# Numerical Relativity 2023−2024

Alessia Maria Castoldi - 859109

## Homework 1

## 1 Advection Equation

I studied the advection equation in 1D $\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$ using a domain $x \in [0, 10]$ with periodic boundary conditions and an initial Gaussian profile $u(x, t = 0) = e^{-(x-x_0)^2}$ with $x_0 = 5$, terminating the simulation at a time equal to 20.

I started by using the FTCS method (Forward in Time Centered in Space), which is first order in time and second order in space. By changing the number of points of the grid J and the Courant factor $c_f = \frac{dt}{dx}$, I obtained the results shown in Figure 1.1.

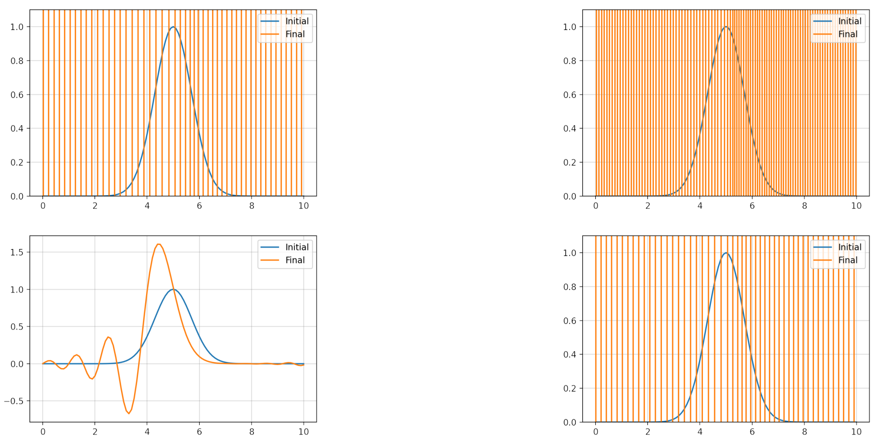

Figure 1.1: Initial and final steps of the simulation of the advection equation using FTCS method. The two top panels show the result using respectively $J = 101$ and $J = 201$ points for the grid and a Courant factor of $c_f = 0.5$. The two bottom panels show the result using respectively $c_f = 0.2$ and $c_f = 0.8$ with $J = 101$.

From the top two panels, I can see that changing the number of points of the grid does not result in a disappearance of the oscillations. From the bottom two panels, it is visible that, when decreasing the Courant factor, the oscillations become less dominant but are still present. The appearance of these oscillations is consistent with the result of the Von Neumann stability analysis, which states that FTCS is not a stable method, so a small perturbation will lead to a exponentially growing error.

I then proceeded by evolving the Advection equation using the Lax-Friedrichs method, obtaining the following results:

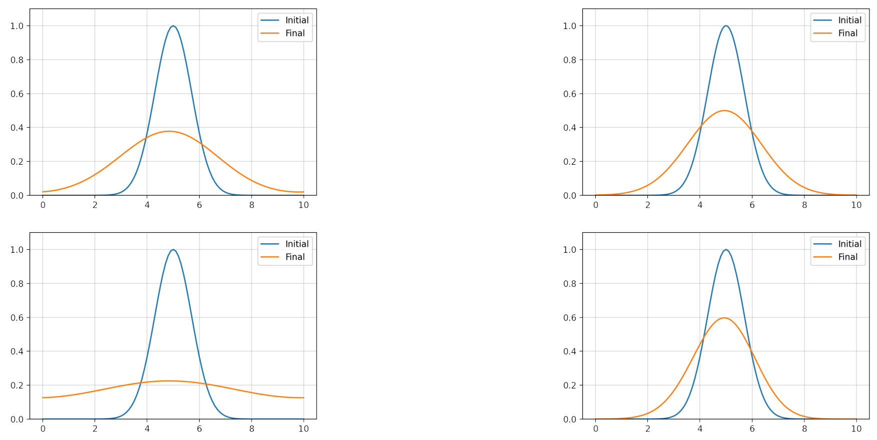

Figure 1.2: Initial and final steps of the simulation of the advection equation using Lax-Friedrichs method. The two top panels show the result using respectively $J = 101$ and $J = 201$ points for the grid and a Courant factor of $c_f = 0.5$. The two bottom panels show the result using respectively $c_f = 0.2$ and $c_f = 0.8$ with $J = 101$.

From the two top panels of Figure 1.2 I can see that, by increasing the grid points, the final result becomes more similar to the expected one, consistent with the fact that the error is proportional to the spacing of the grid $\Delta x$ (error $\xrightarrow[\Delta x \to 0]{} 0$). From the bottom panels I can notice that also an increase in the $c_f$ results in a better outcome. The fact that the final step is less peaked than the initial one is consistent with the fact that, when applying the Lax-Friedrichs scheme to the advection equation, a dissipative term $(\frac{\partial^2 u}{\partial x^2})$ is being added. This term also dumps unwanted oscillations, thus making the scheme stable.

