The Einstein Equations in Spherical Symmetry: Scalar Field Accretion onto a Black Hole

Jonathan Thornburg

Max Planck Institut für Gravitationsphysik, Albert Einstein Institut,
Am Mühlenberg 1, D-14476 Golm, Germany.

(Dated: Id: eesf.tex,v 1.52 2003/09/10 22:21:46 jthorn Exp)

This document describes how to set up the Einstein equations using the $3 + 1$ formalism in spherical symmetry. Our spacetime is asymptotically flat, and contains a black hole at the origin (which is excised from the numerical grid), surrounded by (typically) one or more shells of a self-gravitating scalar field. We use straightforward 2nd order finite differencing, with a method of lines scheme for the time integration.

You should have a basic familiarity with general relativity, the $3 + 1$ formalism, free versus constrained evolution schemes, basic finite differencing, and convergence tests.

By working through the computer exercises here, you will gain practice with a number of important concepts in numerical relativity, including construction of initial data, monitoring of the constraints, coordinate choice, black hole excision, radiation boundary conditions, local and global mass functions, apparent horizons, matter accretion onto a black hole, convergence tests, the method of lines integration of time-dependent partial differential equations (PDEs), the Courant-Friedrichs-Lewy (CFL) stability limit, one-sided and upwind finite differencing, and computer visualization/animation of time-dependent results.

I. INTRODUCTION

In this course we will study a model problem in numerical general relativity: the Einstein equations together with a self-gravitating massless scalar field, moving in the spacetime surrounding a black hole. You will write (portions of) a computer program to take as input a set of field variables describing the state of the system at some initial time, and evolve this forward in time to determine the state of the system at later times.

To keep this writeup to a reasonable length, we have relgated the construction of suitable initial data to a separate writeup. We have supplied a sample program to construct the initial data, and you can study it later if you're interested.

You can work in any of C, Fortran 77, or Fortran 90. We have supplied several sample programs in each of these languages, each broken down into a main driver routine and a set of relatively-independent subroutines. The **computer exercises** will have you write your own versions of the various subroutines to replace the sample ones. You will probably only have time to do this for a few of the subroutines, but by following the interface of the sample subroutines, you can combine ones you've rewritten with the sample ones, to obtain complete running programs. There are also more ambitious **problems** (for which we have not supplied sample solutions) if you want to explore further on your own.

II. CONTINUUM PHYSICS

Our formulation of the problem, and our presentation here, are adapted from [1]. Another key reference in this area is [2].

A. Notation

We use indices abc for spacetime (4-)tensors, and indices $ijkl$ for 3-tensors defined solely within a slice. We use the Penrose abstract-index notation, so (for example) g_{ab} and g_{ij} mean the 4-metric tensor and 3-metric tensor respectively, not individual components. We use indices $r\theta\phi$ for components. Spacetime tensors are marked by a ${}^{(4)}$ prefix; all other tensors are 3-tensors defined within a slice.

B. 3 + 1 Einstein Equations

We use the usual $3+1$ formalism ([3]), with spacetime coordinates $x^a = (t, x^i)$, and a line element

$$ {}^{(4)}ds^2 \equiv {}^{(4)}g_{ab}\, dx^a\, dx^b = -(\alpha^2 - \beta_i\beta^i)\, dt^2 + 2\beta_i\, dt\, dx^i + g_{ij}\, dx^i\, dx^j \tag{1} $$

where α is the $3+1$ **lapse function**, β^i the $3+1$ **shift vector**, and g_{ij} the **3-metric**. We use g_{ij} to raise and lower 3-tensor indices.

Given g_{ij}, we define the usual 3-covariant derivative operator ∇_i and (3-) Ricci tensor and scalar R_{ij} and $R \equiv R_i{}^i$. Taking n^a to be the future pointing timelike unit normal to the slices, we define the usual (3-) extrinsic curvature $K_{ij} \equiv \frac{1}{2}\pounds_n g_{ij} \equiv -\nabla_i n_j$.

We decompose the spacetime stress-energy tensor $T_{ab} \equiv {}^{(4)}T_{ab}$ into the $3+1$ variables the **energy density** $\rho \equiv n^a n^b T_{ab}$, the **momentum density** $j^i \equiv -n_a T^{ai}$, and the **stress tensor** $T_{ij} \equiv {}^{(3)}T_{ij} = {}^{(4)}T_{ij}$ (that is, the 3-tensor T_{ij} is equal to the spatial components of the 4-tensor T_{ab}).

In terms of these $3+1$ variables, the 10 Einstein equations $G_{ab} = 8\pi T_{ab}$ split up into 4 **constraint equations** which don't contain any time derivatives

$$ C \equiv \left(R - K_{ij}K^{ij} + K^2\right) - \left(16\pi\rho\right) \;=\; 0 \tag{2a} $$

$$ C^i \equiv \left(\nabla_j K^{ij} - \nabla^i K\right) - \left(8\pi j^i\right) \;=\; 0 \tag{2b} $$

and 6 **evolution equations** giving the time dependence of the field variables,

$$ \partial_t g_{ij} \;=\; -2\alpha K_{ij} + \pounds_\beta g_{ij} \tag{3a} $$

$$ \begin{aligned} \partial_t K_{ij} \;=\; & -\nabla_i\nabla_j\alpha + \alpha\left(R_{ij} - 2g^{kl}K_{ik}K_{jl} + KK_{ij}\right) + \pounds_\beta K_{ij} \\ & + 4\pi\alpha\left((T-\rho)g_{ij} - 2T_{ij}\right) \end{aligned} \tag{3b} $$

where the **Lie derivative** terms are given by

$$ \pounds_\beta g_{ij} \;=\; \nabla_i\beta_j + \nabla_j\beta_i \tag{4a} $$

$$ \;=\; \beta^k\partial_k g_{ij} + g_{ik}\partial_j\beta^k + g_{jk}\partial_i\beta^k \tag{4b} $$

$$ \pounds_\beta K_{ij} \;=\; \beta^k\nabla_k K_{ij} + K_{ik}\nabla_j\beta^k + K_{jk}\nabla_i\beta^k \tag{4c} $$

$$ \;=\; \beta^k\partial_k K_{ij} + K_{ik}\partial_j\beta^k + K_{jk}\partial_i\beta^k \tag{4d} $$

(We keep the Lie derivative terms in the evolution separate because we'll need to treat them specially in the finite differencing; this is discussed in section III C.)

C. Free versus Constrained Evolutions

Notice that the combined system of the evolution equations (3) and the constraints (2) are *overdetermined*: assuming that the lapse function α and shift vector β^i are somehow chosen, there are more equations (16) than there are variables (the 12 g_{ij} and K_{ij}). This is not a problem mathematically, because the constraints are **consistent** with the evolution equations: the Bianchi identities imply that if the constraints (2) are satisfied on the initial slice, they will be satisfied on all future slices.

However, in constructing a computational algorithm things are not so simple: because there are more equations than variables, we must somehow choose a subset of the equations to update the variables as we step forward from one time slice to the next. Different choices here may have very different computational properties. In particular, there are two basic approaches:

free evolution
: The simplest scheme is to use only the evolution equations (3) to update the field variables during the evolution, using the constraints (2) only as diagnostics of the evolution's accuracy. This is the approach we take here.

constrained evolution
: Another possibility is to use some or all (say N_c), of the 4 constraint equations (2), together with $12 - N_c$ of the evolution equations (3), to update the field variables during the evolution, using the remaining constraint and evolution equations only as diagnostics of the evolution's accuracy. Note that since the constraint equations (2) are elliptic PDEs, whereas the evolution equations (3) are **"hyperbolic"** (they have hyperbolic-like causality properties, even if they're not actually hyperbolic in the strict mathematical sense), a constrained evolution scheme requires quite different boundary conditions from a free evolution scheme. We will see the importance of this later when we discuss black hole excision in section II H.

In an actual numerical evolution, numerical errors will cause the constraints (2) to not be exactly satisfied; we can monitor their (nonzero) values to get an indication of the evolution's accuracy. This immediately raises the question, how small should we expect these nonzero values to be? To determine a scale, observe that the left hand sides in (2) are each the sums of several terms, which should cancel to give zero sums. This suggests that a suitable scale is given by the sizes of the individual left-hand-side terms, so we define the **relative constraints** (also known as the **normalized constraints**)

$$C_{\rm rel} \equiv \frac{R - K_{ij}K^{ij} + K^2 - 16\pi\rho}{|R| + |K_{ij}K^{ij}| + |K^2| + |16\pi\rho|} \tag{5a}$$

$$C^i_{\rm rel} \equiv \frac{\nabla_j K^{ij} - \nabla^i K - 8\pi j^i}{|\nabla_j K^{ij}| + |\nabla^i K| + |8\pi j^i|} \tag{5b}$$

These should be $\ll 1$ everywhere in spacetime.

