atmospheric re-entry of spacecraft. The laval nozzles are used extensively in steam turbines in power plants, in rockets for providing sufficient thrust and in supersonic gas turbines. For the shock tube we have a diaphragm sitting in the middle of the tube(x=5) and we have a high pressure and density on the left side of the diaphragm and low pressure and density on the right side. At time t=0, the diaphragm breaks and the high gradients at x=5 causes a shock wave to propagate to the right side and an expansion fan towards the left. This is a standard method that is used in actual shock tubes to create shock waves. We have a high reynold's number flow and therefore we have an compressible and inviscid flow. We consider a 1-D flow along X-direction. This reduces the full NS equations into the following set of PDEs. Since the flow is compressible we need to consider the energy equation and therefore we need extra constitutive relations which are also stated below

$$\frac{\partial Q}{\partial t} + \frac{\partial F(Q)}{\partial x} = g$$

$$Q = \begin{bmatrix} \rho S \\ \rho u S \\ \rho E S \end{bmatrix} \qquad F = \begin{bmatrix} \rho u S \\ (\rho u^2 + p)S \\ \rho u H S \end{bmatrix} \qquad g = \begin{bmatrix} 0 \\ p\frac{dS}{dx} \\ 0 \end{bmatrix}$$

$$p = (\gamma - 1)(\rho E - \tfrac{1}{2}\rho u^2) \qquad H = E + \tfrac{p}{\rho} = c_V T + \tfrac{1}{2}u^2 + \tfrac{p}{\rho}$$

Here S is the shape of the tube/nozzle. S=1 for the tube. The geometry of the nozzle will be discussed later in this section.

Initial and boundary conditions for the shock tube:-

We have two separate initial conditions for the tube, on for *x<5* and another for *x>5*, which is as follows:-

$$\left.\begin{aligned}\rho_L &= 1.0\ \text{kg/m}^3\\ u_L &= 0.0\ \text{m/s}\\ p_L &= 1.0\times10^5\ \text{Pa}\end{aligned}\right\}\ 0\le x\le 5 \qquad \left.\begin{aligned}\rho_R &= 0.125\ \text{kg/m}^3\\ u_R &= 0.0\ \text{m/s}\\ p_R &= 1.0\times10^4\ \text{Pa}\end{aligned}\right\}\ 5< x\le 10$$

We run the simulation for a small time, so that the shock wave and the expansion fans don't reach the end of the tubes and therefore it is assumed that we have fixed conditions at either boundaries. Where left boundary has all the properties fixed same as the initial condition on the left side and right boundary has all the properties as the initial condition on the right.

Boundary conditions for the shock tube:-

The shape for the C-D nozzle is given by the following equations.

$$S(x)=\begin{cases}1+\frac{3}{2}\left(1-\frac{1}{5}x\right)^2, & 0\le x\le 5\\ 1+\frac{1}{2}\left(1-\frac{1}{5}x\right)^2, & 5< x\le 10\end{cases}$$

We are solving for the steady state solution for this problem. We have the boundary conditions given for both the boundaries for 2 cases, a) Subsonic, b) Transonic. Following are the BCs used for these cases.

a) Subsonic

$$\begin{aligned}\rho_1 &= 1.1409\ \text{kg/m}^3 & \rho_2 &= 1.1008\ \text{kg/m}^3\\ u_1 &= 65.451\ \text{m/s} & u_2 &= 113.060\ \text{m/s}\\ p_1 &= 9.7534\times10^4\ \text{Pa} & p_2 &= 9.2772\times10^4\ \text{Pa}\end{aligned}$$

$$\begin{cases}p_{total} = 1.0\times10^5\ \text{Pa}\\ T_{total} = 300\ \text{K}\\ p_{back} = 9.2773\times10^4\ \text{Pa}\end{cases}$$

b) Transonic

<u>**CASE 2**</u> (Transonic Nozzle)

$$\begin{aligned}\rho_1 &= 1.1308\ \text{kg/m}^3 & \rho_2 &= 1.0275\ \text{kg/m}^3\\ u_1 &= 80.054\ \text{m/s} & u_2 &= 150.535\ \text{m/s}\\ p_1 &= 9.6328\times10^4\ \text{Pa} & p_2 &= 8.4974\times10^4\ \text{Pa}\end{aligned} \quad \text{or} \quad \begin{cases}p_{total} = 1.0\times10^5\ \text{Pa}\\ T_{total} = 300\ \text{K}\\ p_{back} = 8.4974\times10^4\ \text{Pa}\end{cases}$$

Any initial guess can be used and the code has to be run till convergence is achieved, but to achieve convergence faster, a smarter choice is made where we assume that the left half of the nozzle has the same properties as the left boundary and same for the right side. This is similar to what we have in the previous case.

Numerical Scheme

The classical NS devolves into system of nonlinear PDEs with 3 PDEs. Yee-Roe Flux limited scheme is being used for discretization in space. The scheme is 2nd order accurate in space in general, but it reverts back to first order near discontinuities to maintain stability. An explicit euler scheme is used to iterate the solution in time. The schematics are presented as follows:-

$$Q_i^{n+1} = Q_i^n - \Delta t\left(\left[\delta_x F_i\right] - g\right)$$

$$\left[\delta_x F_i\right] = \frac{1}{2\Delta x}\left[F_{i+1} - F_{i-1} - (R^{-1}|\lambda|N)_{i+\frac{1}{2}} + (R^{-1}|\lambda|N)_{i-\frac{1}{2}}\right]$$

$$N_i = M_i - \ \mathrm{minmod}(M_{i-1}, M_i, M_{i+1})$$
$$M_i = R_i(Q_{i+\frac{1}{2}} - Q_{i-\frac{1}{2}})$$

$|\lambda_i| \equiv$ absolute value of the eigenvalues of the Flux Jacobian $\dfrac{\partial F_i}{\partial Q_i}$

$R_i^{-1} \equiv$ left eigenvectors of the Flux Jacobian $\dfrac{\partial F_i}{\partial Q_i}$

$R_i \equiv$ right eigenvectors of the Flux Jacobian $\dfrac{\partial F_i}{\partial Q_i}$

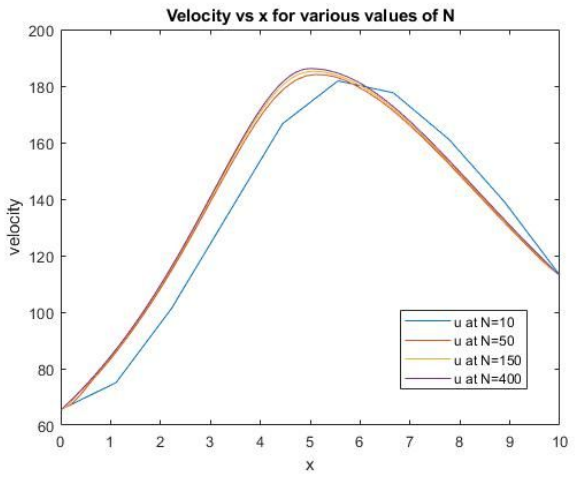
$[\quad]_{i\pm\frac{1}{2}}$ means evaluation at the Roe Average State

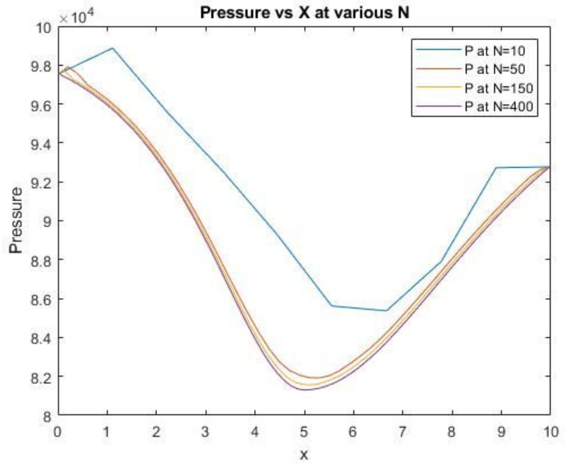
Here the fluxes are calculated by Roe’s approximate Reimann solution. The minmod function here is very smart way of choosing the right points to calculate the fluxes and mainly as it reverts back to first order there is a discontinuity at some point. The scheme helps to properly upwind the system. The minmod function chooses the points to calculate fluxes depending on the nature of characteristics, it chooses the points ahead if all the characteristics are positive while uses points behind if we have negative characteristics. The 2nd order nature of scheme at most of the places gets rid of diffusion error, while proper upwinding helps in getting rid of the “oscillating” dispersive error in the region with huge gradients ergo the scheme is TVD. Also by reverting back to first order near discontinuities the scheme adds some diffusion to smooth out the high gradient across the discontinuity. But the scheme has one disadvantage in terms of having to calculate fluxes at ghost nodes beyond the two boundary nodes. To tackle this, the scheme is made first order at the boundaries and there will be a smarter approach to this later mentioned in the project. For the transonic case, the results of this scheme are compared to the second order scheme without limiter where we use properties only from either of the two sides to calculate the fluxes. Additionally in case of shock tube, the results are also compared with first order scheme as well. The comparison between these is discussed in the sections to come.

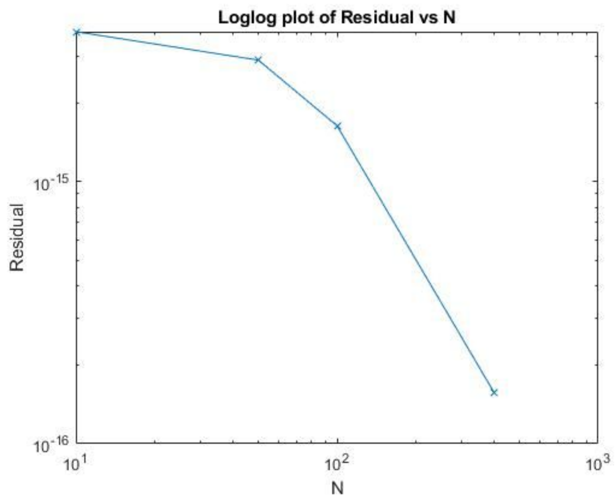
Numerical Results

Case 1 (Subsonic)

First the code was run for a large value of N(=300) and small timestep(10e-5) for 1 seconds to evaluate the velocities accurately, the assumption that the steady state has been achieved is validated by calculating the residual which comes out to be $res = 8.3808 \times 10^{-16}$. The 3 characteristics of this system are u, u+a, u-a, where a is the speed of sound. For the CFL condition the courant number should be calculated for maximum possible value of the three characteristics, it is intuitive that u+a will have the maximum value in this flow. When calculated the maximum value of u+a was found to be max(u+a)=523.1340m/s. This when substituted in courant number gives, the maximum timestep possible as $dt_{max} = dx/523.134$. With CFL=0.5, the maximum residuals were calculated at the steady state and we used 10^{-16} as the tolerance for the residual. *As the value of N increases, the velocity matches closer to the steady state. Same can be observed for the pressure plots. We get reduction in the residuals when you increase N.*







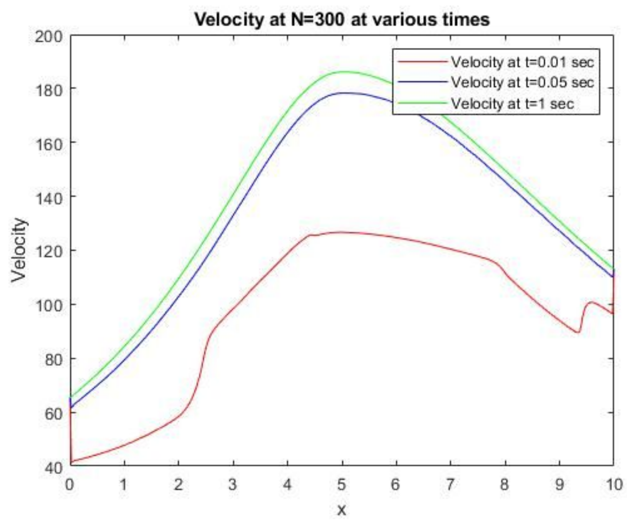
We can see that from the loglog plot on the bottom right corner. The slope ideally should be -2 and we need more values of N to get that result. The residue almost remains constant N=400.

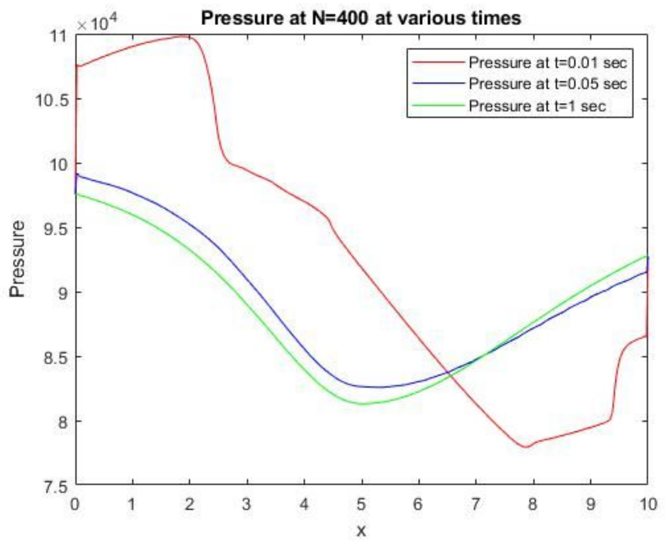
Following are the values of residuals for various values of N

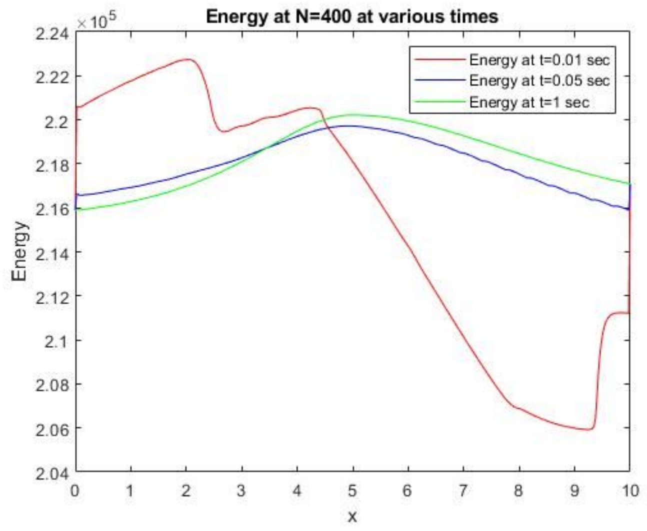
N	Residues
10	3.7367×10^{-11}
50	2.474×10^{-15}
150	1.6612×10^{-15}
400	7.4617×10^{-16}

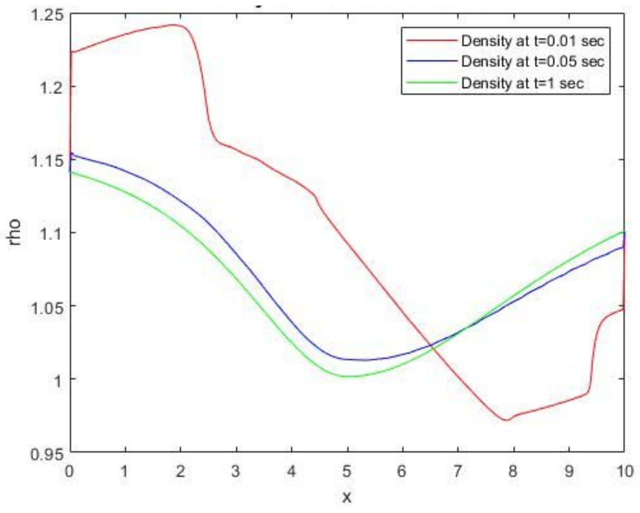
We use N=400 as a round off value.

Following are the plots of solutions for t=0.01, t=0.1 and t=1.