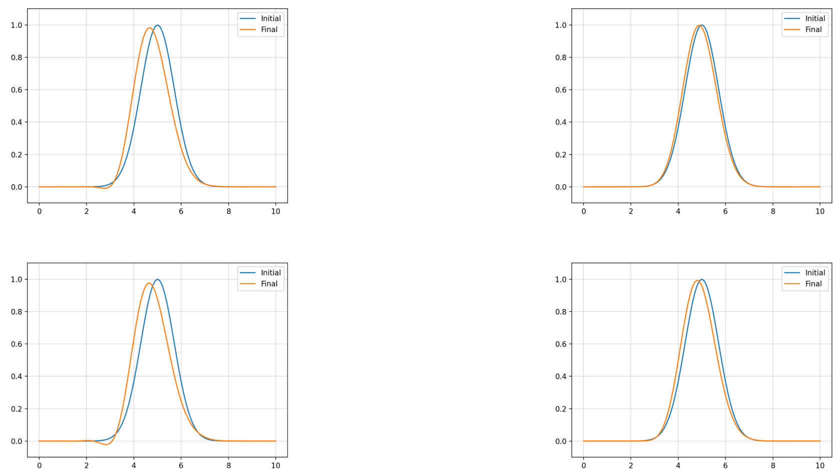

Figure 1.3: Initial and final steps of the simulation of the advection equation using the Leapfrog method. The two top panels show the result using respectively $J = 101$ and $J = 201$ points for the grid and a Courant factor of $c_f = 0.5$. The two bottom panels show the result using respectively $c_f = 0.2$ and $c_f = 0.8$ with $J = 101$.

Proceeding, I evolved the equation using the Leapfrog scheme, obtaining the outcomes visible in Figure 1.3. As already said for the previous scheme, also this one results in a better outcome when increasing the number of grid points and when using a higher Courant factor. In particular, this method gives the best results compared to the previous ones, with the final step of the simulation almost coinciding with the attended one. Indeed, Leapfrog method is second order in time, so the error is proportional to $\Delta x^2$ and the accuracy grows faster with a lower grid spacing compared to a first order method.

Finally, I ran the simulation using the Lax-Wendroff method:

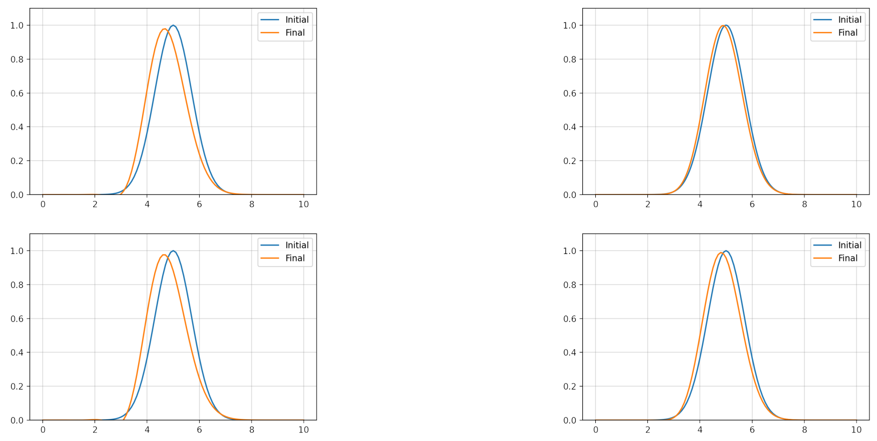

Figure 1.4: Initial and final steps of the simulation of the advection equation using Lax-Wendroff method. The two top panels show the result using respectively $J = 101$ and $J = 201$ points for the grid and a Courant factor of $c_f = 0.5$. The two bottom panels show the result using respectively $c_f = 0.2$ and $c_f = 0.8$ with $J = 101$.

The two top panels visible in Figure 1.4 are consistent with what stated for the last two methods, so that by increasing the grid points, the result approaches the expected one, consistent with the error being proportional to $\Delta x^2$. Additionally, using the Lax-Wendroff scheme with a higher $c_f$ results in a better outcome. When applying this scheme to the advection equation, both a dispersive term $(\frac{\partial^3 u}{\partial x^3})$ and a dissipative one $(\frac{\partial^4 u}{\partial x^4})$ are added. The latter is however subdominant to the dispersive term, which makes some oscillations appear due to the fact that not all parts of the wave are moving at the same speed.

A useful tool to compare the accuracy and stability of the different methods is the $l_2$ norm. Figure 1.5 shows the evolution of the $l_2$ norm of the different methods used to run the simulations.

The $l_2$ norm of the FTCS method diverges to infinity, which is consistent with the fact that this method is unstable. For the Lax-Friedrichs method the $l_2$ norm has a decreasing fashion with respect to time, which is consistent with the presence of a dissipative term, which prevents error growth, but can introduce an excessive smoothing, making the

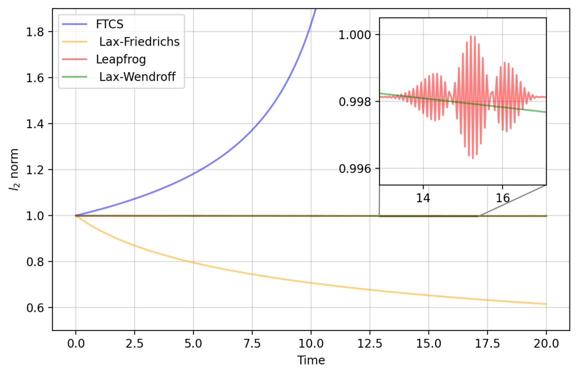

Figure 1.5: Comparison of the evolution of the $l_2$ norm for the different methods used when considering J=101 and $c_f = 0.5$.

scheme less accurate. At last, both Leapfrog and Lax-Wendroff present a $l_2$ norm which remains approximately constant on large scales, and presents some slight oscillations or decreasing trend only on smaller scales, highlighting the two method's stability.

