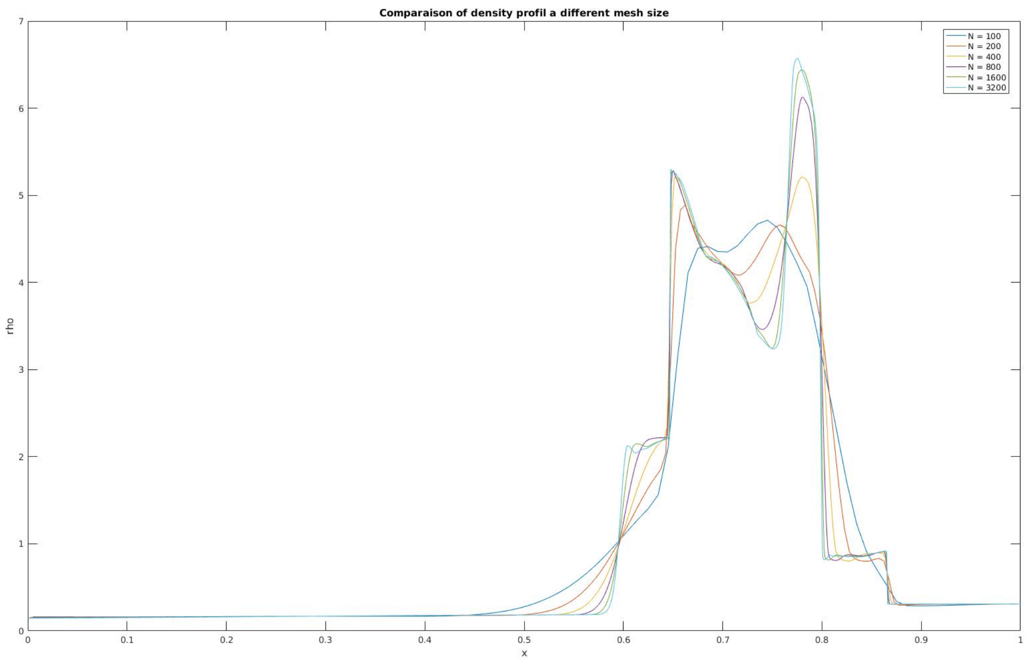
Internship Report

Second Order Method for the Euler's Gas Equation on Non Regular Grid

Universität zu Köln
Mathematishes Institut
INSA de Rouen Normandie - 2017



Author :
Timothée Schmoderer

Referent :
Gregor Gassner
Arnaud Knippel

Contents

1 Introduction

This report present my experience during my fourth year internship for the National Institute for Applied Science of Rouen (France). This three month internship was realised during the summer 2017 in the University of Cologne (Germany) under the supervision of Pr. Gregor Gassner [1].
The internship takes part in my study in applied mathematics, particularly the approximation of the solutions of partial differential equations. I chose to join this team because I wanted to get in touch with the university research world and I also find the topic studied there very interesting.

The project : My project is included in the numerical simulation area of applied mathematics. This research field deal with the numerical approximation of partial differential equations (PDE) which is a strong challenge in the XXIth century since this kind of equations occurs in many fields (economy, biology, astrophysics, chemistry, aeronautics ...). Precisely, I dealt with fluids dynamics equations and, for instance, my work could be applied to the motion of gas in pipe between a manufacture and your heater but also between a tank and the motor of a rocket.
The purpose of my project was to implement and test a numerical method of the computational fluid dynamic (CFD) and then to upgrade the method by extending it on non-uniform meshes. I had to call a lot of the skills I learned in my study such as implementation, concepts of the numerical simulation, numerical integration ... Thus this internship is totally part of my formation in numerical analysis and approximation of partial differential equations.
The main interest of this work was to discover the work in a university laboratory, which I consider to make my job later. This was also the root of the main difficulty I encounter during this summer experience, I had to learn alone about high skilled concept and try to find answer by myself (reading research paper is not that easy). However, I also enjoyed learning advanced numerical methods and techniques which are specific to the CFD.

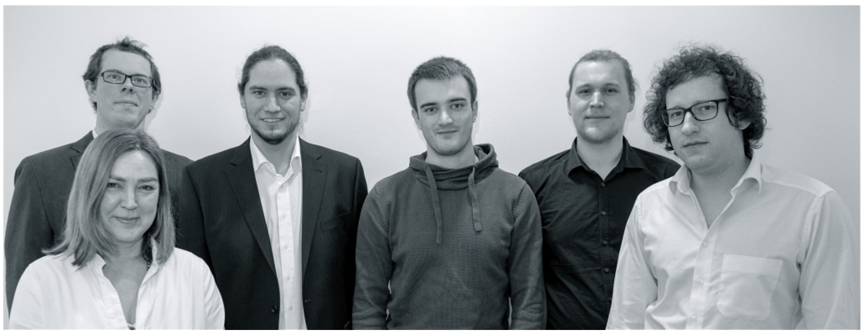


FIGURE 1 – The numerical simulation team in the mathematical institute of Cologne. (left to right : Dr. Winters, Pr. Gassner, M. Wintermeyer, M. Friedrich, M. Bohm, Ms. Musielack-Erle)

The team : The university of Cologne (Universität zu Köln) is the biggest university of Germany, with 7 research area and almost 50000 students. The university is in the top 200 university in the Shanghaï ranking of university. She has many international partner, for example the university of Clermont, Paris I (Sorbonne) and HEC for France. Another

1. Personal web page : http://www.mi.uni-koeln.de/NumSim/gregor-gassner/

example of the good international reputation of the university is that in my team there were one American and one Russian.

FIGURE 2 – Logo of the university

Speaking of the team, I was welcomed in a 8 member team (see figure 1). A striking thing is that the team is very young (average age is bellow 30). Four were doctor and three were in a post-doctoral position. All are working under the supervision of Pr. Gassner, giving lectures, making project with students .. To me, the team looks like very productive[2].

Before beginning, I would like to thank all of them for the reception I received and the help they provided me when I needed. And particularly Niklas Wintermeyer and Andrew Winters my two office neighbour for the litters of coffee we drank together.

How to read : In the next 40 pages we will go through the problem I tried to give a solution during my internship.
Here is a brief summary of this report, the section 2 gives a brief introduction to the Euler's gas equations, which will be solved in the next sections, the third section develop the method implemented. Then fourth section explain the implementation and show the codes (this section could be avoid if the reader is not interested in code questions). The fifth is the heart of this survey, it develop the convergence analysis of the method and the study of several example to illustrate the efficiency of the method.
Finally section 6 and 7 explain the adaptation of the method on the non-regular grid we wanted and redo all the convergence analysis and the example showcase.

2. See : http://www.mi.uni-koeln.de/NumSim/gregor-gassner/publications/for a list of publications.

2 Euler's Gas Equation

Let's begin with a brief introduction about the euler gas equations (see [1, 2] for more details). These equations were first described by Euler in 1757 to model fluid behaviour when the flow is adiabatic, without momentum exchange by viscosity or energy exchange by thermal conduction. That is to say, it especially works well with gas.
In this report we will be interested in one dimensional flow. This is the standart model for gas moving in a pipe. Thus,the governing equations are the following :

$$\boxed{\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho u\\ E\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ P+\rho u^2\\ u(E+P)\end{pmatrix}=0 \qquad \forall x\in\Omega,\ \forall t\geq 0} \tag{1}$$

Actually, these equations are just a formulation for the conservation of mass, momentum and energy across time and space.

We have :

1. ρ stands for the fluid density is strictly positive.
2. P stands for the fluid pressure and is also strictly positive.
3. E is the total energy of the system which by (2) is also strictly positive.
4. u is the fluid velocity which could be either positive (if gas is moving toward increasing value in space) or negative.

All this variable depends of time and space. However, we omit this dependence for lighter notation.

The system (1) is closed by a fourth equation, a thermodynamics equation of state which bound the four variable together :

$$\boxed{E=\frac{P}{\gamma-1}+\frac{\rho u^2}{2} \qquad \forall x\in\Omega,\ \forall t\geq 0} \tag{2}$$

Where γ is the heat capacity ratio (or adiabatic index) of the fluid. It will be taken equal to : 1.4 for dry air at 0°C.
We will also need to compute the speed of sound in the gas, which is given by :

$$c=\sqrt{\frac{\gamma P}{\rho}} \qquad \forall x\in\Omega,\ \forall t\geq 0 \tag{3}$$

In general there is no analytic solution of the system (1) - (2). That's why we need numerical scheme that can good approximate the solution. Moreover we sometimes can evaluate a numerical scheme by comparing the result of a computation to the analytic solution but as we don't know it, we will have to consider different kind of convergence and error analysis of our numerical scheme.

The system (1) describe non linear hyperbolic partial differential equations. A new difficulty is that, even if the initial conditions are smooth, the solution could present discontinuities, or in a more physical vocabulary, shock waves. This is a major problem to construct stable numerical method to capture them.

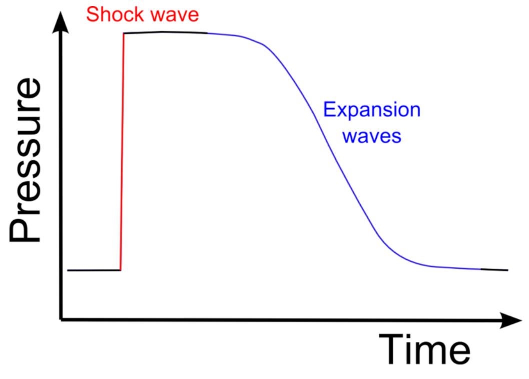


FIGURE 3 – Instance of shock waves from [3]

To put the system (1) is a more compact way, let $U = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}$ be the state vector of the system and $f(U) = \begin{pmatrix} \rho u \\ P + \rho u^2 \\ u(E+P) \end{pmatrix}$ be the flux vector such as (1) rewrite as :

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

This a general notation for hyperbolic systems.
For instance, in one dimension if we choose f to be the identity then we get the advection equation :

$$u_t + u_x = 0$$

The problem (1)-(2) is complete with appropriate initial conditions and boundary conditions. I will describe the different kind of boundary conditions we will meet in this work (assume that $\Omega = [a,b]$, and that the conditions hold during the whole simulation) :

1. **Periodic :** This is simply that what append at one side of the domain is reported at the other side :

$$\Phi(a,t) = \Phi(b,t) \qquad \Phi = u,\ \rho \text{ and } \ P \tag{4}$$

2. **Solid wall :** The fluid will behave like there is two solid impenetrable wall at the two sides of the tube. In term of equation, it is :

$$\begin{aligned} u(a,t) = u(b,t) = 0 \\ \frac{\partial \Phi}{\partial x}(a,t) = \frac{\partial \Phi}{\partial x}(b,t) = 0 \qquad \Phi = \rho \text{ and } P \end{aligned} \tag{5}$$

3. **Inlet :** This condition prescribe the value of the flow at the boundary x_B, like there is a source of fluid :

$$\Phi(x_B,t) = \Phi_{in}(t) \qquad \Phi = u,\ \rho \text{ and } \ P$$

4. **Outflow :** All the fluids that reaches the boundary x_B is evacuate outside of the domain. In terms of equations :

$$\frac{\partial \Phi}{\partial x}(x_B, t) = 0 \qquad \Phi = u,\ \rho \text{ and } P$$

The purpose is to implement the scheme describe in [4], check that the convergence rate is 2 and to test the scheme on different cases. The difficulty in solving this equations is that even when the initials conditions are smooth, shock (discontinuities) could occurs in the solution. Shocks are characterised numerically with an infinite gradient which computer doesn't like. All the problem is to find methods that capture shocks.

3 Numerical scheme

Notice that the method describe in [4] focus on moving boundary in two dimensions. But we forgot that and focus on the case the boundaries are fixed and the dimension is one.

3.1 General method for hyperbolic system

Many methods have been developed to solve hyberbolics equations. We can think about finite volume method which are very appropriate for these kind of problem, but also finite differences, finite element, discontinuous Galerkin method ..
The method implemented in this work use finite differences. This kind of method are in general easy to implement, very efficient on Cartesian grid and very easy to make high order. The general receipt for this kind of method is to get approximation (as precise as possible) of the solution of the PDE's system on a discretization of the domain Ω (the nodes). The approximation is computed by forming differences of the value at the nodes to compute the derivative.

3.1.1 Cells and Nodes

Choose a integer N (obviously not zero, we won't go far), and divide Ω into N equals grid "cells" (Figure 4), let x_i be the cell-centers and Δx be the cell's width.

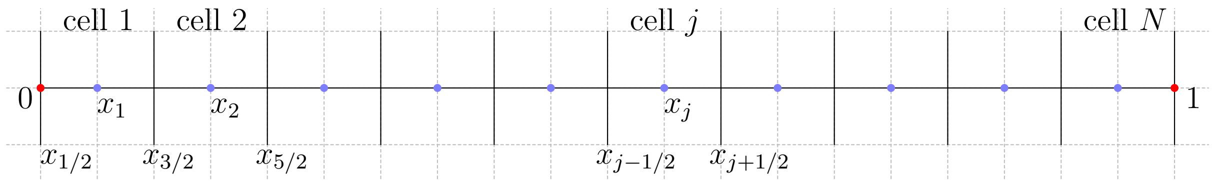


FIGURE 4 – Cells construction on Ω

We denote by $x_{j\pm1/2}$ the cells interfaces : $x_j \pm \dfrac{\Delta x}{2}$.

3.1.2 Algorithm

The purpose of the discretization is to get the following equations :

$$\boxed{\frac{dU_j}{dt} = -\frac{\hat{f}_{j+1/2} - \hat{f}_{j-1/2}}{\Delta x} \qquad j = 1, \ldots, N \qquad (6)}$$

This form introduce us to the main ingredient of the method, the computation of the fluxes at the cell's interfaces. Providing U_j for a certain time t we compute the fluxes at the interfaces and then the time-variation of the state is equal to the difference of fluxes (it seems very logical, isn't it ?). Because we know the sate only in the center of the cell we can't directly compute the flux at the interfaces, the main difficulty is to get a good approximation of the flux value those points.

As proposed in [4], we compute the $\{\hat{f}_{j\pm1/2}\}$ (which are called *numerical fluxes* by analogy with finite volume method) as follows :

1. Compute the spectral radius of the Jacobian a_j at $x = x_j$. Analytic computation (see [2] for details) let us know that for compressible Euler equation this is :

$$a_j = |u_j| + c_j$$

2. We split the flux function f as :

$$f(U_j) = f_j^+ + f_j^- \qquad f_j^\pm = \frac{1}{2}\left(f(U_j) \pm a_j U_j\right)$$

3. We compute the slopes in each cells using the minmod function :

$$(f_x)_j^\pm = minmod\left(\theta\frac{f_j^\pm - f_{j-1}^\pm}{\Delta x}, \frac{f_{j+1}^\pm - f_{j-1}^\pm}{2\Delta x}, \theta\frac{f_{j+1}^\pm - f_j^\pm}{\Delta x}\right)$$

Where $\theta \in [1,2]$ is a parameter that control the amount of numerical dissipation, it will be taken at 1.5.
The *minmod* function is defined by :

$$minmod(a,b,c) = \begin{cases} \min(a,b,c) & \text{if } a > 0,\ b > 0 \text{ and } c > 0 \\ \max(a,b,c) & \text{if } a < 0,\ b < 0 \text{ and } c < 0 \\ 0 & \text{else} \end{cases}$$

4. Construct f^E and f^W as :

$$f_j^E = f_j^+ + \frac{\Delta x}{2}(f_x)_j^+ \qquad f_j^W = f_j^- - \frac{\Delta x}{2}(f_x)_j^-$$

5. Finally :

$$\hat{f}_{j+1/2} = f_j^E + f_{j+1}^W \quad \hat{f}_{j-1/2} = f_{j-1}^E + f_j^W$$

The interesting part here is the *minmod* function which is the shock capturing procedure. At step *2*, the split fluxes have the following property :

$$\frac{\partial f^+}{\partial x} \geq 0 \qquad \frac{\partial f^-}{\partial x} \leq 0$$

Hence the *minmod* function is a switch on this property. If it doesn't hold, it means we are in a sensible region, better to set the slopes to 0 to keep a constant flux, else we take the closest value from zeros in order to avoid the infinite gradient.