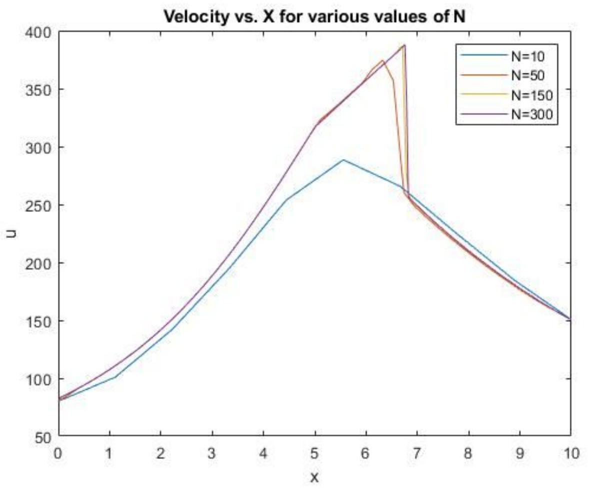


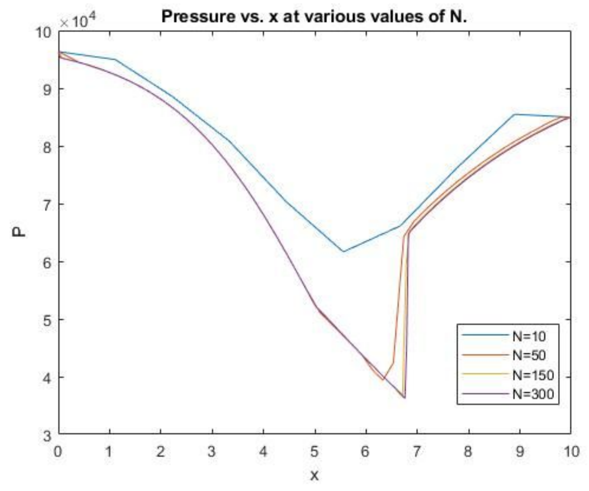


Numerical Results.

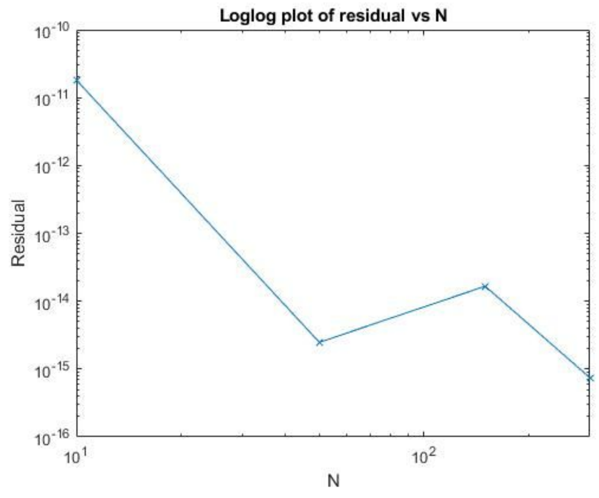
Case 3 (Transonic)

First the code was run for a large value of N(=300) and small timestep(10e-5) for 1 seconds to evaluate the velocities accurately, the assumption that the steady state has been achieved is validated by calculating the residual which comes out to be $res = 4.5592 \times 10^{-18}$. The 3 characteristics of this system are u, u+a, u-a, where a is the speed of sound. For the CFL condition the courant number should be calculated for maximum possible value of the three characteristics, it is intuitive that u+a will have the maximum value in this flow. When calculated the maximum value of u+a was found to be max(u+a)= 688.2959m/s. This when substituted in courant number gives, the maximum timestep possible as $dt_{max} = dx/688.3$. This was then used with various CFL values. With CFL=0.5, maximum residues were calculated at the steady state. It is assumed that a residue of 10^{-15} is tolerated, and the value of N was increased till the residue falls below the tolerance limit. *As expected the velocity reaches closer and closer to the actual ideal state solution as we increase the value of N. Same can be observed for the pressure plots. The code fails to capture features of shock for small N.*





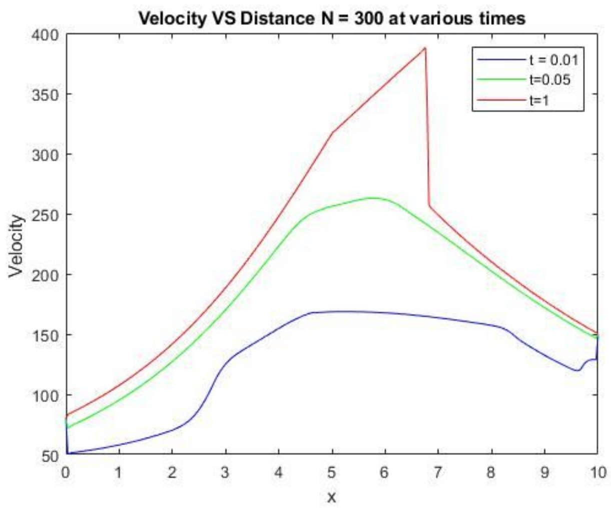
We further plot the loglog plot of the residual vs. N to see that the residual decreases drastically with N. The residue almost remains constant after N=300.

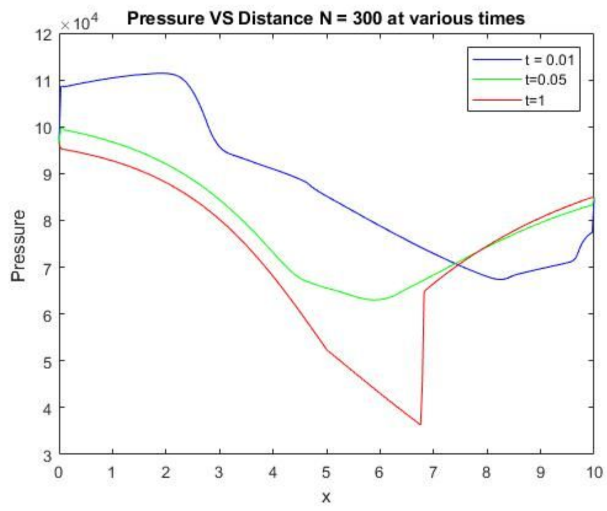


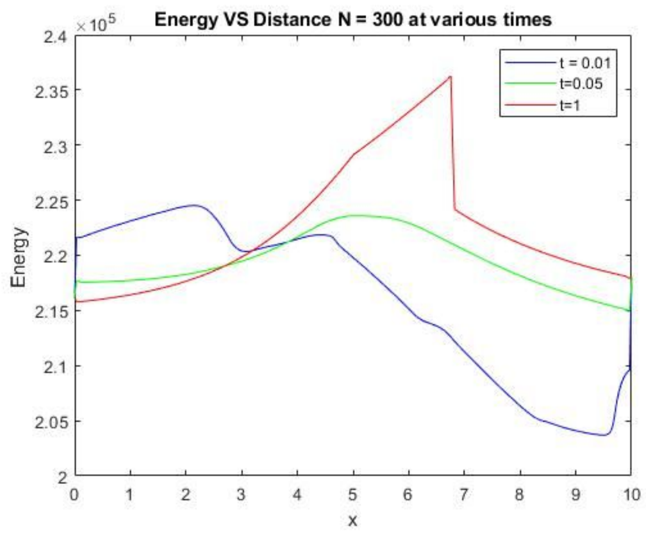
Following is the table for residues at various values of N.

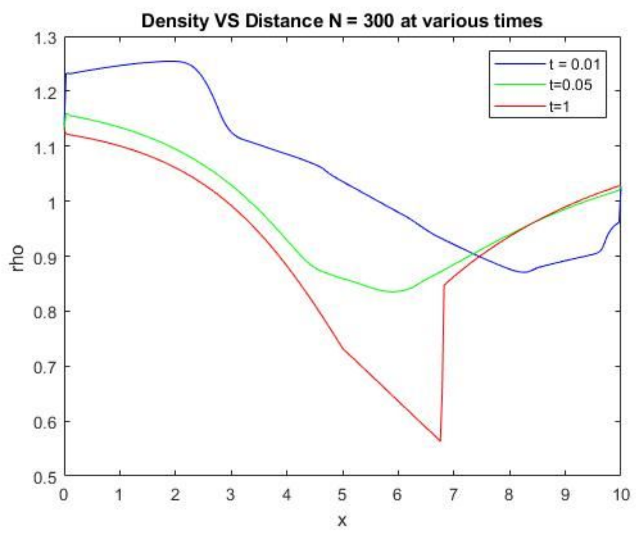
N	Residues
10	1.812×10^{-11}
50	2.474×10^{-15}
150	1.6612×10^{-15}
300	7.4617×10^{-16}

Although the residue reaches less than tolerance between 250 and 300, we round it off to **N=300**. Following are the results at various timesteps. These are the plots with limiter.

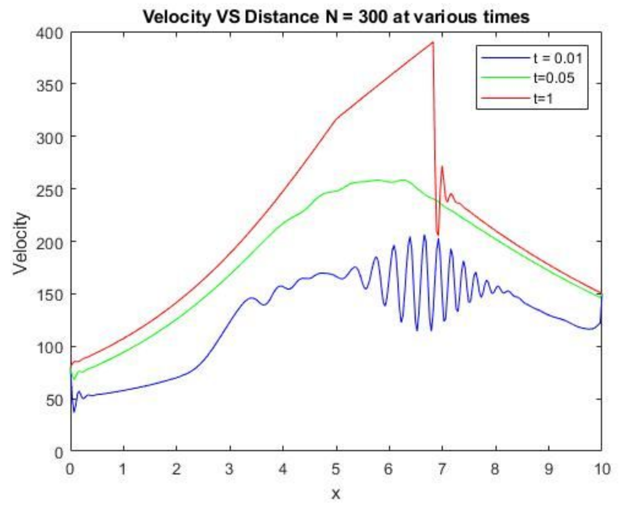


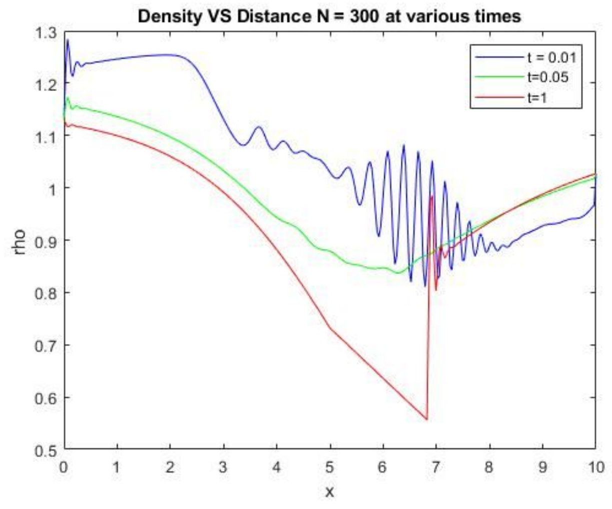


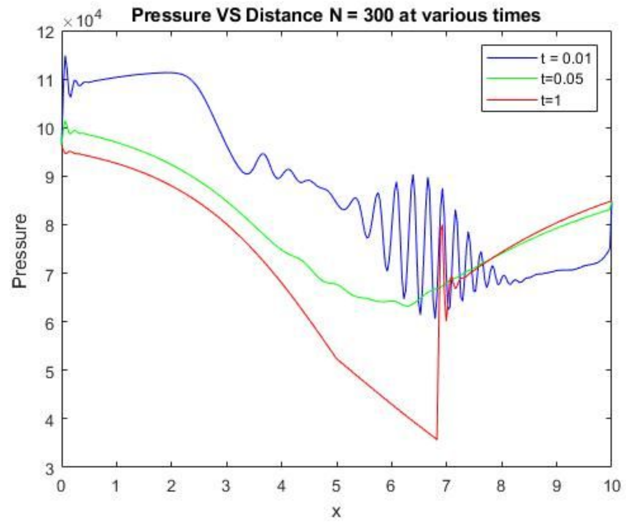


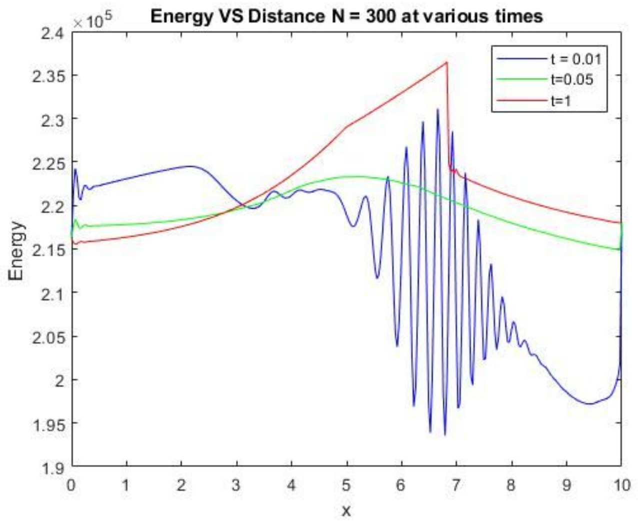


Following are plots for code without the flux limiter, **N=300, CFL=0.5**

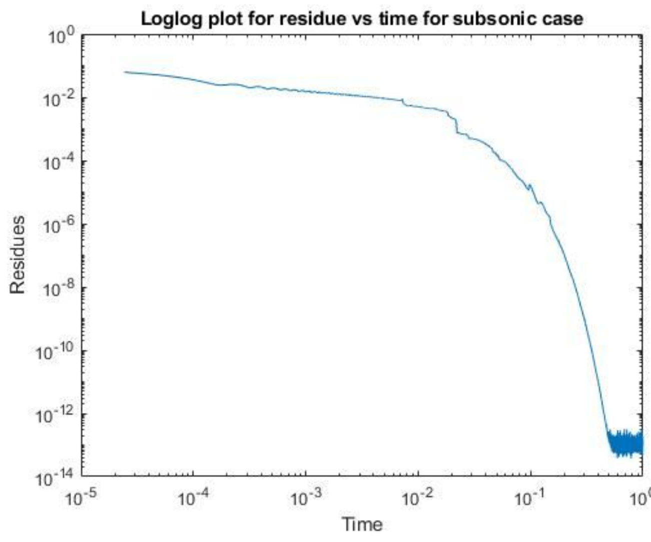


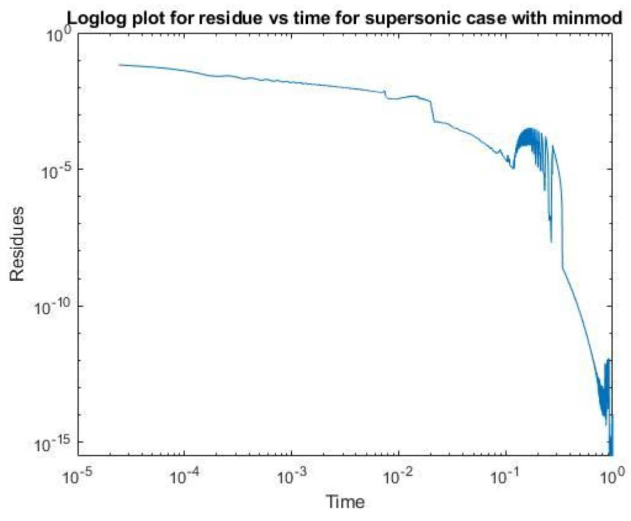






Residuals for subsonic and supersonic





Discussion on the results:

a) **Subsonic:-** There are no discontinuities in the subsonic flow, and thus the minmod never has to revert back to first order and in general there is 2nd order accuracy except at the boundary, this is why no diffusion or dispersion is observed throughout the system. The residues drop at a slow rate initially but it drops much faster as the steady state is approached. The fluctuations towards the end can be attributed to machine zero.

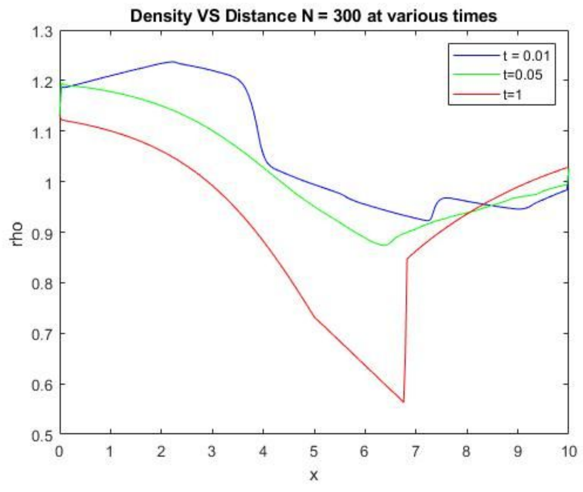
b) **Transonic:-**A stark difference is immediately observed between the results for code with and without limiter. The scheme without limiter is not TVD and the dispersive error is expected and since, there’s a sharp gradient at the shock, the dispersion is very prominent near the shock. Also large dispersion error is observed in initial steps, this can be attributed to discontinuous initial guess given at the mid-point x=5. Also the dispersion is observed on the right side of the shock, because $M_{i-\frac{1}{2}}$ was considered as the flux instead of the minmod, the dispersion would have been on the left if $M_{i+\frac{1}{2}}$ was considered as the flux. For the solution with minmod, small amount of diffusion is observed at the shock for small values of N as the scheme reverts back to first order near the discontinuity. The diffusion becomes insignificant for large values of N. This is one more reason to choose large values of N=300, as the increase in N implies reduced dx and dt, the diffusion is directly dependent on these values. The dispersion in the results can’t be tolerated as they are unphysical and violate 2nd law of thermodynamics. We discard it as a viable scheme and make a conscious decision to avoid discussion about the scheme without any limiter.

