Computational study to solve the Shock-tube problem with Lax initial conditions for 1D Euler's equation using 1st order, 2nd order, 3rd order & Monotonicity preserving schemes

Name: Chinmay Rajesh Chavan

Course: MEEN 689 – Computational Fluid Dynamics

Final Project

Date: 12-15-2022

UIN: 633002377

UIN Digit	9th digit – **7**	8th digit – **7**	7th digit – **3**	Bonus
0-2	Godunov	Lax-Wendroff	FCT	**ENO**
3-5	Upwind – Flux vector splitting	McCormack	**MUSCL**	
6-9	**Upwind – Flux difference splitting**	**R-K with Artificial Dissipation**	R-K with TVD	

Introduction

To find the solutions to the shock-tube problem we will solve the 1-D Euler's equation for gas dynamics given by,

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}(\vec{Q})}{\partial x} = 0 \ \ldots\ldots\ldots\ldots\ldots Eq.(1)$$

where,

$$\vec{Q} = \begin{Bmatrix} Q1 \\ Q2 \\ Q3 \end{Bmatrix} = \begin{Bmatrix} \rho \\ m \\ E \end{Bmatrix} \ \ldots\ldots\ldots\ldots\ldots Eq.(2)$$

and

$$\vec{E}(\vec{Q}) = \begin{Bmatrix} E1 \\ E2 \\ E3 \end{Bmatrix} = \begin{Bmatrix} m \\ mu + p \\ Eu + pu \end{Bmatrix} \ \ldots\ldots\ldots\ldots\ldots Eq.(3)$$

Here, the values for density(ρ), velocity (u), pressure(p) and internal energy (e) can be found using following equations in terms of $\vec{Q}$:

$$\rho = Q1 \ \ldots\ldots\ldots\ldots\ldots Eq.(4)$$

Momentum is given by, $m = \rho u \xrightarrow{yields} u = \frac{m}{\rho} \Rightarrow u = \frac{Q2}{Q1} \ \ldots\ldots\ldots\ldots\ldots Eq.(5)$

Equation of state is (here, γ=1.4), $E = \frac{1}{2}\rho u^2 + \frac{p}{\gamma - 1} = \frac{1}{2}\rho u^2 + \frac{p}{0.4}$

Rearranging terms we get, $p = 0.4\left(E - \frac{1}{2}\rho u^2\right) \Rightarrow p = 0.4\left(Q3 - \frac{1}{2} \times \frac{Q2^2}{Q1}\right) \ldots Eq.(6)$

Total energy is given by, $E = \rho e + \frac{1}{2}\rho u^2 = \rho\left(e + \frac{1}{2}u^2\right)$

Rearranging terms we get, $e = \frac{E}{\rho} - \frac{1}{2}u^2 \Rightarrow e = \frac{Q3}{Q1} - \frac{1}{2}\left(\frac{Q2}{Q1}\right)^2 \ \ldots\ldots\ldots\ldots\ldots Eq.(7)$

The terms in the dependent flux function ($\vec{E}$) can be given in terms of $\vec{Q}$ with following equations:

$$E1 = m \Rightarrow E1 = Q2 \ \ldots\ldots\ldots\ldots\ldots Eq.(8)$$

$$E2 = mu + p \Rightarrow E2 = \frac{Q2^2}{Q1} + 0.4\left(Q3 - \frac{1}{2} \times \frac{Q2^2}{Q1}\right) \ldots\ldots\ldots\ldots Eq.(9)$$

$$E3 = Eu + pu = u(E + p) \Rightarrow E3 = \frac{Q2}{Q1}\left[Q3 + 0.4\left(Q3 - \frac{1}{2} \times \frac{Q2^2}{Q1}\right)\right] \ldots\ldots\ldots Eq.(10)$$

We have considered the domain from $0 \leq x \leq 1$ and obtained solutions for final time T=0.16 in this study.

We are solving for Lax initial conditions given by,

$$\vec{Q} = \begin{pmatrix} Q1 \\ Q2 \\ Q3 \end{pmatrix} = \begin{pmatrix} \rho \\ m \\ E \end{pmatrix} = \begin{pmatrix} 0.445 \\ 0.311 \\ 8.928 \end{pmatrix} \quad for\ x \in (0, 0.5)$$

$$\vec{Q} = \begin{pmatrix} Q1 \\ Q2 \\ Q3 \end{pmatrix} = \begin{pmatrix} \rho \\ m \\ E \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0 \\ 1.4275 \end{pmatrix} \quad for\ x \in (0.5, 1)$$

In this study, two cases are considered for all the methods as shown in Table 1 for solving the shock-tube problem with lax initial conditions. Case 1 is solved using space discretization with 101 grid points and case 2 is solved using a finer mesh having 1001 grid points.

Method used	Case 1		Case 2	
	Δx	Δt	Δx	Δt
Upwind with Flux difference splitting	0.01	0.0001	0.001	0.00001
Runge-Kutta with Artificial dissipation	0.01	0.001	0.001	0.0001
Godunov with MUSCL scheme	0.01	0.0001	0.001	0.00001
Godunov with ENO scheme	0.01	0.0001	0.001	0.00001

Table 1. *Cases considered for evaluating the 1D shock-tube problem.*

Method of Solution

For better representing the difference equations used in the different methods for this study, we will denote single component of $\vec{Q}$ (i.e., any one of $Q1, Q2, Q3$) generally as u. And we will denote components of corresponding flux function $\vec{E}$ (i.e., either of $E1, E2, E3$) generally as E.

A.) Upwind – Flux difference splitting (1st order) method:

In this method, we calculate fluxes at the cell boundaries which are at $i \pm {}^{1}/_{2}$ for cell i. Flux at cell boundaries is calculated using,

$$\vec{E}^{(n)}_{(i+\frac{1}{2})} = \frac{1}{2}\left(\vec{E}^{R} + \vec{E}^{L}\right) - \frac{1}{2}|A|\left(\vec{Q}^{R} + \vec{Q}^{L}\right) \dots\dots\dots Eq.(11)$$

where,

$$\vec{Q}^{R} = \vec{Q}_{(i+{}^{1}/_{2})^{+}} = \vec{Q}^{(n)}_{(i+1)} \;\; and \;\; \vec{Q}^{L} = \vec{Q}_{(i+{}^{1}/_{2})^{-}} = \vec{Q}^{(n)}_{(i)} \dots\dots\dots Eq.(12)$$

$$\vec{E}^{R} = \vec{E}_{(i+{}^{1}/_{2})^{+}} = \vec{E}^{(n)}_{(i+1)} \;\; and \;\; \vec{E}^{L} = \vec{E}_{(i+{}^{1}/_{2})^{-}} = \vec{E}^{(n)}_{(i)} \dots\dots\dots Eq.(13)$$

$$|A| = \max|\lambda| = \max\left(\vec{Q}^{L}, \vec{Q}^{L} + a, \vec{Q}^{L} - a, \vec{Q}^{R}, \vec{Q}^{R} + a, \vec{Q}^{R} - a\right) \dots\dots\dots Eq.(14)$$

here, $a = \sqrt{\frac{\gamma p}{\rho}}$ $\Rightarrow$ speed of sound.

Value at next time step is calculated by using,

$$\vec{Q}^{(n+1)}_{(i)} = \vec{Q}^{(n)}_{(i)} - \frac{\Delta t}{\Delta x}\left(\vec{E}^{(n)}_{\left(i+\frac{1}{2}\right)} - \vec{E}^{(n)}_{\left(i-\frac{1}{2}\right)}\right) \dots\dots\dots Eq.(15)$$

Figure 1 shows the stencil used in upwind with flux difference splitting method and describes how this method is implemented.

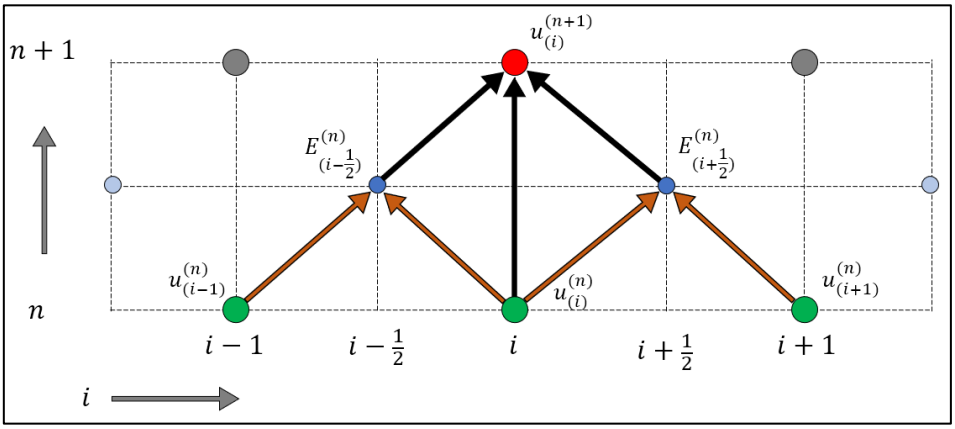

Fig.1. *Sketch of stencil used for Upwind with flux difference splitting scheme.*

B.) Runge-Kutta with Artificial dissipation (2nd order) method:

In this method we will use the modified version of the 2nd order Runge-Kutta method which has 2 steps computed one after the other to give value at next time step. Here, we will use superscript "2" to denote the intermediate step before computing the value at next time step.

The difference equations for 2nd order Runge-Kutta are given by,

$$u_{(i)}^{(2)} = u_{(i)}^{(n)} - \frac{1}{4}\frac{\Delta t}{\Delta x}\left(E_{(i+1)}^{(n)} - E_{(i-1)}^{(n)}\right) \dots\dots\dots\dots Eq.(16)$$

$$u_{(i)}^{(n+1)} = u_{(i)}^{(n)} - \frac{1}{2}\frac{\Delta t}{\Delta x}\left(E_{(i+1)}^{(2)} - E_{(i-1)}^{(2)}\right) \dots\dots\dots\dots Eq.(17)$$

where $E^{(2)}$ is the intermediate flux computed using $u^{(2)}$. This scheme has oscillations. So, we will artificially introduce an explicit 4th order dissipation term to the RHS of Eq. (17) to reduce the oscillations. This gives us one final step, which is,

$$u_{(i)}^{(n+1)} = u_{(i)}^{(n+1)} - \varepsilon\left(u_{(i-2)}^{(n)} - 4u_{(i-1)}^{(n)} + 6u_{(i)}^{(n)} - 4u_{(i+1)}^{(n)} + u_{(i+2)}^{(n)}\right) \dots\dots\dots\dots Eq.(18)$$

Here, Ɛ is a positive dissipation constant of very small magnitude. Figure 2 shows the stencil used for Runge-Kutta with artificial dissipation method and describes how this method is implemented. Here, the dotted arrow lines are showing the points that are included only in the dissipation term when advancing solution in time.