D. Scalar Field

We take the scalar field ϕ to satisfy the (4-)scalar wave equation $\Box\phi \equiv \nabla_a \nabla^a \phi = 0$, and to have the usual stress-energy tensor

$$4\pi T_{ab} = (\partial_a \phi)(\partial_b \phi) - \tfrac{1}{2} g_{ab} (\partial_c \phi)(\partial^c \phi) \tag{6}$$

For numerical purposes it's convenient to work instead with the $3+1$ field variables

$$P_i = \nabla_i \phi \tag{7a}$$
$$Q = \frac{1}{\alpha} \left(\partial_t \phi - \beta^i \nabla_i \phi \right) \tag{7b}$$

so that

$$\partial_t \phi = \alpha Q + \beta^k P_k \tag{8}$$

If ϕ itself is needed it's easy to recover it from P_i and Q by time-integrating (8).

Straightforward but tedious calculations then give the scalar field evolution equations as

$$\partial_t P_i = \nabla_i \left(\alpha Q + \beta^k P_k \right) \tag{9a}$$
$$\partial_t Q = \nabla_i \left(\alpha P^i \right) + \alpha K Q + \beta^i \nabla_i Q \tag{9b}$$

and the $3+1$ scalar field variables as

$$4\pi\rho = \tfrac{1}{2} \left(P_k P^k + Q^2 \right) \tag{10a}$$
$$4\pi j_i = -P_i Q \tag{10b}$$
$$4\pi T_{ij} = P_i P_j + \tfrac{1}{2} g_{ij} \left(-P_k P^k + Q^2 \right) \tag{10c}$$
$$4\pi T = -\tfrac{1}{2} P_k P^k + \tfrac{3}{2} Q^2 \tag{10d}$$

ρ and j^r are *volume* densities, but in practice it's often more convenient to integrate over the angular dimensions, using the radial densities $4\pi g_{\theta\theta}\rho$ and $4\pi g_{\theta\theta} j^r$ as matter diagnostics. (This also gets rid of a number of 4π factors in the equations.)

E. Spherical Symmetry

We now assume that spacetime is spherically symmetric, and that the spatial coordinates are $x^i = (r, \theta, \varphi)$, with the usual polar spherical topology. We parameterize the various $3+1$

tensors as follows:[1]

$$g_{ij} \equiv \begin{bmatrix} A & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B\sin^2\theta \end{bmatrix} \tag{11a}$$

$$K_{ij} \equiv \begin{bmatrix} X & 0 & 0 \\ 0 & Y & 0 \\ 0 & 0 & Y\sin^2\theta \end{bmatrix} \tag{11b}$$

$$\beta^i \equiv \begin{bmatrix} \beta & 0 & 0 \end{bmatrix} \tag{11c}$$

$$P_i \equiv \begin{bmatrix} P & 0 & 0 \end{bmatrix} \tag{11d}$$

$$C^i_{\rm rel} \equiv \begin{bmatrix} C^r_{\rm rel} & 0 & 0 \end{bmatrix} \tag{11e}$$

With these assumptions it's then straightforward, though tedious, to express all the other $3+1$ variables and equations in terms of the **state variables** A, B, X, Y, P, and Q:

The individual terms in the constraints (2) and (5) become

$$R = -2\frac{\partial_{rr}B}{AB} + \tfrac{1}{2}\frac{(\partial_r B)^2}{AB^2} + \frac{(\partial_r A)(\partial_r B)}{A^2 B} + \frac{2}{B} \tag{12a}$$

$$K_{ij}K^{ij} = \frac{X^2}{A^2} + 2\frac{Y^2}{B^2} \tag{12b}$$

$$K = \frac{X}{A} + 2\frac{Y}{B} \tag{12c}$$

$$\nabla_j K^{rj} = -\frac{X\partial_r A}{A^3} + \frac{X\partial_r B}{A^2 B} - \frac{Y\partial_r B}{AB^2} + \frac{\partial_r X}{A^2} \tag{12d}$$

$$\nabla^r K = -\frac{X\partial_r A}{A^3} - 2\frac{Y\partial_r B}{AB^2} + \frac{\partial_r X}{A^2} + 2\frac{\partial_r Y}{AB} \tag{12e}$$

$$4\pi\rho = \frac{1}{2}\left(\frac{P^2}{A} + Q^2\right) \tag{12f}$$

$$4\pi j^r = -\frac{PQ}{A} \tag{12g}$$

---

[1] Notice that for historical reasons, we do *not* factor out the flat-space r^2 dependence from $g_{\theta\theta} \equiv B$. It might well be that factoring out this r^2 factor (i.e. using $g_{\theta\theta}/r^2$ as our field variable for finite differencing) would give more accurate results. The interested student is invited to try this...

The evolution equations (3) become

$$\begin{aligned}
\partial_t A &= -2\alpha X + \pounds_\beta A && \text{(13a)}\\
\partial_t B &= -2\alpha Y + \pounds_\beta B && \text{(13b)}\\
\partial_t X &= -\partial_{rr}\alpha + \tfrac{1}{2}(\partial_r\alpha)\frac{\partial_r A}{A} \\
&\quad -\alpha\frac{\partial_{rr}B}{B} + \tfrac{1}{2}\alpha\left(\frac{\partial_r B}{B}\right)^2 + \tfrac{1}{2}\alpha\frac{(\partial_r A)(\partial_r B)}{AB} + 2\alpha\frac{XY}{B} - \alpha\frac{X^2}{A} \\
&\quad + \pounds_\beta X - 2\alpha P^2 && \text{(13c)}\\
\partial_t Y &= -\tfrac{1}{2}(\partial_r\alpha)\frac{\partial_r B}{A} \\
&\quad -\tfrac{1}{2}\alpha\frac{\partial_{rr}B}{A} + \tfrac{1}{4}\alpha\frac{(\partial_r A)(\partial_r B)}{A^2} + \alpha + \alpha\frac{XY}{A} \\
&\quad + \pounds_\beta Y && \text{(13d)}\\
\partial_t P &= (\partial_r\alpha)Q + \alpha\partial_r Q \\
&\quad + \pounds_\beta P && \text{(13e)}\\
\partial_t Q &= (\partial_r\alpha)\frac{P}{A} - \tfrac{1}{2}\alpha\frac{\partial_r A}{A}\frac{P}{A} + \alpha\frac{\partial_r B}{B}\frac{P}{A} + \alpha\frac{\partial_r P}{A} \\
&\quad + \alpha\frac{X}{A}Q + 2\alpha\frac{Y}{B}Q \\
&\quad + \pounds_\beta Q && \text{(13f)}
\end{aligned}$$

where the Lie derivative terms (4) are given by

$$\begin{aligned}
\pounds_\beta A &= \beta\partial_r A + 2(\partial_r\beta)A && \text{(14a)}\\
\pounds_\beta B &= \beta\partial_r B && \text{(14b)}\\
\pounds_\beta X &= \beta\partial_r X + 2(\partial_r\beta)X && \text{(14c)}\\
\pounds_\beta Y &= \beta\partial_r Y && \text{(14d)}\\
\pounds_\beta P &= \beta\partial_r P + (\partial_r\beta)P && \text{(14e)}\\
\pounds_\beta Q &= \beta\partial_r Q && \text{(14f)}
\end{aligned}$$

F. Diagnostics

1. *Light Cones*

It's useful to compute the light cones in terms of our field variables. In particular, it's useful to know, at a given event, which infinitesimal displacements (dr, dt) are timelike. From the line element (1) and our definitions for the 3-metric, lapse, and shift-vector, it's easy to see that this is given by

$$\text{timelike} \quad \Leftrightarrow \quad c_- \equiv -\beta - \frac{\alpha}{\sqrt{A}} \le \frac{dr}{dt} \le -\beta + \frac{\alpha}{\sqrt{A}} \equiv c_+ \qquad (15)$$

Problem 1 (Event Horizons)
Suppose we have an evolution covering all of our (spherically symmetric) spacetime. Given the light-cone information (15) (again assume here that this is available everywhere in spacetime), how would you go about finding the event horizon? How could you find a good approximation to this given only an evolution covering a finite region of spacetime? This topic is discussed in [4–6].

