Politecnico
di Bari

RELAZIONE TEMA D'ANNO IN FLUIDODINAMICA COMPUTAZIONALE

Metodi Flux Vector Splitting

Fracchiolla Beniamino.

INDICE

1. Abstract

Lo scopo del tema d'anno qui mostrato è quello di ottenere l'andamento di grandezze primitive quali densità, velocità e pressione di un flusso d'aria (modellizzato come gas perfetto) all'interno di un tubo d'urto. Nello specifico, il problema da risolvere tramite procedura computazionale sarà il problema di Riemann al sistema di Eulero. Nel caso del problema di Riemann abbiamo condizioni iniziali costanti a destra e sinistra del tubo d'urto e una discontinuità nel diaframma dello stesso. Le equazioni del sistema di Eulero, invece, caratterizzano il comportamento di flussi non viscosi e senza conduzione termica, il tutto si traduce in un sistema non-lineare.

La procedura computazionale adottata consiste nello svolgere e confrontare i risultati di 5 Test proposti dal testo "*Riemann Solvers and Numerical Methods for Fluid Dynamics*" con tre codici differenti:

1. Il codice "*nozzle_w.f*" che risolve le equazioni di Eulero quasi-1d mediante metodo di ROE con discretizzazione al primo ordine;
2. Il codice "*nozzle_w.f*" ma questa volta con discretizzazione fornita dallo schema MUSCL;
3. Implementare un codice con diverso metodo, ovvero uno schema Flux-Vector-Splitting, nello specifico il metodo di Liou-Steffen.

I test consistono nell'utilizzare condizioni al contorno tali da produrre:

- Test 1: una onda d'urto destra, un'onda di contatto che si muove verso destra e una zona di espansione sinistra;
- Test 2: due zone di espansione simmetriche e una zona di contatto a velocità zero compresa tra le due;
- Test 3: una forte onda d'urto destra (Mach=198), superficie di contatto e zona di espansione a sinistra;
- Test 4: due forti onde d'urto che viaggiano l'una verso l'altra producendo tre discontinuità;
- Test 5: come il test 3 ma con una velocità di background negativa in modo da ottenere una discontinuità stazionaria.

A valle dei seguenti Test seguiranno commenti circa i risultati ottenuti ed infine nelle conclusioni un commento che mette a paragone i vari metodi.

2. Metodo di Roe

Il metodo di Roe è utilizzato per la soluzione di sistemi di equazioni differenziali iperbolici quando invece di usare un solutore esatto del problema di Riemann si ricorre ad un solutore approssimato per ridurre il costo computazionale del problema.

L'approssimazione proposta consiste nel linearizzare il problema introducendo, sotto determinate ipotesi, *la linearizzazione di Roe* $\boldsymbol{A}(\boldsymbol{u},\boldsymbol{v})$. Questa consiste nel linearizzare opportunamente il Jacobiano della trasformazione ottenendo così il problema di Riemann linearizzato:

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

La soluzione esatta del problema di Riemann per un sistema lineare consiste nella risoluzione di due integrali in questo caso, invece, essendo il Jacobiano della trasformazione non costante, lo si valuta all'interfaccia di due celle. Ciò comporta una *linearizzazione locale* dello stesso:

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

Questo permette di esprimere il flusso numerico come:

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

Dove $(\boldsymbol{v} - \boldsymbol{u})$ può esser espressa come la differenza delle onde:

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

A questo punto, il flusso numerico di 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 tutto ciò non abbiamo ancora trovato la forma della matrice Jacobiana $\boldsymbol{A}$ linearizzata. L'applicazione di tale metodo per la soluzione delle equazioni di Eulero consiste nel valutare la *matrice di Roe*:

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

Dove $M(u,v)$ è un'opportuna media tra gli stati $\boldsymbol{u}$ e $\boldsymbol{v}$, l'operatore M verrà determinato mediante un cambio di variabile o *parametro vettore*:

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

Nel caso delle equazioni di Eulero $\boldsymbol{w}$ è definito come:

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

Mentre la matrice $\boldsymbol{A}$ e gli autovettori destri associati:

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

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

Arrivati a questo punto, il problema non è ancora linearizzato, bisognerà ricorrere ai *valori medi di Roe*

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

Questi faranno dipendere per prime le grandezze primitive alle condizioni locali presenti a destra e sinistra della cella:

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

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

Successivamente tali medie faranno rendere matrice jacobiana e autovettori destri funzione dalle condizioni locali presenti a destra e sinistra della cella:

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

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

Rendendo, infine, anche il flusso numerico dipendente da ciò che succede localmente a destra e sinistra della cella $\boldsymbol{g_{ROE}} = f(\boldsymbol{u_R}, \boldsymbol{u_L}, \overline{\boldsymbol{u}}) = f(\boldsymbol{\omega_R}, \boldsymbol{\omega_L})$:

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

3. Codice implementato per il metodo di Roe

La procedura adottata consiste nell'utilizzare il codice "*nozzle_w.f*". Rispetto al codice originale sono state implementate, nel codice "*eus1d_ROE.f*", le condizioni al contorno di estrapolazione di tipo trasmissivo nella 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)
```

Per risolvere i test 4 e 5, a causa delle velocità negative, è stato necessario modificare l'algoritmo che ricava il *passo temporale globale* nel rispetto della condizione CFL, questo per tener conto dei casi in cui l'autovalore maggiore possa essere $|(u-a)|$:

```
            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)
```

Infine, per poter ottenere una discretizzazione al primo ordine sono state commentate le porzioni di codice relative alla ricostruzione MUSCL:

```
!         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

I dati sono costituiti da due stati iniziali costanti (ρ_L, u_L, p_L) e (ρ_R, u_R, p_R), separati da una discontinuità in posizione $x = x_0$. Tutti e cinque i test presentano un domino spaziale unitario suddiviso in cento celle.

