Politecnico
di Bari

REPORT IN
COMPUTATIONAL FLUID DYNAMICS
Flux Vector Splitting Methods

Fracchiolla Beniamino.

INDEX

1. Abstract

The purpose of the report shown here is to obtain the trend of primitive quantities such as density, speed and pressure of an air flow (modeled as a perfect gas) inside a shock tube. Specifically, the problem to be solved by computational procedure will be the Riemann problem to the Euler system. In the case of the Riemann problem we have constant initial conditions to the right and left of the shock tube and a discontinuity in the diaphragm of the same. The equations of the Euler system, on the other hand, characterize the behavior of non-viscous flows and without thermal conduction, all this translates into a non-linear system.

The computational procedure adopted consists in carrying out and comparing the results of 5 tests proposed by the text "Riemann Solvers and Numerical Methods for Fluid Dynamics" with three different codes:

1. The code "nozzle_w. f" solving the Euler equations quasi-1d by ROE method with first order discretization;
2. The code "nozzle_w. f" but this time with discretization provided by the MUSCL scheme;
3. Implement a code with different method, namely a Flux-Vector-Splitting scheme, specifically the Liou-Steffen method.

The tests consist of using boundary conditions that produce:

- Test 1: a right shock wave, a contact wave moving to the right and a left expansion zone;
- Test 2: two symmetrical expansion zones and a zero-speed contact zone between the two;
- Test 3: a strong right shock wave (Mach=198), contact surface and expansion zone on the left;
- Test 4: two strong shock waves traveling towards each other producing three discontinuities;
- Test 5: as test 3 but with a negative background speed so as to obtain a stationary discontinuity.

Downstream of the following tests will follow comments about the results obtained and finally in the conclusions a comment that compares the various methods.

2. Roe method

Roe's method is used for solving systems of hyperbolic differential equations when instead of using an exact solver of the Riemann problem an approximate solver is used to reduce the computational cost of the problem.

The proposed approximation consists in linearizing the problem by introducing, under certain assumptions, the linearization of **Roe A(u,v)**. This consists in suitably linearizing the Jacobian of the transformation thus obtaining the linearized Riemann problem:

$$\begin{cases} \dfrac{\partial \boldsymbol{w}}{\partial t} + \boldsymbol{A}(\boldsymbol{u}_L, \boldsymbol{u}_R)\dfrac{\partial \boldsymbol{w}}{\partial x} = \boldsymbol{0}, \\ \boldsymbol{w}(x,0) = \begin{cases} \boldsymbol{u}_L, & x < 0, \\ \boldsymbol{u}_R, & x > 0, \end{cases} \end{cases}$$

The exact solution of the Riemann problem for a linear system consists in solving two integrals in this case, instead, being the Jacobian of the transformation not constant, it is evaluated at the interface of two cells. This implies a local linearization of the same:

$$\boldsymbol{A}(\boldsymbol{v}_j^n, \boldsymbol{v}_{j+1}^n) = \boldsymbol{A}_{j+\frac{1}{2}}^n,$$

This allows you to express the numerical flow as:

$$\boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}(\boldsymbol{f}(\boldsymbol{u}) + \boldsymbol{f}(\boldsymbol{v})) - \frac{1}{2} \mid \boldsymbol{A}(\boldsymbol{u}, \boldsymbol{v}) \mid (\boldsymbol{v} - \boldsymbol{u}).$$

Where (v-u) can be expressed as the difference of the waves:

$$\boldsymbol{v} - \boldsymbol{u} = \sum_{k=1}^{p} \Delta\alpha_k(\boldsymbol{u}, \boldsymbol{v})\boldsymbol{r}_k(\boldsymbol{u}, \boldsymbol{v}).$$

At this point, the numerical flow of Roe:

$$\boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}(\boldsymbol{f}(\boldsymbol{u}) + \boldsymbol{f}(\boldsymbol{v})) - \frac{1}{2}\sum_{k=1}^{p}(\Delta\alpha_k \mid \lambda_k \mid \boldsymbol{r}_k)(\boldsymbol{u}, \boldsymbol{v}).$$

In all this we have not yet found the form of the Jacobian A linearized matrix. The application of this method to the solution of the Euler equations consists in evaluating the Roe matrix:

$$\boldsymbol{A}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{A}(M(\boldsymbol{u}, \boldsymbol{v}))$$

Where M(u,v) is an appropriate mean between the u and v states, operator M will be determined by a variable change or vector parameter:

$$\boldsymbol{w} \rightarrow \boldsymbol{u}(\boldsymbol{w})$$

In the case of the Euler equations w is defined as:

$$\boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} \rho^{\frac{1}{2}} \\ \rho^{\frac{1}{2}} u \\ \rho^{\frac{1}{2}} H \end{pmatrix}$$

Mentre la Matrice A e gli autovettori destri Associati:

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 \\ (\gamma-3)\dfrac{u^2}{2} & (3-\gamma)u & \gamma-1 \\ (\gamma-1)u^3 - \gamma u e & \gamma e - \dfrac{3}{2}(\gamma-1)u^2 & \gamma u \end{pmatrix}$$

$$\boldsymbol{r}_1(\boldsymbol{u}) = \begin{pmatrix} 1 \\ u-c \\ H-uc \end{pmatrix}, \quad \boldsymbol{r}_2(\boldsymbol{u}) = \begin{pmatrix} 1 \\ u \\ H - \dfrac{c^2}{\gamma-1} \end{pmatrix}, \quad \boldsymbol{r}_3(\boldsymbol{u}) = \begin{pmatrix} 1 \\ u+c \\ H+uc \end{pmatrix}.$$

At this point, the problem is not yet linearized, we will have to use the average values of Roe

$$\boldsymbol{w}^* = \frac{\boldsymbol{w}_L + \boldsymbol{w}_R}{2}$$

These will first make the primitive quantities depend on the local conditions on the right and left of the cell:

$$\overline{u} = \frac{w_2^*}{w_1^*}, \quad \overline{H} = \frac{w_3^*}{w_1^*}$$

$$\overline{c^2} = (\gamma-1)\left(\overline{H} - \frac{\overline{u^2}}{2}\right)$$

Later such averages will make jacobian matrix and right autovectors function from the local conditions present to the right and left of the cell:

$$\boldsymbol{A}(\overline{\boldsymbol{u}})$$

$$\boldsymbol{r}(\overline{\boldsymbol{u}})$$

Finally, making the numerical flow dependent on what happens locally on the right and left of the cell $\boldsymbol{g_{ROE}} = f(\boldsymbol{u_R}, \boldsymbol{u_L}, \overline{\boldsymbol{u}}) = f(\boldsymbol{\omega_R}, \boldsymbol{\omega_L})$:

$$\boldsymbol{g}_{Roe} = \frac{1}{2}(\boldsymbol{f}(\boldsymbol{u}_R) + \boldsymbol{f}(\boldsymbol{u}_L)) - \frac{1}{2}\sum_{k=1}^{3}(\Delta\alpha_k \mid \lambda_k(\overline{\boldsymbol{u}}) \mid \boldsymbol{r}_k(\overline{\boldsymbol{u}}))$$

3. Code implemented for the Roe method

The procedure adopted is to use the code "nozzle_w. f". Compared to the original code, the code "eus1d_ROE has been implemented. f", the boundary conditions of extrapolation of transmissive type in the subroutine *bcw*:

```
C supersonic inlet condition at left boundary/extrapolation condition
      do 10 ibc=1,mbc
        q(1-ibc,1)=q(1,1)
        q(1-ibc,2)=q(1,2)
        q(1-ibc,3)=q(1,3)
C supersonic outlet condition at right boundary/extrapolation condition
       do 20 ibc=1,mbc
        q(mx+ibc,1)=q(mx,1)
        q(mx+ibc,2)=q(mx,2)
        q(mx+ibc,3)=q(mx,3)
```

