Advanced Numerical Relativity

Dr Katy Clough
Ernest Rutherford Fellow
Queen Mary University of London

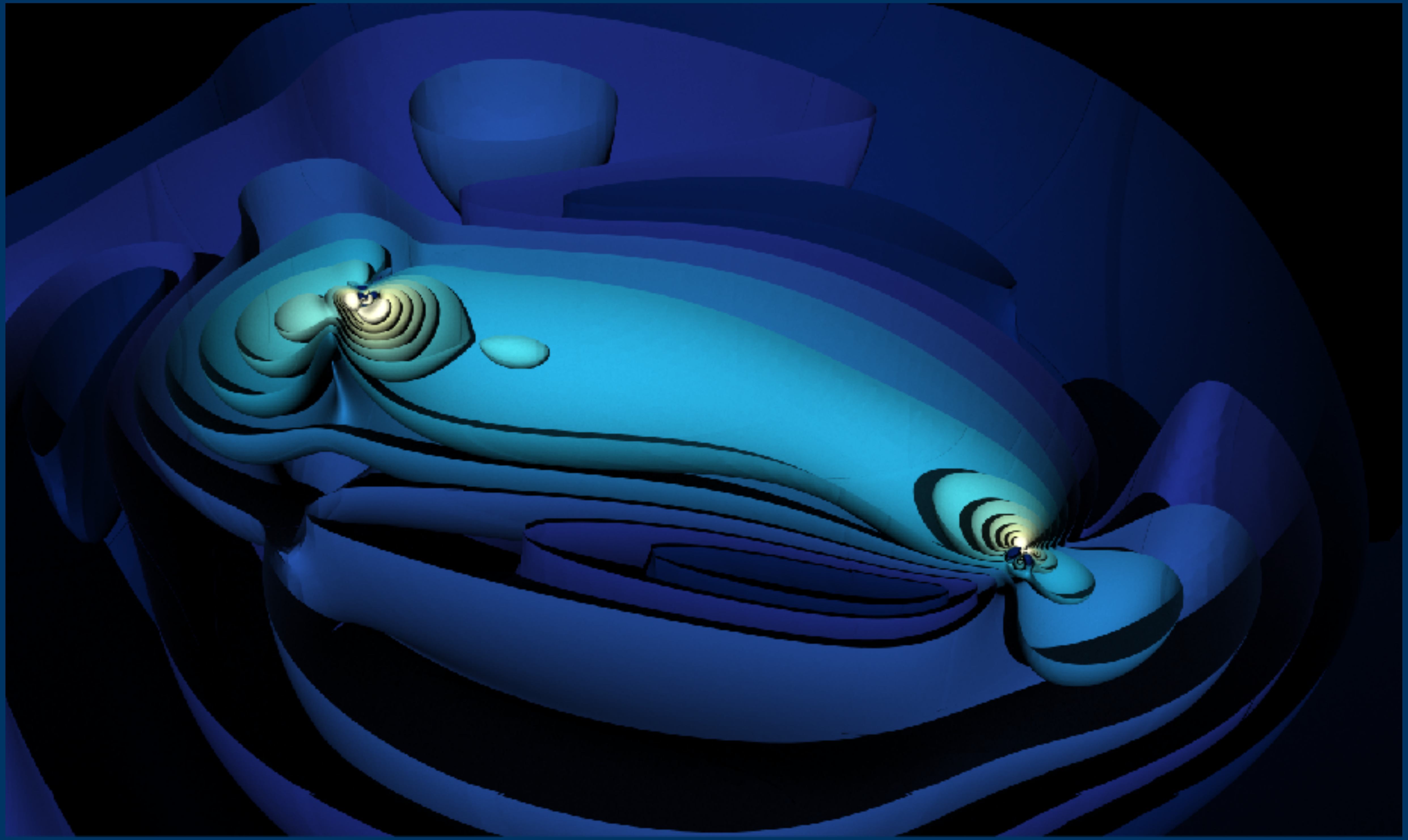
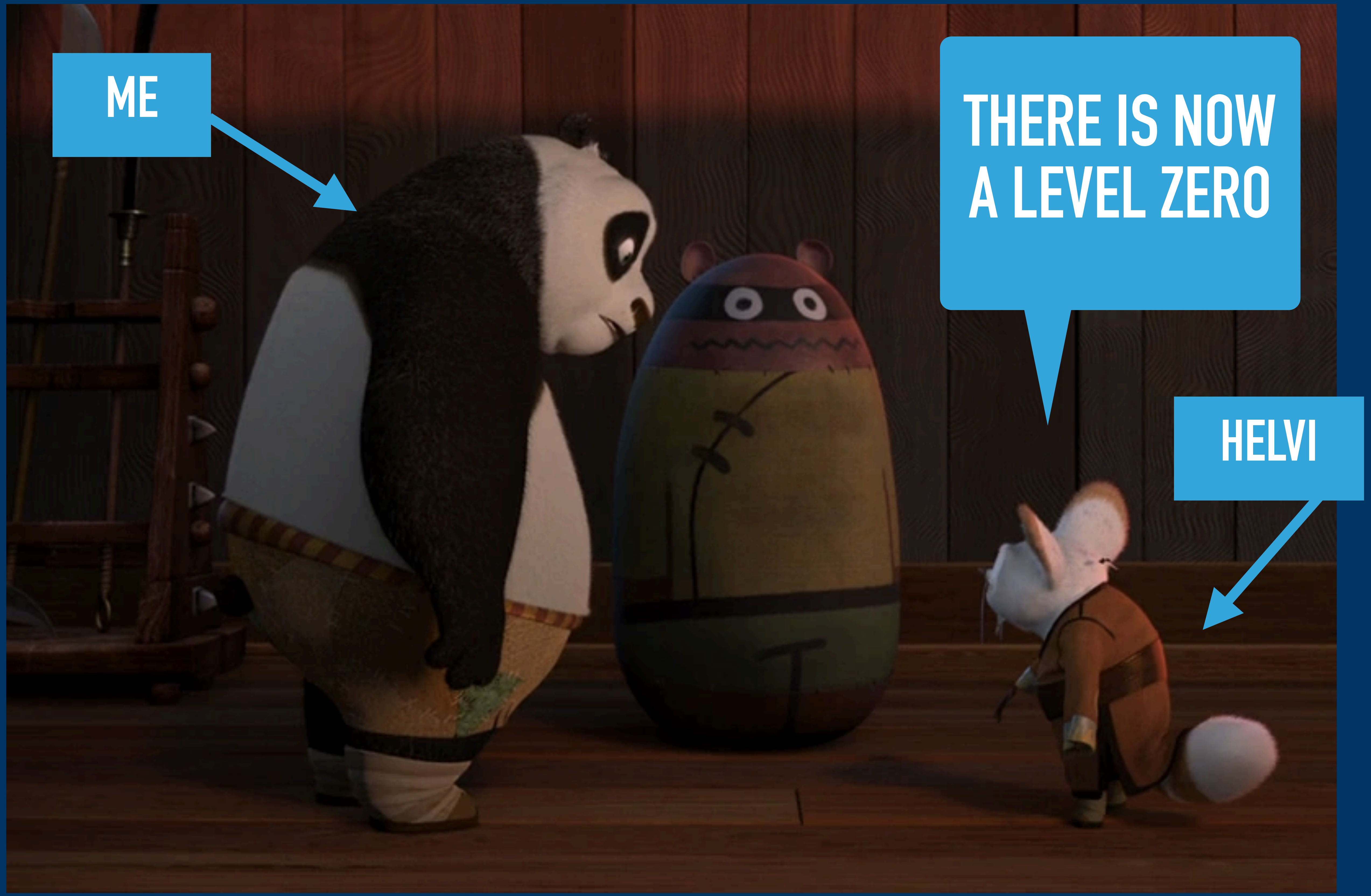


