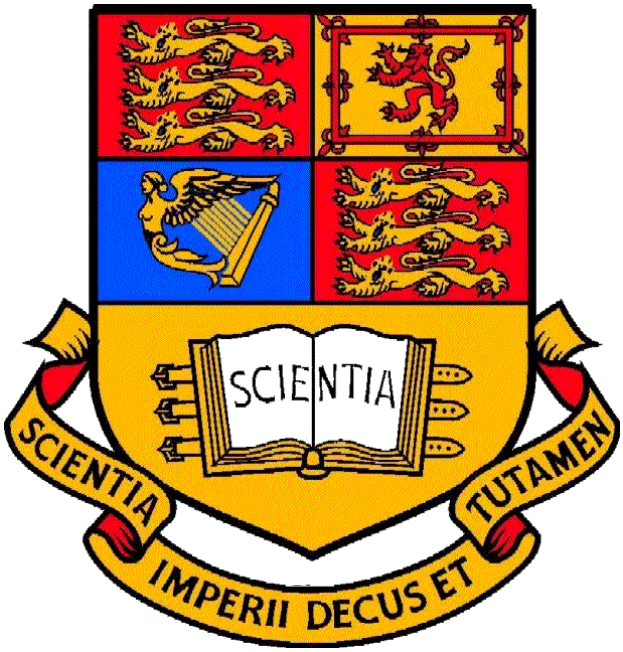
Computational Fluid Dynamics

Coursework 2



Numerical Solutions of 1D Euler Equations Flux Vector Splitting Method

Student: Mr Xerxes Chong Xian

CID: 01389744

Submission Date: 27th January 2020

Question 1

The analytic solution to the Riemann Shock Tube problem was obtained by running the MATLAB code "Shock_tube_analytic.m" to a time $t = 0.5$. A pressure ratio of 10 and a density ratio of 8 were the inputs to the code. An arbitrary value for $p_1 = 1$ and $\rho_1 = 1$ was selected to initialise the solver, representing the parameters of the driven air on the right of the diaphragm. Thus, the values of p_2 and ρ_2 were found. In theory, the initial values of p_1, p_2, ρ_1 and ρ_2 can be set and measured and are not limited to the values used here. The velocity on both sides of the diaphragm, u_1 and u_2 are initially at zero as the air is still.

Question 2

The 1D Euler equations from the handout are first written in the conservative form shown below.

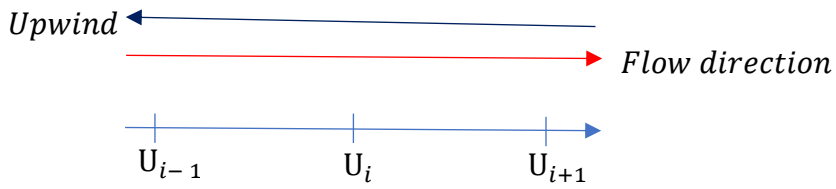
$$\frac{\partial \mathrm{U}}{\partial \mathrm{t}} + \frac{\partial F(\mathrm{U})}{\partial \mathrm{x}} = 0$$

The flux vector is F. A Flux Vector Splitting Method is employed that splits the flux vector F into 2 components F^+ and F^- such that $\mathrm{F} = F^+ + F^-$. The conservative form on the 1D Euler equations are now of the form.

$$\frac{\partial \mathrm{U}}{\partial t} + \frac{\partial F^+(\mathrm{U})}{\partial x} + \frac{\partial F^-(\mathrm{U})}{\partial x} = 0$$

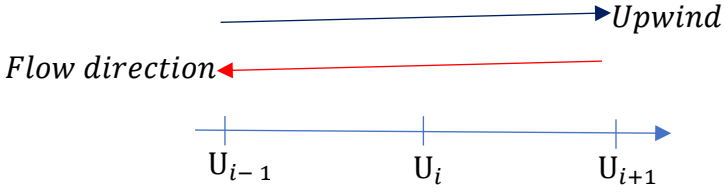
The flux derivatives were then discretised using the first order upwind schemes as shown. For $\frac{\partial F^+(\mathrm{U})}{\partial \mathrm{x}}$, which corresponds to positive eigenvalues, the flow direction is indicated as follows and hence a first order upwind scheme is expressed in the form. This is first order accurate in space.