3. Misner-Sharp Mass Function

As first shown by Misner and Sharp ([7]), in spherical symmetry we can define a *global* mass function giving the mass contained within a given radius, and which obeys a work-energy conservation law (the first law of thermodynamics).[2] In terms of our field variables, this can be written as as

$$m_{\rm MS} = \tfrac{1}{2}\sqrt{B}\left(1 - \tfrac{1}{4}\frac{(\partial_r B)^2}{AB} + \frac{Y^2}{B}\right) \tag{16}$$

Unfortunately, the definition (16) turns out to be quite sensitive to small numerical errors at large r.[3] As shown by [9], the Misner-Sharp mass can be rewritten in an alternate form as the volume integral of a (positive definite) local mass density μ:

$$m_\mu = \int \mu \, d^3x \tag{17}$$

or (since we excise a black hole at the origin of our numerical grid)

$$m_\mu(r) = m_{\rm MS}(r_{\rm min}) + \int_{r_{\rm min}}^{r} 4\pi B\sqrt{A}\,\mu \, dr \tag{18a}$$

where the local mass density μ can be written as

$$\mu = \tfrac{1}{2}\frac{\partial_r B}{\sqrt{AB}}\rho - \frac{\sqrt{A}}{\sqrt{B}} Y j^r \tag{18b}$$

Another useful quantity is the total mass within the outer grid boundary, which we call the **slice mass** $m_{\rm slice}$. This is bounded above by the total mass of the slice, the **Arnowitt-Deser-Misner** (**ADM**) mass $m_{\rm ADM}$, but $m_{\rm slice} < m_{\rm ADM}$ if any scalar field has propagated off the outer boundary of the grid.

---

[2] Unfortunately, the Misner-Sharp definition works only in spherical symmetry, and can't be extended to more general spacetimes. See [8,section 23.5] for a good discussion of this, and of the Misner-Sharp mass function in general.

[3] Since $B \equiv g_{\theta\theta} = O(r^2)$ at large r, the leading term in (16) is $O(r)$ for large r. For our slices $m_{\rm MS}$ is always $O(1)$, so the remaining terms in (16) (must) nearly cancel the leading term. This cancellation causes the numerical sensitivity.

We assume that each slice contains a black hole at the origin. Given a reasonable slicing, this implies the existence of one or more apparent horizons in each slice. In terms of the $3+1$ variables, a **marginally trapped surface** (**MTS**) is defined by the condition ([10])

$$\Theta \equiv \nabla_i n^i + K_{ij} n^i n^j - K = 0 \tag{19}$$

where we take n^i to be the outward-pointing unit 3-normal to the horizon. An **apparent horizon** (**AH**) is then an MTS which is not contained inside any other MTS. In practice, we follow the usual (sloppy) convention in numerical relativity and blur the distinction between MTSs and AHs: we refer to any MTS as an AH.

In general, solving the apparent horizon equation (19) is quite difficult, but in spherical symmetry it's much easier: the apparent horizon equation becomes

$$\Theta \equiv \frac{\partial_r B}{\sqrt{A}\, B} - 2\frac{Y}{B} = 0 \tag{20}$$

and it suffices to just look for zero crossings of $\Theta = \Theta(r)$.

For our coordinates an apparent horizon at a radius $r_{\rm AH}$ always has an **apparent horizon mass** (the Misner-Sharp mass function $m_{\rm MS}$ interpolated to the apparent horizon radius) given by $m_{\rm AH} = \frac{1}{2} m_{\rm AH}$.

G. Coordinate Choice

So far we have left the coordinate choice – specified in the $3+1$ formalism by the lapse function α and the shift vector β^i – arbitrary.

To choose our slicing, we impose the Eddington-Finkelstein condition ([8,box 31.2]) that $t+r$ be an ingoing null coordinate. A straightforward calculation shows that this is equivalent to the condition

$$\frac{\alpha}{\sqrt{A}} + \beta = 1 \tag{21}$$

To choose our spatial coordinates, we maintain the areas of constant-r surfaces (coordinate spheres) temporally constant during the evolution, i.e. $\partial_t g_{\theta\theta} = 0$ or

$$\partial_t B = 0 \tag{22}$$

By virtue of the evolution equation (13b) and Lie derivative (14b), this condition is equivalent to

$$-2\alpha Y + \beta \partial_r B = 0 \tag{23}$$

Notice that since the spatial coordinate condition gives $\partial_t B = 0$, we could omit the time evolution of B, leaving it at its initial value throughout an evolution. However, in practice evolving B through the full evolution equation (13b) (as we do here) adds little extra complexity, and makes the code more modular and easier to generalize to other possible coordinate conditions.

Note that although our spatial coordinates are chosen to maintain the areas of constant-r surfaces (coordinate spheres) temporally constant during the evolution, there's no requirement or assumption that that area be $4\pi r^2$, i.e. that r be an **areal** radial coordinate. Rather, this is determined by the coordinates on the initial slice.

H. Black Hole Excision; Inner Boundary Conditions

.

To avoid having to deal with the $r = 0$ coordinate singularity of our (r, θ, φ) polar spherical spatial coordinates, and to add to the physical interest, we assume that there is a black hole at the origin, and we **excise** it from the numerical grid. That is, we use a numerical grid which only covers the portion $r \in [r_{\min}, r_{\max}]$ of each slice, where $r_{\min} > 0$ is chosen to be inside the black hole, but safely away from $r = 0$.

Because there are no elliptic equations in our evolution system, only "hyperbolic" ones (cf. our discussion of free versus constrained evolution systems in section II C), so long as the light cones point only *inward* at $r = r_{\min}$, i.e. so long as we have $dr/dt < 0$ for any null or timelike motion there, we don't need any additional boundary conditions on the excision boundary $r = r_{\min}$ to keep the evolution well-posed (causal). Given our definitions (15), this inner-boundary causality condition becomes

$$c_+ < 0 \tag{24}$$

The statement that no additional boundary conditions are needed at the inner boundary is true for the continuum PDEs, but numerically things are a bit more complicated. We discuss numerical inner boundary conditions in section III B.

I. Outer Boundary Conditions

Since our numerical grid only extends out to a finite outer radius $r = r_{\max}$, we need outer boundary conditions there to make our system of equations well-posed. Our goal here is to choose "outgoing radiation" boundary conditions such that the dynamics of our finite-domain system reasonably approximate those of the corresponding region of an infinite-domain isolated system. That is, we would like there to be no incoming signals at $r = r_{\max}$, and we would like any outgoing signals at $r = r_{\max}$ to propagate smoothly outwards off the edge of the grid. Due to the nonlinearity of the Einstein equations this can't be done exactly at any finite $r_{\max}$, but there are several possibilities for how to approximate it:

1. Static Boundary Conditions

One very simple outer boundary condition would be to just keep the state vector static (time-independent) there, i.e. all its time derivatives to zero:

$$\partial_t u = 0 \qquad \text{for each state-vector variable } u \tag{25}$$

If B is being kept constant (i.e. if it's being evolved by (22)), then the boundary condition (25) is fine for it. Otherwise, or for any of the other field variables, this boundary condition has two serious problems:

- If the fields near the outer boundary have any time dependence, this will reflect off of the outer boundary, causing spurious signals to propagate back in towards smaller radia.

- This reflection will almost certainly violate the constraints, so a wave of constraint violation will propagate inwards from the outer boundary at (roughly) the speed of light. Once the constraints are significantly violated, we no longer have a good approximation to the Einstein equations.

2. Flat-Space Outgoing Radiation Boundary Conditions

A rough approximation to outgoing-radiation conditions is to apply a flat-space outgoing radiation (Sommerfeld) condition to each component of the state vector. That is, we require (assume) that each field variable u must be of the form $f_u(r - c_+ t)$ near the boundary, so that $\partial_t u = -c_+ \partial_r u$ there. This gives the outgoing-radiation boundary conditions

$$\partial_t u = -c_+ \partial_r u \qquad \text{for each state-vector variable } u \text{ (except possibly } B) \tag{26}$$

3. Background Subtraction

There are many possible improvements to the outgoing-radiation boundary conditions (26). Here we consider several improvements:

background subtraction
: One easy improvement is to apply the boundary conditions only to the deviations of the field variables from their values in some (time-independent) background spacetime, for example Schwarzschild spacetime.

choice of background
: Our derivation below assumes that the background is time-independent. However, our slices actually have time-dependent masses (decreasing as scalar field leaves the grid at the outer boundary), so no single Schwarzschild spacetime is a good choice for the background throughout an entire evolution. Instead, it's preferable to, for each slice (say with slice mass m_{slice}), use as a background a Schwarzschild slice of mass m_{slice}. (This violates our assumption of a time-dependent background, but in practice the variable background works better than the fixed one so long as m_{slice} varies relatively slowly.)

$1/r^n$ **falloffs**
: Because of the spherical topology, we expect the dynamic variations in the field variables to have amplitudes which fall off as various analytically-calculable powers of r along outgoing null cones.

Combining both these improvements, we assume that each state-vector field variable (except possibly B) satisfies

$$\frac{u - u_{\text{bg}}}{r^m} = \frac{f(r - c_+ t)}{r^n} \tag{27}$$

at the outer boundary, where u_{bg} is the (time-independent) value of u (at the same radius) in the background spacetime, and m and n are suitable parameters (which may be vary from one field variable to another). This gives the outgoing-radiation boundary condition

$$\partial_t (u - u_{\text{bg}}) = -c_+ \left[\partial_r (u - u_{\text{bg}}) + \frac{n - m}{r} (u - u_{\text{bg}}) \right] \tag{28}$$

Field Variable	m	n
A	0	2
B	2	1
X	0	3
Y	0	2
P	0	1
Q	0	1

TABLE I: This table gives the parameters m and n used for the evolution outer boundary condition (28).

