Fusion Plasma Heating Via NEUTRAL BEAM INJECTION

Silvia ESPINOSA-GUTIEZ
MIT ID: 919 277 866

18.086 **- Computational Science And Engineering II**
Spring 2014

Contents

Chapter 1

ABSTRACT

Gudonov's Method is only first order accurate. Therefore, so as to obtain a precise enough numerical solution, an extremely fine mesh has to be used. Furthermore, there are ten possible wave configurations.
Consequently, when using the Exact Riemann solver, the computational cost is prohibitively expensive.

In order to improve the accuracy with an adequate computational cost, in this report they have been implemented different fluxes (HLL and HLLC approximate Riemann solvers) and different reconstructions (2^{nd} order reconstruction, with the MINMOD slope limiter), tested and compared when solving the cylindrical explosion problem for two-dimensional Euler equations using grids between 201^2 and 4001^2 cells.

Between the results obtained, it is worth to point out the followings:

- Regarding the **approximate Riemann Solvers**, both HLL and HLLC satisfy excellently the entropy property.

  Nevertheless, although both Riemann solvers converge with the grid refinement, the rate of decay of the absolute error with the grid refinement is slightly $(1 - 3\%)$ higher for the HLLC.
  Furthermore, as soon as the mesh is fine enough the highest differences are found, as expected, in the contact wave. For example, when using the 4001^2 cells mesh, which has been run on The Grid, the contact wave is solved in a 5% less cells.
  In addition, the error is 5 to 10 % lower, depending on the grid and variable studied.

- Although **2^{nd} order reconstruction** brings spurious oscillations, there is no oscillation thanks to the Slope Limiter (in this case, MINMOD).

  When using the HLLC approximate Riemann solver and the 4001^2 cells grid, the absolute error is reduced up to 95%, and its mean is reduced from 45% to 70% depending on the variable. Moreover, the number of points required to solved the shock and contact wave is reduced an 8%.

Chapter 2

INTRODUCTION

Contents

2.1 Motivation

In **fusion** reactions, two light atomic nuclei fuse to form a heavier nucleus. In doing so they release a comparatively large amount of safe, sustainable and environmentally responsible energy, arising from the binding energy due to the strong nuclear force. Fusion power is a primary area of research in plasma physics.

The leading designs for controlled fusion research use magnetic (tokamak design) or inertial (laser) confinement of a plasma. To initiate a sustained fusion reaction, it is usually necessary to use many methods to heat the plasma, including RF heating, electron cyclotron resonance heating (ECRH), ion cyclotron resonance heating (ICRH), and neutral beam injection.

Neutral Beam Injection (NBI) involves injecting a high-energy beam of neutral atoms, typically a hydrogen isotope such as deuterium, into the core of the plasma. These energetic atoms transfer their energy to the plasma, raising the overall temperature.

In this project, the transmission of heat from the core of the plasma to the border is studied, via the modeling of low-collisionality plasmas with the simplified **MagnetoHydrodynamics (MHD) single-fluid model**.

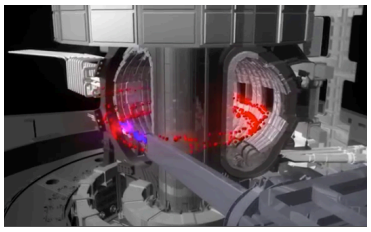


Figure 2.1: Neutral Beam Injection [1].

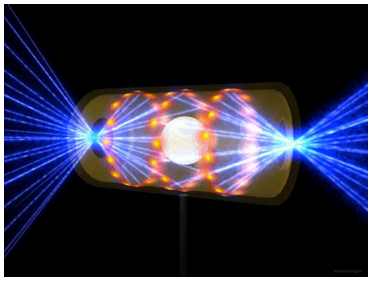


Figure 2.2: Inertial confinement fusion [2].

2.2 Euler Equations and Directional Splitting

2.2.1 Euler Equations

Applicability

Euler Equation are the Navier-Stokes equations particularized for a fluid with zero or negligible viscosity.

1D Euler equations

Conservative form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0 \tag{2.1}$$

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{pmatrix} \tag{2.2}$$

$$E = \rho\left(\frac{u^2}{2} + e\right) \tag{2.3}$$

For a γ law gas:

$$e = \frac{p}{\rho\left(\gamma - 1\right)} \tag{2.4}$$

2D Euler equations

Conservative form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}\left(\mathbf{U}\right)}{\partial x} + \frac{\partial \mathbf{G}\left(\mathbf{U}\right)}{\partial y} = 0 \tag{2.5}$$

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u\left(E + p\right) \end{pmatrix}, \qquad \mathbf{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v\left(E + p\right) \end{pmatrix} \tag{2.6}$$

$$E = \rho\left(\frac{u^2 + v^2}{2} + e\right) \tag{2.7}$$

For a γ law gas:

$$e = \frac{p}{\rho\left(\gamma - 1\right)} \tag{2.8}$$

2.2.2 Directional splitting

Main advantage

On 2D, the stability condition of the split method is twice that of the unsplit one:

$$\begin{array}{rcl} \text{Unsplit} & : & CFL \leq \frac{1}{2} \\ \text{Split} & : & CFL \leq 1 \end{array} \tag{2.9}$$

Split 2D Euler equations

Main idea:

The 2D hyperbolic system is replaced by two 1D hyperbolic systems:

1. Sweep in the x coordinate direction:

$$\left.\begin{array}{ll} \text{PDE:} & \dfrac{\partial \mathbf{U}}{\partial t} + \dfrac{\partial \mathbf{F}\left(\mathbf{U}\right)}{\partial x} = 0 \\ \text{Initial data:} & \mathbf{U}^n \end{array}\right\} \longrightarrow \mathbf{U}^{n+\frac{1}{2}} \tag{2.10}$$

2. Sweep in the y coordinate direction:

$$\left.\begin{array}{ll} \text{PDE:} & \dfrac{\partial \mathbf{U}}{\partial t}+\dfrac{\partial \mathbf{G}(\mathbf{U})}{\partial y}=0 \\ \text{Initial data:} & \mathbf{U}^{n+\frac{1}{2}} \end{array}\right\} \longrightarrow \mathbf{U}^{n} \tag{2.11}$$

Implementation:

In practice, just the first PDE is programmed, and before performing the second step, the data is rotated:

```
! rotate the data
s = CDL(2)
CDL(2) = CDL(3)
CDL(3) = S
s = CDR(2)
CDR(2) = CDR(3)
CDR(3) = S
```

$$\left\| \begin{array}{l} \mathbf{F}(2) \leftrightarrow \mathbf{F}(3) \\ \mathbf{U}(2) \leftrightarrow \mathbf{U}(3) \end{array} \right. \tag{2.12}$$

$$\mathbf{U}=\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \qquad \mathbf{F}=\begin{pmatrix} \rho u \\ \rho u^{2}+p \\ \rho u v \\ u(E+p) \end{pmatrix}, \qquad \mathbf{G}=\begin{pmatrix} \rho v \\ \rho u v \\ \rho v^{2}+p \\ v(E+p) \end{pmatrix} \tag{2.13}$$

2.3 Godunov's Method for Non-linear Systems

Problem definition

In both cases, the following Hyperbolic Boundary-Initial Value Problem has to be solved:

$$\left\| \begin{array}{lll} \text{Partial Differential Equations} & : & \frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}(\mathbf{U})}{\partial x}=0, \quad x \in [0,L], \quad t \in [0,T] \\ \text{Initial Conditions} & : & \mathbf{U}(x,0)=\mathbf{U}^{(0)}(x) \\ \text{Boundary Conditions} & : & \mathbf{U}(0,t)=\mathbf{U}_{L}(t), \ \mathbf{U}(L,t)=\mathbf{U}_{R}(t) \end{array} \right. \tag{2.14}$$

Grid generation

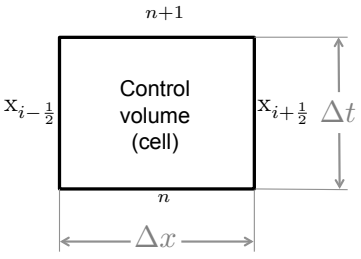


Figure 2.3: Control volume.

- **Spatial grid:**
  It is structured, cartesian (rectangular) and uniform. As it has been explained in Section 2.2.2, only the variations in one direction is considered simultaneously.
- **Temporal grid:**
  The time step has to be large enough in order to lead an efficient scheme but small enough so as to be stable.

Discretization

So as to admit discontinuous solutions, Equation 2.14 is integrated over a control volume, as the one sketched in Figure 2.3.

After utilising the Divengence Theorem (Gauss-Ostrogradski):

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}\left(x, t^{n+1}\right) dx = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}\left(x, t^{n}\right) dx + \int_{t^n}^{t^{n+1}} \mathbf{F}\left(x_{i-\frac{1}{2}}, t\right) dt - \int_{t^n}^{t^{n+1}} \mathbf{F}\left(x_{i+\frac{1}{2}}, t\right) dt \qquad (2.15)$$

Assumption of piece-wise constant distribution

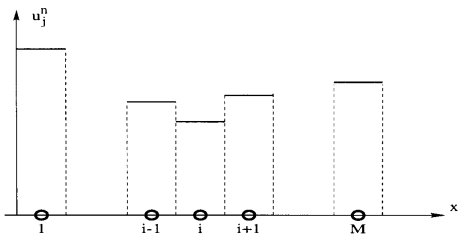


Figure 2.4: Assumption of piece-wise constant distribution of data at level n (Toro [3])

Although within cell i there may be spatial variations of the variables value, a basic assumption of the method is that, at a given time level n, the data has a piece-wise constant distribution.

The value within each cell is supposed to be constant and equal to the cell average:

$$\mathbf{U}_i^n = \frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}\left(x, t^{n}\right) dx \qquad (2.16)$$

Local Riemann problem

Hence, at a given time level n , at each intercell boundary $x_{i+\frac{1}{2}}$, we have the *local* Riemann problem $RP\left(\mathbf{U}_i^n, \mathbf{U}_i^n\right)$ with modified initial data $\left(\mathbf{U}_i^n, \mathbf{U}_{i+1}^n\right)$:

$$\left\| \begin{array}{lll} \text{Partial Differential Equations} & : & \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0, \quad x \in \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right], \quad t \in \left[t^n, t^{n+1}\right] \\ \text{Initial Conditions} & : & \mathbf{U}\left(x, 0\right) = \mathbf{U}^{(0)}\left(x\right) = \left\{ \begin{array}{ll} \mathbf{U}_i^n & if \quad x < x_i \\ \mathbf{U}_{i+1}^n & if \quad x > x_i \end{array} \right. \end{array} \right. \qquad (2.17)$$

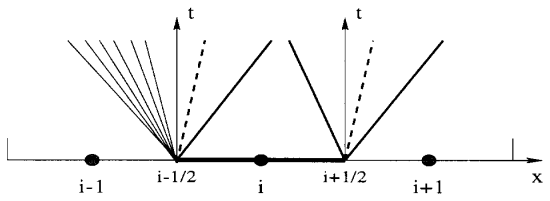


Figure 2.5: Local Riemann problems (Toro [3])

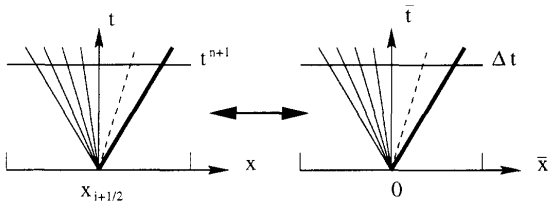


Figure 2.6: Change of variables (Toro [3])

The following change of variables is used:

$$\left\| \begin{array}{l} \bar{x} = x - x_{i+\frac{1}{2}} \\ \bar{t} = t - t_{i+\frac{1}{2}} \end{array} \right. \qquad (2.18)$$

For a time step Δt that is sufficiently small to avoid wave interaction, the global solution $\tilde{\mathbf{U}}(x,t)$ is given by:

$$\tilde{\mathbf{U}}(x,t) = \mathbf{U}_{i+\frac{1}{2}}\left(\frac{\bar{x}}{\bar{t}}\right), \quad x \in [x_i, x_{i+1}] \tag{2.19}$$

Godunov's First-Order Upwind Method

The Godunov's First-Order Upwind Method [4] is a conservative numerical scheme of the form:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i-\frac{1}{2}} - \mathbf{F}_{i+\frac{1}{2}}\right) \tag{2.20}$$

with intercell numerical flux given by:

$$\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}\left(\mathbf{U}_{i+\frac{1}{2}}\left(\frac{\bar{x}}{\bar{t}} = 0\right)\right) \tag{2.21}$$

if the time step Δt satisfies the condition [3]:

$$\Delta t \leq \frac{\Delta x}{S_{max}} \tag{2.22}$$

Hence, the intercell numerical fluxes are computed by using solutions of local Riemann problems.

2.4 Solution of the Riemann problem for 2D Euler equations

2.4.1 Characteristic variables

The system 2.17 can be expressed as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial x} = 0, \qquad \text{where } \mathbf{A}(\mathbf{U}) = \frac{\partial \mathbf{F}(\mathbf{U})}{\partial \mathbf{U}} \tag{2.23}$$

Since the system is hyperbolic:

$$\mathbf{A} = \mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^{-1}$$

$$\left\| \begin{array}{ll} \text{Eigenvectors:} & \mathbf{R} = (\mathbf{R_1}, \mathbf{R_2}, \mathbf{R_3}, \mathbf{R_4}) \\ \text{Eigenvalues:} & \boldsymbol{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \lambda_3, \lambda_4) \end{array} \right. \tag{2.24}$$

Characteristic variables $\mathbf{W}$ are defined as:

$$\mathbf{W} = \mathbf{R}^{-1}\mathbf{U} \tag{2.25}$$

The system decouples as follows:

$$\frac{\partial w_i}{\partial t} + \lambda_i \frac{\partial w_i}{\partial x} = 0, \qquad i = 1,2,3,4 \tag{2.26}$$

Characteristic w_i is constant along the curve $\frac{dx}{dt} = \lambda_i$.

2.4.2 Solution structure

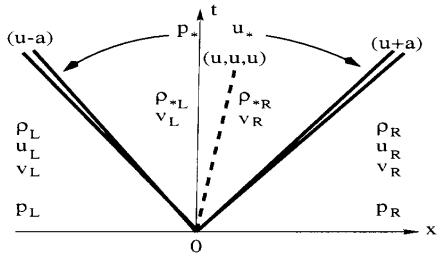


Figure 2.7: Structure of the solution of the Riemann problem for the split two dimensional Euler Equations (Toro[5])

There are four wave families associated with the eigenvalues $u-a$, u (of multiplicity 2) and $u+a$.

The similarity solution $\bar{\mathbf{U}}\left(\frac{x}{t}\right)$ consists on 4 constant states separeated by 3 waves.

The waves may be discontinuities, such as shocks waves & contact waves, or smooth transitions waves, such as rarefactions:

The speed of waves are not, in general, the characteristics speeds, given by eigenvalues.

Contact discontinuity

It is a single jump discontinuity of characteristic speed S_i in a linearly degenerated field (the gradient of the eigenvalue is perpendicular to the eigenvector) [5].

The followings conditions apply:

- Rankine-Hogoniout conditions:

$$\mathbf{F}\left(\mathbf{U}_L\right)-\mathbf{F}\left(\mathbf{U}_R\right)=S_i\left(\mathbf{U}_L-\mathbf{U}_R\right) \tag{2.27}$$

- Generalised Riemann invariants:

$$\frac{dw_1}{r_1^{(i)}}=\frac{dw_2}{r_2^{(i)}}=\frac{dw_3}{r_3^{(i)}} \quad \overset{\lambda=u \text{ wave}}{\Longrightarrow} \quad \left\| \begin{array}{l} du=0 \\ dp=0 \end{array} \right. \tag{2.28}$$

The middle wave is always a contact discontinuity.

- Parallel characteristic conditions:

$$\lambda_i\left(\mathbf{U}_L\right)=\lambda_i\left(\mathbf{U}_R\right) \tag{2.29}$$

Shock wave

It is a single jump discontinuity of characteristic speed S_i in a genuinely non-linear field (the gradient of the eigenvalue is not perpendicular to the eigenvector) [5].

The followings conditions apply:

- Generalised Riemann invariants:

$$\frac{dw_1}{r_1^{(i)}}=\frac{dw_2}{r_2^{(i)}}=\frac{dw_3}{r_3^{(i)}} \tag{2.30}$$

- The entropy condition:

$$\lambda_i\left(\mathbf{U}_L\right)>S_i>\lambda_i\left(\mathbf{U}_R\right) \tag{2.31}$$

Rarefaction fan

It is a smooth transition in a genuinely non-linear field (the gradient of the eigenvalue is not perpendicular to the eigenvector) [5].

The followings conditions apply:

- Rankine-Hogoniout conditions:

$$\mathbf{F}(\mathbf{U}_L) - \mathbf{F}(\mathbf{U}_R) = S_i(\mathbf{U}_L - \mathbf{U}_R) \tag{2.32}$$

- Divergence of characteristics:

$$\lambda_i(\mathbf{U}_L) < \lambda_i(\mathbf{U}_R) \tag{2.33}$$

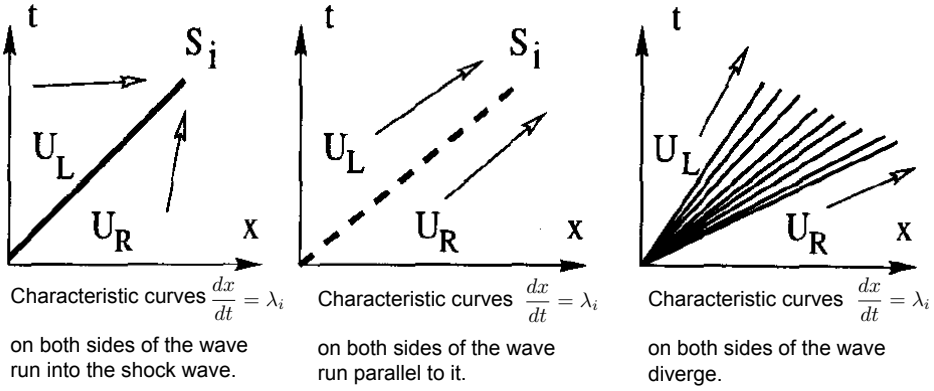


Figure 2.8: Characteristics curves near shock and contact waves and rarefaction fans. [5]

2.4.3 Possible waves patterns

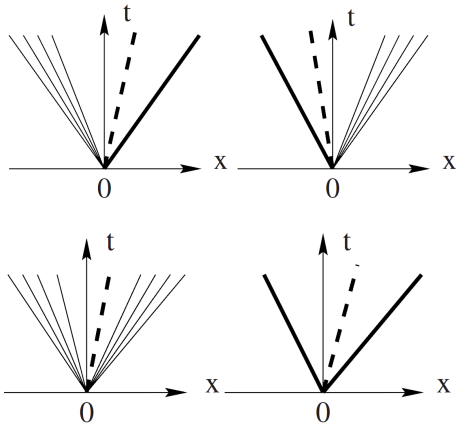


Figure 2.9: Possible wave patterns. [6]

From the previous dissertation, the possible waves patterns are the ones sketched in Figure 2.9:

Remarks:

1. The middle wave is always a contact wave.
2. The left and right waves can be a rarefaction fan or a shock wave.

2.5 Calculation of the fluxes

2.5.1 Approximate Riemann solvers

Motivation of the approximate Riemann solvers

In general, for the 1D Euler equations, there are 10 possible wave configurations to consider in the solution sampling.

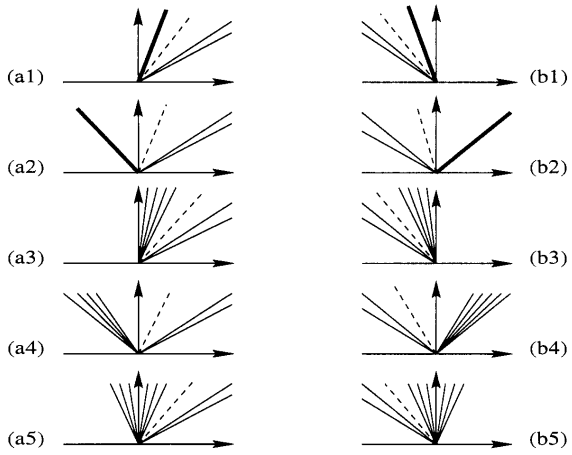


Figure 2.10: Possible configuration for the solution of 1D Euler equations (Toro[6])

The use of approximate Riemann Solvers (such as, for example, the HLL or the HLLC) can increase significantly the efficient of the scheme.

The Harten-Lax-van Leer approach (HLL) approximate Riemann solver, 1983

The solution consists of three constant states separates by two waves. The left wave propagates with velocity S_L and the right wave with velocity S_R.

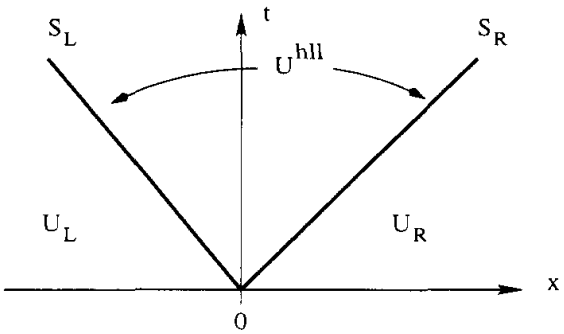


Figure 2.11: Structure of the HLL solution (Toro[7]).

The *Star Region* consists of a single constant state; all intermediate steps are lumped into the $\mathbf{U}^{HLL}$ state.

$$\tilde{\mathbf{U}}(x,t) = \begin{cases} \mathbf{U}_L & if \quad \frac{x}{t} \leq S_L \\ \mathbf{U}^{HLL} & if \quad S_L \leq \frac{x}{t} \leq S_R \\ \mathbf{U}_R & if \quad \frac{x}{t} \geq S_R \end{cases} \tag{2.34}$$

in local coordinates.

U^{HLL} is obtained from integral relations:

$$\mathbf{U}^{HLL} = \frac{S_R \mathbf{U}_R - S_L \mathbf{U}_L + \mathbf{F}_L - \mathbf{F}_R}{S_R - S_L} \tag{2.35}$$

The complete derivation can be found in Toro [7], pages 317-319.