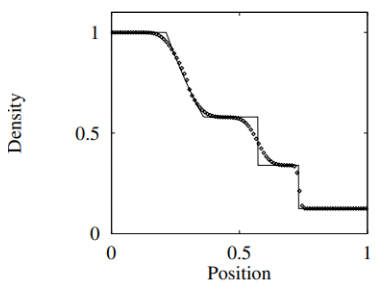
Test	x_0	T	ρ_L	u_L	p_L	ρ_R	u_R	p_R
1	0.3	0.2	1.0	0.70.05	1.0	0.125	0.0	0.1
2	0.5	0.25	1.0	-2.0	0.4	1.0	2.0	0.4
3	0.5	0.012	1.0	0.0	1000.0	1.0	0.0	0.01
4	0.4	0.035	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950
5	0.8	0.012	1.0	-19.59745	1000.0	1.0	-19.59745	0.01

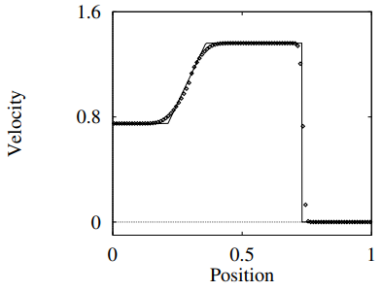
I dati di input comuni a tutti i test, e caratteristici del codice utilizzato sono:

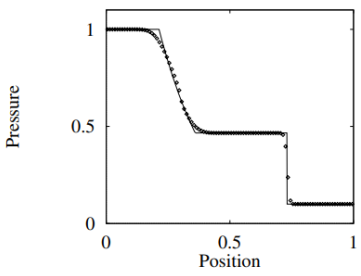
Input gamma:	1.4
Input dx:	0.01
Input cfl, itmp, itsmp:	0.9 10000 10
MUSCL coefficient:	-1
Input epsilon:	1.e-10

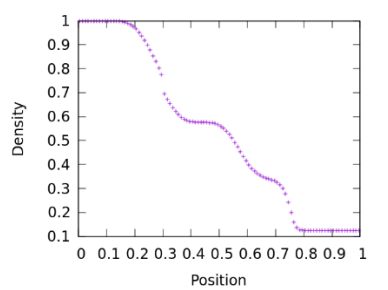
Tutti e cinque i test presentano: una soluzione numerica e una esatta in linea continua fornita dal testo [1] (pag. 372) e una soluzione numerica fornita dal codice utilizzato:

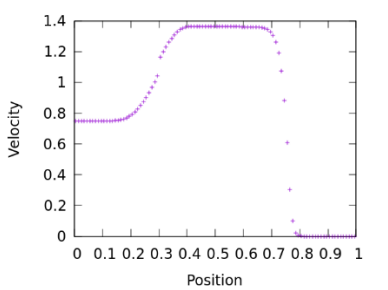
Test1

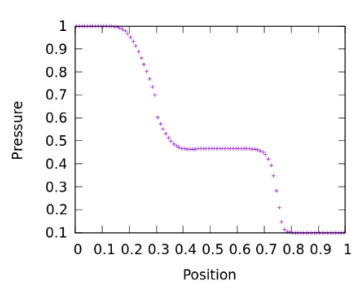




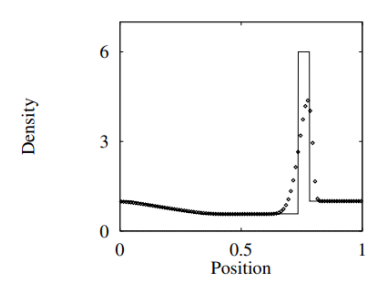


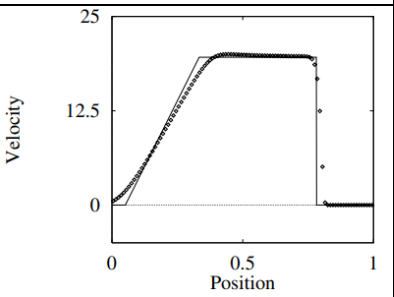


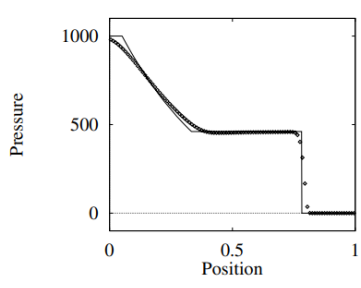


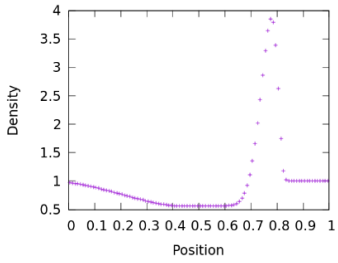
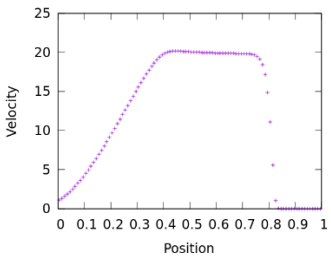
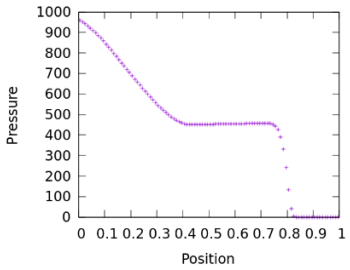
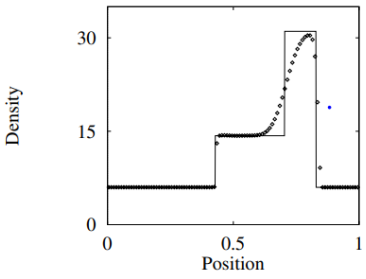
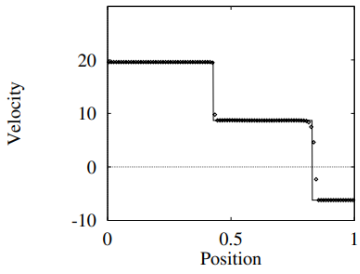
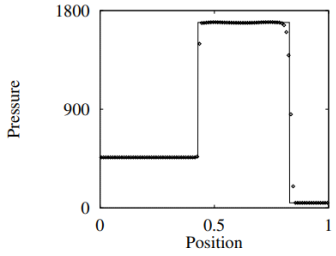
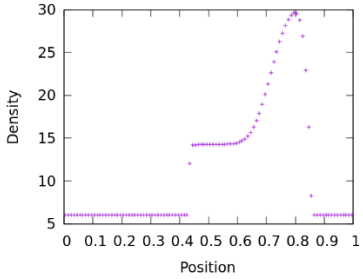
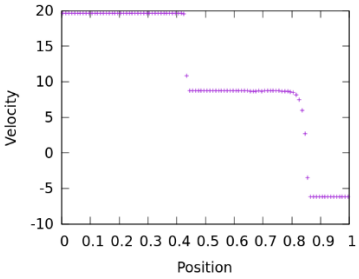
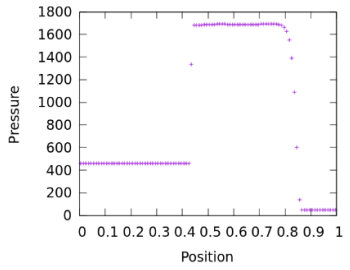
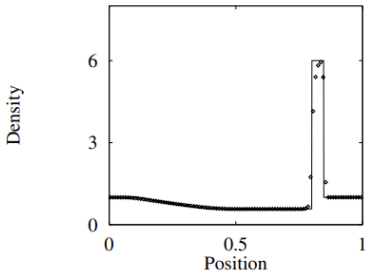
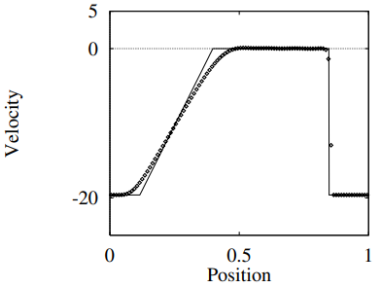
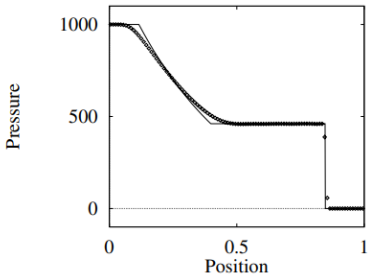
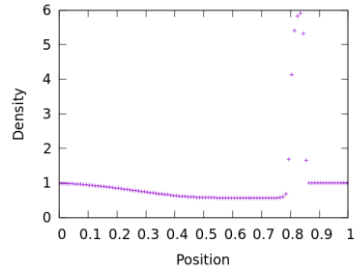
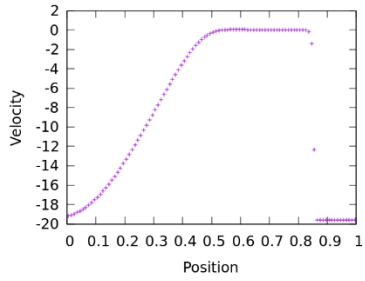
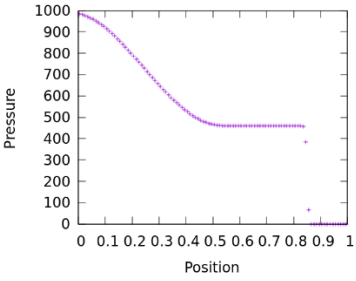


Test 3







			
Test 4			
			
Test 5			
			

5. Osservazioni test metodo di Roe

Nel caso del Test 2 si ottiene l'errore di pressione negativa e in tal caso anche sul libro non è riportato alcun risultato. In questo test abbiamo due onde di espansione e "vuoto", tale errore è dovuto alla scarsa capacità del metodo di Roe di distinguere tra urto ed espansione data la linearizzazione locale del problema adottata da tale metodo.

6. Metodo di Roe con discretizzazione a schema MUSCL

Poiché i solutori di Riemann (esatti e non) sono accurati al primo ordine nello spazio in quanto i flussi sono funzione delle variabili nelle celle adiacenti all'interfaccia, gli schemi di discretizzazione MUSCL permettono di aumentare tale *accuratezza spaziale*. Tutto ciò è possibile tramite ricostruzione della variabile dipendente nello spazio su più celle ed usando quest'ultima per valutare i flussi all'interfaccia.

