UNIVERSITY OF TARTU

Faculty of Science and Technology

Institute of Physics

Marie Femke Jaarma

Numerical simulation of a self-interacting scalar field in spherical symmetry

Bachelor's Thesis (12 ECTS)

Supervisor: María José Guzmán Monsalve, PhD

Tartu 2024

Numerical simulation of a self-interacting scalar field in spherical symmetry

Abstract:

The recent detections of gravitational waves produced by a black hole merger and colliding neutron stars by the LIGO/Virgo Scientific Collaboration became a significant source of motivation for two fields of research: modified gravity and numerical relativity. It is the opinion of both fields that the key to understanding the physics of strong gravitational regimes lies in numerical simulations, as they enable us to simulate nonlinear dynamics. This has further motivated numerical approaches since these simulations can be computationally expensive and take a long time to run. In this thesis, we have worked out one of the simplest but most important models in numerical relativity: the self-interacting scalar field in spherical symmetry. Following the work of F.S. Guzmán, we obtained agreement in the evolution of both the metric variables and the scalar field. The code, which was written in Python with the use of the NumPy library, will be improved upon in the future to more accurately match convergence tests.

Keywords: numerical relavity, numerical methods, scalar field collapse

CERCS: P190 - Mathematical and general theoretical physics, classical mechanics, quantum mechanics, relativity, gravitation, statistical physics, thermodynamics.

Iseendaga interakteeruva skalaarvälja sfääriliselt sümmeetriline numbriline simulatsioon

Lühikokkuvõte:

LIGO/Virgo teaduslik koostööprojekt tuvastas hiljuti mustade aukude ühinemise ja neutrontähtede kokkupõrkel tekkinud gravitatsioonilaineid. Saavutus osutus oluliseks motivatsiooniallikaks kahe uurimisvaldkonna jaoks: alternatiivne gravitatsiooniteooria ja numbriline relatiivsusteooria. Mõlemad valdkonnad on ka arvamusel, et tugeva gravitatsiooni režiimide füüsika mõistmise võti peitub numbrilistes simulatsioonides, kuna nende abil on võimalik jäljendada mittelineaarset dünaamikat. See on omakorda suurendnud huvi numbriliste lähenemise vastu, kuna seda tüüpi simulatsioonid võivad olla arvutuslikult kulukad ning nende käivitamine võtab kaua aega. Käesoleva lõputöö raames kontrollisime numbrilise relatiivsusteooria ühte lihtsaimat, kuid olulisemat mudelit: iseendaga interakteeruv skalaarväli sfäärilises sümmeetrias. Võttes eeskujuks F.S. Guzmáni poolt toodud meetodid ja algoritmi, saime tulemuseks meetrika muutujate ja skalaarvälja evolutsiooni, mis on Guzmáni tulemustega kooskõlas. Kood, mis on kirjutatud Pythonis

kasutades NumPy teeki, nõuab tulevikus täiustamist, et parandada numbrilise vea koondumist.

Võtmesõnad: numbriline relatiivsus, numbrilised meetodid, skalaarvälja kollaps

CERCS: P190 - Matemaatiline ja üldine teoreetiline füüsika, klassikaline mehaanika, kvantmehaanika, relatiivsus, gravitatsioon, statistiline füüsika, termodünaamika.

Contents

1 General relativity **6**

1.1 Geometry 6

1.2 Curvature 6

1.3 Einstein's field equations 8

2 Numerical relativity **10**

2.1 3+1 formalism 10

2.2 Hyperbolicity and well-posedness 11

2.2.1 Well-posedness 11

2.2.2 Hyperbolicity 12

3 Numerical methods **14**

3.1 Finite difference 14

3.2 Numerical stability and convergence 16

3.3 Method of lines 17

3.4 Advection equation 19

4 Self-interacting scalar field **21**

4.1 Initial data 23

4.2 Evolving the data 23

4.3 Results 24

5 Conclusion **28**

References **29**

A Derivations **32**

A.1 Deriving Einstein's equation 32

A.2 Deriving Klein-Gordon equations 34

B Code **35**

B.1 Advection equation 35

B.2 Self-interacting scalar field 36

Introduction

In 1915, Albert Einstein formulated a new theory of gravity: general relativity (GR). This new theory was more accurate than the previously widely accepted Newtonian gravity, explaining why light bends around massive objects and why Mercury's observed orbit doesn't behave as theory suggests [1, 2]. The theory isn't without its flaws, though. Without the addition of dark matter, GR on its own isn't able to explain the observed rotation curves of galaxies; without dark energy, it can't explain the accelerating expansion of the universe [3].

With the recent visual confirmation of a black hole [4] and the first detected gravitational waves from two black holes colliding [5], scientists have gotten confirmation of gravitational phenomena predicted by GR. This, however, does not mean that GR is accurate at describing all gravitational events. Gravity has only been well tested on the scale of our solar system and in relatively weak gravity regimes. It is likely that more considerable distances and stronger gravity regimes require corrections in GR [6]. These shortcomings have motivated scientists to find other theories or formulations of gravity that don't fail where GR does.

Another idea to combat this lack of analytical insight is to take a numerical approach. This field of research, known as numerical relativity, aims to solve Einstein's equations numerically so that the absence of an exact analytical solution would no longer prove to be an issue. This numerical approach is also a source of motivation for finding and testing modified theories of gravity. More specifically, for theories belonging to the geometric trinity of gravity (composed of GR, the symmetric teleparallel equivalent of GR, and the teleparallel equivalent of GR), there is hope that the differences between the formulations will emerge in the numerical domain [7].

The aim of this thesis is to recreate the results of the example of the self-gravitating scalar field introduced in F.S. Guzmán's article "Introduction to numerical relativity through examples." This goal is achieved by first decomposing the system of equations using the appropriate numerical methods and implementing the algorithm in Python using the NumPy library.

The thesis is divided into four main sections. Section 1 provides an overview of GR and its geometry, focusing on curvature and Einstein's field equations. Section 2 discusses the 3+1 decomposition of GR, as well as gives an overview of well-posedness and hyperbolicity, which are required for a numerically stable system of equations. Section 3 focuses on the specific numerical methods used and their implementation via the example of the advection equation. Section 4 describes the setup and process of evolving a scalar field in spherical symmetry, and presents the results of the simulations. The overall conclusion is given in Section 5.

1 General relativity

The aim of this chapter is to give an overview of Einstein's equations and the basics of general relativity that form the theoretical foundation of the first part of the thesis. This section is based on [8] unless stated otherwise.

1.1 Geometry

We will be using the Greek alphabet to denote indices related to spacetime (i.e. $\mu, \nu, \sigma, \rho = 0, 1, 2, 3$) and (i, j, k = 1, 2, 3) to denote purely spatial indexes. Our chosen signature for the metric is (-, +, +, +) so that for a flat Minkowski spacetime the metric reads

$$g_{\mu\nu} \equiv \eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The line element is calculated from the metric as

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu, \tag{1}$$

where dx^μ and dx^ν are infinitesimal coordinate displacements.

From the metric we can also define the connection coefficients (also called Christoffel symbols):

$$\Gamma^\lambda{}_{\mu\nu} = \frac{1}{2} g^{\lambda\rho} \left(\partial_\nu g_{\mu\rho} + \partial_\mu g_{\nu\rho} - \partial_\rho g_{\mu\nu} \right). \tag{2}$$

The connection is an integral part of describing curved spacetime and is used to define the generalization of the partial derivative, called the covariant derivative:

$$\nabla_\mu V^\nu = \frac{\partial V^\nu}{\partial x^\mu} + \Gamma^\nu{}_{\mu\sigma} V^\sigma \equiv \partial_\mu V^\nu + \Gamma^\nu{}_{\mu\sigma} V^\sigma. \tag{3}$$

For a flat spacetime the connection is vanishing, so the covariant derivative turns into a regular partial derivative.

1.2 Curvature

According to Einstein's theory of general relativity, objects with mass cause spacetime to distort. This distortion, more commonly referred to as curvature, is typically quantified by the Riemann

tensor

$$R^{\rho}{}_{\sigma\mu\nu} = \partial_{\mu}\Gamma^{\rho}{}_{\nu\sigma} - \partial_{\nu}\Gamma^{\rho}{}_{\mu\sigma} + \Gamma^{\rho}{}_{\mu\lambda}\Gamma^{\lambda}{}_{\nu\sigma} - \Gamma^{\rho}{}_{\nu\lambda}\Gamma^{\lambda}{}_{\mu\sigma}. \tag{4}$$

To further examine the properties of this tensor we will be lowering the indices

$$R_{\rho\sigma\mu\nu} = g_{\rho\lambda}R^{\lambda}{}_{\sigma\mu\nu}. \tag{5}$$

In this fully covariant form, two important qualities appear:

- the sum of cyclic permutations of the last three indices is zero:

$$R_{\rho[\sigma\mu\nu]} = 0; \tag{6}$$

- the cyclic permutations of the first three indices of a covariant derivative is zero (also known as the Bianchi identity):

$$\nabla_{[\lambda}R_{\rho\sigma]\mu\nu} = 0. \tag{7}$$

These qualities can be used to simplify complex equations and will be useful later on.

The Riemann tensor fully describes the curvature, but due to its four indices calculating components by hand or decomposing the tensor may prove to be difficult. It is because of this that we are interested in finding a coordinate invariant tensor with fewer indices. Contracting over the upper and second lower index we get the Ricci tensor

$$R_{\mu\nu} = R^{\lambda}{}_{\mu\lambda\nu}, \tag{8}$$

which in turn can be used to find the Ricci scalar

$$R = g^{\mu\nu}R_{\mu\nu}. \tag{9}$$

Using equations Eq. (8) and Eq. (9) we can define the Einstein tensor as

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R. \tag{10}$$

An important property arises when we contract twice over the Bianchi identity:

$$
\begin{aligned}
0 &= g^{\nu\sigma} g^{\mu\lambda} (\nabla_\lambda R_{\rho\sigma\mu\nu} + \nabla_\rho R_{\sigma\lambda\mu\nu} + \nabla_\sigma R_{\lambda\rho\mu\nu}) \\
&= \nabla^\mu R_{\rho\mu} - \nabla_\rho R + \nabla^\nu R_{\rho\nu} \\
&\implies \nabla^\mu R_{\rho\mu} = \frac{1}{2} \nabla_\rho R. \qquad (11)
\end{aligned}
$$

This in turn is equivalent to

$$\nabla^\mu G_{\mu\nu} = 0 \qquad (12)$$

1.3 Einstein's field equations

In the absence of matter or energy, spacetime can be described by just Eq. (10). We are not interested in simulating standalone black holes, so this case does not interest us. We will be introducing matter in the form of the energy-momentum tensor $T_{\mu\nu}$. In GR the energy-momentum conservation is stated in the form

$$\nabla^\mu T_{\mu\nu} = 0. \qquad (13)$$

From Eq. (12) we know that the Einstein tensor obeys the same condition, so we propose the following field equation for the metric:

$$G_{\mu\nu} = \kappa T_{\mu\nu}, \qquad (14)$$

where $\kappa = 8\pi G$. Combining equations Eq. (10) and Eq. (13) we get Einstein's equation for general relativity

$$R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R = 8\pi G T_{\mu\nu}, \qquad (15)$$

which tells us how spacetime curves in the presence of energy-momentum.