Applying the Rankine-Hugoniout conditions, for instance, across the left wave:

$$\mathbf{F}^{HLL} - \mathbf{F}_L = S_L \left(\mathbf{U}^{HLL} - \mathbf{U}_L \right) \tag{2.36}$$

Substituting Equation 2.35 into Equation 2.36:

$$\mathbf{F}^{HLL} = \frac{S_R \mathbf{F}_L - S_L \mathbf{F}_R + S_L S_R (\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} \tag{2.37}$$

$$\boxed{\mathbf{F}^{HLL} = \begin{cases} \mathbf{F}_L & \text{if} \quad 0 \leq S_L \\ \frac{S_R \mathbf{F}_L - S_L \mathbf{F}_R + S_L S_R (\mathbf{U}_R + \mathbf{U}_L)}{S_R - S_L} & \text{if} \quad S_L \leq 0 \leq S_R \\ \mathbf{F}_R & \text{if} \quad 0 \geq S_R \end{cases}} \tag{2.38}$$

where:

$$\boxed{S_L = \min(u_L - a_L, u_R - a_R)} \tag{2.39}$$

$$\boxed{S_R = \min(u_L + a_L, u_R + a_R)} \tag{2.40}$$

Disadvantages [8]:

The absence of intermediate waves in the structure of the HLL leads to bad resolution of:

- Entropy waves
- Slip surfaces
- Material interfaces
- Vortical flows
- Ignition fronts
- Shear layers
- Contact discontinuities

```
Subroutine HLL(CDL,CDR,Flux)
  !Purpose: to compute the HLL Flux
  real CDL(4),CDR(4),Flux(4)
  real FDL(4), FDR(4)
  real SL, SR
  real al,ar,ul,ur
  Integer k

  CALL FluEval(CDL,FDL)
  CALL FluEval(CDR,FDR)

  !   Calculate estimates for wave speed
  al = ComputeSoundSpeed(CDL)
  ar = ComputeSoundSpeed(CDR)
  ul = CDL(2)/CDL(1)
  ur = CDR(2)/CDR(1)
  SL  = min(ul  - al, ur - ar)
  SR  = max(ul  + al, ur + ar)

  ! finally the flux
  If (SL .ge. 0.d0) Then
     Flux = FDL
  Else if (SR .le. 0.d0) Then
     Flux = FDR
  Else
     Do k=1,4
        Flux(K) = (SR*FDL(k)-SL*FDR(k)+SL*SR*(CDR(k)-CDL(k)))/(SR-SL)
     Enddo
  Endif

End subroutine HLL
```

Evaluation of the fluxes:

$$\mathbf{U_L} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}_L \Longrightarrow \mathbf{F}_L = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}_L$$

$$\mathbf{U_R} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}_R \Longrightarrow \mathbf{F}_R = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}_R$$

Estimates for wave speed:

Sound speed:

$$a = \sqrt{\gamma \frac{p}{\rho}}$$

Characteristic speed:

$$S_L = \min(u_L - a_L, u_R - a_R)$$
$$S_R = \min(u_L + a_L, u_R + a_R)$$

HLL Flux:

$$\mathbf{F}^{HLL} = \begin{cases} \mathbf{F}_L & \text{i} \quad 0 \leq S_L \\ \frac{S_R \mathbf{F}_L - S_L \mathbf{F}_R + S_L S_R (\mathbf{U}_R + \mathbf{U}_L)}{S_R - S_L} & \text{i} \quad S_L \leq 0 \leq S_R \\ \mathbf{F}_R & \text{i} \quad 0 \geq S_R \end{cases}$$

The Harten-Lax-van Leer - Contact approach (HLLC) approximate Riemann solver, Toro et al

In order to improve the resolution of the contact and shear waves, the structure of the HLLC approach consists of four constant states separated by three waves, as it has been sketched in Figure 2.12.

In addition to the slowest and fastest signal speeds S_L and S_R, corresponding to the eigenvalues $\lambda_1 + u$ and $\lambda_5 - u$, a middle wave of speed S^* is included, corresponding to the multiple eigenvalue $\lambda_2 = \lambda_3 = \lambda_4 = u$.

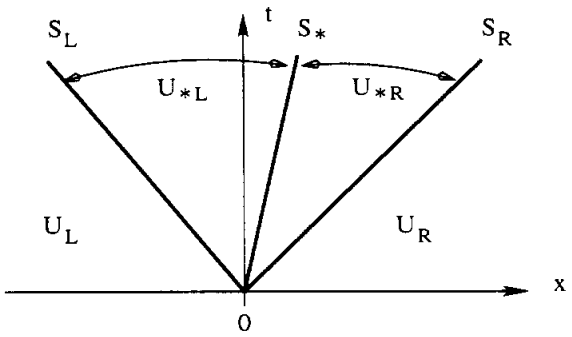


Figure 2.12: Structure of the HLLC solution (Toro[7])

The expression for the state is then:

$$\tilde{\mathbf{U}}(x,t) = \begin{cases} \mathbf{U}_L & if \quad \frac{x}{t} \leq S_L \\ \mathbf{U}_{*L} & if \quad S_L \leq \frac{x}{t} \leq S_* \\ \mathbf{U}_{*R} & if \quad S_* \leq \frac{x}{t} \leq S_R \\ \mathbf{U}_R & if \quad \frac{x}{t} \geq S_R \end{cases} \tag{2.41}$$

Applying Rankine-Hugoniout conditions:

$$\begin{aligned} \mathbf{F}_{*L} - \mathbf{F}_L &= S_L \left(\mathbf{U}_{*L} - \mathbf{U}_L \right) \\ \mathbf{F}_{*R} - \mathbf{F}_{*L} &= S_* \left(\mathbf{U}_{*R} - \mathbf{U}_{*L} \right) \\ \mathbf{F}_{*R} - \mathbf{F}_R &= S_R \left(\mathbf{U}_{*R} - \mathbf{U}_R \right) \end{aligned} \tag{2.42}$$

Since there are more unknowns than equations, the following conditions (which are satisfied by the exact solution) are imposed:

$$\begin{aligned} u_{*L} = u_{*R} = u_* \\ p_{*L} = p_{*R} = p_* \\ v_{*L} = v_L \text{ and } v_{*R} = v_R \\ w_{*L} = w_L \text{ and } w_{*R} = w_R \end{aligned} \tag{2.43}$$

Moreover, it is also imposed that:

$$S_* = u_* \tag{2.44}$$

Substituting Equations 2.43 and 2.44 into 2.42, the closed form expressions for the star values are obtained:

$$\boxed{\mathbf{U}_{*R} = \rho_R \left(\frac{S_R - u_R}{S_R - S_*} \right) \begin{pmatrix} 1 \\ S_* \\ v_R \\ \dfrac{E_R}{\rho_R} + (S_* - u_R) \left(S_* - \dfrac{p_R}{\rho_R (S_R - u_R)} \right) \end{pmatrix}} \tag{2.45}$$

$$\boxed{\mathbf{U}_{*L} = \rho_L \left(\frac{S_L - u_L}{S_L - S_*} \right) \begin{pmatrix} 1 \\ S_* \\ v_L \\ \dfrac{E_L}{\rho_L} + (S_* - u_L) \left(S_* - \dfrac{p_L}{\rho_L (S_L - u_L)} \right) \end{pmatrix}} \tag{2.46}$$

Finally, the HLLC flux is given by:

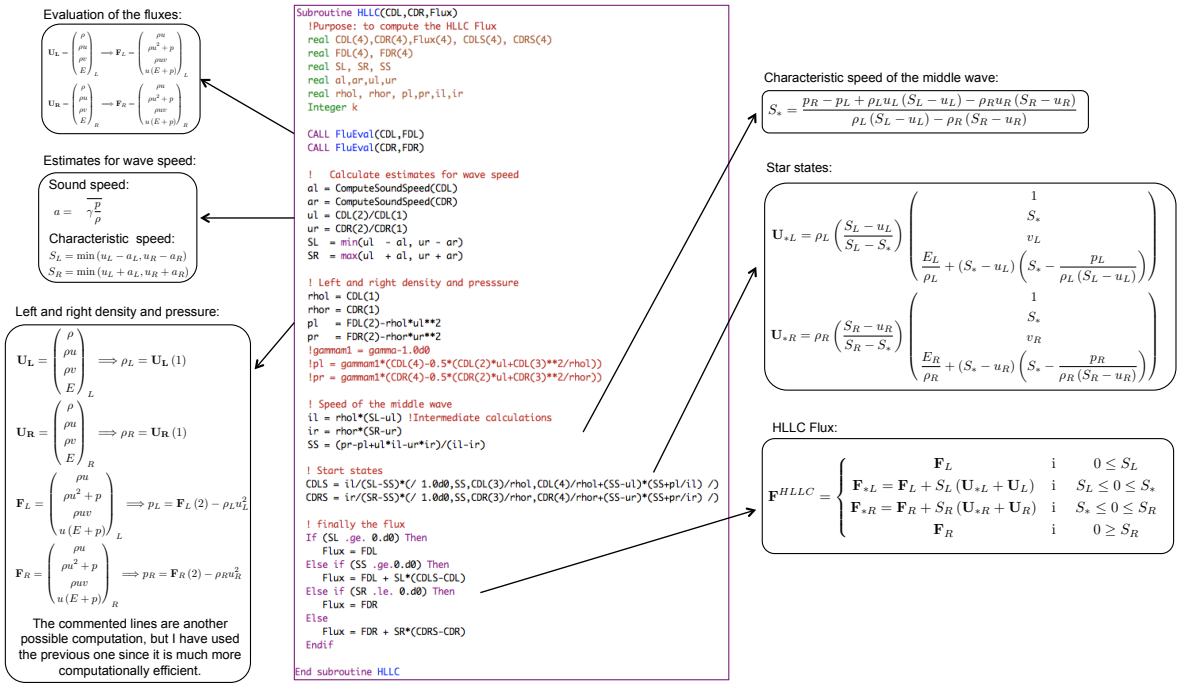
$$\boxed{\mathbf{F}^{HLLC} = \begin{cases} \mathbf{F}_L & \text{if} \quad 0 \leq S_L \\ \mathbf{F}_{*L} = \mathbf{F}_L + S_L \left(\mathbf{U}_{*L} + \mathbf{U}_L \right) & \text{if} \quad S_L \leq 0 \leq S_* \\ \mathbf{F}_{*R} = \mathbf{F}_R + S_R \left(\mathbf{U}_{*R} + \mathbf{U}_R \right) & \text{if} \quad S_* \leq 0 \leq S_R \\ \mathbf{F}_R & \text{if} \quad 0 \geq S_R \end{cases}} \tag{2.47}$$

where:

$$\boxed{S_* = \frac{p_R - p_L + \rho_L u_L (S_L - u_L) - \rho_R u_R (S_R - u_R)}{\rho_L (S_L - u_L) - \rho_R (S_R - u_R)}} \tag{2.48}$$

$$\boxed{S_L = \min \left(u_L - a_L, u_R - a_R \right)} \tag{2.49}$$

$$\boxed{S_R = \min \left(u_L + a_L, u_R + a_R \right)} \tag{2.50}$$



2.6 Variable reconstruction

2.6.1 Total Variation Diminishing schemes

Monotonicity

- **Definition:**

  A scheme:

$$u_i^{n+1} = H\left(u_{i-r+1}, \ldots, u_{i+s}^n\right), \qquad r, s > 0 \tag{2.51}$$

  is monotone if

$$\frac{\partial H}{\partial u_j^n} \geq 0, \qquad \forall j \tag{2.52}$$

- **Main property:**

  A monotonous scheme do not generate spurious oscillations.

Gudonov's Theorem (1959)

"There are no monotone linear schemes of second or higher order of accuracy".

Total Variation Diminishing

A scheme is Total Variation Diminishing (TVD) if:

$$TV\left(u^{n+1}\right) \leq TV\left(u^{n}\right) \tag{2.53}$$

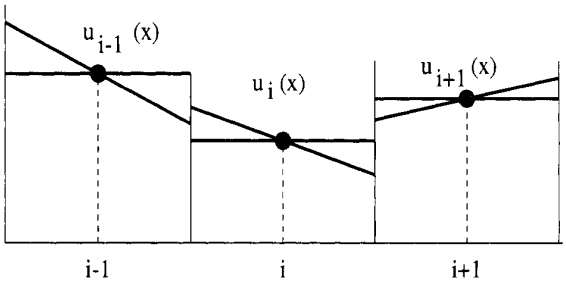
where the total variation is defined as follows:

$$TV\left(u^{n}\right) = \sum_{\infty}^{+\infty} \left|u_{i+1}^{n} - u_{i}^{n}\right| \tag{2.54}$$

High order reconstruction

The main idea is to consider that at a certain time n, the data has a linear distribution, instead of the piece-wise constant distribution assumed by the first-order Godunov Method, presented in Section 2.3.

However, according to the Godunov's Theorem, this will produce spurious oscillations.
Therefore, slope constraints are needed so as to make the scheme Total Variation Diminishing.



2.6.2 Second order reconstruction

Condition for the scheme slope so as to be Total Variation Diminishing

For any component of the vector of conserved variables **U**:

$$\boxed{q\left(x\right) = q_i + \frac{\Delta i}{\Delta x}\left(x - x_i\right)} \tag{2.55}$$

The slope is generally calculated by using three-point stencil:

$$\Delta i = \frac{1+\omega}{2}\left(q_i - q_{i-1}\right) + \frac{1-\omega}{2}\left(q_{i+1} - q_i\right) \tag{2.56}$$

where:

- $w = -1 \longrightarrow$ Backwards differencing scheme
- $w = 0 \longrightarrow$ Center differencing scheme
- $w = 1 \longrightarrow$ Forwards differencing scheme

In order to avoid spurious oscillations, the slope is limited as follows:

$$\overline{\Delta i} = \epsilon(r)\Delta i \tag{2.57}$$

where:

$$\left\| \begin{array}{l} \\ r = \dfrac{\Delta_{upwind}}{\Delta_{local}}, \end{array} \right. \qquad \begin{array}{c} \epsilon(r) \equiv \text{slope limiting function} \\ \text{for instance: } r_{i+\frac{1}{2}} = \begin{cases} \dfrac{u_i^n - u_{i-1}^n}{u_{i+1}^n - u_i^n}, & a > 0 \\ \dfrac{u_{i+2}^n - u_{i+1}^n}{u_{i+1}^n - u_i^n}, & a < 0 \end{cases} \end{array} \tag{2.58}$$

The scheme will be TVD if the limited slope verifies:

$$\boxed{\overline{\Delta i} \leq \frac{\text{sign}\left(\Delta_{i-\frac{1}{2}}\right) + \text{sign}\left(\Delta_{i+\frac{1}{2}}\right)}{2} \min\left(\frac{2\left|\Delta_{i-\frac{1}{2}}\right|}{1+CFL}, \frac{2\left|\Delta_{i+\frac{1}{2}}\right|}{1-CFL}\right)} \tag{2.59}$$

Actually, Sweby [9] suggested the following reduced portion of the TVD region as a suitable range for the flux limiting function:

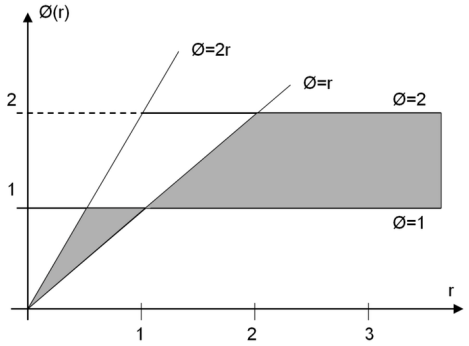


Figure 2.13: Admissible limiter region for second-order TVD schemes [9].

MINMOD slope limiter

Kolgan [10] proposed the following slope limiter:

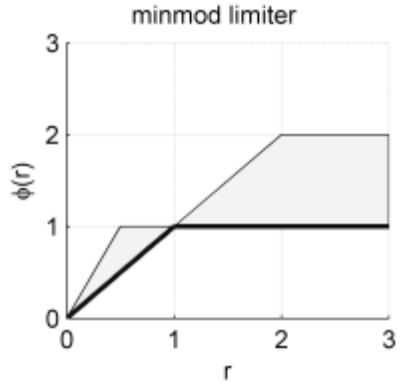


Figure 2.14: MINMOD slope limiter. Created in Matlab.

$$\epsilon(r) = \begin{cases} 0 & r \leq 0 \\ r & 0 \leq r \leq 1 \\ 1 & r \geq 1 \end{cases} \tag{2.60}$$

Hence,

$$\boxed{\Delta i = \text{minmod}\left(q_i - q_{i-1}, q_{i+1} - q_i\right)} \tag{2.61}$$

$$\boxed{\text{minmod} = \frac{1}{2}\left[sign(x) + sign(y)\right] \min\left(|x|, |y|\right)} \tag{2.62}$$

```
Subroutine Reconstruction(U1D,CDL,CDR)
  ! Purpose: reconstruction procedure
  Integer F
  Real U1d(4,-1:2),CDL(4),CDR(4)


  Select Case(SpatialOrder)

  Case(1)   ! First order
     Do f=1,4
        CDL(f) = U1D(f,0)
        CDR(f) = U1D(f,1)
     Enddo

  Case(2)   ! minbee limiter
     Do f=1,4
        CDL(f) = U1D(f,0) + 0.5* minmod(U1D(f,0)-U1D(f,-1), U1D(f,1)-U1D(f,0))
        CDR(f) = U1D(f,1) - 0.5* minmod(U1D(f,1)-U1D(f,0),  U1D(f,2)-U1D(f,1))
     Enddo

  Case default
     print*,' Wrong Limiter Type. Stop the code'
     stop
  end select

end subroutine Reconstruction
```

$$\Delta i = \text{minm} \quad (q_i - q_{i-1}, q_{i+1} - q_i)$$
$$q(x) = q_i + \frac{\Delta i}{\Delta x}(x - x_i)$$

```
Real function minmod(x,y)
  ! Purpose: compute slope limiter function minmod
  Real x,y

  minmod = (max(0.0d0, sign(0.5d0,x))+max(0.0d0, sign(0.5d0,y)))*min(abs(x),abs(y))

End function minmod
```

$$\text{minm} = \frac{1}{2}\left[sign(x) + sign(y)\right] \min\left(|x|, |y|\right)$$

2.7 Boundary conditions

In order to settle them, fictitious cells are introduced, as is shown in Figure 2.15:

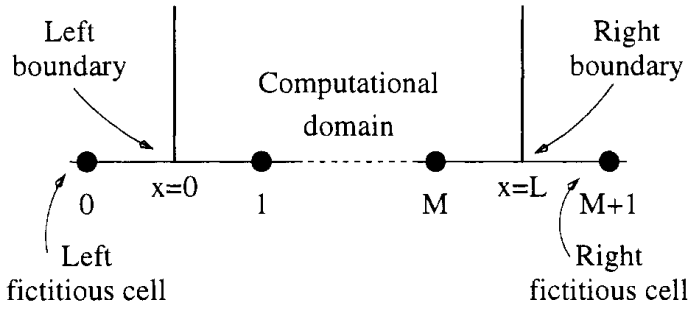


Figure 2.15: Fictitious cells so as to enforce the boundary conditions [6].

Two types of boundary conditions are studied:

1. Reflective boundary conditions (solid walls):

$$\left\| \begin{array}{c} \rho^n_{M+1} = \rho^n_M \\ u^n_{M+1} = -u^n_M \\ p^n_{M+1} = p^n_M \end{array} \right. \tag{2.63}$$

2. Non-reflective (transmissive) boundary conditions:

$$\left\| \begin{array}{l} \rho^n_{M+1} = \rho^n_M \\ u^n_{M+1} = u^n_M \\ p^n_{M+1} = p^n_M \end{array} \right. \tag{2.64}$$

2.8 Stable time step and time evolution

Stable time step

$$\Delta t = \frac{K_{CFL}\Delta x}{S^n_{max}}, \qquad 0 < K_{CFL} < 1 \tag{2.65}$$

There are two possible choices for S^n_{max} exist:

1. $S^n_{max} = \max_i \left(\left| S^L_{i+\frac{1}{2}} \right|, \left| S^R_{i+\frac{1}{2}} \right| \right)$
2. $S^n_{max} = \max_i \left(|u_i| + a_i \right)$

Time evolution: 2^{nd} order Runge-Kutta

For this kind of problems, the 1^{st} order Euler method is not recommended. Therefore, the 2^{nd} order Runge-Kutta method is used:

1. Predictor:
$$\mathbf{U}^{n+\frac{1}{2}}_i = \mathbf{U}^n_i - \frac{1}{2}\frac{\Delta t}{\Delta x}\left[\mathbf{F}_{i+\frac{1}{2}}\left(\mathbf{U}^n\right) - \mathbf{F}_{i-\frac{1}{2}}\left(\mathbf{U}^n\right) \right] \tag{2.66}$$
2. Corrector:
$$\mathbf{U}^{n+1}_i = \mathbf{U}^n_i - \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{i+\frac{1}{2}}\left(\mathbf{U}^{n+\frac{1}{2}}\right) - \mathbf{F}_{i-\frac{1}{2}}\left(\mathbf{U}^{n+\frac{1}{2}}\right) \right] \tag{2.67}$$

Chapter 3

PROBLEM FORMULATION

Contents

3.1 Programming language, editor, compiler and processor

The code developed was written in FORTRAN 90, utilising the editor Aquamacs (Emacs), version 2.4 for MacOSX.

The compiler used was Intel Fortran Composer XE 2011 5.209 for MacOSX.
The code was compiled with double precision (-r8), in order that the computer round-off error affected as less as possible the numerical results.

The cases in that the grid consisted of 201^2, 401^2 and 801^2 cells were run in a MacBook Pro (MacOS version 10.6.8) with a 2.66GHz Intel Core Duo processor.
Nevertheless, the case in which the grid consisted of 4001^2 cells were performed by using "The Grid" of the Massachusetts Institute of Technology (MIT Athena cluster).

3.2 Problem definition

3.2.1 Governing equations

Two dimensional compressible equations of a gamma-law gas.

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} = 0\,, \qquad (x,y) \in [-1,1] \times [-1,1]\,. \tag{3.1}$$

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u\,(E+p) \end{pmatrix}, \qquad \mathbf{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v\,(E+p) \end{pmatrix} \tag{3.2}$$

$$E = \rho\left(\frac{u^2+v^2}{2} + e\right) \tag{3.3}$$

$$e = \frac{p}{\rho\,(\gamma - 1)} \tag{3.4}$$

In this case, the fluid considered is air and, consequently, $\gamma = 1.4$.

3.2.2 Boundary conditions

The boundary conditions are **transmissive**:

$$\left\| \begin{array}{ll} \rho^n_{M+1} = \rho^n_M, & \rho^n_{M+2} = \rho^{n-1}_M \\ u^n_{M+1} = u^n_M, & u^n_{M+2} = u^{n-1}_M \\ p^n_{M+1} = p^n_M, & p^n_{M+2} = p^{n-1}_M \end{array} \right. \tag{3.5}$$

3.2.3 Stable time step

$$\Delta t = \frac{K_{CFL}\Delta x}{S^n_{max}} \tag{3.6}$$

For the first 10 iterations, $K_{CFL} = 0.1CFL$. From then on, $K_{CFL} = CFL$. In this case, $CFL = 0.35$.

Moreover, S^n_{max} is chosen as $S^n_{max} = \max_i (|u_i| + a_i)$.

3.2.4 Initial conditions and reported time

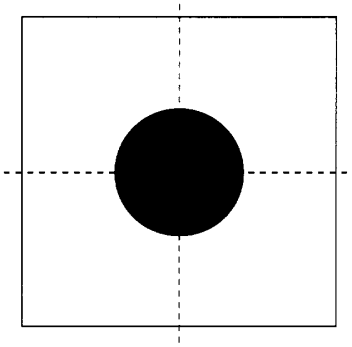


Figure 3.1: At $t = 0$, there is a region with high density and pressure in the center.