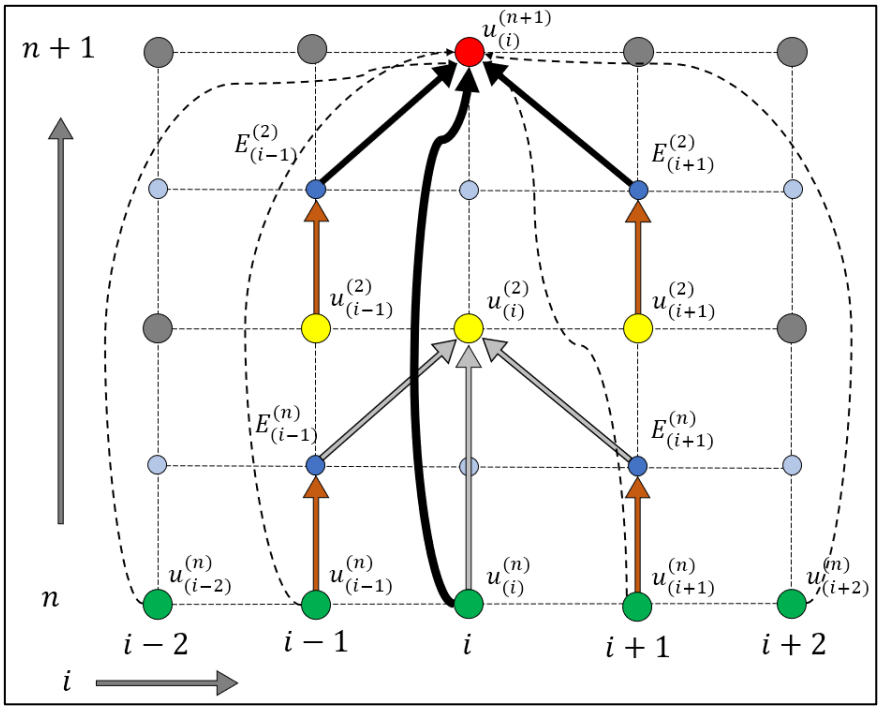

Fig.2. *Sketch of stencil used for Runge-Kutta with artificial dissipation scheme.*

C.) Godunov with MUSCL scheme (monotonicity preserving) method:

This method is all about finding a piecewise polynomial which will be 2nd order accurate in the range $x \in \left(x_{(i-1/2)},\ x_{(i+1/2)}\right)$ which is given as,

$$\tilde{u}(x) = u(x_{(i)}) + u'(x_{(i)}).(x - x_{(i)}) + O(\Delta x^2) \dots\dots\dots\dots Eq.(19)$$

Then to find the 1st order accurate derivative $[u'(x_{(i)})]$, in order to maintain the 2nd order accuracy of the polynomial in the defined range of x for cell i, we define a slope limiter function $u'(x_{(i)}) = S_{(i)}$ which is given as follows,

$$S_{(i)} = \min mod\left\{\frac{\bar{u}_{(i+1)}^{(n)} - \bar{u}_{(i)}^{(n)}}{\Delta x}, \frac{\bar{u}_{(i)}^{(n)} - \bar{u}_{(i-1)}^{(n)}}{\Delta x}\right\} \dots\dots\dots\dots Eq.(20)$$

where, $\bar{u}$ is the cell average value.

Then we can write the approximate value at cell boundaries using Eq. (19) & (20) as follows,

$$u\left(i+{}^{1}/_{2}\right)^{-} = \bar{u}(i) + S_{(i)}\left({}^{\Delta x}/_{2}\right) \ldots\ldots\ldots\ldots Eq.(21)$$

$$u\left(i-{}^{1}/_{2}\right)^{+} = \bar{u}(i) - S_{(i)}\left({}^{\Delta x}/_{2}\right) \ldots\ldots\ldots\ldots Eq.(22)$$

Using Eq. (21) & (22), we can compute boundary values for all neighboring cells in the grid. Flux at cell boundaries using the Lax-Friedrich flux function is calculated using,

$$F_{(i+{}^{1}/_{2})} = \frac{1}{2}(E(u_R) + E(u_L)) - \frac{1}{2}\alpha(u_R - u_L) \ldots\ldots\ldots Eq.(23)$$

where,

$$u_L = u\left(i+{}^{1}/_{2}\right)^{-} \Rightarrow \text{computed using Eq. (21) for the } i^{th} \text{ cell}$$

$$\text{and} \quad u_R = u\left(i+{}^{1}/_{2}\right)^{+} \Rightarrow \text{computed using Eq. (22) for the } (i+1)^{th} \text{ cell}$$

$$\text{and} \quad \alpha = \max|\lambda| = \max(u_L, u_L + a, u_L - a, u_R, u_R + a, u_R - a)$$

here, $a = \sqrt{\frac{\gamma p}{\rho}} \Rightarrow$ speed of sound.

Value at next time step is calculated by using,

$$u_{(i)}^{(n+1)} = u_{(i)}^{(n)} - \frac{\Delta t}{\Delta x}\left(F_{\left(i+\frac{1}{2}\right)}^{(n)} - F_{\left(i-\frac{1}{2}\right)}^{(n)}\right) \ldots\ldots\ldots\ldots Eq.(24)$$

Figure 3 shows the stencil used for Godunov with MUSCL scheme and describes how this method is implemented.

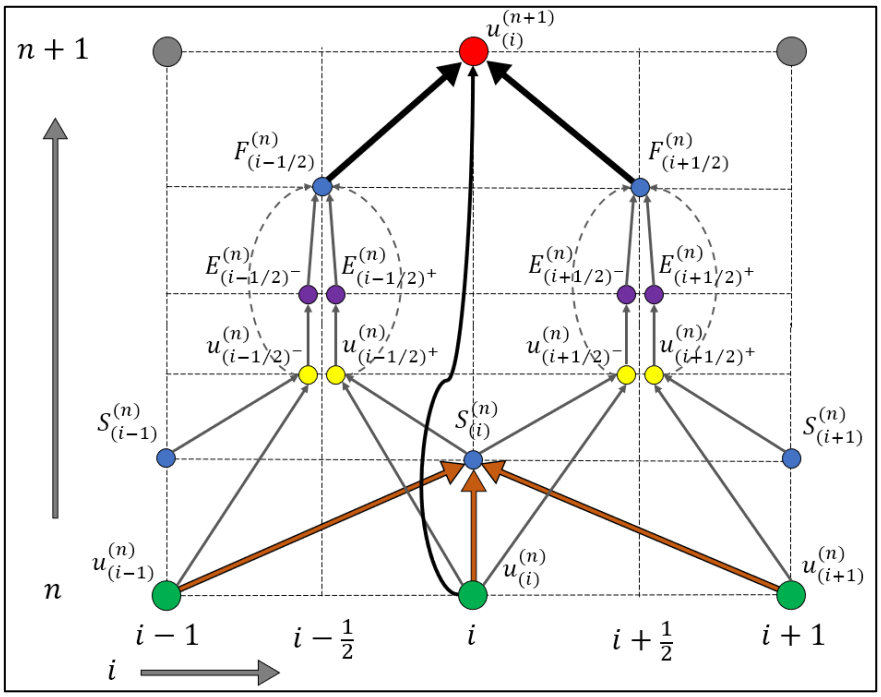

Fig.3. *Sketch of stencil used for Godunov with MUSCL scheme.*

D.) Godunov with ENO scheme (3rd order) method:

In this method we find a piecewise polynomial which will be 3rd order accurate in the range $x \in \left(x_{(i-{}^{1}/_{2})},\ x_{(i+{}^{1}/_{2})}\right)$. Here we will use a 3-point stencil with $(i-r)$ being the left-most

point in the stencil, as shown in Fig. (4), and $r = 0, 1, 2$ to get 3rd order accurate results. To select starting point to get smoothest stencil, we use Newton divided difference and select value of r such that we get lowest divided difference. (Detailed steps discussed in appendix) Using the selected stencil, we can write the approximate value at cell boundaries using,

$$u\left(i + {}^{1}/_{2}\right)^{-} = \sum_{k=0}^{2} C_{rk}\bar{u}_{(i-r+k)} \dots\dots\dots\dots Eq.(25)$$

$$u\left(i - {}^{1}/_{2}\right)^{+} = \sum_{k=0}^{2} C_{(r-1)k}\bar{u}_{(i-r+k)} \dots\dots\dots\dots Eq.(26)$$

where, $\bar{u}$ is the cell average value and coefficient C_{rk} is computed as per formula given in appendix. Using Eq. (25) & (26), we can compute boundary values for all neighboring cells in the grid. Flux at cell boundaries using the Lax-Friedrich flux function is calculated using,

$$F_{(i+1/2)} = \frac{1}{2}(E(u_R) + E(u_L)) - \frac{1}{2}\alpha(u_R - u_L) \dots\dots\dots Eq.(27)$$

where,

$$u_L = u\left(i + {}^{1}/_{2}\right)^{-} \Rightarrow \text{computed using Eq. (25) for the } i^{th} \text{ cell}$$

and $$u_R = u\left(i + {}^{1}/_{2}\right)^{+} \Rightarrow \text{computed using Eq. (26) for the } (i+1)^{th} \text{ cell}$$

and $\alpha = \max|\lambda| = \max(u_L, u_L + a, u_L - a, u_R, u_R + a, u_R - a)$ … here, $a = \sqrt{\frac{\gamma p}{\rho}}$.

Value at next time step is calculated by using,

$$u_{(i)}^{(n+1)} = u_{(i)}^{(n)} - \frac{\Delta t}{\Delta x}\left(F_{\left(i+\frac{1}{2}\right)}^{(n)} - F_{\left(i-\frac{1}{2}\right)}^{(n)}\right) \dots\dots\dots Eq.(28)$$

Figure 4 shows the stencil used for Godunov with ENO scheme and describes how this method is implemented. For representation, we have considered the case when $r = 1$, which means a central stencil. Also, here we have depicted implementation for cell j, for which we have taken $j = i - r + 1$.