Other than Eq. (13), we still don't know anything about the energy-momentum tensor. To get some insight, we will be considering the action

$$S = \frac{1}{16\pi G} S_H + S_M, \qquad (16)$$

where $S_H = \int \sqrt{-g} R d^n x$ is the Hilbert action and S_M the action for matter. Varying S_H with respect to the metric we obtain

$$\delta S_H = \int d^n x \sqrt{-g} \left[R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} \right] \delta g^{\mu\nu}. \tag{17}$$

The functional derivative of the action satisfies the condition

$$\delta S = \int \sum_i \left(\frac{\delta S}{\delta \phi^i} \delta \phi^i \right) d^n x, \tag{18}$$

where $\{\phi^i\}$ is a complete set of fields being varied, which in our case is just $g^{\mu\nu}$. From this we can derive Einstein's equation in vacuum:

$$\frac{1}{\sqrt{-g}} \frac{\delta S_H}{\delta g^{\mu\nu}} = R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} = 0. \tag{19}$$

Reintroducing the matter action, we obtain the following solution for the whole action:

$$\frac{1}{\sqrt{-g}} \frac{\delta S}{\delta g^{\mu\nu}} = \frac{1}{16\pi G} \left(R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} \right) + \frac{1}{\sqrt{-g}} \frac{\delta S_M}{\delta g^{\mu\nu}} = 0. \tag{20}$$

With the choice

$$T_{\mu\nu} = -2 \frac{1}{\sqrt{-g}} \frac{\delta S_M}{\delta g^{\mu\nu}} \tag{21}$$

we recover Einstein's equation in its original form of Eq. (15).

Now that we know how exactly the energy-momentum tensor depends on the action, we can specify what type of matter we are considering. In this thesis, the source of matter is a single scalar field, which has the action

$$S_\phi = \int \left[-\frac{1}{2} g^{\mu\nu} (\nabla_\mu \phi)(\nabla_\nu \phi - V(\phi)) \right] \sqrt{-g} d^n x, \tag{22}$$

where $V(\phi)$ is the potential of the scalar field. Varying this action with respect to the inverse metric we obtain the energy-momentum tensor for a scalar field:

$$T^{(\phi)}_{\mu\nu} = -2 \frac{1}{\sqrt{-g}} \frac{\delta F_\phi}{\delta g^{\mu\nu}} = \nabla_\mu \phi \nabla_\nu \phi - \frac{1}{2} g_{\mu\nu} \left(g^{\rho\sigma} \nabla_\rho \phi \nabla_\sigma \phi - 2V(\phi) \right).$$

Another important feature that can be derived from the action of the scalar field is the Klein-Gordon equation. To obtain this, we instead vary Eq. (22) with respect to the scalar field [9] and eventually get

$$\Box \phi - m^2 \phi = 0. \tag{23}$$

2 Numerical relativity

This section explains the basic formalism used in numerical relativity and introduces the conditions that need to be met to have a stable and accurate system of equations. It is based on [10] unless stated otherwise.

2.1 3+1 formalism

So far we have only worked with Einstein field equations in a fully covariant way, with no meaningful distinction between space and time. To simulate any physical scenario or evolve something in time, this split has to be made. The most intuitive split is done using the 3+1 formalism, which we will be focusing on.

The basis of 3+1 formalism is relatively simple: spacetime is sliced into three-dimensional spacelike slices, where each slice represents some certain time t. This sliced version of spacetime is commonly referred to as the foliation of spacetime, where each slice is often viewed as a hypersurface, noted by Σ_t.

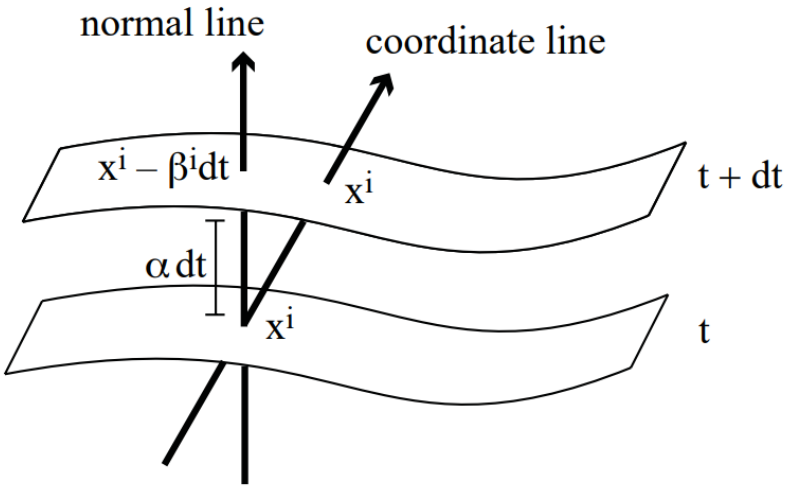

Figure 1. Two neighboring "leaves" of foliation [10].

Figure 1 shows two adjacent hypersurfaces, as well as the definitions of two important functions, α and β. The proper time $d\tau$ of Eulerian observers moving along the normal line is defined as

$$d\tau = \alpha(t, x^i)dt, \tag{24}$$

where α is the lapse function, and t and dt are the universal time and infinitesimal increment

of universal time respectively. Eulerian observers traveling along the normal line, as opposed to the coordinate line, experience a coordinate displacement. Their new position on the second hypersurface depends on the original position as

$$x^i_{d+dt} = x^i_t - \beta^i(t, x^i)dt, \tag{25}$$

where β^i is the shift vector.

By using the three-dimensional metric γ_{ij}, we can define proper distance within the hypersurface as

$$dl^2 = \gamma_{ij}dx^i dx^j. \tag{26}$$

Using equations (24-26), one is able to derive Einstein's equations for 3+1 decomposition using differential geometry. For a spherically symmetric spacetime, the problem can be simplified into 1+1 dimensions, and the equations can be instead derived using the covariant formalism as done in Appendix A.1. The 3+1 split has been thoroughly explained in Chapter 2 in [10], but is otherwise outside the scope of this thesis.

2.2 Hyperbolicity and well-posedness

Using the 3+1 formalism, one can derive countless variations of the evolution equations for the same system. It stands then to reason that some of these equations are more numerically stable and physically accurate than others. It is then the opinion of numerical relativists and mathematicians, that in order for a system of equations to result in a stable simulation, the equations have to be well-posed and strongly hyperbolic.

2.2.1 Well-posedness

Well-posedness as a property states, that the solutions of a system must depend continuously on the initial data. This means that, for an initial value or Cauchy problem, any small changes in the initial data cause small changes in the solution.

For a more formal definition, let us consider the following system of partial equations:

$$\partial_t u = P(D)u, \tag{27}$$

where u is a system of equations comprising n functions of space and time, and $P(D)$ is a $n \times n$ differential operator, whose components depend only on spatial derivatives. The initial data for u can be written as $u(t = 0, x)$. It is stated, that a system of partial differential equations is well-posed if we can define a norm $|| \cdot ||$ so that

$$||u(t, x)|| \leq ke^{\alpha t}||u(0, x)||, \tag{28}$$

where k and α are constants that do not depend on the initial data.

Let us look at the example of the advection equation [11]

$$\partial_t u(x, t) + v\partial_x u(x, t) = 0, \quad x, v \in \mathbb{R} \tag{29}$$

with the initial condition

$$u(x, 0) = e^{i\omega x}\hat{f}(\omega). \tag{30}$$

The advection equation has a well-known analytical solution, which in this example is

$$u(x, t) = e^{i\omega(x-vt)}\hat{f}(\omega). \tag{31}$$

This example is clearly bounded and represents a wave with a constant amplitude. If the speed v is positive, the wave will travel to the right; if negative, it will travel to the left.

2.2.2 Hyperbolicity

For a larger system of equations or non-exponential initial conditions, finding the exact solution may prove to be difficult. We are thus motivated to introduce another property. If a system is found to be strongly hyperbolic, it will behave similarly to a simple wave equation, which is why the initial value problem for this equation is well-posed. To explain this concept, we are now considering a first-order system of evolution equations written as

$$\partial_t u + M^i \partial_i u = s(u), \tag{32}$$

where M^i are $n \times n$ matrices and $s(u)$ is a source term not dependent on any derivatives of u. M^i, which are known as characteristic matrices, can be used to compose the matrix

$$P(n_i) = M^i n_i, \tag{33}$$

where n_i is some unit vector. Equation (32) is said to be strongly hyperbolic if $P(n_i)$ has real eigenvalues and a complete set of eigenvectors for all n_i. If the first condition is met, but P doesn't have a complete set of eigenvectors, the system is weakly hyperbolic and, therefore, either weakly well-posed or ill-posed.

3 Numerical methods

Due to the complexity of most differential equations in physics, a numerical approach is often used, the most common being finite differencing methods. This chapter aims to give an overview of the main numerical methods used in numerical relativity and, by extension, this thesis. This chapter is based on Chapter 9 of [10] unless it is specified otherwise.

3.1 Finite difference

Spacetime as we know it consists of infinitesimally small elements, which is impossible to replicate on computers. To overcome this issue, spacetime is discretized to have a set of discrete points, called a computational grid, instead of a continuous spacetime. An example of a 1+1 dimensional discretized grid can be seen in Fig 2, where distances between two points are denoted by Δ. With this notation, any current time or spatial coordinate value can be given as the stepsize multiplied by the coordinate index: $t^n = n\Delta t$ and $x_i = i\Delta x$.