Density:

$$\rho(x, y, 0) = \begin{cases} 1.0\ , & \text{if } r \leq 0.4 \\ \rho_0\ , & \text{if } r > 4 \end{cases}, \quad r^2 = x^2 + y^2 \tag{3.7}$$

Velocity:

$$u(x, y, 0) = v(x, y, 0) = 0 \tag{3.8}$$

Pressure:

$$p(x, y, 0) = \begin{cases} 1.0\ , & \text{if } r \leq 0.4 \\ p_0\ , & \text{if } r > 4 \end{cases}, \quad r^2 = x^2 + y^2 \tag{3.9}$$

Density ρ_0	Pressure p_0	Output time T
0.5	0.1	0.3

Table 3.1: Parameters for the initial condition and reported time.

Please note that the variables have been nondimensionalized and, therefore, they have no units.

3.2.5 "Cylindrical" explosion problem

The reference radial solution is obtained by solving on a very fine mesh the Euler equation with a geometric source term.

At $t = 0.3$, as it can be seen in Figure 3.2, the solution exhibits:

- A circular **shock wave** traveling away from the center. It can be observed that across the shock wave the density and the pressure increase.
- A circular **contact wave** traveling to the center. Note that velocity is continuous across a contact wave (otherwise, a vacuum could be generated) and pressure is continuous too (otherwise, it would move until this condition was true.)
- A circular **rarefaction fan** traveling to the center, thought which the density and pressure decrease while the velocity module increases.

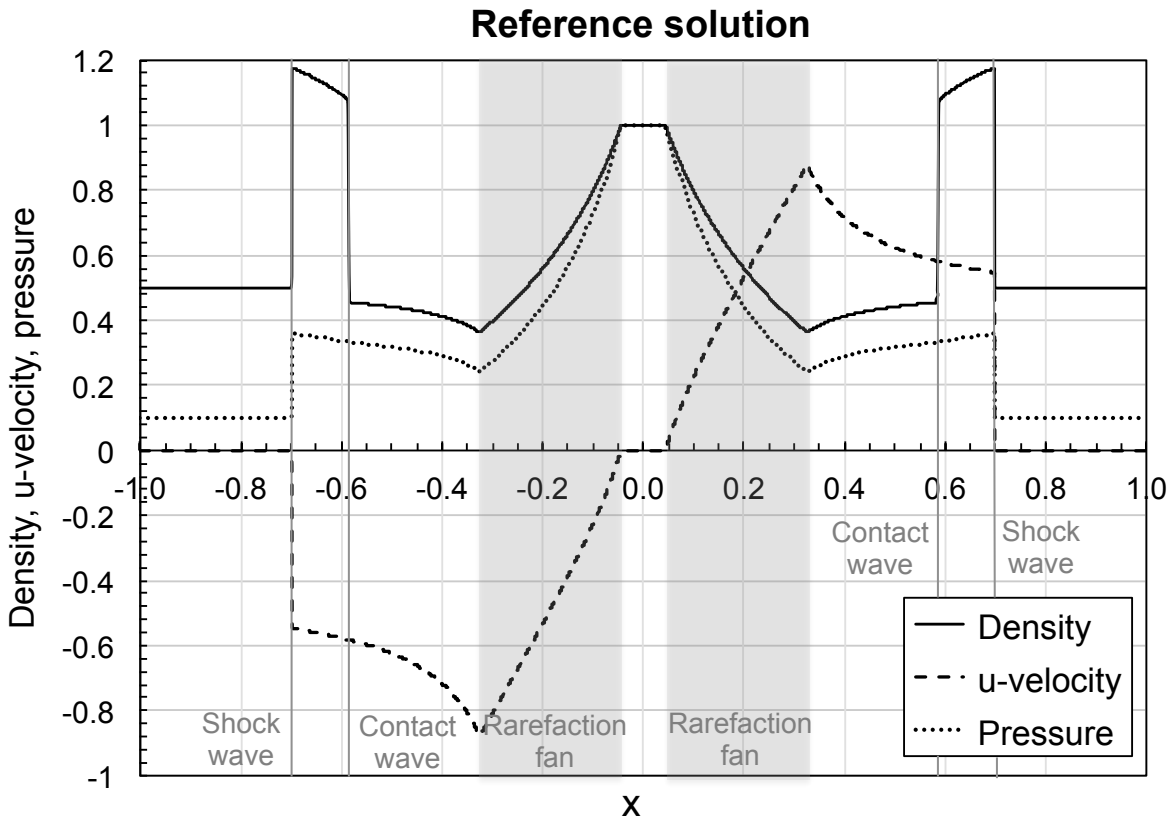


Figure 3.2: Identification of the shock waves, contact waves and rarefaction fans by analysis the density, u-velocity and pressure of the reference solution ($\rho_0 = 0.5$, $p_0 = 0.1$, $T = 0.3$).

Chapter 4

RESULTS

Contents

4.1 Approximate Riemann solvers

4.1.1 HLL approximate Riemann solver

Qualitative comparison

In Figures 4.1 and 4.2, pressure, u-velocity and pressure along the line $y = 0$ obtained by using the Godunov method in conjunction with the HLL approximate Riemann solver and the 1^{st} order reconstruction ('HLL') has been plotted against the reference solution ('Experimental') for a wide range of grids : 201^2, 401^2, 801^2 and 4001^2 cells ($\rho_0 = 0.5$, $p_0 = 0.1$, $T = 0.3$).

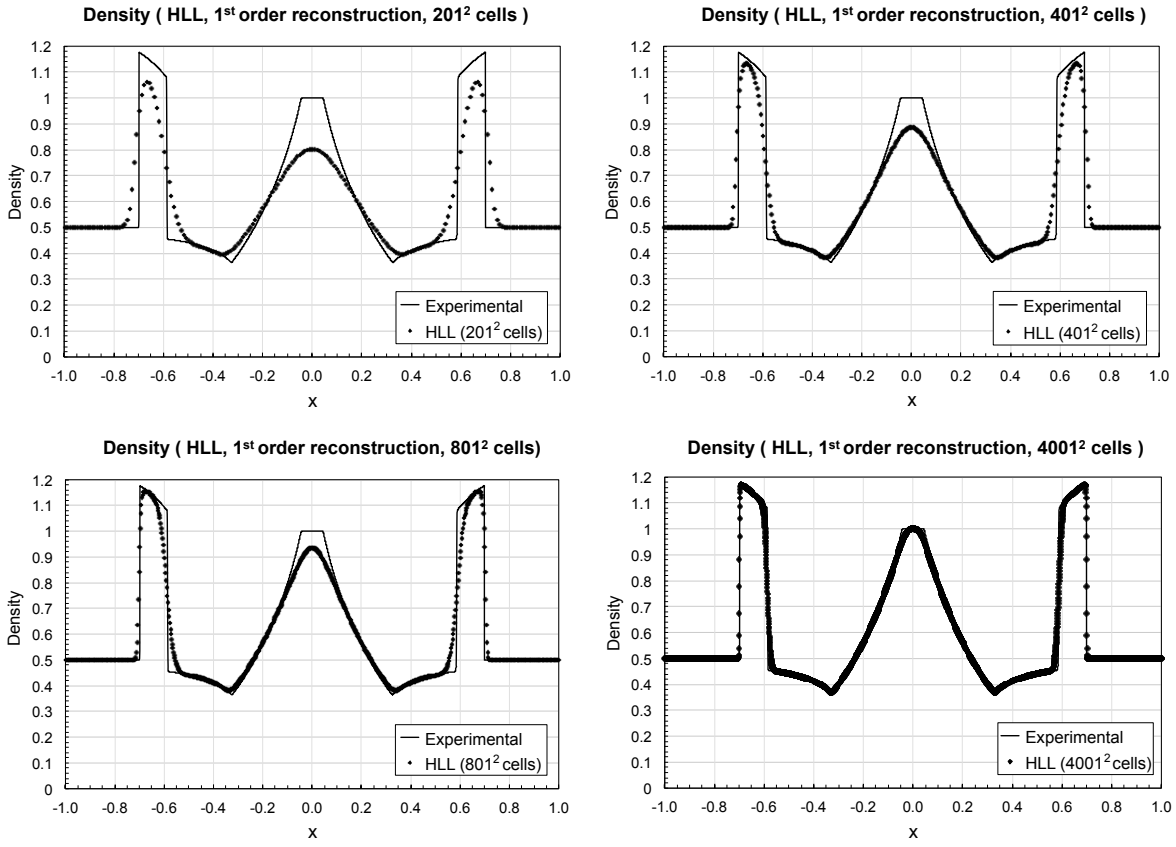


Figure 4.1: Density along the line $y = 0$ obtained by using the Godunov method in conjunction with the HLL approximate Riemann solver and first order reconstruction for the range of grids: 201^2, 401^2, 801^2 and 4001^2 cells ($\rho_0 = 0.5$, $p_0 = 0.1$, $T = 0.3$).

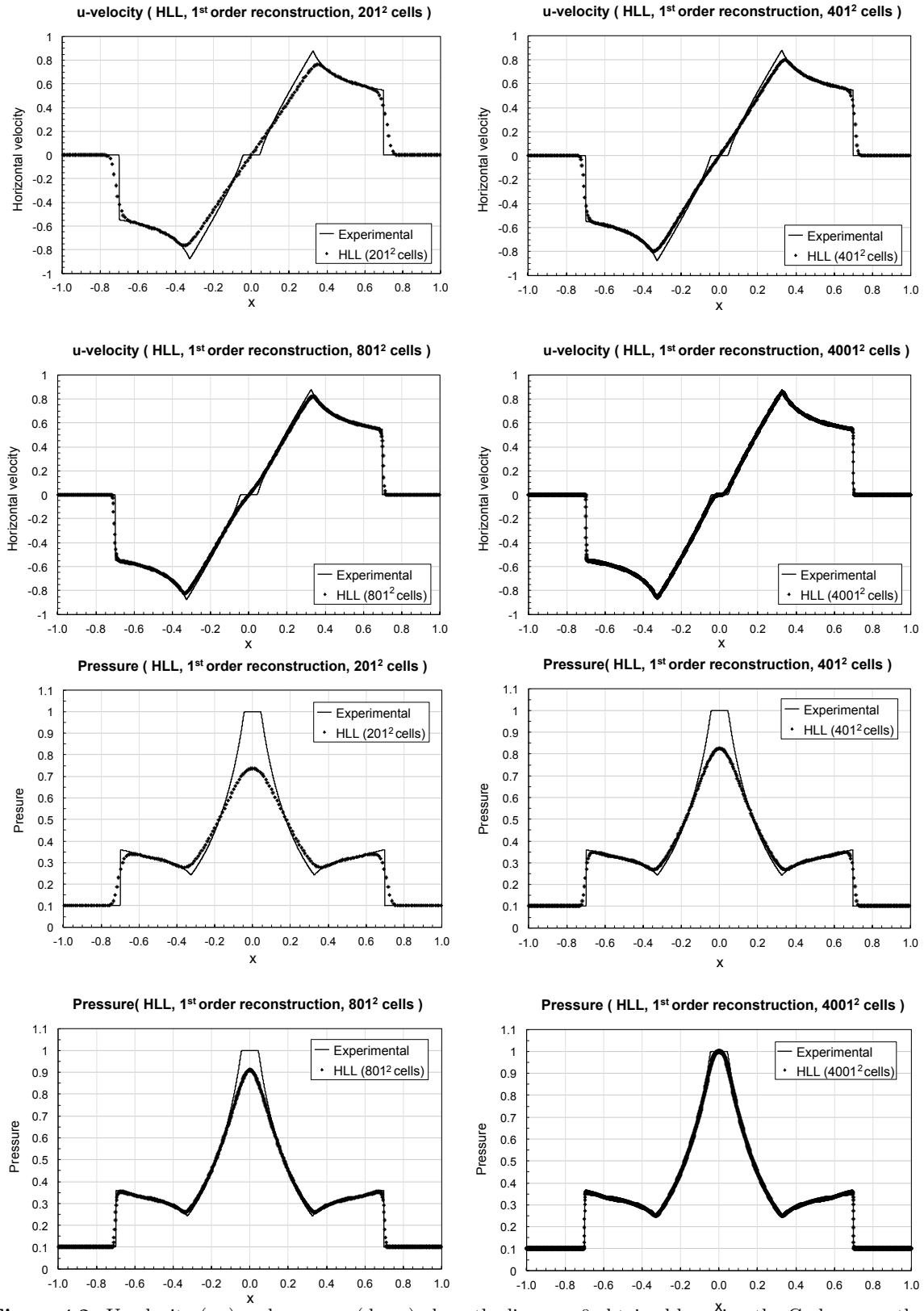


Figure 4.2: U-velocity (up) and pressure (down) along the line $y = 0$ obtained by using the Godunov method in conjunction with the HLL approximate Riemann solver (symbols) compared to the reference solution (line) .

Remarks:

1. The sharper discrepancies between both can be found near the shock and contact waves and at the beginning and at the end of the rarefaction fan.
2. For the first three cases, the correct value at the center is not reached. That is why it has been decided to run a very fine mesh (4001^2 cells) by using The Grid, so as to be completely sure that the implementation was correct.
3. The results obtained with the finest mesh match brilliantly the reference results. Therefore, it can be concluded that the implementation is correct.
4. On the one hand, the computational cost grows as the total number of cells is increased. However, on the other hand, the finer the mesh is, the closer to the reference solution the numerical solution obtained with the HLL approximate Riemann solver becomes.

Quantitative comparison

In order to analyse the convergence of the solution with the grid refinement, it has be considered that considering how the error decreases for just a single point is not representative, as it does not reflect what happens in each discontinuity.

Therefore, so as to take into account all the regions near the waves, maximum and mean absolute and relative errors have been studied.

The maximum absolute and relative error were registered at the shock waves. However, there are not significant so as to study the convergence with the grid refinement because, as the shock wave cannot be solved with just one point, the error will be larger as the values behind the shock wave increased and that happens when the solution goes closer to the reference.

As a conclusion, the absolute and relative error mean should be used so as to study the convergence with the grid refinement.

- **Density:**

	Maximum absolute error $\lvert\rho_{HLL}-\rho_{ref}\rvert_{max}$	**Location of the maximum absolute error** $x_{\lvert\rho_{HLL}-\rho_{ref}\rvert_{max}}$	**Maximum relative error (%)** $\left\lvert\frac{\rho_{HLL}-\rho_{ref}}{\rho_{ref}}\right\rvert_{max}$	**Location of the maximum relative error** $x_{\left\lvert\frac{\rho_{HLL}-\rho_{ref}}{\rho_{ref}}\right\rvert_{max}}$	**Absolute error mean** $\overline{\lvert\rho_{HLL}-\rho_{ref}\rvert}$	**Relative error mean (%)** $\overline{\left\lvert\frac{\rho_{HLL}-\rho_{ref}}{\rho_{ref}}\right\rvert}$
201^2 cells	0.3585	0.7065	71.70 %	0.7065	**0.061521**	**8.6183 %**
401^2 cells	0.3629	0. 7032	72.58 %	0.7032	**0.038863**	**5.5735 %**
801^2 cells	0. 4383	0.6991	87.66 %	0.6991	**0.025218**	**3.6587 %**
4001^2 cells	0.3980	0.6988	86.65 %	0.6988	**0.007900**	**1.1570 %**

Table 4.1: Absolute error maximum value and its location, maximum relative error and its location and absolute and relative error means of the density when compared to the reference data (HLL approximate Riemann solver, 1^{st} order reconstruction) .

In Figure 4.3, it has been found a correlation for the absolute and relative error as a function of the grid refinement. The most accurate correlation was obtained by using a Power Law.

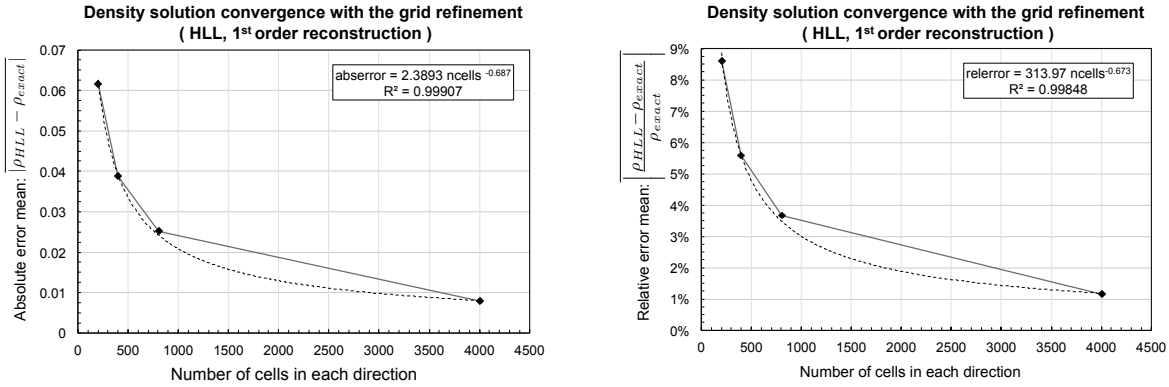


Figure 4.3: Pressure solution convergence with the grid refinement.

Remarks:

1. It can be observed that the pressure solution converges with the grid refinement.
2. When the number of cells is doubled, the absolute error mean is divided by 1.61 and the relative error mean by 1.59.
3. For the finest grid, the mean absolute error is lower than just 0.01 and the mean relative error 1.1%.

- **u-velocity**:

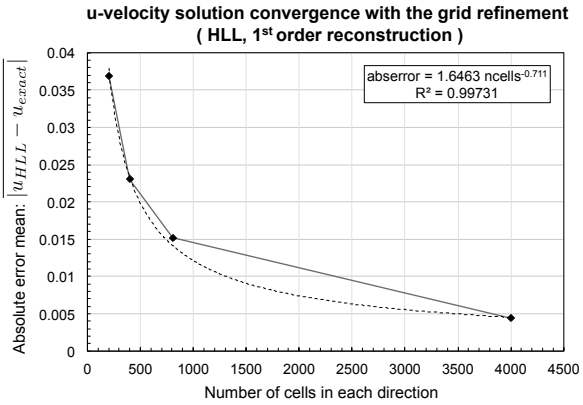
The u-velocity is zero in some points and, therefore, the relative error cannot be defined for those.

Furthermore, although the previous points were not taken into account, for the points whose u-velocity is almost zero the relative error is so large that other points relative error becomes negligible.
Actually, the larger the number of cells is, the closer to 0 the lower (in module) u-velocity becomes and, hence, the larger the relative error mean is.

As a conclusion, it makes no sense to study of the tendency of the relative error mean for the u-velocity, and only the absolute error mean is analysed.

	Maximum absolute error $\|u_{HLL} - u_{ref}\|_{max}$	Location of the maximum absolute error $x_{\|u_{HLL}-u_{ref}\|_{max}}$	Absolute error mean $\overline{\|u_{HLL} - u_{ref}\|}$
201^2 cells	0.3500	0.7065	**0.036827**
401^2 cells	0.3471	0. 7032	**0.023014**
801^2 cells	0. 4006	0.6991	**0.015226**
4001^2 cells	0.3684	0.6988	**0.004394**

Table 4.2: Absolute error maximum value and its location, maximum relative error and its location and absolute and relative error means of the u-velocity when compared to the reference data (HLL approximate Riemann solver, 1^{st} order reconstruction) .



Remarks:

1. It can be observed that the velocity solution converges with the grid refinement.
2. When the number of cells is doubled, the absolute error mean is divided by 1.64.
3. For the finest grid, the mean absolute error is lower than just 0.05 .

Figure 4.4: Velocity solution convergence with the grid refinement.

- **Pressure:**

	Maximum absolute error $\lvert p_{HLL} - p_{ref} \rvert_{max}$	Location of the maximum absolute error $x_{\lvert p_{HLL} - p_{ref} \rvert_{max}}$	Maximum relative error (%) $\left\lvert \frac{p_{HLL} - p_{ref}}{p_{ref}} \right\rvert_{max}$	Location of the maximum relative error $x_{\left\lvert \frac{p_{HLL} - p_{ref}}{p_{ref}} \right\rvert_{max}}$	Absolute error mean $\lvert p_{HLL} - p_{ref} \rvert$	Relative error mean (%) $\left\lvert \frac{p_{HLL} - p_{ref}}{p_{ref}} \right\rvert$
201^2 cells	0.2815	0.0398	130.9 %	0.7065	**0.036827**	**3.6827 %**
401^2 cells	0. 2060	0.0449	130.8 %	0.7032	**0.023014**	**2.3015 %**
801^2 cells	0. 1607	0.6991	160.7 %	0.6991	**0.015226**	**1.5227 %**
4001^2 cells	0.1441	0.6988	163.2 %	0.6988	**0.004394**	**0.4394 %**

Table 4.3: Error analysis of the pressure when compared to the reference data (HLL approximate Riemann solver, 1^{st} order reconstruction) .

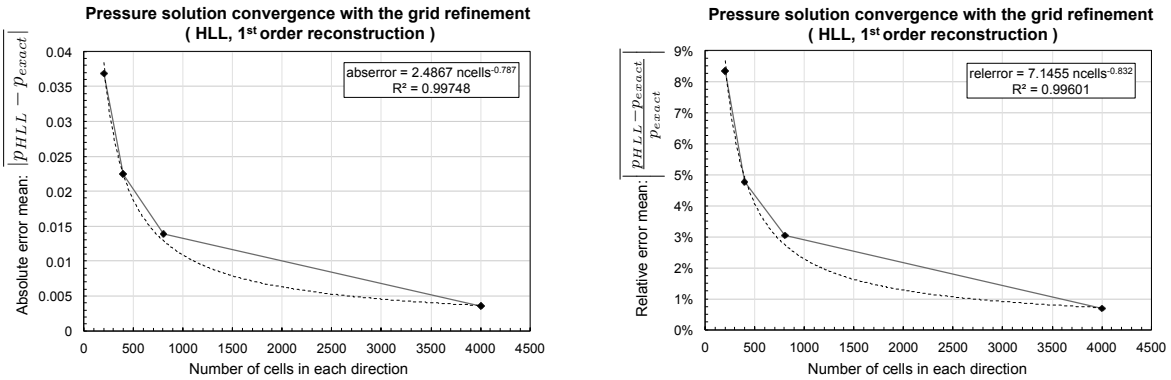


Figure 4.5: Pressure solution convergence with the grid refinement.

Remarks:

1. It can be observed that the pressure solution converges with the grid refinement.
2. When the number of cells is doubled, the absolute error mean is divided by 1.725 and the relative error mean by 1.78.
3. For the finest grid, the mean absolute error is lower than 0.005% and the mean relative error 0.8%.

4.1.2 HLLC approximate Riemann solver

Qualitative comparison

In Figures 4.6 and 4.7, pressure, u-velocity and pressure along the line $y = 0$ obtained by using the Godunov method in conjunction with the HLLC approximate Riemann solver and the 1^{st} order reconstruction ('HLLC') has been plotted against the reference solution ('Experimental') for a wide range of grids : 201^2, 401^2, 801^2 and 4001^2 cells.

