HOMEWORK 2 - Numerical Relativity 2023 - 2024

In this homework I will analyze the SOD shock tube problem, studying the accuracy of the evolution results depending on the resolution in the x-axis. Then, I will consider the evolution of the TOV equations to describe a stable, non rotating and spherically symmetric neutron star. In particular, I will focus on studying how a different resolution, a pressure perturbation or a different grid can affect the results of the evolution. Finally, I will describe in detail the meaning of certain parameters present in the `.par` file used for the evolution of the TOV equations for the neutron star introduced above.

SOD SHOCK TUBE PROBLEM

The SOD shock tube problem involves a system in which the initial condition are defined by a constant velocity - set equal to 0 - and a discontinuity both in the rest mass energy density and the pressure. Therefore, it can be linked to the Riemann problem, and it is usually used to test the accuracy of numerical methods.

In order to evolve the system, I needed to follow the Euler equations, which describe the evolution of a perfect fluid in Newtonian physics. This can be done using the Einstein Toolkit. I then considered a polytropic equation of state: in particular, I used an exponent $K = 5/3$ to recover the equation of state of an ideal gas. Finally, in order to simulate the system evolution, I relied on the HLLE approximate Riemann Solver.

To study the impact of the resolution on the evolution, I modified the value of the dx parameter in the parameter file, in order to consider different samplings and compare them with the exact solution. In particular, I chose for the comparison the following values: $dx = 0.01$ (100 grid points), $dx = 0.0025$ (400 grid points), $dx = 0.001$ (1000 grid points). However, even if the third case resulted to be more accurate with respect to the exact solution, even a lower resolution is able to reach a suitable accuracy. Here, I show the resulting evolution and the comparison between the different scenarios for the density, the pressure and the velocity along the x-direction respectively.

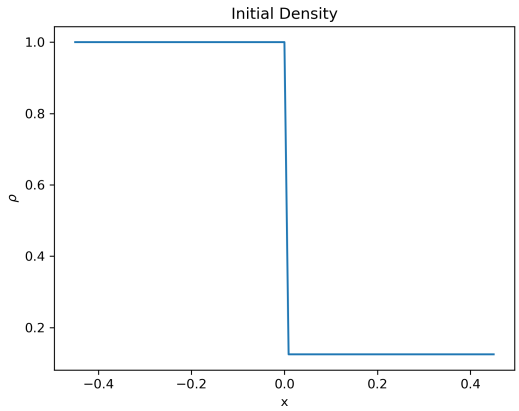

Figure 1: Initial condition for the density.

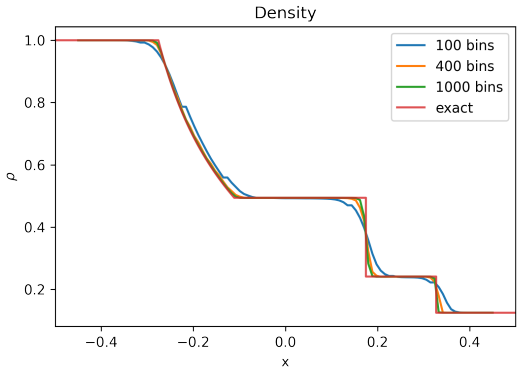

Figure 2: Comparison for the different resolutions of the density evolution.

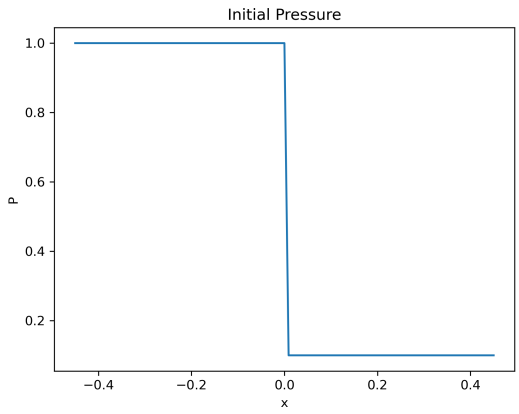

Figure 3: Initial condition for the pressure.