Nel caso del metodo di Roe valuto i flussi al prim'ordine mediante ricostruzione delle variabili dipendenti tenendo conto solo del valore che esse assumono nelle celle u_j e u_{j+1}:

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

L'accuratezza superiore si ottiene utilizzando il valore all'interfaccia di tali celle:

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

E tali valori all'interfaccia possono essere valutati mediante espansione in serie di Taylor ampliando, in un certo senso, la molecola computazionale dello schema numerico:

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

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

7. Codice implementato per il metodo di Roe con discretizzazione MUSCL

La procedura adottata consiste nell'utilizzare il codice "*nozzle_w.f*". Rispetto al codice originale sono state implementate nel codice "*eus1d_ROE_MUSCL.f*" le condizioni al contorno di estrapolazione di tipo trasmissivo nella 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)
```

Per risolvere i test 4 e 5, a causa delle velocità negative, è stato necessario modificare l'algoritmo che ricava il *passo temporale globale* nel rispetto della condizione CFL, questo per tener conto dei casi in cui l'autovalore maggiore possa essere $|(u-a)|$:

```
          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)
```

Infine, per poter ottenere una discretizzazione con ordine superiore al primo sono state de-commentate le porzioni di codice relative alla ricostruzione MUSCL:

```
        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. Test numerici metodo di Roe con discretizzazione MUSCL

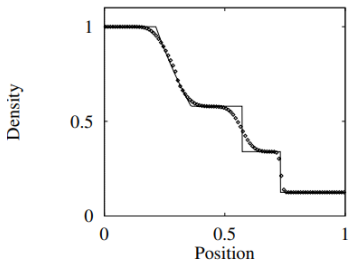
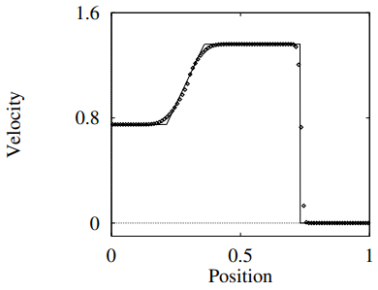
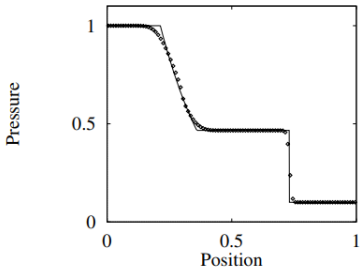
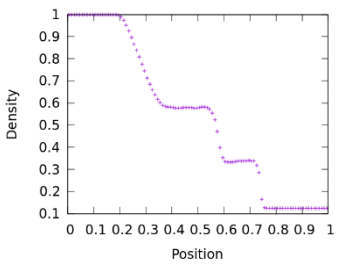
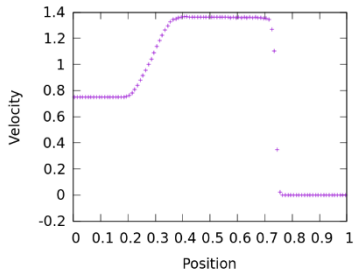
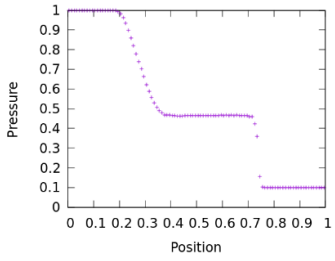
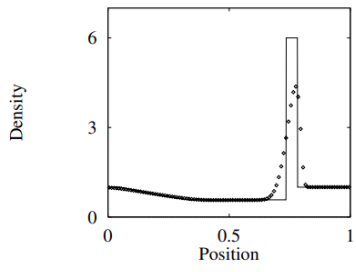
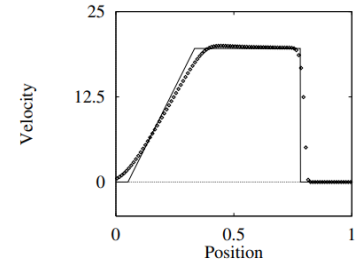
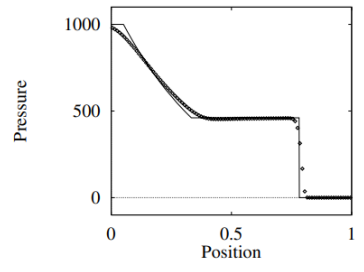
I dati sono costituiti da due stati iniziali costanti (ρ_L, u_L, p_L) e (ρ_R, u_R, p_R), separati da una discontinuità in posizione $x = x_0$. Tutti e cinque i test presentano un domino spaziale unitario suddiviso in cento celle.

