Consolidated Project Report

Reimann Problem for Certain Hyperbolic Systems

Sahil Sudesh - 2020B4A31868G

August to December 2023

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI - KK BIRLA GOA CAMPUS

A Report

On

Reimann Problem for Certain Hyperbolic Systems

Sahil Sudesh - 2020B4A31868G

August to December 2023

Prepared in partial fulfillment of the Study Oriented Project
Course No: MATH F266

At

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI - KK BIRLA GOA CAMPUS

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI - KK BIRLA GOA CAMPUS

Department of Mathematics

Type of Project : Study Oriented Project

Duration : 11/08/2023 - 08/12/2023

Date Of Submission : 03/12/2023

Title of Project : Reimann Problem for Certain Hyperbolic System

Name of Student : Sahil Sudesh

ID Number : 2020B4A31868G

Discipline of Student : Mathematics & Electronics Engineering

Name and Designation of Instructor: Professor P Minhajul, Department of Mathematics, BITS Goa

Project Areas : Partial Differential Equations, Fluid Dynamics, Reimann Problem

Abstract : To study the Reimann Problem for a Hyperbolic System and implement Numerical methods to get the solution

Signature of Student : [illegible]

Acknowledgements

I extend my heartfelt gratitude to Professor P Minhajul of the Department of Mathematics at Birla Institute of Technology and Science, KK Birla Goa Campus. His mentorship, expert guidance, and unwavering support were pivotal to the success of this Study Oriented Project on ”Riemann Problem for certain Hyperbolic systems, MATH F266.” His encouragement and constructive feedback enriched my understanding and enhanced my problem-solving skills.

I also express my appreciation to the Department of Mathematics for providing a conducive research environment. The academic resources, collaborative opportunities, and scholarly atmosphere contributed significantly to my academic growth. My fellow students and professors were vital to our project's success through insightful discussions and contributions. This project has been an intellectually stimulating and enriching experience, and I am grateful to all those who, directly or indirectly, contributed to its realization.

Sahil Sudesh
2020B4A31868G

Signature of Student : [illegible]

Signature of Professor :

Contents

1 One Dimensional Euler Equations **vii**

1.1 Conservative Formulation vii

1.2 Non Conservative Formulation viii

1.2.1 Primitive Variable Formulation viii

1.2.2 Entropy Formulation x

1.3 Elementary Wave Solutions to Reimann Problem xii

1.3.1 Contact Discontinuities xiv

1.3.2 Rarefaction Waves xv

1.3.3 Shock Waves xvii

2 Reimann Problem for Euler Equations **xix**

2.1 Analytical Solution xix

2.2 Solutions for Pressure and Particle Velocity xx

2.2.1 Left Shock Wave xxii

2.2.2 Left Rarefaction Wave xxii

2.2.3 Right Shock Wave xxiii

2.2.4 Right Rarefaction Wave xxiv

2.3 Exact Solutions in Matlab xxv

2.3.1 Test 1 xxv

2.3.2 Test 2 xxix

2.3.3 Test 3 xxxii

2.3.4 Test 4 xxxvi

2.3.5 Test 5 xl

2.4 Numerical Solutions xliv

2.4.1 Lax Freidrichs Scheme xliv

2.4.2 Lax Wendroff's Scheme xlv

2.4.3 Godunov's Scheme xlvi

2.5 Numerical Solutions in Matlab xlvii

2.5.1 Linear Advection xlvii

2.5.2 Euler Equations l

3 Conclusion **lvi**

4 References **lvii**

Abstract

This project explores the Riemann problem within hyperbolic systems, investigating fundamental components such as shocks, rarefactions, and their interactions. Analytical schemes for left shocks, left rarefactions, and combinations with right shocks and rarefactions were formulated and implemented in MATLAB. Additionally, numerical schemes using Godunov's method and Lax-Friedrichs method were developed and applied to solve these tests, elucidating the behavior of hyperbolic systems under various configurations.

The study delves into the application of these analytical and numerical approaches to the linear advection equation, specifically exploring a square wave initial condition. Godunov's method and Lax-Wendroff scheme were employed to simulate the linear advection equation, providing insights into their effectiveness in capturing wave propagation and resolving discontinuities.

1 One Dimensional Euler Equations

Here, we use both conservative and non-conservative formulas to examine the one-dimensional time-dependent Euler equations with an ideal Equation of State. An outline of the fundamental structure of the Riemann issue solution is provided.

1.1 Conservative Formulation

The conservative formulation of the Euler equations, in differential form, is

$$\mathbf{U}_{\mathrm{t}} + \mathbf{F}(\mathbf{U})_{\mathrm{x}} = \mathbf{0}$$

where $\mathbf{U}$ and $\mathbf{F(U)}$ are the vectors of conserved variables and fluxes, given respectively by,

$$\mathbf{U} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}$$

,

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}$$

Here ρ is density, p is pressure, u is particle velocity and E is total energy per unit volume.

$$E = (\frac{1}{2}u^2 + e)\rho$$

where e is the specific internal energy given by a caloric Equation of State,

$$e = e(\rho, p) = \frac{p}{(\gamma - 1)\rho}$$

Such Conservation laws can be written in Quasi Linear form as,

$$\mathbf{U}_{\mathrm{t}} + \mathbf{A}(\mathbf{U})\mathbf{U}_{\mathrm{x}} = 0$$

where,

$$\mathbf{A}(\mathbf{U}) = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \frac{\partial f_2}{\partial u_3} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} & \frac{\partial f_3}{\partial u_3} \end{bmatrix}$$

1.2 Non Conservative Formulation

1.2.1 Primitive Variable Formulation

It is possible to formulate and solve the equations using variables other than the conserved variables in order to achieve smooth solutions.

Selecting a vector is one option for the one-dimensional scenario. So we choose $\mathbf{W} = (\rho, u, p)^{\mathrm{T}}$.

The proof is as follows :

We have,

$$\mathbf{U}_{\mathrm{t}} + \mathbf{F}(\mathbf{U})_{\mathrm{x}} = 0$$

where,

$$\mathbf{U} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}$$

,

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}$$

Now on substitution and partially differentiating, we get

$$\rho_{\mathrm{t}} + u\rho_{\mathrm{x}} + \rho u_{\mathrm{x}} = 0 \ ----(1)$$

and from the 2nd row, we get the second equation as,

$$\rho_{\mathrm{t}} u + u_{\mathrm{t}}\rho + \rho_{\mathrm{x}} u^2 + 2uu_{\mathrm{x}}\rho + \rho_{\mathrm{x}} = 0 \ ----(2)$$

On rearranging (2) we get,

$$u[\rho_{\mathrm{t}} + u\rho_{\mathrm{x}} + \rho u_{\mathrm{x}}] + \rho[u_{\mathrm{t}} + uu_{\mathrm{x}} + \frac{\rho_{\mathrm{x}}}{\rho}] = 0 \ ----(3)$$

and on substituting (2) in (3), we get

$$[u_{\mathrm{t}} + uu_{\mathrm{x}} + \frac{\rho_{\mathrm{x}}}{\rho}] = 0 \ ----(4)$$

Now, from the 3rd row we obtain, and using appropriate substitution, we get

$$\rho_{\mathrm{t}} + \rho u^2 u_{\mathrm{x}} + up_{\mathrm{x}} = 0 \ ----(5)$$

In Matrix notaion, we can write it as,

$$\mathbf{W}_{\mathrm{t}} + \mathbf{A}(\mathbf{W})\mathbf{W}_{\mathrm{x}} = 0$$

where,

$$\mathbf{W} = \begin{bmatrix} \rho \\ u \\ p \end{bmatrix}, \quad \mathbf{A}(\mathbf{W}) = \begin{bmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \rho a^2 & u \end{bmatrix}$$

One May also find the Eigen values and Eigen Vectors of this system, using $|A - \lambda I| = 0$, getting

$$\begin{vmatrix} u-\lambda & \rho & 0 \\ 0 & u-\lambda & \frac{1}{\rho} \\ 0 & \rho a^2 & u-\lambda \end{vmatrix} = 0 \tag{1}$$

On solving this the EigenValues and EigenVectors come out to be,

$$\lambda_1 = u - a \ , \ \mathbf{K}^{(1)} = \begin{bmatrix} 1 \\ \frac{-a}{\rho} \\ a^2 \end{bmatrix}$$

$$\lambda_2 = u \ , \ \mathbf{K}^{(2)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\lambda_3 = u + a \ , \ \mathbf{K}^{(3)} = \begin{bmatrix} 1 \\ \frac{a}{\rho} \\ a^2 \end{bmatrix}$$

1.2.2 Entropy Formulation

Basically, here we take $\mathbf{W} = (\rho, u, s)^{\mathrm{T}}$.
The entropy s can be written as,

$$s = C_{\mathrm{v}} \ln\left(\frac{p}{\rho^\gamma}\right) + C_0$$

so, p becomes

$$p = C_1 \rho^\gamma e^{\frac{s}{C_{\mathrm{v}}}}$$

The Entropy Formulation satisfies the PDE,

$$s_{\mathrm{t}} + u s_{\mathrm{x}} = 0$$

Now,

$$s = C_{\mathrm{v}}[\ln(p) - \gamma \ln(\rho)] + C_0$$

Differentiating with respect to t and x, we get

$$s_{\mathrm{t}} = \frac{C_{\mathrm{v}}}{p}[p_{\mathrm{t}} - \gamma \rho_{\mathrm{t}}], \quad s_{\mathrm{x}} = \frac{C_{\mathrm{v}}}{p}[p_{\mathrm{x}} - a^2 \rho_{\mathrm{x}}]$$

and we also know that $\rho_{\mathrm{t}} + \rho a^2 u_{\mathrm{x}} + u p_{\mathrm{x}} = 0$, so suing all these, we obtain

$$s_{\mathrm{t}} + u s_{\mathrm{x}} = 0 = \frac{ds}{dt}$$

So in regions of smooth flow, the entropy s is constant along particle paths dx/dt = u. Hence, along a particle path one has the isentropic law given by

$$p = C\rho^{\gamma}$$

Finally, the governing equations for the entropy formulation, written in quasilinear form as,

$$\mathbf{W}_{\mathrm{t}} + \mathbf{A}(\mathbf{W})\mathbf{W}_{\mathrm{x}} = 0$$

with,

$$\mathbf{A}(\mathbf{W}) = \begin{bmatrix} u & \rho & 0 \\ \frac{a^2}{\rho} & u & \frac{1}{\rho}\frac{\partial p}{\partial s} \\ 0 & 0 & u \end{bmatrix}$$

On Solving for the EigenValues and EigenVectors, we get

$$\lambda_1 = u - a \ , \ \mathbf{K}^{(1)} = \begin{bmatrix} 1 \\ \frac{-a}{\rho} \\ 0 \end{bmatrix}$$

$$\lambda_2 = u \ , \ \mathbf{K}^{(2)} = \begin{bmatrix} \frac{-\partial \rho}{\partial s} \\ 0 \\ a^2 \end{bmatrix}$$

$$\lambda_3 = u + a \ , \ \mathbf{K}^{(3)} = \begin{bmatrix} 1 \\ \frac{a}{\rho} \\ 0 \end{bmatrix}$$

1.3 Elementary Wave Solutions to Reimann Problem

The Riemann problem for the one–dimensional, time dependent Euler equations with with data $(\mathbf{U_L},\mathbf{U_R})$ is the IVP

$$\mathbf{U}_\mathrm{t} + \mathbf{F}(\mathbf{U})_\mathrm{x} = \mathbf{0}$$

$$\mathbf{U}(\mathbf{x},\mathbf{0}) = \mathbf{U}^0(\mathbf{x}) = \begin{cases} U_\mathrm{L} & \text{if } x < 0 \\ U_\mathrm{R} & \text{if } x > 0 \end{cases}$$

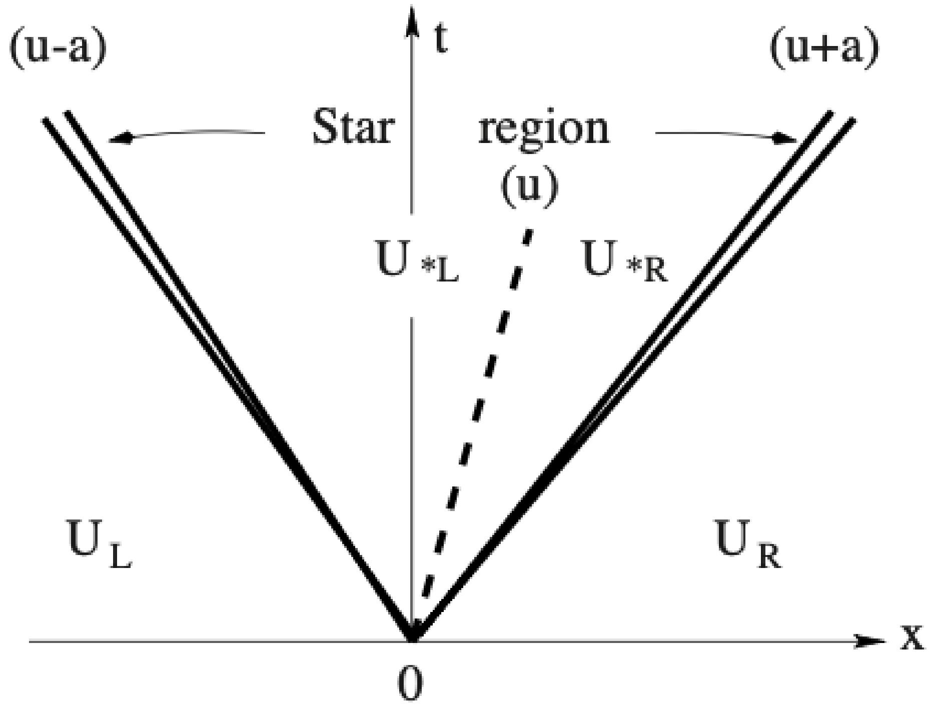

Figure 1: Structure of the solution of the Riemann problem in the x–t plane for the time–dependent, one dimensional Euler equations. There are three wave families associated with the eigenvalues u − a, u and u + a

The three characteristic fields that correspond to the eigenvectors K(i), i = 1, 2, 3, are connected with three waves. When the character of the outer waves is uncertain, we follow the norm of depicting them as two rays coming from the origin for the outer waves and a dashed line for the central wave. Each wave family and its related eigenvalue are displayed. Four steady states are divided by the three waves. The wave associated with the K(2) characteristic field is a **contact discontinuity** and those associated with the K(1), K(3) characteristic fields will either be **rarefaction waves (smooth)** or **shock waves (discontinuities)**. The only exception is the middle wave, which is always a contact discontinuity.