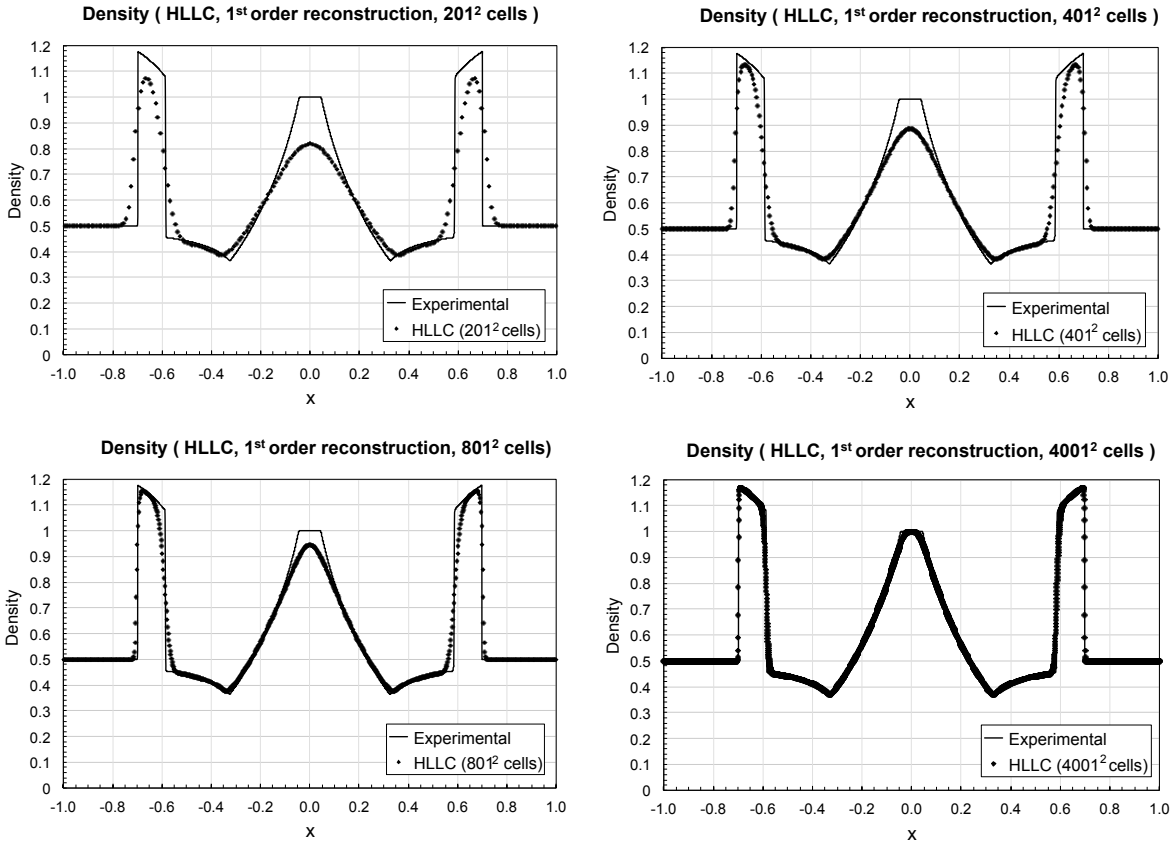


Figure 4.6: Density along the line $y = 0$ obtained by using the Godunov method in conjunction with the HLLC approximate Riemann solver and first order reconstruction for the range of grids: 201^2, 401^2, 801^2 and 4001^2 cells ($\rho_0 = 0.5$, $p_0 = 0.1$, $T = 0.3$).

Remarks:

1. The sharper discrepancies between both can be found near the shock and contact waves and at the beginning and at the end of the rarefaction fan.
2. For the first three cases, the correct value at the center region is not reached. That is why it has been decided to run a very fine mesh (4001^2 cells) by using The Grid.
3. The results obtained with the finest mesh match brilliantly the reference results. Therefore, it can be concluded that the implementation is correct.
4. On the one hand, the computational cost grows as the total number of cells is increased. However, on the other hand, the finer the mesh is, the closer to the reference solution the numerical solution obtained with the HLLC approximate Riemann solver becomes.

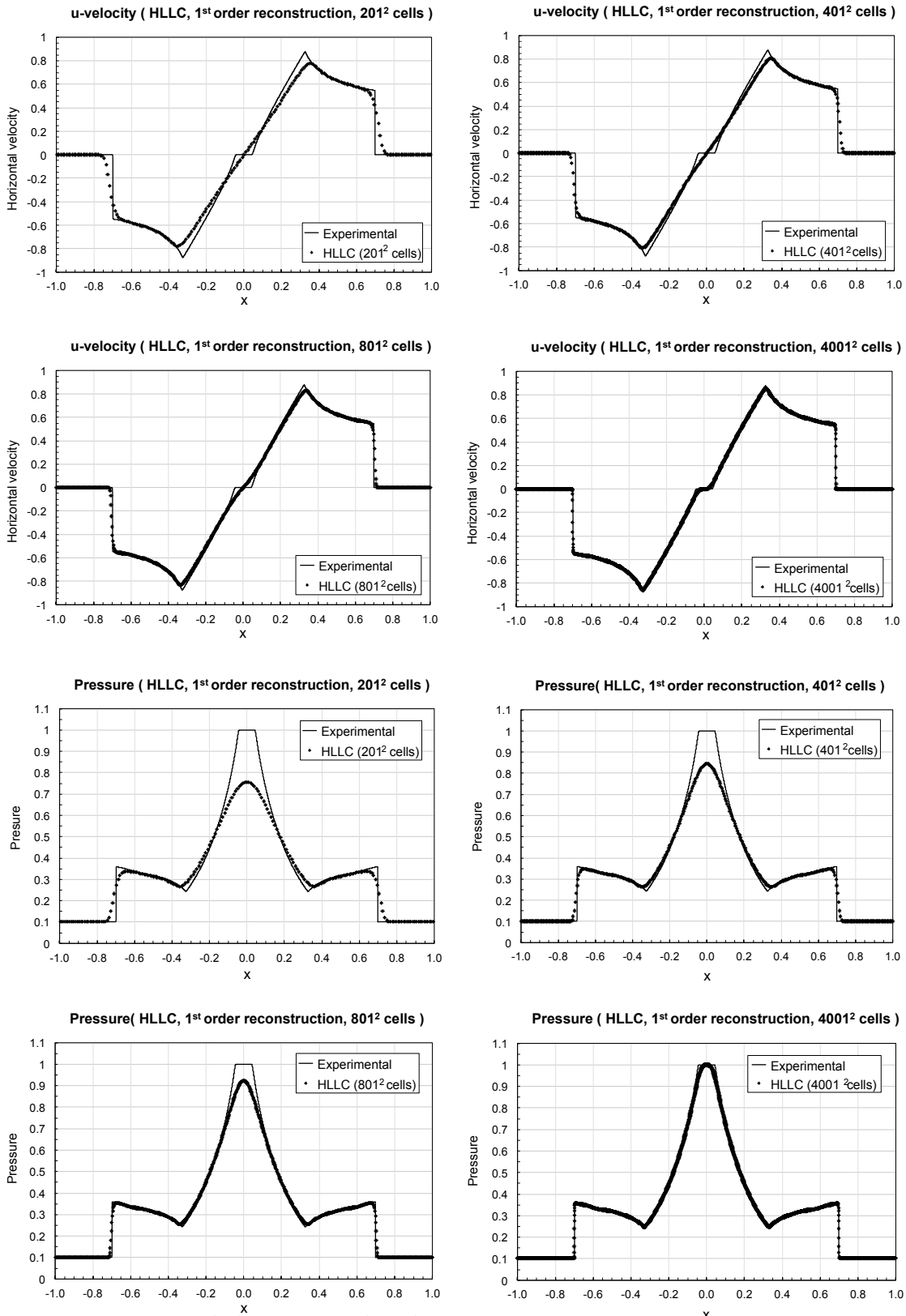


Figure 4.7: U-velocity(up) and pressure(down) along the line $y = 0$ obtained by using the Godunov method in conjunction with the HLLC approximate Riemann solver (symbols) compared to the reference data (line).

Quantitative comparison

- **Density:**

As it has been justified before, the absolute and relative error mean are used so as to study the convergence of the solution with the grid refinement.

	Maximum absolute error $\lvert\rho_{HLLC}-\rho_{ref}\rvert_{max}$	Location of the maximum absolute error $x_{\lvert\rho_{HLLC}-\rho_{ref}\rvert_{max}}$	Maximum relative error (%) $\left\lvert\frac{\rho_{HLLC}-\rho_{ref}}{\rho_{ref}}\right\rvert_{max}$	Location of the maximum relative error $x_{\left\lvert\frac{\rho_{HLLC}-\rho_{ref}}{\rho_{ref}}\right\rvert_{max}}$	Absolute error mean $\overline{\lvert\rho_{HLLC}-\rho_{ref}\rvert}$	Relative error mean (%) $\overline{\left\lvert\frac{\rho_{HLLC}-\rho_{ref}}{\rho_{ref}}\right\rvert}$
201^2 cells	0.3688	0.7065	73.76 %	0.7065	**0.056795**	**7.8693 %**
401^2 cells	0. 3710	0.7032	74.20 %	0.7032	**0.035407**	**5.0234 %**
801^2 cells	0. 4456	0.6991	89.12 %	0.6991	**0.022928**	**3.3129 %**
4001^2 cells	0. 4048	0.6988	83.60 %	0.6988	**0.006950**	**1.0134 %**

Table 4.4: Absolute error maximum value and its location, maximum relative error and its location and absolute and relative error means of the density when compared to the reference data (HLLC approximate Riemann solver, 1^{st} order reconstruction) .

In Figure 4.8, it has been found a correlation for the absolute and relative error as a function of the grid refinement. Once again, the most accurate correlation was obtained by using a Power Law.

Remarks:

1. It can be observed that the pressure solution converges with the grid refinement.
2. When the number of cells is doubled, the absolute error mean is divided by 1.63 and the relative error mean by 1.81. Both of these number greater than the values obtained for the HLL.
3. For the finest grid, the mean absolute error is lower than just 0.007 and the mean relative error 1%.

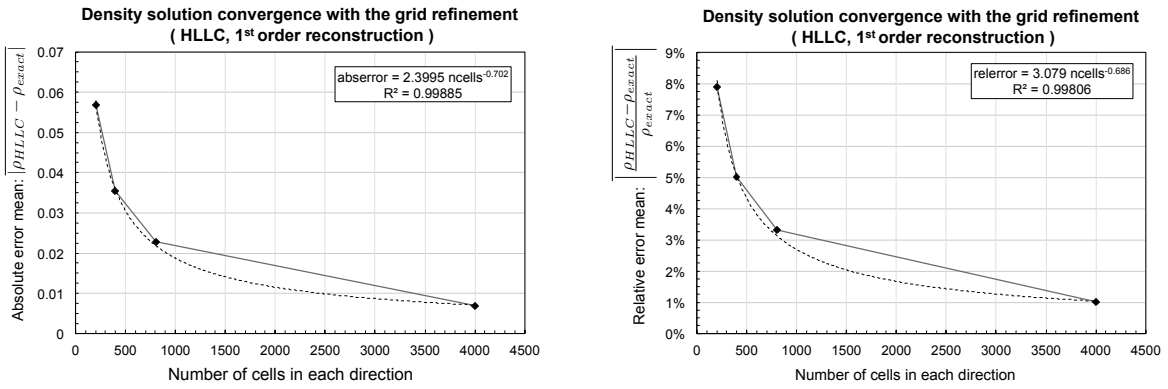


Figure 4.8: Pressure solution convergence with the grid refinement.

- **u-velocity**:

	Maximum absolute error $\lvert u_{HLLC} - u_{ref} \rvert_{max}$	Location of the maximum absolute error $x_{\lvert u_{HLLC} - u_{ref} \rvert_{max}}$	Absolute error mean $\overline{\lvert u_{HLLC} - u_{ref} \rvert}$
201^2 cells	0.3581	0.7065	**0.056795**
401^2 cells	0. 3537	0.7032	**0.035407**
801^2 cells	0.4059	0.6991	**0.022928**
4001^2 cells	0.3738	0.6988	**0.006950**

Table 4.5: Absolute error maximum value and its location, maximum relative error and its location and absolute and relative error means of the u-velocity when compared to the reference data (HLLC approximate Riemann solver, 1^{st} order reconstruction) .

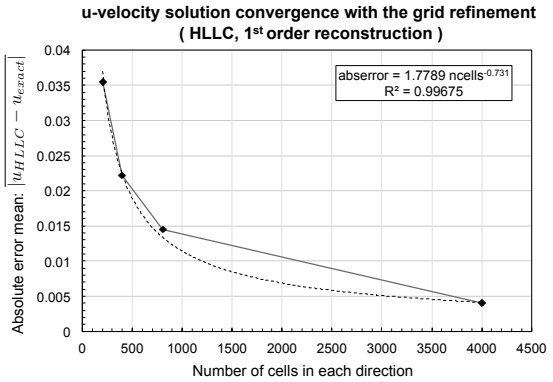


Figure 4.9: Velocity solution convergence with the grid refinement.

Remarks:

1. It can be observed that the velocity solution converges with the grid refinement.
2. When the number of cells is doubled, the absolute error mean is divided by 1.64.
3. For the finest grid, the mean absolute error is lower than just 0.005 .

- **Pressure:**

	Maximum absolute error $\lvert p_{HLLC} - p_{ref} \rvert_{max}$	Location of the maximum absolute error $x_{\lvert p_{HLLC} - p_{ref} \rvert_{max}}$	Maximum relative error (%) $\left\lvert \frac{p_{HLLC} - p_{ref}}{p_{ref}} \right\rvert_{max}$	Location of the maximum relative error $x_{\left\lvert \frac{p_{HLLC} - p_{ref}}{p_{ref}} \right\rvert_{max}}$	Absolute error mean $\overline{\lvert p_{HLLC} - p_{ref} \rvert}$	Relative error mean (%) $\overline{\left\lvert \frac{p_{HLLC} - p_{ref}}{p_{ref}} \right\rvert}$
201^2 cells	0.2635	0.0398	134.0 %	0.7065	**0.033402**	**8.0499 %**
401^2 cells	0. 1921	0.0448	133.4 %	0.7032	**0.020667**	**4.5313 %**
801^2 cells	0. 1633	0.6991	163.3 %	0.6991	**0.012738**	**2.8960 %**
4001^2 cells	0. 1465	0.6988	156.9 %	0.6999	**0.003085**	**0.6251 %**

Table 4.6: Absolute error maximum value and its location, maximum relative error and its location and absolute and relative error means of the pressure when compared to the reference data (HLLC approximate Riemann solver, 1^{st} order reconstruction) .

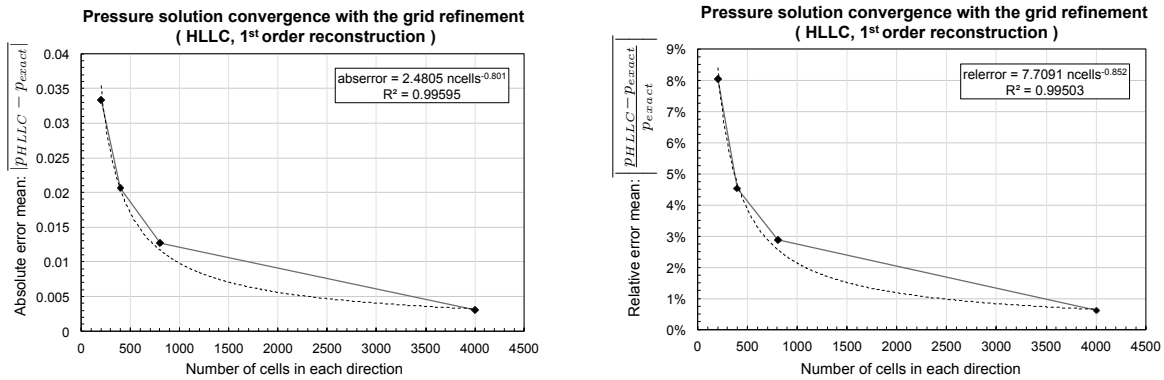


Figure 4.10: Pressure solution convergence with the grid refinement.

Remarks:

1. It can be observed that the pressure solution converges with the grid refinement.
2. When the number of cells is doubled, the absolute error mean is divided by 1.74 and the relative error mean by 1.81. Both of these number greater than the values obtained for the HLL.
3. For the finest grid, the mean absolute error is just 0.003% and the mean relative error 0.6%.

4.1.3 Comparison between HLL and HLLC

Global accuracy

In Figure 4.11, it can be observed that the absolute and relative error mean is always lower for the HLLC when compared to the HLL for all the variables studied.

The coarse the mesh is, the most significant this difference is. The improvement obtained is of the order of 0.5 % for the coarsest grid.

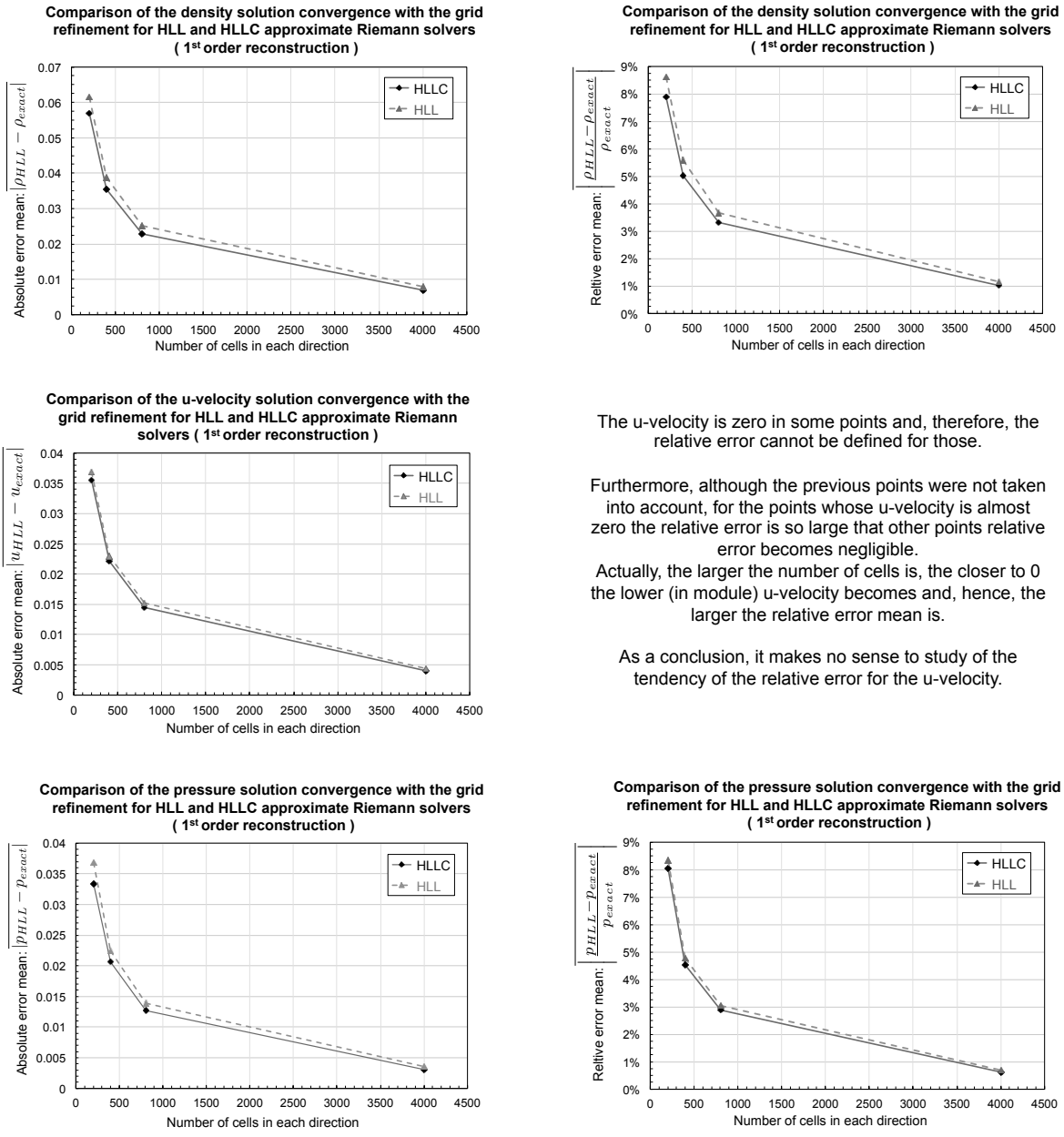


Figure 4.11: Comparison between the absolute and relative error mean obtained with the HLL and The HLLC approximate Riemann Solvers.

Identification of the regions where this improvement is more significant

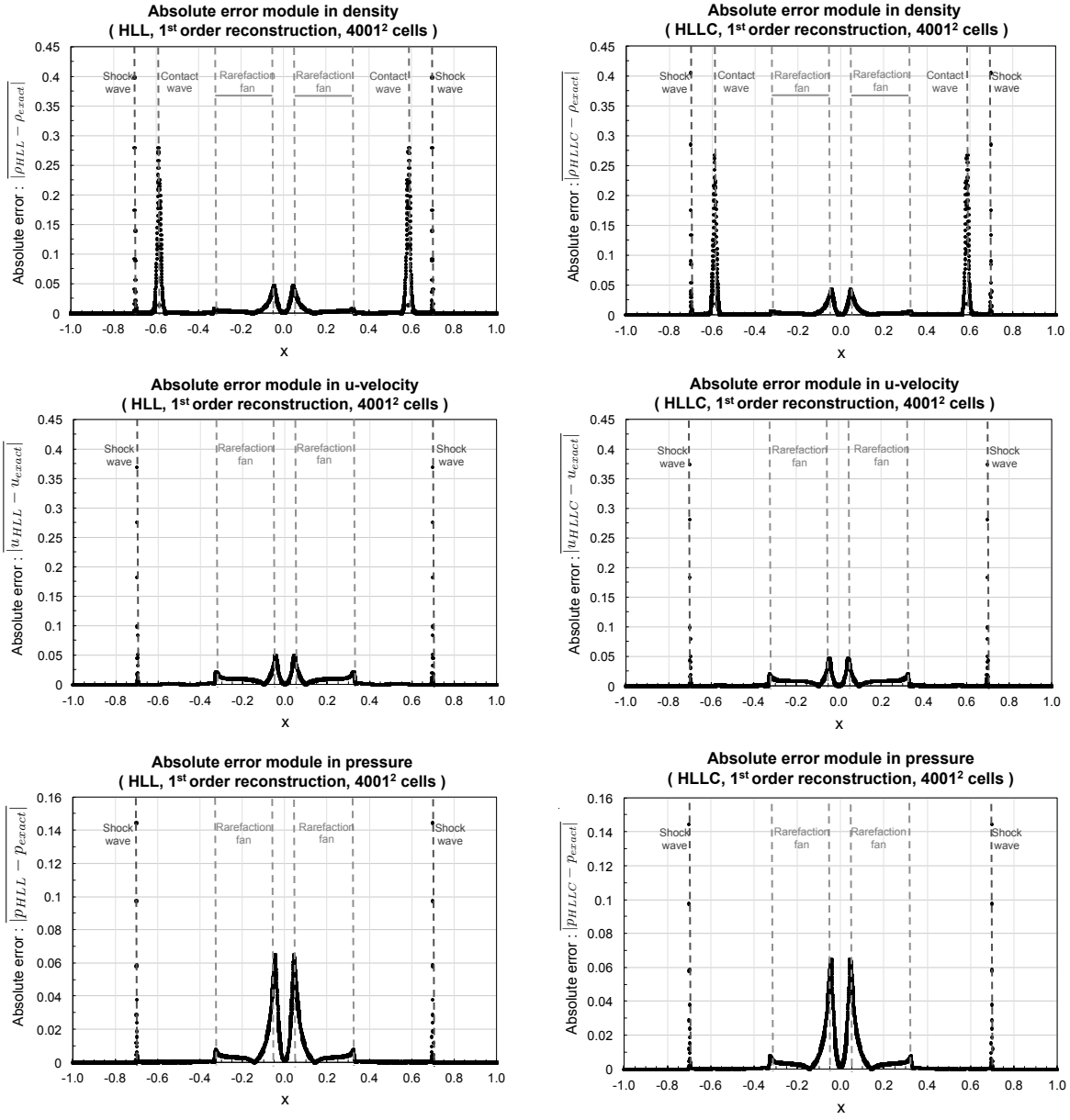


Figure 4.12: Identification of the regions where this improvement is more significant with the aid of the absolute error.