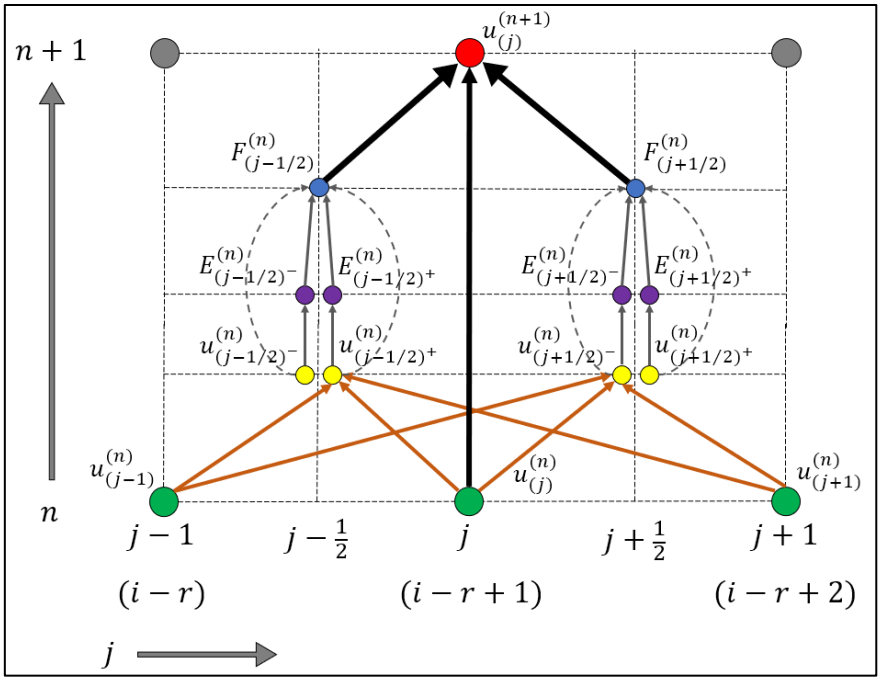

Fig.4. *Sketch of stencil used for Godunov with ENO scheme.*

Discussion of Results

A.) Solutions using **1st order Upwind – Flux difference splitting** method:

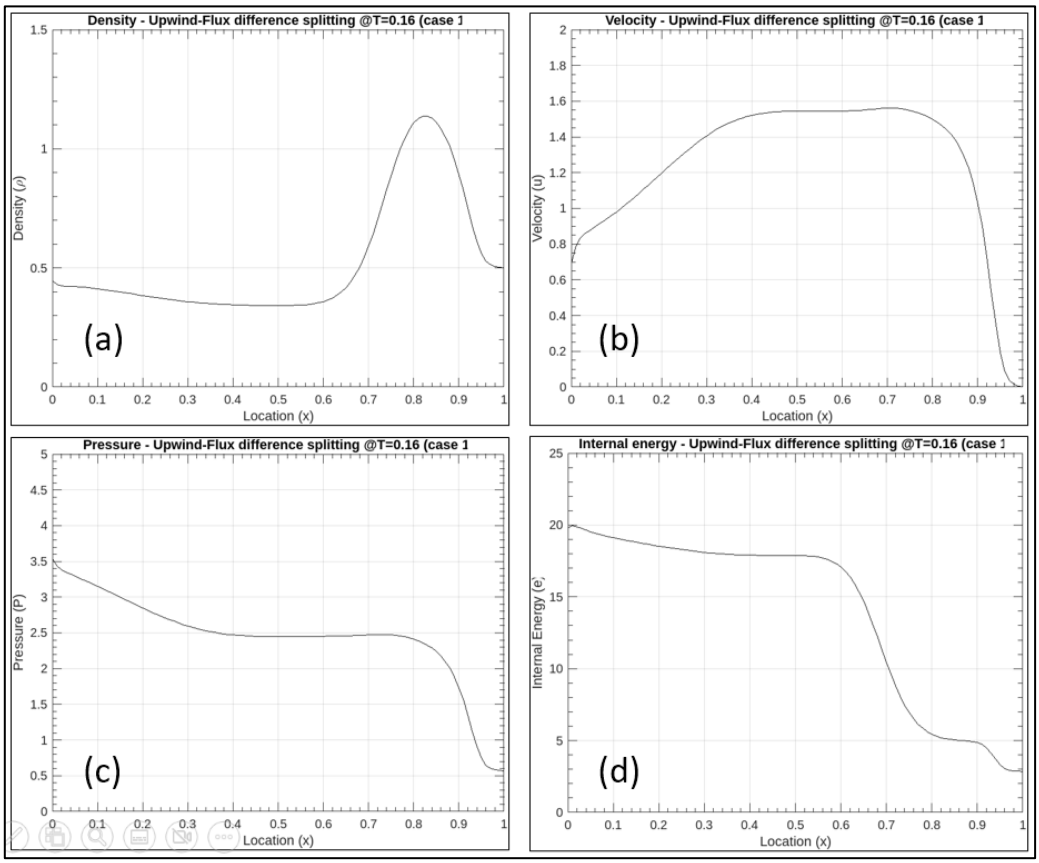

Fig.5. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Upwind – Flux difference splitting scheme (case 1 – Δx=0.01).*

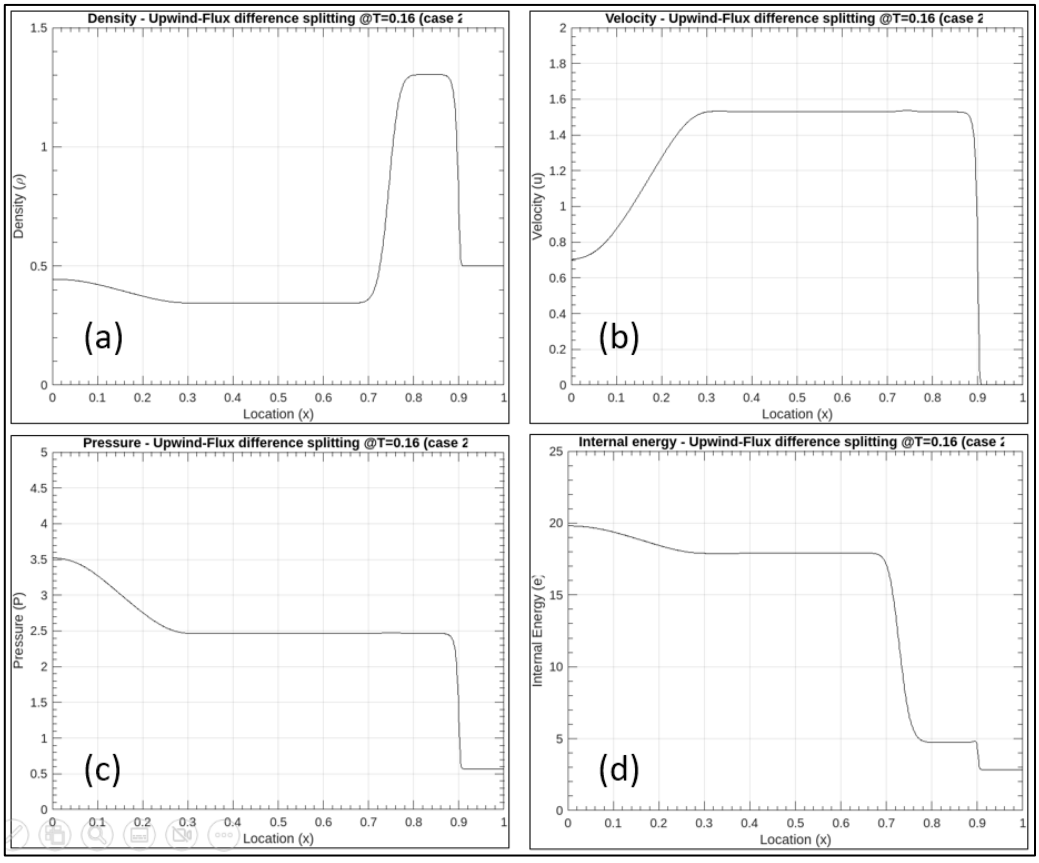

Fig.6. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Upwind – Flux difference splitting scheme (case 2 – Δx=0.001).*

B.) Solutions using **2nd order Runge-Kutta with Artificial dissipation** method:

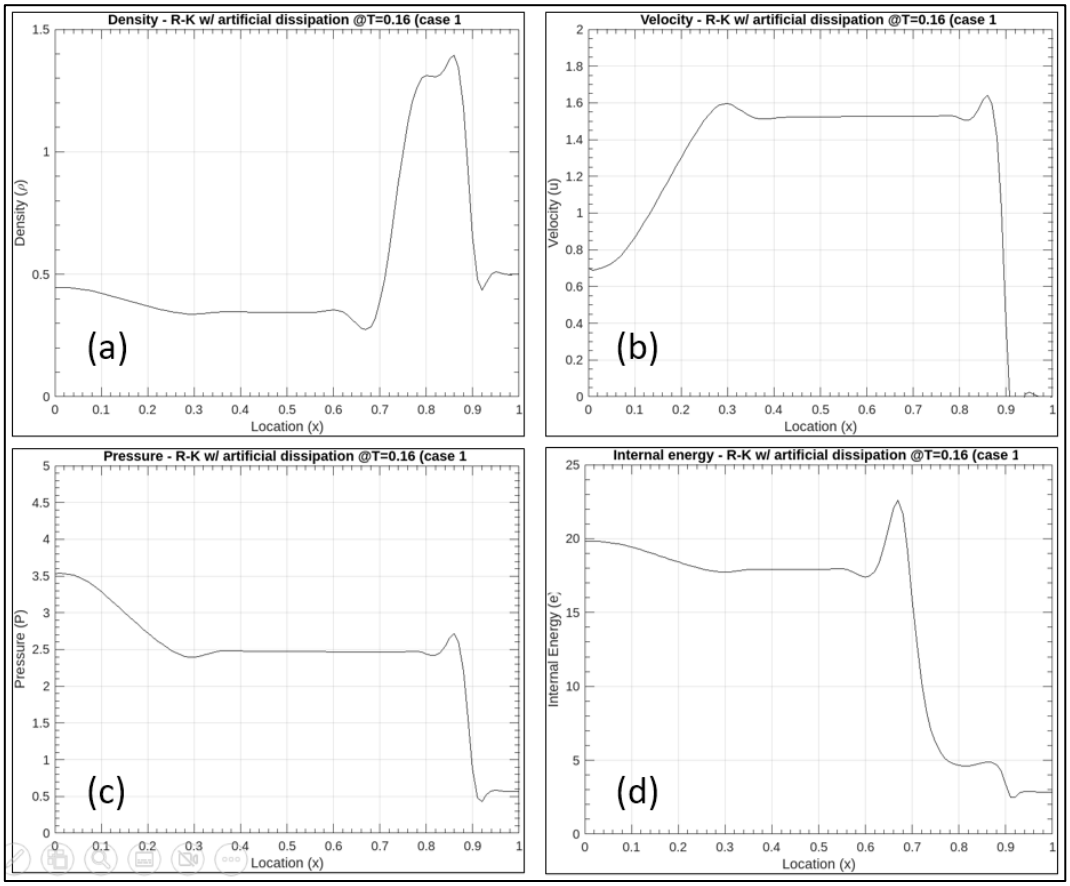

Fig.7. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Runge-Kutta with Artificial dissipation scheme (case 1 – Δx=0.01).*