3.2 Ghost Nodes

While presenting the scheme I haven't discuss on which nodes we have to apply the method. Obviously there is no problem when we are in the middle of the domain problems arises when we come close to the boundaries.
In order to achieve the method two ghosts point are needed on each side, with value dicted by the boundary conditions.
Why two ? Remember that we want values at each cell's interfaces (keep in mind figure 4). Then for instance to get $\hat{f}_{1-1/2}$ we will need f_0^E so at this point we need one ghost cell left. Moreover to get f_0^E we will need to get the slopes for $j = 0$ then we might need $f_{-1}^{\pm}$ in the *minmod* function. So one more ghost cell. (And reciprocally on the right side).

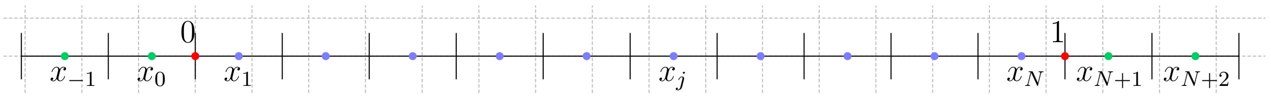


FIGURE 5 – Ω with ghost points

The whole method can be summarize in this array (red star are ghost values, and dot show for what j we can compute the value) :

cell	-1	0	1	2	$\dots$	j	$\dots$	$N-1$	N	$N+1$	$N+2$
a_j	★	★	·	·	·	·	·	·	·	★	★
$f^{\pm}$	★	★	·	·	·	·	·	·	·	★	★
$f_j^{\pm} - f_{j-1}^{\pm}$		★	·	·	·	·	·	·	·	★	
$f_{j+1}^{\pm} - f_{j-1}^{\pm}$		★	·	·	·	·	·	·	·	★	
$f_{j+1}^{\pm} - f_j^{\pm}$		★	·	·	·	·	·	·	·	★	
$(f_x)^{\pm}$		★	·	·	·	·	·	·	·	★	
f^E		★	·	·	·	·	·	·	·	★	
f^W		★	·	·	·	·	·	·	·	★	
$\hat{f}_{j+1/2}$			·	·	·	·	·	·	·		
$\hat{f}_{j-1/2}$			·	·	·	·	·	·	·		
$\dfrac{dU_j}{dt}$			·	·	·	·	·	·	·		

Finally we got the time variation of the state vector in each cell, that exactly what we wanted.

3.3 Boundary condition

The boundary conditions are reflected in the value set in the ghost cells. I will describe how *wall* and *periodic* conditions are set, because, *inlet* conditions cause no problem and *outflow* is easy deduce from the wall conditions.
I also only describe them for left boundary the right case is identical.

3.3.1 Periodic conditions

These conditions describe the gas as if it is moving in a circle. As we want the value at $x = 0$ and $x = 1$ to coincide, the following figure (6) allow us to understand what happen.

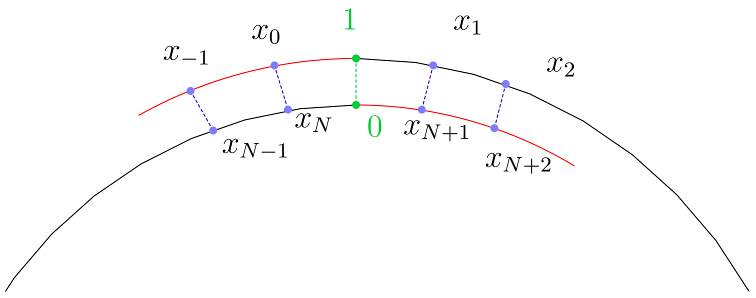


FIGURE 6 – Ω seen with periodic boundary conditions

Then it is easy to deduce the conditions :

$$\begin{aligned} \rho_0 &= \rho_N & u_0 &= u_N & P_0 &= P_N \\ \rho_{-1} &= \rho_{N-1} & u_{-1} &= u_{N-1} & P_{-1} &= P_{N-1} \end{aligned}$$

That is to say for the state vector :

$$U_0 = U_N \qquad U_{-1} = U_{N-1}$$

3.3.2 Wall conditions

From equations (5) we deduce the value in the value of ρ, u and P at the two ghost cells centers :

$$\begin{aligned} \rho_0 &= \rho_1 & u_0 &= -u_1 & P_0 &= P_1 \\ \rho_{-1} &= \rho_2 & u_{-1} &= -u_2 & P_{-1} &= P_2 \end{aligned}$$

Which in term of U is the following :

$$U_0 = \begin{pmatrix} U_1^1 \\ -U_1^2 \\ U_1^3 \end{pmatrix} \qquad U_{-1} = \begin{pmatrix} U_2^1 \\ -U_2^2 \\ U_2^3 \end{pmatrix}$$

where the upperscript is the denotes the i-th component.

3.4 Integration of the semi-discrete system

Once we get the scheme (6) we have to numerically integrate it. Here I quickly present two method to do it.
The easiest way to achieve the integration is with the well-known Euler forward scheme. We took a time discretization $\{t_n = n\Delta t\}$ (note that there is no need that the time discretization is regular as we will present it bellow) and we approximate the time derivative by :

$$\frac{dU_j}{dt} \approx \frac{U_j^{n+1} - U_j^n}{\Delta t}$$

Thus the final discrete equation we get is :

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta x}\left(\hat{f}_{j+1/2} - \hat{f}_{j-1/2}\right)$$

However this method present the inconvenient that depending on the choice of Δt the numerical method could be unstable (unbounded oscillations appears in the numerical solution). Thus we will have to determine the so called Courant's number which bound the time step to the space step.

Courant's number : This number is a constant C for which the method is stable and such as :

$$\frac{|\nu|\Delta t}{\Delta x} \leq C$$

Where ν is the velocity, in our case it is the sum of the gas velocity and the sound velocity, which is a. Moreover, here we know, that the best constant we can choose is 1.
The smaller is the space step, the smaller must be the time step. Then a first formula for the time step, that will avoid the method to be unstable is :

$$\Delta t = \frac{\Delta x}{\max_{t\geq 0} \max_{x\in\Omega} a}$$

One improvement of this result we can make is to dynamically change the time step. As the spectral radius of the Jacobian change in time, we can update the time step and get for each iteration the biggest step possible :

$$\Delta t^n = \frac{\Delta x}{\max_{x\in\Omega} a^n} \tag{7}$$

This result is optimal and will guaranty the smallest execution time of the method.

Moreover, even with a "good" choose of Δt some oscillation could occurs near the discontinuities which are only numerical. This oscillations are due to the choice of the integration scheme. To handle this problem I used a Strong Stability Preserving Runge-Kutta method of order 3 (SSP - RK). It presents two main advantages. First the courant number is the same as the one with the Euler scheme, and second, this method is TVD (total variation diminution) :

$$TV(U^{n+1}) \leq TV(U^n) \qquad TV(U^n) = \sum_j |U_{j+1}^n - U_j^n|$$

Based on [5, 6, 7] and as suggests by the author of the method I implemented the following 3rd order SSP Runge-Kutta method :

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n) \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}) \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}) \end{aligned} \tag{8}$$

Where the L operator is the one that give us the second member. This scheme is SSP with the same Courant's number as the Euler Forward scheme and third order in time.

4 Implementation

In this section, I will describe briefly the code implemented for this method. I chose the Matlab language because we do not searched for a speed code and as we will see below, the operation performed with this language are very natural and easy to implement.

4.1 Data Representation

In this part, I will abuse with the "dot" notation of Matlab meaning component by component operation. Choosing Matlab for programming language was a logical first choice to do, because many operation are transparent in Matlab. However we should keep in mind that greater efficiency could be achieve with more basic language.
With a very natural way, the partial differential equation is describe by a vector U whose size is : 3 row times (N+4) columns. Why N+4 ? Because there is N internal nodes and 2 ghost points on each sides.
Thus in only on structure I could represent all the system at time t. For instance for $N = 10$ cells and $t = 0$ for the blast wave problem (with solid wall boundary conditions, see below for details) I get the structure :

$$\begin{pmatrix}\rho\\ \rho u\\ E\end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1\\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0\\ 0.025 & 2500 & 2500 & 0.025 & 0.025 & \dots & 0.025 & 0.025 & 250 & 250 & 0.025 \end{pmatrix}$$

4.2 Code

I will describe the code implementing the method in Matlab language. I recall that all the operation should be thinking made in one cell. The choice of the Matlab language offers us "element by element" operations which allow us to compute all the cell in the same operation. Please note that I know that my functions are not optimal, but I preferred to develop a bit the computation for greater understanding (we are not up to an allocation and a multiplication).

First is the function implementing the flux : $\begin{pmatrix}\rho u\\ \rho u^2 + P\\ u(E+P)\end{pmatrix}$. Given the vector of state U we can get easily u, ρ and E but we will need the adiabatic constant γ to compute the pressure with the equation of state (2). Thus the flux is given by :

$$f(U) = U. * u + \begin{bmatrix}0_{\mathbb{R}^{N+4}}\\ P\\ P. * u\end{bmatrix}$$

```matlab
function y = f(U,gamma)
  % Density
  rho = U(1,:);
  % Velocity
  v = U(2,:)./rho;
  % Energy
  E = U(3,:);
  % Pressure
  P = (gamma-1)*(E-0.5*rho.*v.*v);
  % Flux
  y = U.*v+[zeros(size(P));P;P.*v];
end
```

Code 1 – Implementation of the flux function - f.m

Second easy step is to compute the speed of the sound in the domain. Same argumentation leads us to compute density and pressure from the vector of state and the adiabatic constant :

$$c = \sqrt{\gamma P./\rho}$$

```
function c = speedofsound(U,gamma)
  % Density
  rho = U(1,:);
  % Velocity
  v = U(2,:)./rho;
  % Energy
  E = U(3,:);
  % Pressure
  P = (gamma-1)*(E-0.5*rho.*v.*v);
  % Speed of sound
  c = sqrt(gamma*P./rho);
end
```

Code 2 – Implementation of the computation of the speed of sound - speedofsound.m

As we saw before, we will need different boundary conditions implemented. The following function show some of the possibility (please report to the section 5 to understand the conditions).

```
function U = boundary(U,bound)
  switch bound
    case 'periodic'
      U(:,[1 2 end-1 end]) = U(:,[end-3 end-2 3 4]);
    case 'in-out-shu'
      U(:,[1 2]) = [3.857143;10.142;39.167].*ones(3,2);
      U(:,[end-1 end]) = U(:,[end-2 end-3]);
    otherwise % Wall
      U(:,[1 2 end-1 end]) = [1;-1;1].*U(:,[4 3 end-2 end-3]);
  end
end
```

Code 3 – Extract of the boundary conditions function - boundary.m

One important function in the procedure is the *minmod* function. This is not the best way to implement this function[3] because by using twice the function *find* we loop twice over the data and we do a lot of memory call. However, the function is more concise this way and it doesn't affect significantly the execution time.

$$minmod(a,b,c) = \begin{cases} min(a,b,c) & \text{if } a > 0,\ b > 0 \text{ and } c > 0 \\ max(a,b,c) & \text{if } a < 0,\ b < 0 \text{ and } c < 0 \\ 0 & \text{else} \end{cases}$$

Keep in mind that in our case, a, b and c are $3 \times (N+4)$ matrix (refers to table at the end of section 3.2 for the size of each data).

3. For instance, one can implement this function with only one loop over the range of all data.

```
function y = minmod(a,b,c)
  y = zeros(size(a));
  % Index where the three numbers are positive
  iM = find(a > 0 & b > 0 & c > 0);
  y(iM) = min(min(a(iM),b(iM)),c(iM));
  % Index where the three numbers are negative
  im = find(a < 0 & b < 0 & c < 0);
  y(im) = max(max(a(im),b(im)),c(im));
end
```

Code 4 – Implementation of the minmod function - minmod.m

Finally the hearth of the method is contain in this last function file. Given U, the adiabatic constant, the minmod reconstruction constant, the mesh size and the Courant's number, it will compute the second member in (6) and the best time step with the formula (7).

```
function [q, dt] = qf(U,gamma,theta,dx,cfl)
  fU = f(U,gamma);
  c = speedofsound(U,gamma);
  a = abs(U(2,:)./U(1,:)) + c;

  % Compute f+ & f-
  fp = 0.5 * (fU + a.*U);
  fm = 0.5 * (fU - a.*U);

  % Compute df+
  dfp0 = theta*(fp(:,2:end-1) - fp(:,1:end-2)); % Option 1
  dfp1 = (fp(:,3:end) - fp(:,1:end-2))/2; % Option 2
  dfp2 = theta*(fp(:,3:end) - fp(:,2:end-1)); % Option 3

  dfp = minmod(dfp0,dfp1,dfp2);

  % Compute df-
  dfm0 = theta*(fm(:,2:end-1) - fm(:,1:end-2)); % Option 1
  dfm1 = (fm(:,3:end) - fm(:,1:end-2))/2; % Option 2
  dfm2 = theta*(fm(:,3:end) - fm(:,2:end-1)); % Option 3

  dfm = minmod(dfm0,dfm1,dfm2);

  fE = fp(:,2:end-1) + 0.5*dfp;
  fW = fm(:,2:end-1) - 0.5*dfm;

  % Compute f+0.5 and f-0.5
  fphalf = fE(:,2:end-1) + fW(:,3:end);
  fmhalf = fE(:,1:end-2) + fW(:,2:end-1);

  % Compute second member
  q = (fphalf - fmhalf)./dx;
  dt = cfl*min(dx)/max(a);
  end
```

Code 5 – Implementation of the computation of the second member - qf.m

In order to get clean code I grouped all the initialisation in one procedure. The following code illustrate the initialisation of the Blast wave problem on an uniform grid.

```
function [data] = initial(N,Nb,a,b,initial,nodes);
  data.cfl = 1;
  data.theta = 1.5;
  data.gamma = 1.4;
      data.dx = l/N;
      data.xG = [a-3*data.dx/2:data.dx:b+3*data.dx/2];
      P0 = zeros(size(data.x));
      P0(find(a < data.x & data.x < 0.1*l)) = 1000;
      P0(find(0.1*l < data.x & data.x < 0.9*l)) = 0.01;
      P0(find(0.9*l < data.x & data.x < b)) = 100;
      rho0 = ones(1,length(data.x));
      v0 = zeros(1,length(data.x));
      data.bound = 'wall';
  E0 = P0/(data.gamma-1) + 0.5*rho0.*v0.*v0;
  data.U(:,3:end-2) = [rho0;rho0.*v0;E0];
  data.U = boundary(data.U,data.bound);
end
```

Code 6 – Extract of the initialisation function - initial.m

And least the main function is setting all the parameters for the study of this case and process the SSP Runge Kutta method.