To determine suitable m and n we need to know the actual large-r falloff behavior of perturbations in the field variables. This can either be calculated analytically (via perturbations around a background solution), or found by numerical tests (using an outer boundary so far out that it doesn't influence the results). Table I gives a set of parameters m and n for our state-vector variables which seem to work fairly well.

Problem 2 (Experiment with Different Falloff Conditions)
You could try experimenting with different values of m and n to see if you can get better boundary conditions (less reflection of outgoing signals off the boundary, and/or less constraint violations generated at the outer boundary and propagating back inwards).

J. Summary

At the continuum level, our computational problem is thus as follows:

- The state of our system at any time t is given by the fields A, B, X, Y, P, and Q, all of which are functions of the radius r in the range $r \in [r_{\rm min}, r_{\rm max}]$.
- The input to our problem will be the state fields at some initial time $t_{\rm init}$.
- Our goal will be to evolve the fields forward in time to compute them for some time interval $t \in [t_{\rm init}, t_{\rm final}]$.
- Given the values of all the fields at a given time, we can compute the lapse function α and shift vector β by solving the two linear equations (21) and (23) at each point.
- We can then evolve the fields using the evolution equations (13) and (14), and the outer boundary conditions (28).
- To monitor the evolution's accuracy, we can compute the relative constraints C and C^r using (5) and (12).
- To help understand the physics, we can compute the local energy and momentum densities $4\pi B\rho$ and $4\pi B j^r$ using (12f) and (12g), the Misner-Sharp mass function using (16) and the positions of the apparent horizon(s) by zero-finding on Θ computed by (20).

III. NUMERICAL METHODS

Because many of our equations are nonlinear, it's not practical to solve them analytically, so we use numerical methods instead. In particular, we use 2nd order finite differencing and the method of lines (MOL) to time-integrate the evolution equations (13) and (14) and the outer boundary conditions (28). Appendix B gives a brief introduction to MOL.

A. Spatial Grid and Centered Finite Differencing

We begin by discretizing the spatial coordinate r: we choose a (fixed) inner boundary position $r_{\min} > 0$ and a (fixed and uniform) grid spacing Δr, and introduce a finite grid of $N+1$ points $r_k = r_{\min} + k\,\Delta r$, $k = 0,\ 1,\ 2,\ \ldots,\ N$, spanning the range from the inner boundary $r = r_{\min}$ to the outer boundary $r = r_{\max} = r_{\min} + N\,\Delta r$.[4] We use the usual notation $u_k \equiv u(r = r_k)$ for any grid function u.

We approximate all spatial derivatives by finite difference operators. Notably, we will use the usual centered 2nd order finite difference approximations

$$(\partial_r u)_k = \frac{u_{k+1} - u_{k-1}}{2\,\Delta r} + O\big((\Delta r)^2\big) \tag{29a}$$

$$(\partial_{rr} u)_k = \frac{u_{k-1} - 2u_k + u_{k+1}}{(\Delta r)^2} + O\big((\Delta r)^2\big) \tag{29b}$$

for most spatial derivatives.

B. Boundary Conditions

Clearly the centered finite difference approximations (29) can't be used as is at either the innermost or the outermost grid point ($k = 0$ or $k = N$ respectively), since this would require values of the grid function at the nonexistent grid points $k = -1$ or $k = N+1$ respectively. There are two plausible ways to solve this problem:

- We could use one-sided finite difference approximations such as

$$(\partial_r u)_k = \frac{-3u_k + 4u_{k+1} - u_{k+2}}{2\,\Delta r} + O\big((\Delta r)^2\big) \tag{30a}$$

  (which is usable at the innermost grid point $k = 0$), and/or

$$(\partial_r u)_k = \frac{3u_k - 4u_{k-1} + u_{k-2}}{2\,\Delta r} + O\big((\Delta r)^2\big) \tag{30b}$$

  (which is usable at the outermost grid point $k = N$).[5]

---

[4] It would be preferable (more accurate results per unit of computation) to use a nonuniformly spaced grid, but this complicates the programming, so we only cover the uniformly-space case here. See problem 4 for a brief discussion of how you might extend the sample program to use a nonuniform grid.

[5] Similar one-sided approximations to $\partial_{rr} u$ also exist (though they're less accurate), but it turns out we won't need them here.

- Alternatively, we could actually have "**ghost zone**" grid points $k = -1$ and $k = N+1$, determining the field variables at these points by *extrapolation* from the nominal grid:

$$u_{-1} = 3u_0 - 3u_1 + u_2 + O\big((\Delta r)^3\big) \tag{31a}$$

$$u_{N+1} = 3u_N - 3u_{N-1} + u_{N-2} + O\big((\Delta r)^3\big) \tag{31b}$$

Both approaches work fairly well. In fact, they're equivalent: convolving the centered finite difference approximations (29) with the extrapolation operators (31) gives precisely the one-sided finite difference approximations (30). Notice, however, that for 2nd derivatives the combination of the extrapolation operators (31) with the centered 2nd order finite difference approximation (29b) is only 1st order accurate.

In practice, though, it's probably more convenient to use the extrapolation technique, because this way the code for the Einstein equations can be written without making the boundary points a special case. This will become clearer in the numerical exercises below.

We refer to the grid points $0 \le k \le N$ as the **nominal** grid, the grid points $1 \le k \le N-1$ (i.e. all the nominal grid points except the boundary points $k = 0$ and $k = N$) as the **interior** grid, and the grid points $-1 \le k \le N+1$ as the **ghosted** grid. We consider the grid functions A, B, X, Y, P, and Q on the nominal grid to be the state vector; we take any other grid functions, and any values in the ghost zones, to be only temporary values, not part of the state vector. Also, unless otherwise stated, all the subroutines in the numerical exercises should use only the nominal-grid parts of their inputs, and need compute their outputs only on the nominal grid.

C. Upwind Finite Differencing

It turns out that if we use the centered finite difference approximations (29) for all the terms in the evolution equations in the interior grid, the time evolution is highly unstable. In particular, the problem lies in our use of the centered finite difference approximation (29a) for the first derivatives (∂_r) in the Lie derivatives (14). Moreover, it turns out that we can solve the problem – and obtain stable evolutions – by using a different finite difference approximation for these derivatives, depending on the sign of the shift vector β at each grid point:

$$\left.\begin{array}{l} \beta > 0 \Rightarrow \text{use the one-sided approximation (30a)} \\ \beta = 0 \Rightarrow \text{use the centered approximation (29a)} \\ \beta < 0 \Rightarrow \text{use the one-sided approximation (30b)} \end{array}\right\} \tag{32}$$

The shift vector β acts rather like a fluid velocity, and we speak of the Lie derivative terms as **advection** terms since they adjust the equations for the fluid motion. In this model, the one-sided approximations are called **upwind** derivatives, since they're one-sided in the "upwind" direction of the fluid flow.

IV. COMPUTER EXERCISES

In order to do convergence tests, when you do the computer exercises you should try running the resulting programs for several different resolutions; it's convenient if these are

in a 1:2:4 ratio of resolutions. A good set of grid parameters to start with is $r_{\rm min} = 1.5$ and $r_{\rm max} = 101.5$, with resolutions $\Delta r = 0.04$ ($N = 2500$), $\Delta r = 0.02$ ($N = 5000$), and $\Delta r = 0.01$ ($N = 10000$). For good 2nd order convergence you may also need $\Delta r = 0.005$ ($N = 20000$).

Computer Exercise 1 (Initial Data)
This exercise has two parts:

Schwarzschild initial data

Using the supplied template file `Schwarzschild.{c,f77,f90}`, write a subroutine `Schwarzschild` to set the state vector (on the nominal grid) to the Schwarzschild analytic values (A1).

Using the appropriate test driver file `setup_Schwarzschild.{c,f77,f90}`, together with your `Schwarzschild` subroutine, generate initial data files for a unit-mass Schwarzschild slice.

Scalar-Field Initial Data Use the sample initial data solver program to construct initial data files containing a black hole surrounded by a scalar field shell. Use a unit-mass Schwarzschild slice for the initial slice, and construct initial data (for each resolution) for two choices of perturbations:

- A Gaussian perturbation in P, with center $r_{\rm init} = 20$, standard deviation $\sigma_{\rm init} = 5$, and amplitude $A = 0.5$. This will produce a relatively thick and moderate-mass scalar field shell. For historical reasons (following the naming conventions of [1, 11]), we call this the **pqw5** initial data family.
- a Gaussian perturbation in P, with center $r_{\rm init} = 20$, standard deviation $\sigma_{\rm init} = 1$, and amplitude $A = 0.1$. This will produce a much thinner and more massive scalar field shell. We call this the **pqw1** initial data family.