To solve tests 4 and 5, because of the negative speeds, it was necessary to modify the algorithm that derives the global time step in compliance with the CFL condition, this to take into account the cases in which the higher eigenvalue can be $|(u-a)|$:

```
          u = qpr(i,2)
          a = sqrt(gamma*qpr(i,3)/qpr(i,1))
          dt1(i) = cfl*dx/dabs(u+a)
          dt2(i) = cfl*dx/dabs(u-a)
          dtgl = min(dt1(i),dt2(i),dtgl)
```

Finally, in order to obtain a discretization at the first order, the portions of code related to MUSCL reconstruction were commented:

```
!         qp(i,m)=qpr(i,m)-0.25*(rkp*phi1*dw(i,m,1)
!    .                      +rkm*phi2*dw(i,m,2))
!         qm(i,m)=qpr(i-1,m)+0.25*(rkm*phi3*dw(i-1,m,1)
!    .                       +rkp*phi4*dw(i-1,m,2))
```

4. Test numerici metodo di Roe

Data consist of two constant initial states (ρ_L, u_L, p_L) e (ρ_R, u_R, p_R), separated by a break in place $x = x_0$. All five tests present a unit spatial domino divided into one hundred cells.

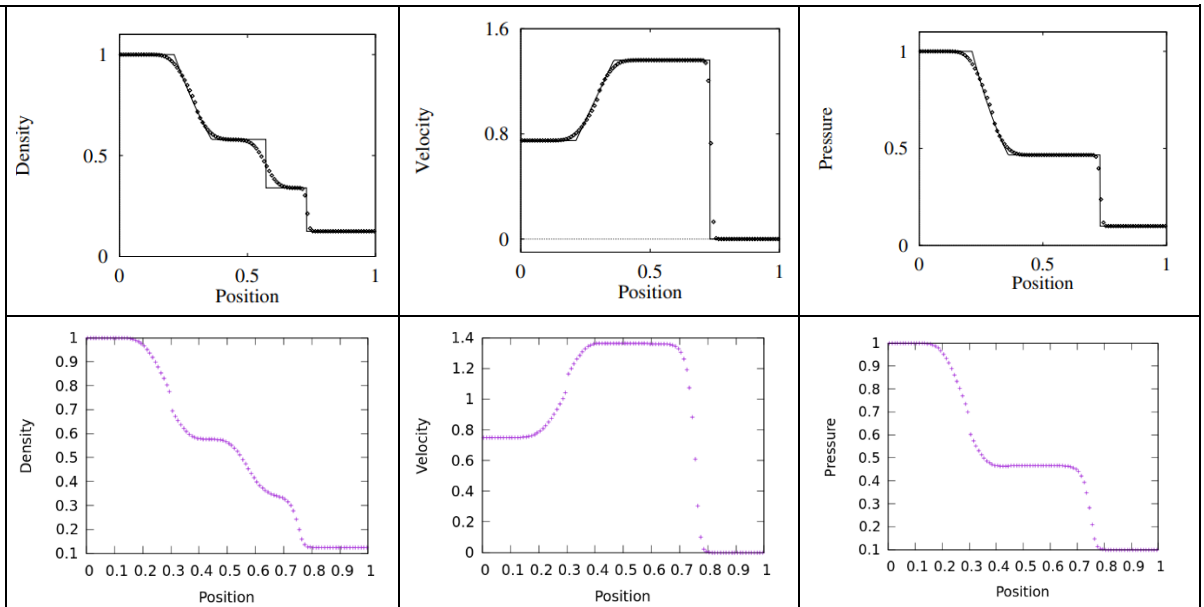
Test	x_0	T	ρ_L	u_L	p_L	ρ_R	u_R	p_R
1	0.3	0.2	1.0	0.70.05	1.0	0.125	0.0	0.1
2	0.5	0.25	1.0	-2.0	0.4	1.0	2.0	0.4
3	0.5	0.012	1.0	0.0	1000.0	1.0	0.0	0.01
4	0.4	0.035	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950
5	0.8	0.012	1.0	-19.59745	1000.0	1.0	-19.59745	0.01

Input data common to all tests, and characteristic of the code used are:

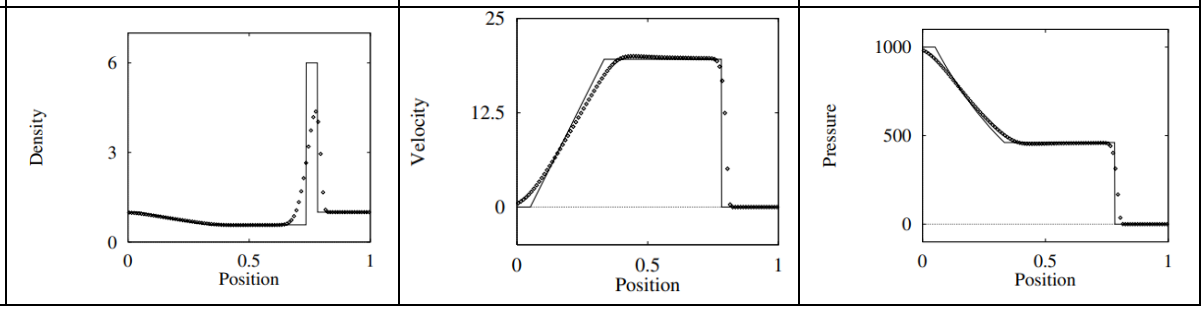
Input gamma:	1.4
Input dx:	0.01
Input cfl, itmax, itsmp:	0.9 10000 10
MUSCL coefficient:	-1
Input epsilon:	1.e-10