The Figure below shows a particular case in which the left wave is a rarefaction, the middle wave is a contact and the right wave is a shock wave.

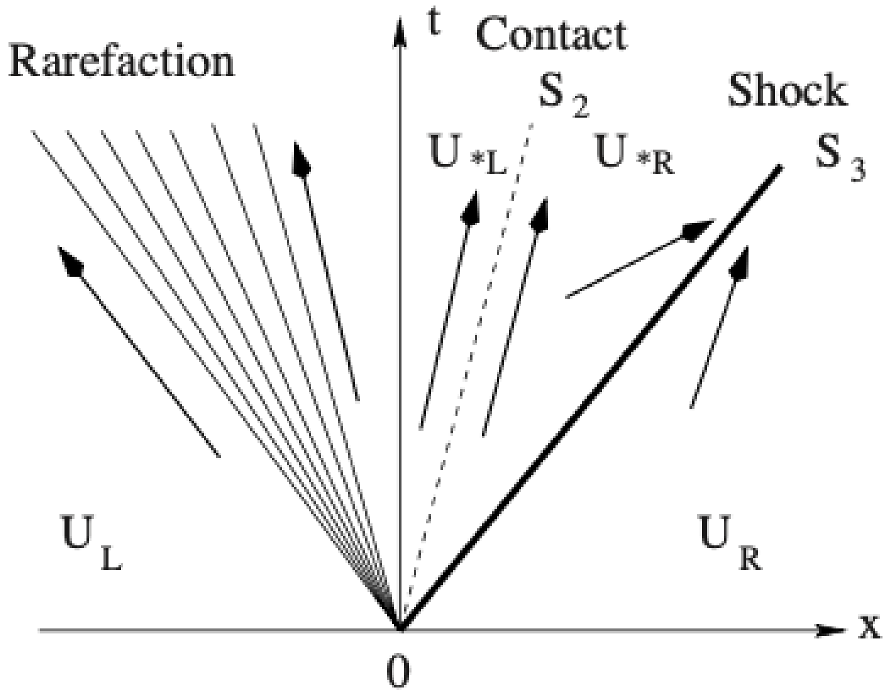

Figure 2: Structure of the solution of the Riemann problem in the x–t plane for the time–dependent, one dimensional Euler equations, in which the left wave is a rarefaction, the middle wave is a contact discontinuity and the right wave is a shock wave

For the rarefaction wave we have, (S_3 is the speed of shock)

$$\lambda_1(U_{\mathrm{L}}) \leq \lambda_1(U_{\mathrm{R}})$$

For the shock wave we have,

$$\lambda_3(U_{*\mathrm{R}}) > S_3 > \lambda_3(U_{*\mathrm{L}})$$

For the contact wave we have, (S_2 is the speed of the contact wave)

$$\lambda_2(U_{*\mathrm{L}}) = S_2 = \lambda_2(U_{*\mathrm{R}})$$

In the Upcoming sections, we look at each of the wave types separately.

1.3.1 Contact Discontinuities

Suppose, we have a system like,

$$\mathbf{U}_{\mathrm{t}} + \mathbf{F}(\mathbf{U})_{\mathrm{x}} = \mathbf{0}$$

where $\mathbf{U}$ and $\mathbf{F(U)}$ are the vectors of conserved variables and fluxes, given respectively by,

$$\mathbf{U} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}$$

,

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}$$

or say in general with,

$$\mathbf{W} = [w_1, w_2, ..., w_{\mathrm{m}}]^{\mathrm{T}},$$

and the Right EigenVectors as,

$$K^{(\mathrm{i})} = [{k_1}^{(\mathrm{i})}, {k_2}^{(\mathrm{i})}, ..., {k_{\mathrm{m}}}^{(\mathrm{i})}]$$

We can write the ith Generalised Reimann Invariant as the (m-1) ODEs

$$\frac{dw_1}{{k_1}^{(\mathrm{i})}} = \frac{dw_2}{{k_2}^{(\mathrm{i})}} = ... = \frac{dw_{\mathrm{m}}}{{k_{\mathrm{m}}}^{(\mathrm{i})}}$$

So in applying this on our system, we get

$$\frac{d\rho}{1} = \frac{d(\rho u)}{u} = \frac{E}{\frac{1}{2}u^2}$$

Manipulation gives us,

$$\mathbf{p = constant, u = constant}$$

So in short, **Contact wave is a discontinuous wave across which both pressure and particle velocity are constant, but density jumps discontinuously as do variables that depend on density, such as specific internal energy, temperature, sound speed, entropy, etc.**

1.3.2 Rarefaction Waves

In the Euler equations, the $K^{(1)}$ and $K^{(3)}$ characteristic fields are related to rarefaction waves. Upon examining the eigenvectors for the primitive-variable formulation, it can be observed that ρ, u and p undergo changes throughout a rarefaction wave. Using Entropy Formulation, we have

$$\mathbf{A(W)} = \begin{bmatrix} u & \rho & 0 \\ \frac{a^2}{\rho} & u & \frac{1}{\rho}\frac{\partial p}{\partial s} \\ 0 & 0 & u \end{bmatrix}$$

On Solving for the EigenValues and EigenVectors, we get

$$\lambda_1 = u - a \ , \ \mathbf{K}^{(1)} = \begin{bmatrix} 1 \\ \frac{-a}{\rho} \\ 0 \end{bmatrix}$$

$$\lambda_2 = u \ , \ \mathbf{K}^{(2)} = \begin{bmatrix} \frac{-\partial\rho}{\partial s} \\ 0 \\ a^2 \end{bmatrix}$$

$$\lambda_3 = u + a \ , \ \mathbf{K}^{(3)} = \begin{bmatrix} 1 \\ \frac{a}{\rho} \\ 0 \end{bmatrix}$$

For K^1, using the Generalised Reimann Invariants,

$$\frac{d\rho}{1} = \frac{d(u)}{\frac{-a}{\rho}} = \frac{ds}{0}$$

From this we get,

$$s = Constant$$

$$du + a\rho d\rho = 0$$

Integrating the 2nd equation, we get

$$u + \int \frac{a}{\rho} d\rho = constant \quad ---(1)$$

Now the Isoentropic law states that $p = C\rho^\gamma$, i.e $(\frac{p}{c})^{\frac{1}{\gamma}} = \rho$, and we also know that $a = \sqrt{\frac{\gamma p}{\rho}}$, using these we get,

$$a = \sqrt{C\gamma}\rho^{\frac{\gamma-1}{2}}$$

Substituting for a in equation (1), we get

$$u + \int \frac{a}{\rho} d\rho \ => \ u + \sqrt{C\gamma} \int \rho^{\frac{\gamma-3}{2}} d\rho \ => \ 2\sqrt{C\gamma} \frac{\rho^{\frac{\gamma-1}{2}}}{\gamma - 1} \ => \ \frac{2a}{\gamma - 1}$$

So finally, we obtain,

$$\mathbf{u} + \frac{\mathbf{2a}}{\gamma - \mathbf{1}} = \mathbf{constant}$$

Similarly we can get the following condition for $\mathbf{K}^{(3)}$,

$$\mathbf{u} - \frac{\mathbf{2a}}{\gamma - \mathbf{1}} = \mathbf{constant}$$

To summarise: a rarefaction wave is a smooth wave associated with the 1 and 3 fields across which ρ, u and p change. The wave has a fan–type shape and is enclosed by two bounding characteristics corresponding to the Head and the Tail of the wave.

1.3.3 Shock Waves

Shock waves are discontinuous waves associated with the genuinely non–linear fields 1 and 3. All three quantities ρ, u and p change across a shock wave. Consider the $K^{(3)}$ characteristic field and assume the corresponding wave is a right–facing shock wave travelling at the constant speed S3; see Fig,

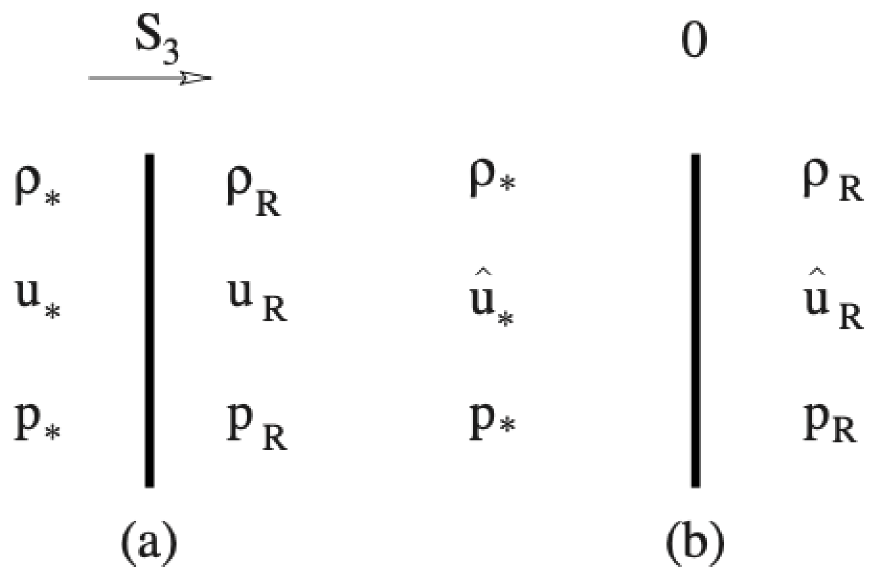

Figure 3: Shock wave facing right: (a) With a stationary frame of reference, the shock has a speed of S3; (b) A moving frame of reference causes the shock to have a speed of zero.

Let the state behind the shock be $\mathbf{W}_* = (\rho_*, u_*, p_*)^{\mathrm{T}}$ and the state ahead of the shock be $\mathbf{W}_{\mathrm{R}} = (\rho_{\mathrm{R}}, u_{\mathrm{R}}, p_{\mathrm{R}})^{\mathrm{T}}$. Densities and pressures remain unaltered while velocities have changed to the relative velocities $\hat{u}_{\mathrm{R}}$ and $\hat{u}_*$ given by,

$$\hat{u}_{\mathrm{R}} = u_{\mathrm{R}} - S_3, \quad \hat{u}_* = u_* - S_3$$

Applying the Rankine Hugoniot conditions, we get

$$\rho_* \hat{u_*} = \rho_{\mathrm{R}} \hat{u_{\mathrm{R}}}$$
$$\rho_* \hat{u_*}^2 + p_* = \rho_{\mathrm{R}} \hat{u_{\mathrm{R}}}^2 + p_{\mathrm{R}}$$
$$u_*(\hat{E_*} + p_*) = u_{\mathrm{R}}(\hat{E_{\mathrm{R}}} + p_{\mathrm{R}})$$

Using the Definitions of Total Energy E , specific internal Energy e , specific enthalpy h and caloric equations of state, followed by algrabic Modifications, we get the following Relations, (For a detailed derivation refer [1])

$$\hat{u_*}^2 = \left(\frac{\rho_{\mathrm{R}}}{\rho_*}\right)\left[\frac{p_{\mathrm{R}} - p_*}{\rho_{\mathrm{R}} - \rho_*}\right], \quad \hat{u_{\mathrm{R}}}^2 = \left(\frac{\rho_*}{\rho_{\mathrm{R}}}\right)\left[\frac{p_{\mathrm{R}} - p_*}{\rho_{\mathrm{R}} - \rho_*}\right]$$

$$h_* - h_{\mathrm{R}} = \left(\frac{1}{2}\right)(p_* - p_{\mathrm{R}})\left[\frac{\rho_* + \rho_{\mathrm{R}}}{\rho_* \rho_{\mathrm{R}}}\right], \quad e_* - e_{\mathrm{R}} = \left(\frac{1}{2}\right)(p_* + p_{\mathrm{R}})\left[\frac{\rho_* - \rho_{\mathrm{R}}}{\rho_* \rho_{\mathrm{R}}}\right]$$

$$\frac{\rho_*}{\rho_{\mathrm{R}}} = \left[\frac{\left(\frac{p_*}{p_{\mathrm{R}}}\right) + \left(\frac{\gamma-1}{\gamma+1}\right)}{\left(\frac{\gamma-1}{\gamma+1}\right)\left(\frac{p_*}{p_{\mathrm{R}}}\right) + 1}\right]$$

Now Introducing Mach Numbers, $M_{\mathrm{R}} = \frac{u_{\mathrm{R}}}{a_{\mathrm{R}}}$, $M_{\mathrm{S}} = \frac{S_3}{a_{\mathrm{R}}}$,

$$\frac{\rho_*}{\rho_{\mathrm{R}}} = \left[\frac{(\gamma+1)(M_{\mathrm{R}} - M_{\mathrm{S}})^2}{(\gamma-1)(M_{\mathrm{R}} - M_{\mathrm{S}})^2 + 1}\right]$$

And the shock speed as a function of pressure ratio across the shock is,

$$S_3 = u_{\mathrm{R}} + a_{\mathrm{R}}\sqrt{\left(\frac{\gamma+1}{2\gamma}\right)\left(\frac{p_*}{p_{\mathrm{R}}}\right) + \left(\frac{\gamma-1}{2\gamma}\right)}$$

The analysis is exactly the same for a 1-shock wave (left facing) moving at velocity S_1.