In conclusion, the Lax-Wendroff scheme presents the best outcome in terms of stability and accuracy. The Leapfrog scheme is stable, but it introduces small oscillations. The Lax-Friedrichs scheme presents an excessive dissipation even if it is stable, while the FTCS scheme is inherently unstable.

## 2 Step Function

I studied the advection equation in 1D using as initial data a step function $u(x, t = 0) = 1$ for $x \in [4, 6]$ and $u(x, t = 0) = 0$ otherwise, with periodic boundary conditions and terminating the simulation at a time equal to 20.

I started by using the Lax-Friedrichs scheme and, changing the number of grid points J and the Courant factor $c_f$, I obtained the following results:

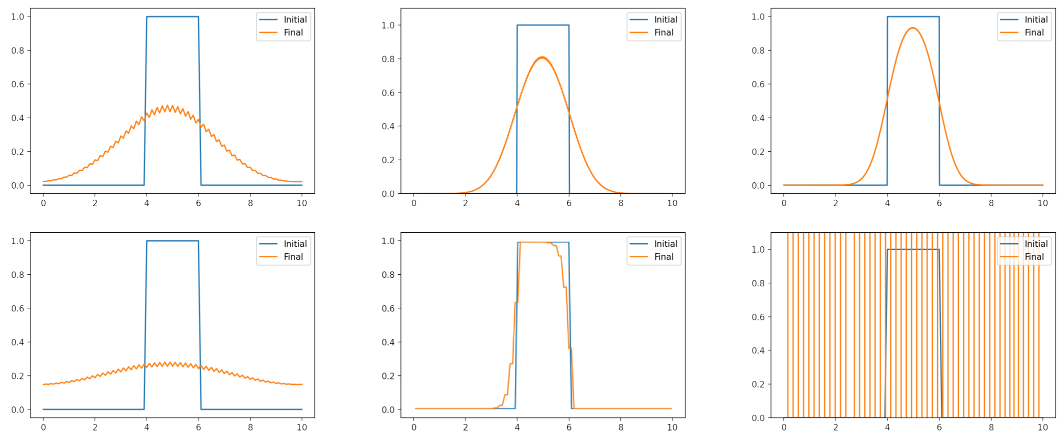

Figure 2.1: Initial and final steps of the simulation of the advection equation using Lax-Friedrichs method. The three top panels show the result using respectively $J = 101$, $J = 501$ and $J = 1001$ points for the grid and a Courant factor of $c_f = 0.5$. The three bottom panels show the result using respectively $c_f = 0.2$, $c_f = 1$ and $c_f = 1.2$ with $J = 101$.

From the top panels of Figure 2.1 it is visible that, by increasing the number of grid points J, the final step of our simulation becomes more and more similar to the expected one. Indeed, the dissipative term introduced when using the Lax-Friedrichs method becomes less important when increasing the resolution of the grid, resulting in a more preserved shape of the step function. From the bottom panels it is confirmed that, when the CFL condition ($c_f \leq 1$) is satisfied, the Lax-Friedrichs method is stable. On the opposite side, when the Courant factor is greater than 1, the scheme becomes unstable and introduces major oscillation in the outcome.

This is also confirmed when looking at the evolution of the $l_2$ norm (Figure 2.2). Indeed, when the Courant factor is bigger than 1, the $l_2$ norm diverges to infinity and the method is unstable. Instead, when increasing the grid points J, the $l_2$ norm decreases less steeply and the solution better resembles the expected one.

I then proceeded by using the Lax-Wendroff scheme, obtaining the results visible in Figure 2.3. This scheme introduces both a dissipative term and a dispersive one, with the latter being responsible for the appearance of the oscillations visible in the top panels. Indeed, due to Godunov theorem, Lax-Wendroff cannot be a linear monotone method since it is second order accurate. In fact, even when increasing the number of grid points, the presence of this oscillations is still visible, although their amplitude decreases. From

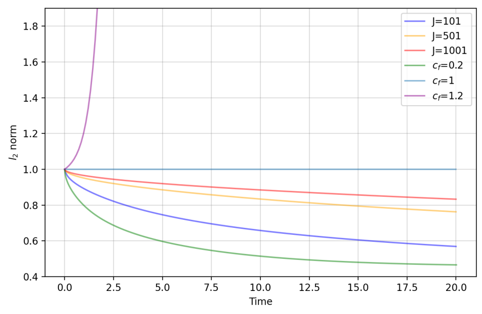

Figure 2.2: Comparison of the evolution of the $l_2$ norm using Lax-Friedrichs when considering different values of J and $c_f$.

the bottom panels, it is observable that a Courant factor lower than 1 results in the Lax-Wendroff method being stable, with the oscillations, as said previously, being only due to the dispersive term. In particular, when the Courant factor is close to 1, the oscillations almost disappear and the solution of the simulation better resembles the correct one. As already said for the previous scheme, when violating the CFL condition, the method is no longer stable, leading to significant oscillations.