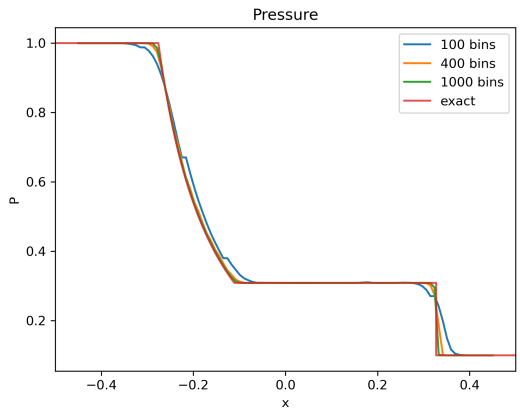

Figure 4: Comparison for the different resolutions of the pressure evolution.

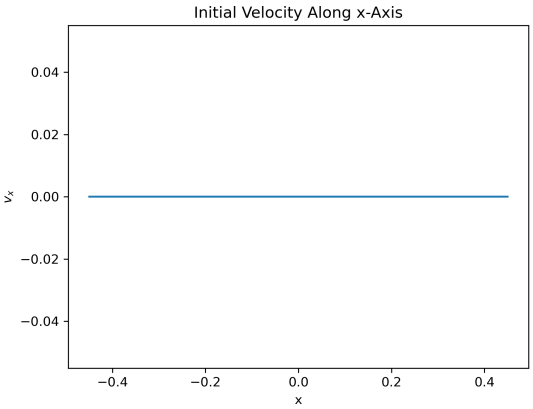

Figure 5: Initial condition for the velocity along the x-axis.

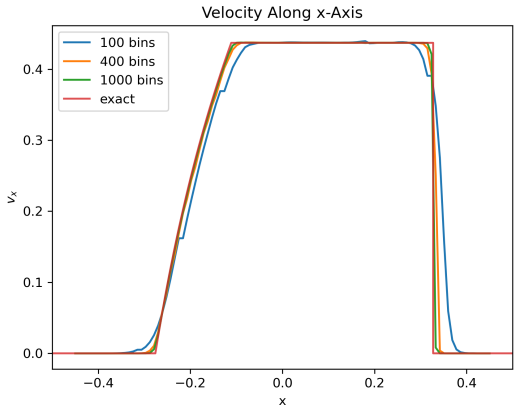

Figure 6: Comparison for the different resolutions of the velocity along the x-axis evolution.

TOV EVOLUTION

Now, I will focus on solving the TOV equations to describe the evolution of a stable, non rotating and spherically symmetric neutron star, through the Einstein Toolkit. Specifically, I will consider a neutron star with an initial central density $\rho_{c,0} = 1.28 \cdot 10^{-3} M_{\odot}$. I will also use a polytrophic equation of state, with $K = 100$ and $\gamma = 2$. In order to optimize the code, I will evolve only one octant of the neutron star, taking advantage of the system symmetry.

I will compare the results of the evolution in different scenarios. Specifically, I will firstly compare different resolutions. Then, I will add a pressure perturbation by changing the parameter K and compare the resulting evolution for different resolutions. Finally, I will consider the impacts of different choices for the grid extension.

1 RESOLUTION

Firstly, I studied how the resolution affects the simulation by changing the size of the cells ΔL - modifying the size in the same way for each direction x, y, z. In particular, for each direction I consider the following values: $\Delta L = 2.0, 1.5, 1.0$. Smaller cells are equivalent to higher resolutions, since a higher number of cells will be required to describe the entire grid, leading to a denser sampling.