2 Reimann Problem for Euler Equations

2.1 Analytical Solution

The Riemann problem for the one–dimensional, time dependent Euler equations with with data $(\mathbf{U_L},\mathbf{U_R})$ is the IVP

$$\mathbf{U}_{\mathrm{t}} + \mathbf{F}(\mathbf{U})_{\mathrm{x}} = \mathbf{0}$$

$$\mathbf{U}(\mathbf{x},\mathbf{0}) = \mathbf{U}^0(\mathbf{x}) = \begin{cases} U_{\mathrm{L}} & \text{if } x < 0 \\ U_{\mathrm{R}} & \text{if } x > 0 \end{cases}$$

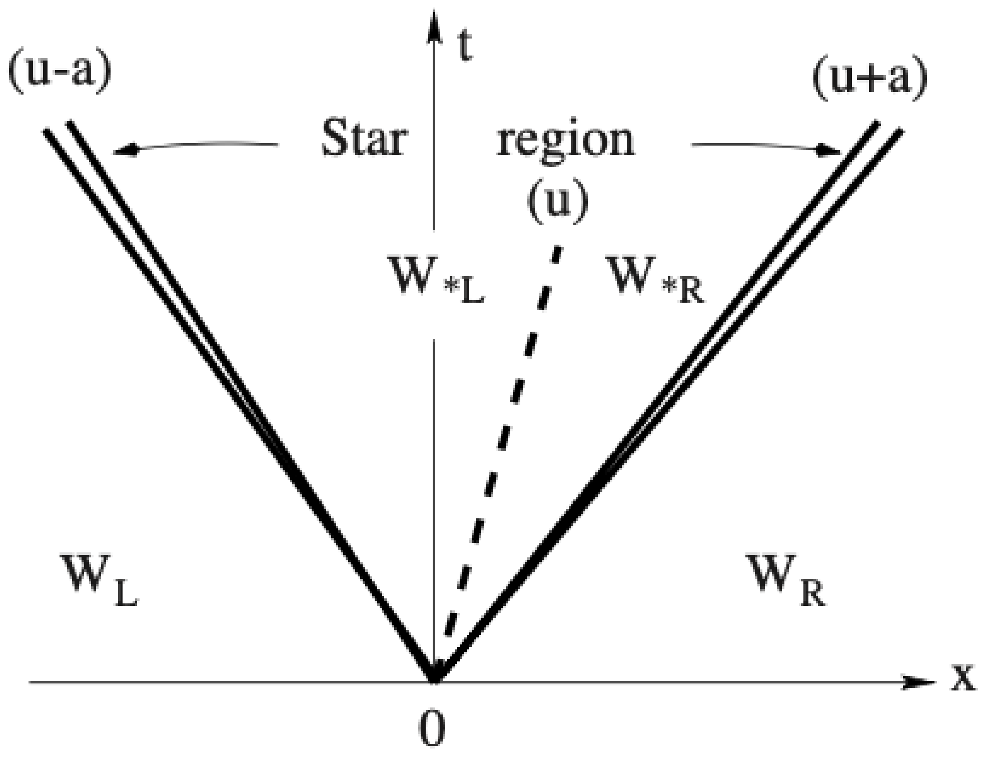

Figure 4: Structure of the solution of the Riemann problem on the x-t plane for the one–dimensional time–dependent Euler equations.

Data consists of just two constant states, which in terms of primitive variables are $\mathbf{W}_{\mathrm{L}} = (\rho_{\mathrm{L}}, \mathbf{u}_{\mathrm{L}}, \mathbf{p}_{\mathrm{L}})^{\mathrm{T}}$ to the left of $x = 0$ and $\mathbf{W}_{\mathrm{L}} = (\rho_{\mathrm{R}}, \mathbf{u}_{\mathrm{R}}, \mathbf{p}_{\mathrm{R}})^{\mathrm{T}}$ to the right of $x = 0$, separated by a discontinuity at $x = 0$.

The exact solution of the Reimann Problem has 3 waves associated with eigenvalues $\lambda_1 = u - a$,$\lambda_2 = u$,$\lambda_3 = u + a$. The three waves separate four constant states, which from left to right are: $\mathbf{W}_{\mathrm{L}}$ (data on the left hand side), $\mathbf{W}_{*\mathrm{L}}$, $\mathbf{W}_{*\mathrm{R}}$ and $\mathbf{W}_{\mathrm{R}}$ (data on the right hand side). The unknown region between the left and right waves, the Star Region, is divided by the middle wave into the two subregions Star Left ($\mathbf{W}_{*\mathrm{L}}$) and Star Right ($\mathbf{W}_{*\mathrm{R}}$).

2.2 Solutions for Pressure and Particle Velocity

The solution for the Pressure p_*, is given by the root of the equation,

$$f(p, \mathbf{W}_{\mathrm{L}}, \mathbf{W}_{\mathrm{R}}) = f(p, \mathbf{W}_{\mathrm{L}}) + f(p, \mathbf{W}_{\mathrm{R}}) + \Delta u = 0$$

where, $\Delta u = u_{\mathrm{R}} - u_{\mathrm{L}}$

Also f_{L} and f_{R} are given by,

$$f(p, \mathbf{W}_{\mathrm{L}}) = \begin{cases} (p - p_{\mathrm{L}}) \left[\frac{A_{\mathrm{L}}}{p + B_{\mathrm{L}}}\right]^{\frac{1}{2}} & \text{if } p > p_{\mathrm{L}} \\ \\ \left(\frac{2a_{\mathrm{L}}}{\gamma - 1}\right) \left[\left(\frac{p}{p_{\mathrm{L}}}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right] & \text{if } p \leq p_{\mathrm{p_L}} \end{cases}$$

$$f(p, \mathbf{W}_{\mathrm{R}}) = \begin{cases} (p - p_{\mathrm{R}}) \left[\frac{A_{\mathrm{R}}}{p + B_{\mathrm{R}}}\right]^{\frac{1}{2}} & \text{if } p > p_{\mathrm{R}} \\ \\ \left(\frac{2a_{\mathrm{R}}}{\gamma - 1}\right) \left[\left(\frac{p}{p_{\mathrm{R}}}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right] & \text{if } p \leq p_{\mathrm{p_R}} \end{cases}$$

and the constants A_{L} , A_{R} , B_{L} , B_{R} are given by,

$$A_{\mathrm{L}} = \left[\frac{2}{(\gamma - 1)\rho_{\mathrm{L}}}\right] , \ A_{\mathrm{R}} = \left[\frac{2}{(\gamma - 1)\rho_{\mathrm{R}}}\right]$$

$$B_{\mathrm{L}} = \left[\frac{\gamma - 1}{(\gamma + 1)}\right] p_{\mathrm{L}}\ ,\ B_{\mathrm{R}} = \left[\frac{\gamma - 1}{(\gamma + 1)}\right] p_{\mathrm{R}}$$

The solution for the particle velocity u_* in the Star Region is,

$$u_* = (\frac{1}{2})(u_{\mathrm{L}} + u_{\mathrm{R}}) + (\frac{1}{2})[f_{\mathrm{R}}(p_*) - f_{\mathrm{L}}(p_*)]$$

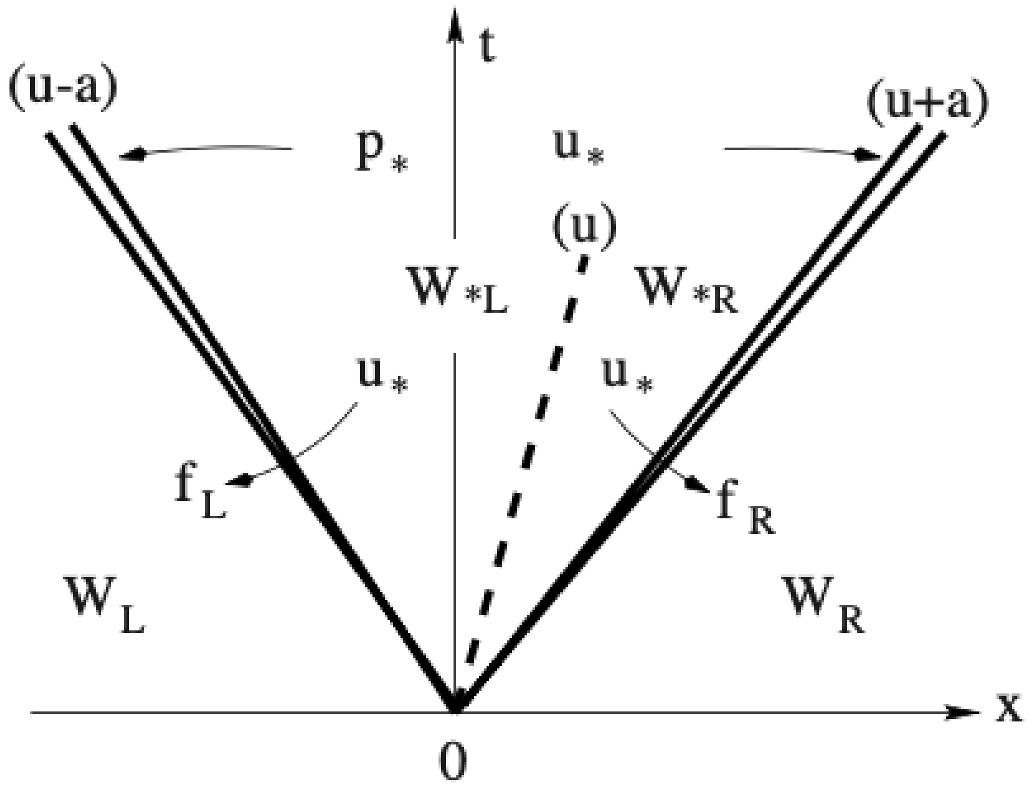

Figure 5: The particle velocity is connected to data on the left and right via functions f_{L} and f_{R}

Now, we still dot not know the sought values $\rho_{*\mathrm{L}}$ and $\rho_{*\mathrm{R}}$ for the density in this region; and also to determine completely the left and right waves. We do this by considering each case separately.