$$\frac{\partial F^+(\mathrm{U})}{\partial x} = \frac{F^+(\mathrm{U}_i) - F^+(\mathrm{U}_{i-1})}{\Delta x} + \mathcal{O}(\Delta x)$$



For $\frac{\partial F^-(\mathrm{U})}{\partial x}$, which corresponds to negative eigenvalues, the flow direction is indicated as follows and hence a first order upwind scheme is of the form below. This is also first order accurate in space. It is coincidentally a backward wards discretisation for when the eigenvalues are positive.

$$\frac{\partial F^-(\mathrm{U})}{\partial x} = \frac{F^-(\mathrm{U}_{i+1}) - F^-(\mathrm{U}_i)}{\Delta x} + \mathcal{O}(\Delta x)$$



The fluxes F+ and F- are evaluated using the form provided in the handout below. Where $\lambda_i^{\pm}$ correspond to the positive and negative eigenvalues.

$$F^{\pm}(\mathrm{U}) = \frac{\rho}{2\gamma}\begin{bmatrix} {\lambda_1}^{\pm} + 2(\gamma-1){\lambda_2}^{\pm} + {\lambda_3}^{\pm} \\ (u-c){\lambda_1}^{\pm} + 2(\gamma-1)u{\lambda_2}^{\pm} + (u+c){\lambda_3}^{\pm} \\ (H-uc){\lambda_1}^{\pm} + (\gamma-1)u^2{\lambda_2}^{\pm} + (H+uc){\lambda_3}^{\pm} \end{bmatrix}$$

The actual eigenvalues are defined as $\lambda_1 = u - c$, $\lambda_2 = u$ and $\lambda_3 = u + c$ and from the equations below, the positive and negative eigenvalues can be found.

$$\lambda_i{}^{+} = \frac{1}{2}(\lambda_i + |\lambda_i|) \qquad\qquad \lambda_i{}^{-} = \frac{1}{2}(\lambda_i - |\lambda_i|)$$

Question 3

Having discretised the flux derivatives, a first order forwards in time scheme was applied to the time derivative, making it a first order scheme.

$$\frac{\partial \mathrm{U}}{\partial t} = \frac{{U_i}^{n+1} - {U_i}^{n}}{\Delta t} + \mathcal{O}(\Delta t)$$

Having obtains discretised forms of all the terms, the conservative form was re-arranged into the equation below. This form allows an iterative method to be applied to obtain the values of U at the next timestep, ${U_i}^{n+1}$ using only the values at ${U_i}^{n}$. This equation is explicit in time and is first order accurate in time and space.

$${U_i}^{n+1} = {U_i}^{n} + \frac{\Delta t}{\Delta x}[F^{+}(\mathrm{U}_i) - F^{+}(\mathrm{U}_{i-1}) + F^{-}(\mathrm{U}_{i+1}) - F^{-}(\mathrm{U}_i)\,] + \mathcal{O}(\Delta x, \Delta t)$$

This first order upwind scheme is conservative, consistent and monotone (1), hence oscillatory behaviour is not expected and if it converges it will converge to a weak solution according to the Lax-Wendroff Theorem (2). A weak solution indicates that the solution exists numerically but may not be physically possible.

Equation above was implemented using the MATLAB code provided in the appendix for 100, 200 and 300 equally space grid points over the range of $-2 < x < 2$ as required. As the solution was evaluated only up till a time of 0.5, the boundary conditions at both ends of the problem remains the same and are easily accounted for in the code. Where numerical boundary conditions to be $\mathrm{U}_0 = \mathrm{U}_1$ and $\mathrm{U}_{N+1} = \mathrm{U}_N$ at the boundaries $x = -2$ and $x = 2$ respectively.

