Numerical Relativity
Homework 2

Zeduri Pietro

A.Y. 2023-2024

1 SOD Shock Tube Problem:

A SOD shock tube problem involves a system in which the initial conditions are defined by a discontinuity in the rest-mass density and pressure, while the velocity is constant and set to zero. As shown in Figure 1, this situation perfectly fits the definition of a Riemann problem.

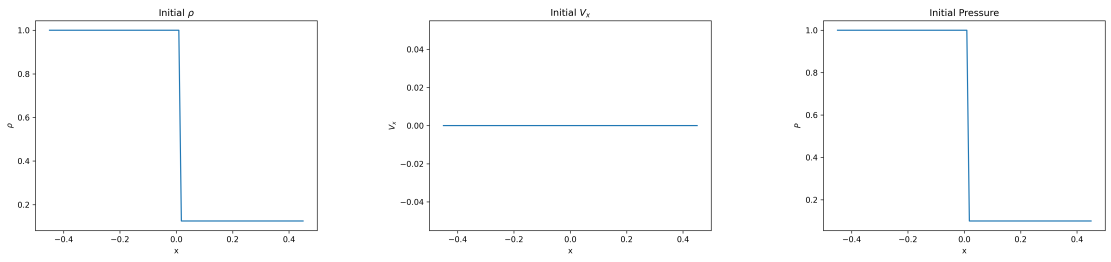

Figure 1: The panels above display the initial condition of the system we need to evolve. From left to right, we have the rest-mass density, the velocity along the x axis, and the pressure.

To evolve this system, we must follow the Euler equations, as they describe the evolution of a perfect fluid in Newtonian physics. To do so, we use the Einstein Toolkit to run simulations. To have a closed set of equations, we introduce a polytropic Equation of State (EoS) with an exponent equal to 5/3, which corresponds to the EoS of an ideal gas. Additionally, we utilize the HLLE approximate Riemann solver to simulate the system's evolution. The simulations run for a time duration of 0.4, and the spatial domain is given by $x \in [-0.5, 0.5]$ (we neglect the other dimensions as the problem is 1D). The resolution of the simulations is determined by the value of dx, which we set to three different values: $dx = 0.02$ (50 grid points), $dx = 0.002$ (500 grid points), and $dx = 0.001$ (1000 grid points). Figure 2 displays the results of these simulations.

Regardless of the resolution, all simulations yield similar solutions. All quantities involved exhibit the formation of a rarefaction wave traveling to the left and a shock wave propagating to the right. While only the rest-mass density shows the formation of a contact discontinuity between the rarefaction wave and the shock wave, where ρ experiences a sudden jump; however, this does not affect the pressure or velocity.

As expected, reducing the value of dx, and thereby increasing the number of grid points, brings the simulated system closer to the exact solution as the resolution increases.

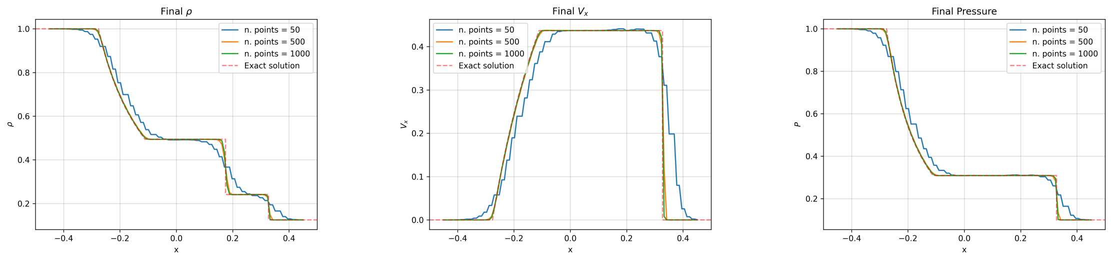

Figure 2: The panels above display the final state of the system when simulated with different resolutions using the Einstein Toolkit (solid lines). The red dashed line is the expected solution. From left to right, we have the rest-mass density, the velocity along x and the pressure.