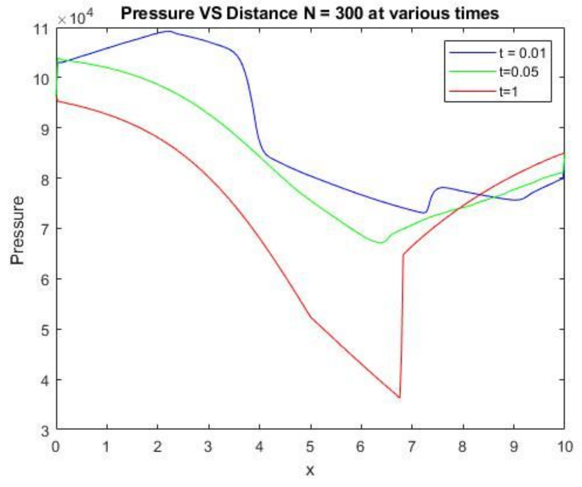
   Talking aboot the residues just like the subsonic case, it drops at a slow rate initially but it drops much faster as the steady state is approached. The fluctuations towards the end can be attributed to machine zero.

Experimentation with CFL:-

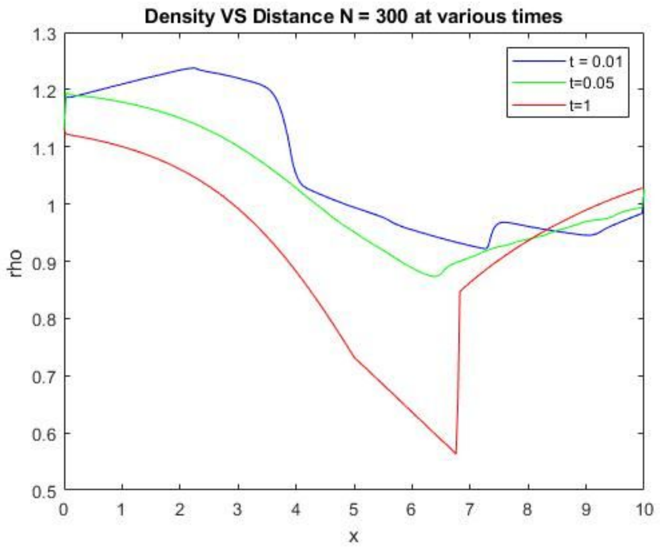
a) **Transonic:-** Density and pressure (these parameters are chosen arbitrarily) are compared for by increasing CFL from 0.5. Following are the plots for the various CFL

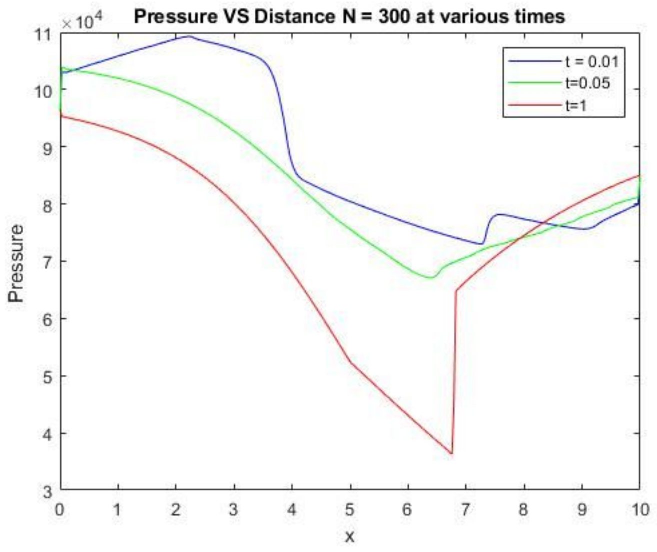
CFL=0.6



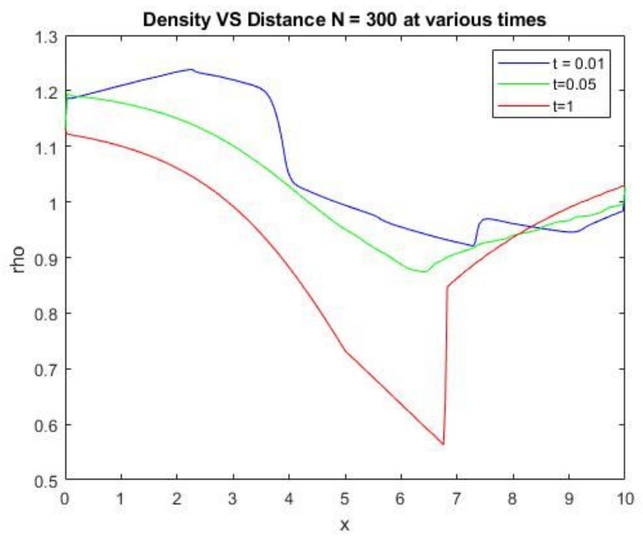


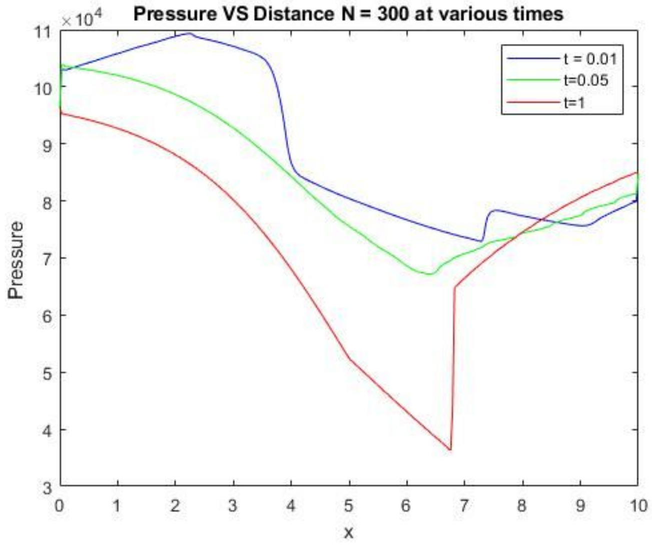
CFL=0.7



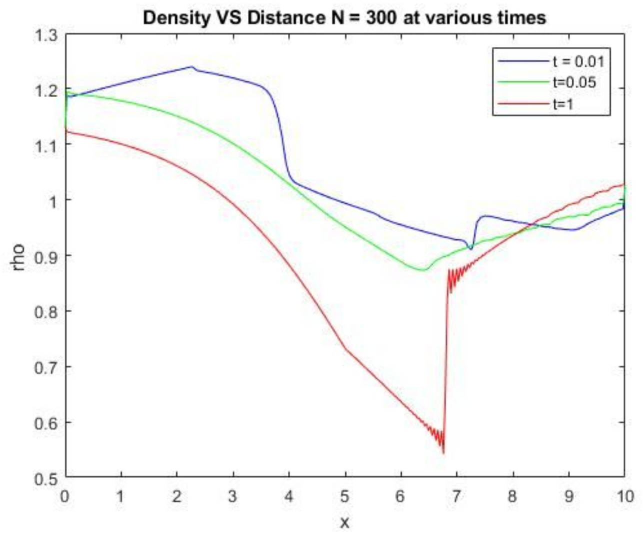


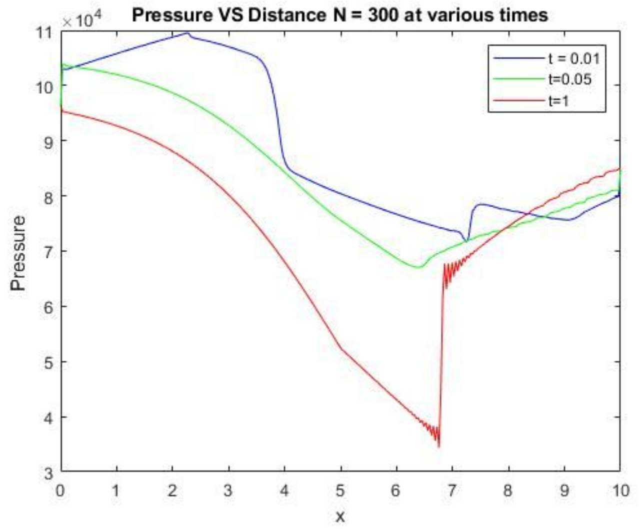
CFL=0.9


Density VS Distance N = 300 at various times
rho
x
t = 0.01
t=0.05
t=1


Pressure VS Distance N = 300 at various times
$\times 10^4$
Pressure
x
t = 0.01
t=0.05
t=1

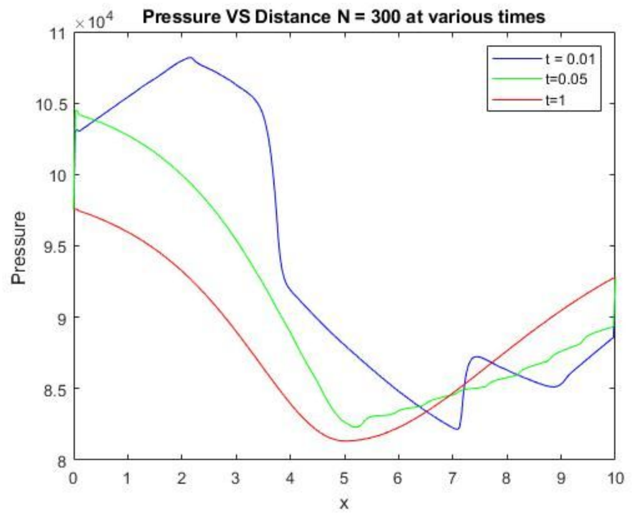
CFL=1.1

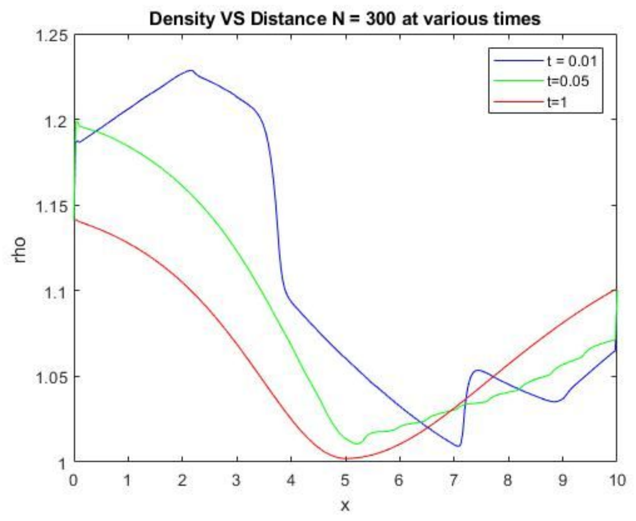

Density VS Distance N = 300 at various times
rho
x
t = 0.01
t=0.05
t=1


Pressure VS Distance N = 300 at various times
$\times 10^4$
Pressure
x
t = 0.01
t=0.05
t=1

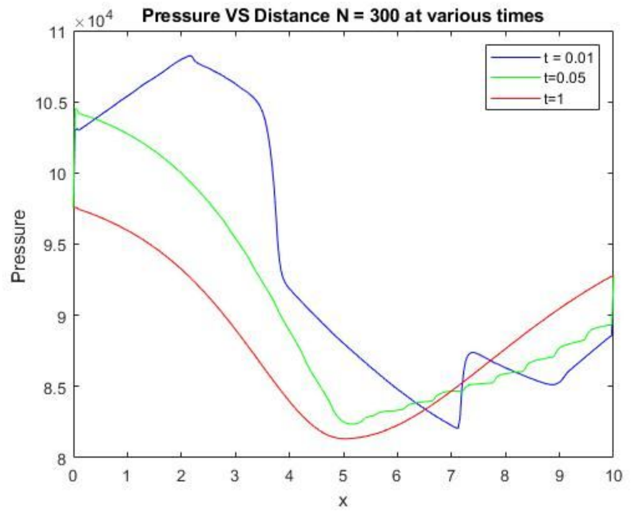
b) Subsonic

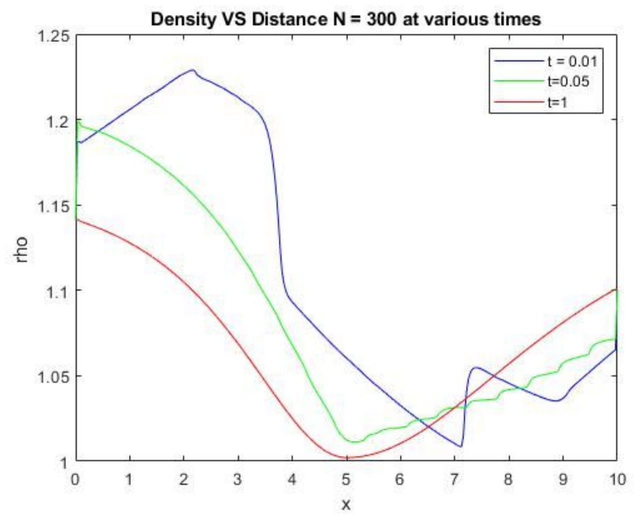
CFL=0.7



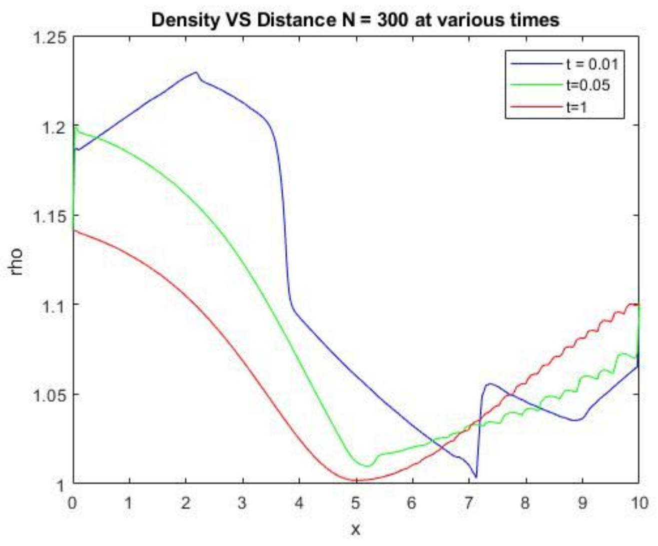


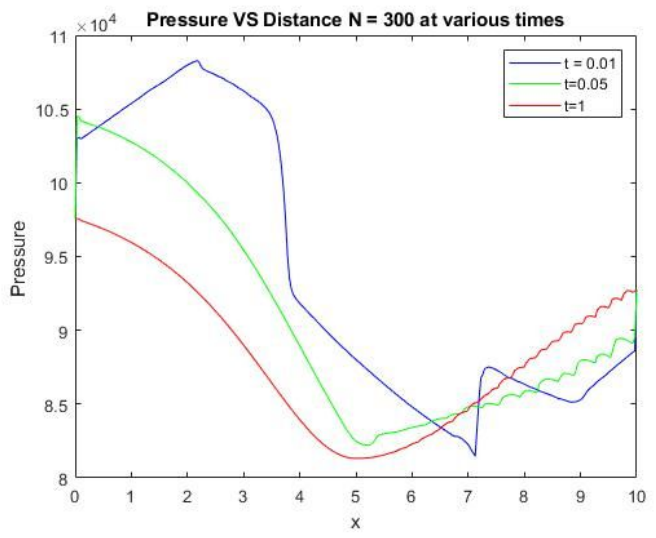
CFL=0.9





CFL=1.1



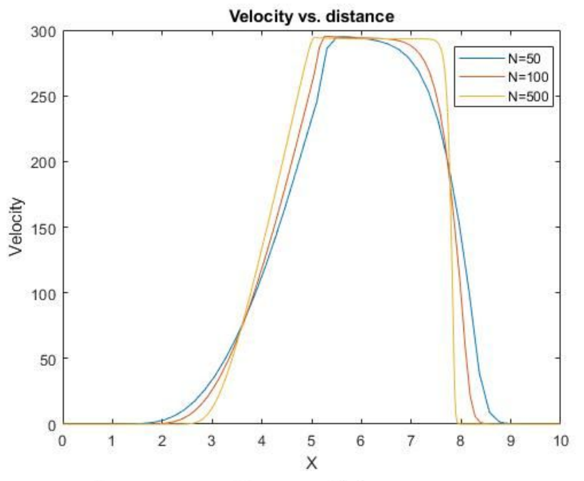


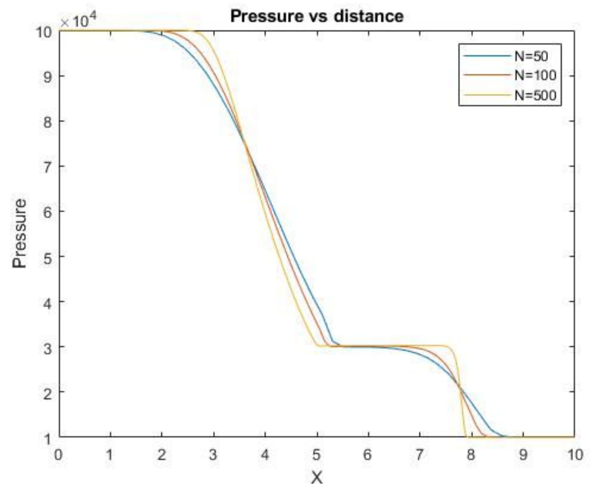
Conclusions:- The courant number predicted for both subsonic and transonic flow was found to be accurate, where the scheme was stable for CFL<1 and showed instability for CFL>1. Reinforcing the CFL condition that dtmax=dx/max(u+a) for both the cases. The plots are considered at different time steps to see how early the scheme shows signs of instability, in both cases for CFL=1.1, the instability is seen for times as small as 0.05 seconds.