Computer Exercise 2 (Basic Infrastructure)
This exercise has several parts, which can be done in any order:

check that the Metric is Positive Definite

Using the supplied template file `check_metric.{c,f77,f90}`, write a subroutine `check_metric` which takes the state vector (defined for the nominal grid) as an input, and checks that the metric is positive definite, i.e. that $A > 0$ and $B > 0$ everywhere in the grid. If so, your subroutine should do nothing (i.e. just return), if not, your subroutine should print a suitable error message and halt execution.

compute algebraic functions of the state vector

Using the supplied template file `trace_K.{c,f77,f90}`, write a subroutine `trace_K` which takes the state vector (defined for the nominal grid) as an input, and computes K as defined by (12c) (at the nominal grid points) as an output.

Using the supplied template file `energy_momentum_density.{c,f77,f90}`, write a subroutine `energy_momentum_density` which takes the state vector (defined for the nominal grid) as an input, and computes $4\pi\rho$ and $4\pi j^r$ as defined by (12f) and (12g), and also the more-useful-in-practice diagnostics $4\pi B\rho$ and $4\pi B J^r$, (all at the nominal grid points) as outputs.

extrapolate to define ghost-zone values

Using the supplied template file `extrapolate.{c,f77,f90}`, write a subroutine `extrapolate` which takes a grid function (defined on the nominal grid) as input, and sets the grid function's ghost-zone values $k = -1$ and $k = N + 1$ using the extrapolation operators (31), so the grid function is now defined on the ghosted grid.

finite differencing

Using the supplied template file `derivatives.{c,f77,f90}`, write subroutines `dr_centered`, `drr_centered`, `dr_1sided_plus`, and `dr_1sided_minus`, each of which should take as input a grid function defined on the ghosted grid, and compute an approximation to the appropriate derivative (defined on the nominal grid for `dr_centered` and `drr_centered`, defined for $0 \le k \le N - 1$ for `dr_1sided_plus`, and defined for $1 \le k \le N$ for `dr_1sided_minus`) as output, using the finite difference approximations (29a), (29b), (30a), and (30b) respectively.

Computer Exercise 3 (Diagnostics)

This exercise has several parts, which should be done in sequence:

compute the Ricci scalar

Using the supplied template file `Ricci.{c,f77,f90}`, write a subroutine `Ricci` which takes as an input the state vector and its derivatives, and computes the Ricci scalar R as defined by (12a).

compute the constraints

Using the supplied template file `constraints.{c,f77,f90}`, write a subroutine `constraints` which takes the state vector, its derivatives, and K, $4\pi\rho$, $4\pi j^r$, and R, and computes the relative constraint grid functions $C_{\rm rel}$ and $C^r_{\rm rel}$ on the nominal grid as outputs, using the definitions (5) and (12). Use the extrapolation operators (31) and the centered finite difference approximations (29).

compute the mass function

Using the supplied template template file `mass_function.{c,f77,f90}`, write a subroutine `mass_function` which takes the state vector and its derivatives as inputs, and computes as output the "geometry-form" Misner-Sharp mass function $m_{\rm MS}$ defined by (16), and the slice mass $m_{\rm slice}$.

find the apparent horizon(s)

Using the supplied template file `apparent_horizons.{c,f77,f90}`, write a subroutine `apparent_horizons` which takes the state vector and its derivatives as inputs, and computes as output the left hand side Θ of the apparent horizon equation (20) and the positions of the apparent horizon(s). Once you've computed Θ, find the apparent horizon(s) by looking for grid cells $[k, k+1]$ where Θ changes sign, then linearly interpolating between r_k and r_{k+1} to find the r where $\Theta = 0$. Also have your horizon finder compute the mass contained within this radius ("the mass of the black hole") by linearly interpolating the Misner-Sharp mass function $m_{\rm MS}$ to this same horizon position.

compute all the diagnostics

Using the supplied template file `diagnostics.{c,f77,f90}`, write a subroutine

`diagnostics` which takes as input the state vector on the nominal grid, and calls all the other subroutines you've written so far to compute all the diagnostics.

convergence tests on Schwarzschild initial data

Using the appropriate test driver file `test_diagnostics.{c,f77,f90}`, together with your `diagnostics` subroutine and the other subroutines you've written, compute all the diagnostics for Schwarzschild initial data.

Check that the values you compute show proper convergence: both C and C^r should show show 2nd order convergence to zero everywhere in the interior of the grid, but you shouldn't expect good convergence within a grid point or two of the boundaries.

$m_{\rm MS}$ should show 2nd order convergence to the (spatially-constant) mass of the Schwarzschild spacetime, and you should find a single apparent horizon, whose position should by very close to $r = 2m = 2.0$. (You won't see exact 2nd order convergence here, because at the highest resolution $r = 2.0$ exactly coincides with a grid point, while at the two lower resolutions it falls between grid points.)

convergence tests on scalar-field–shell initial data

Repeat the above convergence tests for both of the scalar-field initial data families you constructed in exercise 1.

Problem 3 (More Diagnostics)

Some other diagnostics you could compute include the volume-integral form of the Misner-Sharp mass function (18), the **4-Ricci scalar** ${}^{(4)}R \equiv 8\pi(\rho - T)$ (a 4-invariant diagnostics for the scalar field density), and the **Kretschmann scalar**, the 4-Riemann curvature invariant ${}^{(4)}R_{abcd}\,{}^{(4)}R^{abcd}$ (detailed equations for this are given in [11,appendix B]). (The first two of these are easy, the last is conceptually straightforward but the equations are quite complicated.)

Computer Exercise 4 (Coordinates)

This exercise has two parts, which should be done in sequence:

compute the lapse function and shift vector

Using the supplied template file `coordinates.{c,f77,f90}`, write a subroutine `coordinates` which takes the state vector and its derivatives (defined on the nominal grid) as inputs, and computes the lapse function α and shift vector β on the nominal grid as outputs, by solving the two simultaneous linear equations (21) and (23) at each grid point.

compute the light cones

Using the supplied template file `light_cones.{c,f77,f90}`, write a subroutine `light_cones` which takes the relevant metric components and the lapse function and shift vector as inputs, and computes the light cone speeds c_- and c_+ as defined by equation (15).

check the inner boundary causality

Using the supplied template file `check_inner_boundary.{c,f77,f90}`, write a subroutine `check_inner_boundary` which takes the light cone speeds as inputs, and checks that the inner-boundary causality condition (24) is satisfied. This subroutine should

work in the same way as your `check_metric` subroutine from exercise 2: if everything is ok your subroutine should do nothing (i.e. just return), if not, your subroutine should print a suitable error message and halt execution.

convergence tests on Schwarzschild initial data

Using the appropriate test driver `test_coordinates.{c,f77,f90}`, together with your `coordinates` and `light_cones` subroutines and the other subroutines you've written, do a convergence test to check that α, β, c_+, and c_- all show 2nd order convergence to the correct values for the Schwarzschild initial data files.

Computer Exercise 5 (Evolution Equations)

This exercise has several parts, which should be done in sequence:

compute the interior evolution equations

Using the supplied template file `evolution_interior.{c,f77,f90}`, write a subroutine `evolution_interior` which takes the state vector and its derivatives as inputs, and computes the right-hand-side of the evolution equations, i.e. ∂_t of the state vector, at the grid points $0 \le k \le N-1$ (i.e. everywhere in the nominal grid *except* the outer-boundary grid point $k = N$), using the evolution equations (13), (14), and (22). As discussed in sections III A and III C, use the upwind finite differencing approximations (32) for all the (1st) spatial partial derivatives in the Lie derivatives (14), and use the centered finite differencing approximations (29a) for all other spatial partial derivatives appearing in the equations.

compute the radiation outer boundary conditions

Using the supplied template file `outer_boundary.{c,f77,f90}`, write a subroutine `outer_boundary` which takes as inputs a (state-vector) grid function, its derivative, and the slice mass $m_{\rm slice}$, and computes ∂_t of the grid function at the outer-boundary grid point $k = N$ as an output, using the outgoing-radiation outer boundary condition (28).

assemble the right-hand-side subroutine

Using the supplied template file `RHS.{c,f77,f90}`, write a subroutine `RHS` which takes as input the state vector defined on the nominal grid, and the right-hand-side of the evolution equations, i.e. ∂_t of the state vector, everywhere on the nominal grid, by calling the other subroutines you've already written.

Using the appropriate test driver `test_RHS.{c,f77,f90}`, together with your `RHS` subroutine and the other subroutines you've written, do a convergence test on ∂_t of the state vector for the Schwarzschild initial data files. $\partial_t P$, and $\partial_t Q$ should be identically zero everywhere (this is a vacuum slice, with P and Q both identically zero). ∂_t of the other 4 field variables should show 2nd order convergence to zero at all the interior grid points, and at the outer-boundary grid point $k = N$. At the inner-boundary grid point $k = 0$, $\partial_t A$ and $\partial_t B$ should show 2nd order convergence to zero, but because of the 2nd spatial derivatives in the X and Y evolution equations (13c) and (13d), $\partial_t X$ and $\partial_t Y$ should only show 1st order convergence to zero.[6]

[6] Recall from section III B that the combination of the extrapolation operator (31) and the centered 2nd derivative approximation (29b) is only 1st order accurate.