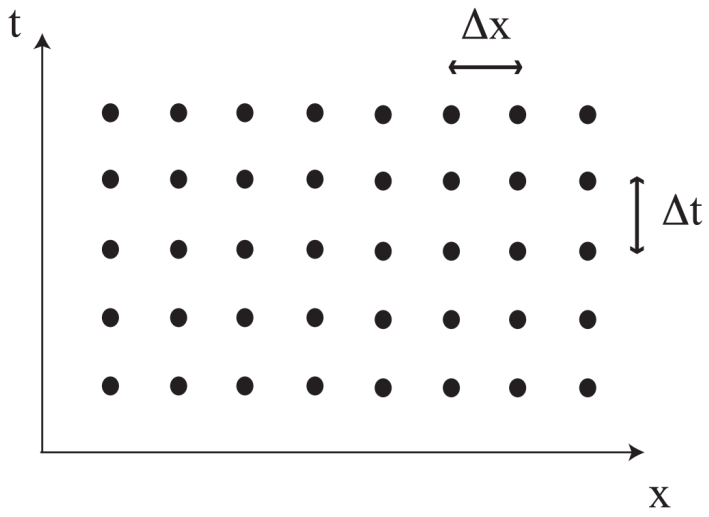

Figure 2. A simple example of a 1+1 dimensional computational grid with uniform distances between gridpoints [10]

To further illustrate the application of this notation, let us look at the function $u \equiv u(x,t)$, which in its discretized version can be written as $u(x_i, t^n) \equiv u_i^n$. By using the Taylor expansion, we can get the following expressions for the u values of the nearest spatial neighbors of u_i:

$$u_{i-1} = u_i - \Delta x u_i' + \frac{\Delta x^2}{2} u_i'' - \frac{\Delta x^3}{6} u_i''' + \mathcal{O}(\Delta x^4) \tag{34}$$

$$u_{i+1} = u_i + \Delta x u_i' + \frac{\Delta x^2}{2} u_i'' + \frac{\Delta x^3}{6} u_i''' + \mathcal{O}(\Delta x^4), \tag{35}$$

where the prime is used to denote spatial derivatives, and $\mathcal{O}$ shows the order of the error. Subtracting Eq. (34) from Eq. (35) gives us the central difference

$$u_{i+1} - u_{i-1} = 2\Delta x u_i' + 2\frac{\Delta x^3}{6} u_i''' + \mathcal{O}(\Delta x^4), \tag{36}$$

which in turn gives us

$$u_i' = \frac{u_{i+1} - u_{i-1}}{2\Delta x} - \frac{\Delta x^3}{6} u_i''' + \mathcal{O}(\Delta x^4). \tag{37}$$

It is possible to rewrite the third-order derivative in a similar manner to get a high-order accurate approximation for u_i'. However, for the sake of this example, we will limit ourselves to a second-order accurate approximation

$$u_i' = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2). \tag{38}$$

The same derivative approximation can be achieved via the limit definition:

$$u_i' = \lim_{\Delta x \to 0} \frac{u_{i+1} - u_i}{\Delta x} = \lim_{\Delta x \to 0} \frac{u_i - u_{i-1}}{\Delta x}, \tag{39}$$

which, when averaged, gives us the expression

$$u_i' = \lim_{\Delta x \to 0} \frac{u_{i+1} - u_{i-1}}{2\Delta x}. \tag{40}$$

This formula for finding the central finite difference approximation (FDA) can be used almost anywhere in the computational grid, except for the edges. There we will instead use the first two components of the Taylor series:

$$u_i' = \frac{u_{i+1} - u_i}{\Delta x} + \mathcal{O}(\Delta x) \tag{41}$$

$$u_i' = \frac{u_i - u_{i-1}}{\Delta x} + \mathcal{O}(\Delta x), \tag{42}$$

which are respectively called forward and backward finite differences [12].

This numerical differential approximation will be directly used for spatial derivatives only and indirectly for temporal derivatives.

3.2 Numerical stability and convergence

Having discretized a basic spatial derivative, we must now ensure that numerical solutions obtained using our FDA match the original differential equation solution. This can be broken down into three conditions: consistency, convergence, and stability. To illustrate these conditions we will be using the differential equation in the general form

$$\mathcal{L}u = 0, \tag{43}$$

where $\mathcal{L}$ is some differential operator acting on u. The discrete version of Eq (43) will thus take the form

$$\mathcal{L}_\Delta u_\Delta = 0. \tag{44}$$

Consistency

An important measure of consistency is the truncation error, which is defined as the result of applying the finite difference operator $\mathcal{L}_\Delta$ to the original differential equation solution u:

$$\tau_\Delta := \mathcal{L}_\Delta u. \tag{45}$$

The minimum requirement for any FDA is that the truncation errors go to zero as the resolution of the grid is increased:

$$\lim_{\Delta \to 0} \tau_\Delta = 0. \tag{46}$$

However, consistency is only a local property and is not enough to guarantee the accuracy of our FDA.

Convergence

Convergence is defined by the solution error as

$$\epsilon_\Delta := u - u_\Delta,$$

$$\lim_{\Delta \to 0} \epsilon_\Delta = 0. \tag{47}$$

For any numerically relevant results, the convergence criterion has to be met. Proving the convergence analytically may prove to be difficult, but numerically it's far from it. The numerical convergence of some function u with a known exact solution:

$$\frac{u_\Delta - u}{u_{\Delta/2} - u} = \frac{\Delta^2 + \mathcal{O}(\Delta x^3)}{\frac{1}{4}\Delta^2 + \mathcal{O}(\Delta)} = 4 + \mathcal{O}(\Delta x^3). \tag{48}$$

Numerical convergence of function u with an unknown exact solution using three resolutions [12]:

$$\frac{u_\Delta - u_{\Delta/2}}{u_{\Delta/2} - u_{\Delta/4}} = \frac{\Delta^2 + \mathcal{O}(\Delta)}{\frac{1}{4}\Delta^2 + \mathcal{O}(\Delta)} = 4 + \mathcal{O}(\Delta). \tag{49}$$

It is important to note that even if the convergence rate is correct, it is possible that the numerical solution is actually converging to something else, not the exact solution.

Stability

While convergence is a good indicator of the accuracy of the numerical scheme, another property that is just as important is stability, which states that the numerical solution should remain bounded. The Lax equivalence theorem allows us to connect all three mentioned conditions. The theorem states, that given a well-posed initial problem and consistent FDA, then stability is a necessary and sufficient condition for convergence [13].

There exists another stability condition that allows us to connect convergence to stability called the Courant-Friedrich-Lewy (CFL) condition. The CFL stability condition states, that the physical domain of dependence has to be smaller than the numerical domain of dependence. This can be written as

$$c\rho \leq 1 \implies c\Delta t \leq \Delta x, \tag{50}$$

where ρ is the Courant parameter.

A more popular notation is $\Delta t = \rho \Delta x$ [14], where $c = 1$ and the Courant parameter is usually chosen to be between 0.1 and 0.3.

3.3 Method of lines

The method of lines is a powerful tool when solving partial differential equations (PDEs), especially mixed PDEs that contain first-order temporal and spatial derivatives. To illustrate

this method, we will be considering a semi-discrete system, where only the time coordinate is discretized and spatial derivatives are assumed to approach the exact solution [15]. The evolution equation of u is written as

$$\partial_t u = S, \tag{51}$$

where S is the right-hand side of the equation containing the remaining terms of the evolution equation, including any spatial derivatives. Equation (51) can be rewritten using the forward difference approximation (41), so that with some modification we obtain the value of u at time t^{n+1}:

$$u^{n+1} = u^n + \Delta t S(u^n). \tag{52}$$

It is clear to see that it is entirely possible to evolve a system in time using only current values. The method as it is currently implemented still has its faults, namely the low order of accuracy. It is possible to increase the order by involving previous time steps, but that comes at a great cost to the computational efficiency. We opt instead to use a fourth-order Runge-Kutta (RK) algorithm, still using Eq. (52) as the foundation. A simple derivation of RK order 4 can be found in [16]. The final evolution equation can then be written as

$$u^{n+1} = u^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(\Delta t^4), \tag{53}$$

where

$$\begin{aligned} k_1 &= S(u^n), \\ k_2 &= S(u^n + k_1 \Delta t/2), \\ k_3 &= S(u^n + k_2 \Delta t/2), \\ k_4 &= S(u^n + k_3 \Delta t). \end{aligned}$$

While this is still computationally less efficient than the forward finite difference method, it is a good compromise regarding computational efficiency and order of accuracy.

Artificial dissipation

Artificial dissipation is often times used to increase the stability of numerical schemes and dampen high-frequency modes [17]. A standard way to apply artificial dissipation is

$$\partial_t u = S + Qu, \tag{54}$$

where Q is a differential operator acting on u. This differential operator is chosen to be higher-order accurate (i.e. a 6th-order accurate FDA for a 4th-order accurate numerical method) and is multiplied by a dampening factor.

3.4 Advection equation

In order to test the accuracy of our numerical setup, we will start with the example of the advection equation:

$$\partial_t u + v\partial_x u = 0, \tag{55}$$

where u is a wave function and v its speed. This equation has an analytical solution and is similar to other equations in this thesis, making it a suitable exercise before moving forward.

We will use the notation introduced in Section 3.1. Using the method of lines, we can rewrite the equation to be

$$\frac{du_m}{dt} = -\frac{v}{2\Delta x}(u_{m+1} - u_{m-1}) \equiv S(u), \tag{56}$$

where we have used centered differences to discretize the spatial derivatives. It is clear to see that we can substitute time derivatives with spatial approximations. Using the 4th order Runge-Kutta method the final solution [10] will be

$$u^{n+1} = u^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{57}$$

where

$$\begin{aligned} k_1 &= S(u^n), \\ k_2 &= S(u^n + k_1\Delta t/2), \\ k_3 &= S(u^n + k_2\Delta t/2), \\ k_4 &= S(u^n + k_3\Delta t). \end{aligned}$$

We can implement this algorithm in Python using the module NumPy. Choosing the initial data to be

$$\begin{aligned} u(0, x) &= e^{-x^2/\sigma} \\ u(-2) &= u(2), \end{aligned}$$

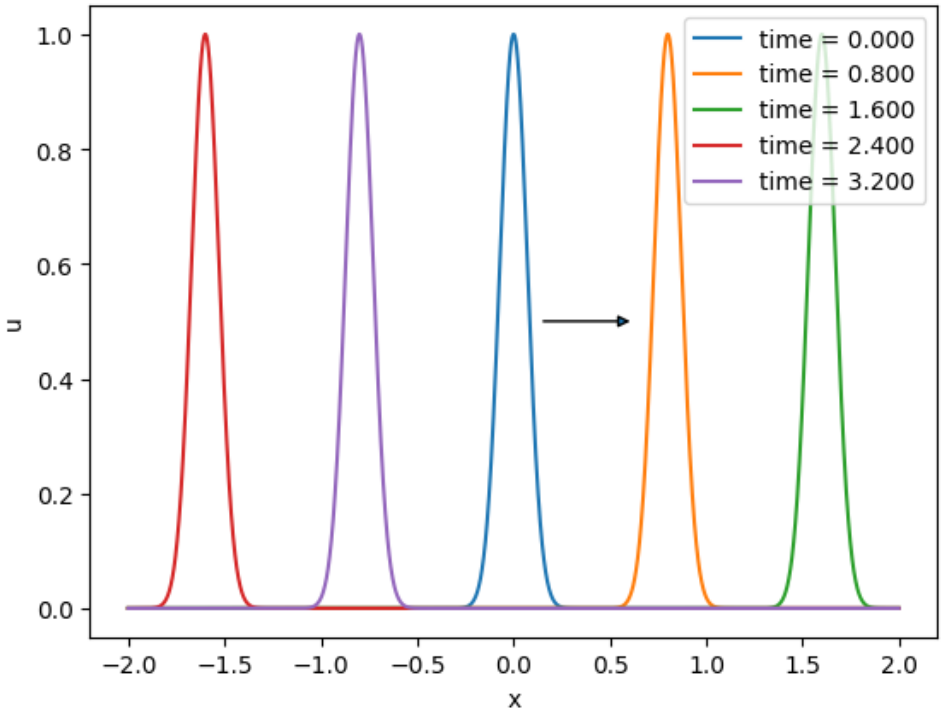

Figure 3. Advection equation for $v = 1$, $\sigma = 0.01$ and enforced periodic conditions. For a positive speed, the wave will propagate toward the right and cross the boundary to $x = -2$.

we obtained the results in Figure 3.