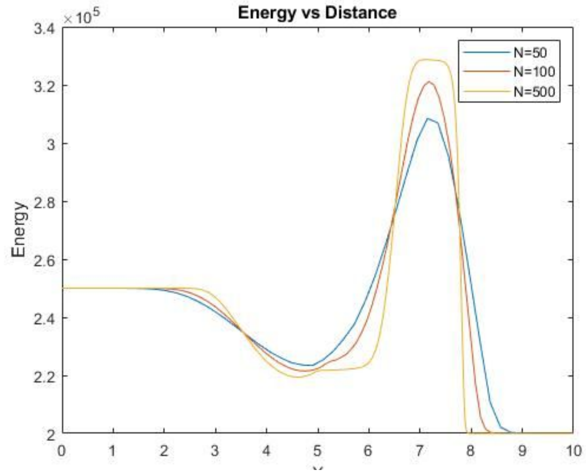
Case 3
Shock Tube

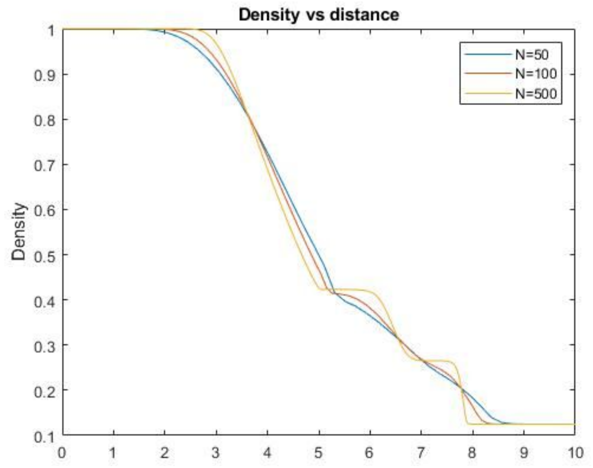
The code is run for 0.005 seconds,the shock reaches almost x=8 and expansion fan reaches almost x=2.75. A condition similar to previous cases is used, courant no. <1. After successfully running the code using N=300 and dt= 10^{-6} to calculate max(u+a) as that can be expected to be the largest value of characteristic and the maximum value of that came out to be 674.8749. This courant number condition was stable with CFL=1 and the code started to blow up at CFL=1.05. Thus we can say that the maximum timestep we can use : dt=dx/(max(u+a))=dx/674.8749. The details of the scheme have been discussed in the previous section.

Following are the plots for shock tube code with limiter

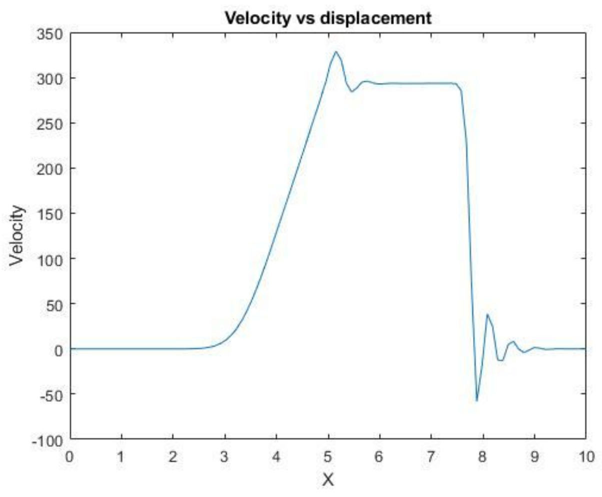


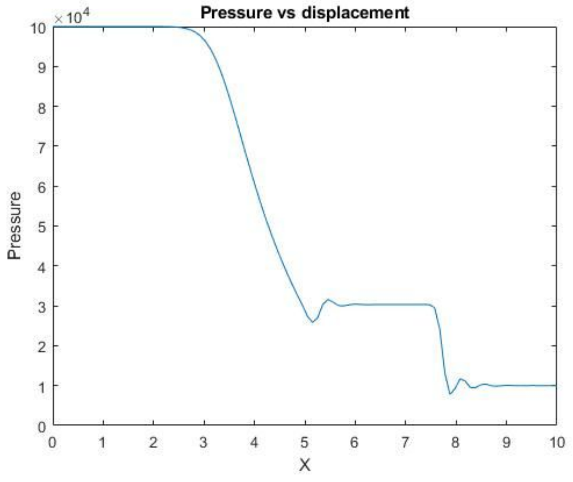


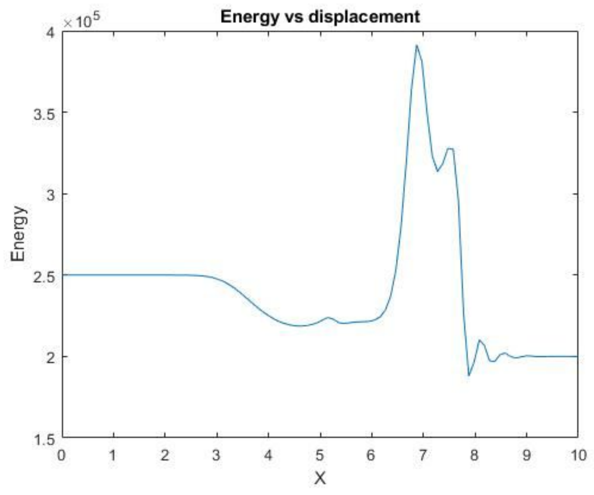


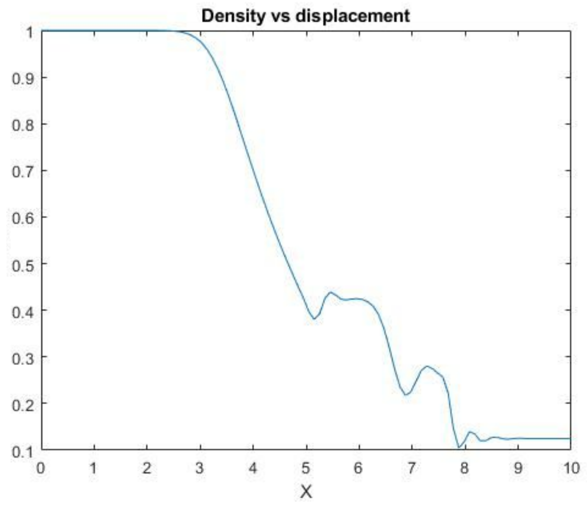


Following are the plots for shock tube without limiter:-

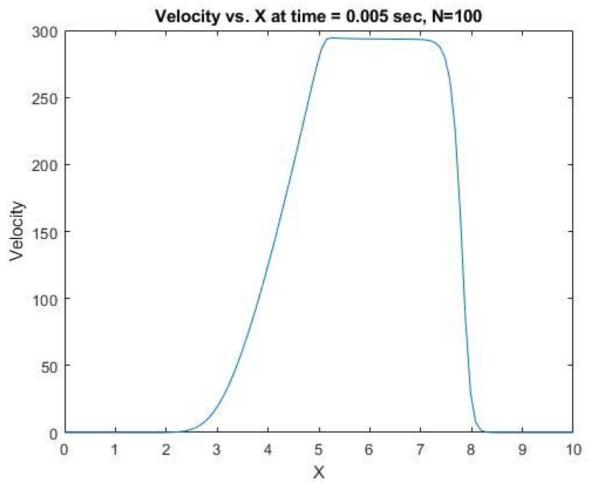


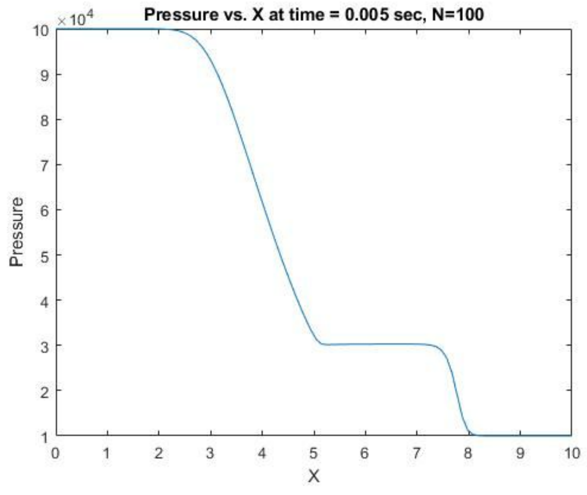


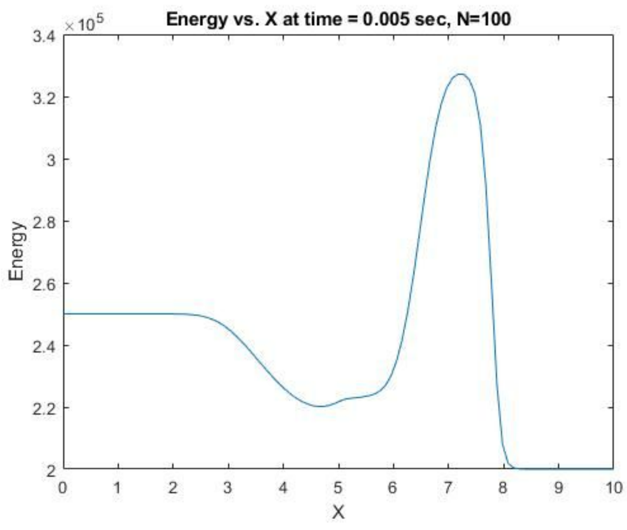


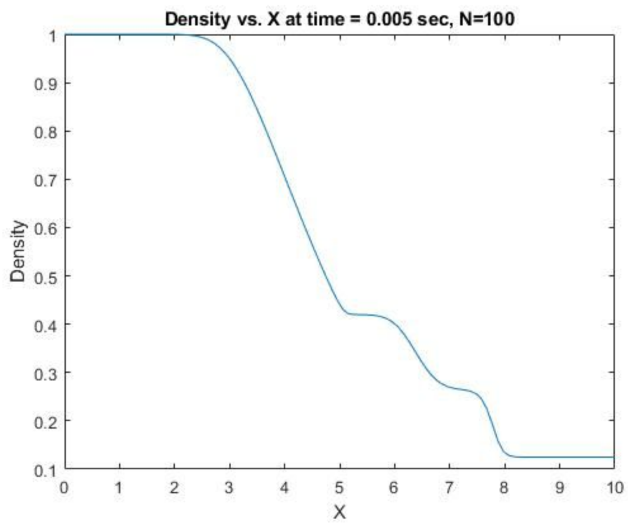


Following are the plots for shock tube by first order scheme









Observations:- Diffusion is observed near shock for the scheme with limiter. The discontinuity at the shock reverts the minmod function to first order and causing diffusive error to smear out the high gradient at the shock. This diffusion is proportional to dx and dt. As the value of N is increased, dx and dt is decreased, decreasing the diffusive error. Sharp gradients in properties are observed for N=500. For the scheme with no limiter as expected dispersion near huge gradients is observed. This can be expected as the scheme isn't TVD. For the first order scheme, diffusion is observed throughout the domain. The entire domain is smeared out and the diffusion is very prominent near the shocks and expansion fans, where the gradients are very steep.

Boundary Conditions

The problem of boundary conditions has been eluded throughout the report. For the scheme with limiter, fluxes from points beyond the boundary are required. This was dealt with initially by ignoring fluxes at points beyond and using fluxes only at 2 points near boundaries. This gives wrong results at the points at the first and last internal nodes. The approach was to use same values upstream and downstream as the boundaries. This gives good results for the shock tube. The boundary was also extended to one point beyond the boundaries on either side. This also gives smoother results at the boundaries and all the results in the report have been taken from this code.

Conclusions

At steady state the flow in nozzle remains subsonic throughout for case 1 and no shock is seen in the domain. For case 2, a shock is observed around 6.9m. The flow accelerates to supersonic till the shock and then it becomes subsonic. In shock tube, the shock travels down the tube from the middle while the expansion fan travels towards left from the middle.

The solver created for the nozzle and shock tube gives quite accurate results. The solver is capable of handling large gradients and eliminates both diffusive and dispersive errors. The scheme is TVD. The scheme without limiter gives very unphysical results and can be immediately discarded. The first order scheme was actually used to solve for the nozzle but the scheme was unstable, further investigation needs to be done regarding this, although even if it were stable, due to the large gradients huge amount of diffusion would have been observed. Even though, the residuals were low for smaller values of N but a large value of N was used to minimize other errors like the diffusion due to the minmod being first order near the discontinuities. Also the steady state can be reached at earlier times but we choose higher time to achieve higher residual because the residual is very low to start off with. If there were memory and time constraints, decent results can be obtained by using N=150~200 and final time~0.4 to 0.8.

The code for the nozzle can be optimized by not storing all the variables at every timestep and just at the current and the previous timestep. The code for the shock can be optimized by storing just the values of primitive values of variables and storing the rest of the variables just for previous and current timesteps. These measures were not employed due to lack of time. The boundary conditions were handled in kind of a hacky way there are definitely better ways oot there to deal with this.

All in all flux limiters turn out to be very strong tools for solving steady and transient hyperbolic systems, honestly I am surprised by the level of accuracy. I have matched the results with some of the analytical solutions available online and they match up pretty accurately