For both cases, the regions where the error is significant are:

- Shock wave.
- Contact wave, for pressure.
- Head of the rarefaction fan.
- Tail of the rarefaction fan.

Since it is the variable with more problematic regions, those are studied in detail for density.

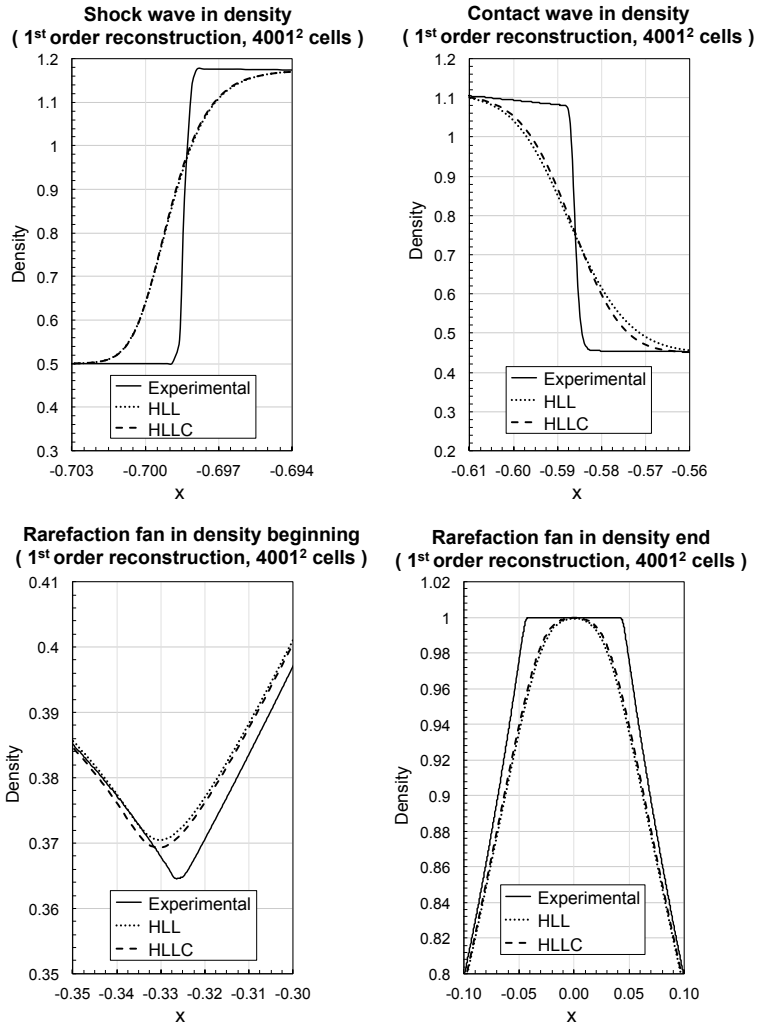


Figure 4.13: Comparison of the density obtained with the HLL and the HLLC in the critical regions (4001^2 cells).

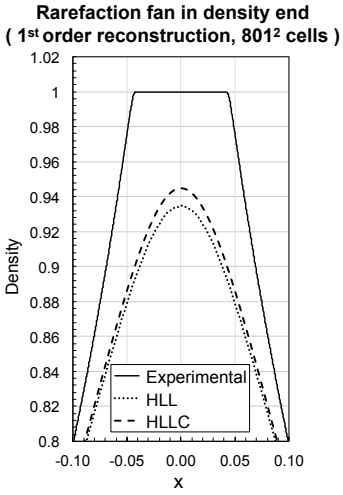


Figure 4.14: Comparison of the density obtained with the HLL and the HLLC at rarefaction head (801^2 cells).

Firstly, both HLL and HLLC satisfy excellently the entropy property.

Secondly, at it was seen in previous subsections, the rate of decay of the absolute error with the grid refinement is slightly $(1 - 3\%)$ higher for the HLLC.

Finally, for the other coarser grids, the maximum difference is found at the rarefaction head, as it can be seen in Figure 4.14.
However, as expected, as soon as the mesh is fine enough the higher differences are found in the contact wave.For example, when using the 4001^2 cells mesh, which has been run on The Grid, the contact wave is solved in a 5% less cells.

The reason is that the HLLC, because in its wave structure it incorporates an intermediate wave, it has much less numerical dissipation for slowly moving contact waves and it gives infinite resolution for stationary contact waves [7]. In this case, although the contact wave is moving pretty fast, the difference between both schemes can be appreciated.

Finally, the error is 5 to 10 % lower, depending on the grid and variable studied.

4.2 Reconstruction

4.2.1 Comparison between the 1^{st} and 2^{nd} order reconstruction for the HLLC approximate Riemann solver

Global comparison

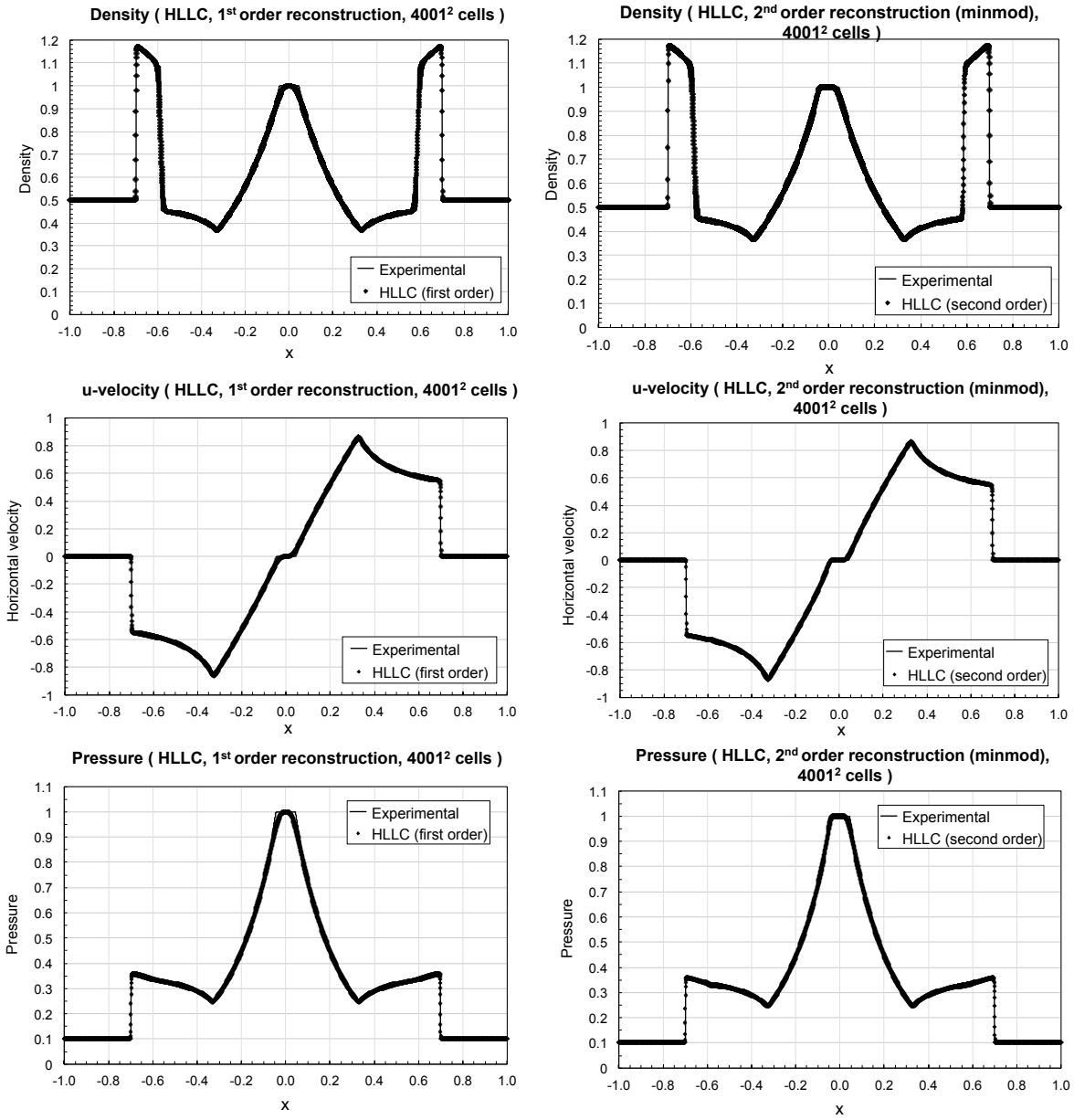


Figure 4.15: Comparison between the solution obtained with 1^{st} and 2^{nd} order reconstruction (MINMOD) in conjunction with the HLLC solver.

With the finest grid, with both reconstruction the numerical results are extraordinarily closer to the reference. Therefore, the absolute error is analysed so as to detect the discrepancies.

Identification of the regions where this improvement is more significant

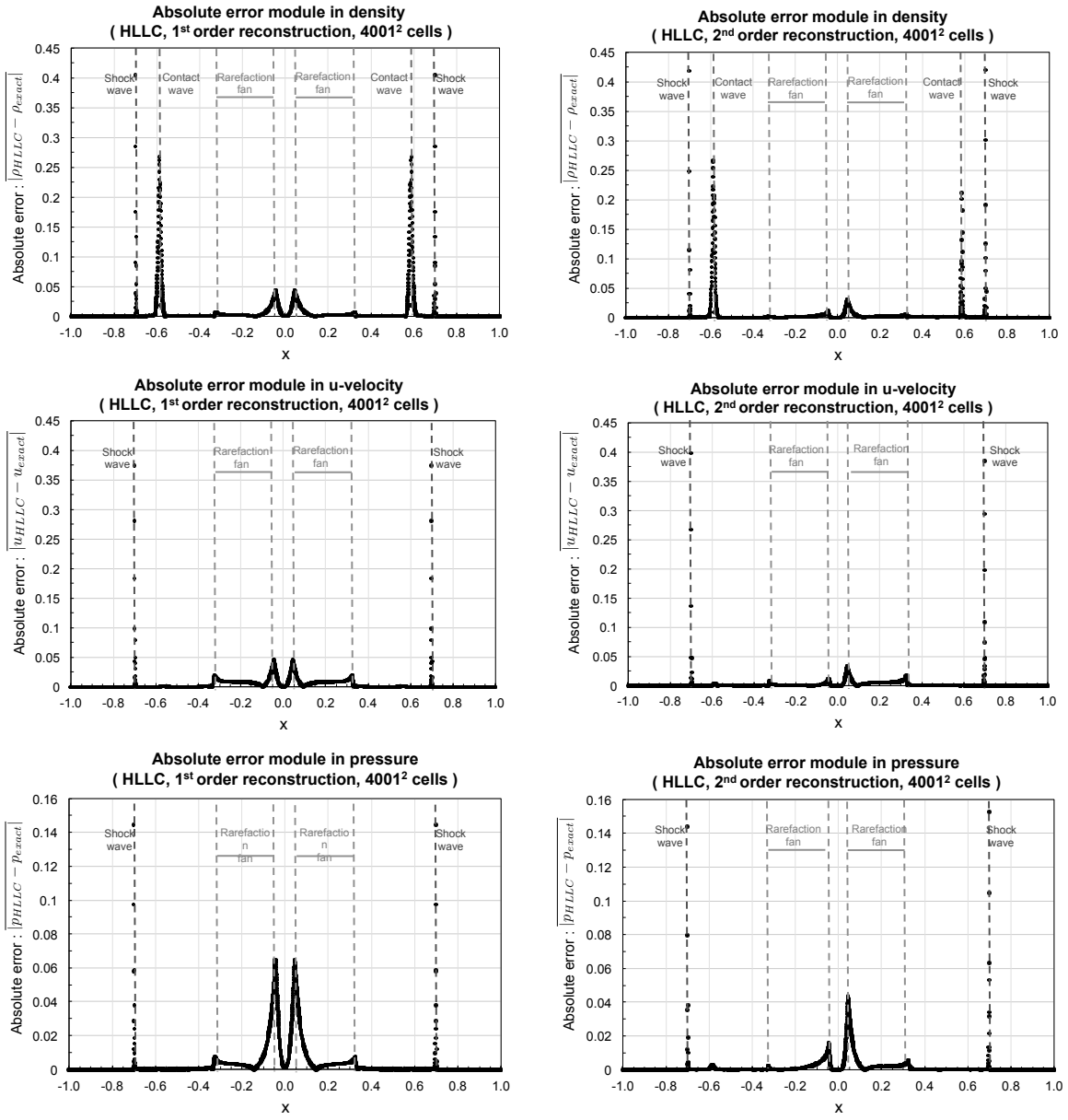


Figure 4.16: Comparison between the absolute errors obtained with 1^{st} and 2^{nd} order reconstruction (MINMOD) in conjunction with the HLLC solver.

Remarks:

1. The absolute error is clearly lower for the 2^{nd} order reconstruction. The error is reduced up to an 95%, specially in the rarefaction wave head and tail.

2. The absolute error distribution is symmetric for the 1^{st} order reconstruction, but not when using the 2^{nd} order reconstruction with the MINMOD slope limiter, mainly since the min function:

$$\text{minmod} = \frac{1}{2}\left[sign(x) + sign(y)\right] \min\left(|x|, |y|\right) \tag{4.1}$$

3. For both cases, the regions where the error is significant are:
    - Shock wave.
    - Contact wave, for pressure.
    - Beginning of the rarefaction fan.
    - End of the rarefaction fan.

Since it is the variable with more problematic regions, those are studied in detail for density.

Qualitative and quantitative comparisons

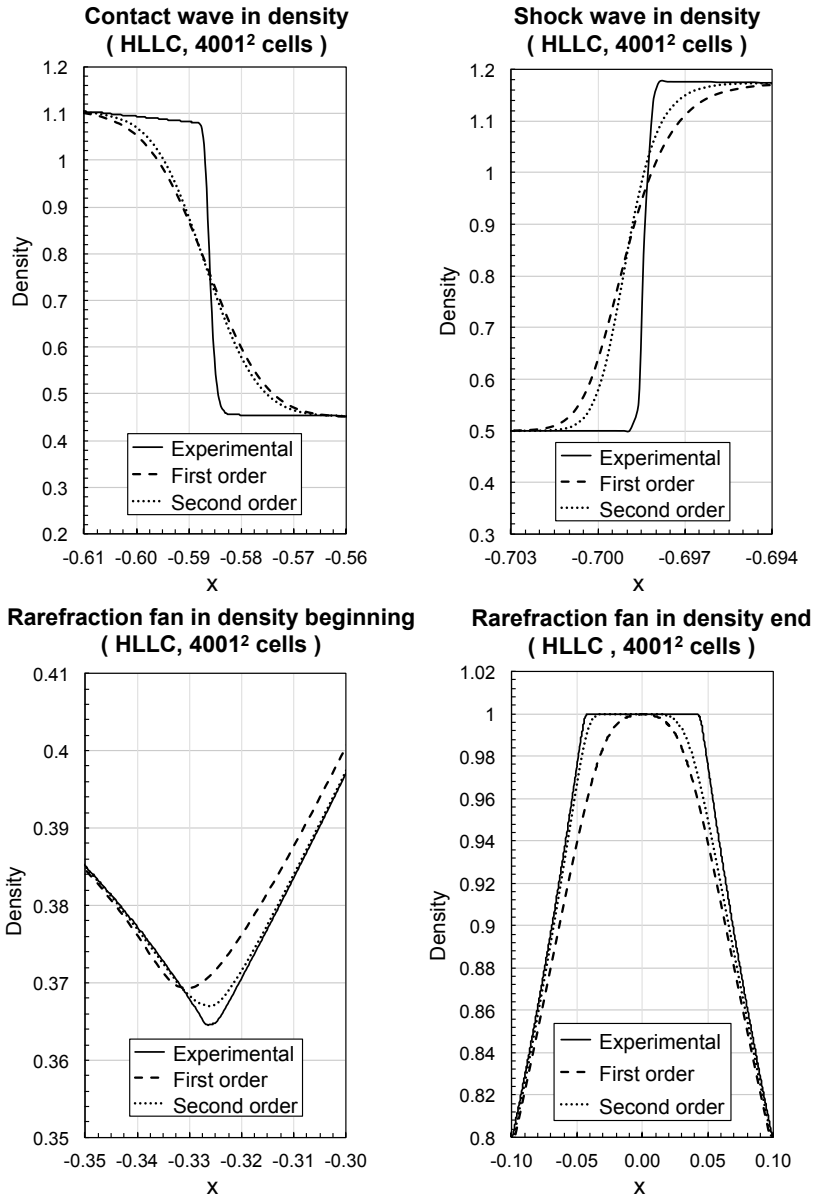


Figure 4.17: Comparison of the density 1^{st} and 2^{nd} order reconstruction (MINMOD) in conjunction with the HLLC solver in the critical regions.

		Maximum absolute error $\lvert\phi_{HLLC} - \phi_{ref}\rvert_{max}$	Location of the maximum absolute error $x_{\lvert\phi_{HLLC}-\phi_{ref}\rvert_{max}}$	Maximum relative error (%) $\left\lvert\frac{\phi_{HLLC} - \phi_{ref}}{\phi_{ref}}\right\rvert_{max}$	Location of the maximum relative error $x_{\left\lvert\frac{\phi_{HLLC}-\phi_{ref}}{\phi_{ref}}\right\rvert_{max}}$	Absolute error mean $\lvert\phi_{HLLC} - \phi_{ref}\rvert$	Relative error mean (%) $\left\lvert\frac{\phi_{HLLC} - \phi_{ref}}{\phi_{ref}}\right\rvert$
Density	1st order reconstruction	0. 4048	0.6988	83.60 %	0.6988	**0.006950**	**1.0134 %**
	2nd order reconstruction	0.4196	0.6988	86.66 %	0.6988	**0.003921**	**0.5803 %**
u-velocity	1st order reconstruction	0.3738	0.6988	-	-	**0.006950**	**-**
	2nd order reconstruction	0.3976	-0.6988	-	-	**0.002075**	**-**
Pressure	1st order reconstruction	0. 1465	0.6988	156.90 %	0.6999	**0.003085**	**0.6251 %**
	2nd order reconstruction	0.1524	0.6988	163.27 %	0.6988	**0.001535**	**0.3982 %**

Table 4.7: Comparison of the absolute error maximum value and its location, maximum relative error and its location and absolute and relative error means between 1^{st} and 2^{nd} order reconstruction (MINMOD) in conjunction with the HLLC solver for the 4001^2 cells grid .

Remarks:

1. In Figure 4.17, it can be confirmed that the 2^{nd} order reconstruction is much more accurate than the 1^{st} order reconstruction. This improvement is specially outstanding in the rarefaction head and tail.
2. Even for a very fine mesh, the absolute error mean is reduced in a:
   - 44% for density.
   - 70% for velocity.
   - 50% for pressure.
3. The relative error mean is reduced in a:
   - 43% for density.
   - 36% for pressure.
4. The number of points required to solve the shock wave is also reduced almost a 10%.
5. Although 2^{nd} order reconstructions brings spurious oscillations, there is no oscillation thanks to the slope limiter.

4.3 3D plots

Density

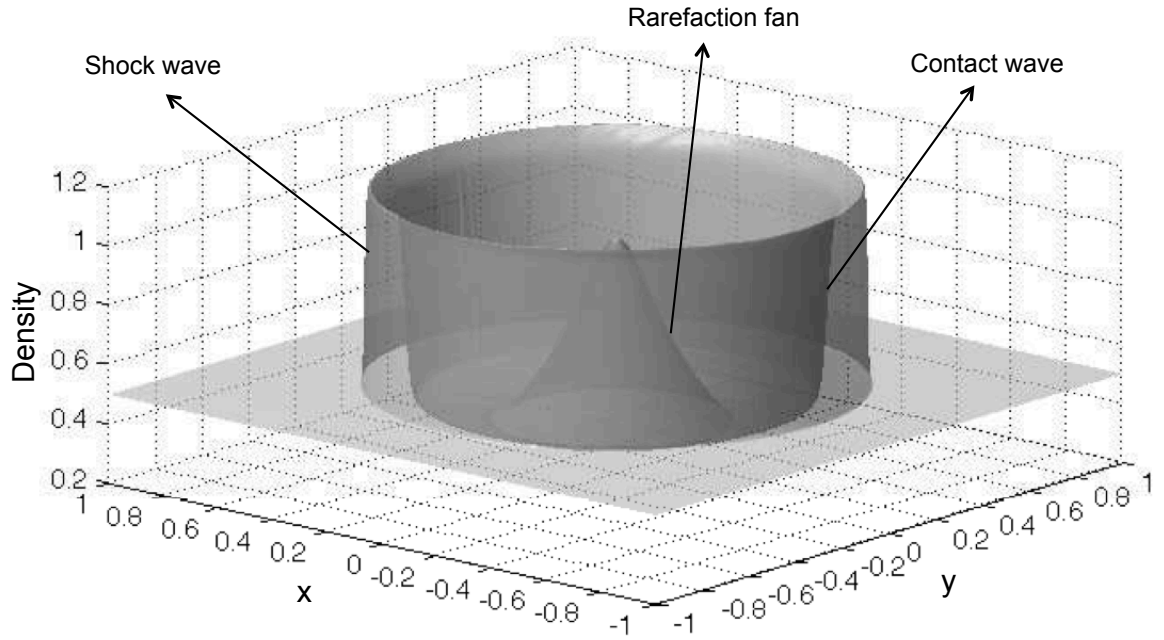


Figure 4.18: Density solution obtained with the Gudonov's Method in conjunction with the HLLC Riemann Solver and a 2^{nd} order reconstruction with the MINMOD slope limiter, for a 801^2 cells grid.