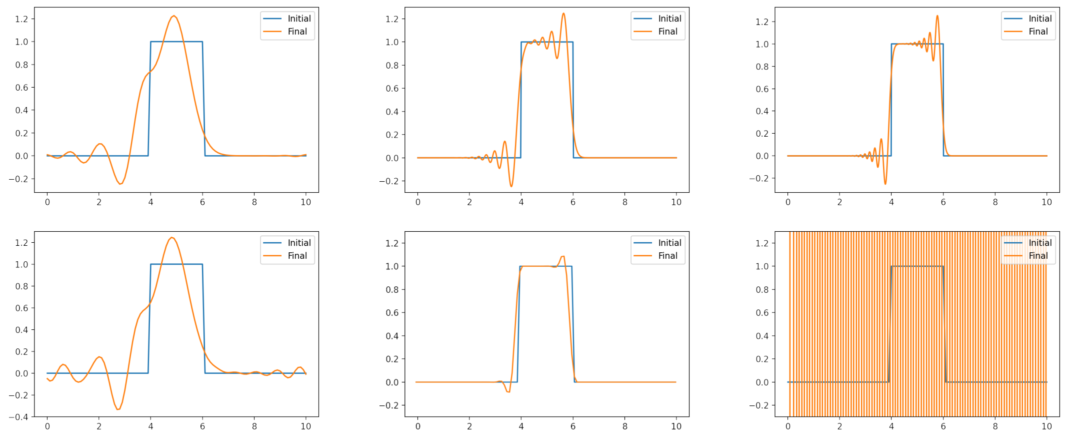

Figure 2.3: Initial and final steps of the simulation of the advection equation using Lax-Wendroff method. The three top panels show the result using respectively $J = 101$, $J = 501$ and $J = 1001$ points for the grid and a Courant factor of $c_f = 0.5$. The three bottom panels show the result using respectively $c_f = 0.2$, $c_f = 1$ and $c_f = 1.2$ with $J = 101$.

Also when using the Lax-Wendroff scheme, the evolution of the $l_2$ norm (Figure 2.4) confirms what previously said: when $c_f > 1$ the scheme is unstable, reflecting in a rapid grow of the $l_2$ norm. Conversely, as J increases, the $l_2$ norm tends to decrease more slowly, indicating a more accurate solution.

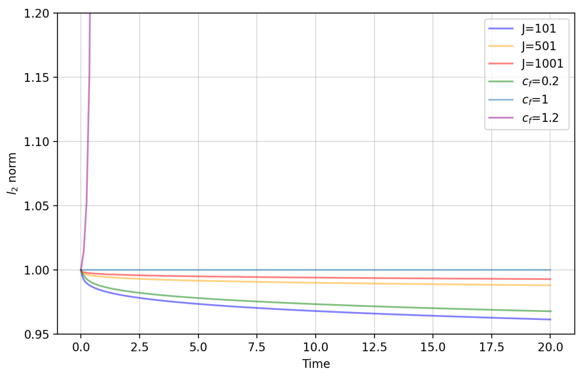

Figure 2.4: Comparison of the evolution of the $l_2$ norm using Lax-Wendroff when considering different values of J and $c_f$.

## 3 Burgers' Equation

I used the Upwind scheme to evolve the Burger's equation, both in its non-flux-conservative (NFC) formulation and in its flux-conservative (FC) formulation:

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} = 0 \tag{NFC}$$

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \quad \textit{with} \quad f(u) = \frac{1}{2}u^2 \tag{FC}$$

I started by evolving the Burgers' equation in its FC formulation, considering a domain of $x \in [0, 10]$, a final time equal to 0.5 and, as initial condition, a Gaussian profile $u(x, t = 0) = 10\,e^{-(x-x_0)^2}$ with $x_0 = 5$ . I repeated the simulation for different numbers of grid points, namely $J = 101$, $J = 201$ and $J = 501$ with a Courant factor of $c_f = 0.5$, obtaining the following results:

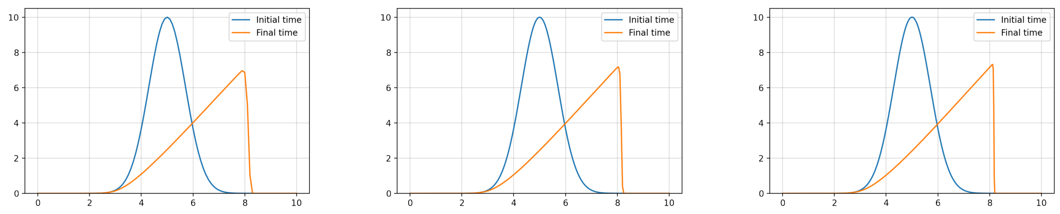

Figure 3.1: Initial and final steps of the simulation of the FC Burger's equation using respectively $J = 101$, $J = 201$ and $J = 501$ points for the grid and $c_f = 0.5$.

When using the FC formulation of the Burgers' equation, there is, as expected, the formation of a rarefaction wave on the left and a shock wave on the right, with the position of the latter matching exactly the theoretical prediction. It is also noticeable that, as the resolution is increased by adding more grid points, the final solution becomes more and more similar to the exact one.

I then proceeded to repeat the simulation using the NFC formulation:

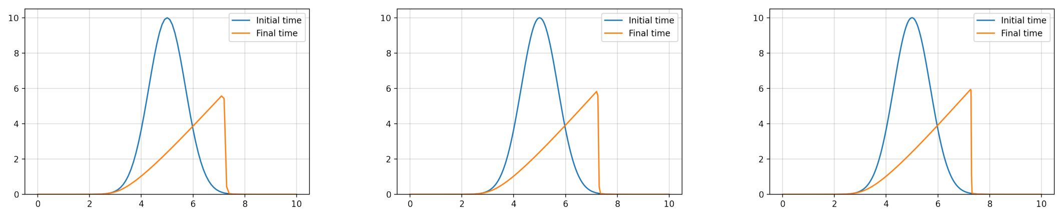

Figure 3.2: Initial and final steps of the simulation of the NFC Burger's equation using respectively $J = 101$, $J = 201$ and $J = 501$ points for the grid and $c_f = 0.5$.