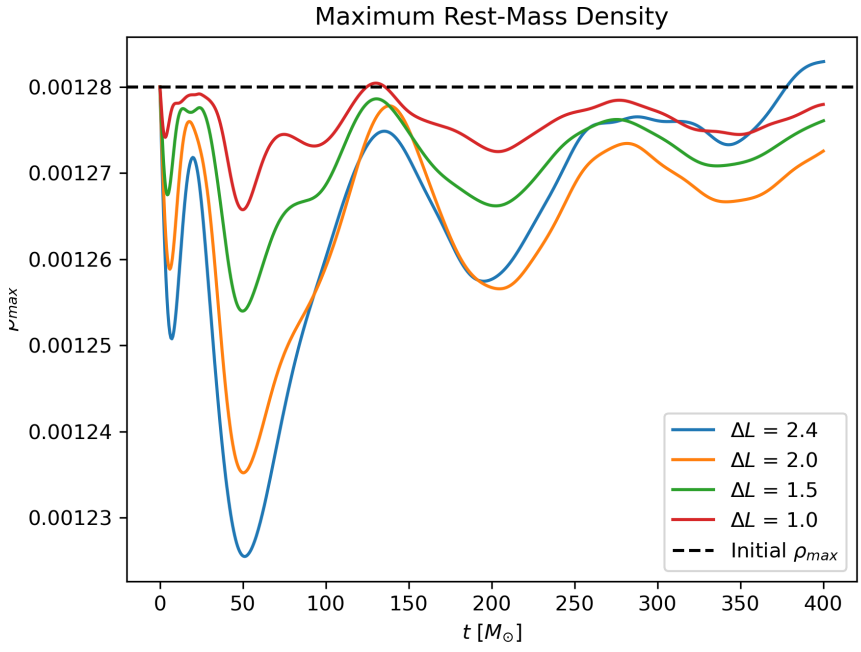

Figure 7: Evolution of ρ_{max} during the simulation for the different resolution. As it can be noticed, the oscillations are lower in the case corresponding to a higher resolution ($\Delta L = 1.0$), and they increase while reaching lower resolutions.

I also show a comparison between the densities at the last step of the evolution for each ΔL. It can be seen here as well that a lower ΔL corresponds to a higher resolution by checking the pixel sizes corresponding to each grid.

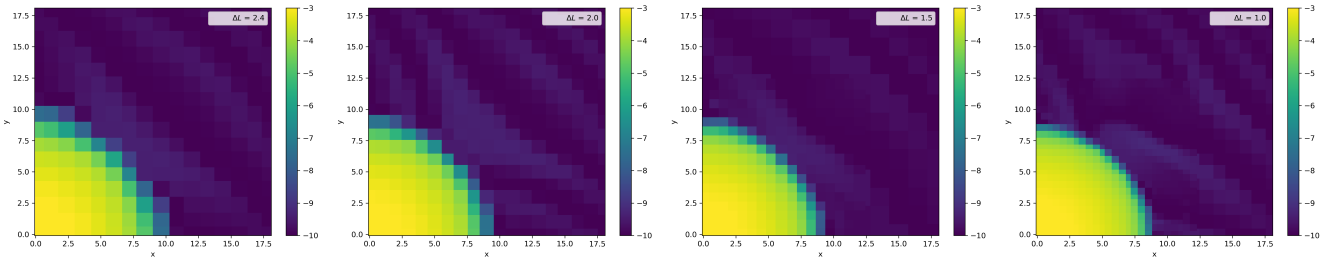

Figure 8: Final density grids resulted from the evolution of the TOV equations for different value of ΔL. As it can be seen from the grids, a smaller cell size is equivalent to a higher resolution due to a denser sampling.

2 GRID EXTENSION AND NUMBER OF GRIDS

Then, I studied the effects of the grid length on the simulation. Specifically, in the original scenario the grid length is $L = 24.0$, which I then changed to $L = 8.0$, $L = 10.0$ and $L = 16.0$.