Velocity module

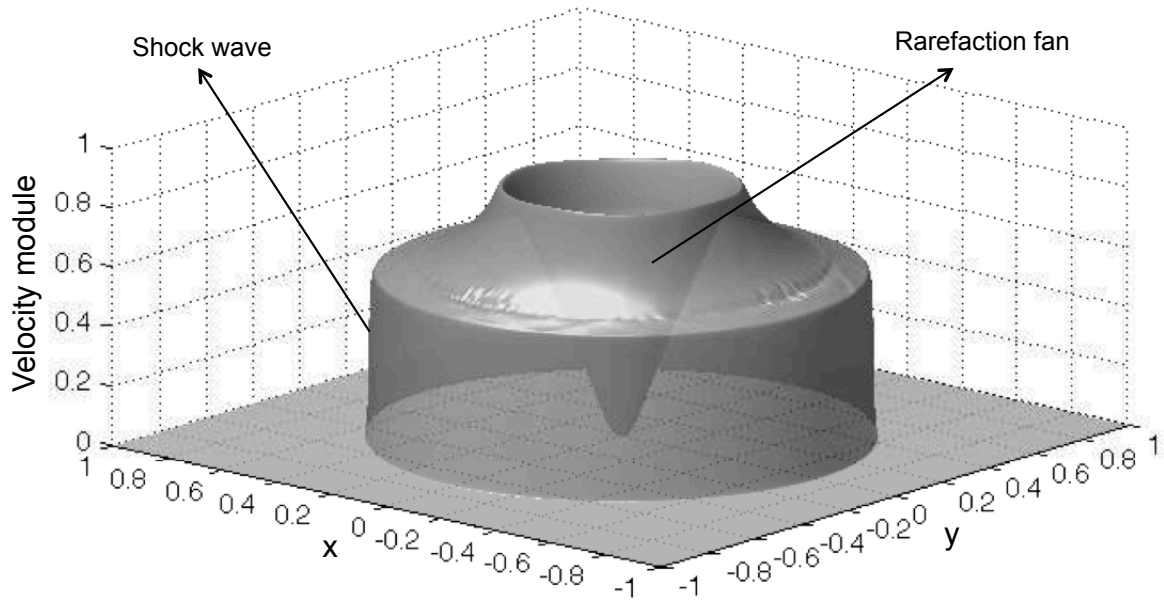


Figure 4.19: Velocity module solution obtained with the Gudonov's Method in conjunction with the HLLC Riemann Solver and a 2^{nd} order reconstruction with the MINMOD slope limiter, for a 801^2 cells grid.

Pressure

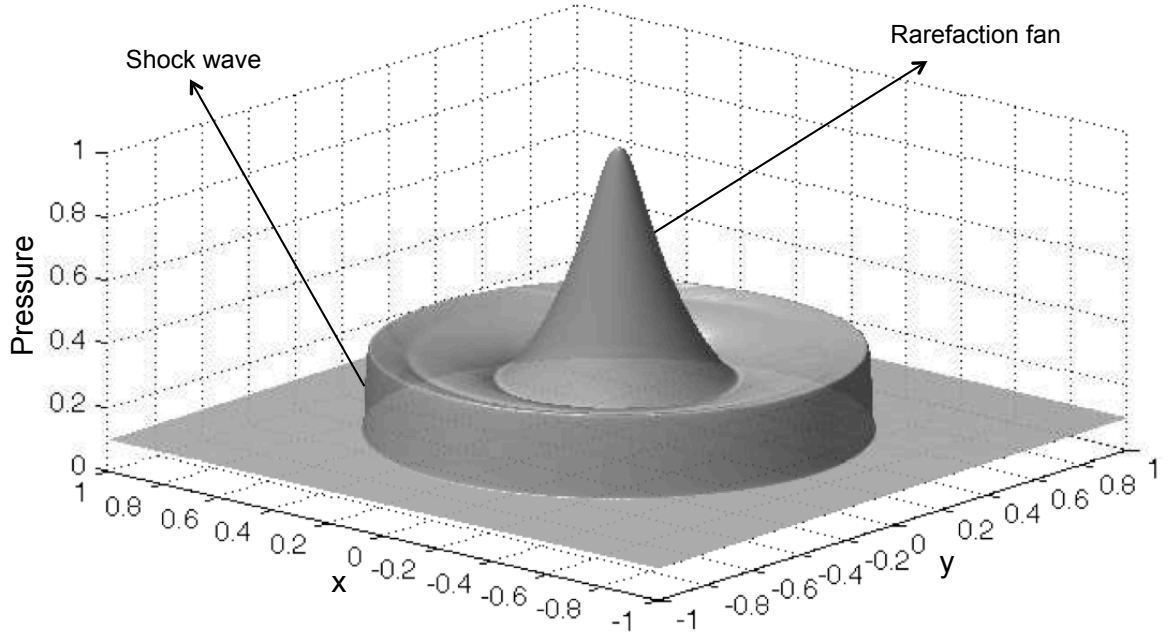


Figure 4.20: Pressure solution obtained with the Gudonov's Method in conjunction with the HLLC Riemann Solver and a 2^{nd} order reconstruction with the MINMOD slope limiter, for a 801^2 cells grid.

In Figures 4.18, 4.19 and 4.20, it can be seen that, at $t = 0.3$, the solution exhibits:

- A circular **shock wave** traveling away from the center. It can be observed that across the shock wave the density and the pressure increase.
- A circular **rarefaction fan** traveling to the center, thought which the density and pressure decrease while the velocity module increases.
- A circular **contact wave** traveling to the center.

  Note that velocity should be continuous across a contact wave (otherwise, a vacuum could be generated) and pressure should be also continuous (otherwise, it would move until this condition was true.) However, there is a slightly perturbation in this region due to a small numerical error.

By using the animation generated by the program, it can be seen how the circular shock wave travels outwards, becoming weaker as time evolves. The contact surface follows the shock becoming weaker also; at some point in time the contact comes to rest and then travels inwards. The rarefaction traveling towards the center reflects, as a rarefaction, and over expands the flow so as to create an inwards traveling sock wave; this circular shock wave implodes into the origin, reflects and travels outwards colliding with the contact surface. And so on.

The results match brilliantly the expected ones, explained in Toro [11] (page 583).

4.4 Double-check of the most accurate combination of Riemann solver/reconstruction/mesh

4.4.1 Most accurate combination of RS, reconstruction & mesh

Taking into account the previous sections, the most accurate combination is given by:

- **Mesh:** 4001^2 cells.
- **RS:** HLLC
- **Reconstruction:** 2^{nd} order, with MINMOD slope limiter.

4.4.2 Double-check

Initial conditions and reported time

Density:

$$\rho\left(x,y,0\right)=\begin{cases}1.0\ , & \text{if } r\leq 0.4\\ \rho_0\ , & \text{if } \ r>4\end{cases},\quad r^2=x^2+y^2 \tag{4.2}$$

Velocity:

$$u\left(x,y,0\right)=v\left(x,y,0\right)=0 \tag{4.3}$$

Pressure:

$$p\left(x,y,0\right)=\begin{cases}1.0\ , & \text{if } r\leq 0.4\\ p_0\ , & \text{if } \ r>4\end{cases},\quad r^2=x^2+y^2 \tag{4.4}$$

Density ρ_0	Pressure p_0	Output time T
0.12	0.2	0.23

Table 4.8: Parameters for the initial condition and reported time.

Boundary conditions

The boundary conditions are **transmissive**.

Qualitative and quantitative comparison

In Figure 4.21 (left), it can be seen that the numerical solution are so close to the reference results that, with the naked eye, it is almost impossible to distinguish between them.

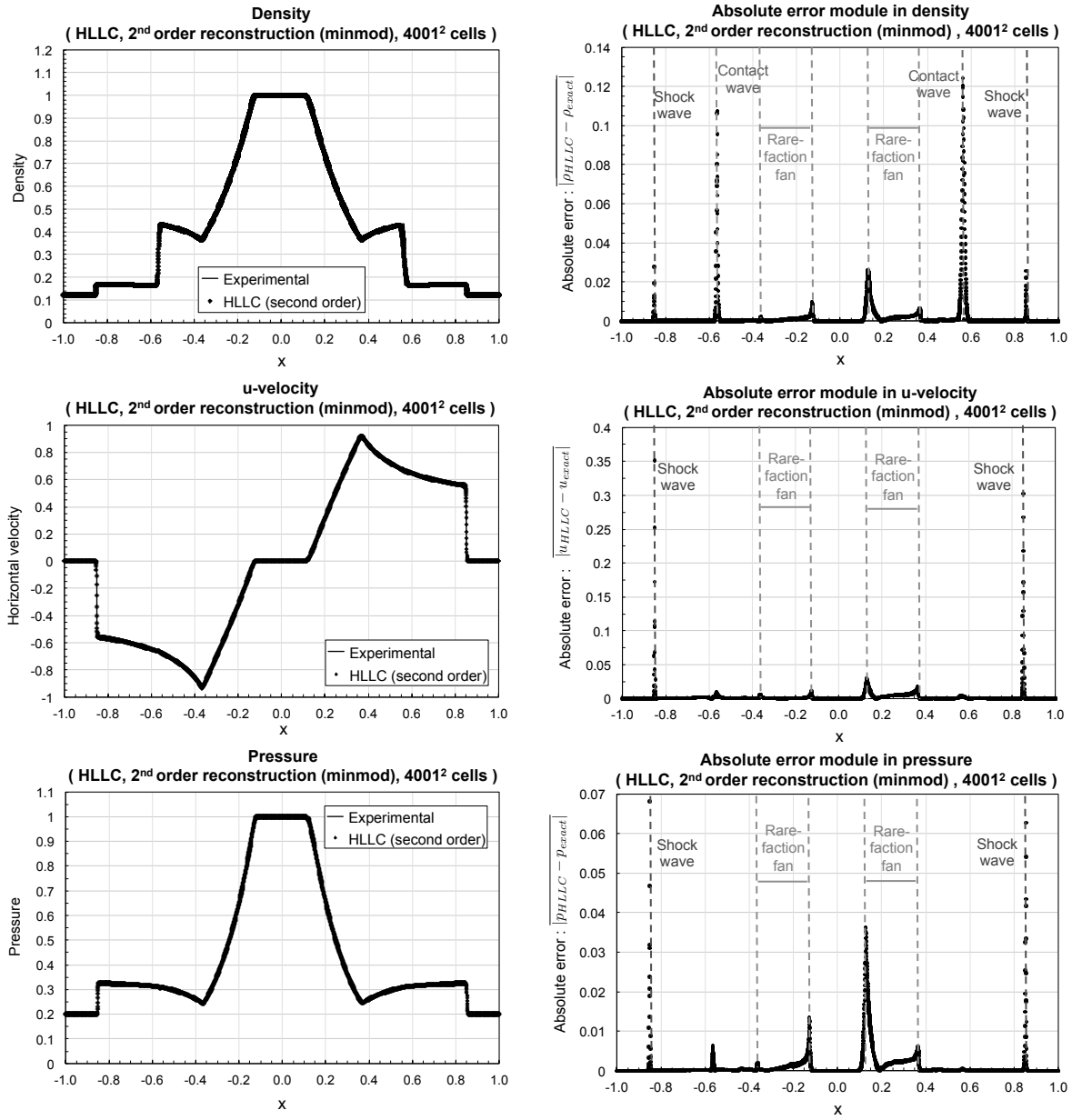


Figure 4.21: Comparison between the results obtained with the HLLC Riemann solver in conjunction with the 2^{nd} order reconstruction with the MINMOD slope limiter (left) and the absolute error distribution (right).

	Maximum absolute error $\lvert\phi_{HLLC} - \phi_{ref}\rvert_{max}$	Location of the maximum absolute error $x_{\lvert\phi_{HLLC}-\phi_{ref}\rvert_{max}}$	Maximum relative error (%) $\left\lvert\frac{\phi_{HLLC} - \phi_{ref}}{\phi_{ref}}\right\rvert_{max}$	Location of the maximum relative error $x_{\left\lvert\frac{\phi_{HLLC}-\phi_{ref}}{\phi_{ref}}\right\rvert_{max}}$	Absolute error mean $\overline{\lvert\phi_{HLLC} - \phi_{ref}\rvert}$	Relative error mean (%) $\overline{\left\lvert\frac{\phi_{HLLC} - \phi_{ref}}{\phi_{ref}}\right\rvert}$
Density	0.1243	0.5634	65.47 %	0.5659	**0.001950**	**0.6214 %**
u-velocity	0.3519	-0.8513	-	-	**0.002294**	**-**
Pressure	0.0682	-0.8513	34.68%	-0.8513	**0.001070**	**0.2357 %**

Table 4.9: Comparison of the absolute error maximum value and its location, maximum relative error and its location and absolute and relative error means (HLLC, 2^{nd} order reconstruction(MINMOD), 4001^2 cells).

Regarding the errors in Figure 4.21 (right) and Table 4.9, it is worth to point out:

1. The regions where the error is significant keeps being the same as for the previous case:
   - Shock wave.
   - Contact wave, for pressure.
   - Head and tail of the rarefaction fan.

   However, for the rest the absolute error is of the order of 10^{-3}.
2. Just for density, the absolute error near the contact wave is larger than near the shock wave. The reason is that the shock wave is weaker.
3. Whether the maximum absolute error occurs near the shock or the contact wave, its value is half the change in the variable value across the wave since the discontinuity needs several points to be solved.
4. As expected, the absolute error distribution is not symmetric (See Section 4.2 for explanation).
5. Since in this second test the changes across this discontinuities is less sharp, the absolute and relative error are 30-50 % lower than in the previous test.
   The mean relative error is just 0.6 % for density and 0.2% for pressure.

Chapter 5

CONCLUSIONS

In order to improve the accuracy of the Gudonov's Method with an adequate computational cost, in this report they have been studied:

- The use of **approximate Riemann Solvers** (such as, for example, the HLL or the HLLC), which can increase significantly the efficient of the scheme, making each iteration faster.

  On the one hand, both HLL and HLLC satisfy excellently the entropy property.

  On the other hand, although both Riemann solvers converge with the grid refinement, the rate of decay of the absolute error with the grid refinement is slightly $(1 - 3\%)$ higher for the HLLC. Furthermore, as soon as the mesh is fine enough the highest differences are found in the contact wave. For example, the 4001^2 cells mesh, which has been run on The Grid, the contact wave is solved in a 5% less cells.

  The reason is that the HLLC, since in its wave structure it incorporates an intermediate wave, it has much less numerical dissipation for slowly moving contact waves and it gives infinite resolution for stationary contact waves. In this case, although the contact wave is moving pretty fast, the difference between both schemes can be appreciated.

  In addition, the error is 5 to 10 % lower, depending on the grid and variable studied.

- The increase of the order of the scheme by using a **2^{nd} order reconstruction**. When using the HLLC RS and the 4001^2 cells grid, the absolute error is reduced up to 95%, and its mean is reduced from 45% to 70% depending on the variable. Moreover, the number of points required to solved the shock and contact wave is reduced an 8%.

  Last but not least, although 2^{nd} order reconstructions brings spurious oscillations, there is no oscillation thanks to the Slope Limiter (in this case, MINMOD).

References

[1] R. Kerrod. *"Hubble: the mirror on the universe"*. David & Charles, January 2003.

[2] R.J.Leveque. *"Computational Methods for Astrophysical Fluid Flow"*. Springer, 1998.

[3] E.F. Toro. *"Notions on Numerical Methods", in "Riemann Solvers and Numerical Methods for fluid dynamics"*. Springer-Verlag, 2nd ed. edition, 1999.

[4] S.K. Gudonov. "A difference scheme for numerical solution of discontinuous solution of hydrodynamic equations". *Math. Sbornik*, 47:271–306, 1959.

[5] E.F. Toro. *"The Riemann Problem for the Euler Equations", in "Riemann Solvers and Numerical Methods for fluid dynamics"*. Springer-Verlag, 2nd ed. edition, 1999.

[6] E.F. Toro. *"The Method of Gudonov for Non-linear systems", in "Riemann Solvers and Numerical Methods for fluid dynamics"*. Springer-Verlag, 2nd ed. edition, 1999.

[7] E.F. Toro. *"The HLL and HLLC Riemann Solvers", in "Riemann Solvers and Numerical Methods for fluid dynamics"*. Springer-Verlag, 2nd ed. edition, 1999.

[8] Toro E F. "HLLC riemann solver". Malaga, Spain, February 2010. Math School on Numerical Solutions of Partial Differential Equations.

[9] P.K. Sweby. "High resolution schemes using flux-limiters for hyperbolic conservation laws". *SIAM J. Num. Anal.*, 21(5):995–1011, 1984.

[10] V.P. Kolgan. "Numerical schemes for discontinuous problems of gas dynamics based on minimization of the solution gradient". *Report*, 1972.

[11] E.F. Toro. *"Multidimensional test problems", in "Riemann Solvers and Numerical Methods for fluid dynamics"*. Springer-Verlag, 2nd ed. edition, 1999.