It is visible from Figure 3.2 that, when using the NFC formulation of Burgers' equation, the final position of the shock wave is incorrect, even when the resolution is increased. This result is consistent with what is stated in the Hou-Le Flock theorem: "Non conservative numerical methods do not converge to the correct solution if a shock wave is present".

In conclusion, from the simulation, I can confirm that the two formulations, while being mathematically equivalent, are not numerically equal. The NFC formulation results in an incorrect final position of the shock wave, while the FC formulation returns the exact value.

Homework 2

## 1 Sod Shock Tube Problem

The Sod shock tube problem consists in a Riemann problem with the initial conditions for the rest-mass density $\rho$, the pressure $P$ and the velocity along the x axis $v_x$ represented in Figure 1.1. In particular, there is a discontinuity in the initial conditions of density and pressure, while the velocity of the gas particles is set to zero. To solve the Euler equations describing this problem, I used the Einstein Toolkit with HLLE as an approximate Riemann solver, while considering for the equation of state a polytrope with index $\gamma = 5/3$, corresponding to an ideal monoatomic gas. The final time of the simulation was set to 0.4 while using a domain $x \in [-0.5, 0.5]$.

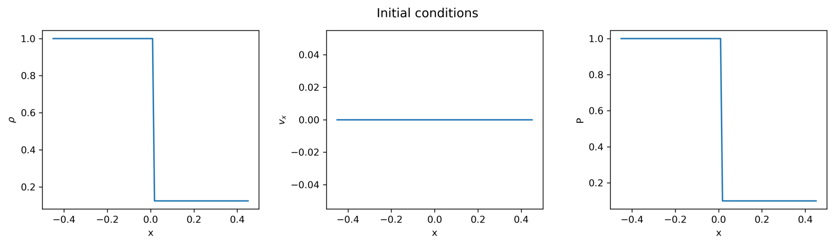

Figure 1.1: Initial condition of the Sod shock tube problem for $\rho$, $P$ and $v_x$. The velocity along the x axis is constant and equal to zero, while both the rest-mass density and the pressure present a shock wave at $x = 0.0$.

Through the simulations, I investigated how the resolution affects the final result. To obtain different resolutions, I changed the value of the grid spacing $dx$, setting it to $dx = 0.025$, 0.0025 and 0.001. This corresponds to having different numbers of grid points, respectively 40, 400 and 1000, computed as the ratio between the length of the domain and $dx$. I compared the results obtained through the simulations with the exact solutions for all the three variables.

As visible from Figure 1.2, as the resolution increases, the numerical solutions become more and more similar to the exact one for all the variables analyzed. Focusing on the top three panels describing the rest-mass density, it is evident on the left side the presence of the rarefaction wave, corresponding to the wave travelling with a negative velocity. Further, one can see the presence of a contact discontinuity in the middle and of a shock wave on the right side, each represented by a “jump” in the graph. In the middle panels it can be seen that, for pressure, only the rarefaction wave and the shock wave are present. The bottom panels show the presence of the rarefaction wave and of the shock wave, while the velocity stays null at the extremes.