Image credit: Bamber/KC/Cielo

Practical ~~Advanced~~ Numerical Relativity

Katy Clough



Secrets of “advanced” NR people:

- Make lots of mistakes

- Take the time to learn from your mistakes

- Immerse yourself in the code, visualise the data

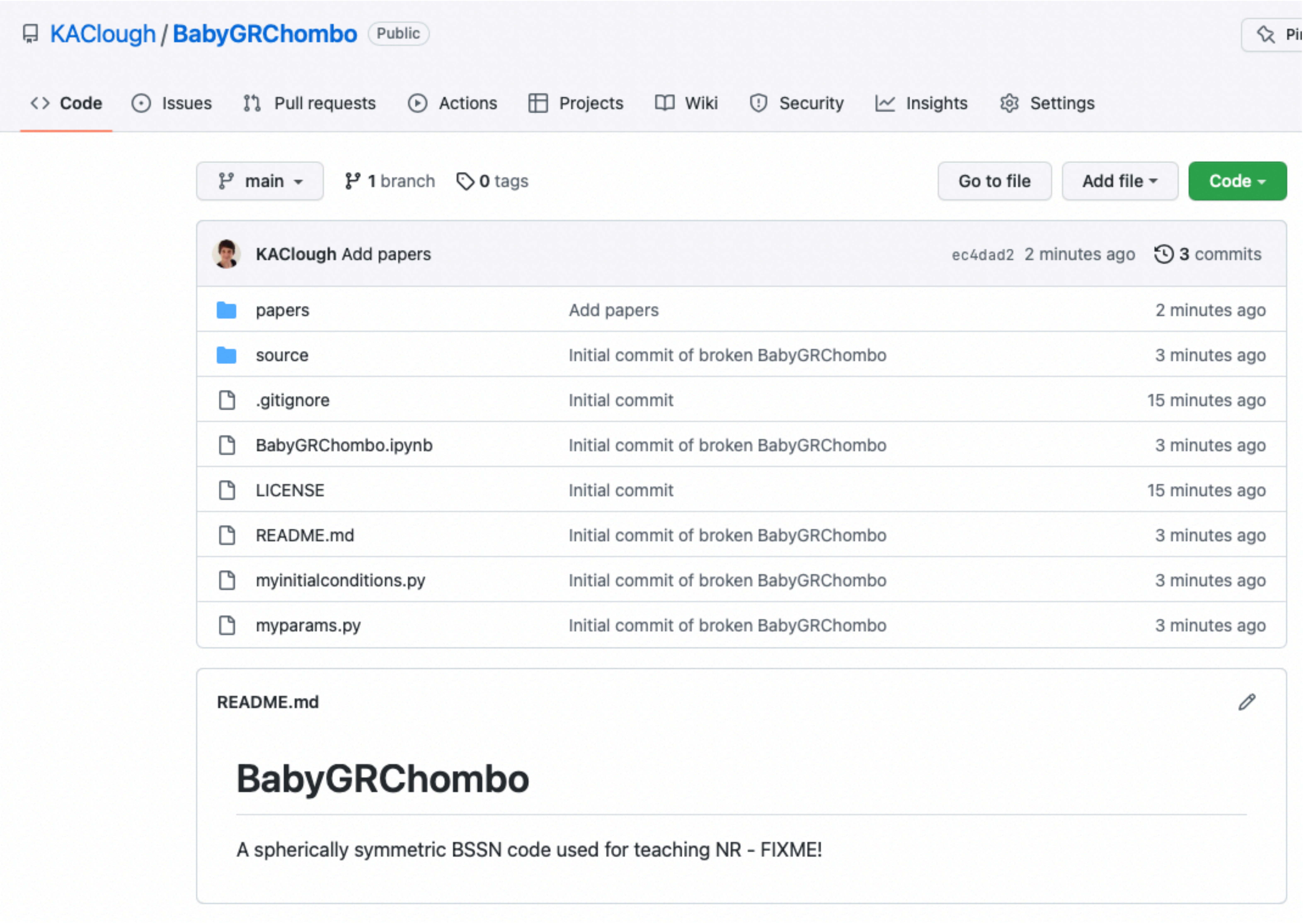
- Don’t give up or lose hope… it will work!

Topic of these sessions:
What are the possible mistakes one can make when setting up a numerical relativity simulation?

Specific questions we will address

- Does the physical problem require NR?
- How do I choose my initial data?
- What is the right formulation and gauge?
- How should I set the (many) parameters?
- What diagnostics do I want to extract?
- How can I be sure my simulation is correct?

BabyGRChombo is broken!



Format for the lectures

Session 1 (now):

- Brief reminder of the big picture - GR and NR 101
- We will discuss and play “spot the deliberate mistake” for each of our questions
- We will look at the relevant parts of BabyGRChombo, a python code that implements NR in spherical symmetry, and think about what could be wrong at each stage

Session 2 (tomorrow):

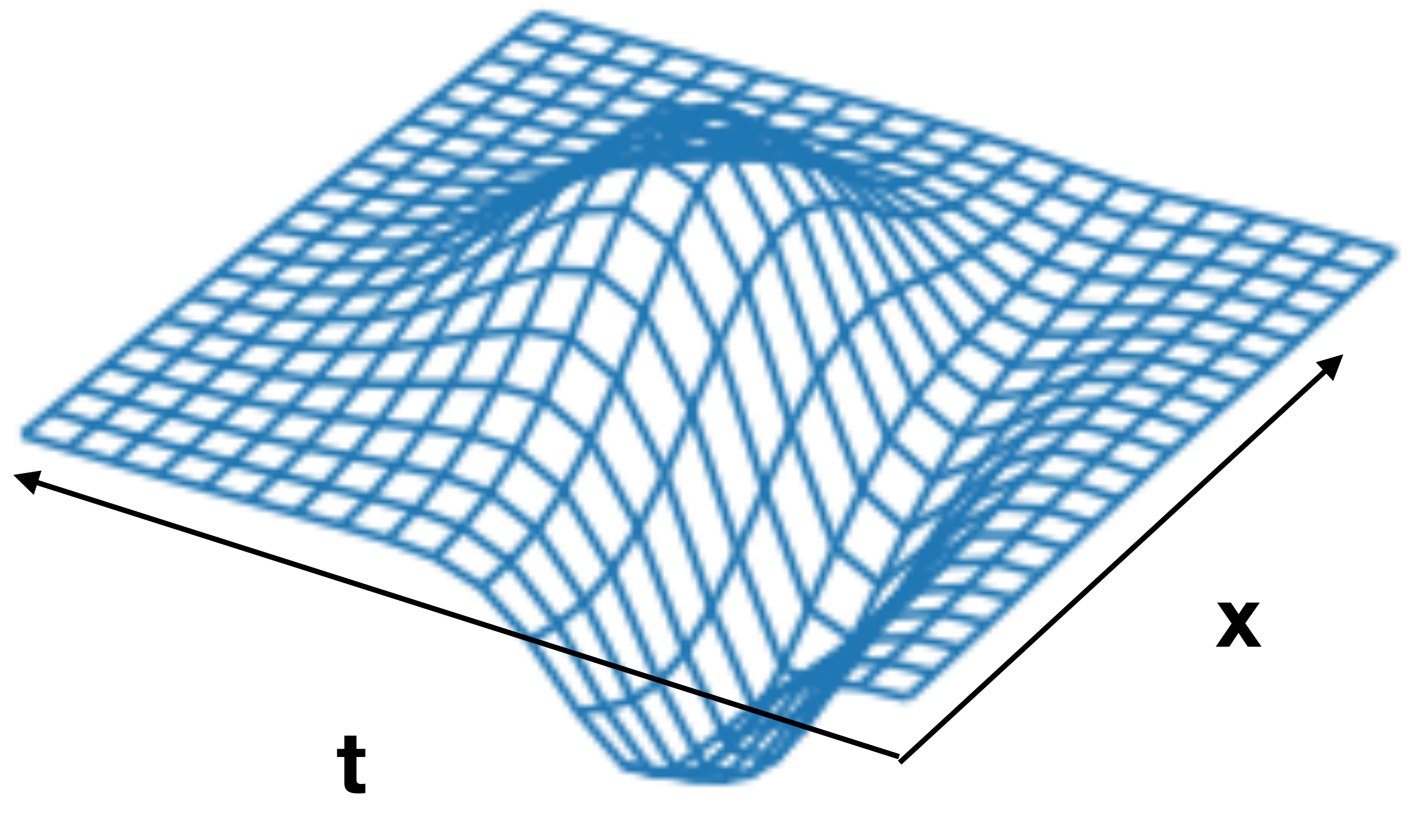
- We will try to fix / upgrade BabyGRChombo
- Ask questions about things you have seen but haven’t understood