2.2.1 Left Shock Wave

The left Shock wave is identified by the condition $p_* > p_\mathrm{L}$. Now we already know the density,

$$\frac{\rho_{*\mathrm{L}}}{\rho_\mathrm{L}} = \left[\frac{\left(\frac{p_*}{p_\mathrm{L}}\right) + \left(\frac{\gamma-1}{\gamma+1}\right)}{\left(\frac{\gamma-1}{\gamma+1}\right)\left(\frac{p_*}{p_\mathrm{L}}\right) + 1}\right]$$

and also the shock speed as,

$$S_\mathrm{L} = u_\mathrm{L} - a_\mathrm{L}\sqrt{\left(\frac{\gamma+1}{2\gamma}\right)\left(\frac{p_*}{p_\mathrm{L}}\right) + \left(\frac{\gamma-1}{2\gamma}\right)}$$

2.2.2 Left Rarefaction Wave

The left Rarefaction wave is identified by the condition $p_* \leq p_\mathrm{L}$. Using the expression of p_* and u_* in the Star region, The density follows as,

$$\rho_{*\mathrm{L}} = \rho_\mathrm{L}\left(\frac{p_*}{p_\mathrm{L}}\right)^{\frac{1}{\gamma}}, \quad a_{*\mathrm{L}} = a_\mathrm{L}\left(\frac{p_*}{p_\mathrm{L}}\right)^{\frac{\gamma-1}{2\gamma}}$$

The rarefaction wave is enclosed by the Head and the Tail, which are the characteristics of speeds given respectively by,

$$S_\mathrm{HL} = u_\mathrm{L} - a_\mathrm{L}, \quad S_\mathrm{TL} = u_* - a_{*\mathrm{L}}$$

Also we know the slope of the characteristic as,

$$\frac{dx}{dt} = \frac{x}{t} = u - a$$

and from genralised reimann invariants we have,

$$u_\mathrm{L} + \frac{2a_\mathrm{L}}{\gamma-1} = u + \frac{2a}{\gamma-1}$$

The simultaneous solution of these two equations for u and a, use of the definition of the sound speed a and the Isentropic law give the result

$$\mathbf{W}_{\mathrm{Lfan}} = \begin{cases} \rho = \rho_{\mathrm{L}} \left[\frac{2}{\gamma+1} + \frac{\gamma-1}{(\gamma+1)a_{\mathrm{L}}}(u_{\mathrm{L}} - \frac{x}{t}) \right]^{\frac{2}{\gamma-1}}, \\ \\ u = \frac{2}{\gamma+1} \left[a_{\mathrm{L}} + (\frac{\gamma-1}{2}) u_{\mathrm{L}} + \frac{x}{t} \right], \\ \\ p = p_{\mathrm{L}} \left[\frac{2}{\gamma+1} + \frac{\gamma-1}{(\gamma+1)a_{\mathrm{L}}}(u_{\mathrm{L}} - \frac{x}{t}) \right]^{\frac{2\gamma}{\gamma-1}} \end{cases}$$

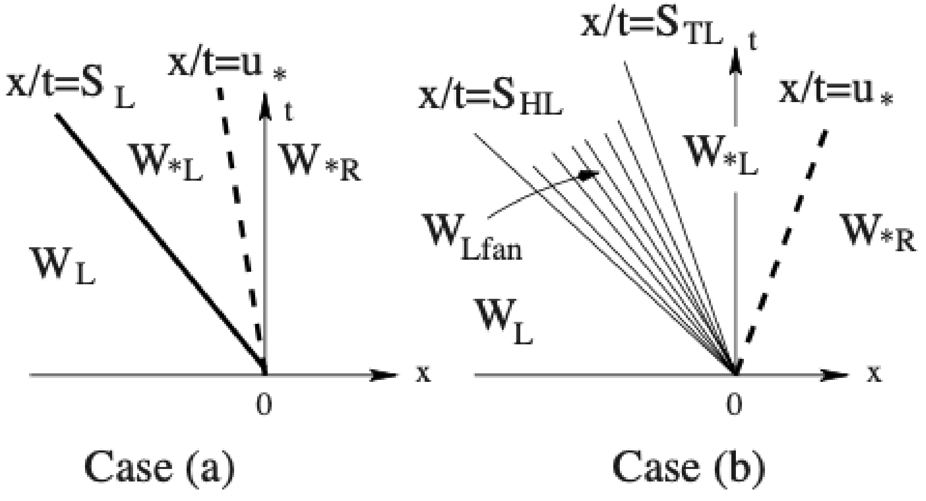

Figure 6: (a) Left wave is a shock (b) Left wave is a rarefaction

A similar kind of analysis on the right side cases yeilds the following.

2.2.3 Right Shock Wave

This wave is identified by the condition $p_* > p_{\mathrm{R}}$.The density $\rho_{*\mathrm{R}}$ is found to be,

$$\frac{\rho_{*\mathrm{R}}}{\rho_{\mathrm{R}}} = \left[\frac{\left(\frac{p_*}{p_{\mathrm{R}}}\right) + \left(\frac{\gamma-1}{\gamma+1}\right)}{\left(\frac{\gamma-1}{\gamma+1}\right)\left(\frac{p_*}{p_{\mathrm{R}}}\right) + 1}\right]$$

and also the shock speed as,

$$S_{\mathrm{L}} = u_{\mathrm{R}} + a_{\mathrm{R}}\sqrt{\left(\frac{\gamma+1}{2\gamma}\right)\left(\frac{p_*}{p_{\mathrm{R}}}\right) + \left(\frac{\gamma-1}{2\gamma}\right)}$$

2.2.4 Right Rarefaction Wave

This wave is identified by the condition $p_* \leq p_{\mathrm{R}}$. The density and sound speed are found to be,

$$\rho_{*\mathrm{R}} = \rho_{\mathrm{R}}\left(\frac{p_*}{p_{\mathrm{R}}}\right)^{\frac{1}{\gamma}}, \quad a_{*\mathrm{R}} = a_{\mathrm{R}}\left(\frac{p_*}{p_{\mathrm{R}}}\right)^{\frac{\gamma-1}{2\gamma}}$$

The speeds for the Head and Tail are given respectively by,

$$S_{\mathrm{HR}} = u_{\mathrm{R}} - a_{\mathrm{R}}, \quad S_{\mathrm{TR}} = u_* - a_{*\mathrm{R}}$$

and finally, the solution inside the fan is found to be

$$\mathbf{W}_{\mathrm{Rfan}} = \begin{cases} \rho = \rho_{\mathrm{R}}\left[\frac{2}{\gamma+1} - \frac{\gamma-1}{(\gamma+1)a_{\mathrm{L}}}(u_{\mathrm{R}} - \frac{x}{t})\right]^{\frac{2}{\gamma-1}}, \\ \\ u = \frac{2}{\gamma+1}\left[-a_{\mathrm{R}} + (\frac{\gamma-1}{2})u_{\mathrm{R}} + \frac{x}{t}\right], \\ \\ p = p_{\mathrm{R}}\left[\frac{2}{\gamma+1} - \frac{\gamma-1}{(\gamma+1)a_{\mathrm{R}}}(u_{\mathrm{R}} - \frac{x}{t})\right]^{\frac{2\gamma}{\gamma-1}} \end{cases}$$

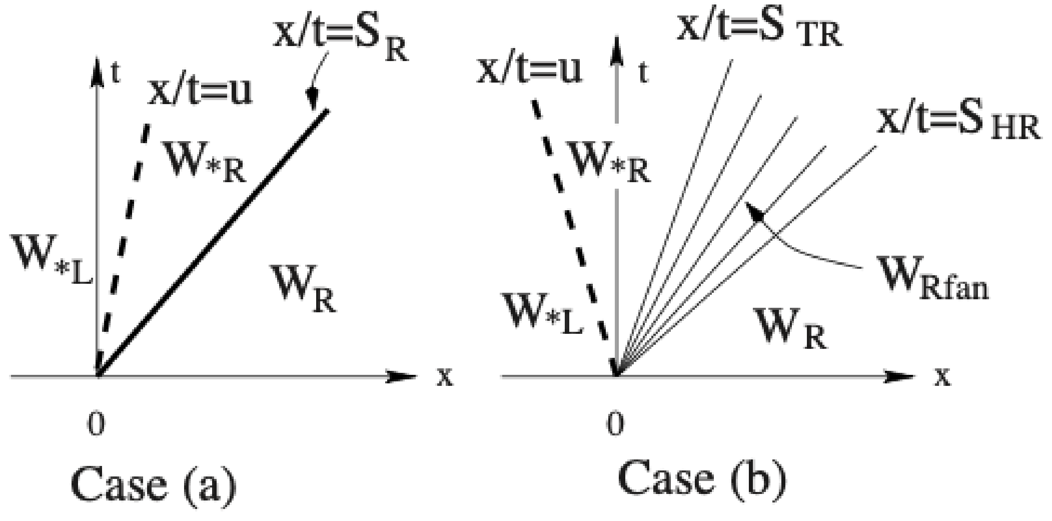

Figure 7: (a) Right wave is a shock (b) Right wave is a rarefaction

2.3 Exact Solutions in Matlab

In this section, we plot the Analytical solution for 4 Tests mainly, which are different versions of the Reimann Problem,

2.3.1 Test 1

Test 1 is the so called Sod test problem [453]; this is a very mild test and its solution consists of a left rarefaction, a contact and a right shock. The main logic is seen below, the complete code can be found in References [2]. The Initial Data was given to be,

Test 1					
ρ_L	u_L	p_L	ρ_R	u_R	p_R
1.0	0.0	1.0	0.125	0.0	0.1

Table 1: Initial Data for Test 1

```
function [data] = Test_1(t)

if nargin < 1
    %set default value
    t = 0.25;
end
%Initial conditions
x0 = 0;
rho_l = 1;
P_l = 1;
u_l = 0;


rho_r = 0.125;
P_r = 0.1;
u_r = 0;

gamma = 1.4;
mu = sqrt( (gamma-1)/(gamma+1) );

%speed of sound
a_l= power( (gamma*P_l/rho_l),0.5);
a_r = power( (gamma*P_r/rho_r),0.5);

%Test 1
P_star = fzero('Function_set',pi);
v_star = 2*(sqrt(gamma)/(gamma - 1))*(1 - power(P_star, (gamma -
    1)/(2*gamma)));
rho_star_R = rho_r*(( (P_star/P_r) + mu^2 )/(1 + mu*mu*(P_star/P_r)));
v_shock = v_star*((rho_star_R/rho_r)/( (rho_star_R/rho_r) - 1));
rho_middle = (rho_l)*power((P_star/P_l),1/gamma);

%Key Positions
x1 = x0 - a_l*t;
x3 = x0 + v_star*t;
x4 = x0 + v_shock*t;
%determining x2
c_2 = a_l - ((gamma - 1)/2)*v_star;
x2 = x0 + (v_star - c_2)*t;

disp("p*")
disp(P_star);
disp("u*")
disp(v_star);
disp("rho*L")
disp(rho_middle);
disp("rho*R")
disp(rho_star_R);
%disp(v_shock);



%start setting values
n_points = 10000;  %set by user
%boundaries (can be set)
x_min = -0.5;
x_max = 0.5;

x = linspace(x_min,x_max,n_points);
data.x = x';
data.rho = zeros(n_points,1); %density
data.P = zeros(n_points,1); %pressure
data.u = zeros(n_points,1); %velocity
data.e = zeros(n_points,1); %internal energy

for index = 1:n_points
    if data.x(index) < x1
       %Solution b4 x1
       data.rho(index) = rho_l;
       data.P(index) = P_l;
       data.u(index) = u_l;
    elseif (x1 <= data.x(index) && data.x(index) <= x2)
       %Solution b/w x1 and x2
       c = mu*mu*((x0 - data.x(index))/t) + (1 - mu*mu)*a_l;
       data.rho(index) = rho_l*power((c/a_l),2/(gamma - 1));
       data.P(index) = P_l*power((data.rho(index)/rho_l),gamma);
       data.u(index) = (1 - mu*mu)*( (-(x0-data.x(index))/t) + a_l);
    elseif (x2 <= data.x(index) && data.x(index) <= x3)
       %Solution b/w x2 and x3
       data.rho(index) = rho_middle;
       data.P(index) = P_star;
       data.u(index) = v_star;
    elseif (x3 <= data.x(index) && data.x(index) <= x4)
       %Solution b/w x3 and x4
       data.rho(index) = rho_star_R;
       data.P(index) = P_star;
       data.u(index) = v_star;
    elseif x4 < data.x(index)
       %Solution after x4
       data.rho(index) = rho_r;
```

```
        data.P(index) = P_r;
        data.u(index) = u_r;
    end
    data.e(index) = data.P(index)/((gamma - 1)*data.rho(index));
end
end
```

The Exact solutions for pressure, speed and densities were found to be,

Test 1			
$p*$	$u*$	$\rho*_{\mathrm{L}}$	$\rho*_{\mathrm{R}}$
0.3031	0.9275	0.4263	0.2656

Table 2: Exact solution for pressure, speed and density for Test 1

And the respective Plots were

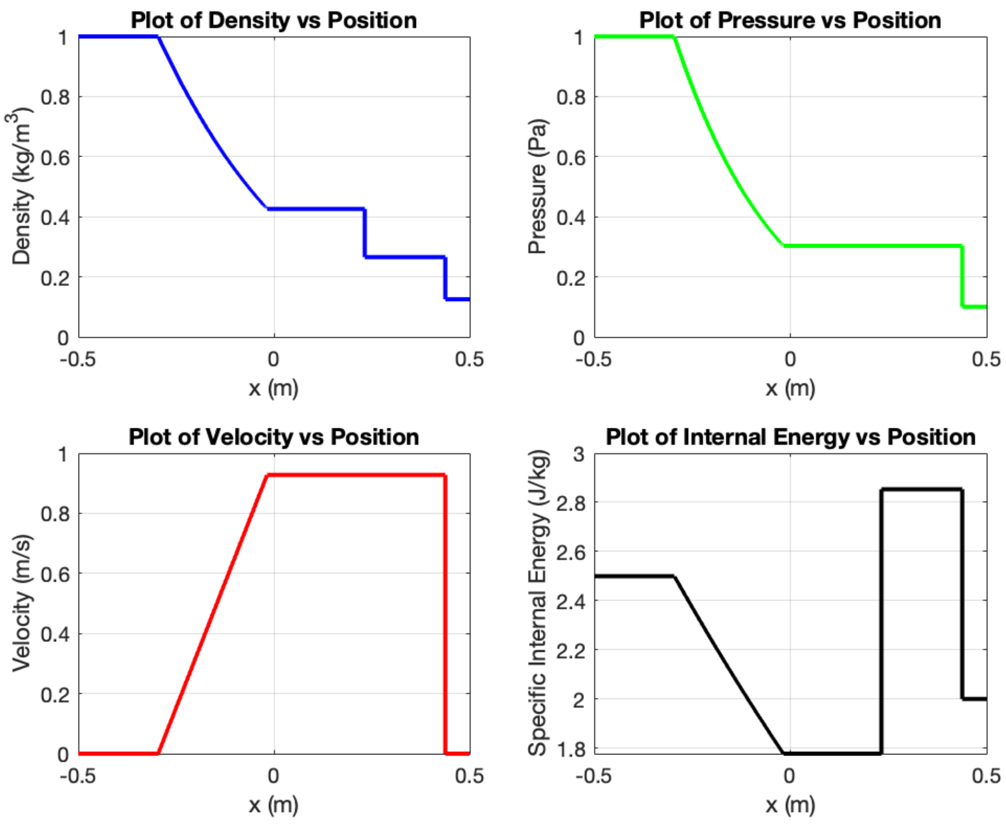

Figure 8: Plots Simulated in Matlab at Time t=0.25 units.