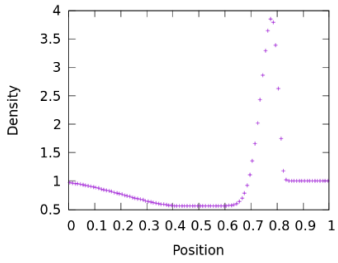
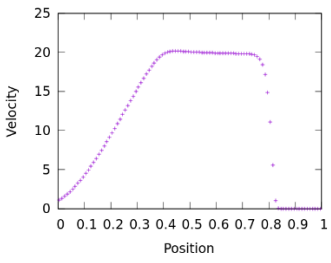
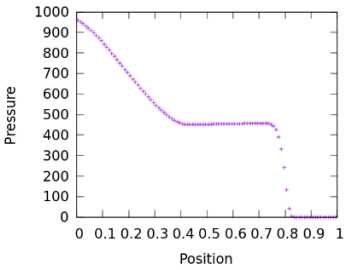
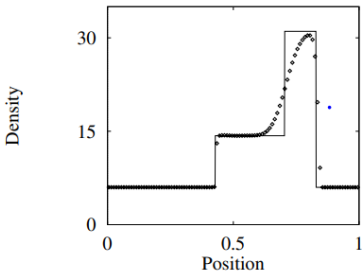
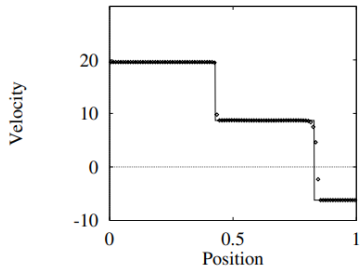
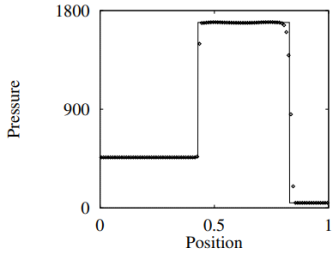
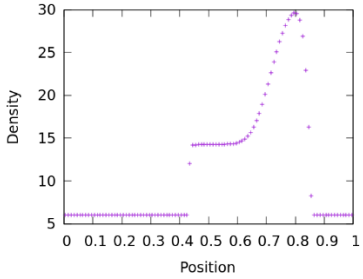
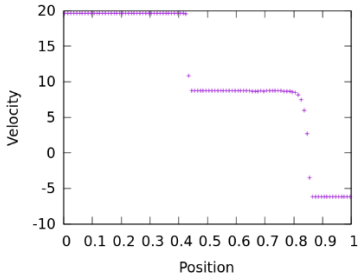
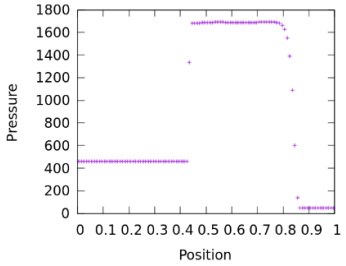
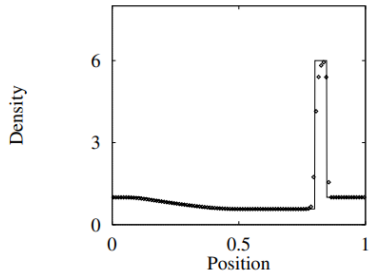
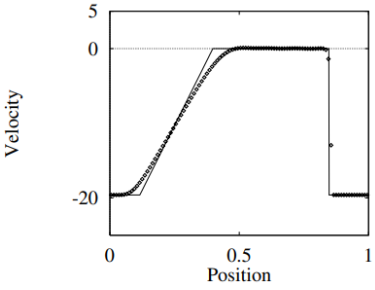
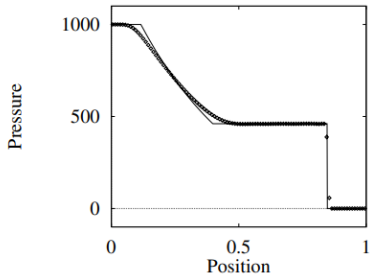
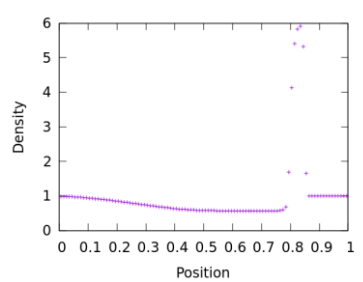
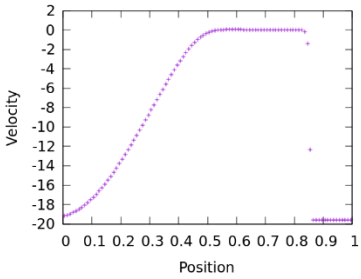
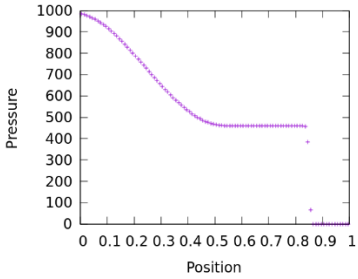
All five tests feature: a numerical solution and an exact inline continuous solution provided by the text [1] (pag. 372) and a numerical solution provided by the code used:

Test1



Test 3



			
Test 4			
			
Test 5			
			

5. **Observations testing method of Roe**

In the case of Test 2 the negative pressure error is obtained and in this case the book is also not reported any results. In this test we have two waves of expansion and "vacuum", this error is due to the poor ability of the Roe method to distinguish between impact and expansion given the local linearization of the problem adopted by this method.

6. Roe method with MUSCL discretization

Because the Riemann solvers (exact and not) are accurate to the first order in space because the flows are a function of the variables in the cells adjacent to the interface, MUSCL discretization schemes increase this spatial accuracy. This is possible by reconstructing the space dependent variable on multiple cells and using it to evaluate flows at the interface.

In the case of the Roe method I evaluate the flows at the first order by reconstructing the dependent variables taking into account only the value they assume in the cells u_j and u_{j+1}:

$$g^{(1)}_{j+\frac{1}{2},Roe} = g(u_j, u_{j+1})$$

Superior accuracy is achieved by using the value at the interface of such cells:

$$g^{(ho)}_{j+\frac{1}{2},Roe} = g(u^l_{j+\frac{1}{2}}, u^r_{j+\frac{1}{2}})$$

And these values at the interface can be evaluated by Taylor series expansion by enlarging, in a sense, the computational molecule of the numerical scheme:

$$g^{(ho)}_{j+\frac{1}{2},Roe} = \frac{1}{2}\left[f(u^l_{j+\frac{1}{2}}) + f(u^r_{j+\frac{1}{2}}) - |\overline{u_{j+\frac{1}{2}}}|\left(u^r_{j+\frac{1}{2}} - u^l_{j+\frac{1}{2}}\right)\right]$$

$$|\overline{u_{j+\frac{1}{2}}}| = \frac{f(u^r_{j+\frac{1}{2}}) - f(u^l_{j+\frac{1}{2}})}{u^r_{j+\frac{1}{2}} - u^l_{j+\frac{1}{2}}}$$

7. Code implemented for the Roe method with MUSCL discretization

The procedure adopted is to use the code "nozzle_w. f". Compared to the original code, they were implemented in the code "eus1d_ROE_MUSCL. f" the boundary conditions of transmissive extrapolation in the subroutine *bcw*:

```
C supersonic inlet condition at left boundary/extrapolation condition
      do 10 ibc=1,mbc
        q(1-ibc,1)=q(1,1)
        q(1-ibc,2)=q(1,2)
        q(1-ibc,3)=q(1,3)
C supersonic outlet condition at right boundary/extrapolation condition
      do 20 ibc=1,mbc
        q(mx+ibc,1)=q(mx,1)
        q(mx+ibc,2)=q(mx,2)
        q(mx+ibc,3)=q(mx,3)
```

To solve tests 4 and 5, because of the negative speeds, it was necessary to modify the algorithm that derives the global time step in compliance with the CFL condition, this to take into account the cases in which the higher eigenvalue can be $|(u-a)|$:

```
          u = qpr(i,2)
          a = sqrt(gamma*qpr(i,3)/qpr(i,1))
          dt1(i) = cfl*dx/dabs(u+a)
          dt2(i) = cfl*dx/dabs(u-a)
          dtgl = min(dt1(i),dt2(i),dtgl)
```

Finally, in order to obtain a discretization with a higher order than the first, the portions of code related to MUSCL reconstruction were de-commented:

```
        qp(i,m)=qpr(i,m)-0.25*(rkp*phi1*dw(i,m,1)
     .                  +rkm*phi2*dw(i,m,2))
        qm(i,m)=qpr(i-1,m)+0.25*(rkm*phi3*dw(i-1,m,1)
     .                   +rkp*phi4*dw(i-1,m,2))
```

8. Roe method numerical tests with MUSCL discretization

Data consist of two constant initial states (ρ_L, u_L, p_L) e (ρ_R, u_R, p_R), separated by a break in place $x = x_0$. All five tests present a unit spatial domino divided into one hundred cells.