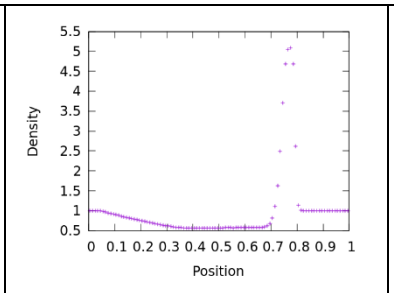
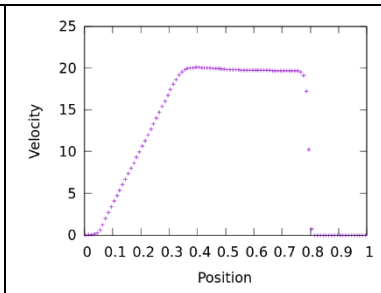
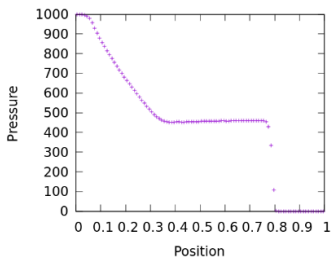
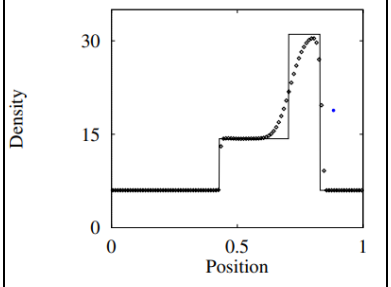
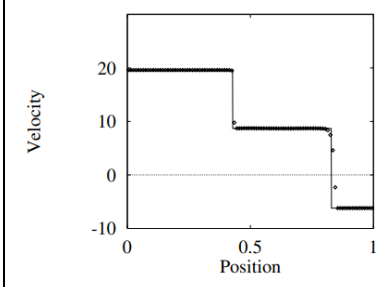
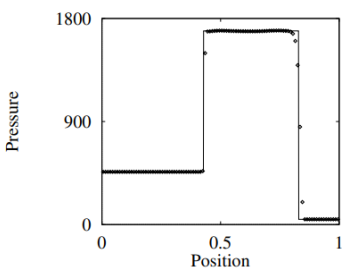
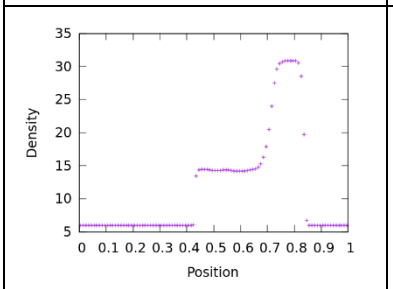
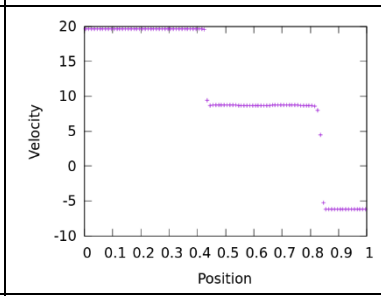
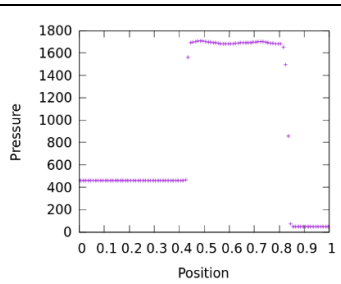
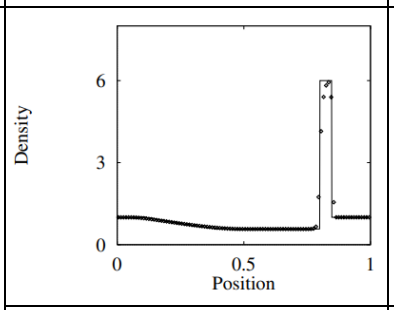
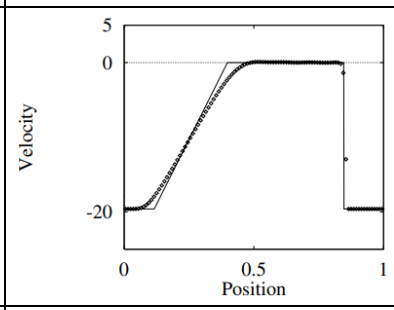
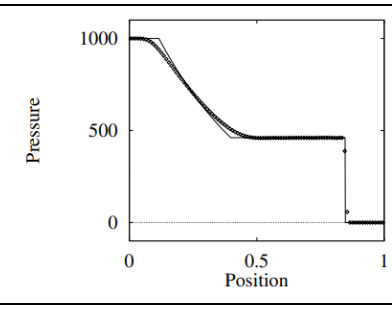
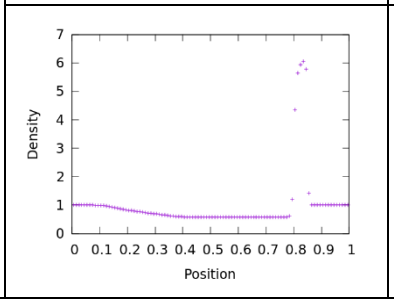
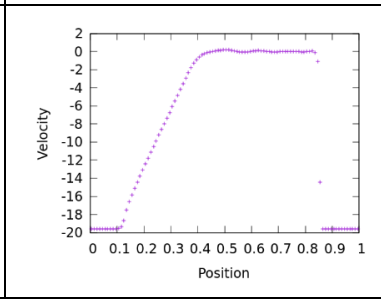
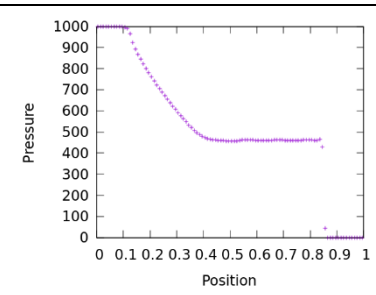
Test	x_0	T	ρ_L	u_L	p_L	ρ_R	u_R	p_R
1	0.3	0.2	1.0	0.70.05	1.0	0.125	0.0	0.1
2	0.5	0.25	1.0	-2.0	0.4	1.0	2.0	0.4
3	0.5	0.012	1.0	0.0	1000.0	1.0	0.0	0.01
4	0.4	0.035	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950
5	0.8	0.012	1.0	-19.59745	1000.0	1.0	-19.59745	0.01

I dati di input comuni a tutti i test, e propri del codice utilizzato sono:

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

Tutti e cinque i test presentano: una soluzione numerica e una esatta in linea continua fornita dal testo [1] (pag. 372) e una soluzione numerica fornita dal codice utilizzato con discretizzazione MUSCL attiva:

Test1			
			
Test 3			

			
Test 4			
			
Test 5			
			

9. Osservazioni test metodo di Roe con discretizzazione MUSCL

Nel caso del Test 2 si ottiene l'errore di pressione negativa e in tal caso anche sul libro non è riportato alcun risultato. In questo test abbiamo due onde di espansione e "vuoto", tale errore è dovuto alla scarsa capacità del metodo di Roe di distinguere tra urto ed espansione come già notato in precedenza.