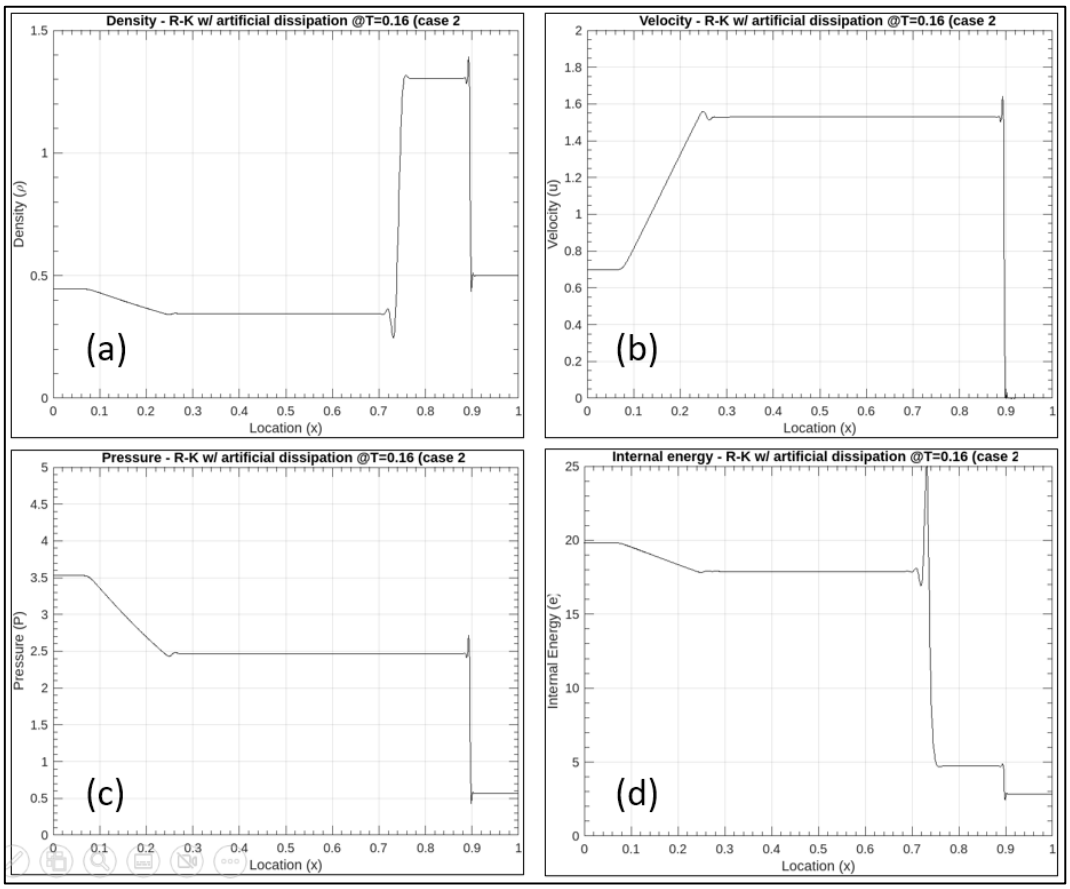

Fig.8. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Runge-Kutta with Artificial dissipation scheme (case 2 – Δx=0.001).*

C.) Solutions using **Godunov with MUSCL slope limiter** method:

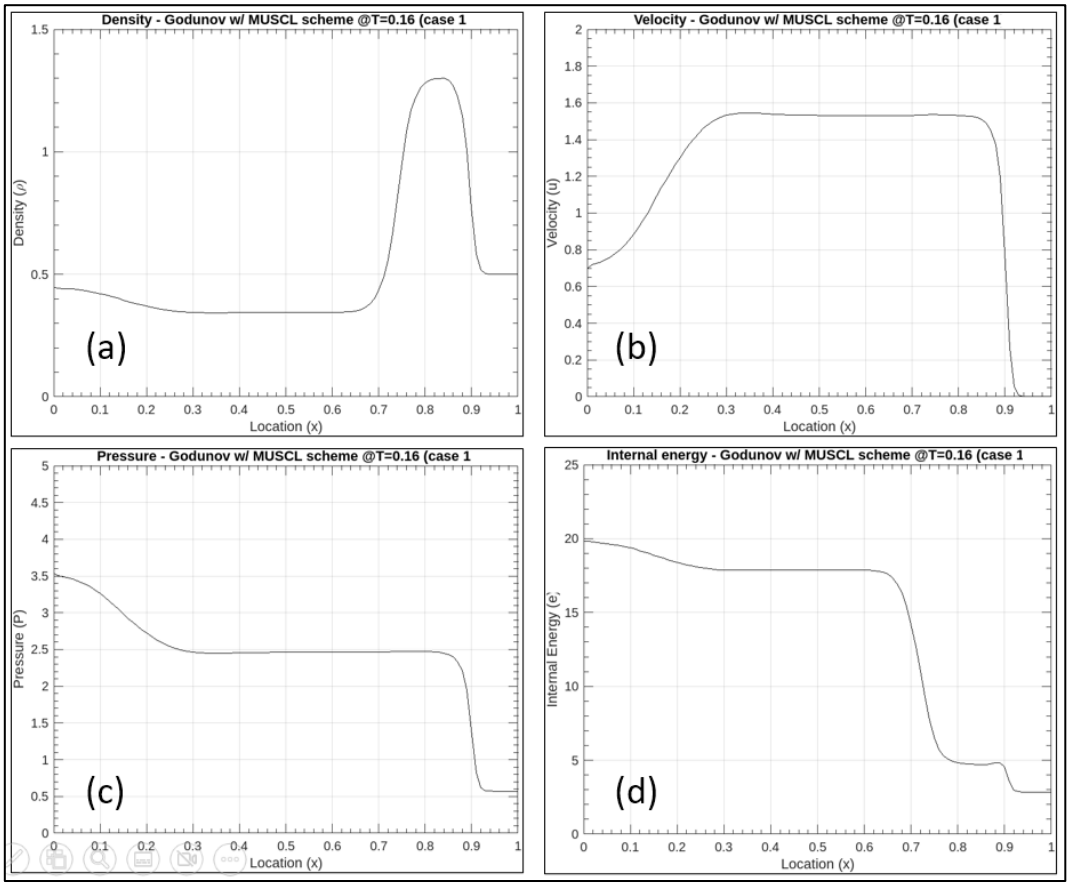

Fig.9. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Godunov with MUSCL slope limiter scheme (case 1 – Δx=0.01).*

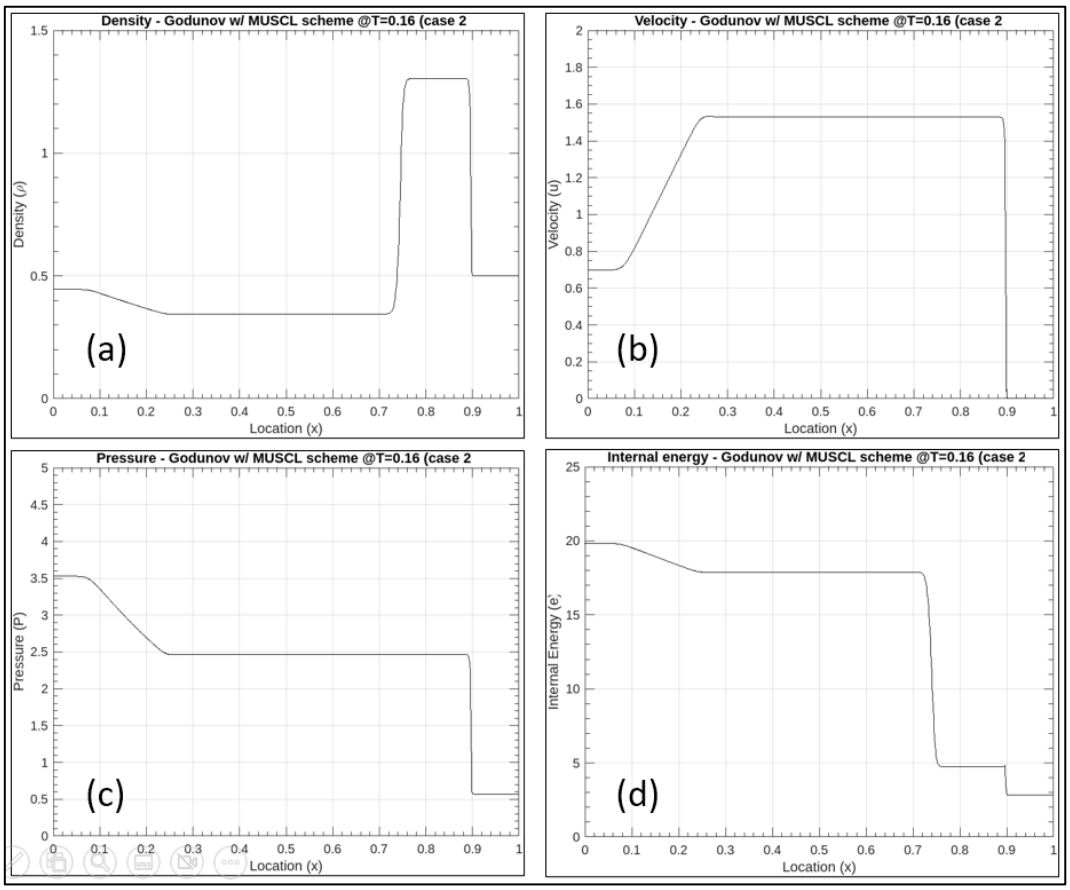

Fig.10. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Godunov with MUSCL slope limiter scheme (case 2 – Δx=0.001).*

D.) Solutions using **Godunov with ENO** method:

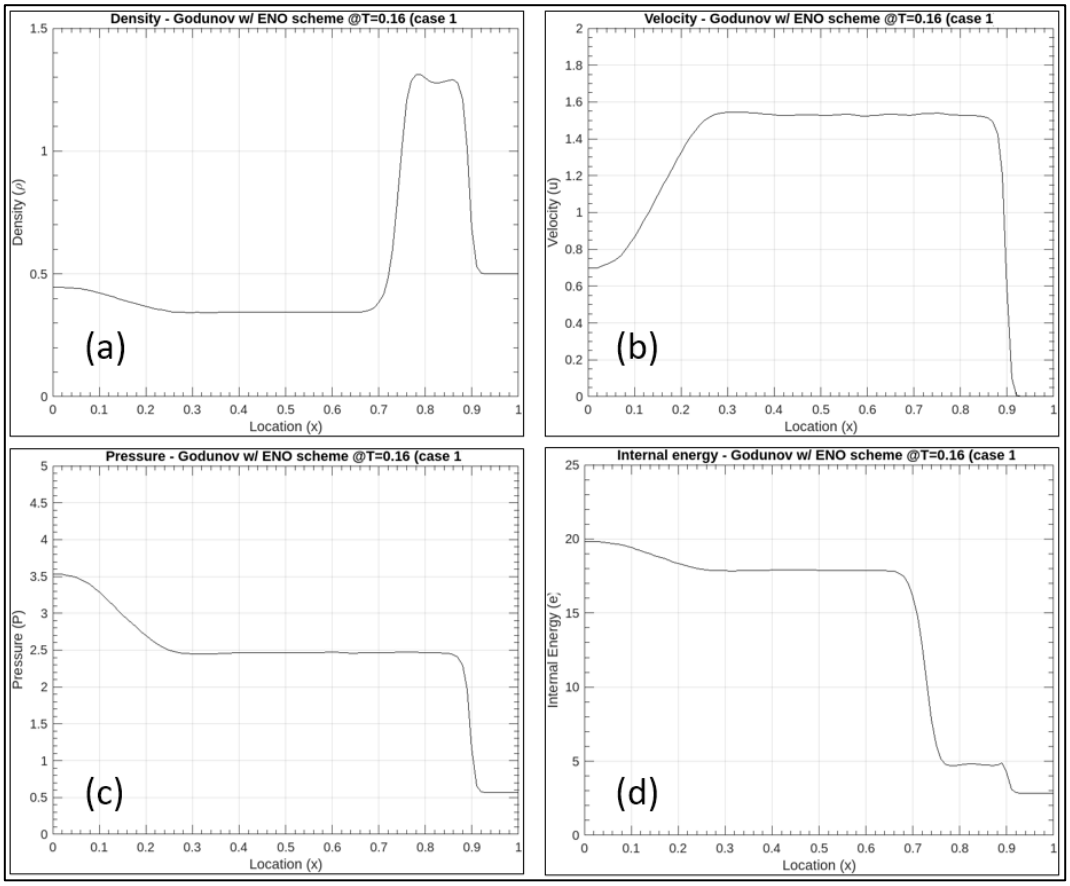

Fig.11. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Godunov with ENO scheme (case 1 – Δx=0.01).*