Two things to note in the code bellow, first on each Runge-Kutta sub-step the boundary conditions are applied. Second, I had this little conditions at the end of the programme to break the computation if the method become unstable (unbounded oscillations in pressure, induce negative pressure, hence negative square root for the speed of sound, hence imaginary numbers)..

```
% Initialsation
N = 6400;
a = 0;
b = 1;

% U at the next time step
U1 = zeros(size(U));
% Reserve memory space for the RK method
U1_1 = zeros(size(U));
U1_2 = zeros(size(U));

dt = 0;
time = 0;

while time < T
  % SSP RK order 3
  [q, dt] = qf(U,gamma,theta,dx,cfl);
  U1_1(:,3:end-2) = U(:,3:end-2) - dt*q;
  U1_1 = boundary(U1_1,bound);

  q1 = qf(U1_1,gamma,theta,dx,cfl);
  U1_2(:,3:end-2) = 0.75*U(:,3:end-2) + 0.25*U1_1(:,3:end-2) -
     0.25*dt*q1;
  U1_2 = boundary(U1_2,bound);

  q2 = qf(U1_2,gamma,theta,dx,cfl);
  U1(:,3:end-2) = (1/3)*U(:,3:end-2) + (2/3)*U1_2(:,3:end-2) -
     (2/3)*dt*q2;
  U1 = boundary(U1,bound);

  % All the quantities we are interested in
  rho = U(1,3:end-2);
  v = U(2,3:end-2)./rho;
  P = (gamma-1)*(U(3,3:end-2) - 0.5*rho.*v.*v);
  c = speedofsound(U(:,3:end-2),gamma);
  E = U(3,3:end-2);
  M = abs(v)./c;

  % Loop
  U = U1;

  if sum(imag(c) > 0) > 0 % in case of instability of the method,
     this criterion will stop the loop
    error('La␣simulation␣est␣instable');
  end
  time = time + dt;
end
```

Code 7 – Implementation of the main function - main_euler.m

5 Numerical Experiment for regular nodes distribution

In this section we will demonstrate the efficiency of the method on several typical example. All computations are done with 1000 nodes. Computation done with higher order method could be found in [8], thus we can convince ourselves that the method worked well.
I also recall that more results and some animations are available at `https://github.com/tschmoderer/euler-prj`.

5.1 Error and convergence

Because we don't know the analytic solution we adopt the *Manufactured Solution* strategy to check the convergence rate of the method :

1. We choose $q(x,t) \in \mathbb{R}^3$ as smooth as possible to be the solution of equations (1).
2. We put q in the Euler's equations and get, in general, a non zeros result (otherwise, it will mean we have an analytical solution, which is in general not possible). Let's call the rest $\mathcal{S}(x,t)$:
$$\frac{\partial q}{\partial t} + \frac{\partial f(q)}{\partial x} = \mathcal{S}$$
3. We now consider the modified problem :
$$\boxed{\begin{gathered}\frac{\partial U}{\partial t} + \frac{\partial f(U)}{\partial x} = \mathcal{S} \\ U(x,0) = q(x,0) \\ \text{Plus boundary conditions to be discusses bellow}\end{gathered} \qquad (9)}$$
For which we know the solution to be q.
4. We apply our method to this problem. It requires minor modification in the code to add the source term.
5. We can now compare the numerical solution to the analytical one. And compute the convergence rate.

A common choice in this approach is to choose periodic boundary conditions in keep smooth solutions (which is not systemically the case with wall conditions). Two choice of q are presented bellow in order to be certain that the method is of order 2.

5.1.1 Case 1

This case could be found in [9] (example 3.3).
For this first case we want the solution to be :

$$q = \left\{ \begin{array}{rcl} \rho(x,t) & = & 1 + 0.2\sin(2\pi(x-t)) \\ u(x,t) & = & 1 \\ P(x,t) & = & 1 \end{array} \right.$$

The solution have to be periodic that's why the choice of the sin function is natural to get both smooth and periodic properties. Then a little computation using the equation of state gives the solution for the energy :

$$E(x,t) = \frac{\gamma+1}{2(\gamma-1)} + 0.1\sin(2\pi(x-t))$$

Hence the initial conditions are the following :

$$\boxed{\begin{aligned} \rho(x,0) &= 1 + 0.2\sin(2\pi x) \quad \forall x \in \Omega \\ u(x,0) &= 1 \quad \forall x \in \Omega \\ P(x,0) &= 1 \quad \forall x \in \Omega \end{aligned} \qquad (10)}$$

In this particular case, the source term is identically zero :

$$\mathcal{S}(x,t) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Thus, here we got an analytical solution to the problem (which enough remarkable).
We make computation for 100, 200, 300, 400, 600, 800, 1200, 1600, 2400, 3200, 4000 6400, 9600, 12800, 19200, 25600, 38400, 51200 nodes and up to 1000 iterations. Hence we are able to compare the exact solution for the density and the numerical approximation. Figures 7 and 8 show the log of the error for the L^1 and L^∞ norms against the log of the number of nodes.

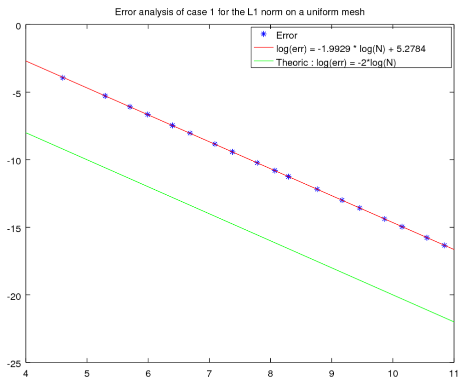


FIGURE 7 – Convergence rate for manufactured solution # 1 in L^1 norm

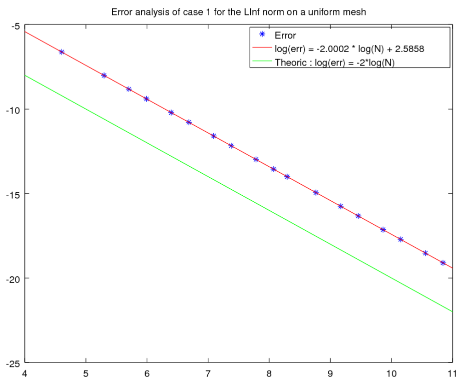


FIGURE 8 – Convergence rate for manufactured solution # 1 in L^∞ norm

Where the L^1 and L^∞ norms are the following :

$$\|U^{ex} - U^a\|_1 = \sum_{j=1}^{N} |U^{ex}(x_j) - U^a_j| \qquad \|U^{ex} - U^a\|_\infty = \max_{j=1..N} |U^{ex}(x_j) - U^a_j|$$

The $\log error$ perfectly fit a -2 slope line. Hence as we hoped, the method seems to be of order 2.

5.1.2 Case 2

The idea behind this case comes mainly from [10]
The second case will comfort us in the fact tat the method is of order 2. We want the

solution to be :

$$q = \begin{cases} \rho(x,t) &= 2 + 0.1\sin(2\pi(x-t)) \\ u(x,t) &= 1 \\ E(x,t) &= 2 + 0.1\cos(2\pi(x-t)) \end{cases}$$

Then, the initial conditions are the following :

$$\boxed{\begin{aligned} u(x,0) &= 1 \quad \forall x \in \Omega \\ \rho(x,0) &= 2 + 0.1\sin(2\pi x) \quad \forall x \in \Omega \\ P(x,0) &= \frac{\gamma-1}{20}\left(20 + 2\cos(2\pi x) - sin(2\pi x)\right) \quad \forall x \in \Omega \end{aligned}} \tag{11}$$

And the source term is given by :

$$\boxed{\mathcal{S}(x,t) = (1-\gamma)\pi(2\rho(x,t) + E(x,t) - 6)\begin{pmatrix}0\\1\\1\end{pmatrix}} \tag{12}$$

After modifications of the code to handle the source term, we again make computation for 100, 200, 300, 400, 600, 800, 1200, 1600, 2400, 3200, 4000 6400, 9600, 12800, 19200, 25600, 38400, 51
nodes with step of 100 nodes up to 1000 iterations. Hence we are able to compare the exact solution and the numerical approximation. Figures 9 and 10 shows the log of the error against the log of the number of nodes.

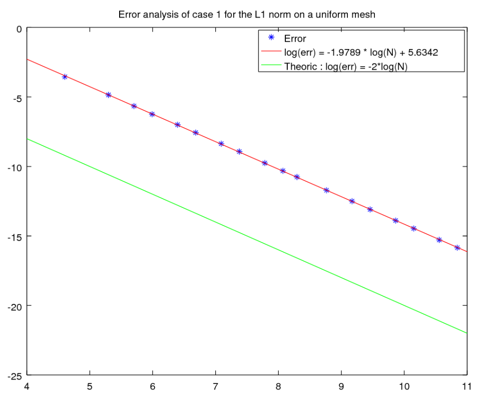


FIGURE 9 – Convergence rate for manufactured solution # 2 in L^1 norm

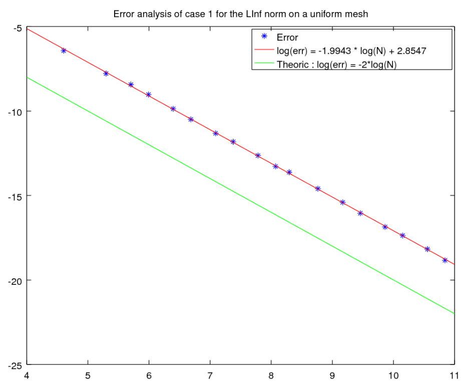


FIGURE 10 – Convergence rate for manufactured solution # 2 in L^∞ norm

Again the convergence rate is really near 2. We can conclude that the method it of order 2 as announced.

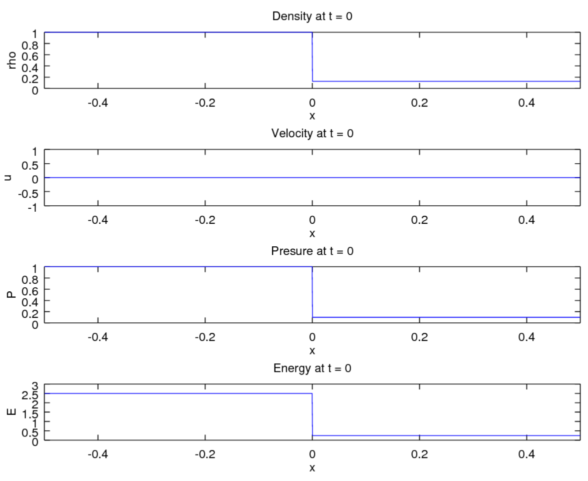
Having satisfying ourselves with the convergence rate we can now observe the method working on several typical example. For each example, I will try to give a real-life situation where the phenomenon describe could occur.

5.2 Sod shock tube

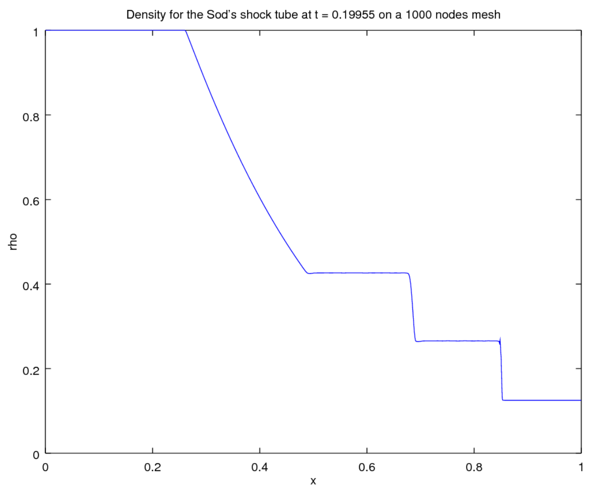
We begin our collection of test with a popular one. Imagine a pipe joining a gas tank from a factory (high pressure and density of gas) to a city (small pressure and density), at $t = 0$ the distribution center decide to open a valve to bring gas to the city.
The domain is $\Omega = [0, 1]$, and the initial conditions are :

$$\boxed{\begin{aligned} u(x,0) &= 0 \qquad \forall x \in \Omega \\ \rho(x,0) &= \begin{cases} 1.0 & x \in [0, 0.5] \\ 0.125 & x \in [0.5, 1] \end{cases} \\ P(x,0) &= \begin{cases} 1.0 & x \in [0, 0.5] \\ 0.1 & x \in [0.5, 1] \end{cases} \end{aligned} \qquad (13)}$$

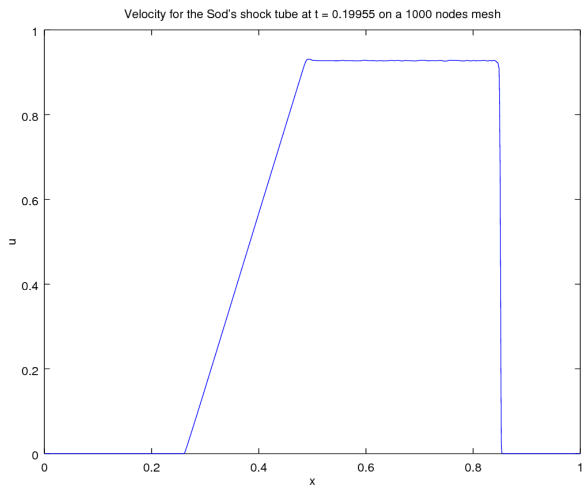
On this we apply wall boundary conditions and look at the result around $t = 0.2$.