The list below describes briefly the steps involved in implementing the equation:

1. Initialisation of boundary conditions across the domain $-2 < x < 2$
2. Evaluating of timestep based on current flow properties
3. Evaluation of eigenvalues (positive and negative) and flux derivatives
4. Use of equation to obtain values of U at the next timestep n+1, ${U_i}^{n+1}$ and increase the time counter by that iteration's value of Δt
5. Flow properties are updated from the new values of U
6. Steps 2 to 5 are repeated up till the time= 0.5 (use of a while loop)
7. Flow properties are evaluated from the most recent value of U

Question 4

According linear stability theory (1), the condition for stability for an explicit first-order upwind scheme is:

$$|{\lambda_i}^{\pm}|\frac{\Delta t}{\Delta x} \leq 1$$

Where 1 is the Courant Number. Hence Δt is determined by rearranging to obtain:

$$\Delta t \leq \frac{\Delta x \times 1}{|\lambda_i{}^{\pm}|}$$

Given that $\lambda_i{}^{\pm}$ varies in each iteration, it is important that Δt satisfies the most limiting case (the smallest limit) for the equation above. This corresponds to the largest value of $\lambda_i{}^{\pm}$ for each iteration. Hence the maximum time step that guarantees stability is:

$$\Delta t = \frac{\Delta x \times 1}{max(|\lambda_i{}^{\pm}|)}$$

For the non-linear 1D Euler Equations, using a value of 1 for the Courant Number resulted in oscillations near discontinuities in the numerical solution. According Hirsch (3), these oscillations are a result of non-linearity effects on the values of the Jacobians and the eigenvalues. Decreasing the Courant Number to a value <1 was found to decrease the degree of these oscillations. A value of 0.95 was chosen based on that used in the text for the numerical solution to a similar 1D shock-tube flow. A reasonable assumption is the lowering of the scheme's Courant Number because of non-linearity effects.

$$\Delta t = \frac{\Delta x \times 0.95}{max(|\lambda_i{}^{\pm}|)}$$

It is postulated that while a value lower than 0.95 could be applied to have the scheme move further into the stable regime, the consequently smaller time steps will increase the number of computational steps and hence the overall cost of the analysis.

Question 5

The numerical solutions obtained with different grid point sizes were superimposed on the analytical solution. Close to discontinuities (regions of sharp gradients), the numerical solutions have been smoothed out. This smoothening effect, known as numerical dissipation, is caused by the implicit artificial viscosity terms that appear during the discretisation of the first derivative terms of the numerical schemes for the Euler Equations. They are called artificial as they have dimensions of viscosity but do not share a relationship to the physical viscosity of fluid mechanics, being of purely numerical origin (4). It is important to note that the Euler Equations do not include viscous terms.

The dissipative quality arises from the even-order derivative terms (4) and their coefficients, on the right-hand side (truncation error) of the modified equation (where the modified equation is the exact solution to the difference scheme derived in Question 3), the decreasing dissipatedness with increasing grid points. Looking at the 5 figures, it is concluded that the mainly dissipative behaviour is a result of leading even-order derivatives in the truncation error.

$$\Delta t = \frac{\Delta x \times 0.95}{max(|\lambda_i{}^{\pm}|)}$$

Numerical dissipation decreases with more grid points used hence the solution with 300 grid points is closest to the analytical solution. From this scheme's method of obtaining Δt shown above, decreasing the grid size will decrease Δt due to the smaller Δx. The decreasing dissipatedness of the numerical solution is postulated to be an effect of the decreasing Δt and Δx terms reducing the "weight" of the leading even-order derivatives, thereby decreasing the truncation error and bringing the numerical solution closer to the exact solution as observed.

Despite the mainly dissipative nature of the numerical solution, elements of a dispersive nature can be observed at the second discontinuity in Figure 1 and to a lesser extent, the plateau of Figure 5. For all 3 grid sizes, the plots of entropy/cv are seen to experience sharp overshoots and undershoots before and after the discontinuity. This effect is much less pronounced in the velocity distribution plot of Figure 5. The dispersive quality arises from the odd-order derivative terms and their coefficients, while clearly not a

dominating term in the truncation error as discussed above, has manifested only on the plots of entropy and velocity distribution. The reason for this is unclear and presents a case to be explored further.

The accuracy of the solution improves with increasing grid points because it was postulated that the resultant smaller Δt and Δx act to decrease the truncation error of the modified equation, tending the numerical solution closer to the analytical. While not apparent in this case, the effects of artificial viscosity terms can be desirable as they improve the stability of a solution despite decreasing its accuracy (4). These artificial viscosity terms may be added explicitly to obtain stable and smooth solutions to flow problems. This highlights the dichotomy between stability and accuracy for flow problems with discontinuities.