For this example, we have chosen $dx = 0.002$ and the speed to be $v = 1$, so after four units of time, the wave has completed one cycle. Comparing the numerical solution to the exact one, we get that the root mean square error (RMSE) for $t = 0$ is 0. This is to be expected, as this is the initial data. Towards the end of the cycle at $t = 3.52$, the error is approximately 0.00161. Decreasing the step size to $dx = 0.001$, the RMSE at that same time is 0.0004. Decreasing the resolution twofold results in the error being reduced four times, which is a good indicator of convergence of our 4th order scheme.

This error is sufficiently small for this time scale. However, lowering the resolution of the numerical grid results in unpredictable oscillating behavior, while increasing the runtime and spatial range increases the error even when the resolution is sufficiently high. The relevant code can be found in Appendix B.1.

4 Self-interacting scalar field

Using the methods described above, we will follow the example of F. S. Guzmán's article "Introduction to numerical relativity through examples" [12], where we have a wave equation that is associated to a scalar field in a curved spacetime with spherical symmetry. The stress-energy tensor for the scalar field is

$$T_{\mu\nu} = \phi_{,\mu}\phi_{,\nu} - \frac{1}{2}g_{\mu\nu}[\phi^{'\alpha}\phi_{'\alpha} + 2V(\phi)], \tag{58}$$

where $V(\phi) = \frac{1}{2}m^2\phi^2$ is the scalar field potential and m is the mass of a spinless boson represented by the scalar field. To remove some restraints later on, we introduce the scaled variables $\phi \to \sqrt{\kappa_0}\phi$ and $t \to mt$. We assume that the line element in Schwarschild coordinates can be written as

$$ds^2 = -\alpha^2(r,t)dt^2 + a^2(r,t)dr^2 + r^2 d\Omega^2, \tag{59}$$

where α is the lapse function on the 3+1 decomposition and $d\Omega^2 = d\theta^2 + \sin^2\theta d\varphi$. In the interests of this thesis, we take the shift vector to be zero. The spacetime metric then reads

$$g_{\mu\nu} = \begin{pmatrix} -\alpha^2 & 0 & 0 & 0 \\ 0 & a^2 & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2\sin^2\theta \end{pmatrix}, \tag{60}$$

$$g^{\mu\nu} = \begin{pmatrix} -1/\alpha^2 & 0 & 0 & 0 \\ 0 & 1/a^2 & 0 & 0 \\ 0 & 0 & 1/r^2 & 0 \\ 0 & 0 & 0 & 1/(r^2\sin^2\theta) \end{pmatrix}. \tag{61}$$

The Ricci scalar for this metric was calculated to be

$$R = \frac{2}{r^2a^3\alpha^3}(-a\alpha^3 + a^3\alpha^3 + 2r\alpha^3\partial_r a - 2ar\alpha^2\partial_r\alpha \tag{62}$$

$$+ r^2\alpha^2\partial_r a\partial_r\alpha - ar^2\alpha^2\partial_r^2\alpha - a^2r^2\partial_t a\partial_t\alpha + a^2r^2\alpha\partial_t^2 a). \tag{63}$$

The Bianchi identity for the stress-energy tensor (58) implies the Klein-Gordon equation:

$$\Box\phi - \phi = 0, \tag{64}$$

where

$$\Box\phi = \frac{1}{\sqrt{-g}}\partial_\mu[\sqrt{-g}g^{\mu\nu}\partial_\nu\phi].$$

In order to numerically evolve the system we have to write out the PDE as a system of equations that only contain first-order derivatives. Thus we will be introducing new first-order variables $\psi = \partial_r\phi$ and $\pi = a\partial_t\phi/\alpha$. With these new variables, we can write Einstein's equations as

$$\frac{\partial_r a}{a} = \frac{1-a^2}{2r} + \frac{r}{4}[\psi^2 + \pi^2 + 2a^2V], \tag{65}$$

$$\frac{\partial_r \alpha}{\alpha} = \frac{\partial_r a}{a} + \frac{a^2-1}{r} - ra^2V, \tag{66}$$

$$\partial_t a = \frac{1}{2}ra\phi\pi, \tag{67}$$

where Eq. (65) is the Hamiltonian constraint, Eq. (66) the (r,r) component of Einstein's equations (also called the slicing condition), and Eq. (67) the momentum constraint. As this set of equations is overdetermined, we can choose to solve the first two equations and use the momentum constraint to monitor the accuracy of the calculations.

The Klein-Gordon equation for this system is a set of three equations:

$$\partial_t\phi = \frac{\alpha}{a}\pi, \tag{68}$$

$$\partial_t\pi = \frac{1}{r^2}\partial_r\left(\frac{r^2\alpha\psi}{a}\right) - a\alpha\phi, \tag{69}$$

$$\partial_t\psi = \partial_r\left(\frac{\alpha\pi}{a}\right), \tag{70}$$

The derivation of the Klein-Gordon equations can be found in Appendix A.2.

4.1 Initial data

The chosen initial profile for ϕ is the $\phi(0,r) = Ae^{-r^2/\sigma^2}$, where A is the amplitude of the scalar field, and its spatial derivative $\psi(0,r) = -\frac{2r}{\sigma^2}Ae^{-r^2/\sigma^2}$. We will be assuming time symmetry at initial time so that $\pi(0,r) = 0$. Knowing ϕ, ψ, and π at the initial time we can construct the initial values for a and α.

For a we will be assuming spatial flatness at the origin ($a|_{r=0} = 1, \partial_r a|_{r=0} = 0$) and evolving the data in accordance to Eq. (65) up to the outer edge of the numerical domain, denoted by r_N.

To evolve α, we assume that the spacetime is Schwarzschild-like at the boundary, and thus $\alpha_{r_N} = 1/a_{r_N}$. The Gaussian profile of our initial scalar field, which is centered at the origin, allows us to assume that $\partial_r \alpha|_{r_N} = 0$, as all field-related variables approach zero. Eq. 66 is then integrated inwards with the same assumption at the origin of $\partial_r \alpha|_{r=0} = 0$.

Both of these integrations, as well as every other integration onward, have been done using the fourth-order Runge-Kutta method mentioned in Section 3.3.

4.2 Evolving the data

To evolve the radial function a and lapse function α in time we first need to obtain new values for ϕ and its radial and temporal derivatives. The general evolution procedure is as follows:

1. input initial data for a and α;
2. use equations (68-70) to evolve the scalar field and its derivatives;
3. enforce boundary conditions to π and ψ;
4. evolve the Hamiltonian constraint (65) outwards assuming spatial flatness at the origin ($a|_{r=0} = 1, \partial_r a|_{r=0} = 0$));
5. evolve the slicing condition (66) from the edge of the numerical domain inward, assuming the spacetime to be Schwarzschild-like ($\alpha(r_N) = 1/a(r_N)$ and smooth ($\partial_r \alpha(r_N) = 0$) at the boundary;
6. use the new values of a and α to evolve the scalar field and its derivatives;
7. go to step (3).

The boundary condition mentioned in step (3) refers to Sommerfeld's outgoing radiation condition as derived in [18]. The reason boundary conditions are not being enforced for ϕ is due to its evolution equation Eq. 68 not containing any derivatives. There is precedent for enforcing the conditions regardless of this (as said by M.Bezares [19]), but since this wasn't done by Guzmán we have also decided against it.

4.3 Results

Using the algorithm mentioned in 4.2, we obtain the full evolutions of all variables and functions. In the interest of this thesis, we will be looking at the evolution of the metric functions a^2 and α^2, as well as the Ricci tensor and the evolution of the scalar field itself. We will be presenting two different sets of results: one, where the choice of parameters didn't cause the scalar field to collapse into a black hole, and one, where it did. Both sets of simulations had the outer radial boundary set at $r_N = 30$, the radial stepsize as $\Delta r = 0.01$, and the Courant factor as $\rho = 0.3$.

Oscillating solution

For the oscillating solution, we chose the parameters $A = 0.3$ and $\sigma = 5.35$, and the evolution took place for 100 time units. The results of the first simulation can be seen in Figures 4 and 5.

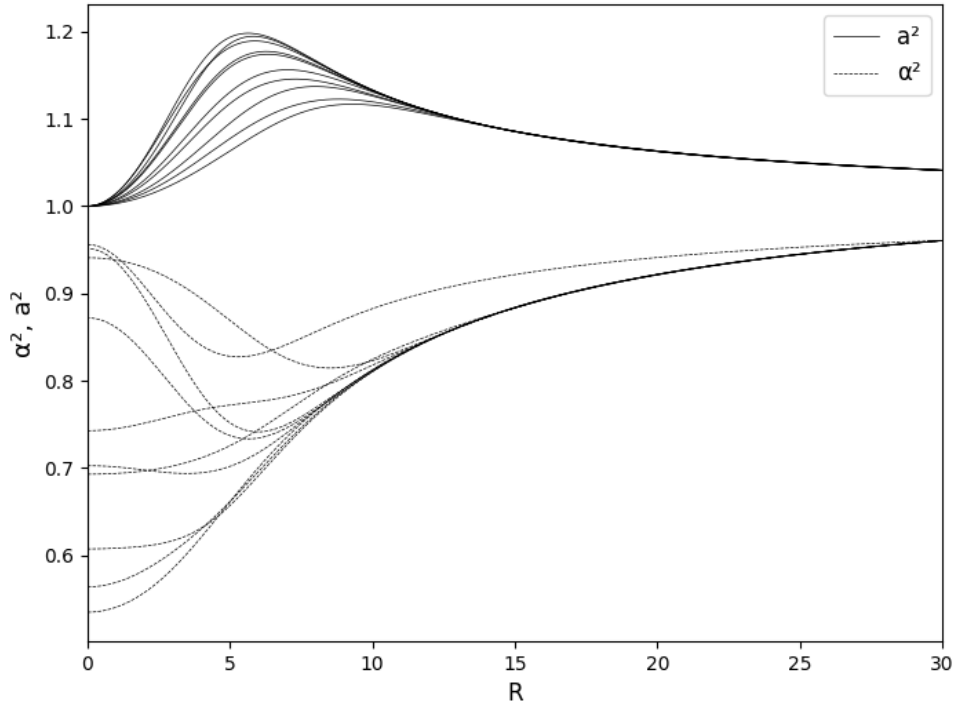

Figure 4. Snapshots of the metric functions a^2 and α^2 during the first half of the simulation.