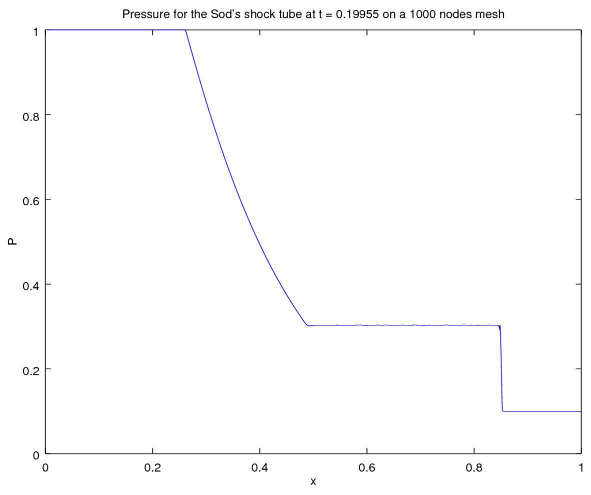
(a) Initial conditions



(b) Density



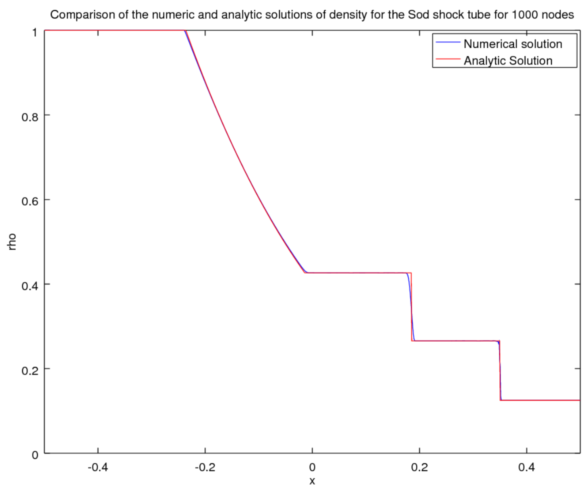
(c) Velocity



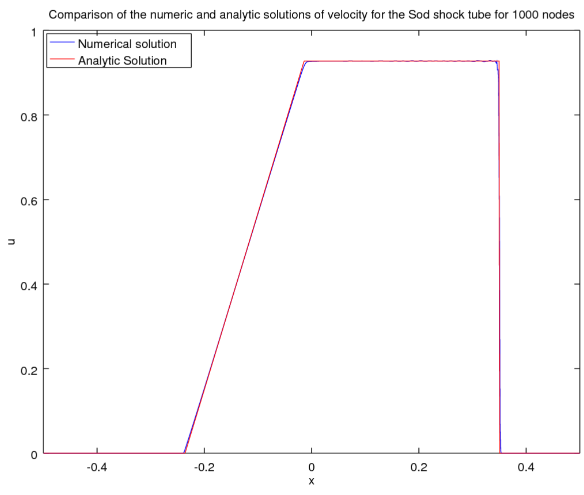
(d) Pressure

FIGURE 11 – Result for the Sod's shock tube

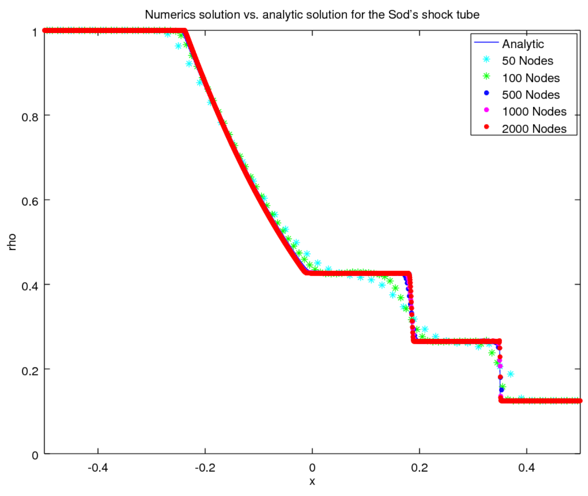
What we observe from this simulation, relative to our gas-factory problem is that a shock wave is going to the city and if (maybe) device are not calibrate strong enough this wave could damage some material. Another interesting fact is the rarefacting wave going backward to the factory.
Based on [11] (and the code in [12]) we can compute the analytic solution for this case. On the figure bellow we can see the difference between the analytic solution and the approximation we computed :



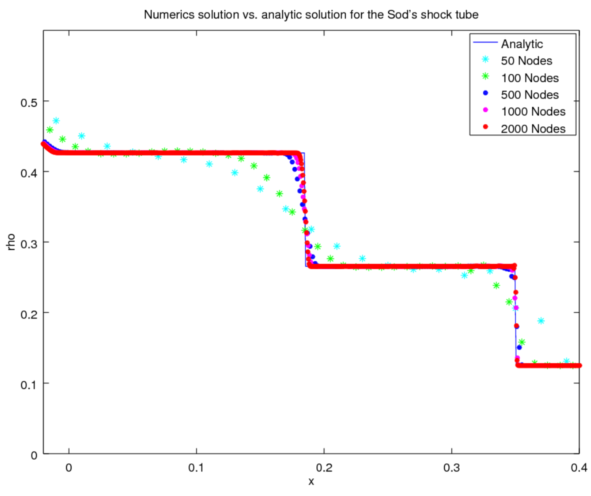
(a) Density



(b) Velocity



(c) Multi nodes



(d) Multi nodes zoom

FIGURE 12 – Result for the Sod's shock tube with the analytic solution and for different mesh size

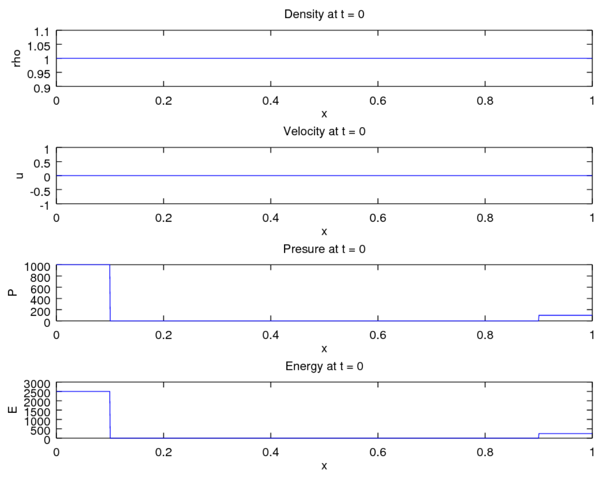
As we can see the method is not perfect. However the results are quiet really good and close to the analytic solution. We see on figures 12-(c) and 12-(d) that the more there is nodes, the closest is the numerical result to the analytic solution. This valid our intuition that the method converged.

5.3 Interacting blast wave

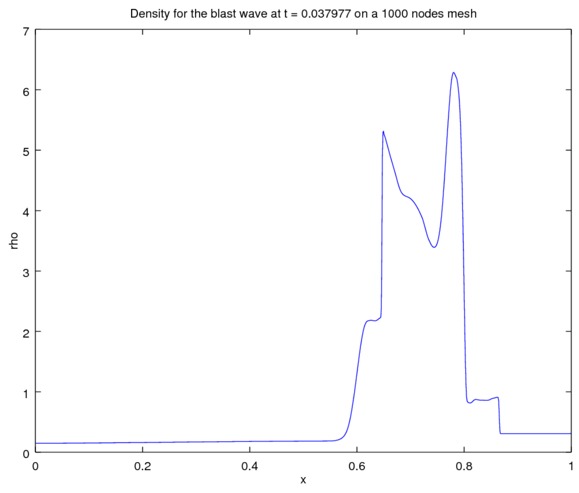
See [13] and [9] (example 3.7). Almost the same story as before, but this time there is two gas tank providing gas to the city, and two valves are open at $t = 0$.
The initial conditions are :

$$\boxed{\begin{aligned} u(x,0) &= 0 \qquad \forall x \in \Omega \\ \rho(x,0) &= 1 \qquad \forall x \in \Omega \\ P(x,0) &= \begin{cases} 1000 & x \in [0, 0.1] \\ 0.01 & x \in [0.1, 0.9] \\ 100 & x \in [0.9, 1] \end{cases} \end{aligned} \tag{14}}$$

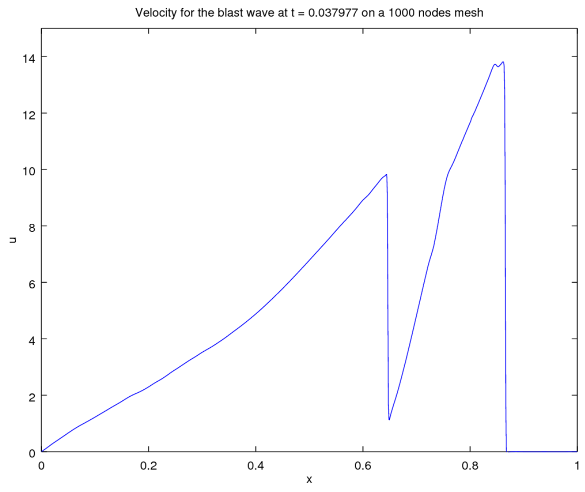
We use again solid wall boundary conditions and watch the result around $t = 0.038$.



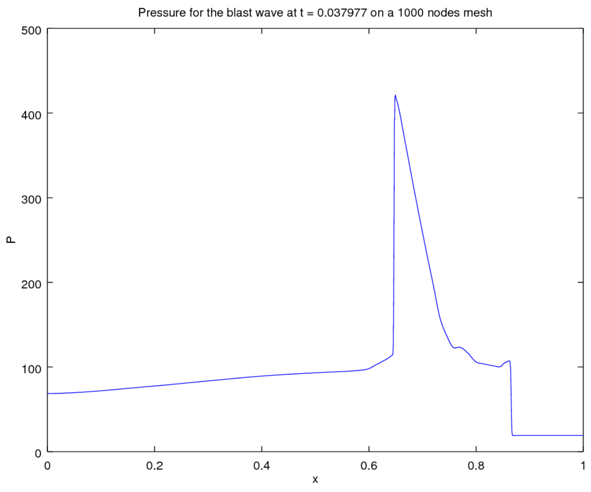
(a) Initial conditions



(b) Density



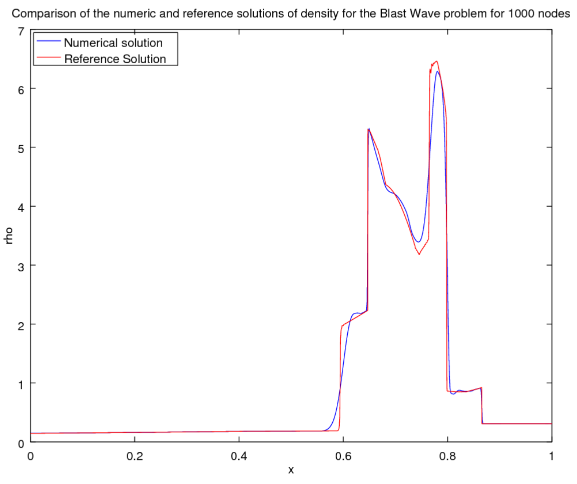
(c) Velocity



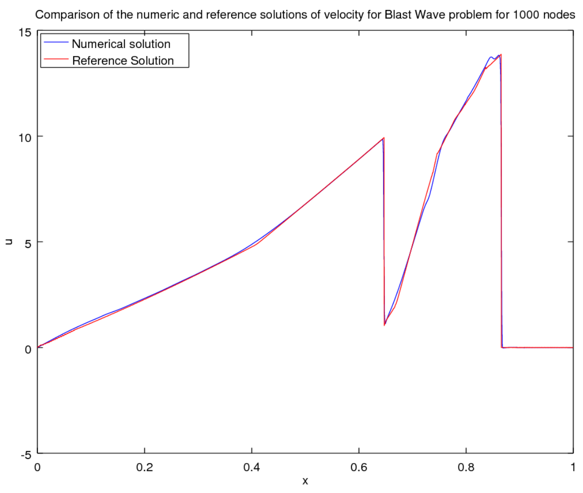
(d) Pressure

FIGURE 13 – Result for the blast wave

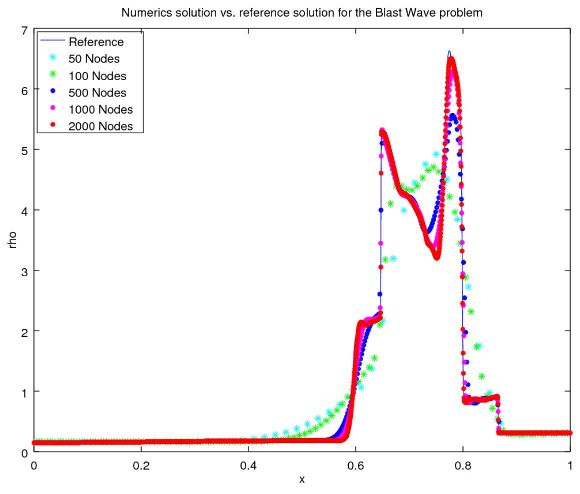
The conclusion of our story is that better not to put a weak device arroun $x = 0.6$. Unfortunately there is no analytic solution for this case. Hence, in order to observe the convergence of the method I used a WENO-5 method, from [14], to compute a reference solution with 16000 nodes.



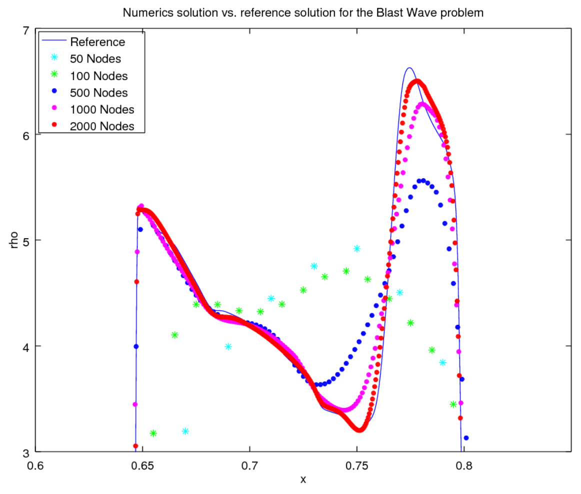
(a) Density



(b) Velocity



(c) Multi nodes



(d) Multi nodes zoom

FIGURE 14 – Result for the blast wave problem with the reference solution and for different mesh size

We can observe that the solution computed is close from the reference one. But we got a a bit more errors than in the previous case. This is probably due to the complexity of the case and the numerical pollution from Matlab execution. However we get the general forms of the solution and the shocks are well disposed so the method did not compute all wrong.

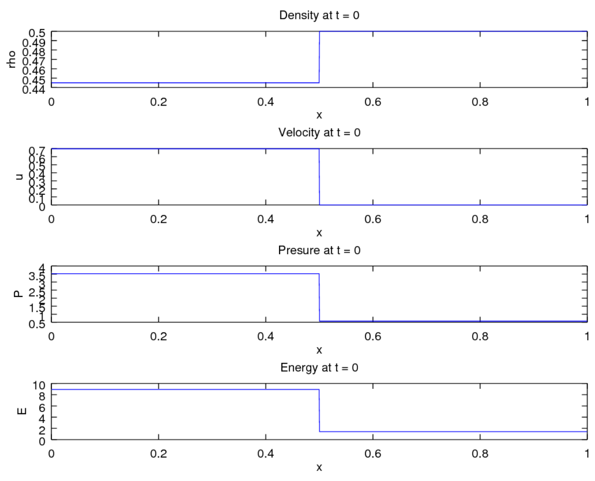
5.4 Lax's shock tube

See [9] (example 3.5). The initial conditions are :

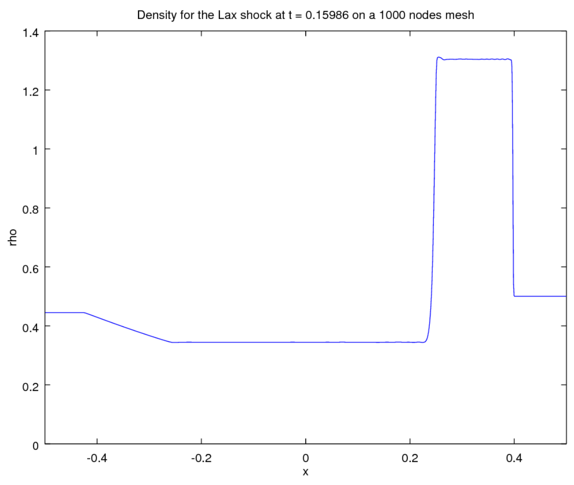
$$
u(x,0)=\begin{cases}0.698 & x\in[0,0.5]\\ 0 & x\in[0.5,1]\end{cases}
$$
$$
\rho(x,0)=\begin{cases}0.445 & x\in[0,0.5]\\ 0.5 & x\in[0.5,1]\end{cases} \qquad (15)
$$
$$
P(x,0)=\begin{cases}3.528 & x\in[0,0.5]\\ 0.571 & x\in[0.5,1]\end{cases}
$$

We use inlet conditions on the left and solid wall conditions in the right boundary and watch the result around $t = 0.16$. Inlet conditions are :