Computer Exercise 6 (Time Stepping)
Write a subroutine `time_step` to take as input the state vector at time t, and update this to give as output the state vector at time $t+\Delta t$, using the 2nd order Runge-Kutta scheme (B9). Your `time_step` subroutine should call the `RHS` subroutine twice.

Problem 4 (Nonuniform Spatial Gridding)
You could make the code much more efficient (comparably-accurate results with much less computation) by switching to a nonuniform spatial grid. There are two reasonably simple ways to do this:

- You could keep the tensor basis using (r, θ, ϕ) coordinates, but use a grid which is uniform in some other radial coordinate $w = w(r)$, with w a suitably-chosen nonlinear function of r. Then

$$\partial_r u = J_1 \frac{\partial u}{\partial w} \tag{33a}$$

$$\partial_{rr} u = J_1^2 \frac{\partial^2 u}{\partial w^2} + J_2 \frac{\partial u}{\partial w} \tag{33b}$$

  where we define

$$J_1 = \frac{\partial w}{\partial r} \tag{34a}$$

$$J_2 = \frac{\partial^2 w}{\partial r^2} \tag{34b}$$

  so you can rewrite the finite differencing routines from exercise 2 to still compute r derivatives even though the grid is uniform in w. $w(r) = \log(r)$ is a common choice for nonuniform grids, but for our purposes it tends to result in inadequate resolution (too large a grid spacing) in the outer parts of the grid. $w(r) = \sqrt{r}$ works better. The code described in [1, 11] uses this approach, though with a more complicated choice for the $w(r)$ function.

- You could use (w, θ, ϕ) as a tensor basis for the $3+1$ equations. This is similar to the previous approach, avoids the requirement to transform the derivatives via the J_1 and J_2 coefficients. On the other hand, it requires tensor-transforming the initial data, and any other quantities you input or output which aren't 3-scalars.

Problem 5 (4th order Finite Differencing)
You could make the code much more accurate by switching to 4th order finite differencing in both space and time. [1] discusses numerical methods for this in detail. Briefly, you would use ghost zones 2 grid points wide, 5-point extrapolation operators to determine the ghost-zone values, 4th order (5-point) centered and off-centered finite difference approximations for the spatial and a 4th order time integrator such as the Runge-Kutta method (B10).

Computer Exercise 7 (Time Evolution of Schwarzschild Initial Data)
Use the sample `evolve` driver program, together with your `time_step`, `RHS`, and `diagnostics` subroutines and the other subroutines they call, to try time-evolving the Schwarzschild initial data to $t = 50$, and check the convergence of the results to the correct values (which should be the same as the initial data).

Look at a movie of the relative energy constraint $C_{\rm rel}$ as a function of space and time. You'll see some small errors propagating in from the outer boundary, and the values well away from the outer boundary will grow rapidly, but if you look at the scale of the plots you'll see that the constraints are overall still preserved to high accuracy.

Computer Exercise 8 (Accretion of a Thick Scalar Field Shell onto the Black Hole)
Repeat exercise 7, this time using the pqw5 initial data slices you constructed in exercise 1.

The initial data has zero momentum ($j^r = 0$), so the initial scalar field shell is actually the superposition of an ingoing and an outgoing shell. If you look at a movie of $4\pi B\rho$, $4\pi B j^r$, and $m^{\rm MS}$, you should see the shell split into these two shells right away. The ingoing shell (which has negative j^r) will fall into the black hole at around $t = 20$, while the outgoing shell (with positive j^r) will propagate out away from the black hole until it reaches the outer grid boundary. You should also see each propagating shell leave a "tail" behind; this is caused by the scalar field scattering off the background spacetime curvature. Figure 1 (taken from [1]) shows the general type of results you should get.

If you look at the time evolution of the apparent horizon mass and/or radius, you should see this increase as the scalar field falls into the black hole. Figure 2 (taken from [1]) shows an example of this.

Look at a movie of the relative energy constraint $C_{\rm rel}$ as a function of space and time. Unfortunately, you'll see a very large wave of constraint violation which starts at the outer boundary and travels in at the speed of light. This constraint violation is a continuum effect from our outer boundary conditions (28), so it doesn't converge away as you increase the resolution. (Check this!) However, if you zoom in on $r \leq 50$ (i.e. on a part of the grid which isn't causally connected to the outer boundary for the length of time of this evolution), then you should see that the constraints are small, and show 2nd order convergence to zero.

Problem 6 (Longer Evolutions; "Tails")
Another interesting thing to look at for this data set is the "tails" formed by the scattering of the scalar field shells from the spacetime curvature near the black hole. You would see these as several further outgoing shells of scalar field at late times, the first being roughly $r = 50$ inside the outermost outgoing scalar field shell. To see these clearly you would need quite long evolutions (at least $t \geq 200$ would be desirable), which would in turn require having the outer boundary quite far out to avoid having outer boundary errors corrupt the tails. Thus these runs would take quite a while.

Computer Exercise 9 (Accretion of a Thin Scalar Field Shell onto the Black Hole)
Repeat exercise 8, this time using the pqw1 initial data slices you constructed in exercise 1.

This scalar field shell is thin and massive (it's about 3 times as massive as the initial black hole). If you look at a movie of $4\pi B\rho$, $4\pi B j^r$, and $m_{\rm MS}$, you should be able to see the ingoing and outgoing-going shells leaving "trails" behind. These are due to the scalar field scattering off the spacetime curvature.

When the ingoing shell falls into the black hole, for a short time (around $t = 19$ to $t = 20$) there are 3 apparent horizons present. Figure 3 shows this.[7] If you look at a movie of Θ

[7] To get a high sampling rate in the interesting time range $t \in [19, 20]$, while not having too-huge data files, there were actually 3 separate runs of the `evolve` driver, one for $t \in [0, 19]$ with output every 1, one for $t \in [19, 20]$ with output every 0.02, and one for $t \in [20, 50]$ with output every 1. The `--append` option

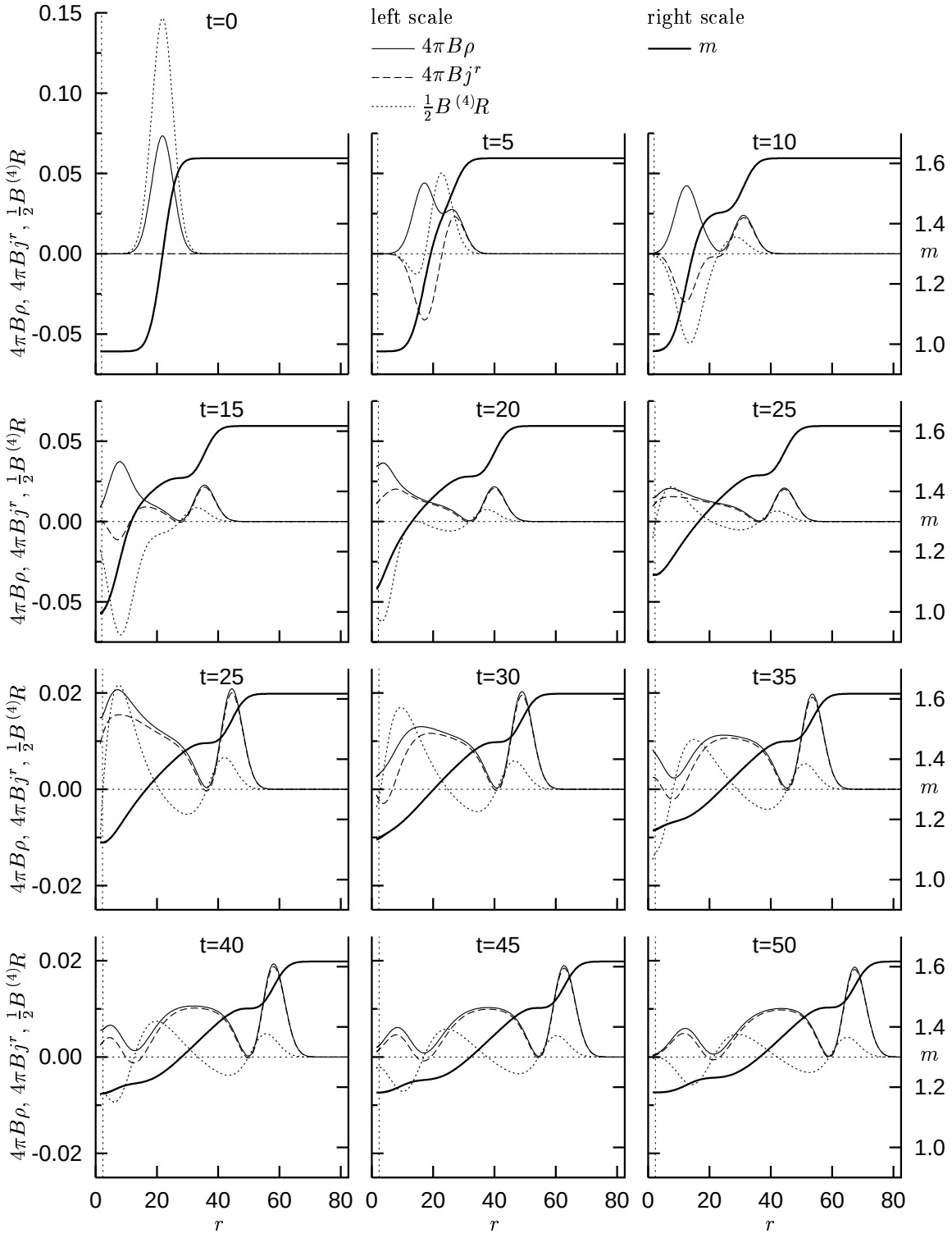

FIG. 1: This figure shows the time evolution of a scalar field shell accreting onto the black hole.

of that driver was used to get all the data into a single set of files.