GR & NR 101

$$R_{ab} - R/2\, g_{ab} = 8\pi\, T_{ab}$$

Curved spacetime

$$ds^2 = f(x,t)\ dt^2 + g(x,t)\ dx^2 + 2\ h(x,t)\ dt\ dx$$



$$ds^2 = \begin{pmatrix} dt & dx \end{pmatrix} \begin{pmatrix} f(x,t) & h(x,t) \\ h(x,t) & g(x,t) \end{pmatrix} \begin{pmatrix} dt \\ dx \end{pmatrix}$$

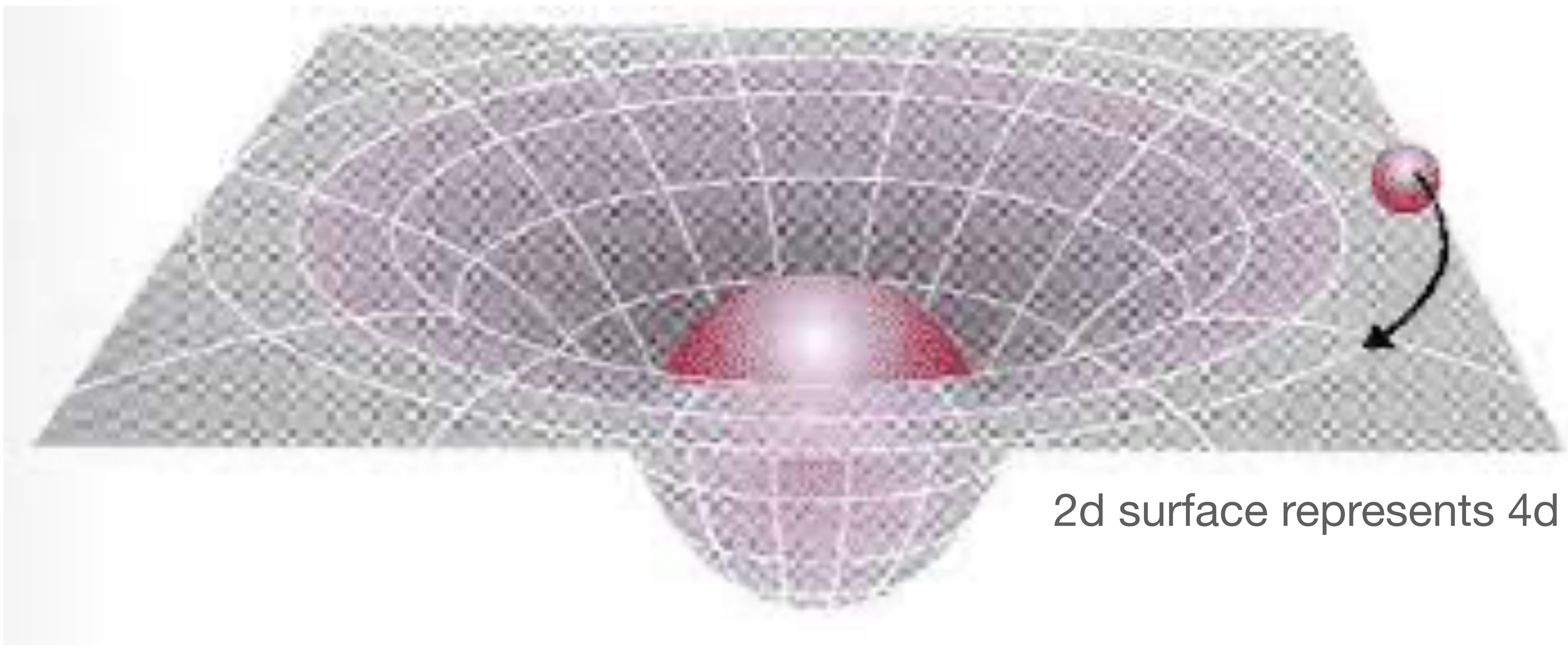
Curved spacetime

$$ds^2 = \begin{pmatrix} dt & dx & dy & dz \end{pmatrix} \underbrace{\begin{pmatrix} g_{00} & g_{01} & g_{02} & g_{03} \\ g_{10} & g_{11} & g_{12} & g_{13} \\ g_{20} & g_{21} & g_{22} & g_{23} \\ g_{30} & g_{31} & g_{32} & g_{33} \end{pmatrix}} \begin{pmatrix} dt \\ dx \\ dy \\ dz \end{pmatrix}$$

“The spacetime metric”

$$g_{ab}(t, \vec{x})$$

The Einstein equation tells us how the metric should look, given some energy/matter distribution