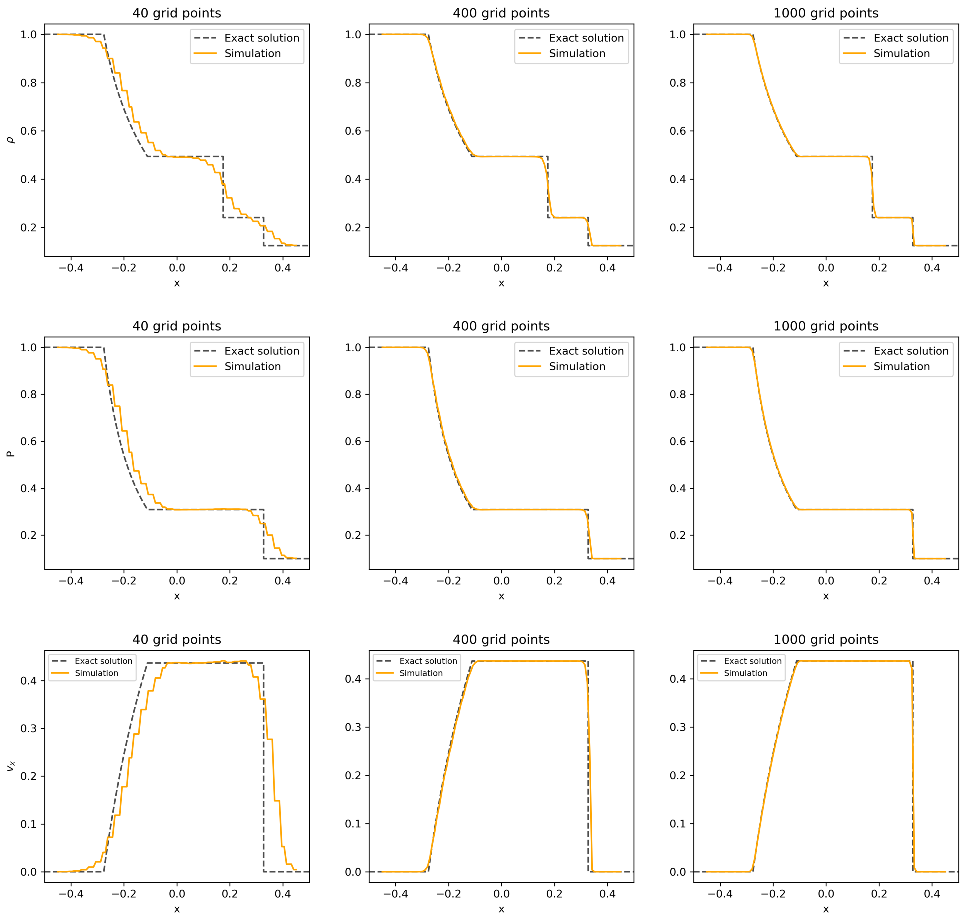

Figure 1.2: Comparison of the exact solution (black dashed line) and the result obtained through the simulation (orange solid line) using three different resolutions (40, 400 and 1000 grid points) for respectively the rest-mass density $\rho$, the pressure $P$ and the velocity along the x axis $v_x$.

## 2 TOV Evolution

The solution of the Tolman–Oppenheimer–Volkoff (TOV) equation describes a static and spherically symmetric Neutron Star. Using the Einstein Toolkit, I simulated the evolution of the star. In particular, in my work, I studied how the result varies when changing the grid setup, specifically the number of refinement levels and the resolution. The resolution of each sub-grid was set such that it is double the one of the previous grid, so that the cell size is half the previous one. All the measurement in the simulations are given in solar masses, with the conversion factor given by $1M_{\odot} = 1.5km$. The domain of the simulation was set to $24M_{\odot}$, corresponding to $36km$. I ran three different simulations: in the first one i considered two grids with extension respectively $r_1 = 24$ and $r_2 = 12$ and a cell size of $dx_1 = 2.0$; in the second one i considered $r_1 = 24$, $r_2 = 12$ and $r_3 = 10$ and $dx_1 = 2.0$; in the last one i considered $r_1 = 24$, $r_2 = 12$ and $r_3 = 10$ and $dx_1 = 1.5$. All the simulations were run up to a final time $t = 400M_{\odot}$, corresponding to $t = 1.97ms$.

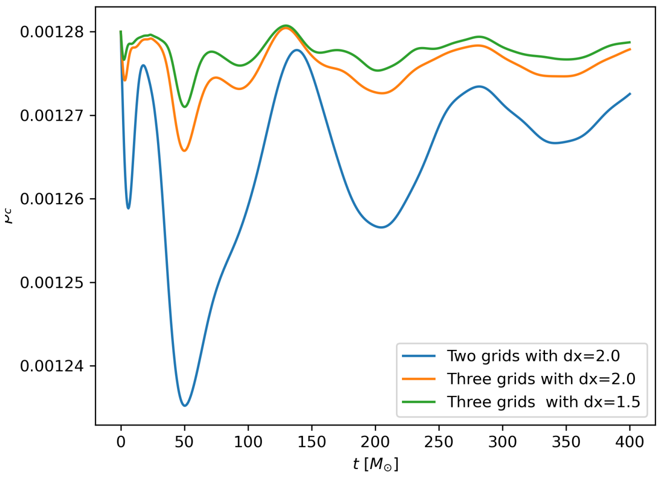

Figure 2.1: Maximum rest-mass density as a function of time for the three different simulations. The blue line correspond to a simulation with two grids and a resolution $dx = 2.0$, the orange line to a simulation with three grids and $dx = 2.0$ and the green line to a simulation with three grids and $dx = 1.5$.

Figure 2.1 shows the evolution of the maximum of the rest-mass density for the three simulations. The oscillations visible in all three plots are only due to numerical perturbations: in fact, a spherical solution was imported into a Cartesian grid with a finite resolution. This approximation results in some cells on the surface of the star being more dense than others. These overdensities act as artificial perturbations, resulting in small variations in pressure with respect to the exact static solution. The first simulation (blue line) is the one that presents the most pronounced variations in the maximum of the rest-mass density. When adding a grid with a higher resolution (orange line) the oscillations decrease drastically, accordingly the the sphere being better approximated. Moreover, when the initial resolution is increased (green line) the oscillations reduce even more, which is consistent with the perturbation only being numerical and not physical. Indeed, the presence of oscillations due to a physical perturbation do not depend on the grid setup of the simulation and should be visible in all cases.

A visual representation of the increase in resolution can be seen in Figure 2.2, where is evident the improvement in the approximation of the spherical solution.