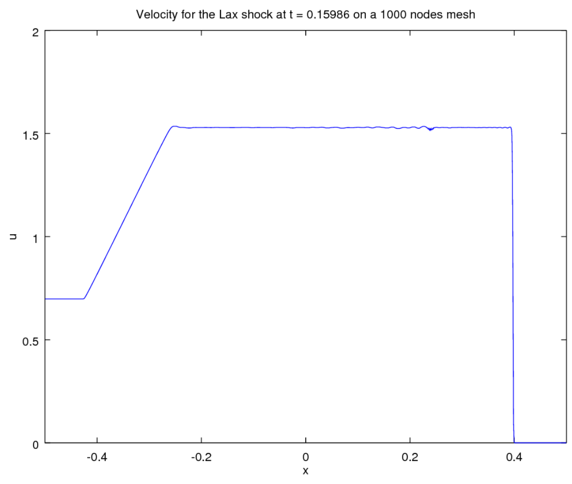
$$
u(0,t)=0.698 \quad \rho(0,t)=0.445 \quad P(0,t)=3.528
$$



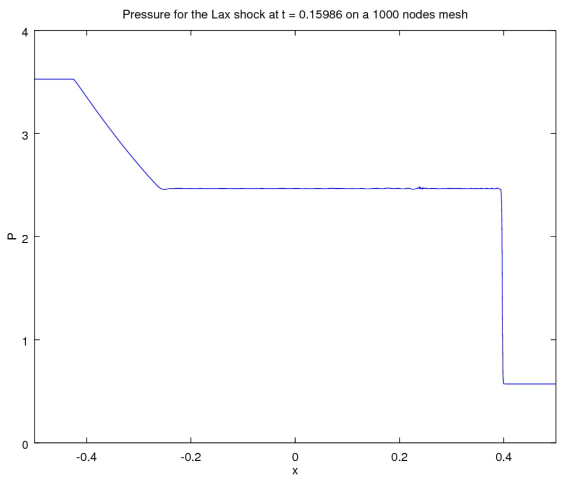
(a) Initial conditions



(b) Density



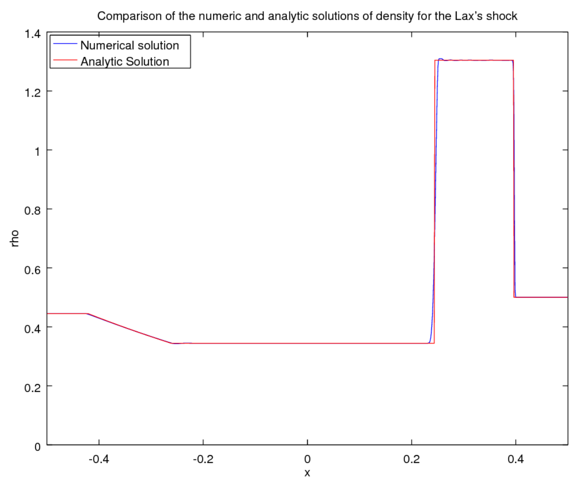
(c) Velocity



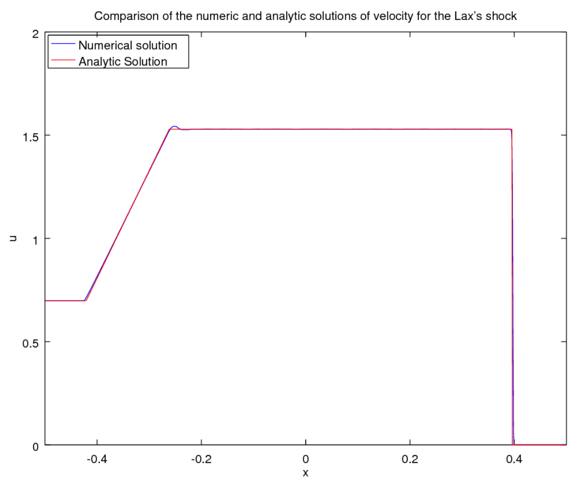
(d) Pressure

FIGURE 15 – Result for the Lax's shock tube

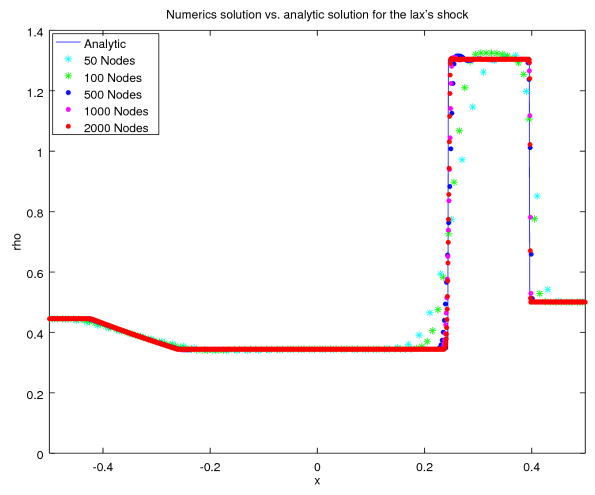
For this case, as for the Sod's shock tube, there exists an analytic solutions. Thus we can compare our method to this solution.



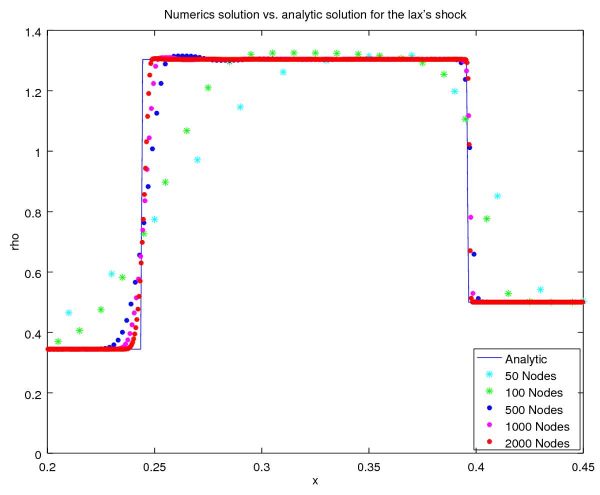
(a) Density



(b) Velocity



(c) Multi nodes



(d) Multi nodes zoom

FIGURE 16 – Result for the Lax's shock tube with the analytic solution and for different mesh size

Again, we observe that the method fit well the analytic solution and captured the shocks.

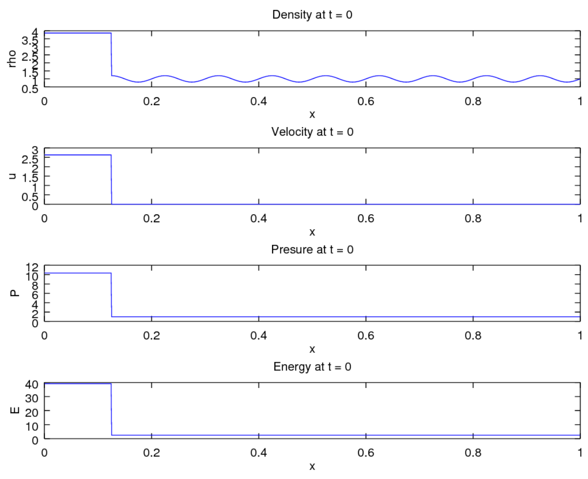
5.5 Shu-Osher's problem

See [15] and [9] (example 3.6). The initial conditions are :

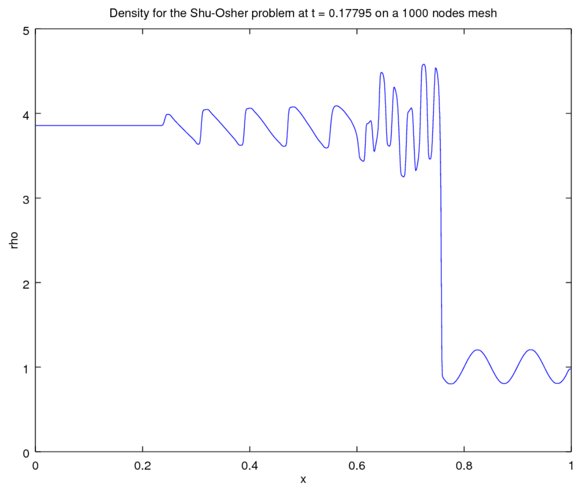
$$
\boxed{
\begin{aligned}
u(x,0) &= \begin{cases} 2.629369 & x \in [0, 0.125] \\ 0 & x \in [0.125, 1] \end{cases} \\
\rho(x,0) &= \begin{cases} 3.857143 & x \in [0, 0.125] \\ 1 + 0.2\sin(20\pi x) & x \in [0.125, 1] \end{cases} \\
P(x,0) &= \begin{cases} 31/3 & x \in [0, 0.125] \\ 1 & x \in [0.125, 1] \end{cases}
\end{aligned}
} \tag{16}
$$

This time we use inlet conditions on the left and outflow conditions on the right. The inlet conditions are :

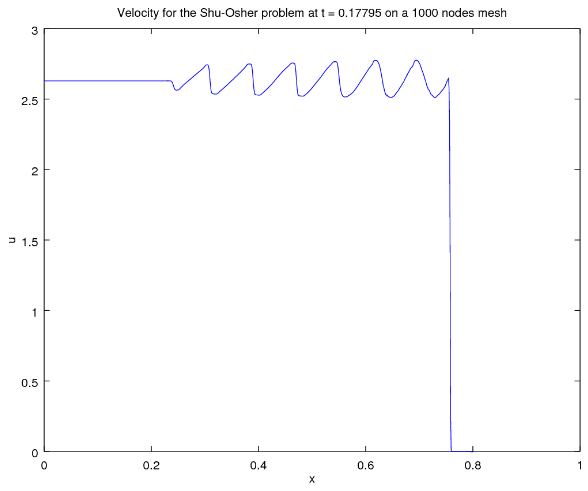
[illegible]



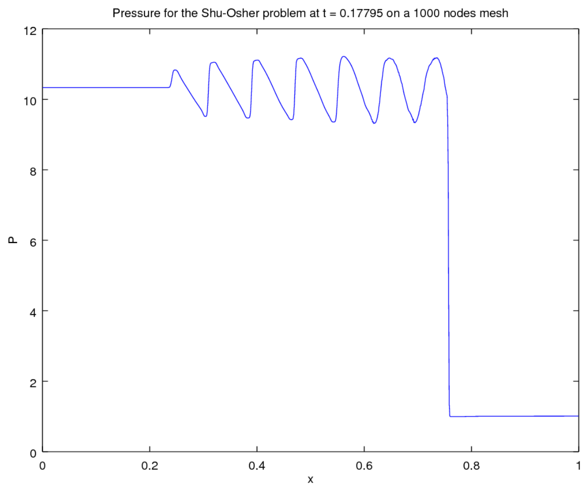
(a) Initial conditions



(b) Density



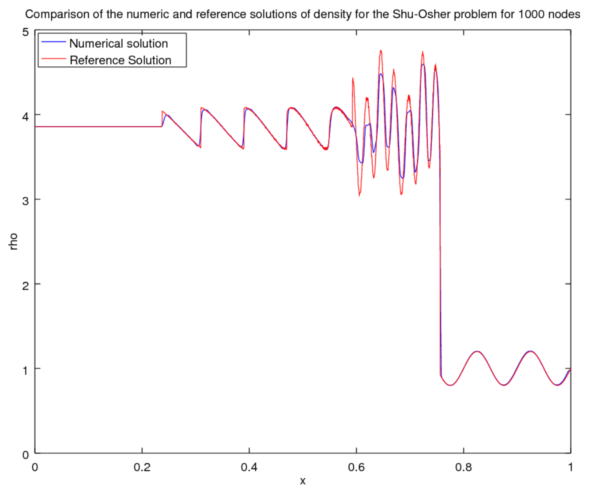
(c) Velocity



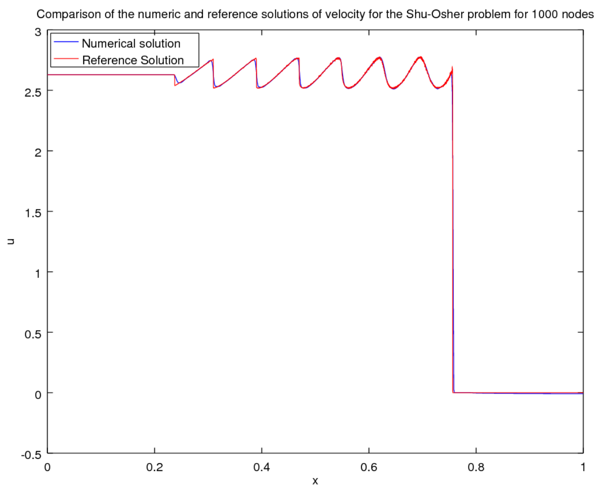
(d) Pressure

FIGURE 17 – Result for the Shu-Osher's problem

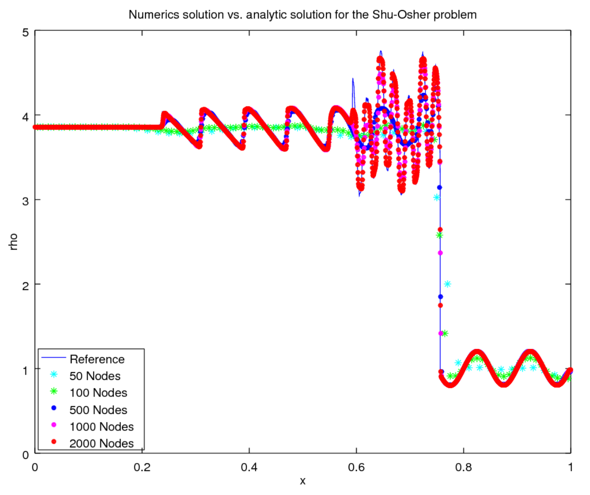
Again we don't know any analytic solution for this case, so the reference solution is given by the WENO-5 method on 16000 nodes.



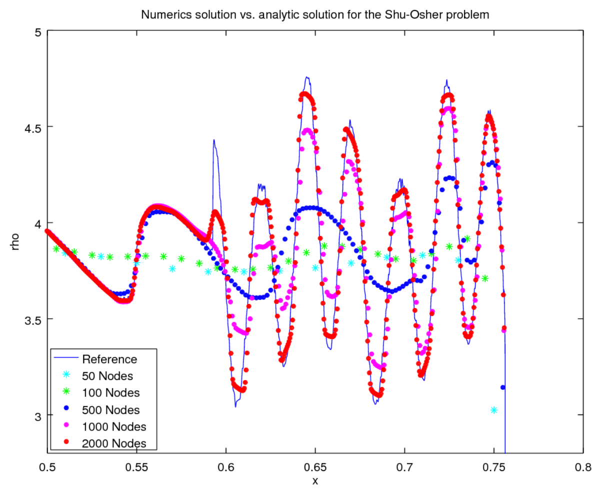
(a) Density



(b) Velocity



(c) Multi nodes



(d) Multi nodes zoom

FIGURE 18 – Result for the Shu-Osher problem with the reference solution and for different mesh size

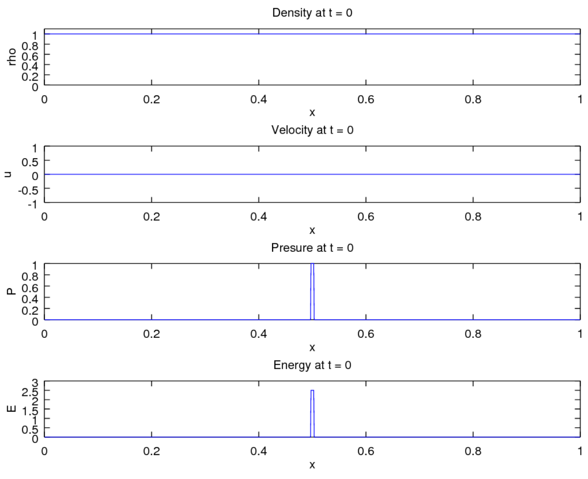
We definitely observe that the method converge to the reference solution.