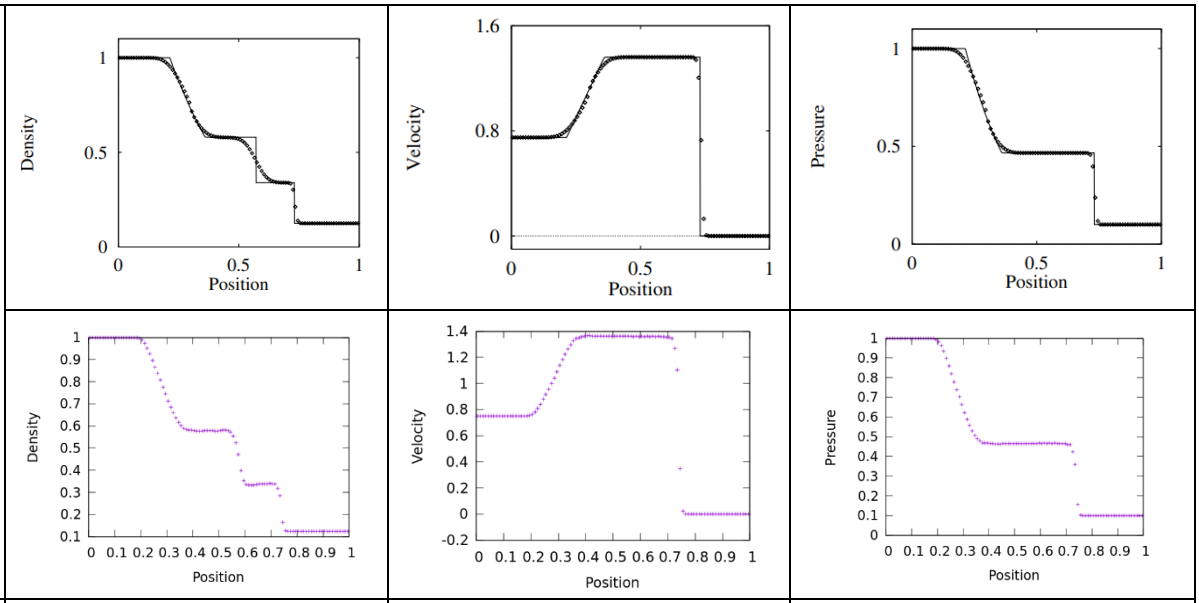
Test	x_0	T	ρ_L	u_L	p_L	ρ_R	u_R	p_R
1	0.3	0.2	1.0	0.70.05	1.0	0.125	0.0	0.1
2	0.5	0.25	1.0	-2.0	0.4	1.0	2.0	0.4
3	0.5	0.012	1.0	0.0	1000.0	1.0	0.0	0.01
4	0.4	0.035	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950
5	0.8	0.012	1.0	-19.59745	1000.0	1.0	-19.59745	0.01

The input data common to all tests, and specific to the code used are:

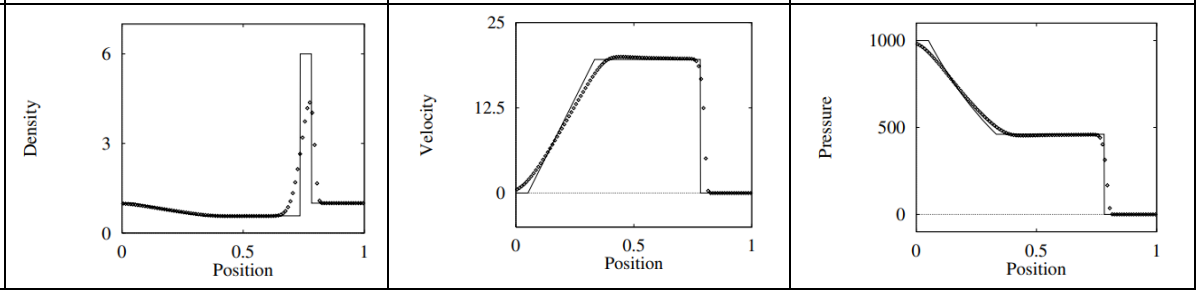
Input gamma:	1.4
Input dx:	0.01
Input cfl, itmax, itsmp:	0.9 10000 10
MUSCL coefficient:	-1
Input epsilon:	1.e-10

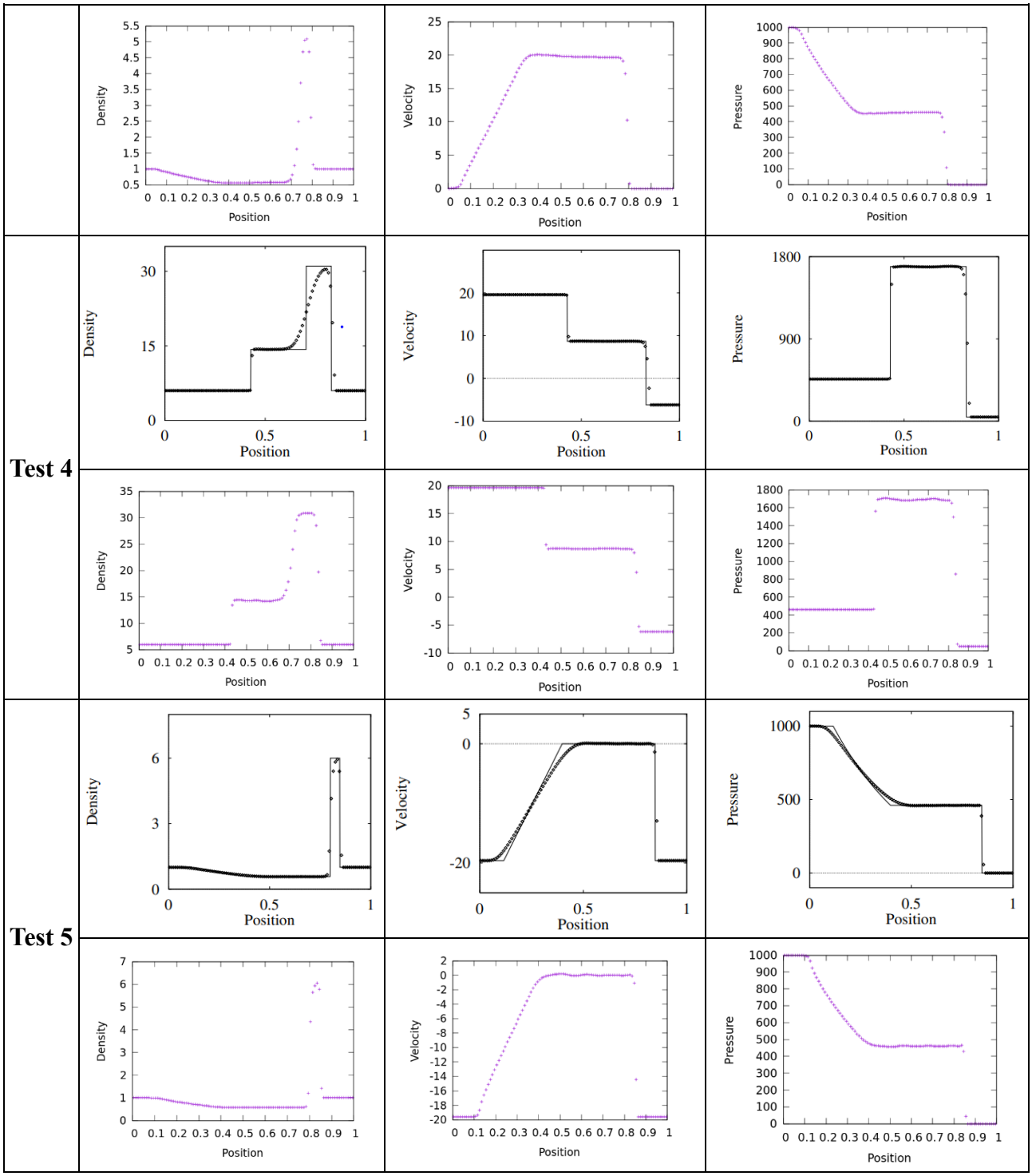
All five tests feature: a numerical solution and an exact inline continuous solution provided by the text [1] (pag. 372) and a numerical solution provided by the code used with active MUSCL discretization:

Test1



Test 3





9. Observations testing method of Roe with MUSCL discretization

In the case of Test 2 the negative pressure error is obtained and in this case the book is also not reported any results. In this test we have two waves of expansion and "vacuum", this error is due to the poor ability of the Roe method to distinguish between impact and expansion as noted earlier.