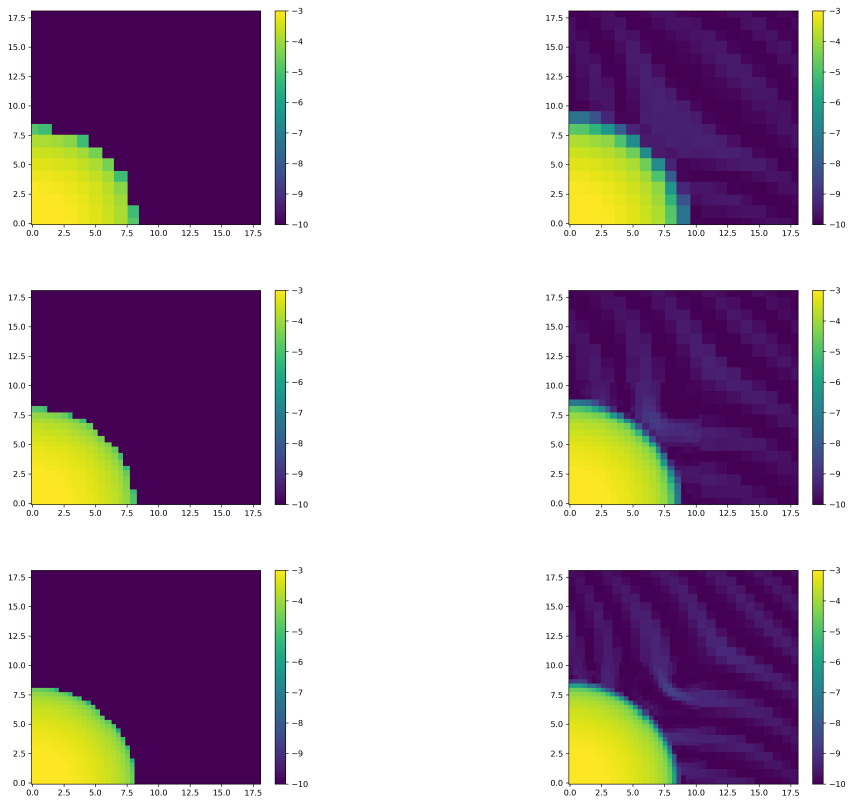

Figure 2.2: Initial and final steps of the simulation of the Advection equation. The two top panels show the result using two grids and a resolution $dx_1 = 2.0$. The two middle panels correspond to a simulation with three grids and $dx_1 = 2.0$. The two bottom panels correspond to a simulation with three grids and $dx_1 = 1.5$.

# 3 Einstein Toolkit Parameters

## 3.1 `MoL::ODE_Method = "rk4"`

The parameter `MoL::ODE_Method="rk4"` refers to the numerical method used to solve the system of ordinary differential equations (ODEs) obtained after applying the Method of Lines (MoL) to the equations governing the evolving object. In particular, the MoL can solve partial differential equations (PDEs) by separating the time integration from the rest of the evolution scheme, resulting in a system of ODEs that can be solved in time. In this case, the method specified by the parameter to perform the time integration is a $4^{\text{th}}$ order Runge-Kutta method. Starting from known initial conditions, this method estimates the solution at each time step by considering four slopes evaluated at different points within the step. This method is $4^{\text{th}}$ order accurate in time and provides a highly accurate result for the subsequent value of the function in time.

## 3.2 `GRHydro::recon_method = "ppm"`

The parameter `GRHydro::recon_method="ppm"` is found when using the module `GRHydro`, which is a code used to simulate astrophysical phenomena involving general-relativistic hydrodynamics, such as in simulations of neutron stars or black holes. In particular, the parameter `recon_method` controls the reconstruction method used to derive the values of the hydrodynamical variables, such as pressure and density, at the interface of each grid cell. The reconstruction method used in this case is a piecewise parabolic method (PPM), which approximates the function within each cell by a parabola, allowing for a more accurate representation of the function's variation across the cell and a better resolution of features such as shock waves. In particular, PPM exploits the values of the function in the nearby cells to reconstructs a parabolic profile in each cell and to derive the values of the variables at the edge of each cell.

## 3.3 `GRHydro::riemann_solver = "Marquina"`

The parameter `GRHydro::riemann_solver="Marquina"` refers to the type of Riemann solver used in the `GRHydro` module. A Riemann solver is a method used to solve a Riemann problem, which is an initial value problem composed of two constant initial states and a discontinuity. This is done through the computation of fluxes across the interfaces between grid cells. Solving numerically a Riemann problem can be computationally expensive, thus the approximate Riemann solvers, such as Marquina, are introduced. Starting from the eigenvalues of the Jacobian matrix of the flux function, Marquina decomposes the discontinuity at the cell interface into the different types of waves, considering also the direction of propagation of the waves. Subsequently, the flux contributions from the left and right sides of the interface are combined using the absolute wave speeds, thus ensuring the method's stability and accuracy in the presence of shock waves and discontinuities.

### 3.4 BSSN parameters

The following block of parameters

```
ML_BSSN::harmonicN       = 1     # 1+log
ML_BSSN::harmonicF       = 2.0   # 1+log
ML_BSSN::ShiftGammaCoeff = 0.75  # this is 3/4
ML_BSSN::BetaDriver      = 2.66  # common choices are 1/M or 1/2M
```

is related to the space-time evolution method used to evolve Einstein's field equations, in particular in the context of Baumgarte-Shapiro-Shibata-Nakamura (BSSN) formulation.