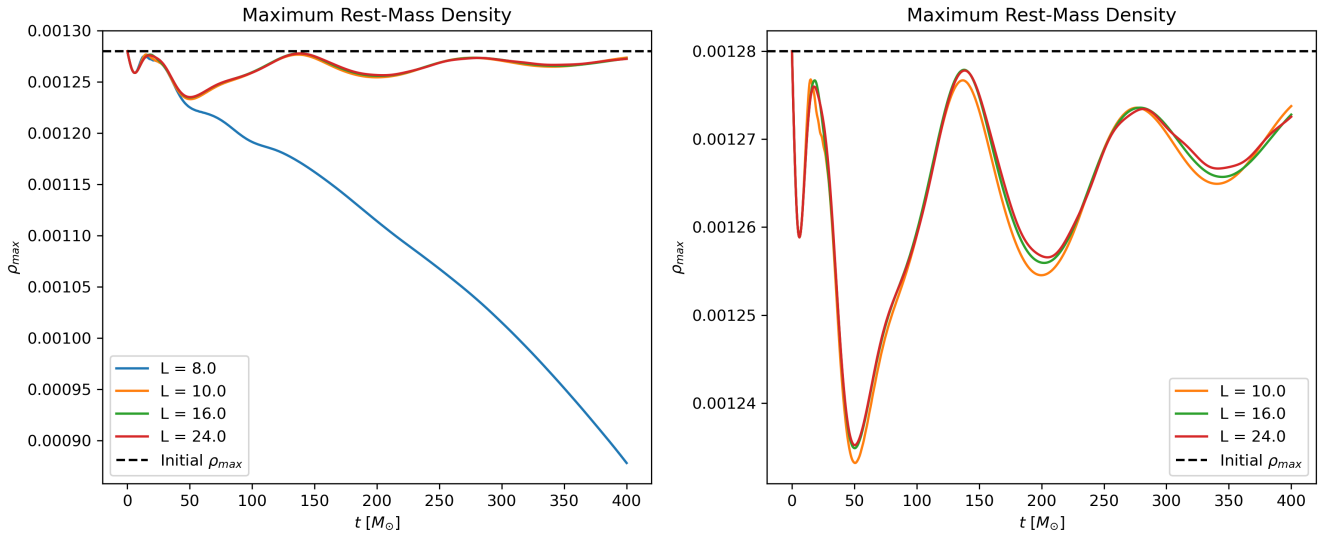

Figure 9: Evolution of ρ_{max} during the simulation for the different grid length. As it can be seen, the oscillations are on the same order for the cases with $L \geq 10$, while in the scenario with $L = 8.0$ ρ_{max} keeps decreasing during the evolution.

As we can see from Figure9, the scenarios with $L \geq 10$ recover the same results, while for $L = 8.0$ the value of ρ_{max} keeps decreasing. Comparing the density grids for the final step of the evolution for each scenario, we can also notice that while for the cases with $L \geq 10$ almost the entirety of the neutron star is present on the grid, in the case with $L = 8.0$ the length is not enough to properly evolve the neutron star.

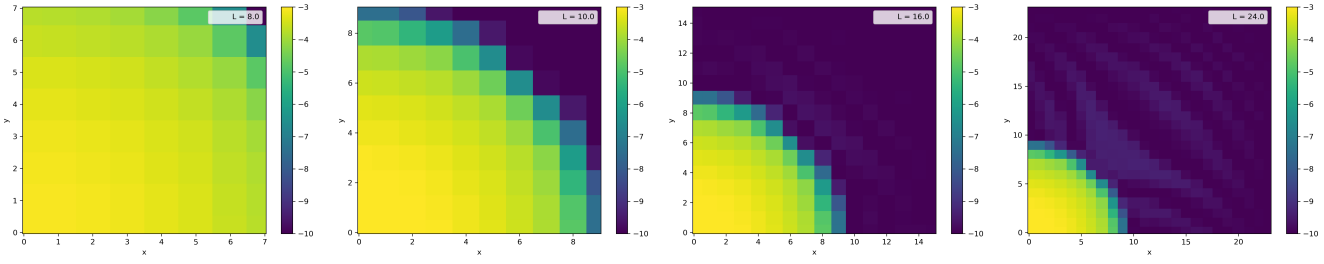

Figure 10: Final density grids resulted from the evolution of the TOV equations for different value of L. As it can be seen from the grids, the resolution is fixed since ΔL is kept constant, while the grid is getting smaller up until the moment in which the neutron star is not inside the grid anymore.

3 PRESSURE PERTURBATION

Here, we consider instead a perturbation in the pressure. As already mentioned, I assumed a polytrophic equation of state, given by:

$$P = K\rho^{\gamma}. \tag{1}$$

Therefore, I added a perturbation in the pressure with respect to the initial scenario with $K = 100$ and $\gamma = 2$ by changing the value of K and leaving γ fixed. Specifically, I compared the original scenario with two different cases, one with $K = 90$ and $K = 105$. As we can already see from equation 1, this means that for a fixed value of ρ, the pressure will be lower or higher with respect to the original case respectively.