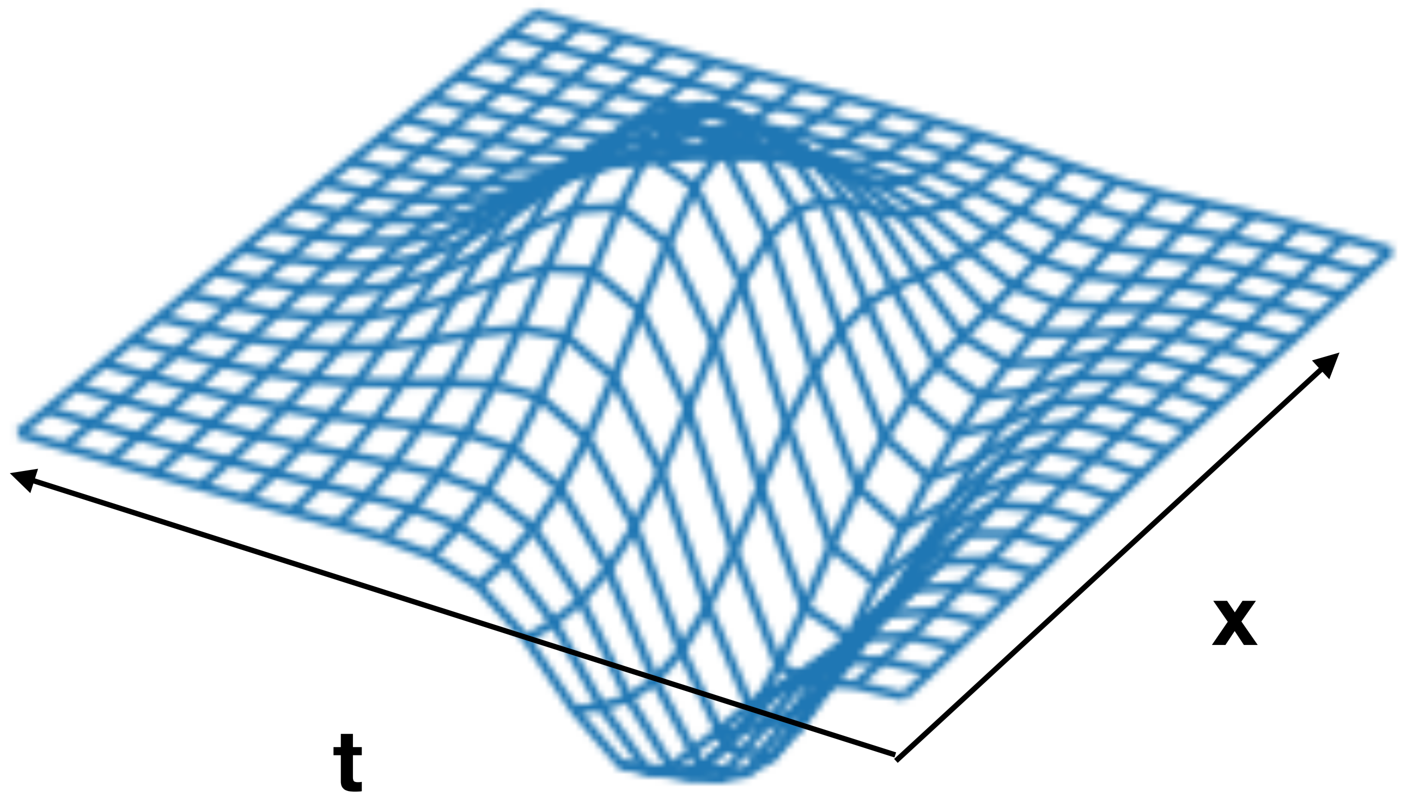


2d surface represents 4d spacetime

$$R_{ab} - R/2\ g_{ab} = 8\pi\ T_{ab}$$

“Matter tells spacetime how to curve…”

The Einstein equation tells us how the metric should look, given some energy/matter distribution



R_{ab} - R/2 g_{ab} = 8π T_{ab}

4 constraint equations for any time slice - non linear elliptic/Poisson equation

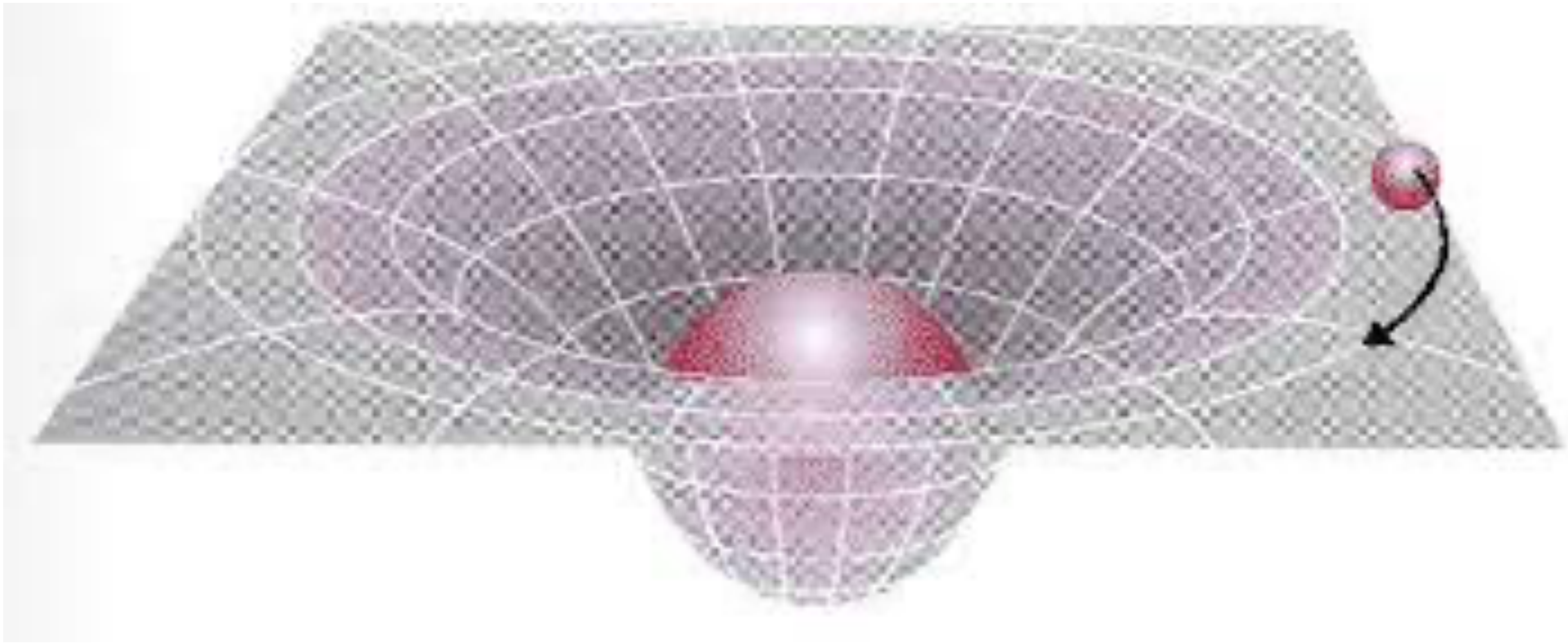
$$\frac{\partial^2 g}{\partial x^2} + \text{non linear terms} = f(\text{energy, momentum})$$

An evolution equation for all time - non linear hyperbolic/wave equation

$$\frac{\partial^2 g}{\partial t^2} - \frac{\partial^2 g}{\partial x^2} + \text{non linear terms} = f(\text{energy, momentum})$$

“Matter tells spacetime how to curve…”