The differences observed between these results and those of the Roe method with first order discretization are a trend of the numerical solution more similar to the exact one in the slope changes given the polynomial and no longer linear approximation provided by the MUSCL methods.

10. Metodi Flux-Vector-Splitting

The method of our interest is the one proposed by Liou and Steffen, from which it takes its name and belongs to the class of FVS methods. This class of methods is used to solve systems of hyperbolic equations explicitly and through two important properties: the first consists in solving an approximate Riemann problem and the second in being able to exploit an upwind scheme that allows a resolution of the equations in the direction of propagation of information characteristic of the physics of the problem under consideration.

Taking a linear hyperbolic system written in conservative form:

$$\mathbf{U}_x + \mathbf{A}\mathbf{U}_y = \mathbf{0}$$

It is possible to decompose the Jacobian matrix of transformation into a positive contribution $\boldsymbol{A}^+$(characterized by positive and zero eigenvalues) and a negative one $\boldsymbol{A}^-$ (characterized by negative and zero eigenvalues):

$$\mathbf{A} = \mathbf{A}^+ + \mathbf{A}^-$$

Where by diagonalization:

$$\mathbf{A}^+ = \mathbf{K}\boldsymbol{\Lambda}^+\mathbf{K}^{-1}\,, \quad \mathbf{A}^- = \mathbf{K}\boldsymbol{\Lambda}^-\mathbf{K}^{-1}$$

So you can solve the problem with explicit upwind scheme both for the part of information that propagates forward and for the part of information that propagates backwards:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta x}{\Delta y}\mathbf{A}^+[\mathbf{U}_i^n - \mathbf{U}_{i-1}^n] - \frac{\Delta x}{\Delta y}\mathbf{A}^-[\mathbf{U}_{i+1}^n - \mathbf{U}_i^n]$$

For a non-linear hyperbolic system written in conservative form, splitting of the A matrix results in a natural flow vector splitting $\boldsymbol{F}$:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{0}$$

$$\mathbf{F} = \mathbf{A}\mathbf{U}$$

$$\mathbf{F} = \mathbf{F}^+ + \mathbf{F}^-$$

All this is possible given the hyperbolic character of the system under consideration. An additional property that must be satisfied in this case is that of homogeneity:

$$\mathbf{F}(\mathbf{U}) = \mathbf{A}(\mathbf{U})\mathbf{U}$$

This results in an appropriate linearization and subsequent splitting of the Jacobian matrix that varies according to the FVS method applied. Finally, the numerical approach of the FVS methods consists in an explicit and conservative scheme:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}[\mathbf{F}_{i+\frac{1}{2}} - \mathbf{F}_{i-\frac{1}{2}}]$$

Where the numerical flow to the interface $i+\frac{1}{2}$ consists of two contributions, or the forward component coming from the left cell F_i^+ and the backward component from the right cell F_{i+1}^-.

$$\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}_i^+(\mathbf{U}_i^n) + \mathbf{F}_{i+1}^-(\mathbf{U}_{i+1}^n)$$

In the specific case of the Liou-Steffen method applied to the Euler equations (hereinafter in the three-dimensional form), the following procedure is adopted to characterize the numerical flows: it starts from the problem written in conservative form and separates the convective component of the flow $\boldsymbol{F^{(c)}}$ from the pressure component $\boldsymbol{F^{(p)}}$:

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \end{bmatrix} = \begin{bmatrix} \rho u \\ \rho u^2 \\ \rho uv \\ \rho uw \\ \rho uH \end{bmatrix} + \begin{bmatrix} 0 \\ p \\ 0 \\ 0 \\ 0 \end{bmatrix} \equiv \mathbf{F}^{(c)} + \mathbf{F}^{(p)}$$

By introducing Mach number and total enthalpy as follows, you can express the convective component in a new way:

$$M = \frac{u}{a}, \qquad H = \frac{E+p}{\rho}$$

$$\mathbf{F}^{(c)} = M \begin{bmatrix} \rho a \\ \rho au \\ \rho av \\ \rho aw \\ \rho aH \end{bmatrix} \equiv M\hat{\mathbf{F}}^{(c)}$$

Now, the numerical flow at the interface can be characterized as follows:

$$\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}^{(c)}_{i+\frac{1}{2}} + \mathbf{F}^{(p)}_{i+\frac{1}{2}}$$

$$\mathbf{F}^{(c)}_{i+\frac{1}{2}} = M_{i+\frac{1}{2}} \left[\hat{\mathbf{F}}^{(c)}\right]_{i+\frac{1}{2}}$$

by defining:

$$[\bullet]_{i+\frac{1}{2}} = \begin{cases} [\bullet]_i & \text{if } M_{i+\frac{1}{2}} \geq 0\,, \\ [\bullet]_{i+1} & \text{if } M_{i+\frac{1}{2}} \leq 0\,. \end{cases}$$

With the previous one you can notice the first property of the method, that is to decide the evolution of the solution in front and back depending on the value of the number of Mach evaluated to the interface $M_{i+1/2}$. For this reason the Method is also called Advection Upstream Splitting Method (AUSM).

The splitting proposed by the diagram and therefore, the linearization of the problem is addressed by the evaluation of the Mach number and the pressure at the interface:

$$M_{i+\frac{1}{2}} = M_i^+ + M_{i+1}^-$$

$$M^{\pm} = \begin{cases} \pm\frac{1}{4}(M\pm 1)^2 & \text{if } \mid M \mid \leq 1 \\ \frac{1}{2}(M\pm \mid M \mid) & \text{if } \mid M \mid > 1 \end{cases}$$

$$p_{i+\frac{1}{2}} = p_i^+ + p_{i+1}^-$$

$$p^{\pm} = \begin{cases} \frac{1}{2}p(1\pm M) & \text{if } \mid M \mid \leq 1 \\ \frac{1}{2}p\frac{(M\pm\mid M\mid)}{M} & \text{if } \mid M \mid > 1 \end{cases}$$

So the code: first evaluate the absolute value of the Mach number locally $|M|$ to understand if you are in subsonic or supersonic condition and then based on the value assumed by the Mach number at the interface $M_{i+1/2}$ evaluate the direction in which to evolve the solution in the calculation domain if forward or backward.