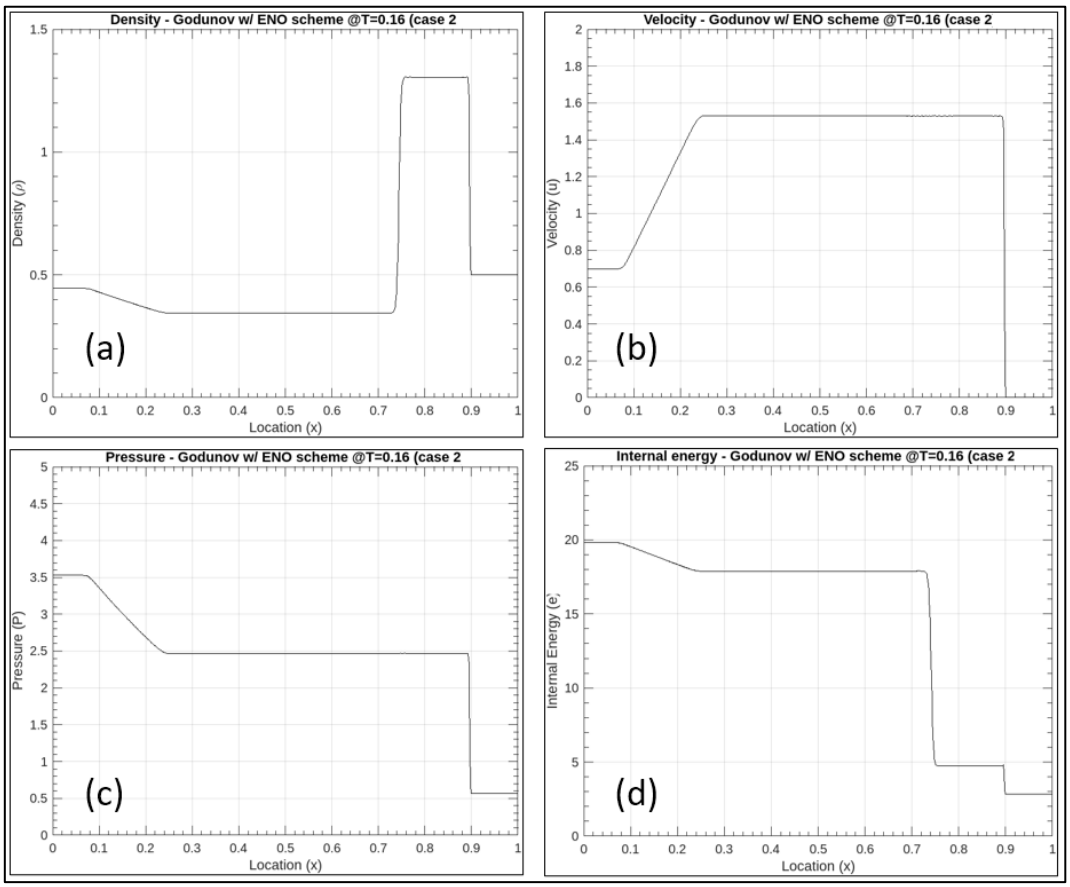

Fig.12. *Solution plots at T=0.16 for (a) Density, (b) Velocity, (c) Pressure and (d) Internal energy using Godunov with ENO scheme (case 2 – Δx=0.001).*

E.) Comparing solutions with benchmark solution:

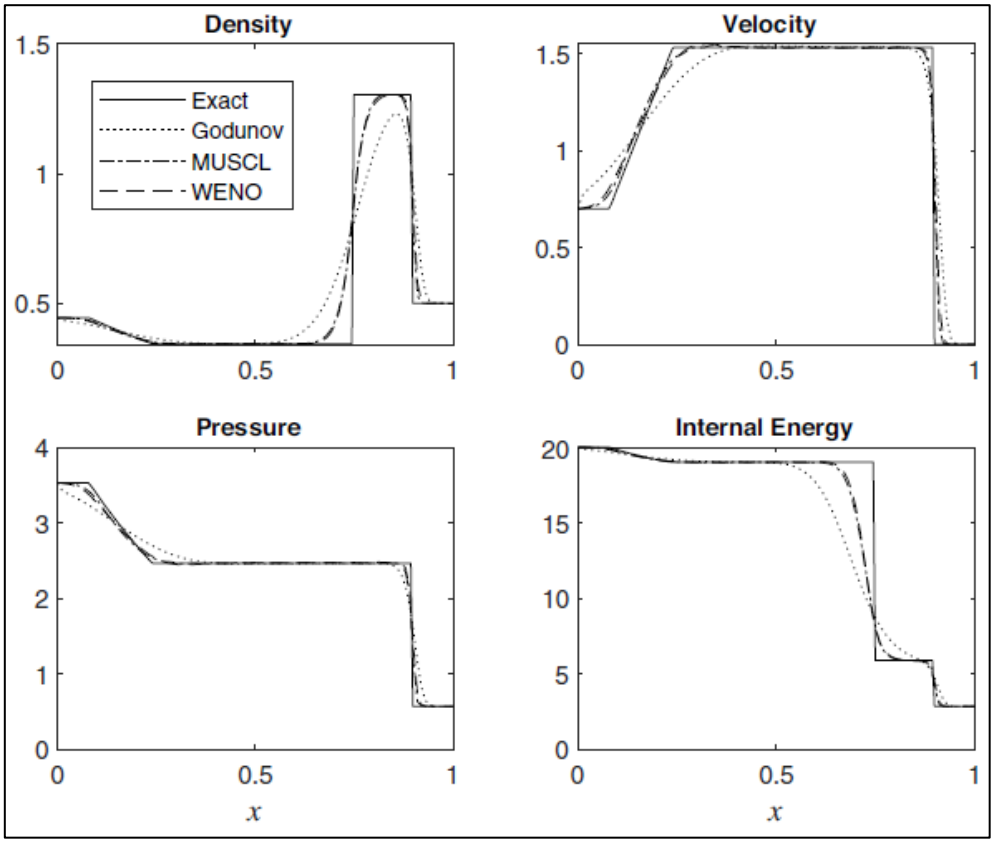

Fig.13. *Benchmark solution at T=0.16 for the shock-tube problem with lax initial conditions.*

For case 1 (Δx=0.01), comparing solutions by all methods from Figs. (5), (7), (9), (11) with benchmark solution from Fig. (13), we can see good correlation between the solutions obtained in this study with the benchmark solution. From Fig. (5), we observe that 1st order upwind with flux difference splitting has solution similar to the Godunov solution in benchmark Fig. (13). In Fig. (7), 2nd order Runge-Kutta with artificial dissipation follows the benchmark closely but with some oscillations which is expected, although the oscillations are damped because of the added 4th order explicit artificial dissipation term. Figure (9) shows that Godunov with MUSCL scheme solution is similar to MUSCL solution in benchmark. Godunov with ENO scheme solutions from Fig. (11) are overall comparable with WENO solution in benchmark from Fig. (13), but there are some places where ENO behaves differently to the WENO benchmark, especially near sharp corners in the exact solution.

For case 2 (Δx=0.001), comparing solutions by all methods from Figs. (6), (8), (10), (12) with benchmark solution from Fig. (13), we can see that along with a good correlation with the benchmark, by increasing the grid points we are getting solutions with all methods closer to the exact solution. We also observe that, in case 2, the 1st order upwind with flux difference splitting solution from Fig. (6) has comparable dissipation to that shown by MUSCL scheme in the benchmark solution in Fig. (13). Figure (8), as expected shows oscillations for the R-K with artificial dissipation scheme but with narrow and tall peaks only at the discontinuities. Figs. (10) and (12) for MUSCL and ENO with finer mesh are almost identical to the exact solution given in Fig. (13).

F.) Comparing solutions of different methods with each other:

i. For Case 1:

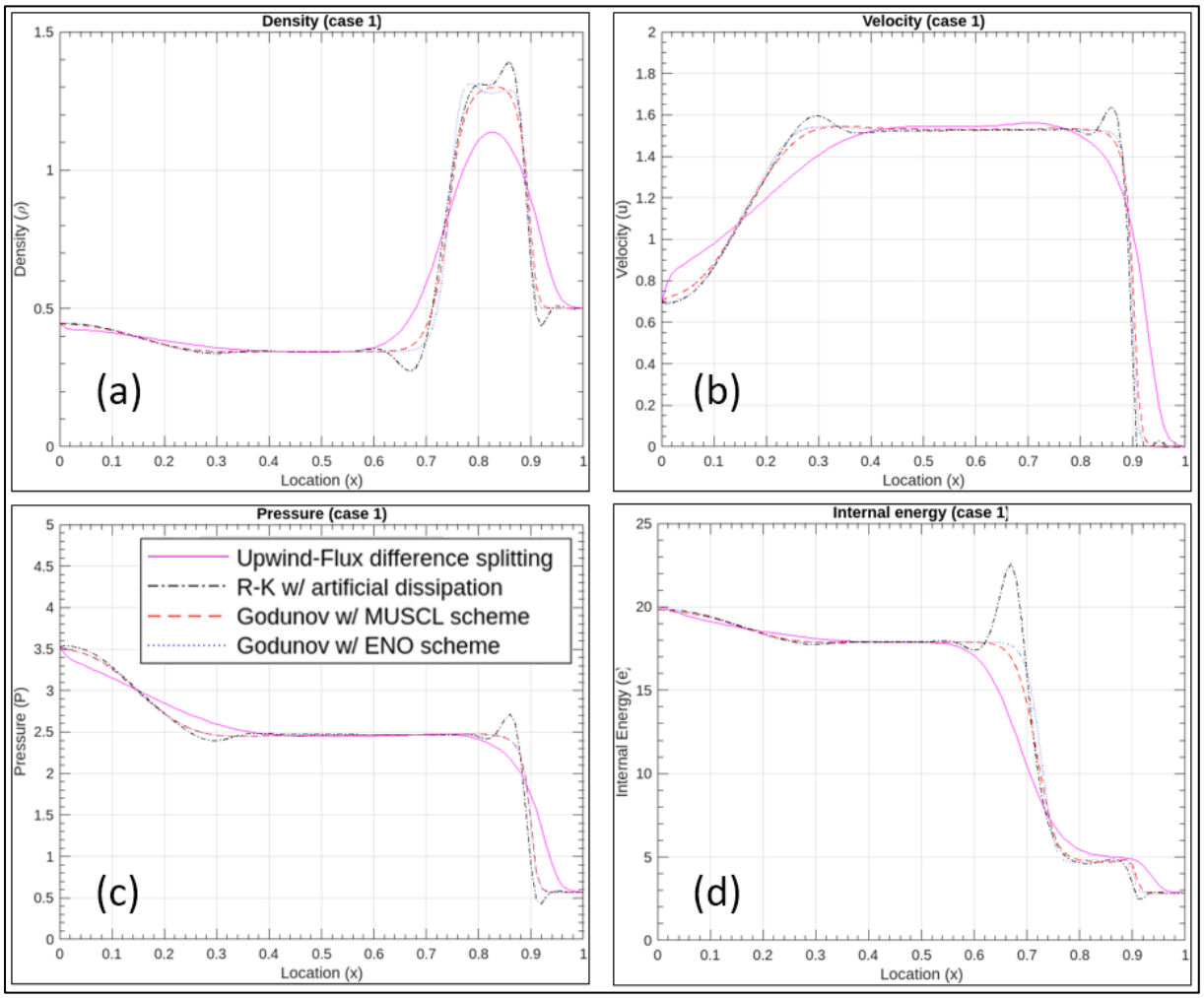

Fig.14. *Solutions for (a) Density, (b) Velocity, (c) Pressure, (d) Internal energy at T=0.16 with Δx=0.01 (case 1) for shock-tube problem with lax initial conditions using different methods.*