Le differenze che si osservano tra questi risultati e quelli del metodo di Roe con discretizzazione al prim'ordine sono un andamento della soluzione numerica più simile a quella esatta nei cambi di pendenza data l'approssimazione polinomiale e non più lineare fornita dai metodi MUSCL.

10. Metodi Flux-Vector-Splitting

Il metodo di nostro interesse è quello proposto da Liou e Steffen, da cui prende il nome e appartiene alla classe dei metodi FVS. Tale classe di metodi viene impiegata per risolvere sistemi di equazioni iperbolici in modo *esplicito* ed attraverso due importanti proprietà: la prima consiste nel risolvere un problema di Riemann *approssimato* e la seconda nel poter sfruttare uno schema *upwind* che permette una risoluzione delle equazioni nella direzione di propagazione delle informazioni caratteristiche della fisica del problema in esame.

Prendendo un sistema iperbolico lineare scritto in forma conservativa:

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

Risulta possibile decomporre la matrice Jacobiana della trasformazione in un contributo positivo $\boldsymbol{A}^{+}$(caratterizzato da autovalori positivi ed uguali a zero) ed uno negativo $\boldsymbol{A}^{-}$ (caratterizzato da autovalori negativi ed uguali a zero):

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

Dove per diagonalizzazione:

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

Così da poter risolvere il problema con schema esplicito upwind sia per la parte di informazioni che si propagano in avanti e sia per la parte di informazioni che si propagano all'indietro:

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

Per un sistema iperbolico non lineare e scritto in forma conservativa, lo splitting della matrice $\boldsymbol{A}$ comporta un naturale splitting del vettore dei flussi $\boldsymbol{F}$:

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

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

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

Tutto ciò è possibile dato il carattere iperbolico del sistema preso in esame. Un'ulteriore proprietà che deve essere soddisfatta in questo caso è quella di *omogeneità*:

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

Ciò si traduce in un'opportuna linearizzazione e conseguente splitting della matrice Jacobiana che varia in base al metodo FVS applicato. L'approccio numerico dei metodi FVS consiste, infine, in uno schema esplicito e conservativo:

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

Dove il flusso numerico all'interfaccia $i+\frac{1}{2}$ è costituito da due contributi, ovvero della componente in avanti proveniente dalla cella di sinistra F_i^+ e della componente all'indietro proveniente dalla cella di destra F_{i+1}^-.

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

Nel caso specifico del metodo di Liou-Steffen applicato alle equazioni di Eulero (di seguito riportate nella forma tridimensionale) si adotta la seguente procedura per caratterizzare i flussi numerici: si parte dal problema scritto in forma conservativa e si separa la componente *convettiva* del flusso $\boldsymbol{F^{(c)}}$ dalla componente di *pressione* $\boldsymbol{F^{(p)}}$:

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

Introducendo il numero di Mach e l'entalpia totale come segue, è possibile esprimere la componente *convettiva* in un nuovo modo:

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

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

Ora, il flusso numerico all'interfaccia potrà esser caratterizzato come segue:

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

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

Definendo:

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

Con la precedente è possibile notare la prima proprietà del metodo, ovvero quella di decidere l'evoluzione della soluzione in avanti ed indietro a seconda del valore del numero di Mach valutato all'interfaccia $M_{i+1/2}$. Per tale motivo il metodo prende anche il nome di Advection Upstream Splitting Method (AUSM).

Lo splitting proposto dallo schema e quindi, la linearizzazione del problema è affrontata mediante la valutazione del numero di Mach e della pressione all'interfaccia:

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

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

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

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

Quindi il codice: prima valuta il valore assoluto del numero di Mach a livello locale $|M|$ per capire se si è in condizione subsonica o supersonica e poi in base al valore assunto dal numero di Mach all'interfaccia $M_{i+1/2}$ valuta la direzione in cui far evolvere la soluzione nel dominio di calcolo se in avanti o all'indietro.

11. Codice implementato per il metodo di Liou-Steffen

Il test consiste nell'implementare il metodo di Liou-Steffen all'interno del codice "*nozzle_w.f*" come riportato di seguito nel nuovo codice "*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. Test numerici metodo di Liou-Steffen

I dati sono costituiti da due stati iniziali costanti (ρ_L, u_L, p_L) e (ρ_R, u_R, p_R), separati da una discontinuità in posizione $x = x_0$. Tutti e cinque i test presentano un domino spaziale unitario suddiviso in cento celle.

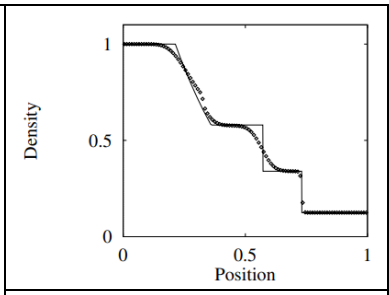
Test	x_0	T	ρ_L	u_L	p_L	ρ_R	u_R	p_R
1	0.3	0.2	1.0	0.70.05	1.0	0.125	0.0	0.1
2	0.5	0.25	1.0	-2.0	0.4	1.0	2.0	0.4
3	0.5	0.012	1.0	0.0	1000.0	1.0	0.0	0.01
4	0.4	0.035	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950
5	0.8	0.012	1.0	-19.59745	1000.0	1.0	-19.59745	0.01

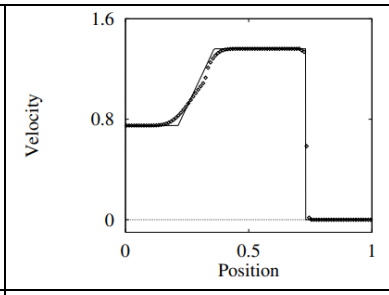
I dati di input comuni a tutti i test, e caratteristici del codice utilizzato sono:

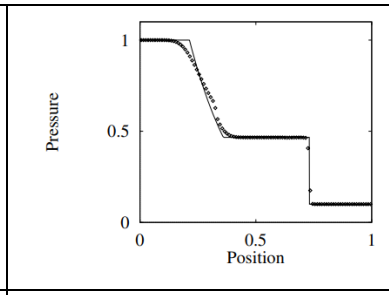
Input position of diaphram:	x_0
Input gamma:	1.4
Input dx:	0.01
Input cfl, itmax, nstmp:	0.9 10000 10
Input tmax:	T

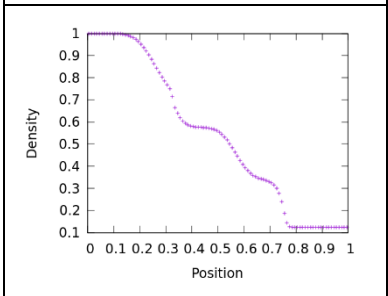
Tutti e cinque i test presentano: una soluzione numerica e una esatta in linea continua fornita dal testo [1] (pag. 280) e una soluzione numerica fornita dal codice utilizzato con discretizzazione MUSCL attiva:

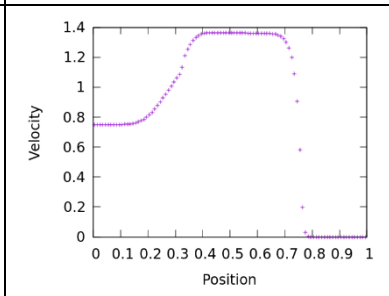
Test1

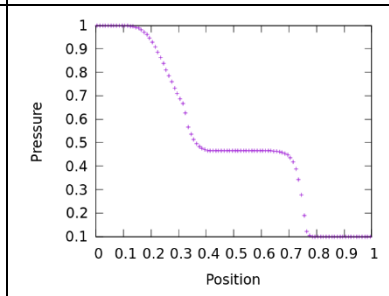




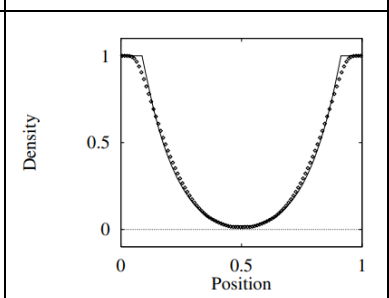


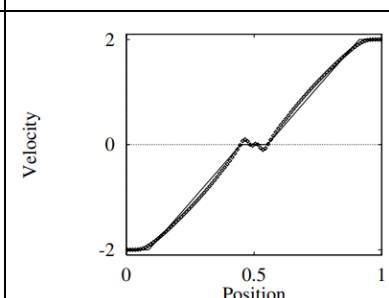


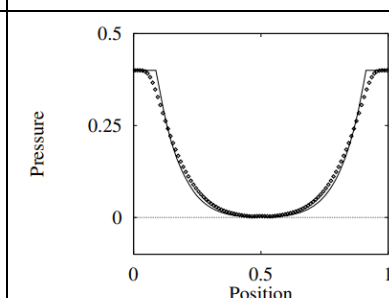


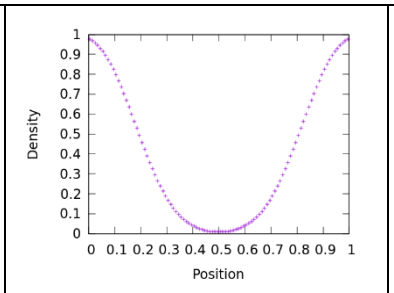
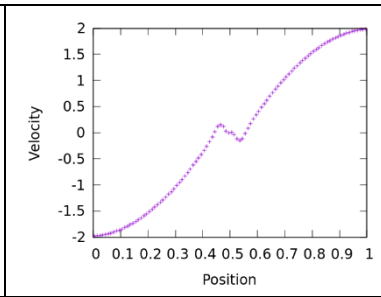
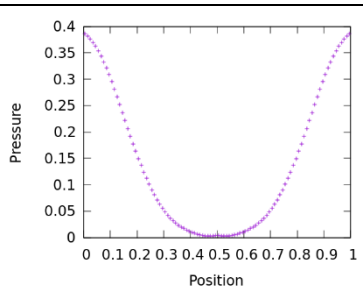
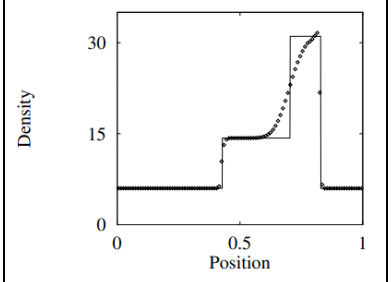
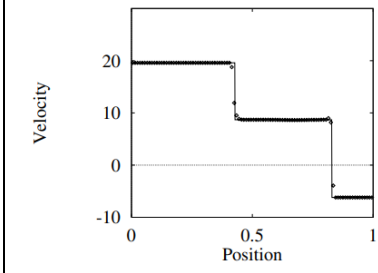
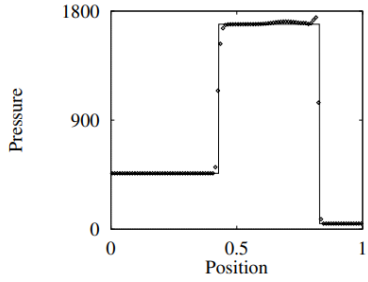
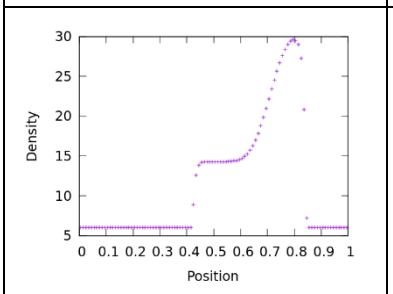
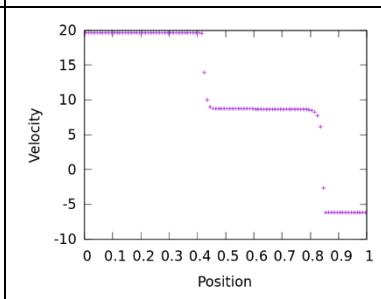
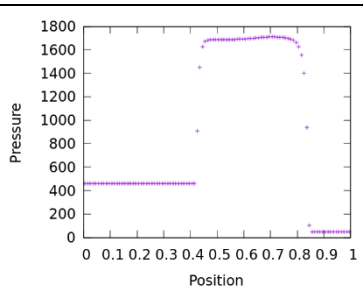
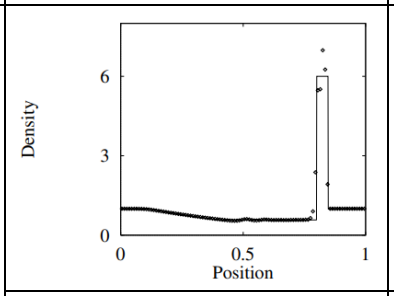
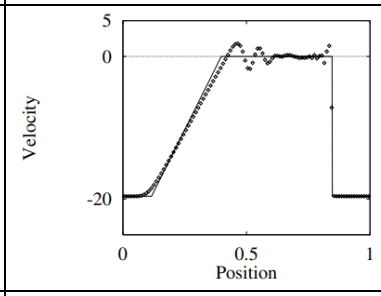
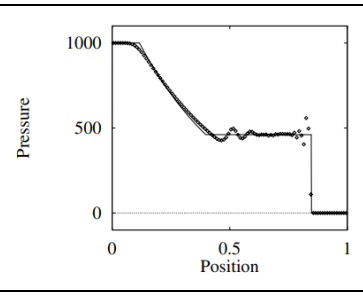
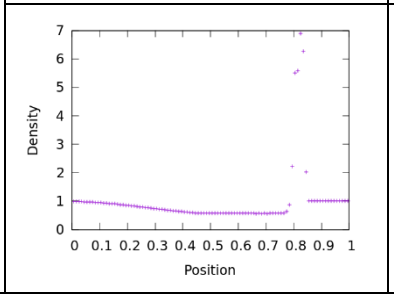
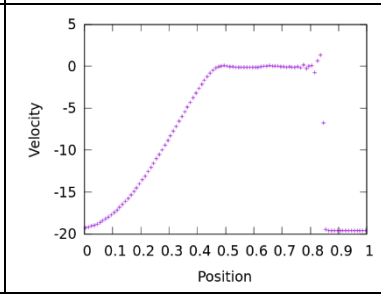
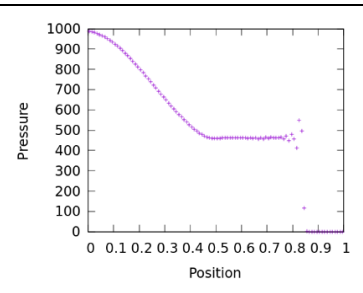


Test 2







			
Test 4			
			
Test 5			
			

13. Osservazioni test metodo di Liou-Steffen

Il Test 3 non presenta risultati sul libro di testo di riferimento e anche con il codice implementato non si ottiene altro che un errore di pressione negativa (anche abbassando il numero di Courant come suggerito sul libro).
Il test 5 presenta sempre delle oscillazioni non fisiche in corrispondenza della discontinuità ma più smussate altrove rispetto al libro nei grafici di velocità e pressione. Il precedente risultato è dovuto al metodo Runge-Kutta a due step, tenuto attivo per tutti i test, il quale comporta un aumento in accuratezza temporale della soluzione.