5.6 Sedov explosion

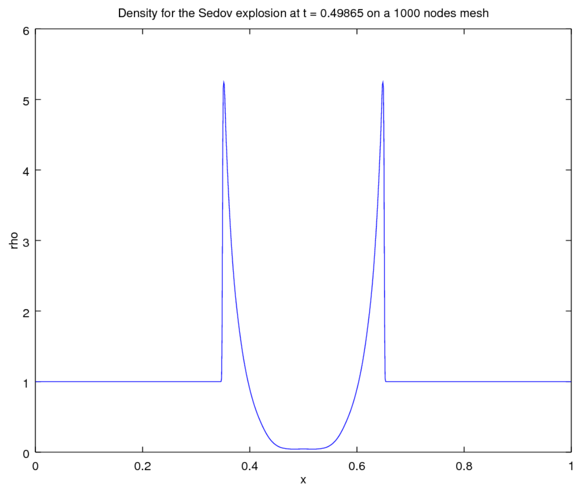
The initial conditions are :

$$
\boxed{
\begin{aligned}
u(x,0) &= 0 \qquad \forall x \in \Omega \\
\rho(x,0) &= 1 \qquad \forall x \in \Omega \\
P(x,0) &= \begin{cases} 1 & x \in [0.5 - 3.5\dfrac{\Delta x}{2}, 0.5 + 3.5\dfrac{\Delta x}{2}] \\ 10^{-5} & \text{else} \end{cases}
\end{aligned}
} \qquad (17)
$$

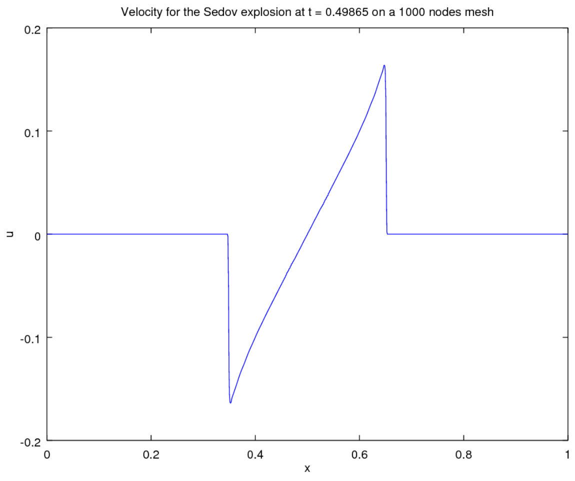
We use again solid wall boundary conditions and watch the result around $t = 0.005$.



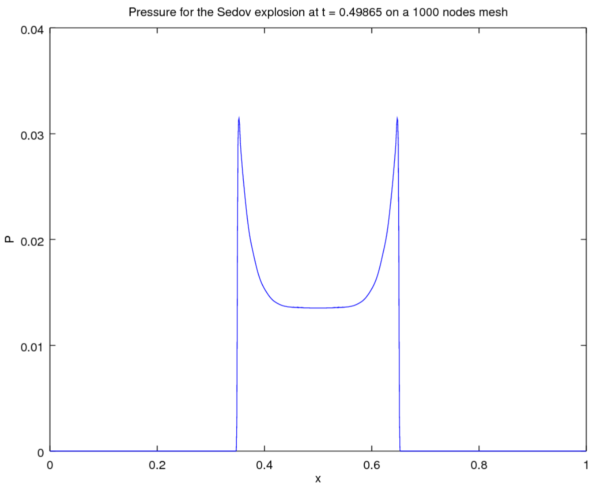
(a) Initial conditions



(b) Density



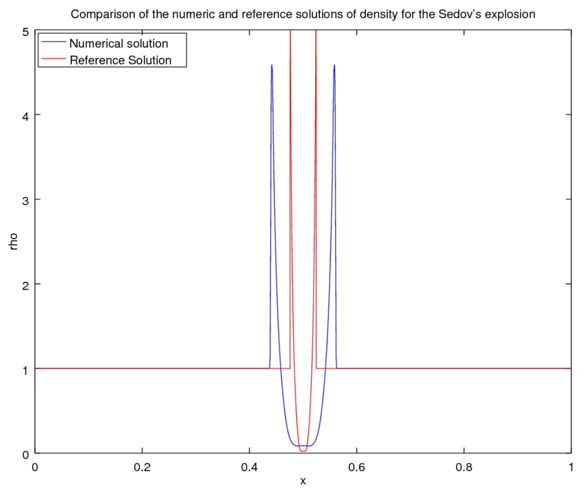
(c) Velocity



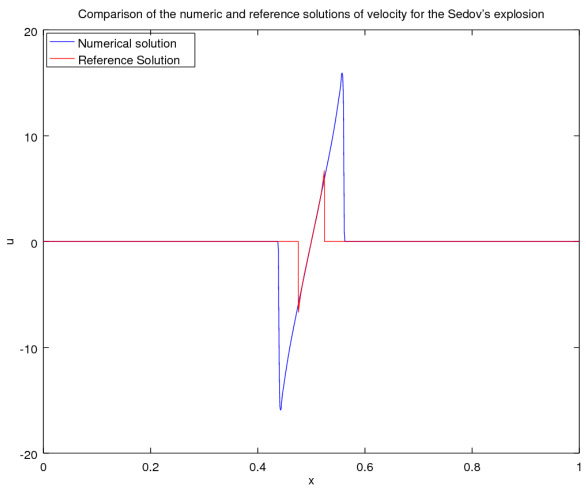
(d) Pressure

FIGURE 19 – Result for the Sedov's explosion

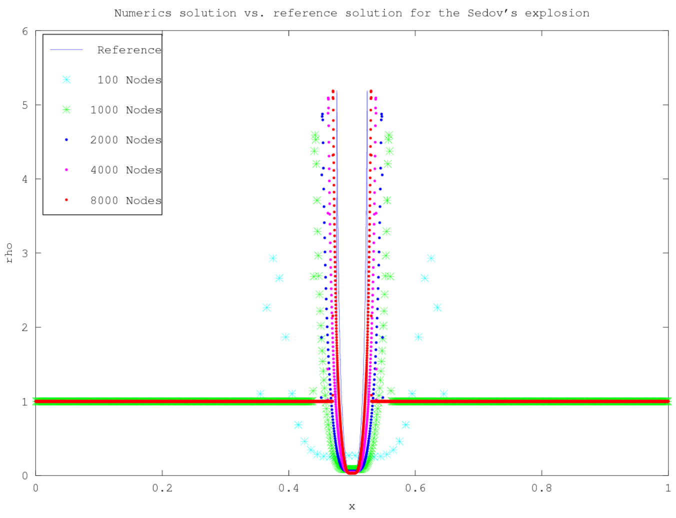
Again we don't know any analytic solution for this case, so the reference solution is given by the WENO-5 method on 16000 nodes. We watch the result at $t = 0.002$.



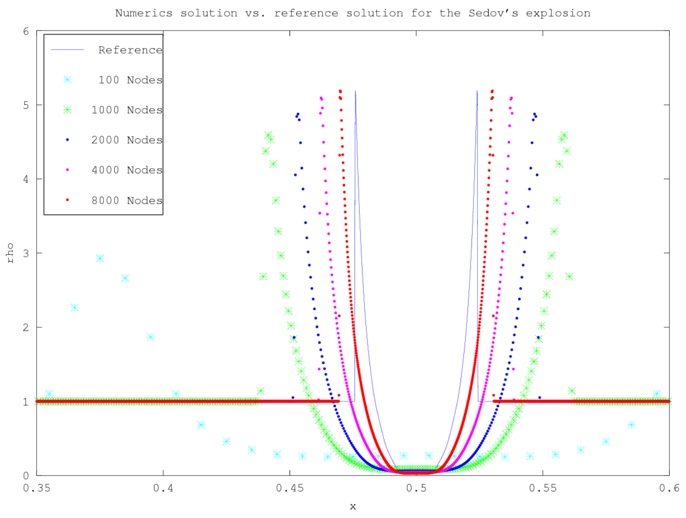
(a) Density



(b) Velocity



(c) Multi nodes



(d) Multi nodes zoom

FIGURE 20 – Result for the Sedov explosion with the reference solution and for different mesh size

This case is very special, because of the sharp discontinuities a lot more of nodes are needed to get a good approximation of the discontinuities.
It is also a good example of nodes-waste : to get a good approximation we should compute on a very coarse grid near the shock front but we don't need a lot of nodes in the extremities. Here an adaptive mesh would improve the execution time and with the same number of nodes (but better disposed) we could achieve greater accuracy.

6 Adaptation on non-regular meshes

We are now interested in adapting our method to non regular meshes. After presenting the new mesh disposition, we will perform again the error test and several example to illustrate the robustness of the method.

6.1 The nodes

The particular distribution of nodes we are interested in is the Gauss-Lobatto nodes. First we defined the n-th Legendre polynomial as follows.

$$\boxed{\begin{aligned} P_0(x) &= 1 \qquad \forall x \in [-1,1] \\ P_1(x) &= x \qquad \forall x \in [-1,1] \\ (n+1)P_{n+1}(x) &= (2n+1)xP_n(x) - nP_{n-1}(x) \qquad \forall x \in [-1,1] \end{aligned} \qquad (18)}$$

The nodes are the $N-2$ roots of the derivative of $(n-1)$-th Legendre polynomial with the two extremities of the domain :

$$-1 = x_1 < x_i \ : \ P'_{n-1}(x_i) = 0 \quad i \in [\![2, N-1]\!] < x_N = 1$$

We can easily shifted them into $[a,b]$ with the formula :

$$x'_i = \frac{b-a}{2}x_i + \frac{a+b}{2}$$

The nodes are computed with the Haylley's method [4], see [16] for further details. The recursion is the following :

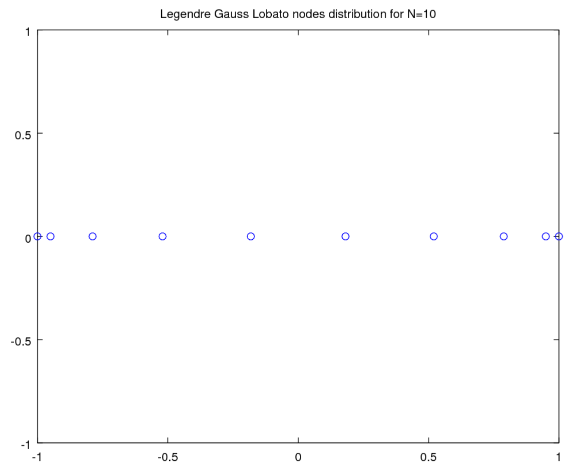
$$\boxed{\begin{aligned} x_0 &= \left[-1, \left(1 - \frac{3(N-1)}{8N^3}\right)\cos\left(\frac{4j+1}{4N+1}\pi\right), 1\right] \quad j \in [\![1, N-1]\!] \\ x_{n+1} &= x_n - 2 * \frac{P'_{n-1}(x_n)P''_{n-1}(x_n)}{2[P''_{n-1}(x_n)]^2 - P'_{n-1}(x_n)P'''_{n-1}(x_n)} \end{aligned} \qquad (19)}$$

And the derivative are given by :

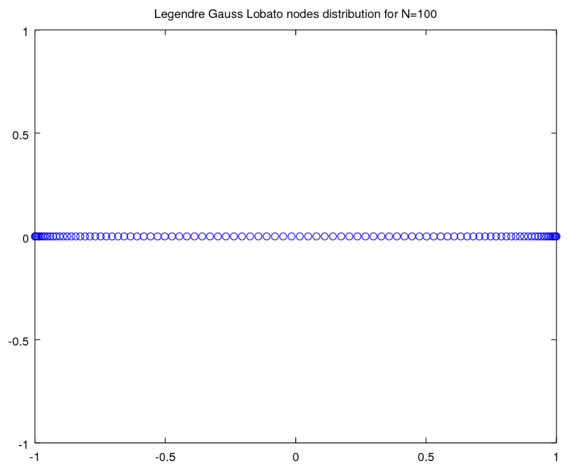
$$\begin{aligned} P'_n(x) &= \frac{n(P_{n-1}(x) - xP_n(x))}{1-x^2} \\ P''_n(x) &= \frac{2xP'_n(x) - n(n+1)P_n}{1-x^2} \\ P'''_n(x) &= \frac{2xP''_n(x) - (n(n+1)-2)P_n}{1-x^2} \end{aligned}$$

As we can see on the picture bellow, the distribution is dense on the extremities of the domain and really spaced in the center.

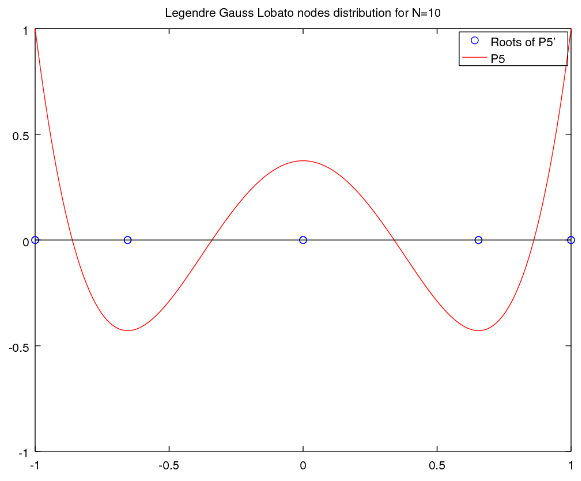
4. A method of order 3 similar to the method of Newton.



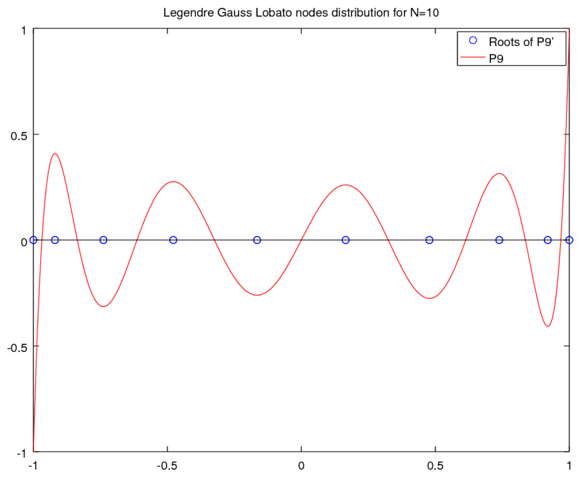
(a) Mesh $N = 10$



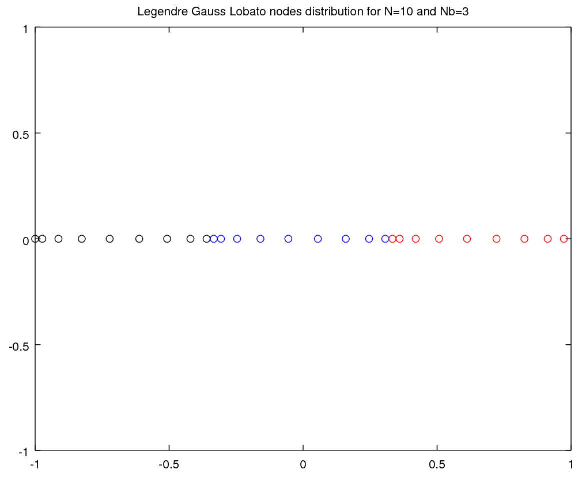
(b) Mesh $N = 100$



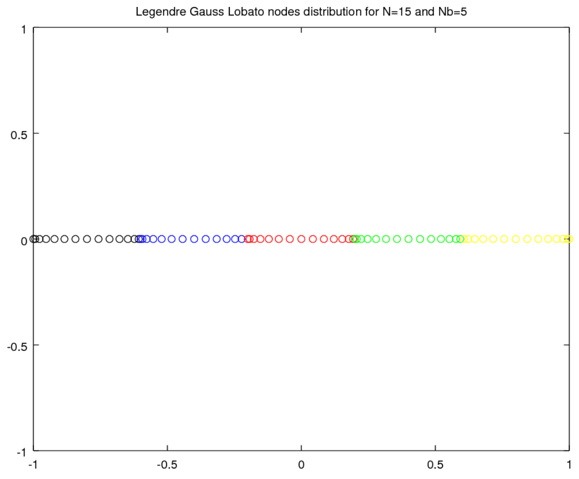
(c) Mesh $N = 5$ with P_5



(d) Mesh $N = 10$ with P_9



(e) Mesh $N = 10$ in $N_b = 3$ blocks



(f) Mesh $N = 15$ in $N_b = 5$ blocks

FIGURE 21 – Legendre-Gauss-Lobato mesh

On figure 21-(c) and 21-(d) we observe that the nodes we computed fit the extrema of the polynomial that is to say we get the roots of the derivative polynomial. It confirms that the code worked well.
We divide our interval $[a, b]$ into N_b equal blocks, and apply this mesh in each block, as we can see on 21-(e) and 21-(f).