Figure (14) shows overall good correlation between solutions obtained using different methods for case 1. The 1st order upwind with flux difference splitting has the highest dissipation among all the methods. 2nd order Runge-Kutta with artificial dissipation shows low dissipation but the solution has oscillations near the discontinuities. In regions other than the discontinuities, the 2nd order R-K with dissipation is closer to MUSCL scheme solution than the 1st order upwind with flux difference splitting. Among all the methods both Godunov with MUSCL and 3rd order ENO have low dissipations and provide sharper gradients.

ii. For Case 2:

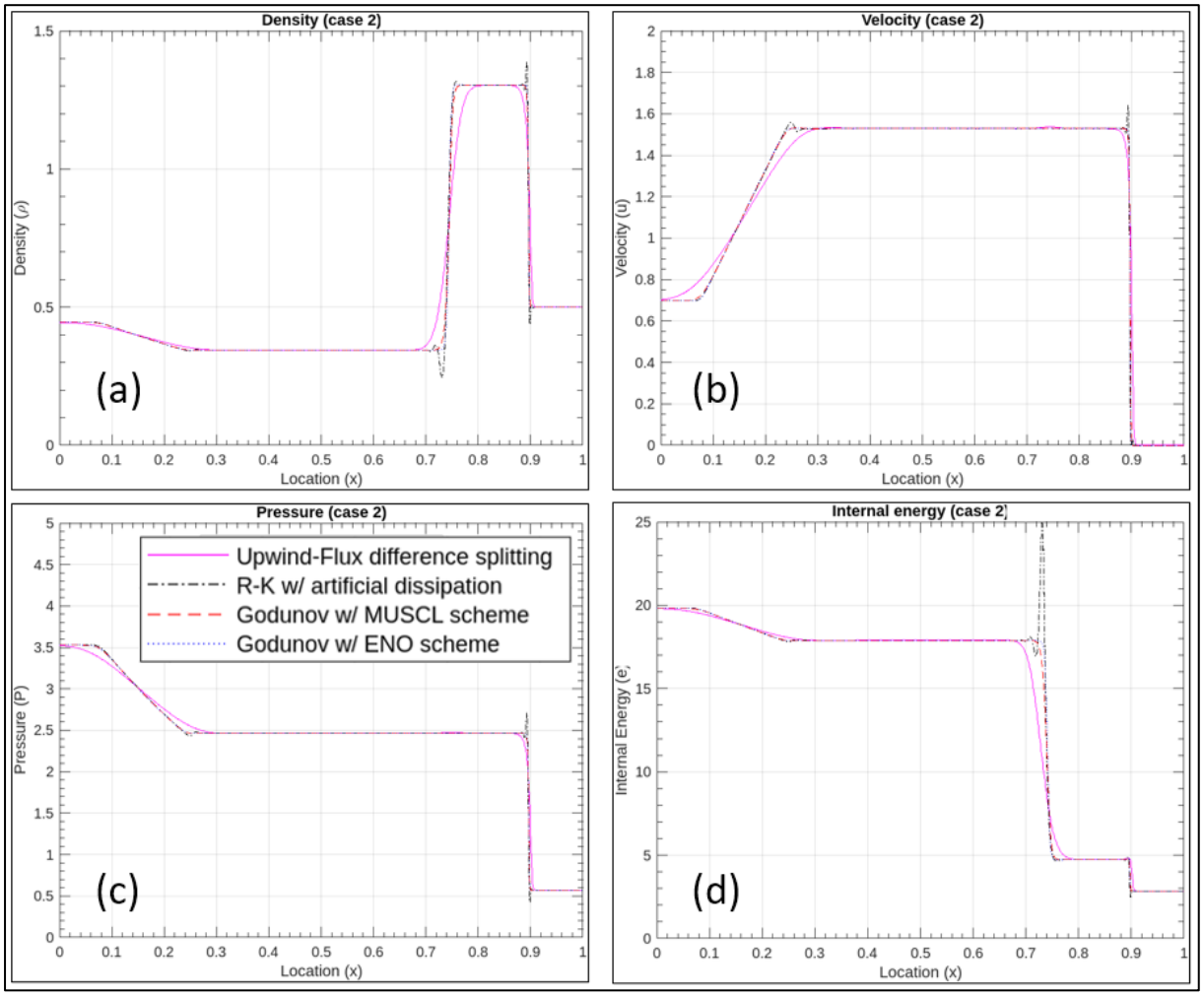

Fig.15. *Solutions for (a) Density, (b) Velocity, (c) Pressure, (d) Internal energy at T=0.16 with Δx=0.001 (case 2) for shock-tube problem with lax initial conditions using different methods.*

Figure (15) shows overall good correlation between solutions obtained using different methods for case 2. The finer mesh in case 2 results in better correlation among solutions with all methods than in case 1. Here also, we observe upwind with flux difference splitting has highest dissipation than other methods. When we look at R-K with artificial dissipation, we can see that oscillation peaks are much sharper and narrower in case 2 and have better correlation with MUSCL solution in most of the domain. MUSCL and ENO solutions are almost exactly the same with minor deviations at some places and both have lower dissipation.

Summary & Conclusions

- From this study we observed good alignment between benchmark solution and solutions found using Upwind with flux difference splitting, Runge-Kutta with artificial dissipation, Godunov with MUSCL and ENO methods.
- Using a finer mesh gives solutions with all methods closely matching with the exact solution.
- We also observed that scheme having lower order of accuracy, such as 1^{st} order upwind, with finer mesh produced results with dissipation similar or smaller than 2^{nd} order accurate methods with coarser mesh.
- The oscillation peaks in Runge-Kutta with artificial dissipation become narrower and taller near discontinuities with increase in number of grid points.
- Higher order schemes give lower dissipation than lower order schemes for the same grid size which aligns with the theoretical concept.
- We have validated the implementation of different methods used in this study as all the solutions are closely aligning with the benchmark solution.

APPENDIX

A. Detailed steps used for Godunov with ENO scheme implementation:

ENO :- ($k = 3$) 3rd order scheme

i. Newton divided difference for selecting stencil

$m < k + 1$

So take $m = 3$

stencil $\Rightarrow \bar{u}_{(i-r)}, \bar{u}_{(i-r+1)}, \cdots, \bar{u}_{(i-r+m-1)}$

here, $\bar{u}_{(i-r)}, \bar{u}_{(i-r+1)}, \bar{u}_{(i-r+2)}$

↓

left-most point in stencil.

condition: $r + s + 1 = k = 3$ for $r, s \geq 0$

possible value of $r \Rightarrow 0, 1, 2$

Take $j = i - r$

Newton divided difference is given by,

$$D(\bar{u}_{(j)}, \bar{u}_{(j+1)}, \bar{u}_{(j+2)}) = \frac{1}{\Delta x}\left\{ D(\bar{u}_{(j+1)}, \bar{u}_{(j+2)}) - D(\bar{u}_{(j)}, \bar{u}_{(j+1)}) \right\}$$

$$= \frac{1}{\Delta x^2}\left[(\bar{u}_{(j+2)} - \bar{u}_{(j+1)}) - (\bar{u}_{(j+1)} - \bar{u}_{(j)}) \right]$$

Compute D for values of r

i.e, $D(r=0)$, $D(r=1)$, $D(r=2)$

then select r for smallest value of $|D|$

ii. Based on value of r get values for coefficient C_{rj} as per:

k	r	j=1	j=2	j=3
3	-1	11/6	-7/6	1/3
	0	1/3	5/6	-1/6
	1	-1/6	5/6	1/3
	2	1/3	-7/6	11/6

iii. Calculate $u_{(i\pm1/2)}$ at cell boundaries:

$$u_{(i+1/2)}^{-} = \sum_{j=0}^{2} C_{rj}\, \bar{u}_{(i-r+j)}$$

$$u_{(i-1/2)}^{+} = \sum_{j=0}^{2} C_{(r-1)j}\, \bar{u}_{(i-r+j)}$$

$u_L \Rightarrow u(i+1/2)^-$ for i^{th} cell

$u_R \Rightarrow u(i-1/2)^+$ for $(i+1)^{th}$ cell

iv. L-F flux function:-

$$F(i) = \frac{1}{2}\left[E(u_L) + E(u_R)\right] - \frac{1}{2}|\alpha|(u_R - u_L)$$

where, $|\alpha| = \max(u_R, u_R+a, u_R-a, u_L, u_L+a, u_L-a)$

$$a = \sqrt{\frac{\gamma p}{\rho}} \Rightarrow a_{(i)} = \sqrt{\frac{1.4 \times P(i)}{\rho(i)}}$$

v. Time advancing:-

$$u_{(i)}^{(n+1)} = u_{(i)}^{(n)} - \frac{\Delta t}{\Delta x}\left[F_{(i)}^{(n)} - F_{(i-1)}^{(n)}\right]$$

Values for coefficient C_{rj} for constant Δx are computed as per the formula given below:

$$c_{rj} = \sum_{m=j+1}^{k} \frac{\sum_{\substack{l=0 \\ l\neq m}}^{k} \prod_{\substack{q=0 \\ q\neq m,l}}^{k} (r-q+1)}{\prod_{\substack{l=0 \\ l\neq m}}^{k} (m-l)}$$