2 TOV Evolution:

The solution to the TOV equations describes a stable, non-rotating, and spherically symmetric Neutron Star (NS), whose evolution can be simulated numerically using the Einstein Toolkit. We start by setting the initial central density of the NS to $\rho_c = 1.28 \times 10^{-3}\mathrm{M}_\odot^{-2}$ (which corresponds to $\sim 7.99 \times 10^{14}\mathrm{gcm}^{-3}$), and we define the equation of state as a polytrope with $K = 100$ and $\gamma = 2$. Due to the symmetry of the problem, the simulations can cover only one octant of the NS without losing information. This region has a size of $24.0\mathrm{M}_\odot$ (~ 36 km), and the simulations run for $t = 400\mathrm{M}_\odot$ (~ 1.97 ms). We initially set the grid spacing to $dx = 2.0$ and run three simulations, varying the value of the coefficient K to introduce a pressure perturbation in the NS. The values of K used are: $K = 100$, 90, and 105.

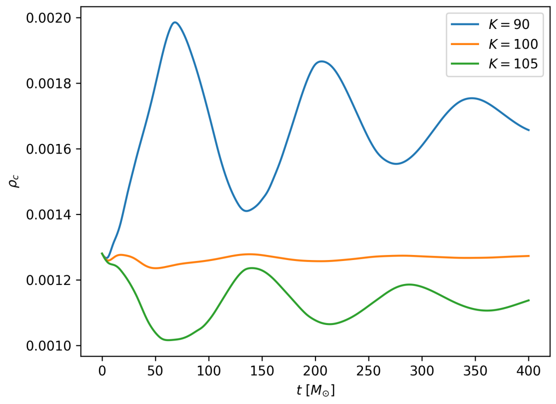

Figure 3: Evolution of the maximum rest-mass density in time for different values of K and with $dx = 2$.

Figure 3 displays the variation in time of the maximum rest-mass density of the NS. In the case of $K = 100$ (in orange), even though there is no physical perturbation, some small oscillations are observable. These are of numerical origin. In our simulations, we attempt to reconstruct a perfect sphere on a discrete grid, which causes a deformation of the NS, resulting in an artificial perturbation that triggers radial oscillations around the equilibrium state. On the other hand, setting $K = 90$ and $K = 105$ introduces a physical perturbation that causes observable oscillations in ρ_c. In the first case, with a lower K, the internal pressure of the star is insufficient to counteract gravity, resulting in the NS shrinking, which increases the rest-mass density. The NS then oscillates around a new stable state. Conversely, $K = 105$ causes the internal pressure to dominate over gravity, leading to an expansion of the NS. This causes ρ_c to decrease to a new equilibrium state, around which the oscillatory motion of the NS continues.

As mentioned earlier, the oscillatory motion observed in Figure 3 can have two causes: the numerical method used to simulate the star or some physical perturbations to the system. The way to differentiate between the two is by changing the resolution of the grid. By reducing the value of dx, we can better reconstruct the perfectly spherical surface of the NS, thereby reducing numerical errors. On the other hand, perturbations of physical origin are expected to decrease more slowly over time, as we can better resolve the region where the oscillatory motion occurs. To verify this, we run two simulations with resolutions $dx = 1.5$ and $dx = 1$. The results, shown in Figure 4, are perfectly consistent with the explanation provided above.

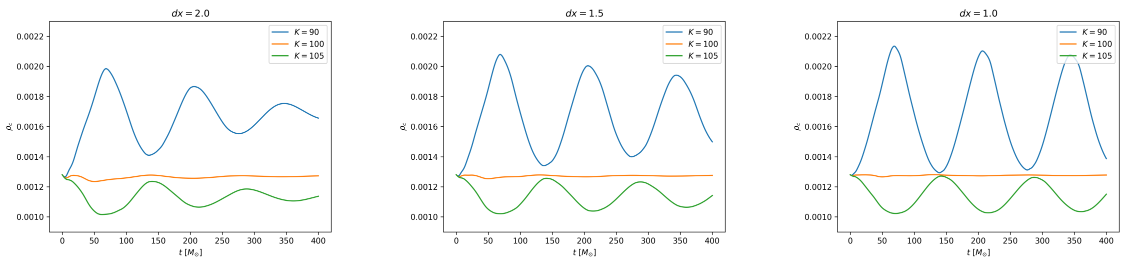

Figure 4: The panels above display the time evolution of ρ_c for three NSs, illustrating how the results change as the resolution of the simulations increases. From left to right, dx is set to 2.0, 1.5, and 1.0, respectively.