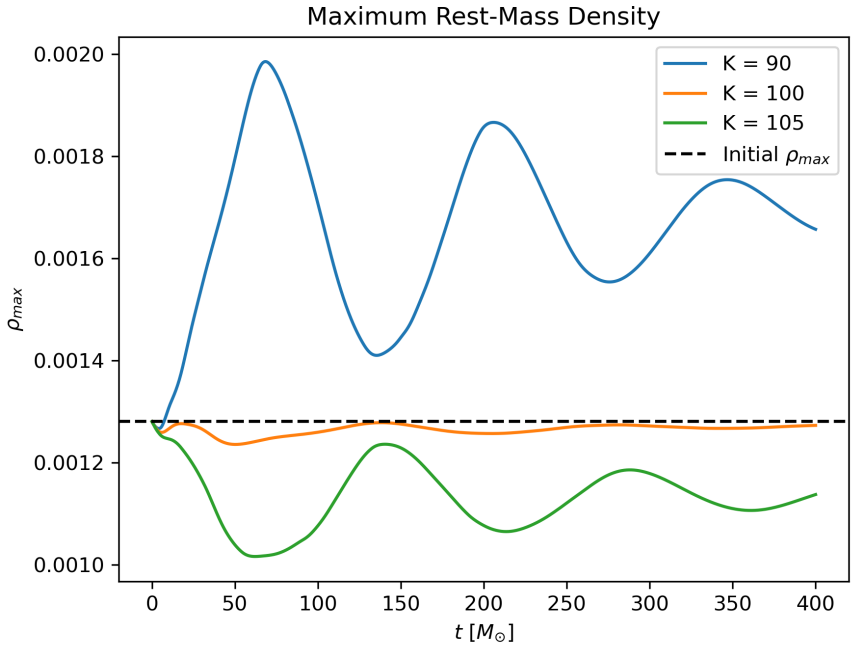

Figure 11: Evolution of ρ_{max} during the simulation for the different pressure scenarios. As it can be seen, the perturbed cases oscillates around different values of ρ_{max} with respect to the unperturbed scenario. Furthermore, both show higher amplitudes in the oscillations.

From figure 11, it is possible to notice that in both perturbed scenarios the value around which ρ_{max} oscillates varies with respect to the original case with $K = 0$. This can be explained by considering the fact that a neutron star has to be in hydrostatic equilibrium, so that the pressure counteracts the gravity in order for the neutron stars to be stable and not collapse on itself. Therefore, in the scenario with $K = 90$ we need to reach higher densities in order to reach the equilibrium between pressure and gravity, while in the case with $K = 105$ the equilibrium will be reached at lower densities - affecting the resulting ρ_{max}. This can also be seen by comparing the densities grids obtained for the three cases, where we can notice that the radius of the neutron star is directly proportional to the value of K: the smaller radius is obtained with $K = 90$, while the larger radius is found in the scenario with $K = 105$. [1]

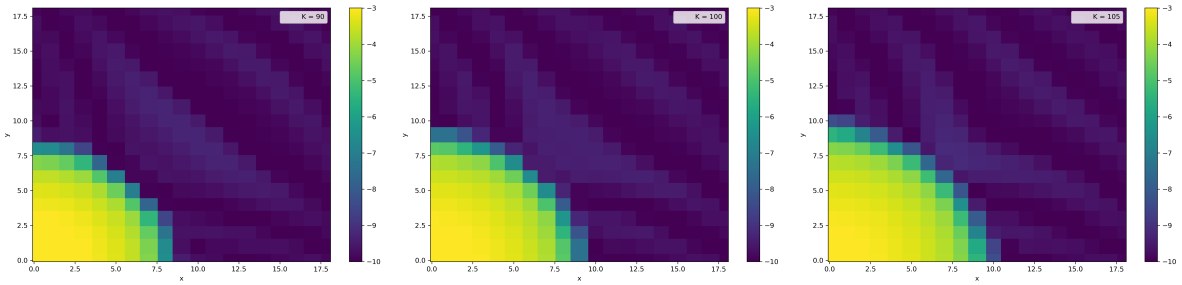

Figure 12: Final density grids resulted from the evolution of the TOV equations for different value of K.As it can be seen from the grids, the resolution is fixed since ΔL is kept constant. Instead, it is possible to see a variation in the radius of the resulting neutron stars due to the equilibrium between the pressure and the gravity being reached at different values of density - that can be translated to different values of radius.