CODES

For nozzle

```
%% Project Shock tube 2nd order with and without limiter and corrected BCs
clc
clear all
%% Defining Variables
L = 10;
N = 300;
y = 1.4;
R = 8.314;
dx = (L/(N-1));
%% Variables at left side
% rho_l = 1.1409;
% u_l = 65.451;    % SUBSONIC
% p_l = 9.7534 * 10^4;
%


rho_l = 1.1308;
u_l = 80.054;    % TRANSONIC
p_l = 9.6328 * 10^4;


CFL=0.5;
%delt=CFL*delx/523.1340;%Subsonic
dt = CFL*dx/688.2959;%transonic
tf = 1;% Final time

Nt = fix(tf/dt);
%% Variables at right side
% rho_r = 1.0275;
% u_r = 150.535;% Subsonic
% p_r = 8.4974 * 10^4;
```

```
rho_r = 1.1008;
u_r = 113.060;%Transonic
p_r = 9.2772 * 10^4;

E_l = ( p_l/(y-1) + 0.5*rho_l*(u_l^2) )/rho_l;
E_r = ( p_r/(y-1) + 0.5*rho_r*(u_r^2) )/rho_r;

H_l = E_l + p_l/rho_l;
H_r = E_r + p_r/rho_r;

u = zeros(N,Nt);
rho = zeros(N,Nt);
p = zeros(N,Nt);
E = zeros(N,Nt);
H = zeros(N,Nt);
Q1 = zeros(N,Nt);
Q2 = zeros(N,Nt);
Q3 = zeros(N,Nt);
F1 = zeros(N,Nt);
F2 = zeros(N,Nt);
F3 = zeros(N,Nt);
S=zeros(N,1);
dS=zeros(N,1);
g2=zeros(N,Nt);
%% Boundary conditions for primitive variables
x=0:dx:10;

for i=1:N/2
    rho(i,1) = rho_l;
    u(i,1) = u_l;
    p(i,1) = p_l;
    H(i,1) = E_l + p_l/rho_l;
    E(i,1) = E_l;
```

```
        S(i)=1+1.5*(0.2*x(i)-1)^2;
        dS(i)=3*0.2*(0.2*x(i)-1);
        g2(i,1)=p(i)*dS(i);
end

for i=N/2+1:N
        rho(i,1) = rho_r;
        u(i,1) = u_r;
        p(i,1) = p_r;
        H(i,1) =  E_r + p_r/rho_r;
        E(i,1) = E_r;
        S(i)=1+0.5*(0.2*x(i)-1)^2;
        dS(i)=0.2*(0.2*x(i)-1);
        g2(i,1)=p(i)*dS(i);
end
%% Initial condition for primitive variables
u(1,:) = u_l;
u(N,:) = u_r;
rho(1,:) = rho_l;
rho(N,:) = rho_r;
p(1,:) = p_l;
p(N,:) = p_r;
E(1,:) = E_l;
E(N,:) = E_r;
H(1,:) = H_l;
H(N,:) = H_r;

%% IC and BC for Q
 Q1(:,1) = rho(:,1) .* S;
 Q2(:,1) = rho(:,1).*u(:,1).*S;
 Q3(:,1) = rho(:,1).*E(:,1).*S;

 Q1(1,:) = rho(1,:) .* S(1);
```

```
Q2(1,:) = rho(1,:).*u(1,:).*S(1);
Q3(1,:) = rho(1,:).*E(1,:).*S(1);

Q1(N,:) = rho(N,:) .* S(N);
Q2(N,:) = rho(N,:).*u(N,:).*S(N);
Q3(N,:) = rho(N,:).*E(N,:).*S(N);

%% IC and BC for F
F1(:,1) = rho(:,1).*u(:,1).*S;
F2(:,1) = ( rho(:,1).*(u(:,1).^2) + p(:,1) ) .* S;
F3(:,1) = rho(:,1).*u(:,1).*H(:,1).*S;

F1(1,:) = rho(1,:).*u(1,:)*S(1);
F2(1,:) = ( rho(1,:).*(u(1,:).^2) + p(1,:) ) .* S(1);
F3(1,:) = rho(1,:).*u(1,:).*H(1,:).*S(1);

F1(N,:) = rho(N,:).*u(N,:).*S(N);
F2(N,:) = ( rho(N,:).*(u(N,:).^2) + p(N,:) ) .* S(N);
F3(N,:) = rho(N,:).*u(N,:).*H(N,:).*S(N);




for k=1:Nt-1
   for i=2:N-1



        if i==2
       %% Roe averaging
       rho_plus2= sqrt(rho(i+2,k)*rho(i+1,k));%rho at i+3/2
       rho_plus1 = sqrt(rho(i,k)*rho(i+1,k));%rho at i+1/2
       rho_minus1 = sqrt(rho(i,k)*rho(i-1,k));%rho at i-1/2
       rho_minus2 = rho_l;%rho at i-3/2
```

```
        u_plus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i+1,k) )*u(i+1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i+1,k))
);%Velocity at i+1/2
        u_plus2 = ( sqrt(rho(i+2,k))*u(i+2,k) + sqrt(rho(i+1,k))*u(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+3/2
        u_minus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i-1,k))
);%Velocity at i-1/2
        u_minus2 = u_I;%Velocity at i-3/2
        H_plus2 = ( sqrt(rho(i+2,k))*H(i+2,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%H at i+3/2
        H_plus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i+1,k))
);%H at i+1/2
        H_minus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i-1,k))
);%H at i-1/2
        H_minus2 = H_I;%H at i-3/2
        a_plus2 = sqrt( (y-1)*(H_plus2 - 0.5*(u_plus2^2)) );
        a_plus1 = sqrt( (y-1)*(H_plus1 - 0.5*(u_plus1^2)) );
        a_minus1 = sqrt( (y-1)*(H_minus1 - 0.5*(u_minus1^2)) );
        a_minus2 = sqrt( (y-1)*(H_minus2 - 0.5*(u_minus2^2)) );
        %% R inverse
        Rinv_plus = [1 (rho_plus1/(sqrt(2)*a_plus1)) (rho_plus1/(sqrt(2)*a_plus1));u_plus1
((u_plus1 + a_plus1)*(rho_plus1/(sqrt(2)*a_plus1))) ((u_plus1 -
a_plus1)*(rho_plus1/(sqrt(2)*a_plus1)));(0.5*(u_plus1^2))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) + a_plus1*u_plus1))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) - a_plus1*u_plus1))];
        Rinv_minus = [1 (rho_minus1/(sqrt(2)*a_minus1))
(rho_minus1/(sqrt(2)*a_minus1));u_minus1 ((u_minus1 +
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1))) ((u_minus1 -
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1)));(0.5*(u_minus1^2))
((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) + (a_minus1^2)/(y-1) +
a_minus1*u_minus1)) ((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) +
(a_minus1^2)/(y-1) - a_minus1*u_minus1))];
        %% R at i+3/2
```

```
    R_plus2 = [(1 - 0.5*(y-1)*(u_plus2^2)/(a_plus2^2)) ((y-1)*u_plus2/(a_plus2^2))
(1-y)/(a_plus2^2);((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) - a_plus2*u_plus2))
((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 - u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1));((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) +
a_plus2*u_plus2 )) -((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 + u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1))];
    %% R at i+1/2
    R_plus1 = [(1 - 0.5*(y-1)*(u_plus1^2)/(a_plus1^2)) ((y-1)*u_plus1/(a_plus1^2))
(1-y)/(a_plus1^2);((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) - a_plus1*u_plus1))
((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 - u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1));((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) +
a_plus1*u_plus1 )) -((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 + u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1))];
    %% R at i-1/2
    R_minus1 = [(1 - 0.5*(y-1)*(u_minus1^2)/(a_minus1^2)) ((y-1)*u_minus1/(a_minus1^2))
(1-y)/(a_minus1^2);((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_minus1^2) -
a_minus1*u_minus1)) ((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 - u_minus1*(y-1)))
((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1));((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_min
us1^2) + a_minus1*u_minus1 )) -((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 +
u_minus1*(y-1))) ((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1))];

    %% R at i-3/2
    R_minus2 = [(1 - 0.5*(y-1)*(u_minus2^2)/(a_minus2^2)) ((y-1)*u_minus2/(a_minus2^2))
(1-y)/(a_minus2^2);((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_minus2^2) -
a_minus2*u_minus2)) ((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 - u_minus2*(y-1)))
((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1));((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_min
us2^2) + a_minus2*u_minus2 )) -((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 +
u_minus2*(y-1))) ((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1))];
    %% eigenmatrix
    L_plus = [abs(u_plus1) 0 0;0 abs(u_plus1 + a_plus1) 0;0 0 abs(u_plus1 - a_minus1)];
    L_minus = [abs(u_minus1) 0 0;0 abs(u_minus1 + a_minus1) 0;0 0 abs(u_minus1 -
a_minus1)];
```

```
        Q_plus2 = [Q1(i+2,k)-Q1(i+1,k);Q2(i+2,k)-Q2(i+1,k);Q3(i+2,k)-Q3(i+1,k)];
        Q_plus1 = [Q1(i+1,k)-Q1(i,k);Q2(i+1,k)-Q2(i,k);Q3(i+1,k)-Q3(i,k)];
        Q_minus1 = [Q1(i,k)-Q1(i-1,k);Q2(i,k)-Q2(i-1,k);Q3(i,k)-Q3(i-1,k)];
        Q_minus2 =
[Q1(i-1,k)-rho_l*(1+1.5*(-0.2*dx-1)^2);Q2(i-1,k)-rho_l*u_l*(1+1.5*(-0.2*dx-1)^2);Q3(i-1,k)-rho_l*E
_l*(1+1.5*(-0.2*dx-1)^2)];

        M_plus2=R_plus2*Q_plus2;%M at i+3/2
        M_plus1=R_plus1*Q_plus1;%M at i+1/2
        M_minus1=R_minus1*Q_minus1;%M at i-1/2
        M_minus2=R_minus2*Q_minus2;% M at i-3/2
        for v=1:3
% With minmod
%            N_plus(v,1)=M_plus1(v)-minmod(M_minus1(v),M_plus1(v),M_plus2(v));%N i+1/2
%            N_minus(v,1)=M_minus1(v)-minmod(M_minus2(v),M_minus1(v),M_plus1(v));%N at
i-1/2, ignoring the M at the -1 node
%           Without minmod
            N_plus(v,1)=M_plus1(v)-M_minus1(v);%N i+1/2
            N_minus(v,1)=M_minus1(v)-M_minus2(v);%N at i-1/2, ignoring the M at the -1 node
        end
        AQ_plus = Rinv_plus*L_plus*N_plus;
        AQ_minus = Rinv_minus*L_minus*N_minus;
        Q1(i,k+1) = Q1(i,k) - dt * 0.5 * ( F1(i+1,k) - F1(i-1,k) - AQ_plus(1) + AQ_minus(1))/dx;
        Q2(i,k+1) = Q2(i,k) - dt * (-g2(i,1)+0.5 * ( F2(i+1,k) - F2(i-1,k) - AQ_plus(2) +
AQ_minus(2)))/dx;
        Q3(i,k+1) = Q3(i,k) - dt * 0.5 * ( F3(i+1,k) - F3(i-1,k) - AQ_plus(3) + AQ_minus(3))/dx;
        %% Extracting the primitive variables
        rho(i,k+1) =(Q1(i,k+1))./S(i);
        u(i,k+1) = Q2(i,k+1)./(rho(i,k+1).*S(i));
        E(i,k+1) = Q3(i,k+1)./(rho(i,k+1).*S(i));
        p(i,k+1) = (y-1) * ( rho(i,k+1).*E(i,k+1) - 0.5*rho(i,k+1).*(u(i,k+1).^2) );
        H(i,k+1) = E(i,k+1) + p(i,k+1)./rho(i,k+1);
        g2(i,k+1)=p(3:N-2,k+1)*dS(3:N-2);
        F1(i,k+1) = rho(i,k+1).*u(i,k+1).*S(i);
        F2(i,k+1) = ( rho(i,k+1).*(u(i,k+1).^2) + p(i,k+1) ) .* S(i);
        F3(i,k+1) = rho(i,k+1).*u(i,k+1).*H(i,k+1).*S(i);



         elseif i==N-1
        %% Roe averaging
        rho_plus2= rho_r;%rho at i+3/2
        rho_plus1 = sqrt(rho(i,k)*rho(i+1,k));%rho at i+1/2
        rho_minus1 = sqrt(rho(i,k)*rho(i-1,k));%rho at i-1/2
        rho_minus2 = sqrt(rho(i-2,k)*rho(i-1,k));%rho at i-3/2
        u_plus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i+1,k) )*u(i+1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i+1,k))
);%Velocity at i+1/2
        u_plus2 = rho_r;%Velocity at i+3/2
        u_minus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i-1,k))
);%Velocity at i-1/2
        u_minus2 = ( sqrt(rho(i-2,k))*u(i-2,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-/2
        H_plus2 =H_r;%H at i+3/2
        H_plus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i+1,k))
);%H at i+1/2
        H_minus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i-1,k))
);%H at i-1/2
        H_minus2 = ( sqrt(rho(i-2,k))*H(i-2,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%H at i-3/2
        a_plus2 = sqrt( (y-1)*(H_plus2 - 0.5*(u_plus2^2)) );
        a_plus1 = sqrt( (y-1)*(H_plus1 - 0.5*(u_plus1^2)) );
        a_minus1 = sqrt( (y-1)*(H_minus1 - 0.5*(u_minus1^2)) );
        a_minus2 = sqrt( (y-1)*(H_minus2 - 0.5*(u_minus2^2)) );
        %% R inverse
        Rinv_plus = [1 (rho_plus1/(sqrt(2)*a_plus1)) (rho_plus1/(sqrt(2)*a_plus1));u_plus1
((u_plus1 + a_plus1)*(rho_plus1/(sqrt(2)*a_plus1))) ((u_plus1 -
a_plus1)*(rho_plus1/(sqrt(2)*a_plus1)));(0.5*(u_plus1^2))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) + a_plus1*u_plus1))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) - a_plus1*u_plus1))];
    Rinv_minus = [1 (rho_minus1/(sqrt(2)*a_minus1))
(rho_minus1/(sqrt(2)*a_minus1));u_minus1 ((u_minus1 +
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1))) ((u_minus1 -
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1)));(0.5*(u_minus1^2))
((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) + (a_minus1^2)/(y-1) +
a_minus1*u_minus1)) ((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) +
(a_minus1^2)/(y-1) - a_minus1*u_minus1))];
    %% R at i+3/2
    R_plus2 = [(1 - 0.5*(y-1)*(u_plus2^2)/(a_plus2^2)) ((y-1)*u_plus2/(a_plus2^2))
(1-y)/(a_plus2^2);((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) - a_plus2*u_plus2))
((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 - u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1));((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) +
a_plus2*u_plus2 )) -((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 + u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1))];
    %% R at i+1/2
    R_plus1 = [(1 - 0.5*(y-1)*(u_plus1^2)/(a_plus1^2)) ((y-1)*u_plus1/(a_plus1^2))
(1-y)/(a_plus1^2);((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) - a_plus1*u_plus1))
((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 - u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1));((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) +
a_plus1*u_plus1 )) -((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 + u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1))];
    %% R at i-1/2
    R_minus1 = [(1 - 0.5*(y-1)*(u_minus1^2)/(a_minus1^2)) ((y-1)*u_minus1/(a_minus1^2))
(1-y)/(a_minus1^2);((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_minus1^2) -
a_minus1*u_minus1)) ((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 - u_minus1*(y-1)))
((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1));((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_min
us1^2) + a_minus1*u_minus1 )) -((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 +
u_minus1*(y-1))) ((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1))];

    %% R at i-3/2
```

```
        R_minus2 = [(1 - 0.5*(y-1)*(u_minus2^2)/(a_minus2^2)) ((y-1)*u_minus2/(a_minus2^2))
(1-y)/(a_minus2^2);((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_minus2^2) -
a_minus2*u_minus2)) ((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 - u_minus2*(y-1)))
((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1));((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_min
us2^2) + a_minus2*u_minus2 )) -((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 +
u_minus2*(y-1))) ((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1))];
        %% eigenmatrix
        L_plus = [abs(u_plus1) 0 0;0 abs(u_plus1 + a_plus1) 0;0 0 abs(u_plus1 - a_minus1)];
        L_minus = [abs(u_minus1) 0 0;0 abs(u_minus1 + a_minus1) 0;0 0 abs(u_minus1 -
a_minus1)];

        Q_plus2 =
[rho_r*(1+0.5*(0.2*(10+dx)-1)^2)-Q1(i+1,k);rho_r*u_r*(1+0.5*(0.2*(10+dx)-1)^2)-Q2(i+1,k);rho_r*
E_r*(1+0.5*(0.2*(10+dx)-1)^2)-Q3(i+1,k)];
        Q_plus1 = [Q1(i+1,k)-Q1(i,k);Q2(i+1,k)-Q2(i,k);Q3(i+1,k)-Q3(i,k)];
        Q_minus1 = [Q1(i,k)-Q1(i-1,k);Q2(i,k)-Q2(i-1,k);Q3(i,k)-Q3(i-1,k)];
        Q_minus2 = [Q1(i-1,k)-Q1(i-2,k);Q2(i-1,k)-Q2(i-2,k);Q3(i-1,k)-Q3(i-2,k)];

        M_plus2=R_plus2*Q_plus2;%M at i+3/2
        M_plus1=R_plus1*Q_plus1;%M at i+1/2
        M_minus1=R_minus1*Q_minus1;%M at i-1/2
        M_minus2=R_minus2*Q_minus2;% M at i-3/2
        for v=1:3
          %With minmod
%            N_plus(v,1)=M_plus1(v)-minmod(M_minus1(v),M_plus1(v),M_plus2(v));%N i+1/2
%           N_minus(v,1)=M_minus1(v)-minmod(M_minus2(v),M_minus1(v),M_plus1(v));%N
%           at i-1/2 %
          %Without minmod
          N_plus(v,1)=M_plus1(v)-M_minus1(v);%N i+1/2
          N_minus(v,1)=M_minus1(v)-M_minus2(v);%N at i-1/2, ignoring the M at the -1 node
        end
        AQ_plus = Rinv_plus*L_plus*N_plus;
        AQ_minus = Rinv_minus*L_minus*N_minus;
        Q1(i,k+1) = Q1(i,k) - dt * 0.5 * ( F1(i+1,k) - F1(i-1,k) - AQ_plus(1) + AQ_minus(1))/dx;
        Q2(i,k+1) = Q2(i,k) - dt * 0.5 * ( F2(i+1,k) - F2(i-1,k) - AQ_plus(2) + AQ_minus(2))/dx;
        Q3(i,k+1) = Q3(i,k) - dt * 0.5 * ( F3(i+1,k) - F3(i-1,k) - AQ_plus(3) + AQ_minus(3))/dx;
        %% Extracting the primitive variables
        rho(i,k+1) =(Q1(i,k+1))./S(i);
        u(i,k+1) = Q2(i,k+1)./(rho(i,k+1).*S(i));
        E(i,k+1) = Q3(i,k+1)./(rho(i,k+1).*S(i));
        p(i,k+1) = (y-1) * ( rho(i,k+1).*E(i,k+1) - 0.5*rho(i,k+1).*(u(i,k+1).^2) );
        H(i,k+1) = E(i,k+1) + p(i,k+1)./rho(i,k+1);
        g2(i,k+1)=p(i,k+1)*dS(i);
        F1(i,k+1) = rho(i,k+1).*u(i,k+1).*S(i);
        F2(i,k+1) = ( rho(i,k+1).*(u(i,k+1).^2) + p(i,k+1) ) .* S(i);
        F3(i,k+1) = rho(i,k+1).*u(i,k+1).*H(i,k+1).*S(i);

         else
            %% Roe averaging
        rho_plus2= sqrt(rho(i+2,k)*rho(i+1,k));%rho at i+3/2
        rho_plus1 = sqrt(rho(i,k)*rho(i+1,k));%rho at i+1/2
        rho_minus1 = sqrt(rho(i,k)*rho(i-1,k));%rho at i-1/2
        rho_minus2 = sqrt(rho(i-2,k)*rho(i-1,k));%rho at i-3/2
        u_plus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i+1,k) )*u(i+1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i+1,k))
);%Velocity at i+1/2
        u_plus2 = ( sqrt(rho(i+2,k))*u(i+2,k) + sqrt(rho(i+1,k))*u(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+3/2
        u_minus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i-1,k))
);%Velocity at i-1/2
        u_minus2 = ( sqrt(rho(i-2,k))*u(i-2,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-/2
        H_plus2 = ( sqrt(rho(i+2,k))*H(i+2,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%H at i+3/2
        H_plus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i+1,k))
);%H at i+1/2
```

```
        H_minus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i,k)) + sqrt(rho(i-1,k))
);%H at i-1/2
        H_minus2 = ( sqrt(rho(i-2,k))*H(i-2,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%H at i-3/2
        a_plus2 = sqrt( (y-1)*(H_plus2 - 0.5*(u_plus2^2)) );
        a_plus1 = sqrt( (y-1)*(H_plus1 - 0.5*(u_plus1^2)) );
        a_minus1 = sqrt( (y-1)*(H_minus1 - 0.5*(u_minus1^2)) );
        a_minus2 = sqrt( (y-1)*(H_minus2 - 0.5*(u_minus2^2)) );
        %% R inverse
        Rinv_plus = [1 (rho_plus1/(sqrt(2)*a_plus1)) (rho_plus1/(sqrt(2)*a_plus1));u_plus1
((u_plus1 + a_plus1)*(rho_plus1/(sqrt(2)*a_plus1))) ((u_plus1 -
a_plus1)*(rho_plus1/(sqrt(2)*a_plus1)));(0.5*(u_plus1^2))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) + a_plus1*u_plus1))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) - a_plus1*u_plus1))];
        Rinv_minus = [1 (rho_minus1/(sqrt(2)*a_minus1))
(rho_minus1/(sqrt(2)*a_minus1));u_minus1 ((u_minus1 +
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1))) ((u_minus1 -
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1)));(0.5*(u_minus1^2))
((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) + (a_minus1^2)/(y-1) +
a_minus1*u_minus1)) ((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) +
(a_minus1^2)/(y-1) - a_minus1*u_minus1))];
        %% R at i+3/2
        R_plus2 = [(1 - 0.5*(y-1)*(u_plus2^2)/(a_plus2^2)) ((y-1)*u_plus2/(a_plus2^2))
(1-y)/(a_plus2^2);((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) - a_plus2*u_plus2))
((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 - u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1));((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) +
a_plus2*u_plus2 )) -((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 + u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1))];
        %% R at i+1/2
        R_plus1 = [(1 - 0.5*(y-1)*(u_plus1^2)/(a_plus1^2)) ((y-1)*u_plus1/(a_plus1^2))
(1-y)/(a_plus1^2);((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) - a_plus1*u_plus1))
((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 - u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1));((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) +
a_plus1*u_plus1 )) -((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 + u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1))];
        %% R at i-1/2
        R_minus1 = [(1 - 0.5*(y-1)*(u_minus1^2)/(a_minus1^2)) ((y-1)*u_minus1/(a_minus1^2))
(1-y)/(a_minus1^2);((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_minus1^2) -
a_minus1*u_minus1)) ((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 - u_minus1*(y-1)))
((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1));((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_min
us1^2) + a_minus1*u_minus1 )) -((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 +
u_minus1*(y-1))) ((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1))];

        %% R at i-3/2
        R_minus2 = [(1 - 0.5*(y-1)*(u_minus2^2)/(a_minus2^2)) ((y-1)*u_minus2/(a_minus2^2))
(1-y)/(a_minus2^2);((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_minus2^2) -
a_minus2*u_minus2)) ((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 - u_minus2*(y-1)))
((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1));((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_min
us2^2) + a_minus2*u_minus2 )) -((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 +
u_minus2*(y-1))) ((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1))];
        %% eigenmatrix
        L_plus = [abs(u_plus1) 0 0;0 abs(u_plus1 + a_plus1) 0;0 0 abs(u_plus1 - a_minus1)];
        L_minus = [abs(u_minus1) 0 0;0 abs(u_minus1 + a_minus1) 0;0 0 abs(u_minus1 -
a_minus1)];

        Q_plus2 = [Q1(i+2,k)-Q1(i+1,k);Q2(i+2,k)-Q2(i+1,k);Q3(i+2,k)-Q3(i+1,k)];
        Q_plus1 = [Q1(i+1,k)-Q1(i,k);Q2(i+1,k)-Q2(i,k);Q3(i+1,k)-Q3(i,k)];
        Q_minus1 = [Q1(i,k)-Q1(i-1,k);Q2(i,k)-Q2(i-1,k);Q3(i,k)-Q3(i-1,k)];
        Q_minus2 = [Q1(i-1,k)-Q1(i-2,k);Q2(i-1,k)-Q2(i-2,k);Q3(i-1,k)-Q3(i-2,k)];

        M_plus2=R_plus2*Q_plus2;%M at i+3/2
        M_plus1=R_plus1*Q_plus1;%M at i+1/2
        M_minus1=R_minus1*Q_minus1;%M at i-1/2
        M_minus2=R_minus2*Q_minus2;% M at i-3/2
        for v=1:3
       % With minmod
%           N_plus(v,1)=M_plus1(v)-minmod(M_minus1(v),M_plus1(v),M_plus2(v));%N i+1/2
%           N_minus(v,1)=M_minus1(v)-minmod(M_minus2(v),M_minus1(v),M_plus1(v));%N at
i-1/2, ignoring the M at the -1 node
%          Without minmod
          N_plus(v,1)=M_plus1(v)-M_minus1(v);%N i+1/2
          N_minus(v,1)=M_minus1(v)-M_minus2(v);%N at i-1/2, ignoring the M at the -1 node

        end
        AQ_plus = Rinv_plus*L_plus*N_plus;
        AQ_minus = Rinv_minus*L_minus*N_minus;
        %% The scheme
        Q1(i,k+1) = Q1(i,k) - dt * 0.5 * ( F1(i+1,k) - F1(i-1,k) - AQ_plus(1) + AQ_minus(1))/dx;
        Q2(i,k+1) = Q2(i,k) - dt * 0.5 * ( F2(i+1,k) - F2(i-1,k) - AQ_plus(2) + AQ_minus(2))/dx;
        Q3(i,k+1) = Q3(i,k) - dt * 0.5 * ( F3(i+1,k) - F3(i-1,k) - AQ_plus(3) + AQ_minus(3))/dx;
        %% Extracting the primitive variables
      rho(i,k+1) =(Q1(i,k+1))./S(i);
      u(i,k+1) = Q2(i,k+1)./(rho(i,k+1).*S(i));
      E(i,k+1) = Q3(i,k+1)./(rho(i,k+1).*S(i));
      p(i,k+1) = (y-1) * ( rho(i,k+1).*E(i,k+1) - 0.5*rho(i,k+1).*(u(i,k+1).^2) );
      H(i,k+1) = E(i,k+1) + p(i,k+1)./rho(i,k+1);
      g2(i,k+1)=p(i,k+1)*dS(i);
      F1(i,k+1) = rho(i,k+1).*u(i,k+1).*S(i);
      F2(i,k+1) = ( rho(i,k+1).*(u(i,k+1).^2) + p(i,k+1) ) .* S(i);
      F3(i,k+1) = rho(i,k+1).*u(i,k+1).*H(i,k+1).*S(i);
         end
      end

 end
 %% Plotting
 plot(x,rho(:,Nt))
 figure
 plot(x,u(:,Nt))
```