Here is the code used to compute the nodes :

```
function x = lglnodes(N,a,b)

  N = N-1;
  P = zeros(N+1,3);
  x = 2;

  % First guess
  x0 = [-1 (1-3*(N-1)/(8*(N)^3))*cos(pi*(4*[1:N-1]+1)/(4*(N)+1))
     1]';

  while max(abs(x-x0)) > eps
    x = x0;

    % Construct the P_n(x)
    P(:,1) = 1;
    P(:,2) = x0;

    for n=2:N-1
    P(:,3) = ((2*n-1)*x0.*P(:,2)-(n-1)*P(:,1))/n;
    P(:,1) = P(:,2);
    P(:,2) = P(:,3);
    end
    P(:,3) = ((2*N-1)*x0.*P(:,2)-(N-1)*P(:,1))/N;

    % Get Pn'
    dP = N*(P(:,end-1)-x0.*P(:,end));%./(1-x0.^2);
    % Get Pn''
    ddP = (2*x0.*dP-N*(N+1)*P(:,end));%./(1-x0.^2);
    % Get Pn'''
    dddP = (2*x0.*ddP-(N*(N+1)-2)*dP);%./(1-x0.^ 2);

    % Apply Hayleys method
    x0 = x - 2*dP.*ddP./(2*ddP.^2-dP.*dddP);
  end

  % Nodes in [a:b]
  x = 0.5*((b-a)*x0' + (a+b));
end
```

Code 8 – Implementation of the construction of the LGL nodes - lglnodes.m

Please note that due to the Matlab incrementation, N is shifted by one.
Before we continue, I precise that we computed the cell edges of our domain and by taking the mean value for two consecutive nodes we get the cell's centers.

7 Numerical experiment for non regular nodes distribution

In the next section we will compute the convergence rate of the method on the non uniform grid. Then we will see the method on only two examples : the Blast Wave 5.3 and the Shu-Osher problem 5.5, because the pictures are essentially the same. We do not present again the initial conditions which are the same as in section 5.
Before beginning, a preliminary remarks about Δt. As we saw in section 3.4, the time step depends on the mesh size. Because we don't have a regular mesh here, we use the following integration time step which guaranty that we don't exceed the Courant's number and let the method stable :

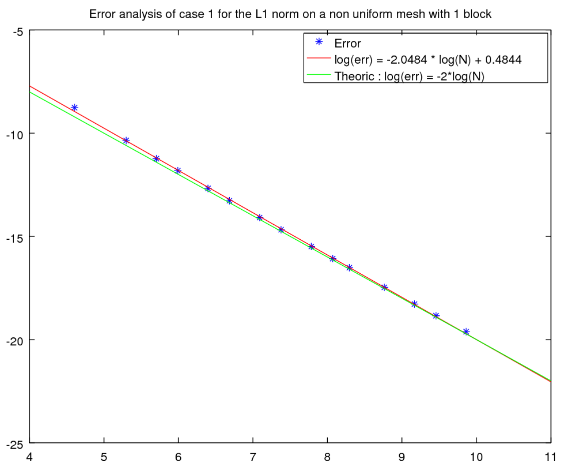
$$\Delta t^n = \frac{\min \Delta x}{\max a^n}$$

7.1 Error and convergence

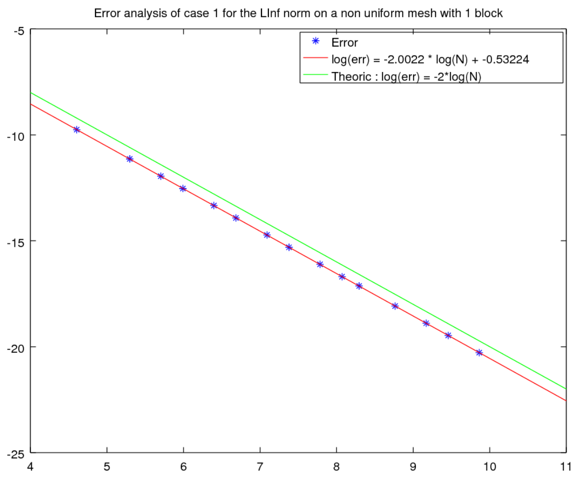
As before we computed the convergence rate with the method of the manufactured solutions. We take the two same previous cases :

$$\begin{aligned}
&\text{Case 1 :} \begin{cases} \rho(x,0) = 1 + 0.2\sin(2\pi x) \quad \forall x \in \Omega \\ u(x,0) = 1 \quad \forall x \in \Omega \\ P(x,0) = 1 \quad \forall x \in \Omega \end{cases} \\
&\mathcal{S}_1(x,t) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\
&\text{Case 2 :} \begin{cases} u(x,0) = 1 \quad \forall x \in \Omega \\ \rho(x,0) = 2 + 0.1\sin(2\pi x) \quad \forall x \in \Omega \\ P(x,0) = \dfrac{\gamma - 1}{20}\left(20 + 2\cos(2\pi x) - sin(2\pi x)\right) \quad \forall x \in \Omega \end{cases} \\
&\mathcal{S}_2(x,t) = (1-\gamma)\pi(2\rho(x,t) + E(x,t) - 6)\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}
\end{aligned} \tag{20}$$

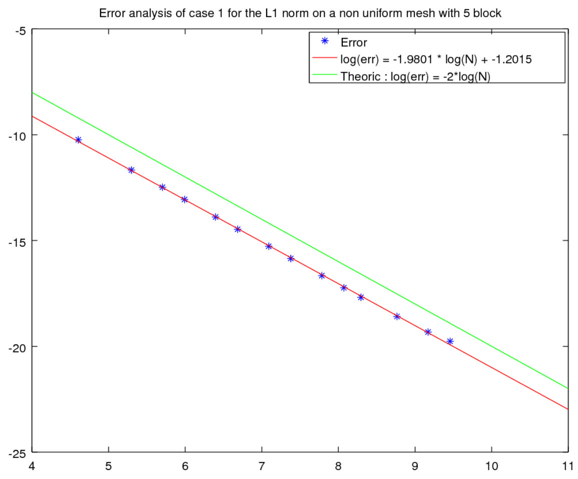
The results are quiet surprising. The convergence rate is the same as for the uniform case. It is surprising that we got such perfect fitting to straight lines. We could imagine that due to this particular distribution, the error would fluctuate more.
It also seems that the convergence rate doesn't depends on the number of blocks.



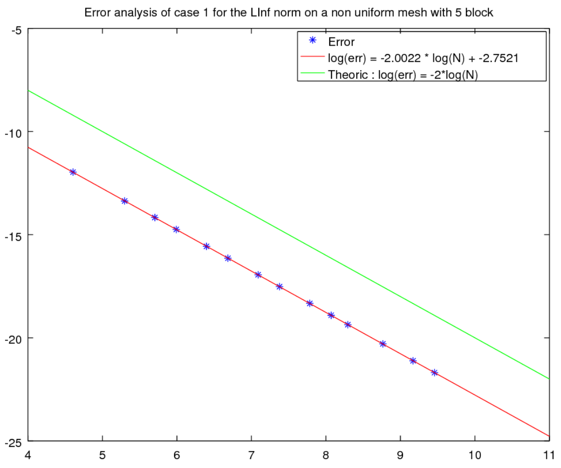
(a) Case 1 : Error L^1 with 1 block



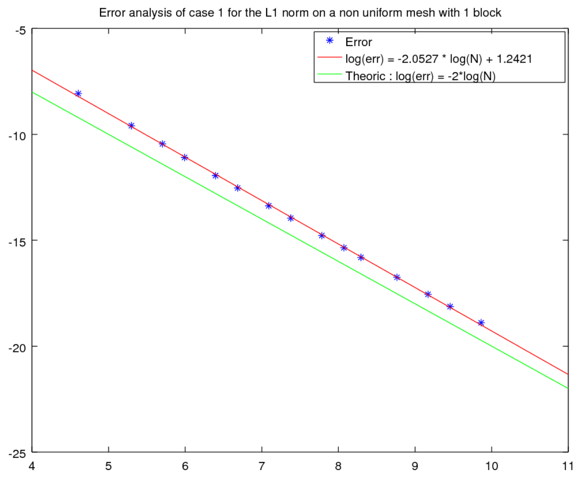
(b) Case 1 : Error L^∞ with 1 block



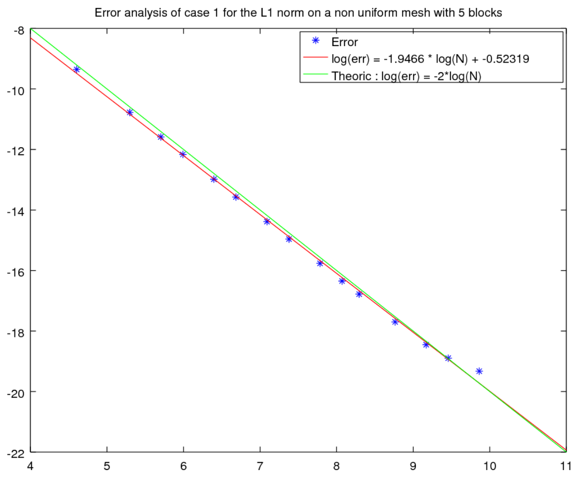
(c) Case 1 : Error L^1 with 5 blocks



(d) Case 1 : Error L^∞ with 5 blocks



(e) Case 2 : Error L^1 with 1 block



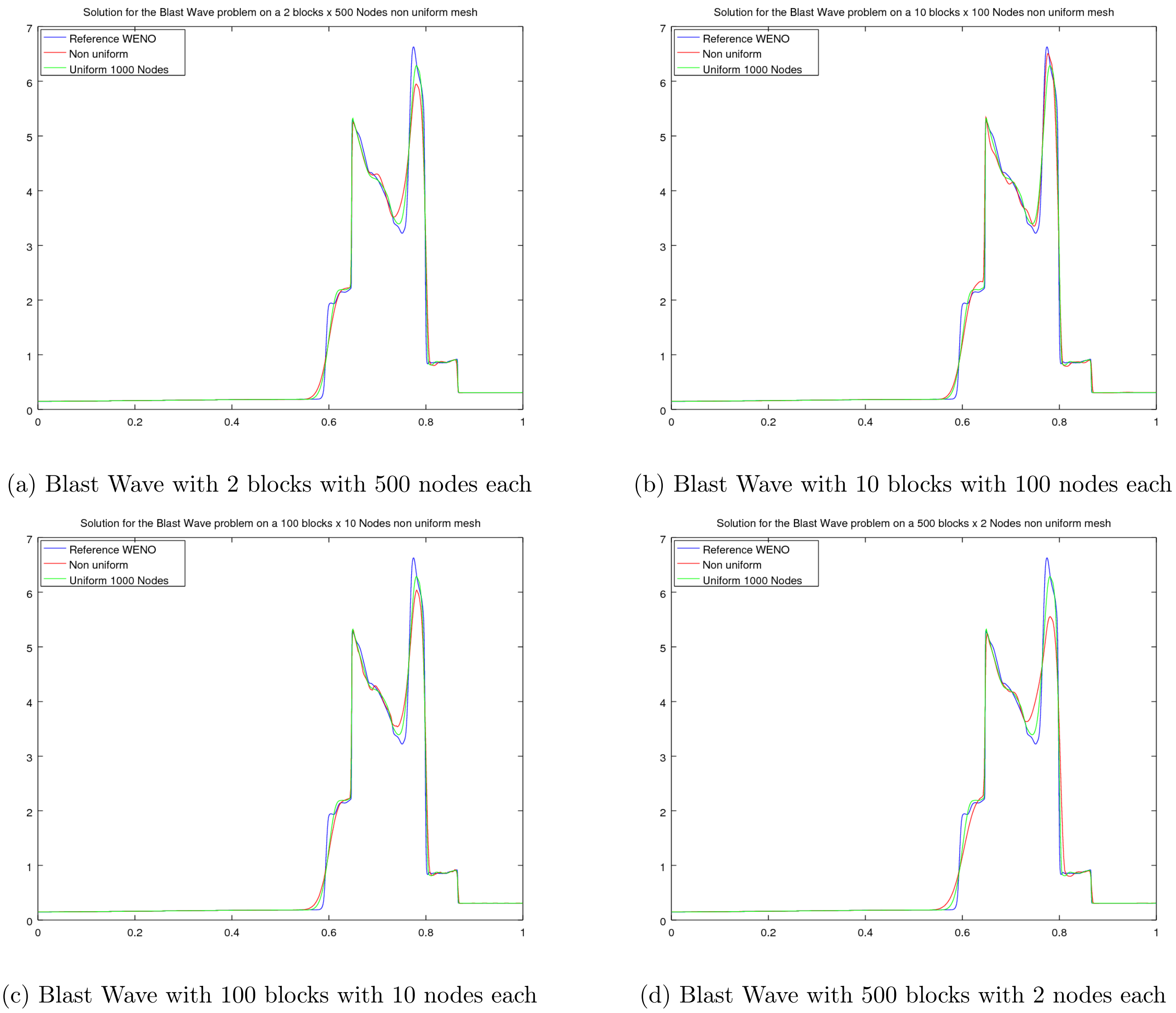
(f) Case 2 : Error L^1 with 5 blocks

FIGURE 22 – Error analysis for non uniform meshes

7.2 Interacting Blast Wave

As announced before, the next page will show the result computed for the interacting blast wave problem. We show the result for different number of blocks but we always kept the total number of nodes to be 1000.
On each figure are plotted the reference solution computed with WENO (16000 nodes), the solution computed on a 1000 uniform mesh and the solution on the non uniform mesh.

My observation are the following, the method converged well on this grid and the shock are well disposed. This solution seems to be smoother than the uniform-one. According to me, that's the reason why the non-uniform-solution is less accurate than the other (particularly on the right pick). Another little inconvenient of the method, is that the execution time is a bit longer due to the dense distribution on the extremities (the ore there is nodes in a block the most time it took) which give a small Δt.