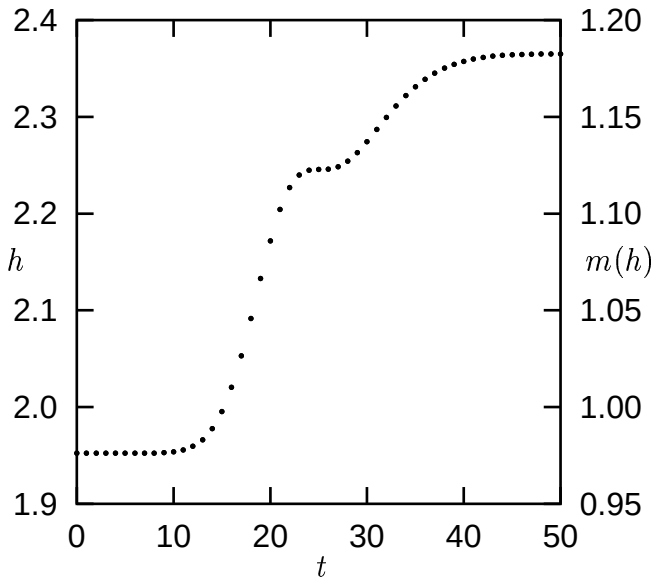

FIG. 2: This figure shows the radius and mass of the black hole growing as the ingoing scalar field shell falls into the black hole. h is the radius of the apparent horizon, and $m(h)$ is the mass contained within that radius, i.e. the black hole's mass.

you should be able to see why the apparent horizons always appear and disappear in pairs. (See [1,figure 8] if you can't figure this out.)

APPENDIX A: SCHWARZSCHILD SPACETIME

.

As discussed in [11], with our choice of field variables and coordinates, an Eddington-Finkelstein slice of Schwarzschild spacetime ([8,box 31.2]) looks like this:

$$\alpha = \frac{1}{\sqrt{1+\frac{2m}{r}}} \tag{A1a}$$

$$\beta = \frac{2m}{r}\frac{1}{1+\frac{2m}{r}} \tag{A1b}$$

$$A = 1+\frac{2m}{r} \tag{A1c}$$

$$B = r^2 \tag{A1d}$$

$$X = -\frac{2m}{r^2}\frac{1+\frac{m}{r}}{\sqrt{1+\frac{2m}{r}}} \tag{A1e}$$

$$Y = 2m\frac{1}{\sqrt{1+\frac{2m}{r}}} \tag{A1f}$$

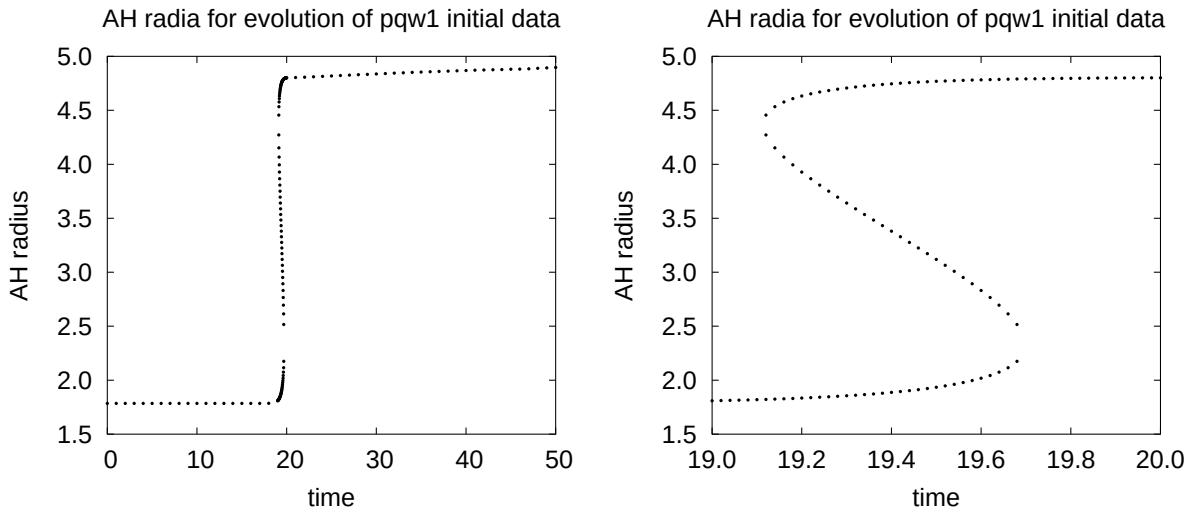

FIG. 3: This figure shows the apparent horizon radia for the pqw1 evolution. Notice that for a short time there are 3 distinct apparent horizons present.

Since Schwarzschild spacetime is vacuum, $P = Q = 0$. As discussed in [12],

$$\Theta = \frac{2(r-2m)}{r^{3/2}\sqrt{r+2m}} \tag{A2}$$

APPENDIX B: A BRIEF INTRODUCTION TO THE METHOD OF LINES

In this appendix we give a brief introduction to the **method of lines** (**MOL**) for parabolic or hyperbolic PDEs. This technique has a number of desirable features, but it's not too widely known. For more information, two excellent introductory MOL references are [13, 14]. Unfortunately, both of these are rather inaccessible. A more accessible reference is [15].

1. Basic Concepts

The basic idea of MOL is simple: We initially finite difference[8] only the spatial derivatives in the PDE, keeping the time derivatives continuous. This yields a set of ODEs – coupled via the spatial finite differencing – for the time dependence of the field variables at the spatial grid points. (In the terminology of relativity, these ODEs give the time dependence of the field variables along the spatial-grid-point world lines.) A suitable ODE integrator is then used to time-integrate these ODEs.

[8] MOL can also be used with finite element or pseudospectral discretizations, but we won't discuss that here.

A simple example should help to clarify this: Consider the flat-space linear scalar advection equation written in 1st order form,

$$\partial_t u = -c\,\partial_x u \tag{B1}$$

on the domain $(t,x) \in [0,\infty) \times [0,1)$, with the (smooth) coefficient function c being everywhere positive (so that propagation is solely rightward), subject to the periodic boundary condition

$$u(t, x{=}0) = u(t, x{=}1) \tag{B2}$$

To treat this problem by MOL, we first discretize the spatial dimension with the uniform grid $x_k = k\,\Delta x$ for $k = 0,\ 1,\ 2,\ \ldots,\ K{-}1$, where $\Delta x \equiv 1/K$ for some integer K. (For the moment we keep the time dimension continuous.) Introducing the usual notation $u_k \equiv u(x_k)$ and $c_k \equiv c(x_k)$, and approximating the spatial derivative in the evolution equation (B1) with the usual centered 2nd order finite differencing molecule, we obtain the coupled system of ODEs

$$\partial_t u_k = -c_k \frac{u_{k+1 \bmod K} - u_{k-1 \bmod K}}{2\,\Delta x} \qquad (\text{for } k = 0,\ 1,\ 2,\ \ldots,\ K{-}1) \tag{B3}$$

for the time dependence of the state variables $\{u_k\}$. These ODEs can then be integrated using any suitable ODE integration scheme.[9]

For example, if we were to use the leapfrog time integrator

$$\mathbf{u}(t + \Delta t) = \mathbf{u}(t - \Delta t) + 2\,\Delta t\,(\partial_t \mathbf{u}\big|_t) + O\big((\Delta t)^2\big) \tag{B4}$$

(where $\mathbf{u}$ denotes the state vector $\{u_k\}$) to time integrate the MOL ODEs (B3), it's easy to see that at each time level $t_n \equiv n\,\Delta t$ we would obtain the finite difference equations

$$\frac{u_k^{n+1} - u_k^{n-1}}{2\,\Delta t} = -c_k \frac{u^n_{k+1 \bmod K} - u^n_{k-1 \bmod K}}{2\,\Delta x} \qquad (\text{for } k = 0,\ 1,\ 2,\ \ldots,\ K{-}1) \tag{B5}$$

where we use the usual notation $u_k^n \equiv u(t{=}t_n, x{=}x_k)$.