[1] For a better observation of the phenomenon, I inserted in the shared Drive folder and the GitHub folder an animated gif of the evolution of the different scenarios. There, it is possible to see that the different cases oscillate around different values of the radius. Filename: `TOV_evolution_K.gif`

Another observation that can be done from figure 11 is the different oscillation amplitudes between the original and the perturbed scenarios. This can be explained by the fact that the perturbed cases introduce observable oscillations due to the physical nature of the perturbation. Instead, in the $K = 100$ scenario, the dominating oscillations are of numerical nature, due to the usage of discrete grids and the limits related to the chosen resolution. Therefore, it could be useful to study how the resolution would affect the evolution in the perturbed scenarios as well. In particular, I repeated the method used in Sec. 1 for the scenarios with $K = 90$ and $K = 105$ and considered for each direction the following length values for the cells: $\Delta L = 2.0, 1.5, 1.0$[2].

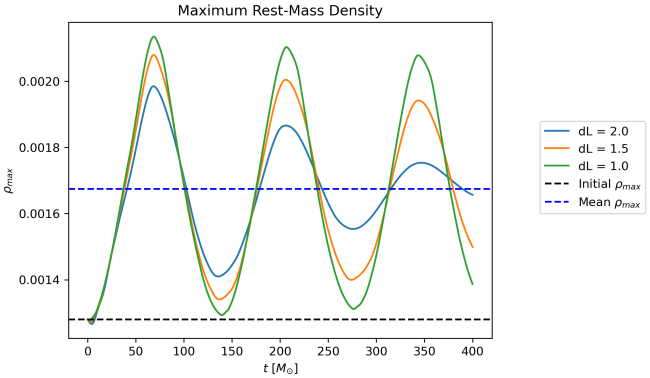

Figure 13: Comparison between ρ_{max} evolution for different resolutions for the scenario with $K = 90$.

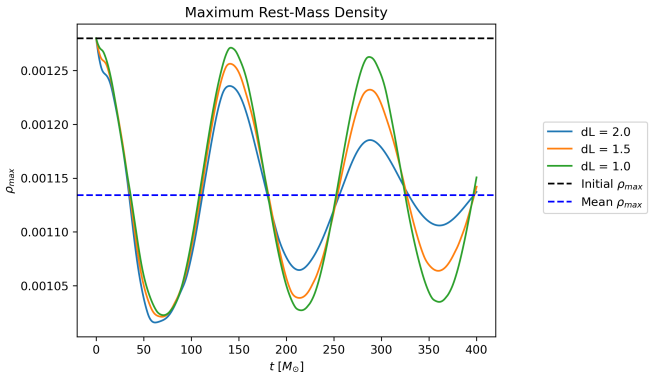

Figure 14: Comparison between ρ_{max} evolution for different resolutions for the scenario with $K = 105$.

As shown in figures 13 and 14, increasing the resolution affects the amplitude of the oscillations and therefore it reduces the dissipation seen in both cases for $\Delta L = 2.0$. This is in agreement with the results obtained in Sec. 1: higher resolutions affect the simulation by reducing the numerical errors. As a result, numerical viscosity effects occur on longer timescales and become less pronounced. Since the fluid is modeled as ideal, no physical viscosity is considered in the simulation: hence, any dissipation observed is purely due to numerical viscosity, which diminishes as the resolution is increased, as previously discussed. Hence, the resulting oscillations maintain a sinusoid-like waveform. However, one can notice that this behavior differs from the $K = 100$ case, where the unperturbed star oscillates at lower amplitude and does not display a clear sinusoid. The higher amplitude of oscillations compared to the scenario with $K = 100$ can be attributed to the physical perturbation introduced by the change in the value of K. The peculiar sinusoid shape, instead, is possibly due to the fact that the perturbed scenarios do not coincide with the corresponding equilibrium solution. Therefore, this leads to the fundamental mode - likely the radial mode - to be excited to larger amplitudes, producing a nearly sinusoidal pattern. Instead, starting near the equilibrium ($K = 100$) leads to a weaker mixture of several modes and overtones: hence, the resulting density evolution looks more complicated than a pure sine wave.