B. MATLAB codes used for CFD final project:-

1.) Code for **Upwind with Flux Difference Splitting (1st order)** method to plot solutions to the shock-tube problem with lax initial conditions:

```
clc;
dx=0.01;  %input dx (case 2 - dx=0.001)
dt=0.0001;  %input dt (case 2 - dt=0.00001)
c=dt/dx;
T=0.16;
NX=1+1/dx;
NT=floor(1+T/dt);
%% Initialise Q1, Q2, Q3 (at t=0)
for it=1:NT
    for j=1:NX
        x=(j-1)*dx;
        if x<0.5
            Q1(it,j)=0.445;
            Q2(it,j)=0.311;
            Q3(it,j)=8.928;
        else
            Q1(it,j)=0.5;
            Q2(it,j)=0;
            Q3(it,j)=1.4275;
        end
    end
end
%% Upwind - Flux difference splitting method
for it=1:NT-1
    t=dt*it;
    % Compute fluxes
    for j=1:NX
        E1(it,j)=Q2(it,j);
        u(j)=Q2(it,j)/Q1(it,j);
        p(j)=0.4*(Q3(it,j)-0.5*((Q2(it,j))^2)/Q1(it,j));
        a(j)=sqrt(abs(1.4*p(j)/Q1(it,j))); %speed of sound
        E2(it,j)=Q2(it,j)*u(j)+p(j);
        E3(it,j)=Q3(it,j)*u(j)+p(j)*u(j);
    end
    for n=1:NX-1
        w=[Q1(it,n) (Q1(it,n)+a(n)) (Q1(it,n)-a(n)) Q1(it,n+1) (Q1(it,n+1)+a(n)) (Q1(it,n+1)-
a(n))];
        F1(n)=0.5*(E1(it,n)+E1(it,n+1))-0.5*max(w)*(Q1(it,n+1)-Q1(it,n));
    end
    for n=1:NX-1
        w=[Q2(it,n) (Q2(it,n)+a(n)) (Q2(it,n)-a(n)) Q2(it,n+1) (Q2(it,n+1)+a(n)) (Q2(it,n+1)-
a(n))];
        F2(n)=0.5*(E2(it,n)+E2(it,n+1))-0.5*max(w)*(Q2(it,n+1)-Q2(it,n));
    end
    for n=1:NX-1
        w=[Q3(it,n) (Q3(it,n)+a(n)) (Q3(it,n)-a(n)) Q3(it,n+1) (Q3(it,n+1)+a(n)) (Q3(it,n+1)-
a(n))];
        F3(n)=0.5*(E3(it,n)+E3(it,n+1))-0.5*max(w)*(Q3(it,n+1)-Q3(it,n));
    end
    % Calculate Q1, Q2, Q3 at t(n+1)
    for k=2:NX-1
        Q1(it+1,k)=Q1(it,k)-c*(F1(k)-F1(k-1));
        Q2(it+1,k)=Q2(it,k)-c*(F2(k)-F2(k-1));
        Q3(it+1,k)=Q3(it,k)-c*(F3(k)-F3(k-1));
    end
end
%% Plotting solution
figure;
d=Q1(NT,:);
```

```
v=Q2(NT,:)./Q1(NT,:);
P=0.4.*(Q3(NT,:)-0.5.*((Q2(NT,:).^2)./Q1(NT,:)));
U=(Q3(NT,:)./Q1(NT,:))-0.5.*(v.^2);
x=linspace(0,1,NX);
plot(x,d,"black");
ylim([0 1.5]);
title('Density - Upwind-Flux difference splitting @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Density (\rho)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,v,"black");
ylim([0 2]);
title('Velocity - Upwind-Flux difference splitting @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Velocity (u)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,P,"black");
ylim([0 5]);
title('Pressure - Upwind-Flux difference splitting @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Pressure (P)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,U,"black");
ylim([0 25]);
title('Internal energy - Upwind-Flux difference splitting @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Internal Energy (e)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
%% Saving solution for comparing
save('UFDS_density.mat','d');
save('UFDS_velocity.mat','v');
save('UFDS_pressure.mat','P');
save('UFDS_energy.mat','U');
```

2.) Code for **Runge-Kutta with Artificial Dissipation (2nd order)** method to plot solutions to the shock-tube problem with lax initial conditions:

```
clc;
dx=0.01;  %input dx (case 2 - dx=0.001)
dt=0.001;  %input dt (case 2 - dt=0.0001)
c=dt/dx;
T=0.16;
NX=1+1/dx;
NT=floor(1+T/dt);
e=0.1;
%% Initialise Q1, Q2, Q3 (at t=0)
for i=1:NT
    for j=1:NX
        x=(j-1)*dx;
        if x<0.5
            Q1(i,j)=0.445;
            Q2(i,j)=0.311;
            Q3(i,j)=8.928;
        else
            Q1(i,j)=0.5;
            Q2(i,j)=0;
```

```
            Q3(i,j)=1.4275;
        end
    end
end
%% R-K (2nd order) w/ 4th order dissipation
for i=1:NT-1
    % R-K loop 1
    for j=1:NX
        E1(i,j)=Q2(i,j);
        u(j)=Q2(i,j)/Q1(i,j);
        p(j)=0.4*(Q3(i,j)-0.5*((Q2(i,j))^2)/Q1(i,j));
        E2(i,j)=Q2(i,j)*u(j)+p(j);
        E3(i,j)=Q3(i,j)*u(j)+p(j)*u(j);
    end
    for j=2:NX-1
        Q1(i+1,j)=Q1(i,j)-c*(E1(i,j+1)-E1(i,j-1))/4;
        Q2(i+1,j)=Q2(i,j)-c*(E2(i,j+1)-E2(i,j-1))/4;
        Q3(i+1,j)=Q3(i,j)-c*(E3(i,j+1)-E3(i,j-1))/4;
    end
    % R-K loop 2
    for j=1:NX
        E1(i,j)=Q2(i+1,j);
        u(j)=Q2(i+1,j)/Q1(i+1,j);
        p(j)=0.4*(Q3(i+1,j)-0.5*((Q2(i+1,j))^2)/Q1(i+1,j));
        E2(i,j)=Q2(i+1,j)*u(j)+p(j);
        E3(i,j)=Q3(i+1,j)*u(j)+p(j)*u(j);
    end
    for j=2:NX-1
        Q1(i+1,j)=Q1(i,j)-(E1(i,j+1)-E1(i,j-1))*c/2;
        Q2(i+1,j)=Q2(i,j)-(E2(i,j+1)-E2(i,j-1))*c/2;
        Q3(i+1,j)=Q3(i,j)-(E3(i,j+1)-E3(i,j-1))*c/2;
    end
    % Artificial dissipation loop
    for j=3:NX-2
        Q1(i+1,j)=Q1(i+1,j)-e*(Q1(i,j-2)-4*Q1(i,j-1)+6*Q1(i,j)-4*Q1(i,j+1)+Q1(i,j+2));
        Q2(i+1,j)=Q2(i+1,j)-e*(Q2(i,j-2)-4*Q2(i,j-1)+6*Q2(i,j)-4*Q2(i,j+1)+Q2(i,j+2));
        Q3(i+1,j)=Q3(i+1,j)-e*(Q3(i,j-2)-4*Q3(i,j-1)+6*Q3(i,j)-4*Q3(i,j+1)+Q3(i,j+2));
    end
end
%% Plot solutions
figure;
d=Q1(NT,:);
v=Q2(NT,:)./Q1(NT,:);
P=0.4.*(Q3(NT,:)-0.5.*((Q2(NT,:).^2)./Q1(NT,:)));
U=(Q3(NT,:)./Q1(NT,:))-0.5.*(u.^2);
x=linspace(0,1,NX);
plot(x,d,"black");
ylim([0 1.5]);
title('Density - R-K w/ artificial dissipation @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Density (\rho)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,v,"black");
ylim([0 2]);
title('Velocity - R-K w/ artificial dissipation @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Velocity (u)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,P,"black");
```

```
ylim([0 5]);
title('Pressure - R-K w/ artificial dissipation @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Pressure (P)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,U,"black");
ylim([0 25]);
title('Internal energy - R-K w/ artificial dissipation @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Internal Energy (e)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
%% Saving solution for comparing
save('RKwAD_density.mat','d');
save('RKwAD_velocity.mat','v');
save('RKwAD_pressure.mat','P');
save('RKwAD_energy.mat','U');
```

3.) Code for **Godunov with MUSCL scheme (Monotonicity Preserving)** method to plot solutions to the shock-tube problem with lax initial conditions:

```
clc;
dx=0.01;  %input dx (case 2 - dx=0.001)
dt=0.0001;  %input dt (case 2 - dt=0.00001)
c=dt/dx;
T=0.16;
NX=1+1/dx;
NT=floor(1+T/dt);
%% Initialise Q1, Q2, Q3 (at t=0)
for it=1:NT
    for j=1:NX
        x=(j-1)*dx;
        if x<0.5
            Q1(it,j)=0.445;
            Q2(it,j)=0.311;
            Q3(it,j)=8.928;
        else
            Q1(it,j)=0.5;
            Q2(it,j)=0;
            Q3(it,j)=1.4275;
        end
    end
end
%% Godunov with MUSCL scheme
for it=1:NT-1
    t=dt*it;
    % Compute speed of sound
    for j=1:NX
        p(j)=0.4*(Q3(it,j)-0.5*((Q2(it,j))^2)/Q1(it,j));
        a(j)=sqrt(abs(1.4*p(j)/Q1(it,j)));
    end
    % MUSCL slope limiter
    S1=slopelim(Q1,it,NX,dx);
    S2=slopelim(Q2,it,NX,dx);
    S3=slopelim(Q3,it,NX,dx);
    % u(i+1/2)+-
    [QL1,QR1]=uaprox(Q1,S1,it,NX,dx);
    [QL2,QR2]=uaprox(Q2,S2,it,NX,dx);
    [QL3,QR3]=uaprox(Q3,S3,it,NX,dx);
```

```
    % Compute fluxes
    for j=1:NX-1
        EL1(it,j)=QL2(it,j);
        u(j)=QL2(it,j)/QL1(it,j);
        p(j)=0.4*(QL3(it,j)-0.5*((QL2(it,j))^2)/QL1(it,j));
        EL2(it,j)=QL2(it,j)*u(j)+p(j);
        EL3(it,j)=QL3(it,j)*u(j)+p(j)*u(j);
    end
    for j=1:NX-1
        ER1(it,j)=QR2(it,j);
        u(j)=QR2(it,j)/QR1(it,j);
        p(j)=0.4*(QR3(it,j)-0.5*((QR2(it,j))^2)/QR1(it,j));
        ER2(it,j)=QR2(it,j)*u(j)+p(j);
        ER3(it,j)=QR3(it,j)*u(j)+p(j)*u(j);
    end
    for n=1:NX-1
        w=[QL1(it,n) (QL1(it,n)+a(n)) (QL1(it,n)-a(n)) QR1(it,n) (QR1(it,n)+a(n)) (QR1(it,n)-
a(n))];
        F1(n)=0.5*(EL1(it,n)+ER1(it,n))-0.5*max(w)*(QR1(it,n)-QL1(it,n));
    end
    for n=1:NX-1
        w=[QL2(it,n) (QL2(it,n)+a(n)) (QL2(it,n)-a(n)) QR2(it,n) (QR2(it,n)+a(n)) (QR2(it,n)-
a(n))];
        F2(n)=0.5*(EL2(it,n)+ER2(it,n))-0.5*max(w)*(QR2(it,n)-QL2(it,n));
    end
    for n=1:NX-1
        w=[QL3(it,n) (QL3(it,n)+a(n)) (QL3(it,n)-a(n)) QR3(it,n) (QR3(it,n)+a(n)) (QR3(it,n)-
a(n))];
        F3(n)=0.5*(EL3(it,n)+ER3(it,n))-0.5*max(w)*(QR3(it,n)-QL3(it,n));
    end
    % Calculate Q1, Q2, Q3 at t(n+1)
    for k=2:NX-1
        Q1(it+1,k)=Q1(it,k)-c*(F1(k)-F1(k-1));
        Q2(it+1,k)=Q2(it,k)-c*(F2(k)-F2(k-1));
        Q3(it+1,k)=Q3(it,k)-c*(F3(k)-F3(k-1));
    end
end
%% Plotting solution
figure;
d=Q1(NT,:);
v=Q2(NT,:)./Q1(NT,:);
P=0.4.*(Q3(NT,:)-0.5.*((Q2(NT,:).^2)./Q1(NT,:)));
U=(Q3(NT,:)./Q1(NT,:))-0.5.*(v.^2);
x=linspace(0,1,NX);
plot(x,d,"black");
ylim([0 1.5]);
title('Density - Godunov w/ MUSCL scheme @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Density (\rho)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,v,"black");
ylim([0 2]);
title('Velocity - Godunov w/ MUSCL scheme @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Velocity (u)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,P,"black");
ylim([0 5]);
title('Pressure - Godunov w/ MUSCL scheme @T=0.16 (case 1)');
```

```
xlabel('Location (x)');
ylabel('Pressure (P)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,U,"black");
ylim([0 25]);
title('Internal energy - Godunov w/ MUSCL scheme @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Internal Energy (e)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
%% Saving solution for comparing
save('GwMUSCL_density.mat','d');
save('GwMUSCL_velocity.mat','v');
save('GwMUSCL_pressure.mat','P');
save('GwMUSCL_energy.mat','U');

% MUSCL slope limiter function
function S=slopelim(u,it,NX,dx)
for i=2:NX-1
    a=(u(it,i+1)-u(it,i))/dx;
    b=(u(it,i)-u(it,i-1))/dx;
    A=[abs(a) abs(b)];
    C=sign(a);
    M=min(A);
    if a*b>0
        S(i)=C*M;
    else
        S(i)=0;
    end
end
S(1)=(u(it,2)-u(it,1))/dx;
S(NX)=(u(it,NX)-u(it,NX-1))/dx;
end

% u(i+1/2)+- Polynomial approx. function
function [uL,uR]=uaprox(u,S,it,NX,dx)
for m=1:NX-1
    uL(it,m)=u(it,m)+S(m)*dx/2;
    uR(it,m)=u(it,m+1)-S(m+1)*dx/2;
end
end
```