11. Code implemented for the Liou-Steffen method

The test consists in implementing the Liou-Steffen method within the code "nozzle_w. f" as shown below in the new code "*eus1d_FVS2.f*":

```
      program eus1d_FVS2
c
c  This program solves the 1D Euler equations for a gamma-law gas
c  global dt evaluation per time-accurate solution and wall condition at left and right boundaries
c  This is done through Flux-Vector-Splitting method, in the specific
c  case with the Liou-Steffen scheme.
c
c     # Variables declaration
      implicit double precision (a-h,o-z)
      parameter (maxmx = 1000)
      parameter (meqn = 3)
      parameter (mbc = 2)
c
      dimension q(1-mbc:maxmx+mbc, meqn)         !conservative vectors
      dimension qold(1-mbc:maxmx+mbc, meqn)      !old conservative vec.
      dimension rho(1-mbc:maxmx+mbc)             !density vector
      dimension u(1-mbc:maxmx+mbc)               !velocity vector
      dimension pr(1-mbc:maxmx+mbc)              !pressure vector
      dimension a(1-mbc:maxmx+mbc)               !sound speed vector
      dimension dt(1-mbc:maxmx+mbc)
      dimension dt1(1-mbc:maxmx+mbc)
      dimension dt2(1-mbc:maxmx+mbc)
      dimension qpr(1-mbc:maxmx+mbc, meqn)       !primitive vectors
      dimension amai(1-mbc:maxmx+mbc)            !cell-interface MACH
      dimension pri(1-mbc:maxmx+mbc)             !cell-interface Pr.
      dimension fluxt(1-mbc:maxmx+mbc, meqn)     !cell-interface Flux
      dimension x(1-mbc:maxmx+mbc)               !cell-centered x-axis
```

```
      dimension area(1-mbc:maxmx+mbc)
      dimension alph(4)                    !RK coefficients vector
      common /param/  gamma,gamma1
c
c
c     # Runge-Kutta parameters
      irkmax = 2
      alph(1) = 0.42
      alph(2) = 1.
c     irkmax = 1
c     alph(1) = 1.
c
c     # Input section
      write(6,*) 'input position of diaphram'
      read(5,*) diaph
      write(6,*) 'input gamma'
      read(5,*) gamma
      gamma1 = gamma - 1.d0
      write(6,*) 'input dx'
      read(5,*) dx
      write(6,*) 'input cfl, itmax, nstmp'      !itmax: max iteration n
      read(5,*) cfl,itmax,nstmp                 !nstmp: out stamp ratio
      write(6,*) 'input tmax'                   !tmax: max time allowed
      read(5,*) tout
c
c     # Set grid and initial conditions
c
c     # Domain is  0 <= x <= 1:
      ax = 0.d0
      bx = 1.d0
      mx = (bx-ax + 1d-12)/dx
```

```
c
c     # Check for errors in data:
c
      if (mx .gt. maxmx) then
        write(6,*) "attento al dimensionamento"
        call exit
      endif
c
      do 10 i=1-mbc,mx+mbc
        x(i) = ax + (i-0.5d0) * dx
        write(20,1020) x(i)
 1020    format(e16.6)
   10   continue
c
c     # Define the area function
      do i=1-mbc,mx+mbc
        area(i) = 1.0d0
      end do
c
c     # Initial condition
      call ic(maxmx,meqn,mbc,mx,x,dx,q,diaph)
c
      call out1eu(maxmx,meqn,mbc,mx,x,q,area)
c
      t = 0.d0
c
c     # Main loop
      do 100 n=1,itmax
c
        dtgl = 1.d10
c
```

```
c       # Extend data from grid to wall boundary cells
        call bcw(maxmx,meqn,mbc,mx,q)
c
c       # Store the solutions in terms of conservative variables
        do m=1,meqn
          do i=1-mbc,mx+mbc
            qold(i,m) = q(i,m)
          end do
        end do
c
c       # Runge-Kutta
        do irk=1,irkmax    !RK
c
c       # Compute primitive variables (rho, u, p) and CFL condition
         do i=1-mbc,mx+mbc
          qpr(i,1) = q(i,1)
          qpr(i,2) = q(i,2)/q(i,1)
          qpr(i,3) = gamma1*(q(i,3)-0.5d0*q(i,2)*q(i,2)/q(i,1))
c          # Rename of primitive variables in different vectors
          rho(i)=qpr(i,1)
          u(i)=qpr(i,2)
          pr(i)=qpr(i,3)
          a(i)=dsqrt(gamma*qpr(i,3)/qpr(i,1))
c          # Check of non-physical solutions
          if(qpr(i,3) .le. 0.d0) then
            write(6,*) "Attento: pressione negativa in i = ",i,n
            write(6,*) "pressione = ", qpr(i,3)
            call exit
          end if
c          # Evaluation of global dt
          if(irk.eq.1) then
```

```
          dt1(i) = cfl*dx/dabs(u(i)+a(i))
          dt2(i) = cfl*dx/dabs(u(i)-a(i))
          dtgl = min(dt1(i),dt2(i),dtgl)
        end if
       end do
c
c      # Lius Steffen scheme
       do i=0,mx+1
c
        ama = u(i)/a(i)                ! local Mach number
c
c      # RECOSTRUCTION OF LEFT CELL STATE AND RIGHT CELL STATE
c
c       # Splitting of the interface Mach number and interface Pr.
        if(dabs(ama).le.1.0d0)then
c             SUBSONIC CASE
c             right/left Mach component
              amar = +0.25d0*(ama+1.d0)**2
              amal = -0.25d0*(ama-1.d0)**2
c             right/left Pressure component
              prr = 0.5d0*pr(i)*(1.d0+ama)
              prl = 0.5d0*pr(i)*(1.d0-ama)
        else
c             SUPERSONIC CASE
c             right/left Mach component
              amar = 0.5d0*(ama+dabs(ama))
              amal = 0.5d0*(ama-dabs(ama))
c             right/left Pressure component
              prr = 0.5d0*pr(i)*(ama+dabs(ama))/ama
              prl = 0.5d0*pr(i)*(ama-dabs(ama))/ama
        end if
```

```
c
c          # Compute interface Mach number and interface pressure
c           M(i+1/2)=M(i)+ + M(i+1)-
c           p(i+1/2)=p(i)+ + p(i+1)-
          if(i.ge.0.and.i.le.mx)then
               amai(i) = amar
               pri(i) = prr
          end if
c
          if(i.ge.1)then
               amai(i-1) = amai(i-1) + amal
               pri(i-1) = pri(i-1) + prl
          end if
        end do
c
c       # COMPUTE INTERFACE NUMERICAL FLUXES
c
        do i=0,mx
          if(amai(i).ge.0.0)then
c              advection from left to right (evalueted in i)
               ene = 0.5d0*u(i)*u(i)+a(i)*a(i)/(gamma1)
c
               fluxt(i,1) = amai(i)*rho(i)*a(i)
               fluxt(i,2) = amai(i)*rho(i)*a(i)*u(i)+pri(i)
               fluxt(i,3) = amai(i)*rho(i)*a(i)*ene
c
          else
c              advection from right to left (evaluated in i+1)
               ene = 0.5d0*u(i+1)*u(i+1)+a(i+1)*a(i+1)/(gamma1)
c
               fluxt(i,1) = amai(i)*rho(i+1)*a(i+1)
```

```
            fluxt(i,2) = amai(i)*rho(i+1)*a(i+1)*u(i+1)+pri(i)
            fluxt(i,3) = amai(i)*rho(i+1)*a(i+1)*ene
c
          end if
        end do
c
c       # Update the solution
        do m=1,meqn
          do i=1,mx
            q(i,m) = qold(i,m) - alph(irk) *
     .               dtgl * (fluxt(i,m)-fluxt(i-1,m)) / dx
          end do
        end do
c
      end do     !RK
c
c       # Update the time
      t = t + dtgl
c
      if(n/nstmp*nstmp.eq.n) then
        write(6,*) n,t,tout
      end if
c
      if(t.ge.tout) go to 101
c
  100   continue
c
  101   continue

      write(6,*) n,t,tout
      call out1eu(maxmx,meqn,mbc,mx,x,q,area)
```

```
c
      stop
      end
c
c =====================================================
      subroutine ic(maxmx,meqn,mbc,mx,x,dx,q,diaph)
c =====================================================
c
c     # Set initial conditions for q.
c
c     # quasi 1d nozzle
c
      implicit double precision (a-h,o-z)
      dimension q(1-mbc:maxmx+mbc, meqn)
      dimension x(1-mbc:maxmx+mbc)
      common /param/  gamma,gamma1
c
c
      write(6,*) 'input left state:p,rho,u'
      read(5,*) pL,rhoL,uL
      write(6,*) "valori L",pL,rhoL,uL
      write(6,*) 'input right state:p,rho,u '
      read(5,*) pR,rhoR,uR
      write(6,*) "Valori R",pR,rhoR,uR

c      write(6,*) pL,rhoL,uL
c      write(6,*) pR,rhoR,uR

c      set the conservative value from wall boundary cells
c      rho, rho*u, p/rho+0.5*rho*u^2
      do 150 i=1-mbc,mx+mbc
```

```
        if(x(i).le.diaph) then
          q(i,1) = rhoL
          q(i,2) = rhoL*uL
          q(i,3) = pL/gamma1 + 0.5d0*rhoL*uL**2
        else
          q(i,1) = rhoR
          q(i,2) = rhoR*uR
          q(i,3) = pR/gamma1 + 0.5d0*rhoR*uR**2
        end if
  150  continue
c
      return
      end
c
c =========================================================
      subroutine out1eu(maxmx,meqn,mbc,mx,x,q,area)
c =========================================================
c
c     # Output the results in primitive variables
c     # (density, velocity, pressure) for the Euler equations.
c
      implicit double precision (a-h,o-z)
      dimension q(1-mbc:maxmx+mbc, meqn)
      dimension x(1-mbc:maxmx+mbc)
      dimension area(1-mbc:maxmx+mbc)
      character*11 fname
      common /param/  gamma,gamma1
c
c
        fname = "outfile"
c       open(unit=100,file=fname,status="unknown",
```

```
c    .      form="formatted")
       open(10,file=fname,status="unknown",
     .      form="formatted")
       fname = "out"
       open(11,file=fname,status="unknown",
     .      form="formatted")



 1001 format(e16.8, 2i5, /)
     write(10,*) 'VARIABLES = "x", "A", "rho", "u", "p", "M"'
     write(10,*) 'ZONE F=POINT, I=',mx
     do 20 i=1,mx
        do m=1,meqn
c          # exponents with more than 2 digits cause problems reading
c          # into matlab... reset tiny values to zero:
           if (dabs(q(i,m)) .lt. 1d-99) q(i,m) = 0.d0
           enddo
        rho = q(i,1)
        pr = gamma1*(q(i,3) - 0.5d0*q(i,2)**2 / rho)
        u = q(i,2)/rho
        c = dsqrt(gamma*pr/rho)
        ama = u/c
c ORIG    write(10,1002) rho,u,pr
        write(10,*) x(i),area(i),rho,u,pr,ama
        write(11,*) x(i),rho,u,pr
 1002    format(3e14.6)
 20      continue
c
     close(10)
     close(11)
     return
```

```
      end
c
c =====================================================
      subroutine bcw(maxmx,meqn,mbc,mx,q)
c =====================================================
c
c     # Extend the data from the computational region
c     #      i = 1, 2, ..., mx2
c     # to the virtual cells outside the region, with
c     #      i = 1-ibc  and   i = mx+ibc   for ibc=1,...,mbc
c
      implicit double precision (a-h,o-z)
      dimension q(1-mbc:maxmx+mbc, meqn)
      common /param/  gamma,gamma1
c
c     # left boundary open wall
c     # right boundary open wall
c     ------------------------------------------
c
C supersonic inlet condition at left boundary
      do 10 ibc=1,mbc
         q(1-ibc,1)=q(1,1)
         q(1-ibc,2)=q(1,2)
         q(1-ibc,3)=q(1,3)
   10    continue
c
C supersonic outlet condition at right boundary/extrapolation condition
       do 20 ibc=1,mbc
         q(mx+ibc,1)=q(mx,1)
         q(mx+ibc,2)=q(mx,2)
         q(mx+ibc,3)=q(mx,3)
```

```
  20      continue
c
      return
      end
c
```