(a) Blast Wave with 2 blocks with 500 nodes each

(b) Blast Wave with 10 blocks with 100 nodes each

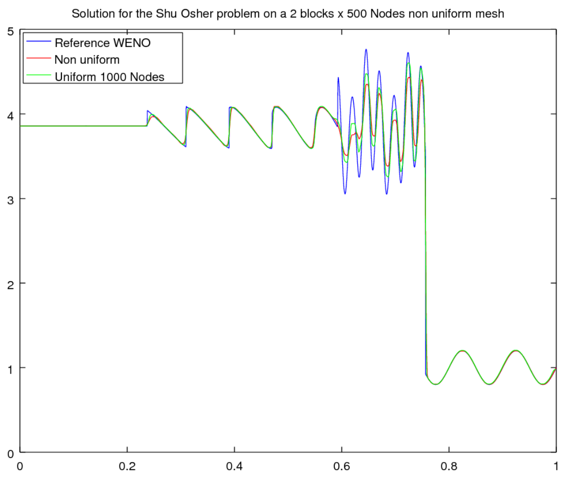
(c) Blast Wave with 100 blocks with 10 nodes each

(d) Blast Wave with 500 blocks with 2 nodes each

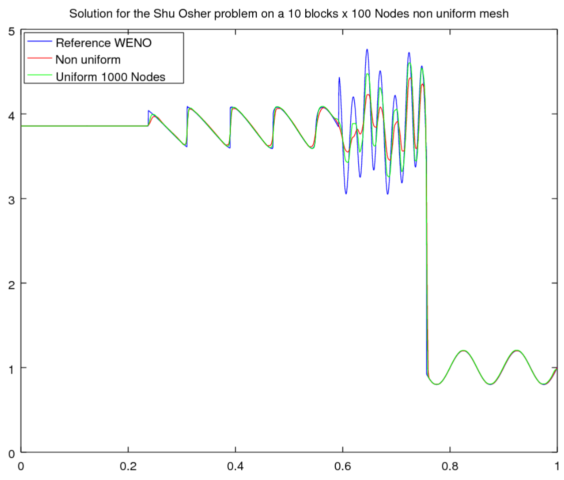
FIGURE 23 – Interacting blast wave on non uniform mesh

7.3 Shu Osher Problem

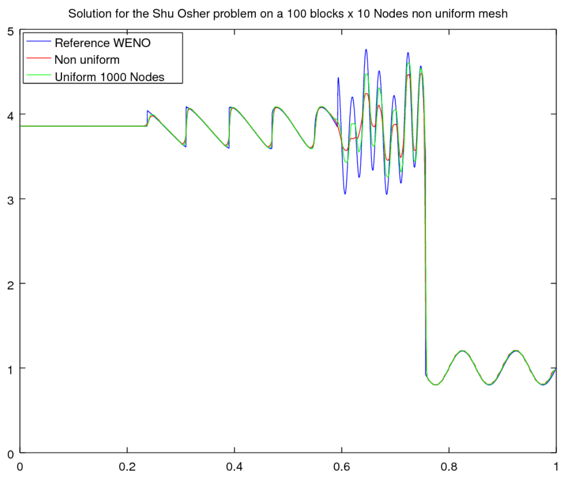
In this case we have almost the same observation as in the previous example. We can just add that the resolution in the left part of the front wave (the part where there is higher frequency oscillations), the resolution is actually not as good as in the uniform case. Moreover, the less nodes there is in a block, the less accurate is the solution.



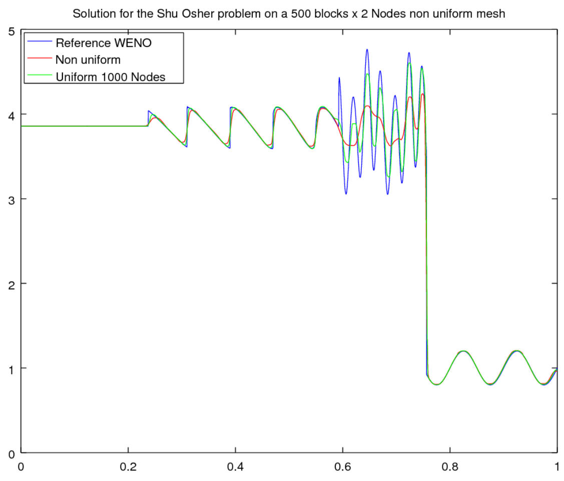
(a) Shu Osher problem with 2 blocks with 500 nodes each



(b) Shu Osher problem with 10 blocks with 100 nodes each



(c) Shu Osher problem with 100 blocks with 10 nodes each



(d) Shu Osher problem with 500 blocks with 2 nodes each

FIGURE 24 – Interacting blast wave on non uniform mesh

8 Concluding remarks

To sum this survey up, we saw through this report the implementation of a second order finite difference method for the compressible Euler's gas equations. We saw how to compute the convergence rate when we do not dispose of analytic solutions. We construct the Legendre-Gauss-Lobatto grid and adapt the method on it. We surprisingly remarked that the method is always second order even on this grid, and two example showed that this non-uniform is less accurate. Finally I would like to stress out the fact that this adaptation on non-uniform mesh is probably not the best update we could do to the method. As we discussed in the Sedov's explosion section, adaptive mesh would have been more efficient on those cases.

To conclude on a personal feeling about this work, I think the most difficult part was to work through the literature and to understand all the differences between finite differences scheme and finite volume scheme. Because in 1D they have really similar form. Then I need a few more time to understand the data structure needed for implementation, especially the nodes numbering (because from one paper to another there is always two king of numbering). But since I have understood the trick with the nodes numbering, the implementation was straight forward.

Finally this internship introduce me to the research in university laboratory. This experiences was really interesting in many ways. First, the topic appealed to lot of my knowledge and using them on such new problem was really satisfying. I finally understood what a weak solution of a partial differential equation is. Second, this experience tough me to work in autonomy, to find solution by myself and to read research article.

Joke :
Have you already wondered why, in deriving exp, the function remains the same ?
Because she saw what Euler did to the governing equations of the fluids dynamics and decided to remain quiet.

Références

[1] Wikipedia. Euler equations (fluid dynamics) — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Euler_equations_(fluid_dynamics)&oldid=784465597, 2017. [Online ; accessed 26-June-2017].

[2] Michael Zingale. Notes on the euler equations. *hydro by example*, 2013.

[3] Wikipedia. Shock wave — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Shock_wave&oldid=786698716, 2017. [Online ; accessed 26-June-2017].

[4] Kurganov Alexander Russo Giovanni Coco Armando, Chertock Alina. A second-order finite-difference method for compressible fluids in domains with moving boundaries. 2017.

[5] Xinghui Zhong. Strong stability-preserving (ssp) high-order time discretization methods, Sep 2009.

[6] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2) :439 – 471, 1988.

[7] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43(1) :89–112, 2001.

[8] *FLASH User's Guide*.

[9] Jun Zhu and Jianxian Qiu. A new fifth order finite difference weno scheme for solving hyperbolic conservation laws. *Journal of Computational Physics*, 318 :110–121, 2016.

[10] Gregor J Gassner. A kinetic energy preserving nodal discontinuous galerkin spectral element method. *International Journal for Numerical Methods in Fluids*, 76(1) :28–50, 2014.

[11] The riemann (or sod) shock-tube problem. http://www.phys.lsu.edu/~tohline/PHYS7412/sod.html.

[12] The riemann (or sod) shock-tube problem. https://fr.mathworks.com/matlabcentral/fileexchange/48734-riemannexact-p1-rho1-u1-p4-rho4-u4-tol-.

[13] A. Baeza, P. Mulet, and D. Zorío. High order boundary extrapolation technique for finite difference methods on complex domains with cartesian meshes. *Journal of Scientific Computing*, 66(2) :761–791, Feb 2016.

[14] Sigal Gottlieb, Julia S Mullen, and Steven J Ruuth. A fifth order flux implicit weno method. *Journal of Scientific Computing*, 27(1-3) :271–287, 2006.

[15] Shu-osher shock tube problem. http://www.ttctech.com/Samples/shockwave/shockwave.htm.

[16] Wikipedia. Halley's method — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Halley%27s_method&oldid=787009965, 2017. [Online ; accessed 25-July-2017].

A Grid of internship proceedings

Grid of internship proceedings	
Week	Description of the undertaken work
1.	First week of work, I met my supervisor, Pr. Gassner, he got me a brief introduction to my internship topic. Then we define the schedule of my work, I got in touch of my colleague and beginning to search and read paper about CFD.
2.	This week was totally devoted to reading papers particularly [4, 13, 7, 6].
3.	During ths week, I created the first program. I familiarized with the data structure and implemented the *minmod* function.
4.	A very productive week, I finished the implementation for regular mesh, I debugged my code and implement the WENO-5 method to get reference solutions.
5.	This week I checked the converge rate of the method and began to generate all the example.
6.	I wrote a first version of the report to present it to Pr. Gassner and we defined the results we wanted to test on non uniform grids.
7.	I read literature about the Legendre-Gauss-Lobatto nodes structure and implemented a function to generate the mesh.
8.	I adapted the method for this non-uniform grid and debugged the code.
9.	Again, I did the error and convergence rate analysis.
10.	I generate all the example for the non-uniform grid case.
11.	I began the final version of the report and present my results to my supervisor.
12.	I finalized the report and said goodbye.

B WENO method

As discussed before, to get references solution I implemented a WENO method. This method is based on finite volume idea (but in 1D this is equivalent to a finite difference method).
We follow the presentation of [14]

1. We use the same flux splitting procedure :

$$f^+(u) = \frac{1}{2}\left(f(u) + au\right) \qquad f^-(u) = \frac{1}{2}\left(f(u) - au\right)$$

2. We calculate the numerical fluxes : $\hat{f}^+_{j+1/2}$ and $\hat{f}^-_{j+1/2}$ with the following procedure (numerical fluxes $\hat{f}^+_{j-1/2}$ and $\hat{f}^-_{j-1/2}$ can be computed with the same formulas with indexes translating by 1).
3. We begin by calculating the smoothness measurements to determine if a shock lies within the stencil :

$$\begin{aligned}
IS_0^+ &= \frac{13}{12}\left(f^+_{j-2} - 2f^+_{j-1} + f^+_j\right)^2 + \frac{1}{4}\left(f^+_{j-2} - 4f^+_{j-1} + 3f^+_j\right)^2 \\
IS_1^+ &= \frac{13}{12}\left(f^+_{j-1} - 2f^+_j + f^+_{j+1}\right)^2 + \frac{1}{4}\left(f^+_{j-1} - f^+_{j+1}\right)^2 \\
IS_0^+ &= \frac{13}{12}\left(f^+_j - 2f^+_{j+1} + f^+_{j+2}\right)^2 + \frac{1}{4}\left(3f^+_j - 4f^+_{j+1} + 3f^+_{j+2}\right)^2
\end{aligned}$$

and

$$\begin{aligned}
IS_0^- &= \frac{13}{12}\left(f^-_{j+1} - 2f^-_{j+2} + f^-_{j+3}\right)^2 + \frac{1}{4}\left(3f^-_{j+1} - 4f^-_{j+2} + f^-_{j+3}\right)^2 \\
IS_0^- &= \frac{13}{12}\left(f^-_j - 2f^-_{j+1} + f^-_{j+2}\right)^2 + \frac{1}{4}\left(f^-_j - f^-_{j+2}\right)^2 \\
IS_0^- &= \frac{13}{12}\left(f^-_{j-1} - 2f^-_j + f^-_{j+1}\right)^2 + \frac{1}{4}\left(f^-_{j-1} - 4f^-_j + 3f^-_{j+1}\right)^2
\end{aligned}$$

4. Next we use the smoothness indicators to calculate the stencil weights :

$$\alpha_0^\pm = \left(\frac{1}{\epsilon + IS_0^\pm}\right)^2, \quad \alpha_1^\pm = \left(\frac{1}{\epsilon + IS_1^\pm}\right)^2, \quad \alpha_2^\pm = \left(\frac{1}{\epsilon + IS_2^\pm}\right)^2$$

and

$$\omega_0^\pm = \frac{\alpha_0^\pm}{\alpha_0^\pm + \alpha_1^\pm + \alpha_2^\pm}, \quad \omega_1^\pm = \frac{\alpha_1^\pm}{\alpha_0^\pm + \alpha_1^\pm + \alpha_2^\pm}, \quad \omega_2^\pm = \frac{\alpha_2^\pm}{\alpha_0^\pm + \alpha_1^\pm + \alpha_2^\pm}$$

5. Finally, the numerical fluxes are :

$$\begin{aligned}
\hat{f}^+_{j+1/2} =& \omega_0^+\left(\frac{2}{6}f^+_{j-2} - \frac{7}{6}f^+_{j-1} + \frac{11}{6}f^+_j\right) + \omega_1^+\left(-\frac{1}{6}f^+_{j-1} + \frac{5}{6}f^+_j + \frac{2}{6}f^+_{j+1}\right) \\
&+ \omega_2^+\left(\frac{2}{6}f^+_j + \frac{5}{6}f^+_{j+1} - \frac{1}{6}f^+_{j+2}\right)
\end{aligned}$$

and

$$\hat{f}^{-}_{j+1/2} = \omega_2^{-}\left(-\frac{1}{6}f^{-}_{j-1} + \frac{5}{6}f^{-}_{j} + \frac{2}{6}f^{-}_{j+1}\right) + \omega_1^{-}\left(\frac{2}{6}f^{-}_{j} + \frac{5}{6}f^{-}_{j+1} + -\frac{1}{6}f^{-}_{j++2}\right) + \omega_0^{-}\left(\frac{11}{6}f^{-}_{j+1} - \frac{7}{6}f^{-}_{j+2} + \frac{2}{6}f^{-}_{j+3}\right)$$

6. Hence we get the following approximation of the flux :

$$f^{+}(u)_x \approx \frac{1}{\Delta x}\left(f^{+}_{j+1/2} - f^{+}_{j-1/2}\right)$$

and

$$f^{-}(u)_x \approx \frac{1}{\Delta x}\left(f^{-}_{j+1/2} - f^{-}_{j-1/2}\right)$$

And the WENO approximation :

$$L(u) = -\frac{1}{\Delta x}\left(f^{+}_{j+1/2} - f^{-}_{j+1/2} + f^{-}_{j-1/2} - f^{+}_{j-1/2}\right) \approx -f(u)_x$$

This method will need three ghost point on each side in order to compute all the stencil. This method is of order 5, as we can verify it on the figure 25 where I used the manufactured solution of case one.

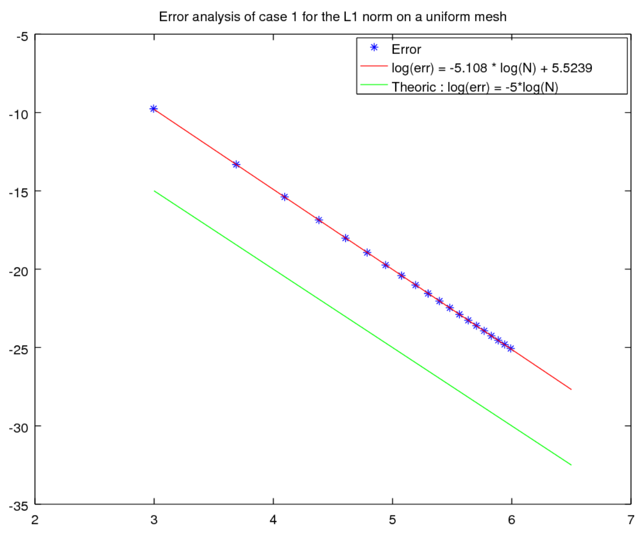


FIGURE 25 – Convergence rate for the WENO method for the L^1 norm

List of Figures

List of Codes