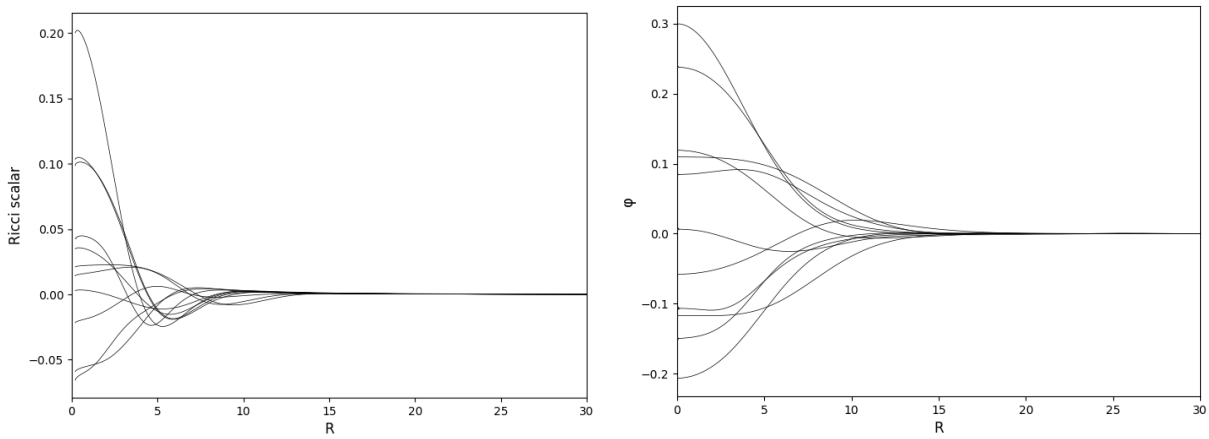

Figure 5. Snapshots of the Ricci scalar (left) and scalar field (right) during the first half of the simulation.

While it is not easy to see from the snapshots, all of the values oscillate for the entire duration of the simulation. As time passes, the oscillations dampen, and the magnitudes lessen. To avoid any visual confusion, the ending stages of this stabilizing behavior have been omitted from the figures.

While it is noteworthy to mention that the simulated behaviors match the example we aimed to recreate, the true test of accuracy lies in the convergence, which can be seen in Figure 6.

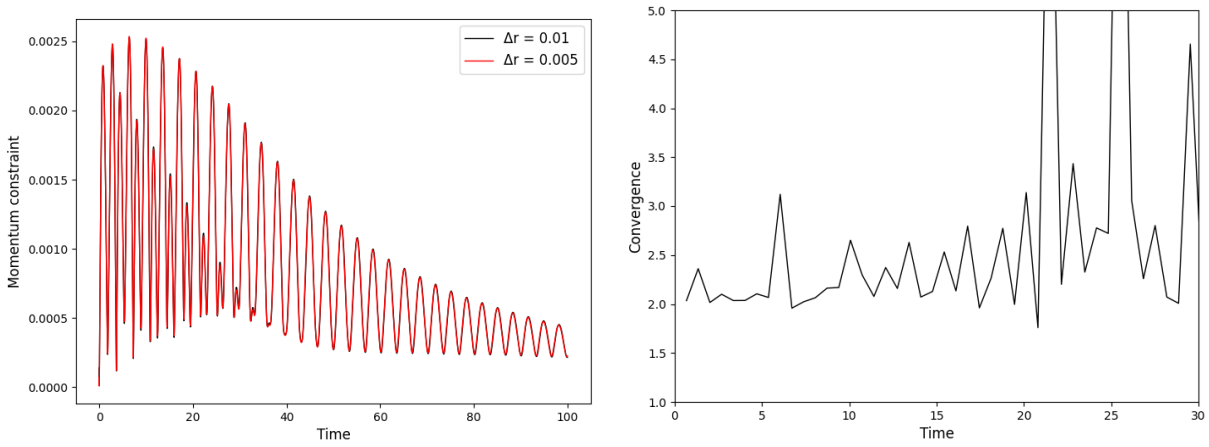

Figure 6. The momentum constraint norm for $\Delta r = 0.01$ and $\Delta r = 0.005$ (left) and convergence for $\Delta r_1 = 0.02$, $\Delta r_2 = 0.01$ and $\Delta r_3 = 0.005$ (right).

While the momentum constraint norm is quite small and grows smaller as the simulation goes on, there is virtually no difference between the momentum constraint norm of two different resolutions. This is not a great sign, as it suggests that the numerical error does not depend on the grid size. The convergence graph doesn't look any more promising, as our 4th-order accurate

numerical scheme does not converge at 4. What is interesting and should be mentioned, is that this pseudo-convergence at two holds true for every single evolved value and function.

Scalar field collapse

For the scalar field collapse, we chose the parameters $A = 0.4$ and $\sigma = 5.35$, and the evolution took place for 60 time units. The shorter runtime was chosen due to the simulation effectively breaking shortly after 60 units and returning only NaN values. The formation of a black hole also meant that many values around $r = 0$ increased substantially and overshadowed previous smaller values of the evolution.

The creation of a black hole is supported by the evolution of the lapse as seen in Figure 7

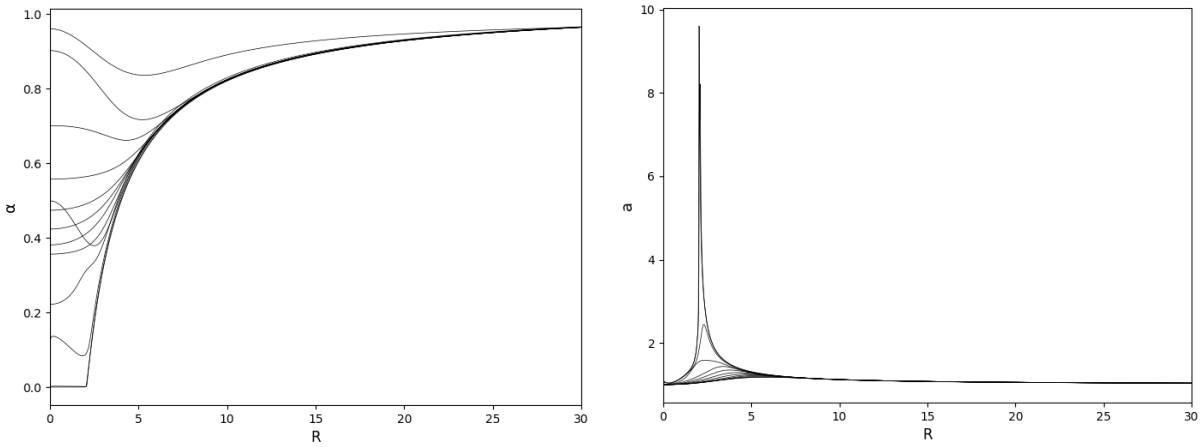

Figure 7. Evolution of the lapse function α (left) and a (right).

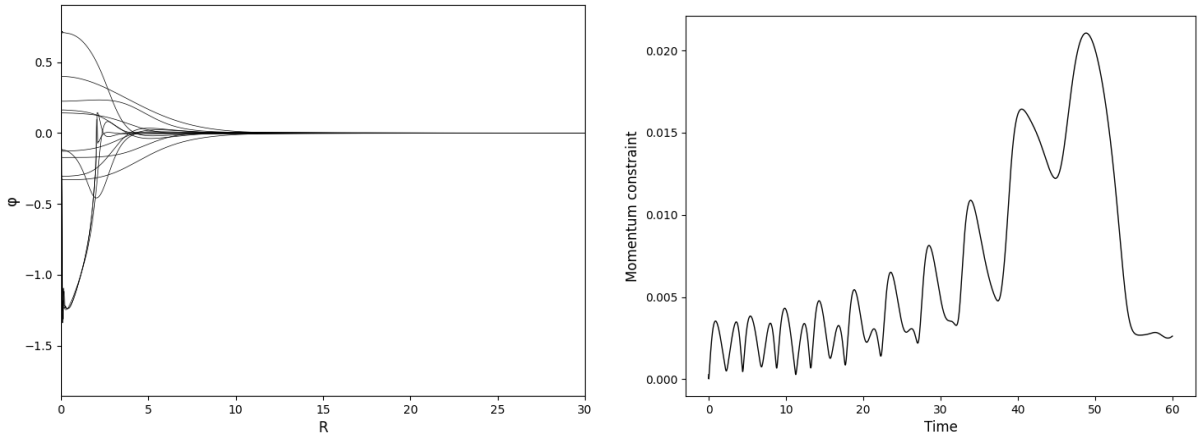

Figure 8. Evolution of the scalar field ϕ (left), and momentum constraint norm (right).

$\alpha = 0$ implies the appearance of an event horizon, where time is essentially frozen. The observation that the lapse is non-zero after $r = 2$ further solidifies the notion of scalar collapse

taking place, as the Schwarschild radius corresponds with that same location.

Figure 8 shows the behavior of the scalar field as it evolves on the left and the momentum constraint norm on the right. As we can see, the scalar field oscillates in both solutions, with the difference being, that for a scalar field collapse, the oscillations grow in magnitude and become erratic. As the system becomes more unstable, the momentum constraint norm grows in size. This has been explored in previous literature and is known to happen as the system approaches collapse.

5 Conclusion

In this thesis, we developed a numerical framework to simulate the evolution of a self-interacting scalar field in spherical symmetry. The framework consisted of both the theoretical decomposition of Einstein's equations and numerical methods, which were used to approximate differential operators. This framework was realized using the code as given in Appendix B.2.

The results of the simulation match that of the example we aimed to replicate. For the oscillating solution, one can observe similar behavior as presented by Guzmán in [12] for the metric variables and Ricci scalar. As the name suggests, all observed variables were observed to be oscillating throughout the entire evolution, with the oscillations dampening as time went on.

For the collapsing solution, only the lapse evolution can be compared with the literature, as no other results were presented. An important distinction between the two simulations is the behavior of the oscillations. For the oscillating solution, the amplitude decreases and, given enough time, approaches an almost flat spacetime. On the other hand, for the scalar field collapse, the system becomes very unstable, causing the amplitudes to grow, which further increases the instability.

This work provides a strong foundation for any future work ahead, as any complex problems can be solved using the methods that were discussed. It is our hope to apply this numerical framework to the teleparallel equivalent of GR, for which the groundwork has been done in [20].

There remained the issue of the lack of convergence, which hasn't yet been solved at the time of submitting the thesis, but we hope to fix it in the near future.