14. Conclusioni

Gli schemi FVS risultano più semplici ed efficienti rispetto agli schemi di Godunov (con invece solutore esatto del problema di Riemann). Dovendo risolvere il problema in esame non in modo esatto ma linearizzato e adottando una discretizzazione upwind, le caratteristiche salienti dei fenomeni fisici vengono incorporate negli schemi stessi. La soluzione del problema viene quindi fatta evolvere nella direzione di propagazione delle informazioni caratteristica del problema apportando molti vantaggi a livello di *efficienza computazionale*.

Il metodo preso in esame quale quello di Liou-Steffen ha anche il vantaggio computazionale di essere in *automatico* upwind rispetto la direzione di propagazione delle informazioni, senza necessitare quindi di distinguere il flusso numerico tra i vari casi di flusso: subsonico, supersonico verso destra o supersonico verso sinistra.

In generale però i risultati forniti presentano svantaggi dal punto di vista fisico: sono più accurati nel metodo di ROE (sempre con solutore approssimato del problema di Riemann) basti vedere la soluzione nei pressi della discontinuità che nei risultati di Liou-Steffen presentano delle oscillazioni non fisiche. I risultati del metodo di Roe sono anche più robusti in quanto il metodo di Liou-Steffen fallisce nel caso del Test 3 in presenza di situazioni estreme a numero di Mach molto elevato.

In conclusione, gli schemi FVS risultano i più efficienti tra i metodi proposti ma anche i meno accurati tra gli stessi in accordo con ciò che dice il libro di testo di riferimento [1].

15. Bibliografia

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

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