SHOCK TUBE SECOND ORDER

```
%% Project Shock tube 2nd order with and without limiter
clc
clear all
%% Defining Variables



L = 10;
N = 100;
y = 1.4;
R = 8.314;
dx = (L/(N-1));
%% Variables at left side
rho_l = 1;
u_l = 0;
p_l = 10^5;
M = 28.97 * 10^-3;
CFL=0.5;
dt = CFL*dx/674.875;% Using Courant number and CFL=0.5
tf = 0.005;% Final time

Nt = fix(tf/dt);
%% Variables at right side
rho_r = 0.125;
u_r = 0;
p_r = 10^4;

E_l = ( p_l/(y-1) + 0.5*rho_l*(u_l^2) )/rho_l;
E_r = ( p_r/(y-1) + 0.5*rho_r*(u_r^2) )/rho_r;

H_l = E_l + p_l/rho_l;
H_r = E_r + p_r/rho_r;
```

```
u = zeros(N,Nt);
rho = zeros(N,Nt);
p = zeros(N,Nt);
E = zeros(N,Nt);
H = zeros(N,Nt);
Q1 = zeros(N,Nt);
Q2 = zeros(N,Nt);
Q3 = zeros(N,Nt);
F1 = zeros(N,Nt);
F2 = zeros(N,Nt);
F3 = zeros(N,Nt);
S=zeros(N,1);
dS=zeros(N,1);
g2=zeros(N,Nt);
%% Boundary conditions for primitive variables
x=0:dx:10;

for i=1:N/2
        rho(i,1) = rho_l;
        u(i,1) = u_l;
        p(i,1) = p_l;
        H(i,1) = E_l + p_l/rho_l;
        E(i,1) = E_l;
        S(i)=1;
        dS(i)=3*(0.2*x(i)-1);
        g2(i,1)=p(i)*dS(i);
end

for i=N/2+1:N
        rho(i,1) = rho_r;
        u(i,1) = u_r;
        p(i,1) = p_r;
        H(i,1) =  E_r + p_r/rho_r;
        E(i,1) = E_r;
        S(i)=1;
        dS(i)=(0.2*x(i)-1);
        g2(i,1)=p(i)*dS(i);
end
%% Initial condition for primitive variables
u(1,:) = u_l;
u(N,:) = u_r;
rho(1,:) = rho_l;
rho(N,:) = rho_r;
p(1,:) = p_l;
p(N,:) = p_r;
E(1,:) = E_l;
E(N,:) = E_r;
H(1,:) = H_l;
H(N,:) = H_r;

%% IC and BC for Q
 Q1(:,1) = rho(:,1) .* S;
 Q2(:,1) = rho(:,1).*u(:,1).*S;
 Q3(:,1) = rho(:,1).*E(:,1).*S;

 Q1(1,:) = rho(1,:) .* S(1);
 Q2(1,:) = rho(1,:).*u(1,:).*S(1);
 Q3(1,:) = rho(1,:).*E(1,:).*S(1);

 Q1(N,:) = rho(N,:) .* S(N);
 Q2(N,:) = rho(N,:).*u(N,:).*S(N);
 Q3(N,:) = rho(N,:).*E(N,:).*S(N);

 %% IC and BC for F
 F1(:,1) = rho(:,1).*u(:,1).*S;
 F2(:,1) = ( rho(:,1).*(u(:,1).^2) + p(:,1) ) .* S;
 F3(:,1) = rho(:,1).*u(:,1).*H(:,1).*S;

 F1(1,:) = rho(1,:).*u(1,:)*S(1);
 F2(1,:) = ( rho(1,:).*(u(1,:).^2) + p(1,:) ) .* S(1);
 F3(1,:) = rho(1,:).*u(1,:).*H(1,:).*S(1);

 F1(N,:) = rho(N,:).*u(N,:).*S(N);
 F2(N,:) = ( rho(N,:).*(u(N,:).^2) + p(N,:) ) .* S(N);
 F3(N,:) = rho(N,:).*u(N,:).*H(N,:).*S(N);




 for k=1:Nt-1
    for i=2:N-1



        if i==2
       %% Roe averaging
       rho_plus2= sqrt(rho(i+2,k)*rho(i+1,k));%rho at i+3/2
       rho_plus1 = sqrt(rho(i,k)*rho(i+1,k));%rho at i+1/2
       rho_minus1 = sqrt(rho(i,k)*rho(i-1,k));%rho at i-1/2
       %rho_minus2 = sqrt(rho(i-2,k)*rho(i-1,k));%rho at i-3/2
       u_plus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i+1,k) )*u(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+1/2
       u_plus2 = ( sqrt(rho(i+2,k))*u(i+2,k) + sqrt(rho(i+1,k))*u(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+3/2
       u_minus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-1/2
       %u_minus2 = ( sqrt(rho(i-2,k))*u(i-2,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-/2
       H_plus2 = ( sqrt(rho(i+2,k))*H(i+2,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%H at i+3/2
```

```
        H_plus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );%H at i+1/2
        H_minus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );%H at i-1/2
        %H_minus2 = ( sqrt(rho(i-2,k))*H(i-2,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%H at i-3/2
        a_plus2 = sqrt( (y-1)*(H_plus2 - 0.5*(u_plus2^2)) );
        a_plus1 = sqrt( (y-1)*(H_plus1 - 0.5*(u_plus1^2)) );
        a_minus1 = sqrt( (y-1)*(H_minus1 - 0.5*(u_minus1^2)) );
        %a_minus2 = sqrt( (y-1)*(H_minus2 - 0.5*(u_minus2^2)) );
        %% R inverse
        Rinv_plus = [1 (rho_plus1/(sqrt(2)*a_plus1)) (rho_plus1/(sqrt(2)*a_plus1));u_plus1
((u_plus1 + a_plus1)*(rho_plus1/(sqrt(2)*a_plus1))) ((u_plus1 -
a_plus1)*(rho_plus1/(sqrt(2)*a_plus1)));(0.5*(u_plus1^2))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) + a_plus1*u_plus1))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) - a_plus1*u_plus1))];
        Rinv_minus = [1 (rho_minus1/(sqrt(2)*a_minus1))
(rho_minus1/(sqrt(2)*a_minus1));u_minus1 ((u_minus1 +
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1))) ((u_minus1 -
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1)));(0.5*(u_minus1^2))
((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) + (a_minus1^2)/(y-1) +
a_minus1*u_minus1)) ((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) +
(a_minus1^2)/(y-1) - a_minus1*u_minus1))];
        %% R at i+3/2
        R_plus2 = [(1 - 0.5*(y-1)*(u_plus2^2)/(a_plus2^2)) ((y-1)*u_plus2/(a_plus2^2))
(1-y)/(a_plus2^2);((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) -
a_plus2*u_plus2)) ((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 - u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1));((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2
^2) + a_plus2*u_plus2 )) -((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 + u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1))];
        %% R at i+1/2
        R_plus1 = [(1 - 0.5*(y-1)*(u_plus1^2)/(a_plus1^2)) ((y-1)*u_plus1/(a_plus1^2))
(1-y)/(a_plus1^2);((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) -
a_plus1*u_plus1)) ((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 - u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1));((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1
^2) + a_plus1*u_plus1 )) -((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 + u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1))];
    %% R at i-1/2
    R_minus1 = [(1 - 0.5*(y-1)*(u_minus1^2)/(a_minus1^2))
((y-1)*u_minus1/(a_minus1^2))
(1-y)/(a_minus1^2);((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_minus1^2) -
a_minus1*u_minus1)) ((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 - u_minus1*(y-1)))
((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1));((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u
_minus1^2) + a_minus1*u_minus1 )) -((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 +
u_minus1*(y-1))) ((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1))];

    %% R at i-3/2
    %R_minus2 = [(1 - 0.5*(y-1)*(u_minus2^2)/(a_minus2^2))
((y-1)*u_minus2/(a_minus2^2))
(1-y)/(a_minus2^2);((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_minus2^2) -
a_minus2*u_minus2)) ((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 - u_minus2*(y-1)))
((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1));((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u
_minus2^2) + a_minus2*u_minus2 )) -((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 +
u_minus2*(y-1))) ((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1))];
    %% eigenmatrix
    L_plus = [abs(u_plus1) 0 0;0 abs(u_plus1 + a_plus1) 0;0 0 abs(u_plus1 - a_minus1)];
    L_minus = [abs(u_minus1) 0 0;0 abs(u_minus1 + a_minus1) 0;0 0 abs(u_minus1 -
a_minus1)];

    Q_plus2 = [Q1(i+2,k)-Q1(i+1,k);Q2(i+2,k)-Q2(i+1,k);Q3(i+2,k)-Q3(i+1,k)];
    Q_plus1 = [Q1(i+1,k)-Q1(i,k);Q2(i+1,k)-Q2(i,k);Q3(i+1,k)-Q3(i,k)];
    Q_minus1 = [Q1(i,k)-Q1(i-1,k);Q2(i,k)-Q2(i-1,k);Q3(i,k)-Q3(i-1,k)];
    %Q_minus2 = [Q1(i-1,k)-Q1(i-2,k);Q2(i-1,k)-Q2(i-2,k);Q3(i-1,k)-Q3(i-2,k)];

    M_plus2=R_plus2*Q_plus2;%M at i+3/2
    M_plus1=R_plus1*Q_plus1;%M at i+1/2
```

```
        M_minus1=R_minus1*Q_minus1;%M at i-1/2
        %M_minus2=R_minus2*Q_minus2;% M at i-3/2
        for v=1:3% With minmod
   %        N_plus(v,1)=M_plus1(v)-minmod(M_minus1(v),M_plus1(v),M_plus2(v));%N i+1/2
    %       N_minus(v,1)=M_minus1(v)-minmod(M_minus1(v),M_plus1(v));%N at i-1/2,
ignoring the M at the -1 node
%           Without minmod
           N_plus(v,1)=M_plus1(v)-M_minus1(v);%N i+1/2
          N_minus(v,1)=M_minus1(v);%N at i-1/2, ignoring the M at the -1 node
        end
        AQ_plus = Rinv_plus*L_plus*N_plus;
        AQ_minus = Rinv_minus*L_minus*N_minus;
        Q1(i,k+1) = Q1(i,k) - dt * 0.5 * ( F1(i+1,k) - F1(i-1,k) - AQ_plus(1) + AQ_minus(1))/dx;
        Q2(i,k+1) = Q2(i,k) - dt * 0.5 * ( F2(i+1,k) - F2(i-1,k) - AQ_plus(2) + AQ_minus(2))/dx;
        Q3(i,k+1) = Q3(i,k) - dt * 0.5 * ( F3(i+1,k) - F3(i-1,k) - AQ_plus(3) + AQ_minus(3))/dx;
        %% Extracting the primitive variables
        rho(i,k+1) =(Q1(i,k+1))./S(i);
        u(i,k+1) = Q2(i,k+1)./(rho(i,k+1).*S(i));
        E(i,k+1) = Q3(i,k+1)./(rho(i,k+1).*S(i));
        p(i,k+1) = (y-1) * ( rho(i,k+1).*E(i,k+1) - 0.5*rho(i,k+1).*(u(i,k+1).^2) );
        H(i,k+1) = E(i,k+1) + p(i,k+1)./rho(i,k+1);
        %g2(i,k+1)=p(3:N-2,k+1)*dS(3:N-2);
        F1(i,k+1) = rho(i,k+1).*u(i,k+1).*S(i);
        F2(i,k+1) = ( rho(i,k+1).*(u(i,k+1).^2) + p(i,k+1) ) .* S(i);
        F3(i,k+1) = rho(i,k+1).*u(i,k+1).*H(i,k+1).*S(i);



         elseif i==N-1
        %% Roe averaging
        %rho_plus2= sqrt(rho(i+2,k)*rho(i+1,k));%rho at i+3/2
        rho_plus1 = sqrt(rho(i,k)*rho(i+1,k));%rho at i+1/2
        rho_minus1 = sqrt(rho(i,k)*rho(i-1,k));%rho at i-1/2
        rho_minus2 = sqrt(rho(i-2,k)*rho(i-1,k));%rho at i-3/2
        u_plus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i+1,k) )*u(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+1/2
        %u_plus2 = ( sqrt(rho(i+2,k))*u(i+2,k) + sqrt(rho(i+1,k))*u(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+3/2
        u_minus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-1/2
        u_minus2 = ( sqrt(rho(i-2,k))*u(i-2,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-/2
        %H_plus2 = ( sqrt(rho(i+2,k))*H(i+2,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%H at i+3/2
        H_plus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );%H at i+1/2
        H_minus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );%H at i-1/2
        H_minus2 = ( sqrt(rho(i-2,k))*H(i-2,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%H at i-3/2
        %a_plus2 = sqrt( (y-1)*(H_plus2 - 0.5*(u_plus2^2)) );
        a_plus1 = sqrt( (y-1)*(H_plus1 - 0.5*(u_plus1^2)) );
        a_minus1 = sqrt( (y-1)*(H_minus1 - 0.5*(u_minus1^2)) );
        a_minus2 = sqrt( (y-1)*(H_minus2 - 0.5*(u_minus2^2)) );
        %% R inverse
        Rinv_plus = [1 (rho_plus1/(sqrt(2)*a_plus1)) (rho_plus1/(sqrt(2)*a_plus1));u_plus1
((u_plus1 + a_plus1)*(rho_plus1/(sqrt(2)*a_plus1))) ((u_plus1 -
a_plus1)*(rho_plus1/(sqrt(2)*a_plus1)));(0.5*(u_plus1^2))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) + a_plus1*u_plus1))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) - a_plus1*u_plus1))];
        Rinv_minus = [1 (rho_minus1/(sqrt(2)*a_minus1))
(rho_minus1/(sqrt(2)*a_minus1));u_minus1 ((u_minus1 +
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1))) ((u_minus1 -
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1)));(0.5*(u_minus1^2))
((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) + (a_minus1^2)/(y-1) +
a_minus1*u_minus1)) ((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) +
(a_minus1^2)/(y-1) - a_minus1*u_minus1))];
    %% R at i+3/2
    %R_plus2 = [(1 - 0.5*(y-1)*(u_plus2^2)/(a_plus2^2)) ((y-1)*u_plus2/(a_plus2^2))
(1-y)/(a_plus2^2);((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) -
a_plus2*u_plus2)) ((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 - u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1));((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2