2.3.2 Test 2

Test 2, called the 123 problem, has solution consisting of two strong rarefactions and a trivial stationary contact discontinuity. The main logic is seen below, the complete code can be found in References [2]. The Initial Data was given to be,

Test 2					
ρ_L	u_L	p_L	ρ_R	u_R	p_R
1.0	-2.0	0.4	1.0	2.0	0.4

Table 3: Initial Data for Test 2

```
function [data] = Test_2(t)

if nargin < 1
    %set default value
    t = 0.25;
end
%Initial conditions
x0 = 0;
rho_l = 1;
P_l = 0.4;
u_l = -2;


rho_r = 1;
P_r = 0.4;
u_r = 2;

gamma = 1.4;
mu = sqrt( (gamma-1)/(gamma+1) );

%speed of sound
a_l= power( (gamma*P_l/rho_l),0.5);
a_r = power( (gamma*P_r/rho_r),0.5);

%Test 2
P_star = fzero('Function_set',0.001);
```

```
v_star = u_r +
    2*(a_r/(gamma-1))*(-1+power((P_star/P_r),((gamma-1)/(2*gamma))));
rho_star_R = (rho_r)*power((P_star/P_r),1/gamma);
%v_shock = v_star*((rho_star_R/rho_r)/( (rho_star_R/rho_r) - 1));
rho_star_L = (rho_l)*power((P_star/P_l),1/gamma);

%Key Positions
x1 = x0+(u_l - a_l)*t;
x3 = x0 + v_star*t;
x5 = x0+(u_r + a_r)*t;
%determining x2
a_star_L=power( (gamma*P_star/rho_star_L),0.5);
a_star_R=power( (gamma*P_star/rho_star_R),0.5);
%c_2 = a_l - ((gamma - 1)/2)*v_star;
%x2 = x0 + (v_star - c_2)*t;
x2 = x0 + (v_star - a_star_L)*t;
%Determine x4
%c_4 = a_r + ((gamma + 1)/2)*v_star;
%x4 = x0 + (v_star +c_4)*t;
x4 = x0 + (v_star +a_star_R)*t;

disp("p*")
disp(P_star);
disp("u*")
disp(v_star);
disp("rho*L")
disp(rho_star_L);
disp("rho*R")
disp(rho_star_R);
%disp(v_shock);



%start setting values
n_points = 1000000;  %set by user
%boundaries (can be set)
x_min = -0.5;
x_max = 0.5;

x = linspace(x_min,x_max,n_points);
data.x = x';
data.rho = zeros(n_points,1); %density
data.P = zeros(n_points,1); %pressure
data.u = zeros(n_points,1); %velocity
data.e = zeros(n_points,1); %internal energy
```

```
for index = 1:n_points
    if data.x(index) < x1
        %Solution b4 x1 (b4 fan_l)
        data.rho(index) = rho_l;
        data.P(index) = P_l;
        data.u(index) = u_l;
    elseif (x1 <= data.x(index) && data.x(index) <= x2)
        %Solution b/w x1 and x2 (in fan_l)
        c = mu*mu*(u_l- ((data.x(index))/t)) + (1 - mu*mu)*a_l;
        data.rho(index) = rho_l*power((c/a_l),2/(gamma - 1));
        data.P(index) = P_l*power((data.rho(index)/rho_l),gamma);
        data.u(index) = (1 - mu*mu)*( ((data.x(index))/t) +
            a_l+((gamma-1)/2)*u_l);
    elseif (x2 <= data.x(index) && data.x(index) < x3)
        %Solution b/w x2 and x3 (btw fan_l and contact)
        data.rho(index) = rho_star_L;
        data.P(index) = P_star;
        data.u(index) = v_star;
    elseif (x3 <= data.x(index) && data.x(index) < x4)
        %Solution b/w x3 and x4 (btw contact and fan_r)
        data.rho(index) = rho_star_R;
        data.P(index) = P_star;
        data.u(index) = v_star;
    elseif (x4 <= data.x(index) && data.x(index) < x5)
        %Solution b/w x4 and x5 (in fan_r)
        c = -mu*mu*(u_r- ((data.x(index))/t)) + (1 - mu*mu)*a_r;
        data.rho(index) = rho_r*power((c/a_r),2/(gamma - 1));
        data.P(index) = P_r*power((data.rho(index)/rho_r),gamma);
        data.u(index) = (1 - mu*mu)*( ((data.x(index))/t) -
            a_r+((gamma-1)/2)*u_r);
    elseif x5 < data.x(index)
        %Solution after x5 (after fan_r)
        data.rho(index) = rho_r;
        data.P(index) = P_r;
        data.u(index) = u_r;
    end
    data.e(index) = data.P(index)/((gamma - 1)*data.rho(index));
end
end
```

The Exact solutions for pressure, speed and densities were found to be,

Test 2			
$p*$	$u*$	$\rho*_{\mathrm{L}}$	$\rho*_{\mathrm{R}}$
0.0019	2.2204e-15	0.0219	0.0219

Table 4: Exact solution for pressure, speed and density for Test 2

And the respective Plots were

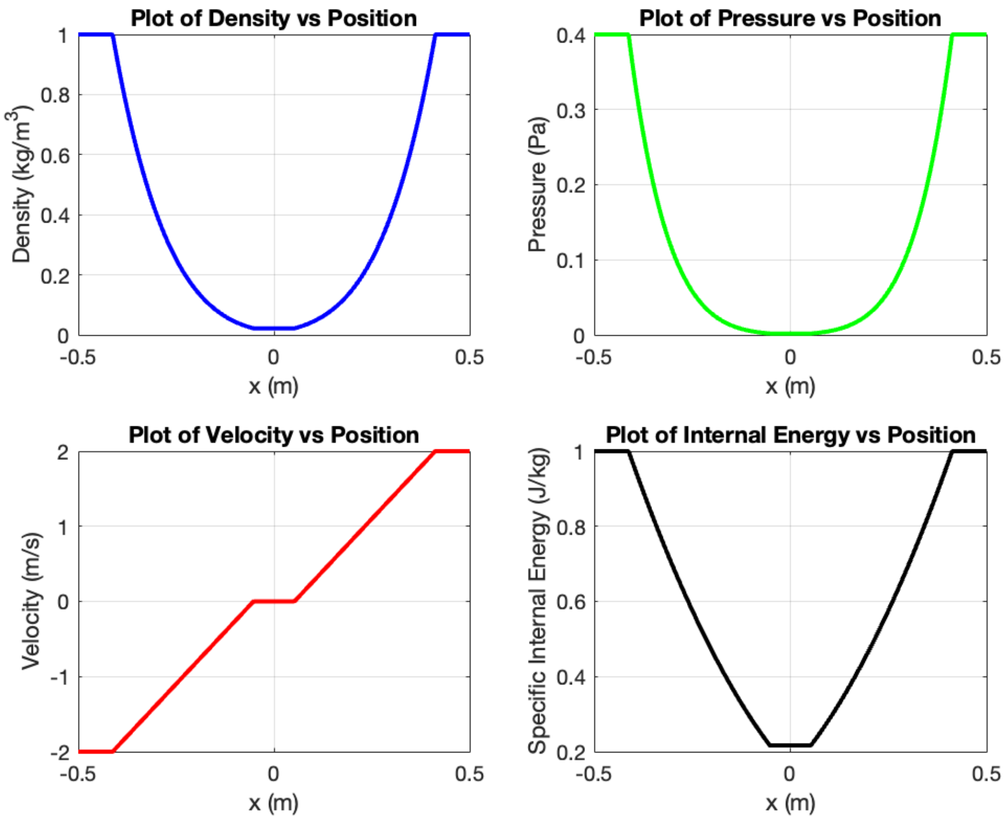

Figure 9: Plots Simulated in Matlab at Time t=0.15 units.

2.3.3 Test 3

Test 3 is a very severe test problem, the solution of which contains a left rarefaction, a contact and a right shock; this test is actually the left half of the blast wave problem of Woodward and Colella . The main logic is seen below, the complete code can be found in References [2]. The Initial Data was given to be,

Test 3					
ρ_L	u_L	p_L	ρ_R	u_R	p_R
1.0	0.0	1000.0	1.0	0.0	0.01

Table 5: Initial Data for Test 3

```
function [data] = Test_3(t)

if nargin < 1
   %set default value
   t = 0.25;
end
%Initial conditions
x0 = 0;
rho_l = 1;
P_l = 1000;
u_l = 0;


rho_r = 1;
P_r = 0.01;
u_r = 0;

gamma = 1.4;
mu = sqrt( (gamma-1)/(gamma+1) );

%speed of sound
a_l= power( (gamma*P_l/rho_l),0.5);
a_r = power( (gamma*P_r/rho_r),0.5);

%Test 1
P_star = fzero('Function_set',500);
%v_star = 2*(sqrt(gamma)/(gamma - 1))*(1 - power(P_star, (gamma -
   1)/(2*gamma)));
v_star = u_l +
   2*(a_l/(gamma-1))*(1-power((P_star/P_l),((gamma-1)/(2*gamma))));
rho_star_R = rho_r*(( (P_star/P_r) + mu^2 )/(1 + mu*mu*(P_star/P_r)));
v_shock = v_star*((rho_star_R/rho_r)/( (rho_star_R/rho_r) - 1));
rho_star_L = (rho_l)*power((P_star/P_l),1/gamma);
```

```
%Key Positions
x1 = x0 - a_l*t;
x3 = x0 + v_star*t;
x4 = x0 + v_shock*t;
%determining x2
c_2 = a_l - ((gamma - 1)/2)*v_star;
x2 = x0 + (v_star - c_2)*t;

disp("p*")
disp(P_star);
disp("u*")
disp(v_star);
disp("rho*L")
disp(rho_star_L);
disp("rho*R")
disp(rho_star_R);
%disp(v_shock);



%start setting values
n_points = 1000000;  %set by user
%boundaries (can be set)
x_min = -0.5;
x_max = 0.5;

x = linspace(x_min,x_max,n_points);
data.x = x';
data.rho = zeros(n_points,1); %density
data.P = zeros(n_points,1); %pressure
data.u = zeros(n_points,1); %velocity
data.e = zeros(n_points,1); %internal energy

for index = 1:n_points
    if data.x(index) < x1
       %Solution b4 x1 (b4 fan)
       data.rho(index) = rho_l;
       data.P(index) = P_l;
       data.u(index) = u_l;
    elseif (x1 <= data.x(index) && data.x(index) <= x2)
       %Solution b/w x1 and x2 (fan)
       c = mu*mu*((x0 - data.x(index))/t) + (1 - mu*mu)*a_l;
       data.rho(index) = rho_l*power((c/a_l),2/(gamma - 1));
       data.P(index) = P_l*power((data.rho(index)/rho_l),gamma);
```

```
        data.u(index) = (1 - mu*mu)*( (-(x0-data.x(index))/t) + a_l);
    elseif (x2 <= data.x(index) && data.x(index) <= x3)
        %Solution b/w x2 and x3(btw fan and contact)
        data.rho(index) = rho_star_L;
        data.P(index) = P_star;
        data.u(index) = v_star;
    elseif (x3 <= data.x(index) && data.x(index) <= x4)
        %Solution b/w x3 and x4 (btw contact and shock)
        data.rho(index) = rho_star_R;
        data.P(index) = P_star;
        data.u(index) = v_star;
    elseif x4 < data.x(index)
        %Solution after x4 (after shock)
        data.rho(index) = rho_r;
        data.P(index) = P_r;
        data.u(index) = u_r;
    end
    data.e(index) = data.P(index)/((gamma - 1)*data.rho(index));
end
end
```