References

[1] Clifford M. Will. “The Confrontation between General Relativity and Experiment”. In: *Living Reviews in Relativity* 17.1 (Dec. 2014), p. 4. ISSN: 1433-8351. DOI: 10.12942/lrr-2014-4.

[2] N. T. Roseveare. *Mercury's Perihelion from Le Verrier to Einstein*. Oxford: Oxford University Press, 1982. ISBN: 0-19-858174-2.

[3] Albrecht Giese. “Ongoing problems with special and general relativity, and solutions”. In: *Journal of Physics: Conference Series* 1251.1 (June 2019). Publisher: IOP Publishing, p. 012017. DOI: 10.1088/1742-6596/1251/1/012017.

[4] The Event Horizon Telescope Collaboration et al. “First M87 Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole”. In: *The Astrophysical Journal Letters* 875.1 (Apr. 2019). Publisher: The American Astronomical Society, p. L1. DOI: 10.3847/2041-8213/ab0ec7.

[5] LIGO Scientific Collaboration and Virgo Collaboration et al. “Observation of Gravitational Waves from a Binary Black Hole Merger”. In: *Physical Review Letters* 116.6 (Feb. 2016). Publisher: American Physical Society, p. 061102. DOI: 10.1103/PhysRevLett.116.061102.

[6] S. Shankaranarayanan and Joseph P. Johnson. “Modified theories of Gravity: Why, How and What?” In: *General Relativity and Gravitation* 54.5 (May 2022). arXiv:2204.06533 [astro-ph, physics:gr-qc, physics:hep-th], p. 44. ISSN: 0001-7701, 1572-9532. DOI: 10.1007/s10714-022-02927-2.

[7] Salvatore Capozziello et al. “The $3+1$ Formalism in the Geometric Trinity of Gravity”. en. In: *The European Physical Journal C* 81.12 (Dec. 2021). arXiv:2108.03075 [astro-ph, physics:gr-qc], p. 1141. ISSN: 1434-6044, 1434-6052. DOI: 10.1140/epjc/s10052-021-09944-6. URL: http://arxiv.org/abs/2108.03075.

[8] Sean M. Carroll. *Spacetime and Geometry: An Introduction to General Relativity*. Cambridge University Press, 2019.

[9] Jeremy Atkins. “The Free Klein Gordon Field Theory”. en. In: (Apr. 2018).

[10] Miguel Alcubierre. *Introduction to 3+1 Numerical Relativity*. Oxford University Press, Apr. 2008. ISBN: 978-0-19-920567-7. DOI: 10.1093/acprof:oso/9780199205677.001.0001.

[11] Heinz-Otto Kreiss and Jens Lorenz. *Initial-Boundary Value Problems and the Navier-Stokes Equations*. en. Society for Industrial and Applied Mathematics, Jan. 2004. ISBN: 978-0-89871-913-0. DOI: 10.1137/1.9780898719130. URL: http://epubs.siam.org/doi/book/10.1137/1.9780898719130.

[12] Francisco S. Guzmán. "Introduction to numerical relativity through examples". In: *Revista Mexicana De Fisica* 53 (2007), pp. 78–93.

[13] P. D. Lax and R. D. Richtmyer. "Survey of the stability of linear finite difference equations". en. In: *Communications on Pure and Applied Mathematics* 9.2 (May 1956), pp. 267–293. ISSN: 0010-3640, 1097-0312. DOI: 10.1002/cpa.3160090206.

[14] Nicola Franchini et al. "Fixing the dynamical evolution in scalar-Gauss-Bonnet gravity". In: *Physical Review D* 106.6 (Sept. 2022). arXiv:2206.00014 [gr-qc], p. 064061. ISSN: 2470-0010, 2470-0029. DOI: 10.1103/PhysRevD.106.064061.

[15] Carles Bona, Carlos Palenzuela-Luque, and Carles Bona-Casas. *Elements of Numerical Relativity and Relativistic Hydrodynamics: From Einstein' s Equations to Astrophysical Simulations*. en. Vol. 783. Lecture Notes in Physics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN: 978-3-642-01164-1. DOI: 10.1007/978-3-642-01164-1.

[16] Musa H., Ibrahim Saidu, and Marianus Waziri. "A Simplified Derivation and Analysis of Fourth Order Runge Kutta Method". In: *International Journal of Computer Applications* 9 (Nov. 2010). DOI: 10.5120/1402-1891.

[17] Gioel Calabrese et al. "Summation by parts and dissipation for domains with excised regions". en. In: *Classical and Quantum Gravity* 21.24 (Dec. 2004). arXiv:gr-qc/0308007, pp. 5735–5757. ISSN: 0264-9381, 1361-6382. DOI: 10.1088/0264-9381/21/24/004.

[18] Justin Lloyd Ripley. "General relativity and its classical modification in gravitational collapse". en. In: (Sept. 2020).

[19] Miguel Bezares. *Private conversations and email exchanges*.

[20] Maria-Jose Guzman. *The Hamiltonian constraint in the symmetric teleparallel equivalent of general relativity*. arXiv:2311.01424 [gr-qc, physics:hep-th]. Apr. 2024. DOI: 10.48550/arXiv.2311.01424. URL: http://arxiv.org/abs/2311.01424.

Appendix

A Derivations

A.1 Deriving Einstein's equation

To derive the equations of motion for GR, we use the relation

$$G_{\mu\nu} = \kappa T_{\mu\nu} = R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu}.$$

The formula for the Einstein field equations was given in [12], and the Ricci tensor and scalar have been calculated from the metric and compiled into the Einstein tensor using Mathematica. The only four components that contribute to the equations of motion are as follows:

$$\begin{aligned} G_{tt} &= \frac{\alpha^2}{a^3 r^2} \left(a(a^2 - 1) + 2r\partial_r a \right) \\ G_{rr} &= \frac{1}{\alpha r^2} \left(\alpha(1 - a^2) + 2r\partial_r \alpha \right) \\ G_{tr} &= G_{rt} = \frac{2\partial_t a}{ar}. \end{aligned}$$

The Hamiltonian constraint is derived from the tt component of the Einstein tensor and field equation as thus:

$$\begin{aligned} T_{tt} &= (\partial_t \phi)^2 + \frac{1}{2}\alpha^2 \left[g^{\alpha\alpha} (\partial_\alpha \phi)^2 + 2V \right] \\ &= (\partial_t \phi)^2 + \frac{1}{2}\alpha^2 \left[g^{tt} (\partial_t \phi)^2 + g^{rr} (\partial_r \phi)^2 + 2V \right] \\ &= (\partial_t \phi)^2 + \frac{1}{2}\alpha^2 \left[-\frac{1}{\alpha^2} (\partial_t \phi)^2 + \frac{1}{a^2} (\partial_r \phi)^2 + 2V \right] \\ &= (\partial_t \phi)^2 - \frac{1}{2} (\partial_t \phi)^2 + \frac{\alpha^2}{2a^2} (\partial_r \phi)^2 + \alpha^2 V \\ &= \frac{1}{2} (\partial_t \phi)^2 + \frac{\alpha^2}{2a^2} (\partial_r \phi)^2 + \alpha^2 V \\ &= \frac{1}{2} \frac{\alpha^2}{a^2} \pi^2 + \frac{\alpha^2}{2a^2} \psi^2 + \alpha^2 V \\ &= \frac{\alpha^2}{2a^2} \left[\pi^2 + \psi^2 + 2a^2 V \right]. \end{aligned}$$

Now we use the relation between geometry and matter:

$$
\begin{aligned}
G_{tt} = \frac{\alpha^2}{a^3 r^2}\left(a(a^2-1)+2r\partial_r a\right) &= \frac{\alpha^2}{2a^2}\left[\pi^2+\psi^2+2a^2V\right] \quad |\cdot\frac{a^2}{\alpha^2}\\
\frac{a(a^2-1)+2r\partial_r a}{ar^2} &= \frac{1}{2}\left[\pi^2+\psi^2+2a^2V\right]\\
\frac{a^2-1}{r^2}+\frac{2\partial_r a}{ar} &= \frac{1}{2}\left[\pi^2+\psi^2+2a^2V\right]\\
\frac{2\partial_r a}{ar} &= \frac{1-a^2}{r^2}+\frac{1}{2}\left[\pi^2+\psi^2+2a^2V\right] \quad |\cdot\frac{r}{2}\\
\frac{\partial_r a}{a} &= \frac{1-a^2}{2r}+\frac{r}{4}\left[\pi^2+\psi^2+2a^2V\right].
\end{aligned}
$$

A similar procedure is done with the rr components:

$$
\begin{aligned}
T_{rr} &= (\partial_r\phi)^2-\frac{1}{2}g_{rr}\left[g^{\alpha\alpha}(\partial_\alpha\phi)^2+2V\right]\\
&= \psi^2-\frac{1}{2}a^2\left[-\frac{1}{\alpha^2}(\partial_t\phi)^2+\frac{1}{a^2}(\partial_r\phi)^2+2V\right]\\
&= \psi^2-\frac{1}{2}a^2\left[-\frac{1}{\alpha^2}\frac{\alpha^2}{a^2}\pi^2+\frac{1}{a^2}\psi^2+2V\right]\\
&= \psi^2-\frac{1}{2}a^2\frac{1}{a^2}\psi^2-\frac{1}{2}a^2\left[-\frac{1}{a^2}\pi^2+2V\right]\\
&= \frac{1}{2}\psi^2+\frac{1}{2}\pi^2-a^2V = \frac{1}{2}\psi^2+\frac{1}{2}\pi^2+a^2V-2a^2V\\
&= \frac{1}{2}\left[\psi^2+\pi^2+2a^2V\right]-2a^2V,
\end{aligned}
$$

$$
\begin{aligned}
G_{rr} = \frac{1}{\alpha r^2}\left(\alpha(1-a^2)+2r\partial_r\alpha\right) &= \frac{1}{2}\left[\psi^2+\pi^2+2a^2V\right]-2a^2V\\
\frac{\alpha(1-a^2)}{\alpha r^2}+\frac{2r\partial_r\alpha}{\alpha r^2} &= \frac{1}{2}\left[\psi^2+\pi^2+2a^2V\right]-2a^2V\\
\frac{2\partial_r\alpha}{\alpha r} &= \frac{a^2-1}{r^2}+\frac{1}{2}\left[\psi^2+\pi^2+2a^2V\right]-2a^2V \quad |\cdot\frac{r}{2}\\
\frac{\partial_r\alpha}{\alpha} &= \frac{a^2-1}{2r}+\frac{r}{4}\left[\psi^2+\pi^2+2a^2V\right]-ra^2V\\
\frac{\partial_r\alpha}{\alpha} &= \frac{a^2-1}{r}+\frac{1-a^2}{2r}+\frac{r}{4}\left[\psi^2+\pi^2+2a^2V\right]-ra^2V\\
\frac{\partial_r\alpha}{\alpha} &= \frac{\partial_r a}{a}+\frac{a^2-1}{r}-ra^2V,
\end{aligned}
$$

and the mixed components:

$$
\begin{aligned}
T_{tr} = \partial_t\phi\partial_r\phi = T_{rt} &= \frac{\alpha}{a}\pi\psi, \\
G_{tr} = \frac{2\partial_t a}{ar} &= \frac{\alpha}{a}\pi\psi \quad | \cdot \frac{ar}{2} \\
\partial_t a &= \frac{1}{2}r\alpha\psi\pi
\end{aligned}
$$