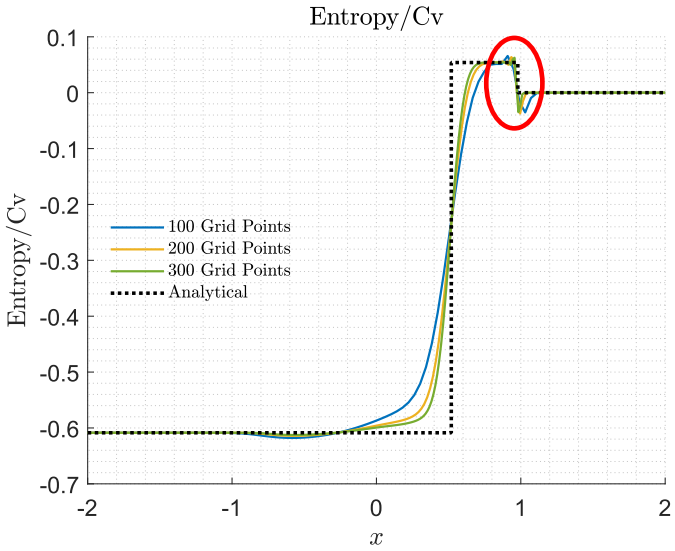


Figure 1. The entropy distribution across the x domain for 3 values of grid points, with a Courant Number = 0.95 was plotted alongside the analytical solution for the Riemann Shock Tube Problem. Some dispersion can still be seen at the second discontinuity circles, circle in red.

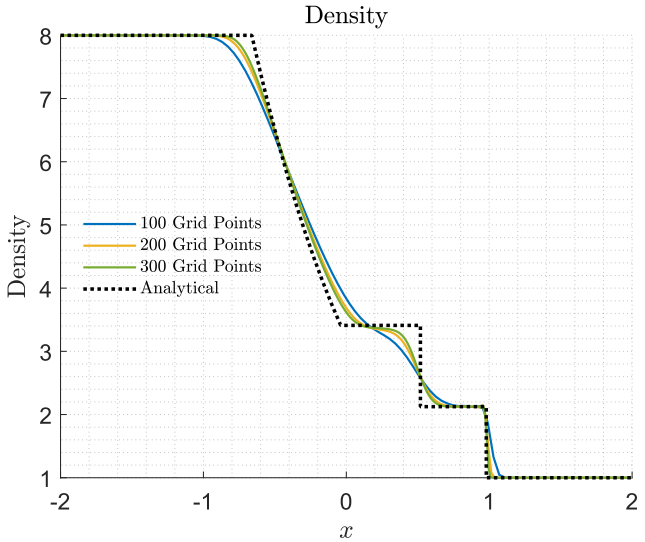


Figure 1. The density distribution across the x domain for 3 values of grid points, with a Courant Number = 0.95 was plotted alongside the analytical solution for the Riemann Shock Tube Problem

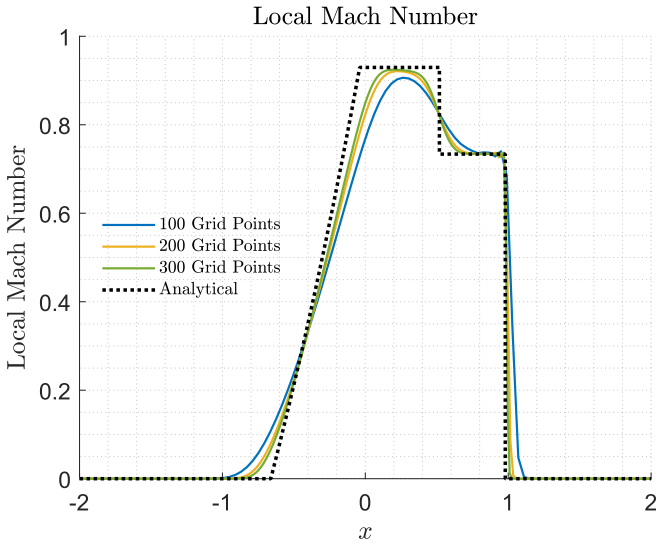


Figure 3. The local mach number distribution across the x domain for 3 values of grid points, with a Courant Number = 0.95 was plotted alongside the analytical solution for the Riemann Shock Tube Problem