Starting from the ADM formulation, BSSN introduces new variable to reduce the order of derivatives and to obtain a strongly hyperbolic system, that is suitable for use in numerical simulations. BSSN equations are the following:

$$\partial_t \phi = -\frac{1}{6}\alpha K + \frac{1}{6}\partial_i \beta^i + \beta^i \partial_i \phi$$

$$\partial_t \tilde{\gamma}_{ij} = -2\alpha \tilde{A}_{ij} + \beta^k \partial_k \tilde{\gamma}_{ij} + \tilde{\gamma}_{ik}\partial_j \beta^k + \tilde{\gamma}_{kj}\partial_i \beta^k - \frac{2}{3}\tilde{\gamma}_{ij}\partial_k \beta^k$$

$$\partial_t K = -D^i D_i \alpha + \alpha \left( \tilde{A}_{ij}\tilde{A}^{ij} + \frac{1}{3}K^2 \right) + 4\pi\alpha(E+S) + \beta^i D_i K$$

$$\begin{aligned}\partial_t \tilde{A}_{ij} = {}& e^{-4\phi}\left[ -\left(D_i D_j \alpha\right)^{\mathrm{TF}} + \alpha\left( {}^{(3)}R_{ij}^{\mathrm{TF}} - 8\pi S_{ij}^{\mathrm{TF}} \right)\right] \\ & + \alpha\left( K\tilde{A}_{ij} - 2\tilde{A}_{ik}\tilde{A}^k_j \right) + \beta^k \partial_k \tilde{A}_{ij} + \tilde{A}_{ik}\partial_j \beta^k + \tilde{A}_{kj}\partial_i \beta^k - \frac{2}{3}\tilde{A}_{ij}\partial_k \beta^k\end{aligned}$$

$$\begin{aligned}\partial_t \tilde{\Gamma}^i = {}& -2\tilde{A}^{ij}\partial_j \alpha + 2\alpha\left( \tilde{\Gamma}^i_{jk}\tilde{A}^{kj} - \frac{2}{3}\tilde{\gamma}^{ij}\partial_j K - 8\pi\tilde{\gamma}^{ij}S_j + 6\tilde{A}^{ij}\partial_j \phi \right) \\ & + \beta^j \partial_j \tilde{\Gamma}^i - \tilde{\Gamma}^j \partial_j \beta^i + \frac{2}{3}\tilde{\Gamma}^i \partial_j \beta^j + \frac{1}{3}\tilde{\gamma}^{li}\partial_l \partial_j \beta^j + \tilde{\gamma}^{lj}\partial_j \partial_l \beta^i\end{aligned}$$

Among all the variables, the most important ones for understanding the above block of parameters are listed below:

- $\tilde{\gamma}_{ij}$ is the conformal transformation of the 3-metric $\gamma_{ij}$. It can be expressed as $\tilde{\gamma}_{ij} = e^{-4\phi}\gamma_{ij}$ where $\phi$ is a function of time and space coordinates;
- $\alpha$ is the lapse function and defines how time changes when moving from an hypersurface to the next one ($d\tau = \alpha\, dt$), with $\alpha$ decreasing the more space-time is curved;
- $\beta^i$ is the shift vector, which is a purely spacial vector. When moving along a normal between two hypersurfaces, the space coordinates change by a factor $-\beta\, dt$;
- $K_{ij}$ is the extrinsic curvature, which tells us how the slices pile up on top of each other, so how $\gamma_{ij}$ changes while moving between hypersurfaces;

- $\tilde{\Gamma}^i$ are the conformal connection functions, representing the divergence of the conformal metric $\tilde{\Gamma}^i = \partial_j \tilde{\gamma}^{ij}$.

The first two parameters of the block, `ML_BSSN::harmonicN = 1` and `ML_BSSN::harmonicF = 2.0`, are related to the hyperbolic K-driver slicing conditions, which controls the evolution of the lapse function $\alpha$ and takes the form $(\partial_t - \beta^i \partial_i)\alpha = -f(\alpha)\alpha^2(K - K_0)$, where $K$ is the trace of the extrinsic curvature and $K_0$ is its value at $t = 0$. The values indicated by the parameters refer to the so called "$1 + log$" slicing condition, which, considering $K_0$ and $\beta^i$ to be both equal to zero, can be expressed as $\partial_t \alpha = -f\alpha^n K$. Thus, the first parameter refer to the exponent $n$ while the second parameter refers to the function $f$, obtaining the equation $\partial_t \alpha = -2\alpha K$. Exploiting the fact that $-\alpha K = \partial_t \ln(\sqrt{\gamma})$, we can finally find the relation $\alpha = 1 + \ln(\gamma)$, explaining the name of the slicing condition. This slicing condition is widely used since it offers computational efficiency together with strong singularity avoidance properties.

The last two parameters, namely `ML_BSSN::ShiftGammaCoeff = 0.75` and `ML_BSSN::BetaDriver = 2.66`, are instead related to Gamma-driver shift condition, which manages the evolution of the shift vector $\beta^i$. This condition can be expressed by the two equations $\partial_t \beta^i - \beta^j \partial_j \beta^i = \frac{3}{4} B^i$ and $\partial_t B^i - \beta^j \partial_j B^i = \partial_t \tilde{\Gamma}^i - \beta^j \partial_j \tilde{\Gamma}^i - \eta B^i$, where $B^i$ is the time derivative of $\beta^i$ and $\eta$ is a constant, typically $\frac{1}{M}$ or $\frac{1}{2M}$, being $M$ the gravitational mass of the system. The first parameter is a coefficient associated with the term containing $\tilde{\Gamma}^i$ and it controls how the spatial curvature influences the evolution of the shift vector. The last parameter refers to the coefficient $\eta$, which controls the damping and evolution rate of $\beta^i$. The two values of the parameters specified in the code ensure an evolution of the shift vector that provides both stable and flexible coordinate to best represent the dynamic evolution of the system.

Source codes used to solve the exercises can be found at:
https://github.com/AlessiaCastoldi/Numerical-Relativity