4.) Code for **Godunov with ENO (3rd order)** method to plot solutions to the shock-tube problem with lax initial conditions:

```
clc;
dx=0.01;  %input dx  (case 2 - dx=0.001)
dt=0.0001;  %input dt  (case 2 - dt=0.00001)
c=dt/dx;
T=0.16;
NX=1+1/dx;
NT=floor(1+T/dt);
%% Initialise Q1, Q2, Q3 (at t=0)
for it=1:NT
    for j=1:NX
        x=(j-1)*dx;
        if x<0.5
            Q1(it,j)=0.445;
            Q2(it,j)=0.311;
            Q3(it,j)=8.928;
        else
            Q1(it,j)=0.5;
```

```
            Q2(it,j)=0;
            Q3(it,j)=1.4275;
        end
    end
end
%% Godunov with ENO scheme
for it=1:NT-1
    t=dt*it;
    % Compute speed of sound
    for j=1:NX
        p(j)=0.4*(Q3(it,j)-0.5*((Q2(it,j))^2)/Q1(it,j));
        a(j)=sqrt(abs(1.4*p(j)/Q1(it,j)));
    end
    % u(i+1/2)+-
    [QL1,QR1]=ENOaprox(Q1,it,NX);
    [QL2,QR2]=ENOaprox(Q2,it,NX);
    [QL3,QR3]=ENOaprox(Q3,it,NX);
    % Compute fluxes
    for j=3:NX-2
        EL1(it,j)=QL2(it,j);
        u(j)=QL2(it,j)/QL1(it,j);
        p(j)=0.4*(QL3(it,j)-0.5*((QL2(it,j))^2)/QL1(it,j));
        EL2(it,j)=QL2(it,j)*u(j)+p(j);
        EL3(it,j)=QL3(it,j)*u(j)+p(j)*u(j);
    end
    for j=3:NX-2
        ER1(it,j)=QR2(it,j);
        u(j)=QR2(it,j)/QR1(it,j);
        p(j)=0.4*(QR3(it,j)-0.5*((QR2(it,j))^2)/QR1(it,j));
        ER2(it,j)=QR2(it,j)*u(j)+p(j);
        ER3(it,j)=QR3(it,j)*u(j)+p(j)*u(j);
    end
    for n=3:NX-3
        w=[QL1(it,n) (QL1(it,n)+a(n)) (QL1(it,n)-a(n)) QR1(it,n+1) (QR1(it,n+1)+a(n))
(QR1(it,n+1)-a(n))];
        F1(n)=0.5*(EL1(it,n)+ER1(it,n+1))-0.5*max(w)*(QR1(it,n+1)-QL1(it,n));
    end
    for n=3:NX-3
        w=[QL2(it,n) (QL2(it,n)+a(n)) (QL2(it,n)-a(n)) QR2(it,n+1) (QR2(it,n+1)+a(n))
(QR2(it,n+1)-a(n))];
        F2(n)=0.5*(EL2(it,n)+ER2(it,n+1))-0.5*max(w)*(QR2(it,n+1)-QL2(it,n));
    end
    for n=3:NX-3
        w=[QL3(it,n) (QL3(it,n)+a(n)) (QL3(it,n)-a(n)) QR3(it,n+1) (QR3(it,n+1)+a(n))
(QR3(it,n+1)-a(n))];
        F3(n)=0.5*(EL3(it,n)+ER3(it,n+1))-0.5*max(w)*(QR3(it,n+1)-QL3(it,n));
    end
    % Calculate Q1, Q2, Q3 at t(n+1)
    for k=4:NX-3
        Q1(it+1,k)=Q1(it,k)-c*(F1(k)-F1(k-1));
        Q2(it+1,k)=Q2(it,k)-c*(F2(k)-F2(k-1));
        Q3(it+1,k)=Q3(it,k)-c*(F3(k)-F3(k-1));
    end
end
%% Plotting solution
figure;
d=Q1(NT,:);
v=Q2(NT,:)./Q1(NT,:);
P=0.4.*(Q3(NT,:)-0.5.*((Q2(NT,:).^2)./Q1(NT,:)));
U=(Q3(NT,:)./Q1(NT,:))-0.5.*(v.^2);
x=linspace(0,1,NX);
plot(x,d,"black");
ylim([0 1.5]);
```

```
title('Density - Godunov w/ ENO scheme @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Density (\rho)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,v,"black");
ylim([0 2]);
title('Velocity - Godunov w/ ENO scheme @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Velocity (u)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,P,"black");
ylim([0 5]);
title('Pressure - Godunov w/ ENO scheme @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Pressure (P)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
figure;
plot(x,U,"black");
ylim([0 25]);
title('Internal energy - Godunov w/ ENO scheme @T=0.16 (case 1)');
xlabel('Location (x)');
ylabel('Internal Energy (e)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
%% Saving solution for comparing
save('GwENO_density.mat','d');
save('GwENO_velocity.mat','v');
save('GwENO_pressure.mat','P');
save('GwENO_energy.mat','U');

% ENO approximation function
function [uL,uR]=ENOaprox(u,it,NX)
% Selecting stencil with divided difference
for i=1:NX-1
    D(1,i)=u(it,i+1)-u(it,i);
end
for i=1:NX-2
    D(2,i)=D(1,i+1)-D(1,i);
end
for i=2:NX-2
    is=i;
    for m=1:2
        if abs(D(m,is-1))<abs(D(m,is))
            is=is-1;
        end
    end
    R(i)=i-is;
end
% u(i+1/2)- & u(i-1/2)+
C=[11/6 -7/6 1/3; 1/3 5/6 -1/6; -1/6 5/6 1/3; 1/3 -7/6 11/6];
for m=3:NX-2
    j=R(m)+2;
    k=R(m);
    uL(it,m)=C(j,1)*u(it,m-k)+C(j,2)*u(it,m-k+1)+C(j,3)*u(it,m-k+2);
    uR(it,m)=C(j-1,1)*u(it,m-k)+C(j-1,2)*u(it,m-k+1)+C(j-1,3)*u(it,m-k+2);
end
end
```

x

5.) Code to **Compare results** obtained using different methods:

```
clc;
dx=0.01; %as per case
NX=1+1/dx;
%% Compare density solution
a1 = matfile('UFDS_density.mat');
d1=a1.d;
a2 = matfile('RKwAD_density.mat');
d2=a2.d;
a3 = matfile('GwMUSCL_density.mat');
d3=a3.d;
a4 = matfile('GwENO_density.mat');
d4=a4.d;
x=linspace(0,1,NX);
figure;
plot(x,d1,"magenta",x,d2,"-.black",x,d3,"--red",x,d4,":blue");
xlabel('Location (x)');
ylabel('Density (\rho)');
ylim([0 1.5]);
title('Density (case 1)');
legend('Upwind-Flux difference splitting','R-K w/ artificial dissipation','Godunov w/ MUSCL
scheme','Godunov w/ ENO scheme','Location','northwest');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
%% Compare velocity solution
b1 = matfile('UFDS_velocity.mat');
v1=b1.v;
b2 = matfile('RKwAD_velocity.mat');
v2=b2.v;
b3 = matfile('GwMUSCL_velocity.mat');
v3=b3.v;
b4 = matfile('GwENO_velocity.mat');
v4=b4.v;
x=linspace(0,1,NX);
figure;
plot(x,v1,"magenta",x,v2,"-.black",x,v3,"--red",x,v4,":blue");
xlabel('Location (x)');
ylabel('Velocity (u)');
ylim([0 2]);
title('Velocity (case 1)');
legend('Upwind-Flux difference splitting','R-K w/ artificial dissipation','Godunov w/ MUSCL
scheme','Godunov w/ ENO scheme','Location','northwest');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
%% Compare pressure solution
c1 = matfile('UFDS_pressure.mat');
p1=c1.P;
c2 = matfile('RKwAD_pressure.mat');
p2=c2.P;
c3 = matfile('GwMUSCL_pressure.mat');
p3=c3.P;
c4 = matfile('GwENO_pressure.mat');
p4=c4.P;
x=linspace(0,1,NX);
figure;
plot(x,p1,"magenta",x,p2,"-.black",x,p3,"--red",x,p3,":blue");
xlabel('Location (x)');
ylabel('Pressure (P)');
ylim([0 5]);
title('Pressure (case 1)');
legend('Upwind-Flux difference splitting','R-K w/ artificial dissipation','Godunov w/ MUSCL
scheme','Godunov w/ ENO scheme','Location','northwest');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
```

```
grid on;
%% Compare internal energy solution
f1 = matfile('UFDS_energy.mat');
e1=f1.U;
f2 = matfile('RKwAD_energy.mat');
e2=f2.U;
f3 = matfile('GwMUSCL_energy.mat');
e3=f3.U;
f4 = matfile('GwENO_energy.mat');
e4=f4.U;
x=linspace(0,1,NX);
figure;
plot(x,e1,"magenta",x,e2,"-.black",x,e3,"--red",x,e4,":blue");
xlabel('Location (x)');
ylabel('Internal Energy (e)');
ylim([0 25]);
title('Internal energy (case 1)');
legend('Upwind-Flux difference splitting','R-K w/ artificial dissipation','Godunov w/ MUSCL
scheme','Godunov w/ ENO scheme','Location','northwest');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
```