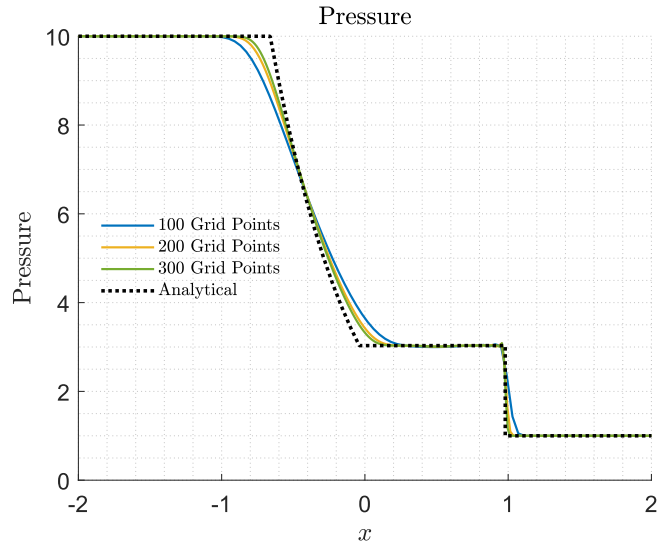


Figure 4. The pressure distribution across the x domain for 3 values of grid points, with a Courant Number = 0.95 was plotted alongside the analytical solution for the Riemann Shock Tube Problem

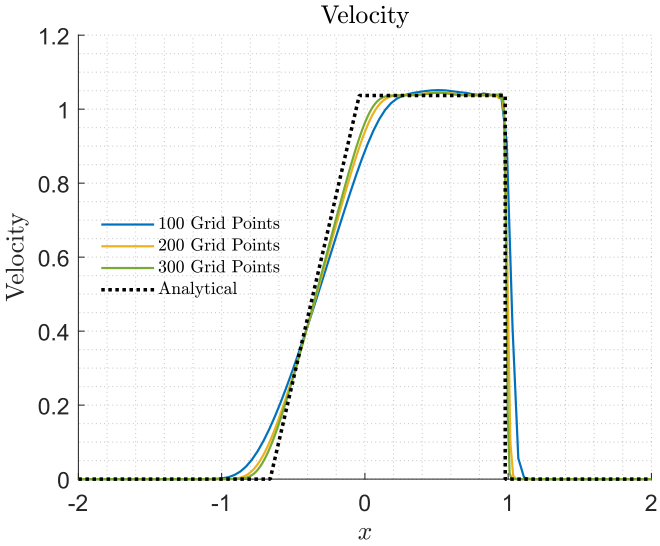


Figure 5. The velocity distribution across the x domain for 3 values of grid points, with a Courant Number = 0.95 was plotted alongside the analytical solution for the Riemann Shock Tube Problem

Bibliography

1. *Flux Vector Splitting of the Inviscid Gasdynamic Equations and Application to the Finite-Difference Methods*.* **Steger, Joseph L and Warming, R.F.** Moffett Field, California : Computational Fluid Dynamics Branch, Armes Research Centre, NASA, 1980, Journal of Computational Physics 40 (1981).

2. **Papadakis, George.** Introduction to CFD A3-313 [part II]. *Introduction to CFD A3-313 [part II].* London : Department of Aeronautics, Imperial College London, 2019.

3. **Hirsch, Charles.** Numerical Computation of Internal and External Flows. *Numerical Computation of Internal and External Flows.* Second. s.l. : John Wiley & Sons, Ltd, 2007.

4. **John D. Anderson, JR.** Computational Fluid Dynamics: The Basics with Applications. *Computational Fluid Dynamics: The Basics with Applications.* Second. s.l. : McGraw-Hill, 1995, 6.