The Exact solutions for pressure, speed and densities were found to be,

Test 3			
$p*$	$u*$	$\rho*_{\mathrm{L}}$	$\rho*_{\mathrm{R}}$
460.8938	19.5975	0.5751	5.9992

Table 6: Exact solution for pressure, speed and density for Test 3

And the respective Plots were

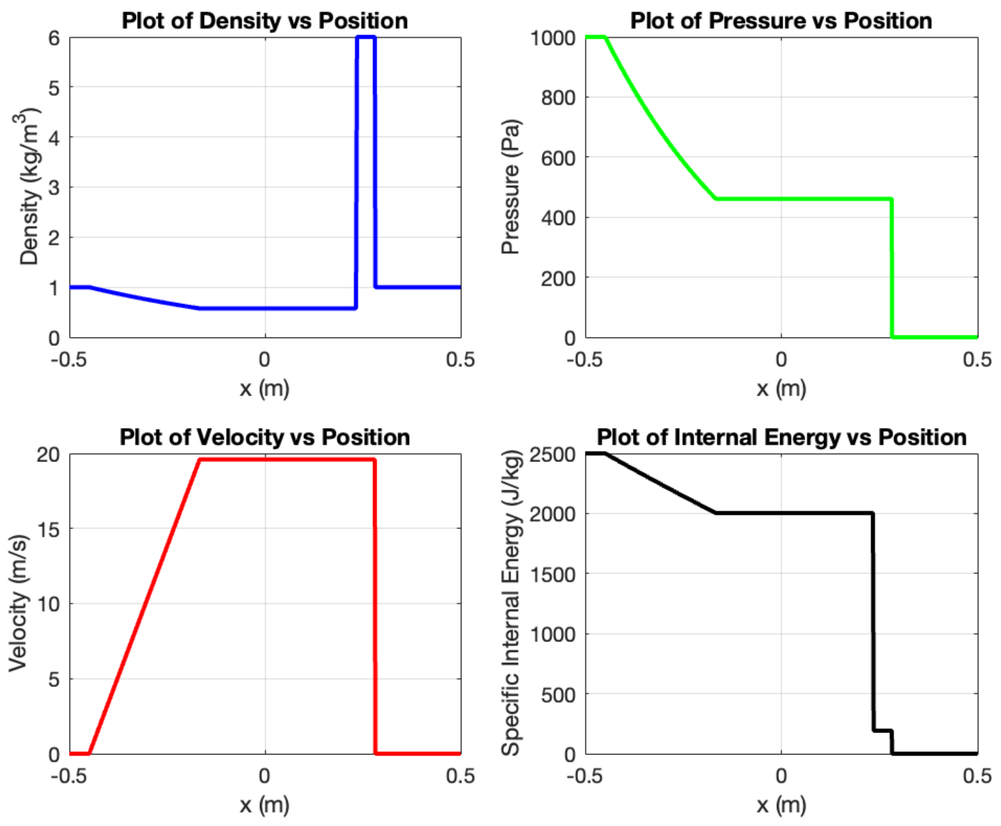

Figure 10: Plots Simulated in Matlab at Time t=0.012 units.

2.3.4 Test 4

Test 4 is the right half of the Woodward and Colella problem; its solution contains a left shock, a contact discontinuity and a right rarefaction. The main logic is seen below, the complete code can be found in References [2]. The Initial Data was given to be,

Test 4					
$\rho_{\rm L}$	$u_{\rm L}$	$p_{\rm L}$	$\rho_{\rm R}$	$u_{\rm R}$	$p_{\rm R}$
1.0	0.0	0.01	1.0	0.0	100.0

Table 7: Initial Data for Test 4

```
function [data] = Test_4(t)

if nargin < 1
    %set default value
    t = 0.25;
end
%Initial conditions
x0 = 0;
rho_l = 1;
P_l = 0.01;
u_l = 0;


rho_r = 1;
P_r = 100;
u_r = 0;

gamma = 1.4;
mu = sqrt( (gamma-1)/(gamma+1) );

%speed of sound
a_l= power( (gamma*P_l/rho_l),0.5);
a_r = power( (gamma*P_r/rho_r),0.5);

%Test 1
P_star = fzero('Function_set',50);
%v_star = 2*(sqrt(gamma)/(gamma - 1))*(1 - power(P_star, (gamma -
    1)/(2*gamma)));
v_star = u_r -
    2*(a_r/(gamma-1))*(1-power((P_star/P_r),((gamma-1)/(2*gamma))));
rho_star_R = (rho_r)*power((P_star/P_r),1/gamma);
rho_star_L = rho_l*(( (P_star/P_l) + mu^2 )/(1 + mu*mu*(P_star/P_l)));
v_shock = v_star*((rho_star_L/rho_l)/( (rho_star_L/rho_l) - 1));

%Key Positions
x1 = x0 + v_shock*t;
x2 = x0 + v_star*t;
x4 = x0 + a_r*t;
%determining x3
c_3 = a_r + ((gamma - 1)/2)*v_star;
x3 = x0 + (v_star + c_3)*t;

disp("p*")
disp(P_star);
disp("u*")
disp(v_star);
disp("rho*L")
disp(rho_star_L);
disp("rho*R")
disp(rho_star_R);
%disp(v_shock);



%start setting values
n_points = 1000000;  %set by user
%boundaries (can be set)
x_min = -0.5;
x_max = 0.5;

x = linspace(x_min,x_max,n_points);
data.x = x';
data.rho = zeros(n_points,1); %density
data.P = zeros(n_points,1); %pressure
data.u = zeros(n_points,1); %velocity
data.e = zeros(n_points,1); %internal energy

for index = 1:n_points
    if data.x(index) < x1
       %Solution b4 x1 (b4 shock)
       data.rho(index) = rho_l;
       data.P(index) = P_l;
       data.u(index) = u_l;
    elseif (x1 <= data.x(index) && data.x(index) <= x2)
       %Solution b/w x1 and x2 (shock to contact)
       data.rho(index) = rho_star_L;
       data.P(index) = P_star;
       data.u(index) = v_star;
    elseif (x2 <= data.x(index) && data.x(index) <= x3)
       %Solution b/w x2 and x3 (contact to fan)
       data.rho(index) = rho_star_R;
       data.P(index) = P_star;
       data.u(index) = v_star;
    elseif (x3 <= data.x(index) && data.x(index) <= x4)
       %Solution b/w x3 and x4 (fan)
       c = -mu*mu*(u_r- ((data.x(index))/t)) + (1 - mu*mu)*a_r;
       data.rho(index) = rho_r*power((c/a_r),2/(gamma - 1));
       data.P(index) = P_r*power((data.rho(index)/rho_r),gamma);
       data.u(index) = (1 - mu*mu)*( ((data.x(index))/t) -
           a_r+((gamma-1)/2)*u_r);
    elseif x4 < data.x(index)
        %Solution after x4 (after fan)
        data.rho(index) = rho_r;
        data.P(index) = P_r;
        data.u(index) = u_r;
    end
    data.e(index) = data.P(index)/((gamma - 1)*data.rho(index));
end
end
```

The Exact solutions for pressure, speed and densities were found to be,

Test 4			
$p*$	$u*$	$\rho*_\mathrm{L}$	$\rho*_\mathrm{R}$
46.0950	-6.1963	5.9924	0.5751

Table 8: Exact solution for pressure, speed and density for Test 4

And the respective Plots were

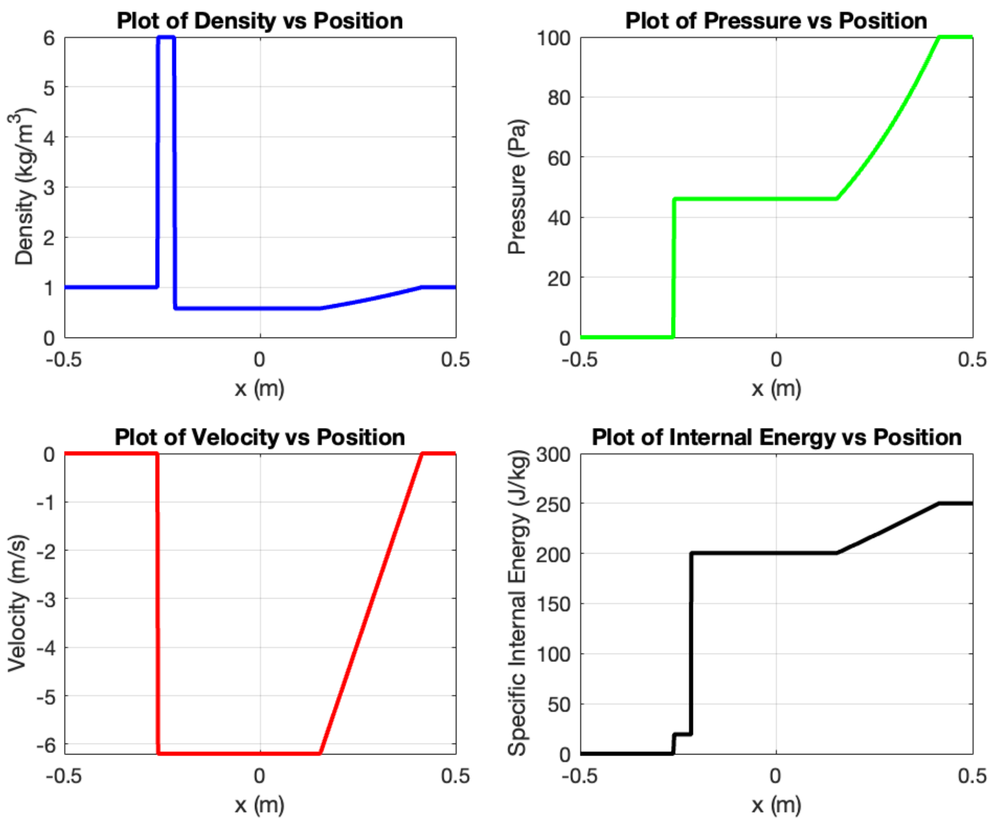

Figure 11: Plots Simulated in Matlab at Time t=0.035 units.

2.3.5 Test 5

Test 5 is made up of the right and left shocks emerging from the solution to tests 3 and 4 respectively; its solution represents the collision of these two strong shocks and consists of a left facing shock (travelling very slowly to the right), a right travelling contact discontinuity and a right travelling shock wave. The main logic is seen below, the complete code can be found in References [2]. The Initial Data was given to be,

Test 5					
ρ_L	u_L	p_L	ρ_R	u_R	p_R
5.99924	19.5975	460.894	5.99242	-6.19633	46.0950

Table 9: Initial Data for Test 5

```
function [data] = Test_5(t)

if nargin < 1
    %set default value
    t = 0.25;
end
%Initial conditions
x0 = 0;
rho_l = 5.99924;
P_l = 460.894;
u_l = 19.5975;


rho_r = 5.99242;
P_r = 46.0950;
u_r = -6.19633;

gamma = 1.4;
mu = sqrt( (gamma-1)/(gamma+1) );

%speed of sound
a_l= power( (gamma*P_l/rho_l),0.5);
a_r = power( (gamma*P_r/rho_r),0.5);

A_R = 2/((gamma+1)*rho_r);
B_R = mu*mu*P_r;
A_L = 2/((gamma+1)*rho_l);
B_L = mu*mu*P_l;


%Test 1
P_star = fzero('Function_set',2000);
v_star = u_l - (P_star-P_l)*power((A_L/(P_star+B_L)),0.5);
v_shock_r = u_r + a_r*power((((gamma+1)/(2*gamma))*(P_star/P_r)) +
    ((gamma-1)/(2*gamma)) ,0.5);
v_shock_l = u_l - a_l*power((((gamma+1)/(2*gamma))*(P_star/P_l)) +
    ((gamma-1)/(2*gamma)) ,0.5);
rho_star_R = rho_r*(( (P_star/P_r) + mu^2 )/(1 + mu*mu*(P_star/P_r)));
rho_star_L = rho_l*(( (P_star/P_l) + mu^2 )/(1 + mu*mu*(P_star/P_l)));
```

```
%v_shock_r = v_star*((rho_star_R/rho_r)/( (rho_star_R/rho_r) - 1));
%v_shock_l = v_star*((rho_star_L/rho_l)/( (rho_star_L/rho_l) - 1));

%Key Positions
x1 = x0 - v_shock_l*t;
x2 = x0 + v_star*t;
x3 = x0 + v_shock_r*t;
%determining x2
%c_2 = a_l - ((gamma - 1)/2)*v_star;
%x2 = x0 + (v_star - c_2)*t;

disp("p*")
disp(P_star);
disp("u*")
disp(v_star);
disp("rho*L")
disp(rho_star_L);
disp("rho*R")
disp(rho_star_R);
%disp(v_shock);



%start setting values
n_points = 10000;  %set by user
%boundaries (can be set)
x_min = -0.5;
x_max = 0.5;

x = linspace(x_min,x_max,n_points);
data.x = x';
data.rho = zeros(n_points,1); %density
data.P = zeros(n_points,1); %pressure
data.u = zeros(n_points,1); %velocity
data.e = zeros(n_points,1); %internal energy

for index = 1:n_points
    if data.x(index) < x1
        %Solution b4 x1 (b4 left shock)
        data.rho(index) = rho_l;
        data.P(index) = P_l;
        data.u(index) = u_l;
    elseif (x1 <= data.x(index) && data.x(index) <= x2)
        %Solution b/w x1 and x2 (btw left shock and contact)
        data.rho(index) = rho_star_L;
        data.P(index) = P_star;
        data.u(index) = v_star;
    elseif (x2 <= data.x(index) && data.x(index) <= x3)
        %Solution b/w x2 and x3 (btw contact and right shock)
        data.rho(index) = rho_star_R;
        data.P(index) = P_star;
        data.u(index) = v_star;
    elseif x3 < data.x(index)
        %Solution after x3 (after right shock)
        data.rho(index) = rho_r;
        data.P(index) = P_r;
        data.u(index) = u_r;
    end
    data.e(index) = data.P(index)/((gamma - 1)*data.rho(index));
end
end
```