12. Numerical test method of Liou-Steffen

Data consist of two constant initial states (ρ_L, u_L, p_L) e (ρ_R, u_R, p_R), separated by a break in place $x = x_0$. All five tests present a unit spatial domino divided into one hundred cells.

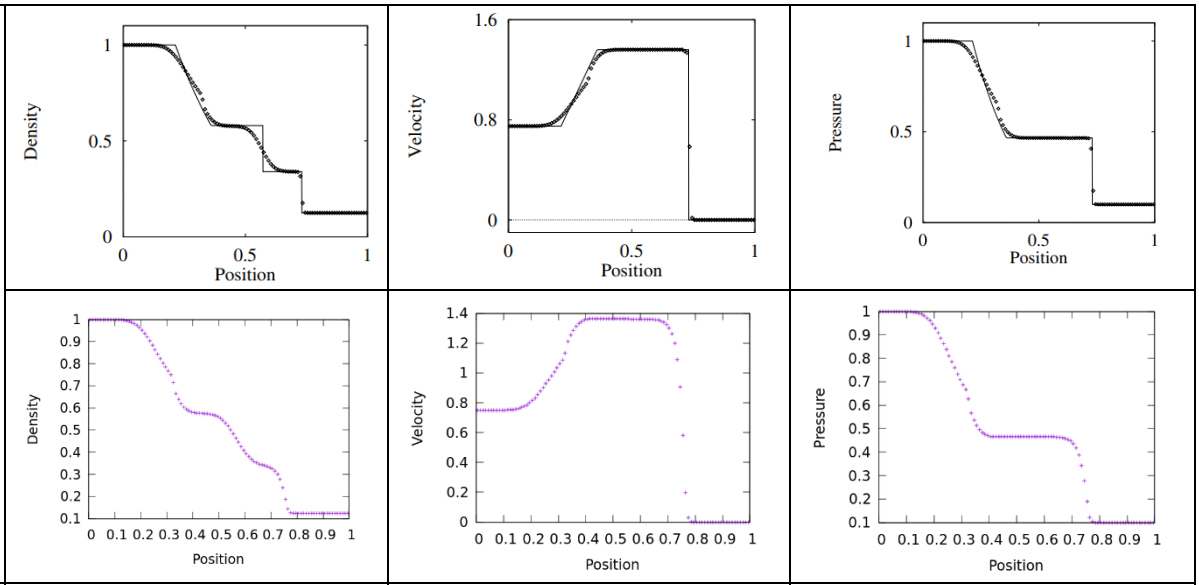
Test	x_0	T	ρ_L	u_L	p_L	ρ_R	u_R	p_R
1	0.3	0.2	1.0	0.70.05	1.0	0.125	0.0	0.1
2	0.5	0.25	1.0	-2.0	0.4	1.0	2.0	0.4
3	0.5	0.012	1.0	0.0	1000.0	1.0	0.0	0.01
4	0.4	0.035	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950
5	0.8	0.012	1.0	-19.59745	1000.0	1.0	-19.59745	0.01

Input data common to all tests, and characteristic of the code used are:

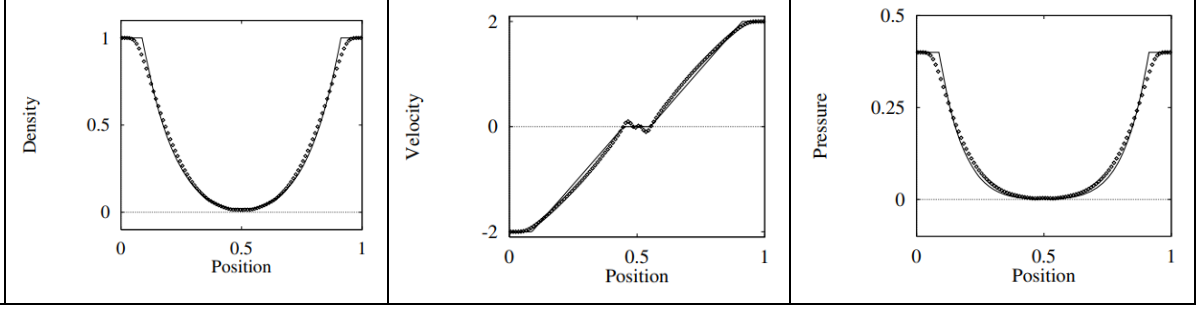
Input position of diaphram:	x_0
Input gamma:	1.4
Input dx:	0.01
Input cfl, itmax, nstmp:	0.9 10000 10
Input tmax:	T