MATLAB Code

```
clear
clc
%This script solves analytically the 1D Euler Equations using the
%Flux-Vector Splitting Method by Steger and Warming
%Written by Mr Xerxes Chong Xian
%CID:01389744
%------------------------------------------------------------------------
%Set/Input Pressure and density ratios and grid sizes
pR=10;
rhoR=8;
tF=0.5;
N=[100 200 300]; %Grid size array
for G=1:length(N)
    % Some constants
    gamma=1.4;
    Courant=0.95; %Courant Number = 0.95<1 to account for non-linearity effects
    x=linspace(-2,2,N(G));
    dx = x(2)-x(1);
    %% Initial Flow Variables(Pressure,Density, Velocity,Energy,Enthalpy)
    p1 = 1;
    rho1 = 1;
    u1 = 0;
    e1 = p1/((gamma-1)*rho1) + (u1^2)/2;
    h1 = e1 + p1/rho1;

    p2 = p1*pR;
    rho2 = rho1*rhoR;
    u2=0;
    e2 = p2/((gamma-1)*rho2) + u2^2/2;
    h2 = e2 + p2/rho2;
    %% Apply Initial Flow Variables across x domain (-2<x<2)
    p = zeros(1,N(G));
    p(1:floor((N(G)+1)/2)) = p2; %set initial values on the left of diaphragm
    p(ceil((N(G)+1)/2):N(G)) = p1; %set initial values on the right of diaphragm

    rho = zeros(1,N(G));
    rho(1:floor((N(G)+1)/2)) = rho2;
    rho(ceil((N(G)+1)/2):N(G)) = rho1;

    u = zeros(1,N(G));
    u(1:floor((N(G)+1)/2)) = u2;
    u(ceil((N(G)+1)/2):N(G)) = u1;

    e = zeros(1,N(G));
    e(1:floor((N(G)+1)/2)) = e2;
    e(ceil((N(G)+1)/2):N(G)) = e1;

    h = zeros(1,N(G));
    h(1:floor((N(G)+1)/2)) = h2;
    h(ceil((N(G)+1)/2):N(G)) = h1;
```

```
`%% Flux-Vector Splitting Scheme
    %Populate U and F matrices
    U=zeros(3,N(G));
    F=zeros(3,N(G));
    U(1,:) = rho;
    U(2,:) = rho.*u;
    U(3,:) = rho.*e;
    F(1,:) = rho.*u;
    F(2,:) = rho.*u.^2+p;
    F(3,:) = rho.*u.*h;
    c = sqrt(gamma.*p./rho);
    %Obtain the eigenvalues
    l1=u-c;
    l2=u;
    l3=u+c;
    %obtain the positive(Lp1,Lp2,Lp3) and negative (Lm1,Lm2,Lm3) eigenvalues
    Lp1=0.5*(l1+abs(l1));
    Lp2=0.5*(l2+abs(l2));
    Lp3=0.5*(l3+abs(l3));
    Lm1=0.5*(l1-abs(l1));
    Lm2=0.5*(l2-abs(l2));
    Lm3=0.5*(l3-abs(l3));

    tcount=0; %Counter to track number of iterations performed
    t=0;
    tic
 while t<tF
     %Calculate maximum time step that satisfies stability condition
     a = abs(u);
     lambda = max(a+c);
     dt = Courant*dx/lambda;

     for i=1:length(x)
         Fpi=rho(i)/(2*gamma)*[Lp1(i)+2*(gamma-1)*Lp2(i)+Lp3(i);(u(i)-c(i))*Lp1(i)+2*(gamma-
1)*u(i)*Lp2(i)+(u(i)+c(i))*Lp3(i);(h(i)-u(i)*c(i))*Lp1(i)+(gamma-1)*(u(i)^2)*Lp2(i)+(h(i)+u(i)*c(i))*Lp3(i)];
         if i==1 %Accounts for Boundary Conditions at start
             Fpim1=rho(i)/(2*gamma)*[Lp1(i)+2*(gamma-1)*Lp2(i)+Lp3(i);(u(i)-c(i))*Lp1(i)+2*(gamma-
1)*u(i)*Lp2(i)+(u(i)+c(i))*Lp3(i);(h(i)-u(i)*c(i))*Lp1(i)+(gamma-1)*(u(i)^2)*Lp2(i)+(h(i)+u(i)*c(i))*Lp3(i)];
         else
             Fpim1=rho(i-1)/(2*gamma)*[Lp1(i-1)+2*(gamma-1)*Lp2(i-1)+Lp3(i-1);(u(i-1)-c(i-1))*Lp1(i-1)+2*(gamma-