The Exact solutions for pressure, speed and densities were found to be,

Test 5			
$p*$	$u*$	$\rho*_{\mathrm{L}}$	$\rho*_{\mathrm{R}}$
1691.6	8.6898	14.2823	31.0426

Table 10: Exact solution for pressure, speed and density for Test 5

And the respective Plots were

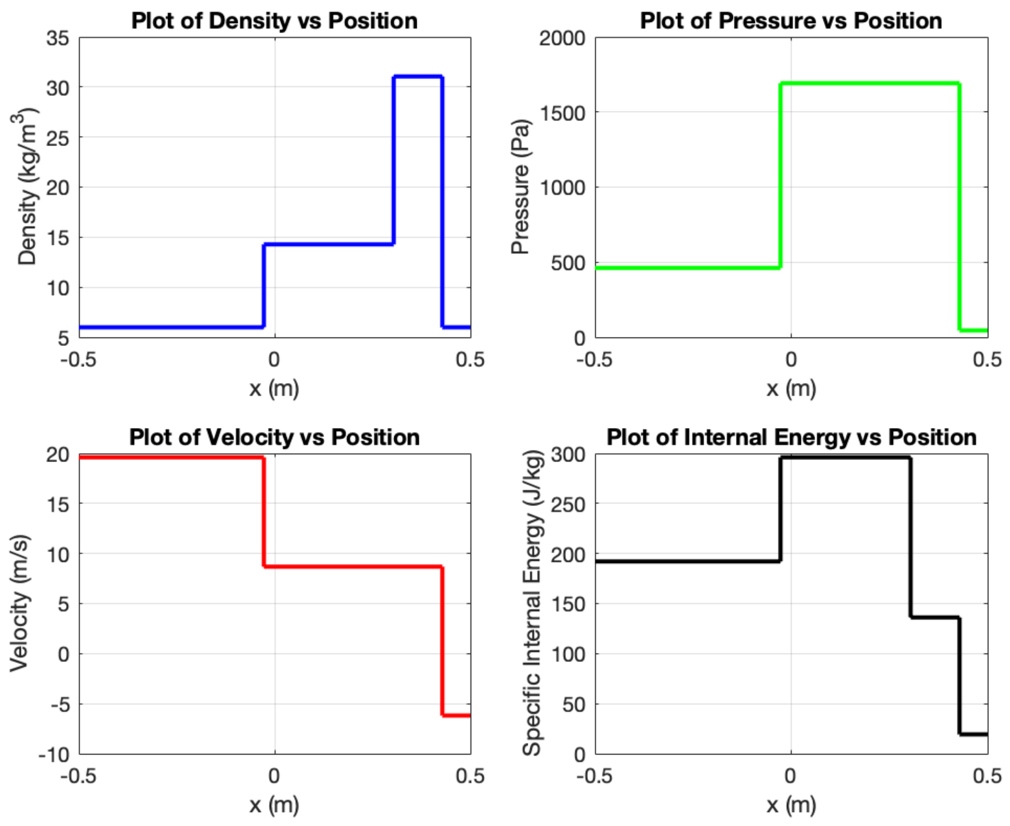

Figure 12: Plots Simulated in Matlab at Time t=0.035 units.

2.4 Numerical Solutions

2.4.1 Lax Freidrichs Scheme

The Lax-Friedrichs scheme is a numerical method used to approximate solutions to hyperbolic partial differential equations. It's a type of finite difference scheme that helps solve problems like the advection equation or traffic flow equations. The scheme is given by the following update equation,

$$U_{\mathrm{i}}{}^{\mathrm{n+1}} = \left(\frac{1}{2}\right)(U_{\mathrm{i\text{-}1}}{}^{\mathrm{n}} + U_{\mathrm{i+1}}{}^{\mathrm{n}}) - \left(\frac{\lambda}{2}\right)(F_{\mathrm{i\text{-}1}}{}^{\mathrm{n}} + F_{\mathrm{i+1}}{}^{\mathrm{n}})$$

- **$\mathbf{U_i{}^{n+1}}$** - represents the value of the solution at grid point i and time n+1.
- **$\mathbf{U_{i\text{-}1}{}^{n}}$ and $\mathbf{U_{i+1}{}^{n}}$** - are values at neighboring grid points at time n.
- **$\mathbf{F_{i\text{-}1}{}^{n}}$ and $\mathbf{F_{i+1}{}^{n}}$** - are the fluxes at the neighboring grid points at time n.
- λ - a parameter controlling the stability and accuracy of the scheme.

The method discretizes both time and space by dividing the domain into a grid of points. The spatial variable x is dicretized into grid points x_{i}, and time t is divided into time steps Δt.

The scheme combines information from neighboring points to update the solution at each grid point in a time-stepping manner. It involves a numerical flux term that incorporates the difference between neighboring fluxes, scaled by the parameter λ

2.4.2 Lax Wendroff's Scheme

The Lax-Wendroff method is another numerical technique used to approximate solutions to hyperbolic partial differential equations (PDEs), similar to the Lax-Friedrichs method. It's a higher-order accurate method and is particularly useful for solving problems involving advection or wave propagation. The update equation for the Lax-Wendroff scheme is,

$$U_{\mathrm{i}}{}^{\mathrm{n+1}} = U_{\mathrm{i}}{}^{\mathrm{n}} - \left(\frac{\lambda}{2}\right)(U_{\mathrm{i+1}}{}^{\mathrm{n}} - U_{\mathrm{i\text{-}1}}{}^{\mathrm{n}}) + \left(\frac{\lambda^2}{2}\right)(U_{\mathrm{i+1}}{}^{\mathrm{n}} + U_{\mathrm{i\text{-}1}}{}^{\mathrm{n}} - 2U_{\mathrm{i\text{-}1}}{}^{\mathrm{n}})$$

- $\mathbf{U_i^{n+1}}$ - represents the value of the solution at grid point i and time n+1.
- $\mathbf{U_{i\text{-}1}^{n}}$ **and** $\mathbf{U_{i+1}^{n}}$ - are values at neighboring grid points at time n.
- λ - a parameter defined as $\frac{c\Delta t}{\Delta x}$ where c is the speed of the wave, Δt is the time step and Δx is a spatial step.

The method discretizes both time and space by dividing the domain into a grid of points. The spatial variable x is dicretized into grid points x_{i}, and time t is divided into time steps Δt.

The Lax-Wendroff method improves accuracy by introducing a second-order correction term that accounts for the curvature of the solution. This correction term helps reduce numerical diffusion present in some first-order methods like Lax-Friedrichs.

2.4.3 Godunov's Scheme

Godunov's Method is a numerical technique used for solving hyperbolic partial differential equations (PDEs), especially in the context of conservation laws. It's a higher-order accurate method, particularly well-suited for problems involving discontinuities or shock waves. Godunov's Method is often applied to problems described by conservation laws, such as the one-dimensional scalar conservation law,

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$$

Similar to other finite difference methods, Godunov's Method discretizes both time and space, dividing the domain into a grid of points. The key feature of Godunov's Method is the solution of Riemann problems at each grid cell interface. A Riemann problem involves the behavior of a discontinuity between two states and determines the flux at the cell interface.

The solution at each grid cell interface is updated using the flux obtained from the Riemann solver. The update equation can be represented as,

$$U_{\mathrm{i}}^{\mathrm{n+1}} = U_{\mathrm{i}}^{\mathrm{n}} - \left(\frac{\Delta t}{\Delta x}\right)(F_{\mathrm{i}+\frac{1}{2}} - F_{\mathrm{i}\text{-}\frac{1}{2}})$$

- $\mathbf{U_i^{n+1}}$ - represents the value of the solution at grid point i and time n+1.
- $\mathbf{F_{i\text{-}\frac{1}{2}}}$ **and** $\mathbf{F_{i+\frac{1}{2}}}$ - represent the fluxes at the cell interfaces.

For a more detailed view of these schemes, refer chapter 4,5 and 6 of the book [1]

2.5 Numerical Solutions in Matlab

2.5.1 Linear Advection

We apply the Numerical schemes to solve the Linear Advection equation,

$$u_{\mathrm{t}} + f(u)_{\mathrm{x}} = 0, \quad f(u) = au, \quad a = constant$$

In the computations we take a = 1.0, $\alpha = 1$, $\beta = 8$ and a CFL coefficient $C_{\mathrm{cfl}} = 0.8$; the initial profile u(x, 0) is evaluated in the interval $-1 \leq$ x ≤ 1. The Initial Data Consists of a square wave,

$$u(x,0) = \begin{cases} 0 & \text{if } x \leq 0.3, \\ 1 & \text{if } 0.3 \leq x \leq 0.7, \\ 0 & \text{if } x \geq 0.7 \end{cases}$$

The code provides a graphic that displays the waves for different numerical schemes evolving as time progresses.

```
clc;
clear;
```

```
% Parameters
num_points = 100;
CFL = 0.8; % Courant number
a = 1;
end_time = 100;

% Define the new initial condition
initial_condition = @(x) 0 * (x <= 0.3) + (x > 0.3 & x <= 0.7) + 0 * (x >=
   0.7);
x_values = linspace(0, 1, num_points + 1);
h = 1 / num_points;
k = CFL * h / a;

% Initialize variables and plot initial conditions
time = 0;
U = zeros(2, num_points + 1);
U = [initial_condition(x_values); initial_condition(x_values)];
U_temp = U;

figure;
plot(x_values, initial_condition(x_values), 'k--'); % Initial condition plot
hold on
theoretical_plot = plot([0, a * time, a * time, 1], [initial_condition(0),
   initial_condition(0), initial_condition(1), initial_condition(1)], 'k-');
LW_plot = plot(x_values, U(1, :), 'bo');
LF_plot = plot(x_values, U(2, :), 'r.');
hold off
xlim([0 1]);
ylim([-0.1 1.1]);
legend('init', 'theo', 'LW', 'LF');
title_text = title(sprintf('t = %0.3f', time));
xlabel('x');
ylabel('u');

% Time evolution loop
while (time + k) < end_time
   for j = 2:num_points
      % Lax-Wendroff scheme
      U_temp(1, j) = U(1, j) - 0.5 * k / h * a * (U(1, j + 1) - U(1, j -
         1)) + 0.5 * (k / h * a)^2 * (U(1, j + 1) - 2 * U(1, j) + U(1, j -
         1));
      % Lax-Friedrichs scheme
      U_temp(2, j) = 0.5 * (U(2, j - 1) + U(2, j + 1)) - 0.5 * k / h * a *
         (U(2, j + 1) - U(2, j - 1));
```

```
    end

    % Apply periodic boundary conditions
    U_temp(:, 1) = U_temp(:, num_points);
    U_temp(:, num_points + 1) = U_temp(:, 2);

    % Update variables for the next iteration
    U = U_temp;
    time = time + k;
    set(theoretical_plot, 'XData', [0, a * time, a * time, 1]);
    set(LW_plot, 'YData', U(1, :));
    set(LF_plot, 'YData', U(2, :));
    set(title_text, 'String', sprintf('t = %0.3f', time));
    drawnow;
end
```

The plots obtain from the Matlab simulations are as follows. The blue dotted line corresponds to the Lax Wendroffs scheme, the red dotted line corresponds to the Lax Friedrichs scheme and the solid black dotted line corresponds to the initial state.