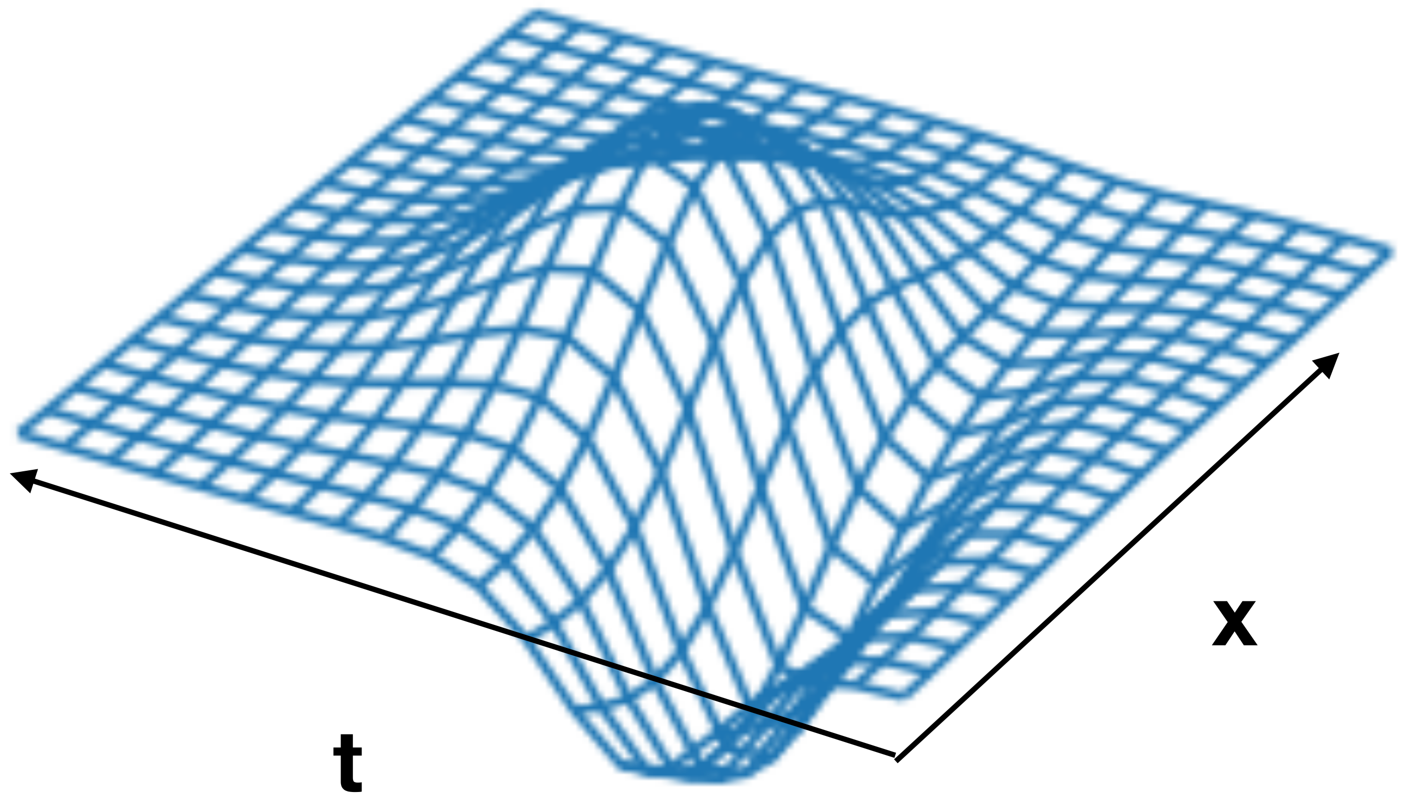
The metric determines the motion of matter



$$R_{ab} - R/2\, g_{ab} = 8\pi\, T_{ab}$$

"…spacetime tells matter how to move."

The metric determines the motion of matter



$R_{ab} - R/2\, g_{ab} = 8\pi\, T_{ab}$

Continuity equation

$$\frac{\partial \rho}{\partial t} + \underbrace{\nabla}_{g_{ab}} \cdot \mathbf{j} = \underbrace{\text{source}}_{g_{ab}}$$

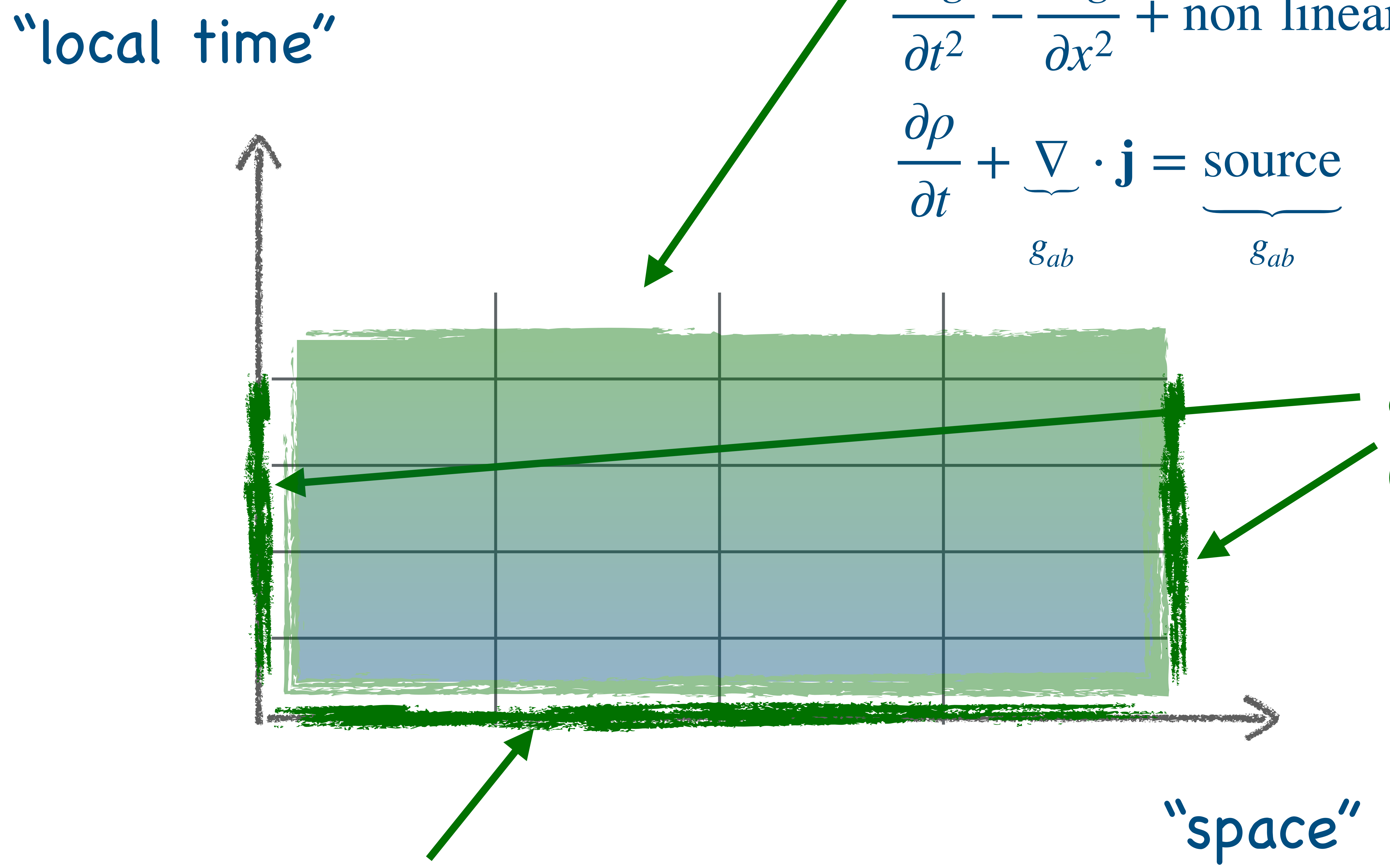
"...spacetime tells matter how to move."

Numerical relativity

Fill using Einstein equation and continuity for matter

$$\frac{\partial^2 g}{\partial t^2} - \frac{\partial^2 g}{\partial x^2} + \text{non linear terms} = f(\text{energy, momentum})$$

$$\frac{\partial \rho}{\partial t} + \underbrace{\nabla}_{g_{ab}} \cdot \mathbf{j} = \underbrace{\text{source}}_{g_{ab}}$$