3 ETK parameters:

3.1 `MoL::ODE_Method = "rk4"`

The `MoL::ODE_Method` parameter specifies which method should be used to solve the Ordinary Differential Equations (ODEs) obtained when applying the Method of Lines (MoL) to the equations describing the system we are studying. This method is used to separate the time integration from the spatial integration, thereby transforming partial differential equations into ODEs, which simplifies the problem. In this case, the `"rk4"` method, which stands for the 4^{th}-order Runge-Kutta method, is used to solve the ODEs. This method starts from known initial conditions and estimates the function at the next time step using a weighted average of four slopes computed at different points within the current interval. As a 4^{th}-order method in time, `rk4` is widely used because it offers significantly better accuracy than lower-order methods without a large increase in complexity.

3.2 `GRHydro::recon_method = "ppm"`

The `GRHydro::recon_method` parameter specifies the reconstruction method used by the `GRHydro` code, a fully general-relativistic, three-dimensional hydrodynamics code widely used in astrophysics for working with realistic equations of state. A reconstruction method is a technique for estimating the values of physical quantities (like density, pressure, or velocity) at the interfaces between computational grid cells. In this case, the method used is the `"ppm"` method, which stands for Piecewise Parabolic Method, a high-order reconstruction technique particularly useful in simulations involving shocks, discontinuities, and complex fluid flow structures. When using `"ppm"`, the values of physical quantities within each grid cell are reconstructed using a parabolic function, which is then combined with a slope-limiting procedure to calculate interface values at cell edges. This method can achieve third-order accuracy in smooth regions of the flow, offering a significant improvement over linear methods.

3.3 `GRHydro::riemann_solver = "Marquina"`

The `GRHydro::riemann_solver` parameter specifies how the `GRHydro` code solves the Riemann problem at the interfaces between cells, where discontinuities in physical variables are expected. The Riemann solver's purpose is to compute the fluxes of these variables to evolve the system in time. In this case, we are using the `"Marquina"` solver, which is particularly useful for handling strong shocks and discontinuities with high accuracy. This method begins by decomposing the flux function into positive and negative components, based on the direction of wave propagation, which is determined by calculating the eigenvalues of the Jacobian matrix for the system of Euler equations. Differently from other methods, `"Marquina"` introduces an entropy fix in order to avoid unphysical results, like the appearance of non-physical oscillations near shock waves. Such approach allows this Riemann solver to introduce minimal numerical dissipation, while being more computationally expensive than simpler solvers like the HLLE solver.

3.4 The `ML_BSSN` parameters:

The block of parameters introduced by `ML_BSSN` refers to the module used to solve the Baumgarte-Shapiro-Shibata-Nakamura (BSSN) formulation of Einstein's field equations. This formulation is an extended and strongly hyperbolic version of the Arnowitt-Deser-Misner (ADM) formalism, which reformulates Einstein's equations into a form suitable for numerical evolution. Both the ADM and BSSN formalisms use the 3+1 decomposition, meaning they rewrite Einstein's equations by separating the spatial component, represented as 3D space-like hypersurfaces at specific times, from the time component, which governs the evolution of these hypersurfaces from one time slice to the next.

The parameters we need to further analyze are the following:

```
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
```

However, to fully understand their meaning, we first need to discuss the parameters that appear in the BSSN equations, such as $\tilde{\gamma}_{ij}$, the conformal transformation of the 3-metric γ_{ij}. The 3-metric describes the geometry of a three-dimensional spatial slice, defining distances and angles between points in 3D space, and its conformal transformation is defined as:

$$\tilde{\gamma}_{ij} = e^{-4\phi}\,\gamma_{ij} \tag{1}$$

where ϕ is a function of spacetime coordinates, chosen to improve numerical stability.