[2]As for the previous case, I inserted in the shared Drive folder and the GitHub folder two animated gifs, to compare the resolution impact on the evolution both for $K = 90$ and $K = 105$. Filenames: `TOV_evolution_K_90_res.gif` and `TOV_evolution_K_105_res.gif`

EINSTEIN TOOLKIT PARAMETERS

1 `MoL::ODE Method = "rk4"`

The Method of Lines (MoL) is a way of separating the time integration from the rest of an evolution scheme. In particular, it is possible in the `MoL` thorn to select different Ordinary Differential Equations (ODEs) integrators to do the time integration while separating it from the spatial integration. This simplifies the problem since partial differential equations are transformed into ODEs. Using the command `MoL::ODE Method = "rk4"`, the integration is done using the 4^{th} order Runge-Kutta integrator. In order to be able to use this method for an initial value problem given by:

$$\frac{dy}{dt} = f(t, y), \tag{2}$$

it is necessary to give the initial condition $t_0, y(t_0)$ and the function f. Then, given a timestep $h > 0$:

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ t_{n+1} = t_n + h \end{cases} \tag{3}$$

gives the `RK4` approximation of $y(t_{n+1})$, here written as y_{n+1}. The value is derived from the current value $y(t_n)$ along with the weighted average of four increments, each of which is the product of h and an estimated slope defined by the function f. The `rk4` integrator is preferred for its significantly improved accuracy over lower-order methods without a major increase in complexity.

2 `GRHydro::recon method = "ppm"`

GRHydro is a fully general-relativistic three-dimensional hydrodynamics code wildly used in astrophysics for working with realistic equations of state and using High Resolution Shock Capturing methods for the hydrodynamics evolution. It uses a Reconstruction-Evolution method in order to estimate the values of physical quantities (e.g. density, pressure, or velocity) at the boundaries between computational grid cells. Hence, the parameter `GRHydro::recon method = "ppm"` suggests that the method chosen to reconstruct these values is the Colella-Woodward Piecewise Parabolic Method (PPM). PPM is a higher-order extension of Godunov's method and uses quadratic functions to represent cell-averages from which new states at cell interfaces are constructed. The fluid state is interpolated using a fourth-order polynomial. It yields third-order accuracy for smooth monotonic functions, whereas other methods - such as the Total Variation Diminishing (TVD) methods - typically yield second-order accurate values.

3 `GRHydro::riemann solver = "Marquina"`

This parameter is also part of the `GRHydro` thorn and is used to specify the selected approximate Riemann solver for the evolution at cell interfaces, where discontinuities in physical quantities are expected. In this context, the chosen method is the `Marquina` solver, which is especially effective for accurately handling strong shocks and discontinuities. Specifically, the `Marquina` solver does not solve the Riemann problem completely, but returns only the flux along the characteristic ray $\xi = 0$. It can be considered as a generalized Roe solver, with a more accurate treatment for supersonic flows. The algorithm for the `Marquina` solver is shown in figure 15.

For $i = 1, \ldots, N$ **do**
  If $\lambda_i(\mathbf{q}_L)\lambda_i(\mathbf{q}_R) > 0$ **then**
    If $\lambda_i(\mathbf{q}_L) > 0$ **then**
      $\phi^i_+ = \phi^i_L$
      $\phi^i_- = 0$
    else
      $\phi^i_+ = 0$
      $\phi^i_- = \phi^i_R$
    endif
  else
    $\alpha^i = \max(|\lambda_i(\mathbf{q}_L), \lambda_i(\mathbf{q}_R)|)$
    $\phi^i_+ = \frac{1}{2}\left(\phi^i_L + \alpha^i \mathbf{w}^i_L\right)$
    $\phi^i_- = \frac{1}{2}\left(\phi^i_R - \alpha^i \mathbf{w}^i_R\right)$
  endif
enddo

Figure 15: Marquina solver algorithm.

4 ML_BSSN (McLachlan BSSN) parameter block

The parameter block introduced by the `ML_BSSN` refers to the module used to solve the Baumgarte-Shapiro-Shibata-Nakamura (BSSN) formulation (shown in Fig. 16), a modification and a strongly hyperbolic version of the Arnowitt-Deser-Misner (ADM) formalism to permit stable and long-term numerical simulations. These two formalisms are useful to rewrite Einstein's field equation in a more suitable form for numerical simulations. Both uses the 3+1 formulation, i.e. they separate the spatial components from the time component.