1)*u(i-1)*Lp2(i-1)+(u(i-1)+c(i-1))*Lp3(i-1);(h(i-1)-u(i-1)*c(i-1))*Lp1(i-1)+(gamma-1)*(u(i-1)^2)*Lp2(i-1)+(h(i-1)+u(i-
1)*c(i-1))*Lp3(i-1)];
         end
         Fmi=rho(i)/(2*gamma)*[Lm1(i)+2*(gamma-1)*Lm2(i)+Lm3(i);(u(i)-c(i))*Lm1(i)+2*(gamma-
1)*u(i)*Lm2(i)+(u(i)+c(i))*Lm3(i);(h(i)-u(i)*c(i))*Lm1(i)+(gamma-1)*(u(i)^2)*Lm2(i)+(h(i)+u(i)*c(i))*Lm3(i)];
         if i==length(x) %Accounts Boundary Conditions at end
             Fmi1=rho(i)/(2*gamma)*[Lm1(i)+2*(gamma-1)*Lm2(i)+Lm3(i);(u(i)-c(i))*Lm1(i)+2*(gamma-
1)*u(i)*Lm2(i)+(u(i)+c(i))*Lm3(i);(h(i)-u(i)*c(i))*Lm1(i)+(gamma-1)*(u(i)^2)*Lm2(i)+(h(i)+u(i)*c(i))*Lm3(i)];
         else
             Fmi1=rho(i+1)/(2*gamma)*[Lm1(i+1)+2*(gamma-1)*Lm2(i+1)+Lm3(i+1);(u(i+1)-
c(i+1))*Lm1(i+1)+2*(gamma-1)*u(i+1)*Lm2(i+1)+(u(i+1)+c(i+1))*Lm3(i+1);(h(i+1)-
u(i+1)*c(i+1))*Lm1(i+1)+(gamma-1)*(u(i+1)^2)*Lm2(i+1)+(h(i+1)+u(i+1)*c(i+1))*Lm3(i+1)];
         end
         Unew(:,i)=U(:,i)-(dt/dx)*(Fpi-Fpim1 + Fmi1-Fmi);%update the values of U at location i at each time step, U on the
left is the same position on the right but at the next time step
     end
```

```
U=Unew; %Update U array with Unew values
    %Reevaluate all flow parameters parameters
    rho = U(1,:);
    u=U(2,:)./U(1,:);
    e=U(3,:)./U(1,:);
    h = e + p./rho;
    p=(gamma-1)*rho.*(e-0.5*(u.^2));
    c = sqrt(gamma*p./rho);
    %Obtain the eigenvalues
    l1=u-c;
    l2=u;
    l3=u+c;
    %obtain the positive (Lp1,Lp2,Lp3) and negative (Lm1,Lm2,Lm3) eigenvalues
    Lp1=0.5*(l1+abs(l1));
    Lp2=0.5*(l2+abs(l2));
    Lp3=0.5*(l3+abs(l3));
    Lm1=0.5*(l1-abs(l1));
    Lm2=0.5*(l2-abs(l2));
    Lm3=0.5*(l3-abs(l3));

    t=t+dt; %time march forward by the dt found at this iteration
    tcount=tcount+1;
end
disp(tcount) %Display the number of iterations
%% Collect data from each iteration (new arrays created for each iteration
%due to changing grid size affecting array size)
if G==1
    x100=x;
    rho100= U(1,:);
    u100=U(2,:)./U(1,:);
    e100=U(3,:)./U(1,:);
    h100= e + p./rho;
    p100=(gamma-1)*rho.*(e-0.5*(u.^2));
    M100= u./c;
    s100= log(p./rho.^gamma);
elseif G==2
    x200=x;
    rho200= U(1,:);
    u200=U(2,:)./U(1,:);
    e200=U(3,:)./U(1,:);
    h200= e + p./rho;
    p200=(gamma-1)*rho.*(e-0.5*(u.^2));
    M200 = u./c;
    s200= log(p./rho.^gamma);
else
    x300=x;
    rho300 = U(1,:);
    u300=U(2,:)./U(1,:);
    e300=U(3,:)./U(1,:);
    h300= e + p./rho;
    p300=(gamma-1)*rho.*(e-0.5*(u.^2));
    M300= u./c;
    s300= log(p./rho.^gamma);
end

end
```

```
%% Import analytical data from Shock_Tube_Analytic.m and plot data
run('Importfile.m')
shockanalytic=table2array(shockanalytic);
xAn=shockanalytic(:,1);
rhoAn=shockanalytic(:,2);
pAn=shockanalytic(:,3);
uAn=shockanalytic(:,4);
MAn=shockanalytic(:,5);
sAn=shockanalytic(:,6);