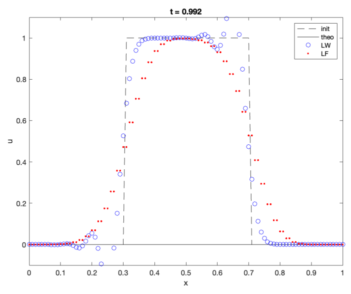

Figure 13: State at Time = 1 units

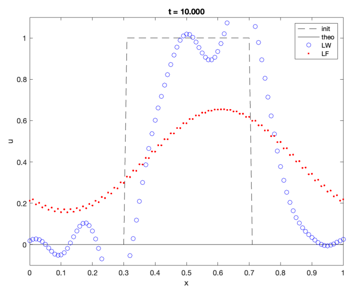

Figure 14: State at Time = 10 units

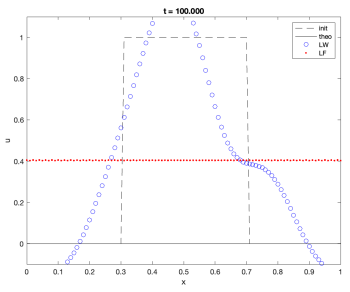

Figure 15: State at Time = 100 units

2.5.2 Euler Equations

In this section, we basically try to implement the Numerical schemes on the various Tests, which are essentially Reimann Problems we saw in section 2.3. Given below is the code which was used in matlab to implement the Numerical methods on Test 1. The provided code conducts a 1D shock tube simulation utilizing the Lax-Friedrichs scheme with adjustable time stepping and compares it with Godunov's method for solving the linear advection equation. The simulation initializes parameters such as the number of grid points, gamma value, physical end time, and CFL number. It sets up the grid and initializes the initial conditions based on a shock tube problem, configuring density, pressure, and velocity distributions. The Lax-Friedrichs method iteratively computes the solution by updating the density, velocity, and energy using fluxes and finite differencing while employing an adjustable time step based on the Courant-Friedrichs-Lewy (CFL) condition. Additionally, the code implements Godunov's method for comparison, initializing conditions and solving the shock tube problem. Finally, it visualizes and compares the density, pressure, and velocity profiles obtained from both methods against the exact solution for the linear advection equation.

```
%% 1D Shock Tube using Lax-Friedrich and Adjustable Time Stepping
close all;
clear;
clc;
%% Initialization of Parameters
N       = 100 ;            % Number of grid points
gamma   = 1.4 ;
endTime = 0.2 ;            % Physical End Time
CFL     = 0.5 ;
%% Grid Initialization
x       = linspace(0,1,N+1) ;
xc      = 0.5 * ( x(1:N) + x(2:N+1) ) ;
xc(1)   = 0 ;              % Xmin
xc(N)   = 1 ;              % Xmax
time    = 0 ;
%% Initial Conidtions
denistyR   = 0.125 ;     densityL   = 1.0 ;
```

```
pressureR  = 0.1   ;      pressureL  = 1.0 ;

rho     = zeros(N,1) ;
p       = zeros(N,1) ;
u       = zeros(N,1) ;

for i = 1:N
    if i<=N/2
        rho(i) = densityL ;
        p(i)   = pressureL ;
    else
        rho(i) = denistyR ;
        p(i)   = pressureR ;
    end
end
e   = p/(gamma-1) + 0.5*rho.*u.*u ;
%%
new_rho = rho ;
new_u   = u   ;
new_e   = e   ;

while time <= endTime

    for i=2:N-1
        p(i)    = (gamma-1)*(e(i) - 0.5*rho(i)*u(i)*u(i)) ;
    end
    a       = sqrt(gamma*p./rho) ;
    lamda   = max(a) ;
    max_vel = max(u) ;

    dt      = CFL/N/(max_vel+lamda) ; % adjustable Time Step
    time    = time + dt ;

    for i=2:N-1
        dx          = xc(i) - xc(i-1) ;

        mom_R       = rho(i+1)*u(i+1) ;   rho_R = rho(i+1) ;    u_R = u(i+1) ;
              p_R = p(i+1) ;
        mom_P       = rho(i)*u(i)    ;   rho_P = rho(i) ;      u_P = u(i)   ;
               p_P = p(i)   ;
        mom_L       = rho(i-1)*u(i-1) ;    rho_L = rho(i-1) ;    u_L = u(i-1)
            ;    p_L = p(i-1) ;

        vel_flux_R = rho_R*u_R*u_R +p_R ; e_R = e(i+1) ;
        vel_flux_P = rho_P*u_P*u_P +p_P ; e_P = e(i)   ;
        vel_flux_L = rho_L*u_L*u_L +p_L ; e_L = e(i-1) ;

        energy_flux_R  = u_R * ( e_R + p_R ) ;
        energy_flux_P  = u_P * ( e_P + p_P ) ;
        energy_flux_L  = u_L * ( e_L + p_L ) ;

        rho_fluxR  = 0.5*( mom_P + mom_R ) -0.5*lamda*( rho_R - rho_P ) ;
        rho_fluxL  = 0.5*( mom_P + mom_L ) -0.5*lamda*( rho_P - rho_L ) ;

        vel_fluxR  = 0.5*( vel_flux_P + vel_flux_R ) -0.5*lamda*( mom_R -
           mom_P ) ;
        vel_fluxL  = 0.5*( vel_flux_P + vel_flux_L ) -0.5*lamda*( mom_P -
           mom_L ) ;

        energy_fluxR   = 0.5*( energy_flux_P + energy_flux_R ) -0.5*lamda*(
           e_R - e_P );
        energy_fluxL   = 0.5*( energy_flux_P + energy_flux_L ) -0.5*lamda*(
           e_P - e_L ) ;

        new_rho(i) = rho_P - dt/dx * ( rho_fluxR - rho_fluxL ) ;
        vel_flux   = mom_P - dt/dx * ( vel_fluxR - vel_fluxL ) ;

        new_u(i)   = vel_flux/new_rho(i) ;
        new_e(i)   = e_P - dt/dx * ( energy_fluxR - energy_fluxL ) ;

    end

    rho     = new_rho ;
    u       = new_u ;
    e       = new_e ;

end

%pressure = dlmread('pressure.dat') ;
%density = dlmread('density.dat') ;
%velocity = dlmread('velocity.dat') ;



%Gudunovs stuff
%Gadunov scheme
L = 1;
dx_gud = 0.001;
x_gud = (0:dx_gud:L);
N_gud = (L/dx_gud)+1;
```

```
T_gud = 0.2;
dt_gud = 0.0001;
n_gud = (T_gud/dt_gud);
C = dt_gud/dx_gud;
gamma = 1.4;

%Initial conditions
for i = 1:N_gud
    if x_gud(i) <= 0.5
        rho_gud(i) = 1;
        u_gud(i) = 0;
        p_gud(i) = 2.5*(gamma - 1);
    else
        rho_gud(i) = 0.125;
        u_gud(i) = 0;
        p_gud(i) = 0.25*(gamma - 1);
    end
end
ET(:) = ((1/2).*rho_gud(:).*(u_gud(:).^2)) + (p_gud(:)/(gamma - 1));



for i = 1:n_gud

    %1) Construct Q and E vectors

    Q_gud(1,:) = rho_gud(:);
    Q_gud(2,:) = rho_gud(:).*u_gud(:);
    Q_gud(3,:) = ET(:);

    E_gud(1,:) = rho_gud(:).*u_gud(:);
    E_gud(2,:) = E_gud(1,:).*u_gud(1,:) + p_gud(1,:);
    E_gud(3,:) = ET(:).*u_gud(:) + p_gud(:).*u_gud(:);

    %2) Calculate Q(n+1) using Gadunov method
    Q1 = Q_gud;
    flux = zeros(3,N_gud);
    for k = 1:N_gud-1
        c = sqrt(gamma*p_gud(k)/rho_gud(k));
        A = [abs(Q_gud(2,k)) abs(Q_gud(2,k)+c) abs(Q_gud(2,k)-c)];
        al = max(A);
        flux(:,k) = 0.5*(E_gud(:,k) + E_gud(:,k+1)) - 0.5*(al)*(Q1(:,k+1) -
           Q1(:,k));
    end
    for k = 2:N_gud-1
```

```
        Q_gud(:,k) = Q1(:,k) - C*(flux(:,k) - flux(:,k-1));
    end

    %3) Update solution (rho,u,p)
    rho_gud(:) = Q_gud(1,:);
    u_gud(1,:) = Q_gud(2,:)./rho_gud(1,:);
    ET(:) = Q_gud(3,:);
    p_gud(1,:) = (ET(1,:) - ((1/2).*rho_gud(1,:).*(u_gud(1,:).^2)))*(gamma -
        1);

end







time = 0.2;
data = Test_1(time);


figure(1)
hold on
%plot(density(:,1),density(:,2),'r--','MarkerSize',12,'MarkerIndices',1:10:length(velocity),
plot(data.x,data.rho,'b','LineWidth',2);
plot(xc,rho,'k','MarkerSize',12,'MarkerIndices',1:10:141,'LineWidth',2);
plot(x_gud, rho_gud, 'g', 'LineWidth', 2);
xlabel(' x ','FontSize',18,'FontWeight','bold');
ylabel(' Density ','FontSize',18,'FontWeight','bold');
legend('Exact','Lax
   Friedrich','Gudunov','Location','northeast','FontSize',16,'FontWeight','bold');
%print(gcf,'Density.jpg','-dpng','-r300');

figure(2)
hold on
%plot(pressure(:,1),pressure(:,2),'r--','MarkerSize',12,'MarkerIndices',1:10:length(velocity
plot(data.x,data.P,'b','LineWidth',2);
plot(xc,p,'k','MarkerSize',12,'MarkerIndices',1:10:141,'LineWidth',2);
plot(x_gud, p_gud, 'g', 'LineWidth', 2);
xlabel(' x ','FontSize',18,'FontWeight','bold');
ylabel(' Pressure ','FontSize',18,'FontWeight','bold');
legend('Exact','Lax
   Friedrich','Gudunov','Location','northeast','FontSize',16,'FontWeight','bold');
%print(gcf,'Pressure.jpg','-dpng','-r300');
```

```
figure(3)
hold on
%plot(velocity(:,1),velocity(:,2),'r--','MarkerSize',12,'MarkerIndices',1:10:length(velocity
plot(data.x,data.u,'b','LineWidth',2);
plot(xc,u,'k','MarkerSize',12,'MarkerIndices',1:10:141,'LineWidth',2);
plot(x_gud, u_gud, 'g', 'LineWidth', 2);
xlabel(' x ','FontSize',18,'FontWeight','bold');
ylabel(' Velocity ','FontSize',18,'FontWeight','bold');
legend('Exact','Lax
   Friedrich','Gudunov','Location','south','FontSize',16,'FontWeight','bold');
%print(gcf,'Velocity.jpg','-dpng','-r300');
```

The plots obtained after simulation were as follows. We can see that

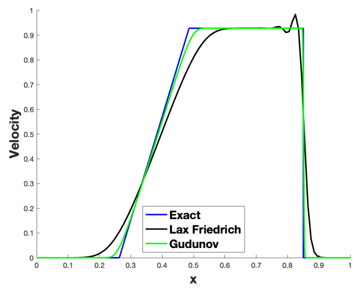

Figure 16: Velocity vs x

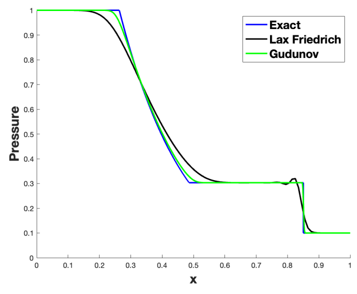

Figure 17: Pressure vs x

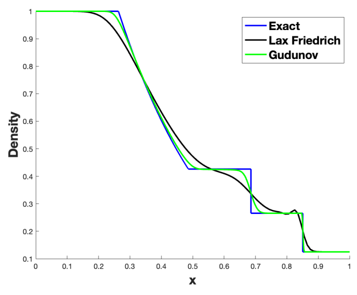

Figure 18: Density vs x

Godunov's Scheme provides a better approximation to the exact solution than Lax Friedrichs.

3 Conclusion

In conclusion, this project has been an illuminating journey into the realm of computational fluid dynamics, delving into the intricacies of Riemann problems, shock waves, and rarefactions. Through rigorous study and practical implementation, I gained a profound understanding of various numerical schemes including Godunov's, Lax-Friedrichs, and Lax-Wendroff methods. Implementing these schemes in MATLAB provided hands-on insights into their behavior, strengths, and limitations in solving the linear advection equation.

The realization of Riemann problems as fundamental in understanding wave behaviors, particularly shocks and rarefactions, showcased the significance of accurate numerical methods in capturing these phenomena. Moreover, contrasting the exact solutions with numerical approximations highlighted the trade-offs between computational efficiency and accuracy inherent in these schemes.

In today's world, where computational simulations drive innovations in numerous fields like climate modeling, aerospace engineering, and beyond, the importance of robust numerical methods in predicting complex wave behaviors cannot be overstated. The skills and knowledge gained from this project resonate deeply with the current demand for precise and efficient simulations in a technologically advancing world.

I extend my heartfelt gratitude to Professor P Minhajul of the Department of Mathematics, BITS Pilani KK Birla Goa Campus, for his invaluable guidance, unwavering support, and the opportunity to work under his mentorship. His expertise and encouragement have been instrumental in shaping this project and fostering my growth in the field of Computational Fluid Dynamics.

4 References

[1] Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Berlin, Heidelberg, 3 edition, 2009.

[2] Sahil Sudesh. Reimann solvers numerical methods. `https://github.com/God-SammY-lord/Reimann_Solvers_Numerical_methods`, 2023.