^2) + a_plus2*u_plus2 )) -((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 + u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1))];
    %% R at i+1/2
    R_plus1 = [(1 - 0.5*(y-1)*(u_plus1^2)/(a_plus1^2)) ((y-1)*u_plus1/(a_plus1^2))
(1-y)/(a_plus1^2);((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) -
a_plus1*u_plus1)) ((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 - u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1));((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1
^2) + a_plus1*u_plus1 )) -((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 + u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1))];
    %% R at i-1/2
    R_minus1 = [(1 - 0.5*(y-1)*(u_minus1^2)/(a_minus1^2))
((y-1)*u_minus1/(a_minus1^2))
(1-y)/(a_minus1^2);((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_minus1^2) -
a_minus1*u_minus1)) ((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 - u_minus1*(y-1)))
((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1));((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u
_minus1^2) + a_minus1*u_minus1 )) -((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 +
u_minus1*(y-1))) ((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1))];

    %% R at i-3/2
    R_minus2 = [(1 - 0.5*(y-1)*(u_minus2^2)/(a_minus2^2))
((y-1)*u_minus2/(a_minus2^2))
(1-y)/(a_minus2^2);((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_minus2^2) -
a_minus2*u_minus2)) ((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 - u_minus2*(y-1)))
((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1));((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u
_minus2^2) + a_minus2*u_minus2 )) -((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 +
u_minus2*(y-1))) ((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1))];
```

```
    %% eigenmatrix
    L_plus = [abs(u_plus1) 0 0;0 abs(u_plus1 + a_plus1) 0;0 0 abs(u_plus1 - a_minus1)];
    L_minus = [abs(u_minus1) 0 0;0 abs(u_minus1 + a_minus1) 0;0 0 abs(u_minus1 -
a_minus1)];

    %Q_plus2 = [Q1(i+2,k)-Q1(i+1,k);Q2(i+2,k)-Q2(i+1,k);Q3(i+2,k)-Q3(i+1,k)];
    Q_plus1 = [Q1(i+1,k)-Q1(i,k);Q2(i+1,k)-Q2(i,k);Q3(i+1,k)-Q3(i,k)];
    Q_minus1 = [Q1(i,k)-Q1(i-1,k);Q2(i,k)-Q2(i-1,k);Q3(i,k)-Q3(i-1,k)];
    Q_minus2 = [Q1(i-1,k)-Q1(i-2,k);Q2(i-1,k)-Q2(i-2,k);Q3(i-1,k)-Q3(i-2,k)];

    %M_plus2=R_plus2*Q_plus2;%M at i+3/2
    M_plus1=R_plus1*Q_plus1;%M at i+1/2
    M_minus1=R_minus1*Q_minus1;%M at i-1/2
    M_minus2=R_minus2*Q_minus2;% M at i-3/2
    for v=1:3

      % With minmod
 %        N_plus(v,1)=M_plus1(v)-minmod(M_minus1(v),M_plus1(v));%N i+1/2
 %        N_minus(v,1)=M_minus1(v)-minmod(M_minus2(v),M_minus1(v),M_plus1(v));%N
at i-1/2, ignoring the M at the -1 node
%        Without minmod
      N_plus(v,1)=M_plus1(v)-M_minus1(v);%N i+1/2
      N_minus(v,1)=M_minus1(v)-M_minus2(v);%N at i-1/2, ignoring the M at the -1 node
    end
    AQ_plus = Rinv_plus*L_plus*N_plus;
    AQ_minus = Rinv_minus*L_minus*N_minus;
    Q1(i,k+1) = Q1(i,k) - dt * 0.5 * ( F1(i+1,k) - F1(i-1,k) - AQ_plus(1) + AQ_minus(1))/dx;
    Q2(i,k+1) = Q2(i,k) - dt * 0.5 * ( F2(i+1,k) - F2(i-1,k) - AQ_plus(2) + AQ_minus(2))/dx;
    Q3(i,k+1) = Q3(i,k) - dt * 0.5 * ( F3(i+1,k) - F3(i-1,k) - AQ_plus(3) + AQ_minus(3))/dx;
    %% Extracting the primitive variables
    rho(i,k+1) =(Q1(i,k+1))./S(i);
    u(i,k+1) = Q2(i,k+1)./(rho(i,k+1).*S(i));
    E(i,k+1) = Q3(i,k+1)./(rho(i,k+1).*S(i));
```

```
        p(i,k+1) = (y-1) * ( rho(i,k+1).*E(i,k+1) - 0.5*rho(i,k+1).*(u(i,k+1).^2) );
        H(i,k+1) = E(i,k+1) + p(i,k+1)./rho(i,k+1);
        %g2(i,k+1)=p(i,k+1)*dS(i);
        F1(i,k+1) = rho(i,k+1).*u(i,k+1).*S(i);
        F2(i,k+1) = ( rho(i,k+1).*(u(i,k+1).^2) + p(i,k+1) ) .* S(i);
        F3(i,k+1) = rho(i,k+1).*u(i,k+1).*H(i,k+1).*S(i);

         else
            %% Roe averaging
        rho_plus2= sqrt(rho(i+2,k)*rho(i+1,k));%rho at i+3/2
        rho_plus1 = sqrt(rho(i,k)*rho(i+1,k));%rho at i+1/2
        rho_minus1 = sqrt(rho(i,k)*rho(i-1,k));%rho at i-1/2
        rho_minus2 = sqrt(rho(i-2,k)*rho(i-1,k));%rho at i-3/2
        u_plus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i+1,k) )*u(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+1/2
        u_plus2 = ( sqrt(rho(i+2,k))*u(i+2,k) + sqrt(rho(i+1,k))*u(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%Velocity at i+3/2
        u_minus1 = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-1/2
        u_minus2 = ( sqrt(rho(i-2,k))*u(i-2,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%Velocity at i-/2
        H_plus2 = ( sqrt(rho(i+2,k))*H(i+2,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i+2,k)) +
sqrt(rho(i+1,k)) );%H at i+3/2
        H_plus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );%H at i+1/2
        H_minus1 = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );%H at i-1/2
        H_minus2 = ( sqrt(rho(i-2,k))*H(i-2,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i-2,k)) +
sqrt(rho(i-1,k)) );%H at i-3/2
        a_plus2 = sqrt( (y-1)*(H_plus2 - 0.5*(u_plus2^2)) );
        a_plus1 = sqrt( (y-1)*(H_plus1 - 0.5*(u_plus1^2)) );
        a_minus1 = sqrt( (y-1)*(H_minus1 - 0.5*(u_minus1^2)) );
        a_minus2 = sqrt( (y-1)*(H_minus2 - 0.5*(u_minus2^2)) );
```

```
    %% R inverse
    Rinv_plus = [1 (rho_plus1/(sqrt(2)*a_plus1)) (rho_plus1/(sqrt(2)*a_plus1));u_plus1
((u_plus1 + a_plus1)*(rho_plus1/(sqrt(2)*a_plus1))) ((u_plus1 -
a_plus1)*(rho_plus1/(sqrt(2)*a_plus1)));(0.5*(u_plus1^2))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) + a_plus1*u_plus1))
((rho_plus1/(sqrt(2)*a_plus1))*(0.5*(u_plus1^2) + (a_plus1^2)/(y-1) - a_plus1*u_plus1))];
    Rinv_minus = [1 (rho_minus1/(sqrt(2)*a_minus1))
(rho_minus1/(sqrt(2)*a_minus1));u_minus1 ((u_minus1 +
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1))) ((u_minus1 -
a_minus1)*(rho_minus1/(sqrt(2)*a_minus1)));(0.5*(u_minus1^2))
((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) + (a_minus1^2)/(y-1) +
a_minus1*u_minus1)) ((rho_minus1/(sqrt(2)*a_minus1))*(0.5*(u_minus1^2) +
(a_minus1^2)/(y-1) - a_minus1*u_minus1))];
    %% R at i+3/2
    R_plus2 = [(1 - 0.5*(y-1)*(u_plus2^2)/(a_plus2^2)) ((y-1)*u_plus2/(a_plus2^2))
(1-y)/(a_plus2^2);((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2^2) -
a_plus2*u_plus2)) ((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 - u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1));((1/(rho_plus2*a_plus2*sqrt(2)))*(0.5*(y-1)*(u_plus2
^2) + a_plus2*u_plus2 )) -((1/(rho_plus2*a_plus2*sqrt(2)))*(a_plus2 + u_plus2*(y-1)))
((1/(rho_plus2*a_plus2*sqrt(2)))*(y-1))];
    %% R at i+1/2
    R_plus1 = [(1 - 0.5*(y-1)*(u_plus1^2)/(a_plus1^2)) ((y-1)*u_plus1/(a_plus1^2))
(1-y)/(a_plus1^2);((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1^2) -
a_plus1*u_plus1)) ((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 - u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1));((1/(rho_plus1*a_plus1*sqrt(2)))*(0.5*(y-1)*(u_plus1
^2) + a_plus1*u_plus1 )) -((1/(rho_plus1*a_plus1*sqrt(2)))*(a_plus1 + u_plus1*(y-1)))
((1/(rho_plus1*a_plus1*sqrt(2)))*(y-1))];
    %% R at i-1/2
    R_minus1 = [(1 - 0.5*(y-1)*(u_minus1^2)/(a_minus1^2))
((y-1)*u_minus1/(a_minus1^2))
(1-y)/(a_minus1^2);((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u_minus1^2) -
a_minus1*u_minus1)) ((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 - u_minus1*(y-1)))
((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1));((1/(rho_minus1*a_minus1*sqrt(2)))*(0.5*(y-1)*(u
_minus1^2) + a_minus1*u_minus1 )) -((1/(rho_minus1*a_minus1*sqrt(2)))*(a_minus1 +
u_minus1*(y-1))) ((1/(rho_minus1*a_minus1*sqrt(2)))*(y-1))];

        %% R at i-3/2
        R_minus2 = [(1 - 0.5*(y-1)*(u_minus2^2)/(a_minus2^2))
((y-1)*u_minus2/(a_minus2^2))
(1-y)/(a_minus2^2);((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u_minus2^2) -
a_minus2*u_minus2)) ((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 - u_minus2*(y-1)))
((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1));((1/(rho_minus2*a_minus2*sqrt(2)))*(0.5*(y-1)*(u
_minus2^2) + a_minus2*u_minus2 )) -((1/(rho_minus2*a_minus2*sqrt(2)))*(a_minus2 +
u_minus2*(y-1))) ((1/(rho_minus2*a_minus2*sqrt(2)))*(y-1))];
        %% eigenmatrix
        L_plus = [abs(u_plus1) 0 0;0 abs(u_plus1 + a_plus1) 0;0 0 abs(u_plus1 - a_minus1)];
        L_minus = [abs(u_minus1) 0 0;0 abs(u_minus1 + a_minus1) 0;0 0 abs(u_minus1 -
a_minus1)];

        Q_plus2 = [Q1(i+2,k)-Q1(i+1,k);Q2(i+2,k)-Q2(i+1,k);Q3(i+2,k)-Q3(i+1,k)];
        Q_plus1 = [Q1(i+1,k)-Q1(i,k);Q2(i+1,k)-Q2(i,k);Q3(i+1,k)-Q3(i,k)];
        Q_minus1 = [Q1(i,k)-Q1(i-1,k);Q2(i,k)-Q2(i-1,k);Q3(i,k)-Q3(i-1,k)];
        Q_minus2 = [Q1(i-1,k)-Q1(i-2,k);Q2(i-1,k)-Q2(i-2,k);Q3(i-1,k)-Q3(i-2,k)];

        M_plus2=R_plus2*Q_plus2;%M at i+3/2
        M_plus1=R_plus1*Q_plus1;%M at i+1/2
        M_minus1=R_minus1*Q_minus1;%M at i-1/2
        M_minus2=R_minus2*Q_minus2;% M at i-3/2
        for v=1:3
% With minmod
        %     N_plus(v,1)=M_plus1(v)-minmod(M_minus1(v),M_plus1(v),M_plus2(v));%N i+1/2
         %   N_minus(v,1)=M_minus1(v)-minmod(M_minus2(v),M_minus1(v),M_plus1(v));%N
at i-1/2, ignoring the M at the -1 node
%           Without minmod
            N_plus(v,1)=M_plus1(v)-M_minus1(v);%N i+1/2
            N_minus(v,1)=M_minus1(v)-M_minus2(v);%N at i-1/2, ignoring the M at the -1 node
        end
        AQ_plus = Rinv_plus*L_plus*N_plus;
        AQ_minus = Rinv_minus*L_minus*N_minus;
        %% The scheme
        Q1(i,k+1) = Q1(i,k) - dt * 0.5 * ( F1(i+1,k) - F1(i-1,k) - AQ_plus(1) + AQ_minus(1))/dx;
        Q2(i,k+1) = Q2(i,k) - dt * 0.5 * ( F2(i+1,k) - F2(i-1,k) - AQ_plus(2) + AQ_minus(2))/dx;
        Q3(i,k+1) = Q3(i,k) - dt * 0.5 * ( F3(i+1,k) - F3(i-1,k) - AQ_plus(3) + AQ_minus(3))/dx;
        %% Extracting the primitive variables
      rho(i,k+1) =(Q1(i,k+1))./S(i);
      u(i,k+1) = Q2(i,k+1)./(rho(i,k+1).*S(i));
      E(i,k+1) = Q3(i,k+1)./(rho(i,k+1).*S(i));
      p(i,k+1) = (y-1) * ( rho(i,k+1).*E(i,k+1) - 0.5*rho(i,k+1).*(u(i,k+1).^2) );
      H(i,k+1) = E(i,k+1) + p(i,k+1)./rho(i,k+1);
      %g2(i,k+1)=p(i,k+1)*dS(i);
      F1(i,k+1) = rho(i,k+1).*u(i,k+1).*S(i);
      F2(i,k+1) = ( rho(i,k+1).*(u(i,k+1).^2) + p(i,k+1) ) .* S(i);
      F3(i,k+1) = rho(i,k+1).*u(i,k+1).*H(i,k+1).*S(i);
         end
      end
      %% Extracting the primitive variables
%     rho(3:N-2,k+1) =(Q1(3:N-2,k+1))./S(3:N-2);
%     u(3:N-2,k+1) = Q2(3:N-2,k+1)./(rho(3:N-2,k+1).*S(3:N-2));
%     E(3:N-2,k+1) = Q3(2:N-2,k+1)./(rho(2:N-2,k+1).*S(3:N-2));
%     p(3:N-2,k+1) = (y-1) * ( rho(3:N-2,k+1).*E(3:N-2,k+1) -
0.5*rho(3:N-2,k+1).*(u(3:N-2,k+1).^2) );
%     H(3:N-2,k+1) = E(3:N-2,k+1) + p(3:N-2,k+1)./rho(3:N-2,k+1);
%     g2(3:N-2,k+1)=p(3:N-2,k+1)*dS(3:N-2);
%     F1(3:N-2,k+1) = rho(3:N-2,k+1).*u(3:N-2,k+1).*S(3:N-2);
%     F2(3:N-2,k+1) = ( rho(3:N-2,k+1).*(u(3:N-2,k+1).^2) + p(3:N-2,k+1) ) .* S(3:N-2);
%     F3(3:N-2,k+1) = rho(3:N-2,k+1).*u(3:N-2,k+1).*H(3:N-2,k+1).*S(3:N-2);
 end
 %% Plotting
%plot(x,rho(:,Nt))
```

```
plot(x,rho(:,Nt))
title('Density vs displacement')
figure
plot(x,u(:,Nt))
 title('Velocity vs displacement')
figure
plot(x,p(:,Nt))
 title('Pressure vs displacement')
figure
plot(x,E(:,Nt))
 title('Energy vs displacement')
```