To fully describe the 4D spacetime in which these hypersurfaces are embedded, we must combine γ_{ij} with the lapse function α, the shift vector β^i, and the extrinsic curvature $K_{\mu\nu}$. The lapse function α determines how proper time flows between different hypersurfaces in the foliation of spacetime, with $d\tau = \alpha\, dt$. Typically, α ranges between 0 and 1, where lower values indicate a more curved geometry. Another key parameter in slicing spacetime is the shift vector β^i, which only has spatial components and describes how spatial coordinates shift between successive time slices, as $dx^i = \beta^i\, dt$. Lastly, we need to introduce the extrinsic curvature $K_{\mu\nu}$ which decribes how a 3-dimensional spatial hypersurface is embedded in the 4-dimensional spacetime, thus measuring the curvature of the hypersurface relative to the surrounding spacetime.

Given the above information, we can start by considering the first two parameters: `ML_BSSN::harmonicN = 1` and `ML_BSSN::harmonicF = 2.0`. These are used to set a specific gauge condition known as the hyperbolic K-driver slicing condition, which controls the evolution of the lapse function. The name is due to the fact that the evolution of α is driven by the value of the trace of the extrinsic curvature K, such that if the curvature remains constant, the lapse does not change. This condition can be written as:

$$\left(\partial_t - \beta^i \partial_i\right) \alpha = -f(\alpha)\alpha^2 \left(K - K_\circ\right), \tag{2}$$

where $f(\alpha)$ is a generic function of the lapse, and $K_\circ$ is the value of the trace of $K_{\mu\nu}$ at $t = 0$. By setting the parameters to the values mentioned above, we are choosing $f(\alpha) = \frac{2}{\alpha}$, which corresponds to the '1+log' slicing condition. This is one of the most popular choices as it effectively slows down the time flow in regions of strong curvature, helping to prevent grid points from falling into singularities during numerical simulations. The name *1+log* derives from the fact that it's common to set initial conditions with $K_\circ = \beta^i = 0$. In this case, equation 2 simplifies to $\partial_t \alpha = -2\alpha K$. Given that $\partial_t \ln(\sqrt{\gamma}) = -\alpha K$, combining these and integrating leads to $\alpha = 1 + \ln(\gamma)$ which explains the origin of the name.

Lastly, the parameters `ML_BSSN::ShiftGammaCoeff = 0.75` and `ML_BSSN::BetaDriver = 2.66` are used to define the shift condition, which determines how the shift vector β^i evolves over time. In this case, the condition used is the Γ-driver shift condition, where Γ refers to the conformal connection functions defined as $\widetilde{\Gamma}^i = \partial_j \tilde{\gamma}^{ij}$. The Γ-driver shift condition is expressed by the following equations:

$$\partial_t \beta^i - \beta^j \partial_j \beta^i = C\, B^i \tag{3}$$

$$\partial_t B^i - \beta^j \partial_j B^i = \partial_t \widetilde{\Gamma}^i - \beta^j \partial_j \widetilde{\Gamma}^i - \eta B^i \tag{4}$$

where B^i is the time derivative of β^i, and C is the relaxation factor, set to $3/4$ by the `ShiftGammaCoeff` parameter. This factor helps control the rate of evolution of the shift vector, making the gauge evolution smoother and preventing excessive numerical instabilities. Additionally, η is the damping parameter, whose value is set by the `BetaDriver` parameter. It is typically chosen between $1/M$ and $1/2M$, where M is the gravitational mass of the system as η has to be constant. The strength of the Γ-driver shift condition lies in its dependence on $\widetilde{\Gamma}^i$. When the metric changes significantly, β^i adjusts accordingly to counteract spatial distortions. This adjustment enhances the stability of numerical simulations, particularly in regions with strong gravitational fields or near singularities.

The parameter files used to run the ETK simulations can be found at https://tinyurl.com/3wfk8694