Chapter 6

APPENDIX

Contents

6.1 FORTRAN code (Gudonov's Method)

```
! ********************************************************************
! Project_Espinosa.f90
!
!                      ------------------------------
!                          Silvia ESPINOSA-GUTIEZ
!                           MIT ID: 919 277 866
!                      ------------------------------
!
! Description:
! Finite-volume code for 2D compressible Euler equations
! Fluxes: HLL and HLLC approximate Riemann solvers
! Reconstructions (2nd order reconstruction,with MONMOD slope limiter)
!
! 18.086 Computational Science and Engineering II
! Spring 2013
! ********************************************************************


IMPLICIT NONE

!----------------------
! variable declaration
!----------------------

! number of cells in the spatial domain
Integer   Nx,Ny
! spatial order & flux type
Integer   SpatialOrder,Fluxtype
! output time
Real  TIME
! initial values of density and pressure
Real RhoInf,Pinf
! CFL number
Real ::  CFL = 0.35

! spatial domain size
Real, Parameter:: Lx = -1.0, Rx = 1.0, Ly = -1.0, Ry = 1.0
Real hx,hy

! gamma = cp/cv
Real, Parameter ::  gamma = 1.4


! Vector of conservative variables CSV(rho,rhou,rhov,E)
!    First  index: RK stage
!    Second index: conservative variable
```

```
!    Third and forth: i,j - spatial coordinates
! Vector of  primitive variables, PV = (rho,u,v,P)
Real, allocatable ::   CSV(:,:,:,:),  PV(:,:,:), FX(:,:,:,:),
FY(:,:,:,:), FluxTotal(:,:,:)

Integer RkStage
! time step dt & flow time t_
Real    DT,T_
! time step counter
Integer IT

! ---------------------------------- START THE PROGRAM----------------
------

print*
print*,'  Home assignment for MSC in CFD 2012'
print*,'  CFD for steady & unsteady compressible flows'

print*
print*,'  Initialise the data'
print*

! initialise everything
CALL INITALL

print*,' Code uses ',nx,' cells in each direction'
print*,' Spatial Order:',SpatialOrder
print*,' Flux type:',FluxType


print*, '  Commense time marching'

! Commense time marching
IT=0
Do

    ! calculate a stable time step
    Call TimeStep
    ! Advance one time step using 2nd order RK method
    CALL RungeKutta

    ! advance time and time counter
    T_=T_ +DT
    IT = IT+1
    If ( MOD(it,10) ==0) Print*,' it= ',it,' t= ',T_

    !  check whether we reached the output time
     If ( ABS(T_ - TIME)/TIME .LE. 1e-5) GOTO 101
```

```
Enddo

101 CONTINUE

  Call Output
  Print*,' The job is done.'


  !////////////////////////////////////////////////////////////
Contains

  Subroutine    Output
    ! Purpose: to write the output files
    Integer i,j
    Real Y

    ! output for gnuplot
101 format(6(2x,F8.4))
    Open(1,file='2D.dat')
    Write(1,*)'       X          Y        RHO         |V|          P'
    Do i=1,nx
       Do j=1,ny
          Write(1,101)LX+i*hx-hx/2,LY+j*hy-
hy/2,PV(1,i,j),sqrt(PV(2,i,j)**2 + PV(3,i,j)**2),PV(4,i,j)
       Enddo
    Enddo
    close(1)

    ! 1d cut
202 format(6(2x,e11.4))
    Open(1,file='1D.dat')
    Write(1,*)'       X             RHO          U          P'
    j=(ny-1)/2+1
    Y = LY+j*hy-hy/2
    Do i=1,nx
       Write(1,202) LX+i*hx -hx/2,PV(1,i,j),PV(2,i,j),PV(4,i,j)
    Enddo
close(1)

    ! output for tecplot
303 format(3(2x,e13.6))
    Open(1,file='2Dtec.dat')
    WRITE(1,*)'TITLE="Density" '
    WRITE(1,*)'VARIABLES="X" "Y" "D"'
    WRITE(1,*)'ZONE ',',I=',NX, ',J=',ny,',F="POINT"'
    Do j=1,ny
       Do i=1,nx
          Write(1,303)LX+i*hx-hx/2,LY+j*hy-hy/2,PV(1,i,j)
```

```
      Enddo
    EndDo
    close(1)

  End subroutine Output


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SUBROUTINE TimeStep
    ! Purpose: calculate stable time step
    Real frac,sx,sy,a
    Integer i,j
    frac = 1d+9

    Do j=1,ny
       Do i=1,nx
          a = computesoundspeed(csv(1,:,i,j))
          sx =  abs(PV(2,i,j)) + a
          sy =  abs(PV(3,i,j)) + a
          frac = min(frac, hx/SX,hy/SY)
       Enddo
    Enddo

  ! set time step
  Dt = MIN(CFL*frac, TIME-T_)
  If ( IT<10)  Then
     Dt = MIN(0.1*CFL*frac, TIME-T_)
  Else
     Dt = MIN(CFL*frac, TIME-T_)
  Endif

END SUBROUTINE TimeStep


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Subroutine RungeKutta
  ! Purpose: advance by one time step using 2nd order RK method
  Integer i,j
  Real EnKin

  Do RkStage=1,2
     Call SetBc
     Call FluxX
     Call FluxY
     Call Update2
  Enddo

  CSV(1,:,:,: ) = CSV(3,:,:,:)
```

```
  ! compute primitive variables (second order) at time t_+dt
  RKSTAGE = 1
  Do i=1,nx
     Do j=1,ny
        PV(1,i,j) = CSV(RKSTAGE,1,i,j)
        PV(2,i,j) = CSV(RKSTAGE,2,i,j)/CSV(RKSTAGE,1,i,j)
        PV(3,i,j) = CSV(RKSTAGE,3,i,j)/CSV(RKSTAGE,1,i,j)
        EnKin     =  0.5*PV(1,i,j)*(PV(2,i,j)**2 + PV(3,i,j)**2)
        PV(4,i,j) = (Gamma-1)*( CSV(RKSTAGE,4,i,j) - EnKiN)
     Enddo
  Enddo

End subroutine RungeKutta


  ! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  Subroutine Update2
    ! Purpose: stage of RK method used in Update 2
    Integer i,j,k

    Do i=1,nx
       Do j=1,ny
          Do k=1,4
             FluxTotal(K,i,j) = - (FX(RKSTAGE,K,i,j) - FX(RKSTAGE,K,i-
1,j))/hx - (FY(RKSTAGE,K,i,j) - FY(RKSTAGE,K,i,j-1))/hy
          Enddo
       Enddo
    Enddo

    Select Case(Rkstage)

    Case(1)
       Do i=1,nx
          Do j=1,ny
             Do k=1,4
                CSV(2,K,i,j)  =  CSV(1,K,i,j)  + dt*FluxTotal(K,i,j)
             Enddo
          Enddo
       Enddo

    Case(2)
       Do i=1,nx
          Do j=1,ny
             Do k=1,4
                CSV(3,K,i,j)  =  0.5*CSV(1,k,i,j)  +
0.5*CSV(2,k,i,j)  + 0.5*dt*FluxTotal(k,i,j)
             Enddo
```

```
            Enddo
         Enddo

      CASE DEFAULT
         Print*,' Runge-Kutta ERROR'
         Read*
         STOP
      End select

    End subroutine Update2


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SUBROUTINE FluxX
    ! Purpose: for given RK stage compute the fluxes in the X
coordinate directon
    Integer i,j
    Real CDL(4),CDR(4),LocalFlux(4)

    Do i=0,nx
       Do j=1,ny

          FX(RKSTAGE,:,i,j) = 0.

          ! call the reconstruction operator
          Call Reconstruction(CSV(RKSTAGE,:,I-1:I+2,J),CDL,CDR)

          ! calculate the numerical flux using reconstructed conserved
vectors CDL, CDR
          Select Case(FluxType)
          Case(1)
             CALL Rusanov(CDL,CDR,LocalFlux)
          Case(2)
             CALL HLL(CDL,CDR,LocalFlux)
          Case(3)
             CALL HLLC(CDL,CDR,LocalFlux)
          Case default
             print*,' the flux is not defined. stop the program'
             read*
             stop
          End select

          FX(RKSTAGE,:,i,j) = LocalFlux

       Enddo
    Enddo
  END SUBROUTINE FluxX
```

```
  ! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  SUBROUTINE FluxY
    ! Purpose: for given RK stage compute the fluxes in the Y
coordinate directon
    Integer i,j
    Real CDL(4),CDR(4),LocalFlux(4),s

    Do i=1,nx
       Do j=0,ny

          FY(RKSTAGE,:,i,j) = 0.
          Call Reconstruction(CSV(RKSTAGE,:,I,J-1:J+2),CDL,CDR)

          ! rotate the data
          s = CDL(2)
          CDL(2) = CDL(3)
          CDL(3) = S
          s = CDR(2)
          CDR(2) = CDR(3)
          CDR(3) = S

          ! calculate the numerical flux using reconstructed conserved
vectors CDL, CDR
          Select Case(FluxType)
          Case(1)
             CALL Rusanov(CDL,CDR,LocalFlux)
          Case(2)
             CALL HLL(CDL,CDR,LocalFlux)
          Case(3)
             CALL HLLC(CDL,CDR,LocalFlux)
          Case default
             print*,' the flux is not defined. stop the program'
             read*
             stop
          End select

          ! rotate the flux back
          s = LocalFlux(2)
          LocalFlux(2) = LocalFlux(3)
          LocalFlux(3) = S

          FY(RKSTAGE,:,i,j) = LocalFlux

       Enddo
    Enddo
  END SUBROUTINE FluxY
```

```
 ! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Subroutine SetBC
   ! Purpose: set up transmissive boundary conditions at every RK
stage
   Integer i,j

   ! Horizontal sweep
   Do j=1,ny
      ! left boundary
      Do i=0,-1,-1
         CSV(rkstage,1,i,j) = CSV(rkstage,1, abs(i)+1, j);
         CSV(rkstage,2,i,j) = CSV(rkstage,2, abs(i)+1, j);
         CSV(rkstage,3,i,j) = CSV(rkstage,3, abs(i)+1, j);
         CSV(rkstage,4,i,j) = CSV(rkstage,4, abs(i)+1, j);
      Enddo

      ! right boundary
      Do i=1,2
         CSV(rkstage,1,nx+i,j) = CSV(rkstage,1, nx-i+1, j);
         CSV(rkstage,2,nx+i,j) = CSV(rkstage,2, nx-i+1, j);
         CSV(rkstage,3,nx+i,j) = CSV(rkstage,3, nx-i+1, j);
         CSV(rkstage,4,nx+i,j) = CSV(rkstage,4, nx-i+1, j);
      Enddo
   Enddo

   ! Y BOUNDARY
   Do i=1,nx
    ! bottom boundary
    Do j=0,-1,-1
       CSV(rkstage,1, i,j) = CSV(rkstage,1, i,abs(j)+1);
       CSV(rkstage,2, i,j) = CSV(rkstage,2, i,abs(j)+1);
       CSV(rkstage,3, i,j) = CSV(rkstage,3, i,abs(j)+1);
       CSV(rkstage,4, i,j) = CSV(rkstage,4, i,abs(j)+1);
    Enddo

    ! top boundary
    Do j=1,2
       CSV(rkstage,1,i,ny+j) = CSV(rkstage,1, i,ny-j+1);
       CSV(rkstage,2,i,ny+j) = CSV(rkstage,2, i,ny-j+1);
       CSV(rkstage,3,i,ny+j) = CSV(rkstage,3, i,ny-j+1);
       CSV(rkstage,4,i,ny+j) = CSV(rkstage,4, i,ny-j+1);
    Enddo
 Enddo

End subroutine SetBC
```

```
! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Subroutine InitAll
  ! Purpose: to read input parameters and initialise the run
  Integer I,J
  Real    x0,y0,QS(1:4),EnKin

  Open(1,file='2d.ini')
  Read(1,*) RhoInf
  Read(1,*) PInf
  Read(1,*) Time
  Read(1,*) SpatialOrder
  Read(1,*) Nx
  Read(1,*) FluxType
  Close(1)

  Ny = Nx

  Hx = (Rx-LX)/Nx
  Hy = (Ry-LY)/Ny

  Allocate(CSV(1:3,1:4,-1:nx+2, -1:ny+2),  PV(1:4,1:nx,1:ny),
FX(1:2,1:4,0:nx,1:ny), FY(1:2,1:4,1:nx,0:ny),
FluxTotal(1:4,1:nx,1:ny))

  RKSTAGE = 1
  Do I=1,Nx
     Do J=1,NY
        x0 = LX+i*hx - hx/2
        y0 = LY+j*hy - hy/2
        CALL  U0(x0,y0,QS)
        CSV(1,:,i,j) = QS
     Enddo
  Enddo

  Do  j=1,ny
     Do I=1,Nx
        PV(1,i,j) = CSV(1,1,I,j)
        PV(2,i,j) = CSV(1,2,I,j)/Csv(1,1,i,j)
        PV(3,i,j) = CSV(1,3,I,j)/Csv(1,1,i,j)
        EnKin =  0.5*PV(1,i,j)*(PV(2,i,j)**2 + PV(3,i,j)**2)
        PV(4,i,j) = (gamma-1)*(CSV(1,4,i,j) - EnKiN)
     Enddo
  Enddo

End subroutine InitAll


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Real function ComputeSoundSpeed(cds)
  ! Purpose: compute sound speed for the given vector of conserved
quantities

  Real cds(4),p,u ,v
  u = cds(2)/cds(1)
  v = cds(3)/cds(1)
  p = (gamma-1)*(cds(4) - 0.5*cds(2)*u - 0.5*cds(3)*v)
  ComputeSoundSpeed=sqrt(gamma*p/cds(1))

End function ComputeSoundSpeed


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  SUBROUTINE FLUEVAL(CS, FLUX)
    !    Purpose: to compute flux vector components FLUX(K) given the
    !             components U(K) of the vector of conserved variables

    Real   CS(4), FLUX(4), D, U, V, P, E
    !
    !     Compute physical variables
    D = CS(1)
    U = CS(2)/D
    V = CS(3)/D
    P = (gamma-1)*(CS(4) - 0.5*D*U*U- 0.5*D*V*V)
    E = CS(4)

    !     Compute fluxes
    FLUX(1) = D*U
    FLUX(2) = D*U*U + P
    FLUX(3) = D*U*V
    FLUX(4) = U*(E + P)
  END subroutine FLUEVAL


  ! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  Subroutine U0(x,y,Q)
    ! Purpose: to compute the solution at t=0

    Real x,y,Q(1:4),x0,y0,r0,r

    x0 = (Rx+LX)/2
    y0 = (Ry+LY)/2
    R0 = 0.4

    r = (x-x0)**2 + (y-y0)**2
    r = sqrt(r)
```

```
    ! Set similar density but much smaller temperature
    If ( r .le. r0) then
       Q(1)  = 1.
       Q(2)  = 1d-10
       Q(3)  = 1d-10
       Q(4)  = Q(1)/(gamma-1)
    Else
       Q(1)  = RhoInf
       Q(2)  = 1d-10
       Q(3)  = 1d-10
       Q(4)  = Pinf/(gamma-1)
    Endif

  End subroutine U0


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SUBROUTINE Rusanov(CDL,CDR,InFlux)

  real CDL(4),CDR(4),InFlux(4)
  real FDL(4), FDR(4)
  real SL, SR,Smax
  real al,ar,ul,ur
  Integer  K

  CALL FluEval(CDL,FDL)
  CALL FluEval(CDR,FDR)

  !   Calculate estimates for wave speed
  al = ComputeSoundSpeed(CDL)
  ar = ComputeSoundSpeed(CDR)
  ul = CDL(2)/CDL(1)
  ur = CDR(2)/CDR(1)
  SL = min(ul -al,ur-ar)
  SR = max(ul +al,ur+ar)
  SMAX = MAX(ABS(SL),ABS(SR))

  ! finally the flux
  Do k=1,4
     Influx(K) = 0.5*(FDL(K)+FDR(K)) - 0.5*SMAX*(CDR(K) - CDL(K))
  Enddo

END SUBROUTINE Rusanov


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Subroutine HLL(CDL,CDR,Flux)
```

```
  !Purpose: to compute the HLL Flux
  real CDL(4),CDR(4),Flux(4)
  real FDL(4), FDR(4)
  real SL, SR
  real al,ar,ul,ur
  Integer k

  CALL FluEval(CDL,FDL)
  CALL FluEval(CDR,FDR)

  !   Calculate estimates for wave speed
  al = ComputeSoundSpeed(CDL)
  ar = ComputeSoundSpeed(CDR)
  ul = CDL(2)/CDL(1)
  ur = CDR(2)/CDR(1)
  SL = min(ul -al,ur-ar)
  SR = max(ul +al,ur+ar)

  ! finally the flux
  If (SL .ge. 0.d0) Then
     Flux = FDL
  Else if (SR .le. 0.d0) Then
     Flux = FDR
  Else
     Do k=1,4
        Flux(K) = (SR*FDL(k)-SL*FDR(k)+SL*SR*(CDR(k)-CDL(k)))/(SR-SL)
     Enddo
  Endif

End subroutine HLL


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Subroutine HLLC(CDL,CDR,Flux)
  !Purpose: to compute the HLLC Flux

  real CDL(4),CDR(4),Flux(4), CDLS(4), CDRS(4)
  real FDL(4), FDR(4)
  real SL, SR, SS
  real al,ar,ul,ur
  real rhol, rhor, pl,pr,il,ir
  Integer k

  CALL FluEval(CDL,FDL)
  CALL FluEval(CDR,FDR)

  !   Calculate estimates for wave speed
  al = ComputeSoundSpeed(CDL)
```

```
  ar = ComputeSoundSpeed(CDR)
  ul = CDL(2)/CDL(1)
  ur = CDR(2)/CDR(1)
  SL =min(ul -al,ur-ar)
  SR =max(ul +al,ur+ar)

  ! Left and right density and presssure
  rhol = CDL(1)
  rhor = CDR(1)
  pl   = FDL(2)-rhol*ul**2
  pr   = FDR(2)-rhor*ur**2
  !gammam1 = gamma-1.0d0
  !pl = gammam1*(CDL(4)-0.5*(CDL(2)*ul+CDL(3)**2/rhol))
  !pr = gammam1*(CDR(4)-0.5*(CDR(2)*ul+CDR(3)**2/rhor))

  ! Speed of the middle wave
  il = rhol*(SL-ul) !Intermediate calculations
  ir = rhor*(SR-ur)
  SS = (pr-pl+ul*il-ur*ir)/(il-ir)

  ! Start states
  CDLS = il/(SL-SS)*(/ 1.0d0,SS,CDL(3)/rhol,CDL(4)/rhol+(SS-ul)*
(SS+pl/il) /)
  CDRS = ir/(SR-SS)*(/ 1.0d0,SS,CDR(3)/rhor,CDR(4)/rhor+(SS-ur)*
(SS+pr/ir) /)

  ! finally the flux
  If (SL .ge. 0.d0) Then
     Flux = FDL
  Else if (SS .ge.0.d0) Then
     Flux = FDL + SL*(CDLS-CDL)
  Else if (SR .le. 0.d0) Then
     Flux = FDR
  Else
     Flux = FDR + SR*(CDRS-CDR)
  Endif

End subroutine HLLC


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Subroutine Reconstruction(U1D,CDL,CDR)

  ! Purpose: reconstruction procedure
  Integer F
  Real U1d(4,-1:2),CDL(4),CDR(4)

  Select Case(SpatialOrder)
```

```
  Case(1)   ! First order
     Do f=1,4
        CDL(f) = U1D(f,0)
        CDR(f) = U1D(f,1)
     Enddo

  Case(2)   ! minbee limiter
       Do f=1,4
          CDL(f) = U1D(f,0) + 0.5* minmod(U1D(f,0)-U1D(f,-1),
U1D(f,1)-U1D(f,0))
          CDR(f) = U1D(f,1) - 0.5* minmod(U1D(f,1)-
U1D(f,0),  U1D(f,2)-U1D(f,1))
       Enddo

    Case default
       print*,' Wrong Limiter Type. Stop the code'
       stop

    end select

end subroutine Reconstruction


! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Real function minmod(x,y)
    ! Purpose: compute slope limiter function minmod

    Real x,y

    minmod = (max(0.0d0,
sign(0.5d0,x))+max(0.0d0,sign(0.5d0,y)))*min(abs(x),abs(y))

  End function minmod

END program
```