SHOCK TUBE
FIRST ORDER

```
clc;
clear all;
%% Defining Variables
S = 1;
L = 10;
N = 100;
y = 1.4;
R = 8.314;
dx = (L/(N-1));
%% Variables at left side
rho_l = 1;
u_l = 0;
p_l = 10^5;
M = 28.97 * 10^-3;

T_L = p_l/(rho_l*R);% Temprature required for speed of sound


%% Variables at right side
rho_r = 0.125;
u_r = 0;
p_r = 10^4;

E_l = ( p_l/(y-1) + 0.5*rho_l*(u_l^2) )/rho_l;
E_r = ( p_r/(y-1) + 0.5*rho_r*(u_r^2) )/rho_r;

H_l = E_l + p_l/rho_l;
H_r = E_r + p_r/rho_r;
a_L = sqrt((y-1)*H_l);%Speed of sound
dt =0.5*dx/669.4888;% Using Courant number
```

```
tf = 0.005;% Final time
Nt = fix(tf/dt);
u = zeros(N,Nt);
rho = zeros(N,Nt);
p = zeros(N,Nt);
E = zeros(N,Nt);
H = zeros(N,Nt);
Q1 = zeros(N,Nt);
Q2 = zeros(N,Nt);
Q3 = zeros(N,Nt);
F1 = zeros(N,Nt);
F2 = zeros(N,Nt);
F3 = zeros(N,Nt);
%% Boundary conditions for primitive variables
x=0:dx:10;
for i=1:N/2
        rho(i,1) = rho_l;
        u(i,1) = u_l;
        p(i,1) = p_l;
        H(i,1) = E_l + p_l/rho_l;
        E(i,1) = E_l;
end

for i=N/2+1:N
        rho(i,1) = rho_r;
        u(i,1) = u_r;
        p(i,1) = p_r;
        H(i,1) =  E_r + p_r/rho_r;
        E(i,1) = E_r;

end
%% Initial condition for primitive variables
u(1,:) = u_l;
```

```
u(N,:) = u_r;
rho(1,:) = rho_l;
rho(N,:) = rho_r;
p(1,:) = p_l;
p(N,:) = p_r;
E(1,:) = E_l;
E(N,:) = E_r;
H(1,:) = H_l;
H(N,:) = H_r;

%% IC and BC for Q
 Q1(:,1) = rho(:,1) .* S;
 Q2(:,1) = rho(:,1).*u(:,1).*S;
 Q3(:,1) = rho(:,1).*E(:,1).*S;

 Q1(1,:) = rho(1,:) .* S;
 Q2(1,:) = rho(1,:).*u(1,:).*S;
 Q3(1,:) = rho(1,:).*E(1,:).*S;

 Q1(N,:) = rho(N,:) .* S;
 Q2(N,:) = rho(N,:).*u(N,:).*S;
 Q3(N,:) = rho(N,:).*E(N,:).*S;

 %% IC and BC for F
 F1(:,1) = rho(:,1).*u(:,1)*S;
 F2(:,1) = ( rho(:,1).*(u(:,1).^2) + p(:,1) ) .* S;
 F3(:,1) = rho(:,1).*u(:,1).*H(:,1).*S;

 F1(1,:) = rho(1,:).*u(1,:)*S;
 F2(1,:) = ( rho(1,:).*(u(1,:).^2) + p(1,:) ) .* S;
 F3(1,:) = rho(1,:).*u(1,:).*H(1,:).*S;

 F1(N,:) = rho(N,:).*u(N,:).*S;
 F2(N,:) = ( rho(N,:).*(u(N,:).^2) + p(N,:) ) .* S;
 F3(N,:) = rho(N,:).*u(N,:).*H(N,:).*S;




 for k=1:Nt-1
    for i=2:N-1
        %% Roe averaging
        rho_plus = sqrt(rho(i,k)*rho(i+1,k));
        rho_minus = sqrt(rho(i,k)*rho(i-1,k));
        u_plus = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i+1,k))*u(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );
        u_p2 = u_plus*u_plus;
        u_p3 = u_plus*u_plus*u_plus;

        u_minus = ( sqrt(rho(i,k))*u(i,k) + sqrt(rho(i-1,k))*u(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );
        u_m_sqr = u_minus*u_minus;
        u_m_cube = u_minus*u_minus*u_minus;

        H_plus = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i+1,k))*H(i+1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i+1,k)) );
        H_minus = ( sqrt(rho(i,k))*H(i,k) + sqrt(rho(i-1,k))*H(i-1,k) )/( sqrt(rho(i,k)) +
sqrt(rho(i-1,k)) );

        a_plus = sqrt( (y-1)*(H_plus - 0.5*(u_plus^2)) );
        a_minus = sqrt( (y-1)*(H_minus - 0.5*(u_minus^2)) );
        %% R inverse
        Rinv_plus = [1 (rho_plus/(sqrt(2)*a_plus)) (rho_plus/(sqrt(2)*a_plus));u_plus ((u_plus
+ a_plus)*(rho_plus/(sqrt(2)*a_plus))) ((u_plus -
a_plus)*(rho_plus/(sqrt(2)*a_plus)));(0.5*(u_plus^2))
((rho_plus/(sqrt(2)*a_plus))*(0.5*(u_plus^2) + (a_plus^2)/(y-1) + a_plus*u_plus))
((rho_plus/(sqrt(2)*a_plus))*(0.5*(u_plus^2) + (a_plus^2)/(y-1) - a_plus*u_plus))];
```

```
        Rinv_minus = [1 (rho_minus/(sqrt(2)*a_minus))
(rho_minus/(sqrt(2)*a_minus));u_minus ((u_minus +
a_minus)*(rho_minus/(sqrt(2)*a_minus))) ((u_minus -
a_minus)*(rho_minus/(sqrt(2)*a_minus)));(0.5*(u_minus^2))
((rho_minus/(sqrt(2)*a_minus))*(0.5*(u_minus^2) + (a_minus^2)/(y-1) +
a_minus*u_minus)) ((rho_minus/(sqrt(2)*a_minus))*(0.5*(u_minus^2) + (a_minus^2)/(y-1) -
a_minus*u_minus))];
        %% R
        R_plus = [(1 - 0.5*(y-1)*(u_plus^2)/(a_plus^2)) ((y-1)*u_plus/(a_plus^2))
(1-y)/(a_plus^2);((1/(rho_plus*a_plus*sqrt(2)))*(0.5*(y-1)*(u_plus^2) - a_plus*u_plus))
((1/(rho_plus*a_plus*sqrt(2)))*(a_plus - u_plus*(y-1)))
((1/(rho_plus*a_plus*sqrt(2)))*(y-1));((1/(rho_plus*a_plus*sqrt(2)))*(0.5*(y-1)*(u_plus^2) +
a_plus*u_plus )) -((1/(rho_plus*a_plus*sqrt(2)))*(a_plus + u_plus*(y-1)))
((1/(rho_plus*a_plus*sqrt(2)))*(y-1))];
        R_minus = [(1 - 0.5*(y-1)*(u_minus^2)/(a_minus^2)) ((y-1)*u_minus/(a_minus^2))
(1-y)/(a_minus^2);((1/(rho_minus*a_minus*sqrt(2)))*(0.5*(y-1)*(u_minus^2) -
a_minus*u_minus)) ((1/(rho_minus*a_minus*sqrt(2)))*(a_minus - u_minus*(y-1)))
((1/(rho_minus*a_minus*sqrt(2)))*(y-1));((1/(rho_minus*a_minus*sqrt(2)))*(0.5*(y-1)*(u_min
us^2) + a_minus*u_minus )) -((1/(rho_minus*a_minus*sqrt(2)))*(a_minus +
u_minus*(y-1))) ((1/(rho_minus*a_minus*sqrt(2)))*(y-1))];
        %% eigenmatrix
        L_plus = [abs(u_plus) 0 0;0 abs(u_plus + a_plus) 0;0 0 abs(u_plus - a_minus)];
        L_minus = [abs(u_minus) 0 0;0 abs(u_minus + a_minus) 0;0 0 abs(u_minus -
a_minus)];

        A_plus = Rinv_plus * L_plus * R_plus;
        A_minus = Rinv_minus * L_minus * R_minus;

        Q_plus = [Q1(i+1,k)-Q1(i,k);Q2(i+1,k)-Q2(i,k);Q3(i+1,k)-Q3(i,k)];
        Q_minus = [Q1(i,k)-Q1(i-1,k);Q2(i,k)-Q2(i-1,k);Q3(i,k)-Q3(i-1,k)];

        AQ_plus = A_plus * Q_plus;
        AQ_minus = A_minus * Q_minus;
        %% The scheme
        Q1(i,k+1) = Q1(i,k) - dt * 0.5 * ( F1(i+1,k) - F1(i-1,k) - AQ_plus(1) + AQ_minus(1))/dx;
        Q2(i,k+1) = Q2(i,k) - dt * 0.5 * ( F2(i+1,k) - F2(i-1,k) - AQ_plus(2) + AQ_minus(2))/dx;
        Q3(i,k+1) = Q3(i,k) - dt * 0.5 * ( F3(i+1,k) - F3(i-1,k) - AQ_plus(3) + AQ_minus(3))/dx;

    end
    %% Extracting the primitive variables
    rho(2:N-1,k+1) =(Q1(2:N-1,k+1))./S;
    u(2:N-1,k+1) = Q2(2:N-1,k+1)./(rho(2:N-1,k+1).*S);
    E(2:N-1,k+1) = Q3(2:N-1,k+1)./(rho(2:N-1,k+1).*S);
    p(2:N-1,k+1) = (y-1) * ( rho(2:N-1,k+1).*E(2:N-1,k+1) -
0.5*rho(2:N-1,k+1).*(u(2:N-1,k+1).^2) );
    H(2:N-1,k+1) = E(2:N-1,k+1) + p(2:N-1,k+1)./rho(2:N-1,k+1);

    F1(2:N-1,k+1) = rho(2:N-1,k+1).*u(2:N-1,k+1).*S;
    F2(2:N-1,k+1) = ( rho(2:N-1,k+1).*(u(2:N-1,k+1).^2) + p(2:N-1,k+1) ) .* S;
    F3(2:N-1,k+1) = rho(2:N-1,k+1).*u(2:N-1,k+1).*H(2:N-1,k+1).*S;

 end
%% Plotting
 plot(x,rho(:,Nt))
 title('Density vs. X at time = 0.005 sec, N=100')
 xlabel('X')
 ylabel('Density')
figure
 plot(x,u(:,Nt))
title('Velocity vs. X at time = 0.005 sec, N=100')
xlabel('X')
ylabel('Velocity')
figure
plot(x,E(:,Nt))
title('Energy vs. X at time = 0.005 sec, N=100')
xlabel('X')
```

```
ylabel('Energy')
figure
plot(x,p(:,Nt))
title('Pressure vs. X at time = 0.005 sec, N=100')
xlabel('X')
ylabel('Pressure')
```

MINMOD

```
function AQ=minmod(M1,M2,M3)


   if ((M1/abs(M1))==1) && ((M2/abs(M2))==1) && ((M3/abs(M3))==1)
      M=[M1 M2 M3];
      AQ=min(M);
   elseif ((M1/abs(M1))==-1) && ((M2/abs(M2))==-1) && ((M3/abs(M3))==-1)
      M=[abs(M1) abs(M2) abs(M3)];
      AQ=min(M);
   else
      AQ=0;
   end

end
```