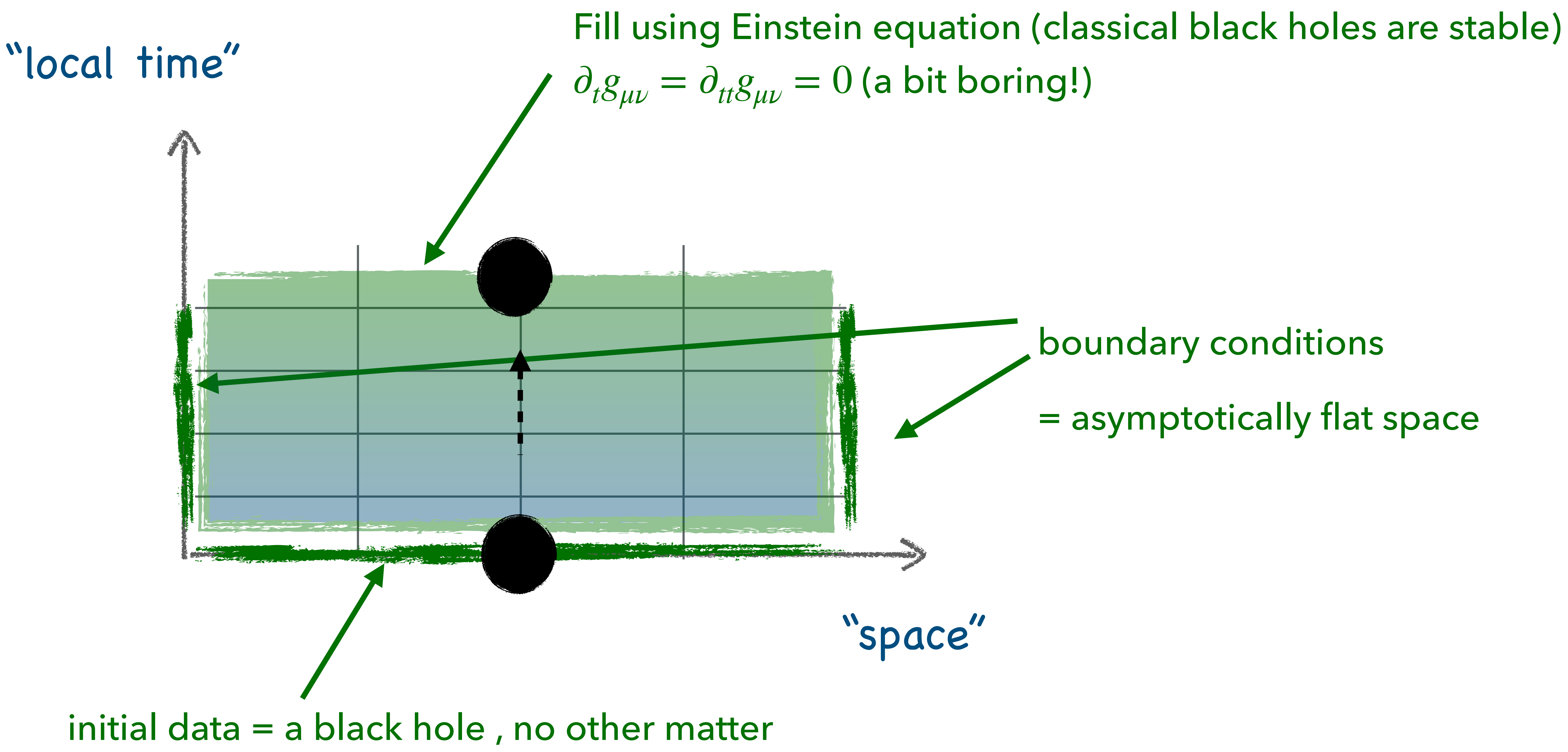


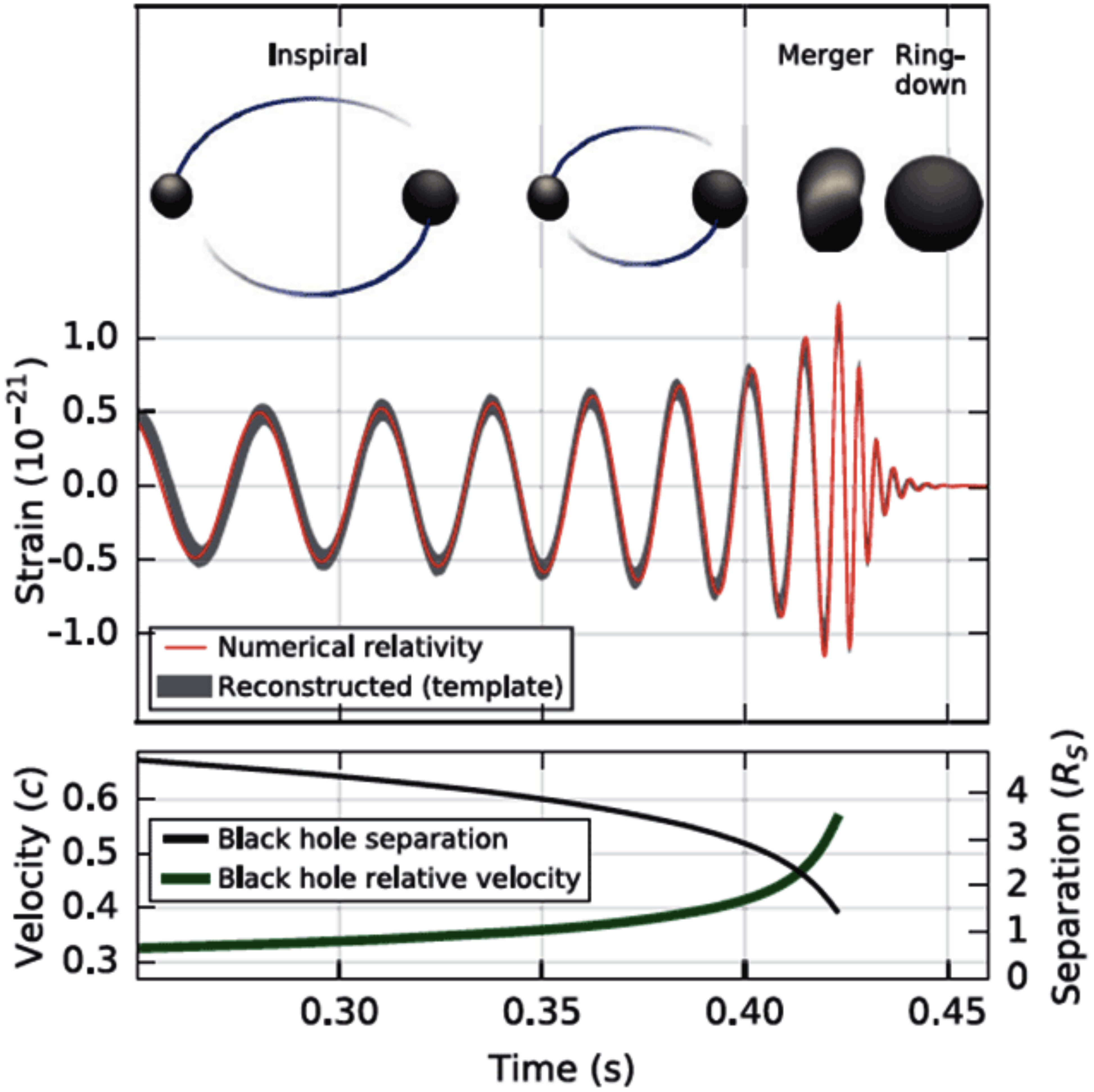
boundary conditions $(\partial_{xx} g_{ab}, \partial_x g_{ab}, g_{ab}, T_{ab})$

initial data $(\partial_t g_{ab}, g_{ab}, T_{ab})$ satisfying

$$\frac{\partial^2 g}{\partial x^2} + \text{non linear terms} = f(\text{energy, momentum})$$