2. Comparison with "Space and Time Together" Methods

Despite its superficial differences, MOL is actually closely connected to the more common "space and time together" (**SATT**) finite difference methods for PDEs (where we finite difference in both space and time together, directly obtaining a set of spacetime finite difference equations): Since all practical ODE integrators use finite differencing in the time dimension, the overall result of applying such an integrator to a system of MOL ODEs, is to (implicitly) construct and solve a (complicated) system of finite difference equations in space and time.

---

[9] In general, it's usually best to integrate ODEs with modern adaptive codes like ODEPACK/LSODE (`http://www.netlib.org/odepack/`), but for MOL these are an unnecessary complication. In particular, in MOL there's normally little point in making the time integration much more sophisticated than the spatial finite differencing. So, if we use a straightforward 2nd order finite differencing scheme in space, with a fixed number and distribution of grid points, it's perfectly reasonable to use a correspondingly simple 2nd order time integrator, with a fixed (time) step size.

For example, looking at the MOL finite difference equations (B5), it's apparent that these are precisely the equations for the standard 2nd order centered-in-time centered-in-space leapfrog finite differencing scheme, applied to the original PDE (B1).

If MOL is "just" a special sort of SATT finite differencing scheme, why bother? There are two key advantages to MOL:

- By decoupling the spatial finite differencing from the time integration, MOL makes the resulting numerical scheme and computer code simpler and more modular.
- There is a well-developed stability theory, and many excellent numerical methods, for the time integration of systems of ODEs; MOL can use these.

Corresponding to the equivalence of a MOL and SATT numerical schemes, the usual Courant-Friedrichs-Lewy stability limit of an explicit SATT scheme has an exact analog in MOL: When time integrating a system of MOL ODEs with an explicit ODE integrator, the maximum time step will be limited by ODE-integration stability considerations, to precisely the same value as the equivalent SATT scheme's CFL limit. For example, the scalar-wave-equation finite difference equations (B5) have the stability limit $|c\,\Delta t| \le \Delta x$, which may be viewed either as the usual CFL limit for the CTCS leapfrog SATT scheme, or the ODE-integration stability limit for the leapfrog time integrator (B4) applied to the system of ODEs (B3).

3. Boundary Conditions, Runge-Kutta Methods

Now consider a variant form of our example problem, where we extend the spatial domain from $[0, 1)$ to $[0, 1]$, and introduce the left boundary condition

$$u(t, x{=}0) = \sin t \tag{B6}$$

(There is no right boundary condition: The system (B1) and (B6) is well-posed without one, since no part of the problem domain causally depends on the right boundary.)

To treat this variant by MOL, we first extend the grid to include $x_K = 1$. We then time-differentiate the left boundary condition (B6) and combine it with the interior evolution equation (B1) to obtain the combined evolution equations

$$\partial_t u = \begin{cases} \cos t & (\text{if } x = 0) \\ -c\,\partial_x u & (\text{if } x > 0) \end{cases} \tag{B7}$$

Finally, we spatially finite difference this in the same manner as before for the grid interior, but using the off-centered finite difference approximation (30b) at the right boundary grid point, to obtain the MOL ODEs

$$\partial_t u_0 = \cos t \tag{B8a}$$

$$\partial_t u_k = -c_k \frac{u_{k+1} - u_{k-1}}{2\,\Delta x} \qquad (\text{for } k = 1,\, 2,\, 3,\, \ldots,\, K{-}1) \tag{B8b}$$

$$\partial_t u_K = -c_K \frac{u_{K-2} - 4u_{K-1} + 3u_K}{2\,\Delta x} \tag{B8c}$$

which we denote in vector form as $\partial_t \mathbf{u} = \mathbf{f}(t, \mathbf{u})$.

For this problem the leapfrog ODE integrator (B4) is not suitable – it's unstable for any choice of the time step Δt! In fact, leapfrog is unstable for "most" systems of equations. An ODE integrator with much better stability properties (both for this and other problems) is the 2nd order Runge-Kutta scheme, where we first compute an approximate "predicted" state vector at the next time level,

$$\mathbf{u}_{\rm pred}^{n+1} = \mathbf{u}^n + \Delta t\, \mathbf{f}(t, \mathbf{u}^n) \tag{B9a}$$

then use this to make the final time step,

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \tfrac{1}{2}\,\Delta t\, \left(\mathbf{f}(t, \mathbf{u}^n) + \mathbf{f}(t + \Delta t, \mathbf{u}_{\rm pred}^{n+1})\right) \tag{B9b}$$

This is stable for most problems (for small enough time step). Another excellent time integrator is the classical 4th order Runge-Kutta method,

$$\begin{aligned}
\mathbf{k}^{(1)} &= \mathbf{f}(t, \mathbf{u}_n) && \text{(B10a)}\\
\mathbf{k}^{(2)} &= \mathbf{f}(t + \tfrac{1}{2}\,\Delta t, \mathbf{u}_n + \tfrac{1}{2}\,\Delta t\, \mathbf{k}^{(1)}) && \text{(B10b)}\\
\mathbf{k}^{(3)} &= \mathbf{f}(t + \tfrac{1}{2}\,\Delta t, \mathbf{u}_n + \tfrac{1}{2}\,\Delta t\, \mathbf{k}^{(2)}) && \text{(B10c)}\\
\mathbf{k}^{(4)} &= \mathbf{f}(t + \Delta t, \mathbf{u}_n + \Delta t\, \mathbf{k}^{(3)}) && \text{(B10d)}\\
\mathbf{u}_{n+1} &= \mathbf{u}_n + \Delta t\, \left(\tfrac{1}{6}\mathbf{k}^{(1)} + \tfrac{1}{3}\mathbf{k}^{(2)} + \tfrac{1}{3}\mathbf{k}^{(3)} + \tfrac{1}{6}\mathbf{k}^{(4)}\right) && \text{(B10e)}
\end{aligned}$$

This is quite efficient, and has excellent stability properties.

---

[1] J. Thornburg (1999), gr-qc/9906022.

[2] R. Marsa and M. W. Choptuik, Phys Rev D **54**, 4929 (1996).

[3] J. York, in *Sources of Gravitational Radiation*, edited by L. Smarr (Cambridge University Press, Cambridge, England, 1979).

[4] E. Malec, Phys Rev D **49**, 6475 (1994).

[5] J. Libson, J. Massó, E. Seidel, W.-M. Suen, and P. Walker, Phys. Rev. D **53**, 4335 (1996).

[6] P. Diener, submitted to Class. Quantum Grav. (2003), gr-qc/0305039, gr-qc/0305039.

[7] C. W. Misner and D. H. Sharp, Physical Review B **136**, 571 (1964).

[8] C. W. Misner, K. S. Thorne, and J. A. Wheeler, *Gravitation* (W. H. Freeman, San Francisco, 1973).

[9] J. Given and N. O. Murchadha, Physical Review D **52**, 758 (1995).

[10] J. York, in *Frontiers in Numerical Relativity*, edited by C. Evans, L. Finn, and D. Hobill (Cambridge University Press, Cambridge, England, 1989), pp. 89–109.

[11] J. Thornburg, Phys. Rev. D **59** (1999), gr-qc/9801087.

[12] J. Thornburg, Phys. Rev. D **54**, 4899 (1996).

[13] J. M. Hyman, Tech. Rep. COO-3077-139, ERDA Mathematics and Computing Laboratory, Courant Institute of Mathematical Sciences, New York University (1976).

[14] J. M. Hyman, in *NATO Advanced Research Workshop on the Numerical Modeling of Nonlinear Stellar Pulsations: Problems and Prospects*, edited by J. R. Buchler (Kluwer, Dordrecht, 1989), pp. 215–237, ISBN 0-7923-0598-1, also available as Los Alamos National Laboratories Report LA-UR 89-3136.

[15] W. E. Schiesser, *The Numerical Method of Lines: Integration of Partial Differential Equations* (Academic Press, New York, 1991), ISBN 0-12-624130-9.

[16] J. W. York, Jr. and T. Piran, in *Spacetime and Geometry: The Alfred Schild Lectures*, edited by R. A. Matzner and L. C. Shepley (University of Texas Press, Austin (Texas), 1982), pp. 147–176, ISBN 0-292-77567-9.

[17] J. York, in *Gravitational Radiation*, edited by N. Deruelle and T. Piran (North-Holland, Amsterdam, 1983), pp. 175–201, ISBN 0-444-86560-8.

[18] J. York, in *Essays in General Relativity: A Festschrift for Abraham Taub*, edited by F. J. Tipler (Academic Press, New York, 1980), pp. 39–58, ISBN 0-12-691380-3.