A.2 Deriving Klein-Gordon equations

Writing out the D'Alambertian gives us

$$
\begin{aligned}
\Box\phi &= \frac{1}{\sqrt{-g}}\partial_\mu[\sqrt{-g}g^{\mu\nu}\partial_\nu\phi] \\
&= \frac{1}{\alpha ar^2\sin\theta}\partial_t[\alpha ar^2\sin\theta g^{t\nu}\partial_\nu\phi] + \frac{1}{\alpha ar^2\sin\theta}\partial_i[\alpha ar^2\sin\theta g^{i\nu}\partial_\nu\phi].
\end{aligned}
$$

Since our selected metric only contains elements on the main diagonal, we can ignore the components where $\mu \neq \nu$

$$
\begin{aligned}
\Box\phi &= \frac{1}{\alpha ar^2\sin\theta}\partial_t[\alpha ar^2\sin\theta g^{tt}\partial_t\phi] + \frac{1}{\alpha ar^2\sin\theta}\partial_i[\alpha ar^2\sin\theta g^{ii}\partial_i\phi] \\
&= \frac{1}{\alpha ar^2\sin\theta}\partial_t\left[\alpha ar^2\sin\theta\left(-\frac{1}{\alpha^2}\right)\partial_t\phi\right] + \frac{1}{\alpha ar^2\sin\theta}\partial_r\left[\alpha ar^2\sin\theta\frac{1}{a^2}\partial_r\phi\right] \\
&+ \frac{1}{\alpha ar^2\sin\theta}\partial_\theta\left[\alpha ar^2\sin\theta\left(\frac{1}{r^2}\right)\partial_\theta\phi\right] + \frac{1}{\alpha ar^2\sin\theta}\partial_\varphi\left[\alpha ar^2\sin\theta\frac{1}{r^2\sin^2\theta}\partial_\varphi\phi\right].
\end{aligned}
$$

We have assumed spherical symmetry so that derivatives with respect to θ and φ will be zero. This also means that $\sin\theta$ is a constant and can therefore be taken out of the derivatives, resulting in it being reduced from the equation altogether:

$$
\begin{aligned}
&= \frac{1}{\alpha ar^2}\partial_t\left[\alpha ar^2\left(-\frac{1}{\alpha^2}\right)\partial_t\phi\right] + \frac{1}{\alpha ar^2}\partial_r\left[\alpha ar^2\frac{1}{a^2}\partial_r\phi\right] \\
&= \frac{1}{\alpha ar^2}\left[-r^2\partial_t\left[\frac{a}{\alpha}\partial_t\phi\right] + \partial_r\left[\frac{\alpha r^2}{a}\partial_r\phi\right]\right].
\end{aligned}
$$

Substituting in $\pi = a\partial_t\phi/\alpha$ and $\psi = \partial_r\phi$ we finally obtain

$$
\Box\phi = \frac{1}{\alpha ar^2}\left[-r^2\partial_t\pi + \partial_r\left[\frac{\alpha r^2}{a}\psi\right]\right].
$$

We can now compute the Klein-Gordon equation

$$\begin{aligned}
\Box\phi - \phi &= 0 \\
\frac{1}{\alpha a r^2}\left[-r^2\partial_t\pi + \partial_r\left[\frac{\alpha r^2}{a}\psi\right]\right] &= \phi \quad |\cdot \alpha a \\
\frac{1}{r^2}\left[-r^2\partial_t\pi + \partial_r\left[\frac{\alpha r^2}{a}\psi\right]\right] &= \alpha a\phi \\
-\partial_t\pi + \frac{1}{r^2}\partial_r\left[\frac{\alpha r^2}{a}\psi\right] &= \alpha a\phi \\
\partial_t\pi &= \frac{1}{r^2}\partial_r\left[\frac{\alpha r^2}{a}\psi\right] - \alpha a\phi.
\end{aligned}$$

We've obtained the evolution equation for π. The evolution equation for ϕ is simply the definition of π rewritten to be

$$\partial_t\phi = \frac{\alpha}{a}\pi.$$

The evolution equation for ψ is just the radial derivative of $\partial_t\phi$:

$$\partial_t\psi = \partial_r(\partial_t\phi) = \partial_r\left(\frac{\alpha}{a}\pi\right).$$

B Code

B.1 Advection equation

```
import numpy as np
import matplotlib.pyplot as plt

#Defining parameters

Nx = 4000                          # num of grid points
xmin = -2                          # spatial domain limits
xmax = 2
#discretized mesh and mesh size
X, dx = np.linspace(xmin, xmax, Nx+1, retstep=True)
c = 1                              # advection speed
tmin=0                             # starting time
tmax=4                             # final time
CFL = 0.8
```

```
dt = CFL*dx/abs(c)
Nt=int((tmax-tmin)/dt)            # num of time steps
T = np.linspace(tmin, tmax, Nt+1)

X = X[:-1] # remove last point, as u(x=-1,t)=u(x=0,t)

sigma=0.01

# initial function
def initial_u(x):
    return np.exp(-x**2 /sigma)

# each value of the U array contains the solution for all x values
    at each timestep
U = np.zeros((Nt+1,Nx),dtype=float) #accounts for time t=0
U[0] = u = initial_u(X)

def derivatives(t,u,c,dx):
    du = np.zeros(len(u))
    p = -c/(2*dx)
    du[0] = p*(u[1]-u[-1])
    du[1:-1] = p*(u[2:]-u[:-2])
    du[-1] = p*(u[0]-u[-2])
    return du

for k in range(Nt):              #Runge-Kutta method
    t = T[k]
    k1 = derivatives(t,u,c,dx)*dt
    k2 = derivatives(t+0.5*dt,u+0.5*k1,c,dx)*dt
    k3 = derivatives(t+0.5*dt,u+0.5*k2,c,dx)*dt
    k4 = derivatives(t+dt,u+k3,c,dx)*dt
    U[k+1] = u = u + (k1+2*k2+2*k3+k4)/6
```