figure(1)
hold on
plot(x100,u100,'color',[0, 0.4470, 0.7410],'LineWidth',1.25);
plot(x200,u200,'color',[0.9290, 0.6940, 0.1250],'LineWidth',1.25);
plot(x300,u300,'color',[0.4660, 0.6740, 0.1880],'LineWidth',1.25);
plot(xAn,uAn,'k:','LineWidth',2.0);
set(gca,'FontSize',13)
grid minor
title('Velocity','interpreter','latex');
xlabel('$x$','interpreter','latex');
ylabel('Velocity','interpreter','latex');
set(legend,'Interpreter','latex','location','west','box','off','FontSize',10)
legend('100 Grid Points','200 Grid Points','300 Grid Points','Analytical');
figure(2)
hold on
plot(x100,p100,'color',[0, 0.4470, 0.7410],'LineWidth',1.25);
plot(x200,p200,'color',[0.9290, 0.6940, 0.1250],'LineWidth',1.25);
plot(x300,p300,'color',[0.4660, 0.6740, 0.1880],'LineWidth',1.25);
plot(xAn,pAn,'k:','LineWidth',2.0);
set(gca,'FontSize',13)
grid minor
title('Pressure','interpreter','latex');
xlabel('$x$','interpreter','latex');
ylabel('Pressure','interpreter','latex');
set(legend,'Interpreter','latex','location','west','box','off','FontSize',10)
legend('100 Grid Points','200 Grid Points','300 Grid Points','Analytical');
figure(3)
hold on
plot(x100,rho100,'color',[0, 0.4470, 0.7410],'LineWidth',1.25);
plot(x200,rho200,'color',[0.9290, 0.6940, 0.1250],'LineWidth',1.25);
plot(x300,rho300,'color',[0.4660, 0.6740, 0.1880],'LineWidth',1.25);
plot(xAn,rhoAn,'k:','LineWidth',2.0);
set(gca,'FontSize',13)
grid minor
title('Density','interpreter','latex');
xlabel('$x$','interpreter','latex');
ylabel('Density','interpreter','latex');
set(legend,'Interpreter','latex','location','west','box','off','FontSize',10)
legend('100 Grid Points','200 Grid Points','300 Grid Points','Analytical');
figure(4)
hold on
plot(x100,M100,'color',[0, 0.4470, 0.7410],'LineWidth',1.25);
plot(x200,M200,'color',[0.9290, 0.6940, 0.1250],'LineWidth',1.25);
plot(x300,M300,'color',[0.4660, 0.6740, 0.1880],'LineWidth',1.25);
plot(xAn,MAn,'k:','LineWidth',2.0);
set(gca,'FontSize',13)
grid minor
title('Local Mach Number','interpreter','latex');
xlabel('$x$','interpreter','latex');
ylabel('Local Mach Number','interpreter','latex');
set(legend,'Interpreter','latex','location','west','box','off','FontSize',10)
legend('100 Grid Points','200 Grid Points','300 Grid Points','Analytical');
```

```
figure(5)
hold on
plot(x100,s100,'color',[0, 0.4470, 0.7410],'LineWidth',1.25);
plot(x200,s200,'color',[0.9290, 0.6940, 0.1250],'LineWidth',1.25);
plot(x300,s300,'color',[0.4660, 0.6740, 0.1880],'LineWidth',1.25);
plot(xAn,sAn,'k:','LineWidth',2.0);
set(gca,'FontSize',13)
grid minor
title('Entropy/Cv','interpreter','latex');
xlabel('$x$','interpreter','latex');
ylabel('Entropy/Cv','interpreter','latex');
set(legend,'Interpreter','latex','location','west','box','off','FontSize',10)
legend('100 Grid Points','200 Grid Points','300 Grid Points','Analytical');
```