6.2 MATLAB code (Data Analysis)

```
%-------------------------------------------------------------------------
% Final_Project_Espinosa.m  Data Post-processing
%
%
%                      ___________________________
%                     |                           |
%                     |   Silvia ESPINOSA-GUTIEZ  |
%                     |    MIT ID: 919 277 866    |
%                     |___________________________|
%
%
% 18.086 Computational Science and Engineering II
% Spring 2013
%-------------------------------------------------------------------------

close all;
clear all;

%------------------
% Data selection
%------------------

% Filenames

data(1).filename  = 'ref13.dat';
data(2).filename  = '1DRusanov201a.dat';
data(3).filename  = '1DRusanov401a.dat';
data(4).filename  = '1DRusanov801a.dat';
data(5).filename  = '1DHLL201a.dat';
data(6).filename  = '1DHLL401a.dat';
data(7).filename  = '1DHLL801a.dat';
data(8).filename  = '1DHLLC201a.dat';
data(9).filename  = '1DHLLC401a.dat';
data(10).filename = '1DHLLC801a.dat';
data(11).filename = '1DHLLC201b.dat';
data(12).filename = '1DHLLC401b.dat';
data(13).filename = '1DHLLC801b.dat';
data(14).filename = '1DHLL4001a.dat';
data(15).filename = '1DHLLC4001a.dat';
data(16).filename = '1DHLLC4001b.dat';
data(17).filename = '1DHLLC4001bana.dat';


data(1).title = 'Central differencing scheme:  40 x 40 cells (Pe = 1.3750)';
data(2).title = 'Central differencing scheme:  35 x 35 cells (Pe = 1.5714)';
data(3).title = 'Central differencing scheme:  30 x 30 cells (Pe = 1.8333)';
data(4).title = 'Central differencing scheme:  28 x 28 cells (Pe = 1.9643)';
data(5).title = 'Central differencing scheme:  27 x 27 cells (Pe = 2.0370)';
data(6).title = 'Central differencing scheme:  25 x 25 cells (Pe = 2.2000)';
data(7).title = 'Central differencing scheme:  24 x 24 cells (Pe = 2.2917)';
data(8).title = 'Central differencing scheme:  22 x 22 cells (Pe = 2.5000)';
data(9).title = 'Central differencing scheme:  20 x 20 cells (Pe = 2.7500)';
data(10).title = 'Upwind differencing scheme:  40 x 40 cells (Pe = 1.3750)';
data(11).title = 'Upwind differencing scheme:  40 x 40 cells (Pe = 1.3750)';
data(12).title = 'Upwind differencing scheme:  40 x 40 cells (Pe = 1.3750)';
data(13).title = 'Upwind differencing scheme:  40 x 40 cells (Pe = 1.3750)';
```

```
%------------------
%  Data analysis
%------------------

% Determine the shape of the matrices (so as to make the program faster)

nfiles = size(data,2);

data(1).ncells = 8000;
data(1).matrix = zeros(data(1).ncells,4);

for i=2:nfiles,
    if i < nfiles-3
        data(i).ncells        = str2num(data(i).filename(end-7:end-5));
    elseif i < nfiles
        data(i).ncells        = str2num(data(i).filename(end-8:end-5));
    else
        data(i).ncells        = str2num(data(i).filename(end-11:end-8));
    end
    data(i).matrix        = zeros(data(i).ncells,4);
    data(i).abserror      = zeros(data(i).ncells,4);
    data(i).abserrorm     = zeros(data(i).ncells,4);
    data(i).relerror      = zeros(data(i).ncells,4);
    data(i).maxabserrorm  = zeros(2,4); % 1st row(max),  2nd row(x)
    data(i).maxrelerror   = zeros(2,4);
    data(i).meanabserrorm = zeros(1,4);
    data(i).meanrelerror  = zeros(1,4);
end

% Importation of the data (x,rho,u,p) and computation of the error

a         = importdata(data(1).filename,' ',1);
data(1).matrix = a.data;
rhospline = spline(data(1).matrix(:,1),data(1).matrix(:,2)); %Cubic spline↵
interpolation
uspline   = spline(data(1).matrix(:,1),data(1).matrix(:,3));
pspline   = spline(data(1).matrix(:,1),data(1).matrix(:,4));

for i = 2:nfiles,

    %Data importation
    a = importdata(data(i).filename,' ',1);
    data(i).matrix = a.data;

    %Absolute error
    x      = data(i).matrix(:,1);
    rhoexp = ppval(rhospline, x);
    uexp   = ppval(  uspline, x);
    pexp   = ppval(  pspline, x);
    data(i).abserror = [x,...
        data(i).matrix(:,2) - rhoexp,...
        data(i).matrix(:,3) - uexp,...
        data(i).matrix(:,4) - pexp];
```

```
    %Absolute error module
    data(i).abserrorm(:,1)   = data(i).abserror(:,1);
    data(i).abserrorm(:,2:4) = abs(data(i).abserror(:,2:4));

    %Maximum absolute error module & location
    [data(i).maxabserrorm(1,2:4), data(i).maxabserrorm(2,2:4)] = ...
        max(data(i).abserrorm(:,2:4));
    for j = 2:4,
        data(i).maxabserrorm(2,j) = data(i).matrix(data(i).maxabserrorm(2,j),↵
1);
    end

    %Absolute error module mean
    data(i).meanabserrorm(1,2:4) = mean(data(i).abserrorm(:,2:4));

    %Relative error
    data(i).relerror(:,1)   = data(i).matrix(1);
    data(i).relerror(:,2:4) = 100.*data(i).abserrorm(:,2:4)./...
        [abs(rhoexp), abs(uexp), abs(pexp)];
    for j = 1: data(i).ncells,
        for k = 2:4,
            if isinf(data(i).relerror(j,k)),
                data(i).relerror(j,k) = 0;
            end
        end
    end

    %Maximum relative error & location
    [data(i).maxrelerror(1,2:4), data(i).maxrelerror(2,2:4)] = ...
        max(data(i).relerror(:,2:4));
    for j = 2:2:4,
        data(i).maxrelerror(2,j) = data(i).matrix(data(i).maxrelerror(2,j),1);
    end

    %Relative error mean
    data(i).meanrelerror(1,2:4) = mean(data(i).relerror(:,2:4));

end

%------------------
% File generation
%------------------

% Open files
fhll4001aabs     = fopen( 'fhll4001aabserrorm.dat','w');
fhllc4001aabs    = fopen('fhllc4001aabserrorm.dat','w');
fhllc4001babs    = fopen('fhllc4001babserrorm.dat','w');
fhllc4001banaabs = fopen('fhllc4001banaabserrorm.dat','w');
fhll4001arel     = fopen(  'fhll4001arelerror.dat','w');
fhllc4001arel    = fopen( 'fhllc4001arelerror.dat','w');
fhllc4001brel    = fopen( 'fhllc4001brelerror.dat','w');
fhllc4001banarel = fopen('fhllc4001banarelerror.dat','w');

% Write absolute and relative errors
```

```
for i= 1:4001,
    fprintf( fhll4001aabs,   '%e %e %e %e \n',data(14).abserrorm(i,:));
    fprintf(fhllc4001aabs,   '%e %e %e %e \n',data(15).abserrorm(i,:));
    fprintf(fhllc4001babs,   '%e %e %e %e \n',data(16).abserrorm(i,:));
    fprintf(fhllc4001banaabs,'%e %e %e %e \n',data(17).abserrorm(i,:));
    fprintf( fhll4001arel,   '%e %e %e %e \n',data(14).relerror(i,:));
    fprintf(fhllc4001arel,   '%e %e %e %e \n',data(15).relerror(i,:));
    fprintf(fhllc4001brel,   '%e %e %e %e \n',data(16).relerror(i,:));
    fprintf(fhllc4001banarel,'%e %e %e %e \n',data(17).relerror(i,:));
end

% Close files
fclose(fhll4001aabs);
fclose(fhllc4001aabs);
fclose(fhllc4001babs);
fclose(fhllc4001banaabs);
fclose(fhll4001arel);
fclose(fhllc4001arel);
fclose(fhllc4001brel);
fclose(fhllc4001banarel);

%------------------
%       Plots
%------------------

i = 1; % Figure number

%------
% HLL
%------

%%%%%%%%%%%%%%%%%%%
j = 2; % Density
%%%%%%%%%%%%%%%%%%%

%% HLL rho (together): convergence with the mesh refinement (Experimental,↙
201,401,801)
hllrho = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(5).matrix(:,1),data(5).matrix(:,j),'k--',...
    data(6).matrix(:,1),data(6).matrix(:,j),'k-.',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k:');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (201 cells, first-order accurate)','HLL (401 cells,↙
first-order accurate)','HLL (801 cells, first-order accurate)')
set(hllrho,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;
```

```
%% HLL rho (alone): convergence with the mesh refinement (Experimental,201 &↙
Experimental,401 & Experimental,801)
hllrhos = figure(i);

hllrhos1 = subplot(2,2,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(5).matrix(:,1),data(5).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (201 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');

hllrhos2 = subplot(2,2,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(6).matrix(:,1),data(6).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (401 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');


hllrhos3 = subplot(2,2,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (801 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');

hllrhos4 = subplot(2,2,4);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(14).matrix(:,1),data(14).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (4001 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
```

```
set(hllrhos,'Color',[1 1 1])
i = i+1;

%% HLL rho convergence
hllrhoc = figure(i);

% HLL rho convergence abserrorm
hllrhoca = subplot(2,2,1);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).maxabserrorm(1,j),data(6).maxabserrorm(1,j),...
        data(7).maxabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (first-order accurate)')
set(hllrhoca,'Color',[1 1 1])
%saveas(hllrhoc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL rho convergence meanabserrorm
hllrhocam = subplot(2,2,2);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanabserrorm(1,j),data(6).meanabserrorm(1,j),...
        data(7).meanabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↵
accurate)','HLL (801 cells, first-order accurate)')
set(hllrhocam,'Color',[1 1 1])
%saveas(hllrhoc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL rho convergence relerror
hllrhocr = subplot(2,2,3);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).maxrelerror(1,j),data(6).maxrelerror(1,j),...
        data(7).maxrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↵
accurate)','HLL (801 cells, first-order accurate)')
set(hllrhocr,'Color',[1 1 1])
%saveas(hllrhoc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL rho convergence meanabserrorm
```

```
hllrhocrm = subplot(2,2,4);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanrelerror(1,j),data(6).meanrelerror(1,j),...
        data(7).meanrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
accurate)','HLL (801 cells, first-order accurate)')
set(hllrhocrm,'Color',[1 1 1])
%saveas(hllrhoc,['spy',data(i).filename(4:end-4)],'pdf');

set(hllrhoc,'Color',[1 1 1])

i = i+1;


%%%%%%%%%%%%%%%%%%%%
j = 3; % Velocity
%%%%%%%%%%%%%%%%%%%%

%% HLL u (together): convergence with the mesh refinement (Experimental,↙
201,401,801)
hllu = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(5).matrix(:,1),data(5).matrix(:,j),'k--',...
    data(6).matrix(:,1),data(6).matrix(:,j),'k-.',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k:');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (201 cells, first-order accurate)','HLL (401 cells,↙
first-order accurate)','HLL (801 cells, first-order accurate)')
set(hllu,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%% HLL u (alone): convergence with the mesh refinement (Experimental,201 &↙
Experimental,401 & Experimental,801)
hllus = figure(i);

hllrhos1 = subplot(2,2,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(5).matrix(:,1),data(5).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
```

```
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (201 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');


hllus2 = subplot(2,2,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(6).matrix(:,1),data(6).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (401 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');


hllus3 = subplot(2,2,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (801 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');


hllus4 = subplot(2,2,4);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(14).matrix(:,1),data(14).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (4001 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllus,'Color',[1 1 1])
i = i+1;

%% HLL u convergence
hlluc = figure(i);

% HLL u convergence abserrorm
hlluca = subplot(2,2,1);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
```

```
        [data(5).maxabserrorm(1,j),data(6).maxabserrorm(1,j),...
        data(7).maxabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
accurate)','HLL (801 cells, first-order accurate)')
set(hlluca,'Color',[1 1 1])
%saveas(hlluc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL u convergence meanabserrorm
hllucam = subplot(2,2,2);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanabserrorm(1,j),data(6).meanabserrorm(1,j),...
        data(7).meanabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
accurate)','HLL (801 cells, first-order accurate)')
set(hllucam,'Color',[1 1 1])
%saveas(hlluc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL u convergence relerror
hllucr = subplot(2,2,3);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).maxrelerror(1,j),data(6).maxrelerror(1,j),...
        data(7).maxrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
accurate)','HLL (801 cells, first-order accurate)')
set(hllucr,'Color',[1 1 1])
%saveas(hlluc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL u convergence meanabserrorm
hllucrm = subplot(2,2,4);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanrelerror(1,j),data(6).meanrelerror(1,j),...
        data(7).meanrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
```

```
        [data(5).maxabserrorm(1,j),data(6).maxabserrorm(1,j),...
        data(7).maxabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
accurate)','HLL (801 cells, first-order accurate)')
set(hlluca,'Color',[1 1 1])
%saveas(hlluc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL u convergence meanabserrorm
hllucam = subplot(2,2,2);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanabserrorm(1,j),data(6).meanabserrorm(1,j),...
        data(7).meanabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
accurate)','HLL (801 cells, first-order accurate)')
set(hllucam,'Color',[1 1 1])
%saveas(hlluc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL u convergence relerror
hllucr = subplot(2,2,3);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).maxrelerror(1,j),data(6).maxrelerror(1,j),...
        data(7).maxrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
accurate)','HLL (801 cells, first-order accurate)')
set(hllucr,'Color',[1 1 1])
%saveas(hlluc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL u convergence meanabserrorm
hllucrm = subplot(2,2,4);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanrelerror(1,j),data(6).meanrelerror(1,j),...
        data(7).meanrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
```

```
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (401 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');


hllps3 = subplot(2,2,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (801 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');


hllps4 = subplot(2,2,4);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(14).matrix(:,1),data(14).matrix(:,j),'k.');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLL (4001 cells, first-order accurate)')
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllps,'Color',[1 1 1])
i = i+1;


%% HLL p convergence abserrorm
hllpc = figure(i);

% HLL p convergence abserrorm
hllpca = subplot(2,2,1);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).maxabserrorm(1,j),data(6).maxabserrorm(1,j),...
        data(7).maxabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↙
```

```
accurate)','HLL (801 cells, first-order accurate)')
set(hllpca,'Color',[1 1 1])
%saveas(hllpc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL p convergence meanabserrorm
hllpcam = subplot(2,2,2);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanabserrorm(1,j),data(6).meanabserrorm(1,j),...
        data(7).meanabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↵
accurate)','HLL (801 cells, first-order accurate)')
set(hllpcam,'Color',[1 1 1])
%saveas(hllpc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL p convergence relerror
hllpcr = subplot(2,2,3);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).maxrelerror(1,j),data(6).maxrelerror(1,j),...
        data(7).maxrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↵
accurate)','HLL (801 cells, first-order accurate)')
set(hllpcr,'Color',[1 1 1])
%saveas(hllpc,['spy',data(i).filename(4:end-4)],'pdf');

% HLL p convergence meanabserrorm
hllpcrm = subplot(2,2,4);
    plot([data(5).ncells,data(6).ncells,data(7).ncells],...
        [data(5).meanrelerror(1,j),data(6).meanrelerror(1,j),...
        data(7).meanrelerror(1,j)],'kx-');
%axis([0.03 0.03001 7*10^4 7.4*10^4])
%title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLL (201 cells, first-order accurate)','HLL (401 cells, first-order↵
accurate)','HLL (801 cells, first-order accurate)')
set(hllpcrm,'Color',[1 1 1])
%saveas(hllpc,['spy',data(i).filename(4:end-4)],'pdf');

set(hllpc,'Color',[1 1 1])
i = i+1;
```

```
%------
% HLLC
%------

%%%%%%%%%%%%%%%%%%%
j = 2; % Density
%%%%%%%%%%%%%%%%%%%

%% HLLC rho (together): convergence with the mesh refinement (Experimental,↙
201,401,801)
hllcrho = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k--',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k-.',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k:');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(hllcrho,'Color',[1 1 1])
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%% HLLC rho (alone): convergence with the mesh refinement (Experimental,201 &↙
Experimental,401 & Experimental,801)
hllcrhos = figure(i);

hllcrhos1 = subplot(2,2,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');

hllcrhos2 = subplot(2,2,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
```

```
grid
legend('Experimental','HLLC (401 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcrhos3 = subplot(2,2,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (801 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcrhos4 = subplot(2,2,4);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(15).matrix(:,1),data(15).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (4001 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcrhos,'Color',[1 1 1])
i = i+1;

%% HLLC rho convergence
hllcrhoc = figure(i);

% HLLC rho convergence abserrorm
hllcrhoca = subplot(2,2,1);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).maxabserrorm(1,j),data(9).maxabserrorm(1,j),...
        data(10).maxabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcrhoca,'Color',[1 1 1])
%saveas(hllcrhoc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC rho convergence meanabserrorm
hllcrhocam = subplot(2,2,2);
```

```
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).meanabserrorm(1,j),data(9).meanabserrorm(1,j),...
        data(10).meanabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↵
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcrhocam,'Color',[1 1 1])
%saveas(hllcrhoc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC rho convergence relerror
hllcrhocr = subplot(2,2,3);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).maxrelerror(1,j),data(9).maxrelerror(1,j),...
        data(10).maxrelerror(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↵
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcrhocr,'Color',[1 1 1])
%saveas(hllcrhoc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC rho convergence meanabserrorm
hllcrhocrm = subplot(2,2,4);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).meanrelerror(1,j),data(9).meanrelerror(1,j),...
        data(10).meanrelerror(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↵
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↵
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcrhocrm,'Color',[1 1 1])
%saveas(hllcrhoc,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcrhoc,'Color',[1 1 1])
i = i+1;


%%%%%%%%%%%%%%%%%%%%
j = 3; % Velocity
%%%%%%%%%%%%%%%%%%%%
```

```
%% HLLC u (together): convergence with the mesh refinement (Experimental,↙
201,401,801)
hllcu = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k--',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k-.',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k:');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(hllcu,'Color',[1 1 1])
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%% HLLC u (alone): convergence with the mesh refinement (Experimental,201 &↙
Experimental,401 & Experimental,801)
hllcus = figure(i);

hllcrhos1 = subplot(2,2,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcus2 = subplot(2,2,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (401 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcus3 = subplot(2,2,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
```

```
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (801 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcus4 = subplot(2,2,4);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(15).matrix(:,1),data(15).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (801 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcus,'Color',[1 1 1])
i = i+1;

%% HLLC u convergence
hllcuc = figure(i);

% HLLC u convergence abserrorm
hllcuca = subplot(2,2,1);
plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).maxabserrorm(1,j),data(9).maxabserrorm(1,j),...
        data(10).maxabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↙
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcuca,'Color',[1 1 1])
%saveas(hllcuc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC u convergence meanabserrorm
hllcucam = subplot(2,2,2);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).meanabserrorm(1,j),data(9).meanabserrorm(1,j),...
        data(10).meanabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
```

```
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(hllcucam,'Color',[1 1 1])
%saveas(hllcuc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC u convergence relerror
hllcucr = subplot(2,2,3);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).maxrelerror(1,j),data(9).maxrelerror(1,j),...
        data(10).maxrelerror(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(hllcucr,'Color',[1 1 1])
%saveas(hllcuc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC u convergence meanabserrorm
hllcucrm = subplot(2,2,4);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).meanrelerror(1,j),data(9).meanrelerror(1,j),...
        data(10).meanrelerror(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(hllcucrm,'Color',[1 1 1])
%saveas(hllcuc,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcuc,'Color',[1 1 1])
i = i+1;

%%%%%%%%%%%%%%%%%%%%
j = 4; % Pressure
%%%%%%%%%%%%%%%%%%%%

%% HLLC p (together): convergence with the mesh refinement (Experimental, 201,401,801)
hllcp = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k--',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k-.',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k:');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic spline','FontWeight','Bold','FontSize',11)
```

```
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(hllp,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%% HLLC p (alone): convergence with the mesh refinement (Experimental,201 &↙
Experimental,401 & Experimental,801)
hllps = figure(i);

hllps1 = subplot(2,2,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps2 = subplot(2,2,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (401 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps3 = subplot(2,2,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (801 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps4 = subplot(2,2,4);
```

```
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(15).matrix(:,1),data(15).matrix(:,j),'k.');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (4001 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcps3,'Color',[1 1 1])
i = i+1;

%% HLLC p convergence
hllcpc = figure(i);

% HLLC p convergence abserrorm
hllcpca = subplot(2,2,1);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).maxabserrorm(1,j),data(9).maxabserrorm(1,j),...
        data(10).maxabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↙
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcpca,'Color',[1 1 1])
%saveas(hllcpc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC p convergence meanabserrorm
hllcpcam = subplot(2,2,2);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).meanabserrorm(1,j),data(9).meanabserrorm(1,j),...
        data(10).meanabserrorm(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↙
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcpcam,'Color',[1 1 1])
%saveas(hllcpc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC p convergence relerror
hllcpcr = subplot(2,2,3);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).maxrelerror(1,j),data(9).maxrelerror(1,j),...
        data(10).maxrelerror(1,j)],'kx-');
```

```
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↙
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcpcr,'Color',[1 1 1])
%saveas(hllcpc,['spy',data(i).filename(4:end-4)],'pdf');

% HLLC p convergence meanabserrorm
hllcpcrm = subplot(2,2,4);
    plot([data(8).ncells,data(9).ncells,data(10).ncells],...
        [data(8).meanrelerror(1,j),data(9).meanrelerror(1,j),...
        data(10).meanrelerror(1,j)],'kx-');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('It does not work, because the maximum are produced next to the first↙
edge','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('HLLC (201 cells, first-order accurate)','HLLC (401 cells, first-order↙
accurate)','HLLC (801 cells, first-order accurate)')
set(hllcpcrm,'Color',[1 1 1])
%saveas(hllcpc,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcpc,'Color',[1 1 1])
i = i+1;

----------------------------
Riemann solvers comparison
----------------------------

%%%%%%% Data %%%%%%

%%%%%%%%%%%%%%%%%%%%
j = 2; % Density
%%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(4).matrix(:,1),data(4).matrix(:,j),'k--',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k-.',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k:');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
```

```
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%%%%%%%%%%%%%%%%%%%%
j = 3; % Velocity
%%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(4).matrix(:,1),data(4).matrix(:,j),'k--',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k-.',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k:');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%%%%%%%%%%%%%%%%%%%%
j = 4; % Pressure
%%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(4).matrix(:,1),data(4).matrix(:,j),'k--',...
    data(7).matrix(:,1),data(7).matrix(:,j),'k-.',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k:');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%%%%%%% Absolute errors %%%%%%

%%%%%%%%%%%%%%%%%%%%
```

```
j = 2; % Density
%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(14).abserrorm(:,1),data(14).abserrorm(:,j),...
    data(15).abserrorm(:,1),data(15).abserrorm(:,j),...
    data(16).abserrorm(:,1),data(16).abserrorm(:,j));
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%%%%%%%%%%%%%%%%%%%
j = 3; % Velocity
%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(14).abserrorm(:,1),data(14).abserrorm(:,j),...
    data(15).abserrorm(:,1),data(15).abserrorm(:,j),...
    data(16).abserrorm(:,1),data(16).abserrorm(:,j));
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%%%%%%%%%%%%%%%%%%%
j = 4; % Pressure
%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(14).abserrorm(:,1),data(14).abserrorm(:,j),...
    data(15).abserrorm(:,1),data(15).abserrorm(:,j),...
    data(16).abserrorm(:,1),data(16).abserrorm(:,j));
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
```

```
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;

%%%%%%% Relative errors %%%%%%

%%%%%%%%%%%%%%%%%%%%
j = 2; % Density
%%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(14).relerror(:,1),data(14).relerror(:,j),...
    data(15).relerror(:,1),data(15).relerror(:,j),...
    data(16).abserrorm(:,1),data(16).abserrorm(:,j));
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


%%%%%%%%%%%%%%%%%%%%
j = 3; % Velocity
%%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(14).relerror(:,1),data(14).relerror(:,j),...
    data(15).relerror(:,1),data(15).relerror(:,j),...
    data(16).relerror(:,1),data(16).relerror(:,j));
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;
```

```
%%%%%%%%%%%%%%%%%%%%
j = 4; % Pressure
%%%%%%%%%%%%%%%%%%%%

%% Riemann solvers comparison
rsc = figure(i);
plot(data(14).relerror(:,1),data(14).relerror(:,j),...
    data(15).relerror(:,1),data(15).relerror(:,j),...
    data(16).relerror(:,1),data(16).relerror(:,j));
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)','HLLC (401↙
cells, first-order accurate)','HLLC (801 cells, first-order accurate)')
set(rsc,'Color',[1 1 1])
%saveas(hllrho,['spy',data(i).filename(4:end-4)],'pdf');
i = i+1;


------------------------------
First & Second Order accurate
------------------------------

%%%%%%%%%%%%%%%%%%%%
j = 2; % Density
%%%%%%%%%%%%%%%%%%%%

%% First and second order accurate
hllps = figure(i);

hllps1 = subplot(1,3,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k.', data(11).matrix(:,1),data↙
(11).matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps2 = subplot(1,3,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k.',data(12).matrix(:,1),data(12).
matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
```

```
title('CFD pressure data interpolation by using a cubic↵
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (401 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps3 = subplot(1,3,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k.',data(13).matrix(:,1),data↵
(13).matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↵
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (801 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcps3,'Color',[1 1 1])
i = i+1;


%%%%%%%%%%%%%%%%%%%%
j = 3; % Velocity
%%%%%%%%%%%%%%%%%%%%

%% First and second order accurate
hllps = figure(i);

hllps1 = subplot(1,3,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k.',data(11).matrix(:,1),data(11).
matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↵
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps2 = subplot(1,3,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k.',data(12).matrix(:,1),data(12).
matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↵
spline','FontWeight','Bold','FontSize',11)
```

```
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (401 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps3 = subplot(1,3,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k.',data(13).matrix(:,1),data↵
(13).matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↵
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (801 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcps3,'Color',[1 1 1])
i = i+1;

%%%%%%%%%%%%%%%%%%%
j = 4; % Presure
%%%%%%%%%%%%%%%%%%%

%% First and second order accurate
hllps = figure(i);

hllps1 = subplot(1,3,1);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(8).matrix(:,1),data(8).matrix(:,j),'k.',data(11).matrix(:,1),data(11).
matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↵
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (201 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps2 = subplot(1,3,2);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(9).matrix(:,1),data(9).matrix(:,j),'k.',data(12).matrix(:,1),data(12).
matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↵
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
```

```
legend('Experimental','HLLC (401 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');


hllcps3 = subplot(1,3,3);
plot(data(1).matrix(:,1),data(1).matrix(:,j),'k-',...
    data(10).matrix(:,1),data(10).matrix(:,j),'k.',data(13).matrix(:,1),data↙
(13).matrix(:,j),'kx');
%axis([0.03 0.03001 10*10^4 10.4*10^4])
title('CFD pressure data interpolation by using a cubic↙
spline','FontWeight','Bold','FontSize',11)
xlabel('Time    t (s)','FontSize',10)
ylabel('Pressure    p (Pa)','FontSize',10)
grid
legend('Experimental','HLLC (801 cells, first-order accurate)')
%saveas(hllcrho,['spy',data(i).filename(4:end-4)],'pdf');

set(hllcps3,'Color',[1 1 1])
i = i+1;

%----------
% 3D plots
%----------

% To fill in by the user
f3d = '2DHLLC801b.dat';

% Reading file: x, y, rho, |v|, p
n        = str2num(f3d(end-7:end-5));
data3d   = importdata(f3d,' ',1);
matrix3d = data3d.data;

size(matrix3d)

% Conversion to surf format
X = zeros(1,n);   % Initialization
Y = zeros(1,n);
D = zeros(n);
U = zeros(n);
P = zeros(n);

Y = matrix3d(1:n,2)'; % Y

D = reshape(matrix3d(:,1),n,n); % A temporary matrix is used, but instead of
X = D(1,:);                     % creating a new one, D is used.

D = reshape(matrix3d(:,3),n,n); % D
U = reshape(matrix3d(:,4),n,n); % U
P = reshape(matrix3d(:,5),n,n); % P


% Create figure
figure1 = figure('XVisual',...
    '0x24 (TrueColor, depth 24, RGB mask 0xff0000 0xff00 0x00ff)',...
```

```
    'Renderer','OpenGL');

% Create axes
axes1 = axes('Parent',figure1,'ZTick',[0 0.2 0.4 0.6 0.8 1 1.2],...
    'YTick',[-1 -0.8 -0.6 -0.4 -0.2 -5.55111512312578e-17 0.2 0.4 0.6 0.8↵
1],...
    'XTick',[-1 -0.8 -0.6 -0.4 -0.2 -5.55111512312578e-17 0.2 0.4 0.6 0.8↵
1],...
    'Position',[0.46171875 0.0638888888888889 0.44328125 0.922222222222222],...
    'FontSize',12);
view(axes1,[-49.5 32]);
grid(axes1,'on');
hold(axes1,'all');

% Create surf
surf(X,Y,D,'Parent',axes1,'FaceLighting','phong',...
    'FaceColor',[0.800000011920929 0.800000011920929 0.800000011920929],...
    'FaceAlpha',0.8,...  %Transparency
    'EdgeColor','none');

% Create light
light('Parent',axes1,'Style','local',...
    'Position',[-10.162701816704 -0.924193626363743 7.49759526419164]);

% Create xlabel
xlabel('x','FontSize',12);

% Create ylabel
ylabel('y','FontSize',12);

set(figure1,'Color',[1 1 1])

% Create figure
figure1 = figure('XVisual',...
    '0x24 (TrueColor, depth 24, RGB mask 0xff0000 0xff00 0x00ff)',...
    'Renderer','OpenGL');

% Create axes
axes1 = axes('Parent',figure1,'ZTick',[0 0.2 0.4 0.6 0.8 1],...
    'YTick',[-1 -0.8 -0.6 -0.4 -0.2 -5.55111512312578e-17 0.2 0.4 0.6 0.8↵
1],...
    'XTick',[-1 -0.8 -0.6 -0.4 -0.2 -5.55111512312578e-17 0.2 0.4 0.6 0.8↵
1],...
    'Position',[0.46171875 0.0638888888888889 0.44328125 0.922222222222222],...
    'FontSize',12);
view(axes1,[-49.5 32]);
grid(axes1,'on');
hold(axes1,'all');

% Create surf
surf(X,Y,U,'Parent',axes1,'FaceLighting','phong',...
    'FaceColor',[0.800000011920929 0.800000011920929 0.800000011920929],...
    'FaceAlpha',0.8,...  %Transparency
    'EdgeColor','none');
```

```
% Create light
light('Parent',axes1,'Style','local',...
    'Position',[-10.162701816704 -0.924193626363743 7.49759526419164]);

% Create xlabel
xlabel('x','FontSize',12);

% Create ylabel
ylabel('y','FontSize',12);

set(figure1,'Color',[1 1 1])

% Create figure
figure1 = figure('XVisual',...
    '0x24 (TrueColor, depth 24, RGB mask 0xff0000 0xff00 0x00ff)',...
    'Renderer','OpenGL');

% Create axes
axes1 = axes('Parent',figure1,'ZTick',[0 0.2 0.4 0.6 0.8 1],...
    'YTick',[-1 -0.8 -0.6 -0.4 -0.2 -5.55111512312578e-17 0.2 0.4 0.6 0.8
1],...
    'XTick',[-1 -0.8 -0.6 -0.4 -0.2 -5.55111512312578e-17 0.2 0.4 0.6 0.8
1],...
    'Position',[0.46171875 0.0638888888888889 0.44328125 0.922222222222222],...
    'FontSize',12);
view(axes1,[-49.5 32]);
grid(axes1,'on');
hold(axes1,'all');

% Create surf
surf(X,Y,P,'Parent',axes1,'FaceLighting','phong',...
    'FaceColor',[0.800000011920929 0.800000011920929 0.800000011920929],...
    'FaceAlpha',0.9,...  %Transparency
    'EdgeColor','none');

% Create light
light('Parent',axes1,'Style','local',...
    'Position',[-10.162701816704 -0.924193626363743 7.49759526419164]);

% Create xlabel
xlabel('x','FontSize',12);

% Create ylabel
ylabel('y','FontSize',12);

set(figure1, 'Color',[1 1 1])

surf(X,Y,P,'FaceColor','red','EdgeColor','none');
camlight left; lighting phong;

% surf(X,Y,P);  %It did not work, even with surf(X,Y,P,P)
% colormap hsv
% alpha(.8)
```

```
% surf(X,Y,P);  %It did not work, even with surf(X,Y,P,P)
% colormap hsv
% shading interp;

% shading interp;
% colormap(pink);
% legend('CFD pressure data','Cubic spline interpolation','Sample using the↙
minimum sampling frequency')
% axis([-1 1 -1 1 min(matrix3d(:,5)) max(matrix3d(:,5))])


i = i+1;


% figure(i)
% plot3(matrix3d(:,1),matrix3d(:,2),matrix3d(:,4));
% shading interp;
% colormap(pink);
% i = i+1;
%
% figure(i)
% plot3(matrix3d(:,1),matrix3d(:,2),matrix3d(:,4));
% shading interp;
% colormap(pink);
% i = i+1;
```