B.2 Self-interacting scalar field

```
import numpy as np
import matplotlib.pyplot as plt
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
```

```
from numba import jit

A=0.3          #for BH is 0.4
m=1
sigma=5.35

sgm=0.01

r0=0
r_max=30
Nr=3000
R, dr=np.linspace(r0, r_max, Nr+1, retstep=True)

tmin=0
tmax=100          #for BH run for 60
CFL = 0.3
dt=CFL*dr
Nt=int((tmax-tmin)/dt)
T=np.linspace(tmin, tmax, Nt+1)

@jit
def S(a, r):
    if r ==0:
        return 0
    else:
        return (a*(1-a**2))/(2*r) + A**2 * np.exp(-2*r**2/sigma**2)
            *a*r/4 * (4*r**2 / sigma**4 + a**2 * m**2)
@jit
def derivative_alpha(alpha, r, a, da):
    if r ==0:
        return 0
    else:
        return alpha*(da/a + (a**2-1)/r -r * a**2 * 0.5 * m**2 *A
            **2 * np.exp(-2*r**2/sigma**2))
@jit
def initial_a():
    a=[1]
    dra=[0]
    for i in range(0, Nr):
```

```
        m1=S(a[-1], R[i])*dr
        m2=S(a[-1] + 0.5*m1, R[i]+0.5*dr)*dr
        m3=S(a[-1] + 0.5*m2, R[i]+0.5*dr)*dr
        m4=S(a[-1] + m3, R[i]+dr)*dr
        dra.append((m1+2*m2+2*m3+m4)/(dr*6))
        a_next=a[-1]+(m1+2*m2+2*m3+m4)/6
        a.append(a_next)
    return a, dra

@jit
def initial_alpha(r):
    revrad=r[::-1] # reversed radius from 30 to 0
    rev_a = initialA[::-1]
    rev_da=a_derivatives[::-1]

    alpha=[1/initialA[-1]]
    alpha_step=[]
    for i in range(0, Nr):
        m1=derivative_alpha(alpha[-1], revrad[i], rev_a[i], rev_da[
            i])*dr
        m2=derivative_alpha(alpha[-1] + 0.5*m1, revrad[i]-0.5*dr, (
            rev_a[i]+rev_a[i+1])/2, (rev_da[i]+rev_da[i+1])/2)*dr
        m3=derivative_alpha(alpha[-1] + 0.5*m2, revrad[i]-0.5*dr, (
            rev_a[i]+rev_a[i+1])/2, (rev_da[i]+rev_da[i+1])/2)*dr
        m4=derivative_alpha(alpha[-1] + m3, revrad[i]-dr, rev_a[i
            +1], rev_da[i+1])*dr
        d_alpha=(m1+2*m2+2*m3+m4)/6
        alpha_next=alpha[-1]-d_alpha
        alpha.append(alpha_next)
        alpha_step.append(d_alpha/dr)
    alpha_step.append(0)
    reversed_alpha=alpha[::-1]
    reversed_step=alpha_step[::-1]
    return reversed_alpha, reversed_step



#recalling initial a and alpha values
```

```
initialA, dra = initial_a()
initialA=np.asarray(initialA)
a_derivatives=np.asarray(dra)

alpha, alpha_step=initial_alpha(R)
initial_Alpha=np.asarray(alpha)
alpha_step=np.array(alpha_step)
alpha_squared=initial_Alpha**2

#Initial data evolution derivatives
@jit
def a_derivative(a, r, phi, pi, psi):
    if r ==0:
        return 0
    else:
        return a*((1-a**2)/(2*r) + r/4 * (psi**2 + pi**2 + a**2 * m
            **2 * phi**2))
@jit
def alpha_derivative(alpha, r, a, da, phi):
    if r ==0:
        return 0
    else:
        return alpha*(da/a + (a**2-1)/r - 0.5 * r * a**2 * m**2 *
            phi**2)

#Initial data RK
@jit
def a_RK(r, phi, pi, psi, dphi, dpi, dpsi):
    a=[1]
    derivatives=[0]
    for i in range(Nr):
        k1=a_derivative(a[i], r[i], phi[i], pi[i], psi[i])*dr
        k2=a_derivative(a[i] + 0.5*k1, r[i] + 0.5*dr, phi[i] + 0.5*
            dphi[i], pi[i] + 0.5*dpi[i], psi[i] + 0.5*dpsi[i])*dr
        k3=a_derivative(a[i] + 0.5*k2, r[i] + 0.5*dr, phi[i] + 0.5*
            dphi[i], pi[i] + 0.5*dpi[i], psi[i] + 0.5*dpsi[i])*dr
        k4=a_derivative(a[i] + k3, r[i] + dr, phi[i] + dphi[i], pi[
            i] + dpi[i], psi[i] + dpsi[i])*dr
        RK=(k1 + 2*k2 + 2*k3 + k4)/6
```

```
        derivatives.append(RK)
        a.append(a[i] +RK)

    return a, derivatives
@jit
def alpha_RK(a, da, r, phi, dphi):
    alpha = [1 / a[-1]]
    dalpha = [0]

    for i in range(Nr - 1, -1, -1):  # Iterate from Nr to 0
        k1 = alpha_derivative(alpha[-1], r[i], a[i], da[i], phi[i])
             * dr
        k2 = alpha_derivative(alpha[-1] + 0.5 * k1, r[i] - 0.5 * dr
            , a[i] + 0.5 * da[i], (da[i] + da[i + 1]) / 2, phi[i] +
            0.5 * dphi[i]) * dr
        k3 = alpha_derivative(alpha[-1] + 0.5 * k2, r[i] - 0.5 * dr
            , a[i] + 0.5 * da[i], (da[i] + da[i + 1]) / 2, phi[i] +
            0.5 * dphi[i]) * dr
        k4 = alpha_derivative(alpha[-1] + k3, r[i - 1], a[i + 1],
            da[i + 1], phi[i + 1]) * dr
        RK = (k1 + 2 * k2 + 2 * k3 + k4) / 6
        alpha.append(alpha[-1] - RK)
        dalpha.append(RK / dr)

    alpha = alpha[::-1]  # Reverse the final result
    derivatives = dalpha[::-1]
    return np.array(alpha), np.array(derivatives)

#Initial values for t=0

def initial_phi(r):
    return A*np.exp(-r**2/sigma**2)

def initial_psi(r):
    return -r*2 / sigma**2 * A * np.exp(-r**2/sigma**2)

Phi=np.array(initial_phi(R))

Psi=np.array(initial_psi(R))
```

```
Pi=np.zeros(len(R))

#Equations (26) - (28) in Guzman's article

def phi_derivative(alpha, a, pi):
    return alpha/a * pi

def pi_derivative(a, r, alpha, psi, phi):
    free_member=a*alpha*phi

    dr3=np.zeros(len(Pi))
    p=1/(2*dr)
    dr3[0]=0
    dr3[1:-1] = 1/r[1:-1]**2 * p*(r[2:]**2*alpha[2:] *psi[2:]/a[2:]
        - r[:-2]**2*alpha[:-2] *psi[:-2]/a[:-2])
    dr3[-1] = 1/r[-1]**2 * 2*p*(- r[-2]**2*alpha[-2] *psi[-2]/a[-2]
        + r[-1]**2*alpha[-1] *psi[-1]/a[-1])
    dr3 -= free_member
    return dr3

def psi_derivative(alpha, a, pi):
    p=1/(2*dr)
    dpsi=np.zeros(len(Psi))
    dpsi[0]=2*p*(alpha[1]*pi[1]/a[1] - alpha[0]*pi[0]/a[0])
    dpsi[1:-1]=p*(alpha[2:]*pi[2:]/a[2:] - alpha[:-2]*pi[:-2]/a
        [:-2])
    dpsi[-1]=-2*p*(alpha[-2]*pi[-2]/a[-2] - alpha[-1]*pi[-1]/a[-1])
    return dpsi

@jit
def dissipation(u):
    q=np.zeros(len(u))
    q[0] = sgm*(u[3]-3*u[2]+3*u[1]-u[0])*48/(59*64*dr)
    q[1] = sgm*(u[4]-6*u[3] + 12*u[2]-10*u[1] + 3*u[0])*48/(43*64*
        dr)
    q[2] = sgm*(u[5]-6*u[4] + 15*u[3]-19*u[2] + 12*u[1] - 3*u[0])
        *48/(49*64*dr)
```

```
    q[-1] = sgm*(u[-4]-3*u[-3]+3*u[-2]-u[-1])*48/(59*64*dr)
    q[-2] = sgm*(u[-5]-6*u[-4] + 12*u[-3]-10*u[-2] + 3*u[-1])
        *48/(43*64*dr)
    q[-3] = sgm*(u[-6]-6*u[-5] + 15*u[-4]-19*u[-3] + 12*u[-2] - 3*u
        [-1])*48/(49*64*dr)

    q[3:-3] = sgm*(u[6:] - 6*u[5:-1] + 15*u[4:-2] - 20*u[3:-3] +
        15*u[2:-4] - 6*u[1:-5] + u[:-6])/(64*dr)

    return q


#Runge-Kutta for each function

def RK(a, r, alpha, phi, pi, psi, da, dalpha):
    da=np.asarray(da)
    #k is for phi, m for pi and n for psi
    #dissipation goes as qxyz
    k1=phi_derivative(alpha, a, pi)*dt
    m1=pi_derivative(a, r, alpha, psi, phi)*dt
    n1=psi_derivative(alpha, a, pi)*dt

    q1=dissipation(phi)
    q2=dissipation(pi)
    q3=dissipation(psi)
    k1= k1+q1
    m1=m1+q2
    n1=n1+q3

    k2=phi_derivative(alpha + 0.5*dalpha, a + 0.5*da, pi + 0.5*m1)*
        dt
    m2=pi_derivative(a + 0.5*da, r + 0.5*dr, alpha + 0.5*dalpha,
        psi + 0.5*n1, phi + 0.5*k1)*dt
    n2=psi_derivative(alpha + 0.5*dalpha, a + 0.5*da, pi + 0.5*m1)*
        dt

    x1=dissipation(phi+0.5*k1)
    x2=dissipation(pi+0.5*m1)
    x3=dissipation(psi+0.5*m1)
```

```
    k2= k2+x1
    m2=m2+x2
    n2=n2+x3

    k3=phi_derivative(alpha + 0.5*dalpha, a + 0.5*da, pi + 0.5*m2)*
        dt
    m3=pi_derivative(a + 0.5*da, r + 0.5*dr, alpha + 0.5*dalpha,
        psi + 0.5*n2, phi + 0.5*k2)*dt
    n3=psi_derivative(alpha + 0.5*dalpha, a + 0.5*da, pi + 0.5*m2)*
        dt

    y1=dissipation(phi+0.5*k2)
    y2=dissipation(pi+0.5*m2)
    y3=dissipation(psi+0.5*n2)
    k3=k3+y1
    m3=m3+y2
    n3=n3+y3



    k4=phi_derivative(alpha + dalpha, a + da, pi + m3)*dt
    m4=pi_derivative(a + da, r + dr, alpha + dalpha, psi + n3, phi
        + k3)*dt
    n4=psi_derivative(alpha + dalpha, a + da, pi + m3)*dt

    z1=dissipation(phi+k3)
    z2=dissipation(pi+m3)
    z3=dissipation(psi+n3)
    k4=k4+z1
    m4=m4+z2
    n4=n4+z3

    dphi=(k1 + 2*k2 + 2*k3 + k4)/6
    dpi=(m1 + 2*m2 + 2*m3 + m4)/6
    dpsi=(n1 + 2*n2 + 2*n3 + n4)/6

    return dphi, dpi, dpsi

#Ricci scalar:
def ricci_scalar(r, a, alpha, dra, dralpha, dta, ddta, dtalpha):
```

```
    ddralpha=np.zeros(len(dralpha))
    ddralpha[0]=(dralpha[1] - dralpha[0])/dr
    ddralpha[1:-1] = (dralpha[2:] - dralpha[:-2])/(2*dr)
    ddralpha[-1] = (dralpha[-2] - dralpha[-1])/dr

    multiplier=2/(r**2 * a**3 * alpha**3)
    first_term = - a*alpha**3
    second_term = a**3 * alpha**3
    third_term = 2*r*alpha**3 * dra
    fourth_term = - 2*a*r*alpha**2*dralpha
    fifth_term = r**2 * alpha**2 * dra * dralpha
    sixth_term= - a*r**2 * ddralpha
    seventh_term = - a**2 * r**2 * dta *dtalpha
    eigth_term = a**2 * r**2 * alpha * ddta
    ricci= multiplier*(first_term + second_term + third_term +
        fourth_term + fifth_term + sixth_term + seventh_term +
        eigth_term)
    ricci[0] = np.nan
    return ricci


plotted_a= initialA
A_current= initialA
ricci_trace= np.ones(len(R))

plotted_alpha=initial_Alpha
Alpha = initial_Alpha #redefining name
plot_phi = Phi

momentum_constraint_norm=[np.nan]
dta = np.zeros(len(R))
for i in range(Nt):
    momentum_constraint=0.5*R*Alpha*Psi*Pi
    dphi, dpi, dpsi = RK(A_current, R, Alpha, Phi, Pi, Psi,
        a_derivatives, alpha_step)
    Phi+=dphi
    Pi += dpi
    Pi[-1]=-dt/R[-1] * (R[-1]*Pi[-1] - R[-2]*Pi[-2])/dr +Pi[-1]-dpi
        [-1]
```

```
        Psi += dpsi
        Psi[-1]= -Pi[-1] - Phi[-1]/R[-1]
        a_new, dra= a_RK(R, Phi, Pi, Psi, dphi, dpi, dpsi)
        a_new=np.asarray(a_new)
        a_derivatives=np.array(dra)
        dta2= a_new - A_current #current loop time diff, dta previous
            loop
        normarray=dta2-momentum_constraint
        normarray=normarray**2
        norm=sum(normarray)/len(normarray)
        squared_norm=np.sqrt(norm)
        momentum_constraint_norm.append(squared_norm)
        alpha_new, dralpha = alpha_RK(a_new, a_derivatives, R, Phi,
            dphi)
        Ricci_scalar = ricci_scalar(R, a_new, alpha_new, a_derivatives,
             dralpha, dta2, dta2-dta, alpha_new-Alpha)
        if i%50 == 0:
            print(i)
            plotted_a = np.vstack((plotted_a, a_new))
            plotted_alpha = np.vstack((plotted_alpha, alpha_new))
            ricci_trace = np.vstack((ricci_trace, Ricci_scalar))
            plot_phi=np.vstack((plot_phi, Phi))
        dta=dta2
        A_current=a_new
        Alpha=np.array(alpha_new)
        alpha_step=np.array(dralpha)

np.save('plot_a_array.npy', plotted_a)
np.save('plot_alpha_array.npy', plotted_alpha)
np.save('momentum_constraint.npy', momentum_constraint_norm)
np.save('phi_evolution.npy', plot_phi)
np.save('ricci_scalar.npy', ricci_trace)
```

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Marie Femke Jaarma**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Numerical simulation of a self-interacting scalar field in spherical symmetry,

   supervised by María José Guzmán Monsalve, PhD.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Marie Femke Jaarma

31/05/2024