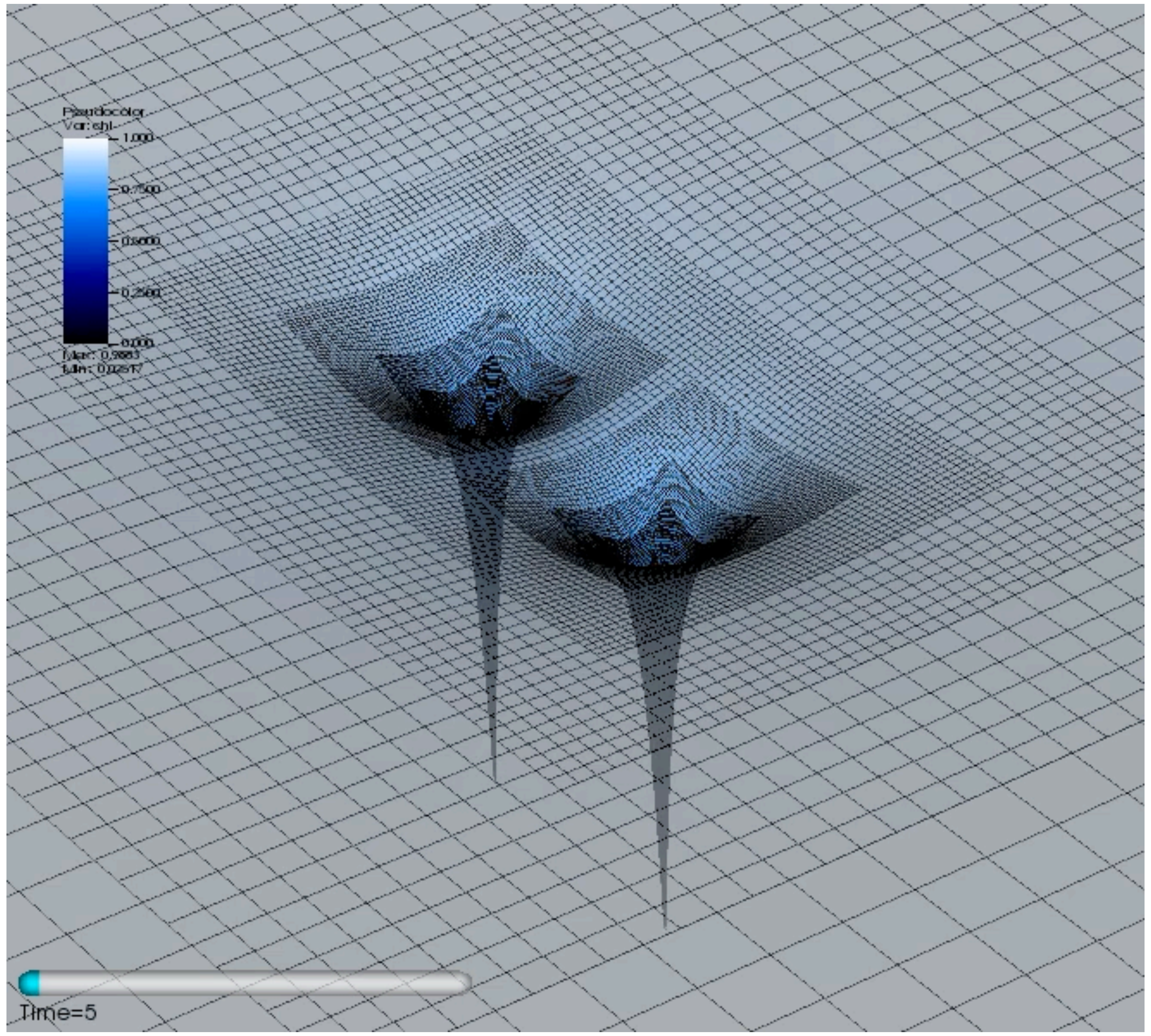
Numerical relativity





GW150914

t=14 September 2015, x = LIGO, Earth



(Roughly) $\frac{1}{det(g_{ab})}$

Topic of this session:
What are the possible mistakes one can make when setting up a numerical relativity simulation?

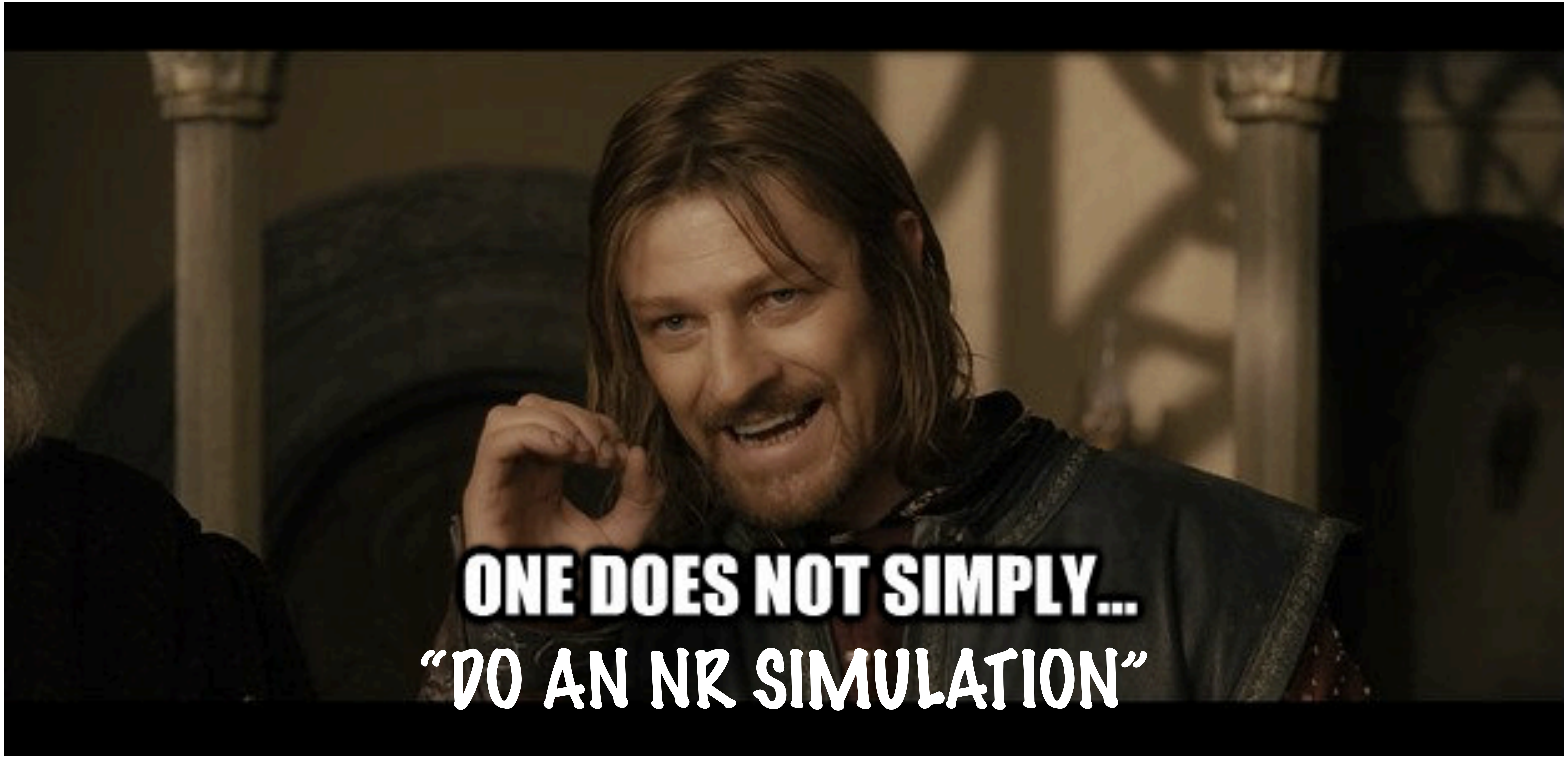
Specific questions we will address

1. Does the physical problem require NR?
2. How do I choose my initial data?
3. What is the right formulation
4. What is the right gauge?
5. How should I set the (many) parameters?
6. What diagnostics do I want to extract?
7. How can I be sure my simulation is correct?