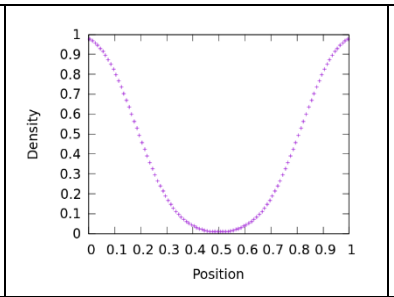
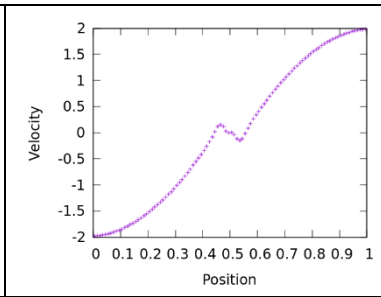
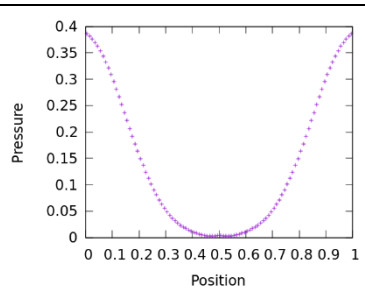
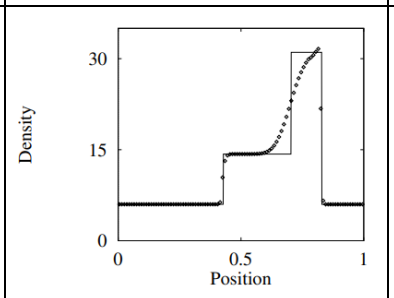
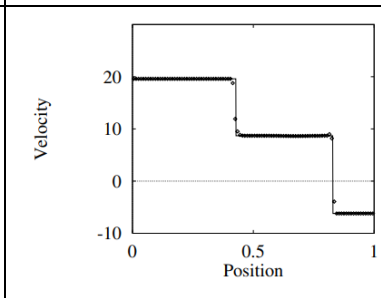
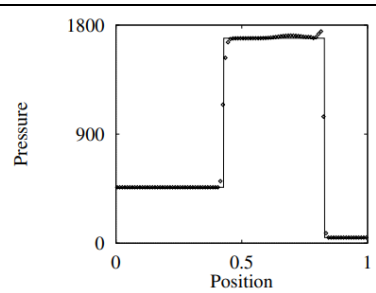
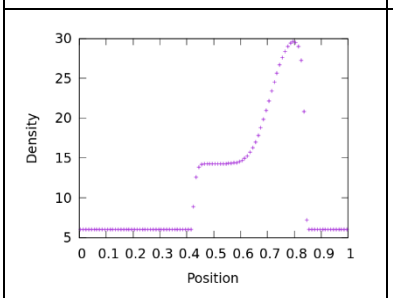
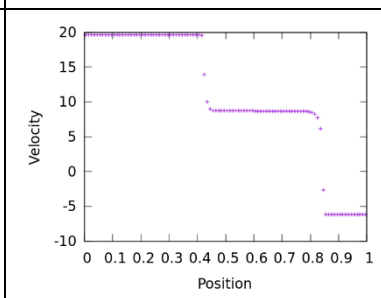
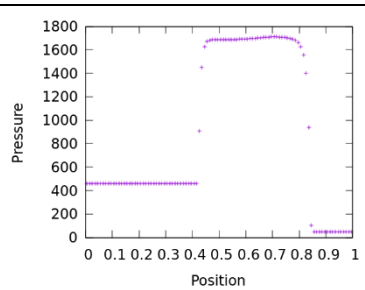
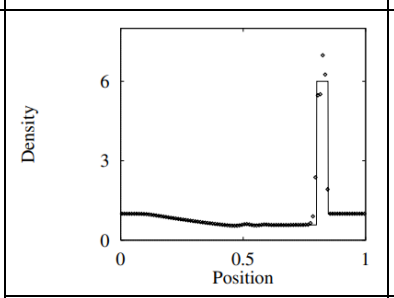
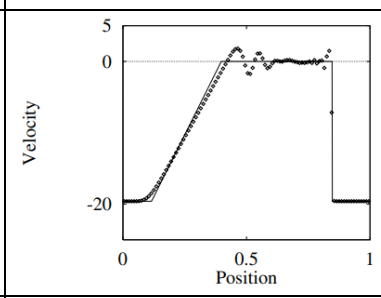
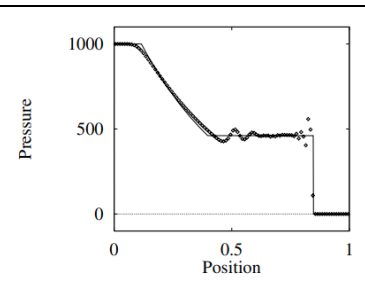
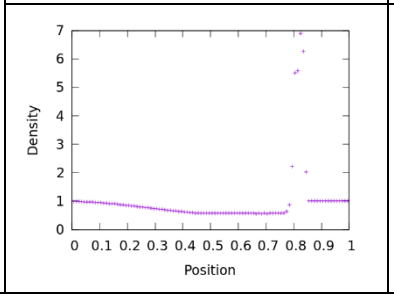
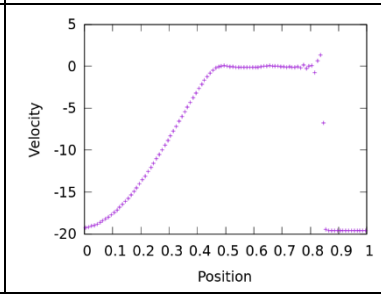
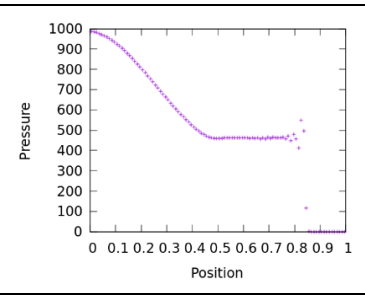
All five tests feature: a numerical solution and an exact inline continuous solution provided by the text [1] (pag. 280) and a numerical solution provided by the code used with active MUSCL discretization:

Test1



Test 2



			
Test 4			
			
Test 5			
			

13. Observations test method of Liou-Steffen

Test 3 has no results on the reference textbook and even with the implemented code you get nothing but a negative pressure error (even lowering the Courant number as suggested in the book).

Test 5 always exhibits non-physical oscillations at discontinuity but smoother elsewhere than in the book in the speed and pressure graphs. The previous result is due to the two-step Runge-Kutta method, kept active for all tests, which leads to an increase in temporal accuracy of the solution.

14. Conclusions

FVS schemes are simpler and more efficient than Godunov schemes (with the exact Riemann problem solver). Having to solve the problem under consideration not in an exact way but linearized and adopting a discretization upwind, the salient characteristics of the physical phenomena are incorporated in the schemes themselves. The solution of the problem is then made to evolve in the direction of propagation of information characteristic of the problem bringing many advantages at the level of computational efficiency.

The method examined such as that of Liou-Steffen also has the computational advantage of being automatically upwind with respect to the direction of propagation of information, without therefore needing to distinguish the numerical flow between the various flow cases: subsonic, supersonic to the right or supersonic to the left.

In general, however, the results provided have disadvantages from the physical point of view: they are more accurate in the method of ROE (always with approximate solver of the Riemann problem) suffice to see the solution near the discontinuity that in the results of Liou-Steffen show non physical oscillations. The results of the Roe method are also more robust as the Liou-Steffen method fails in the case of Test 3 in extreme situations with very high Mach number.

In conclusion, the FVS schemes are the most efficient among the proposed methods but also the least accurate among them in accordance with what the reference textbook says [1].

15. Bibliography

[1] Eleuterio F. Toro - Riemann Solvers and Numerical Methods for Fluid Dynamics;

[2] P. De Palma, G. Pascazio – Appunti di Fluidodinamica Numerica.