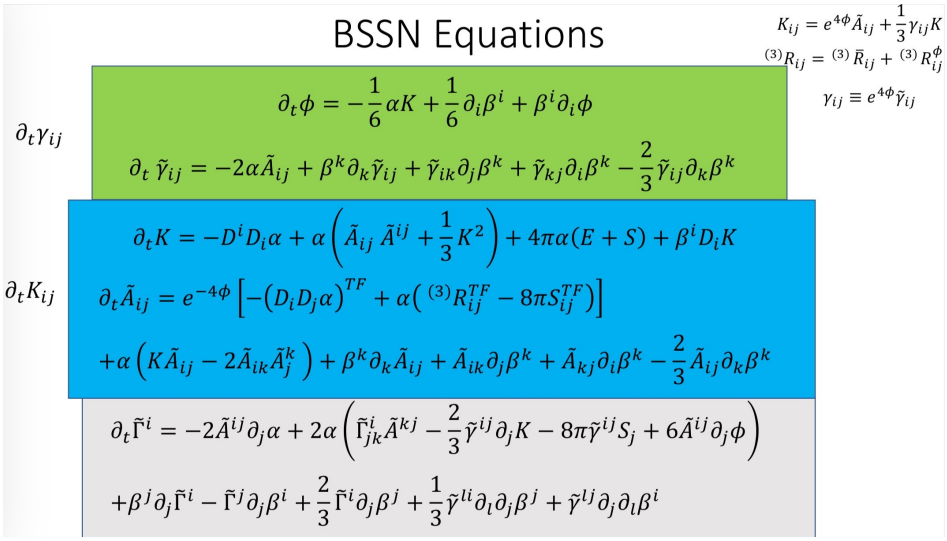

Figure 16: BSSN equations.

The `ML_BSSN parameter block` - here reported in Eq. 4 is related to the free choice for the lapse function α and the shift vector β, which is determined by fixing the Gauge conditions, i.e. by fixing the Slicing condition and the Gamma-Driver Shift condition.

```
ML_BSSN::harmonicN = 1.00 # 1+log
ML_BSSN::harmonicF = 2.00 # 1+log
ML_BSSN::ShiftGammaCoeff = 0.75
ML_BSSN::BetaDriver = 2.66 # common choices are 1/M or 2/M
```
(4)

Slicing Condition: Choosing the right slicing condition is fundamental for the simulation success. In particular, it should verify also the following conditions:

- If singularities are present, these should be avoided (*singularity-avoiding slicing conditions*).
- If coordinate distortions take place, these should be counteracted.
- The gauge conditions should not be computationally expensive.

In particular, the Hyperbolic K-Driver Slicing Condition is usually implemented in numerical simulations and it is given by Eq. 5:

$$(\partial_t - \beta^i \partial_i)\,\alpha = -f(\alpha)\,\alpha^2 (K - K_0), \tag{5}$$

with fixing $f(\alpha) = 1$ or $f(\alpha) = \frac{q}{\alpha}$[3] to recover the *harmonic* slicing condition or the *1+log* slicing condition respectively. Hence, the first two rows in the parameter block in Eq.4 refers to the *1+log* slicing condition, and they are implemented to set $f(\alpha) = \frac{2}{\alpha}$.

Gamma-Driver Shift Condition: The Gamma-Driver Shift Condition is determined by the following set of equations:

$$\begin{cases} \partial_t \beta^i - \beta^j \partial_j \beta^i = C B^i \\ \partial_t B^i - \beta^j \partial_j B^i = \partial_t \Gamma^i - \beta^j \partial_j \Gamma^i - \eta B^i \end{cases} \tag{6}$$

where C and η are the relaxation factor and the damping parameter. Hence, their values are fixed by the last two rows in the parameter block in Eq.4, with C set equal to $\frac{3}{4}$ and $\eta = 2.66$.[4]

[3] Typically, the most used choice is $q = 2$.
[4] Typycally, η varies between $1/2M - 1/M$, where M is the gravitational mass of the system.