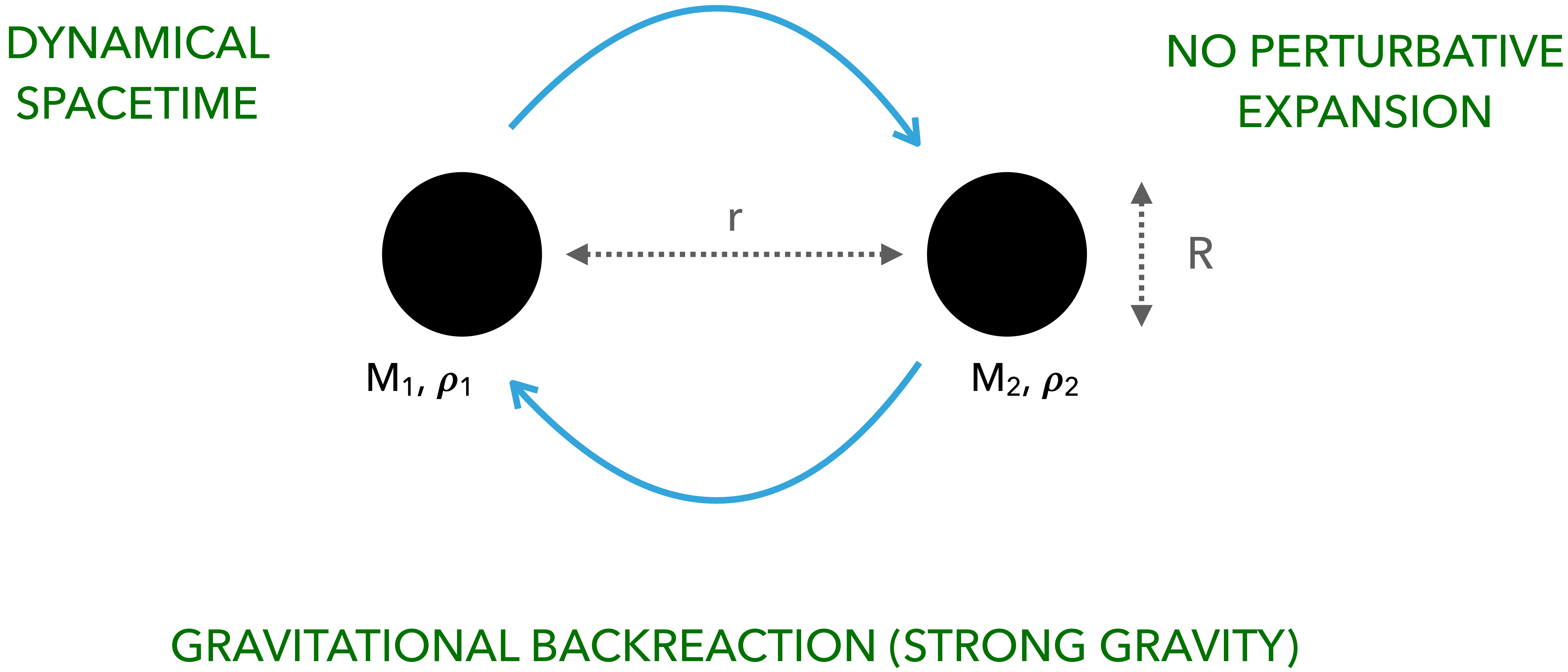
Q1: Does the physical problem require NR?

=

What research problems can I solve that no one else can?

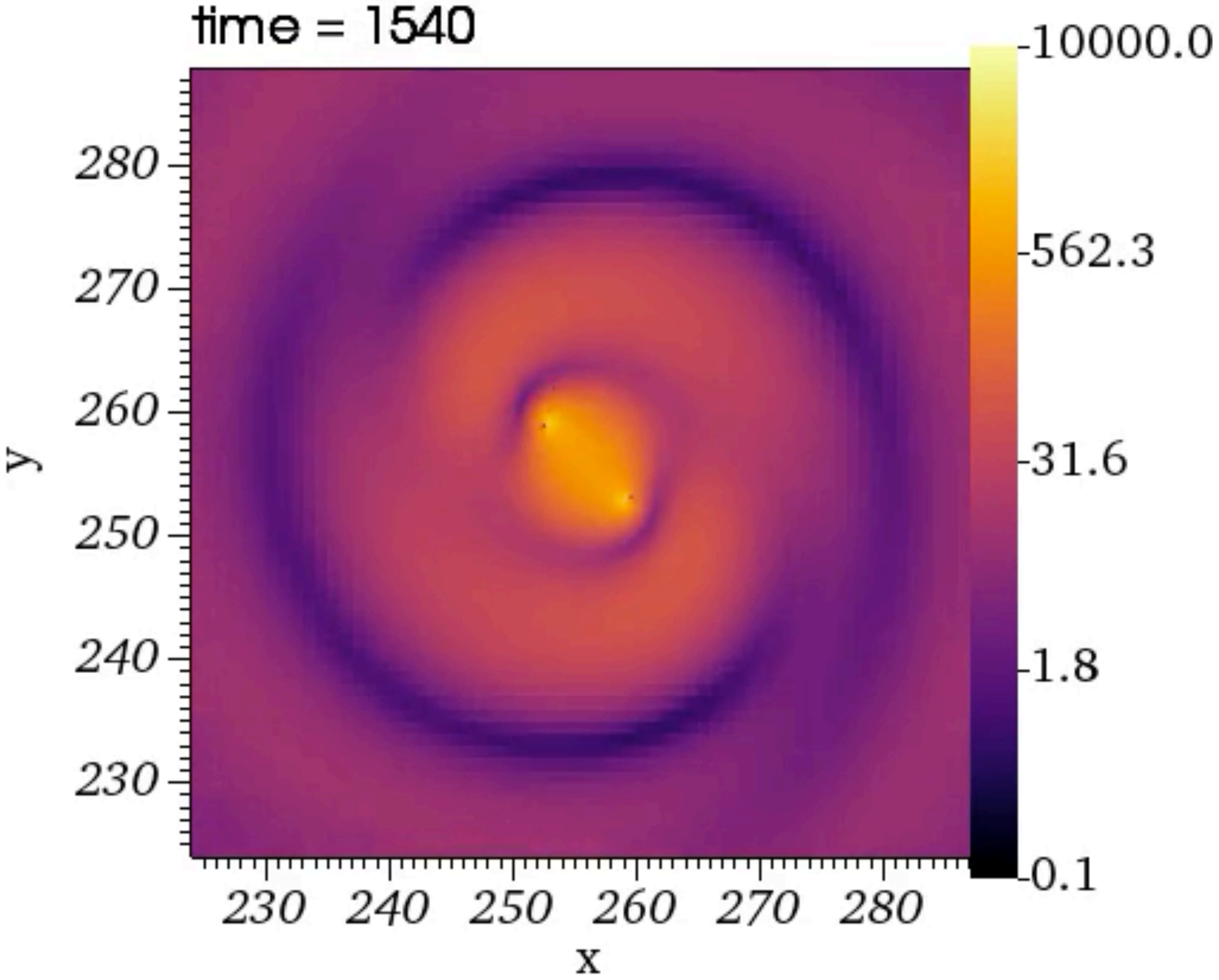

ONE DOES NOT SIMPLY...
"DO AN NR SIMULATION"

Q: When do we need numerical relativity?



e.g. black holes in low mass dark matter

energy density
of the dark matter

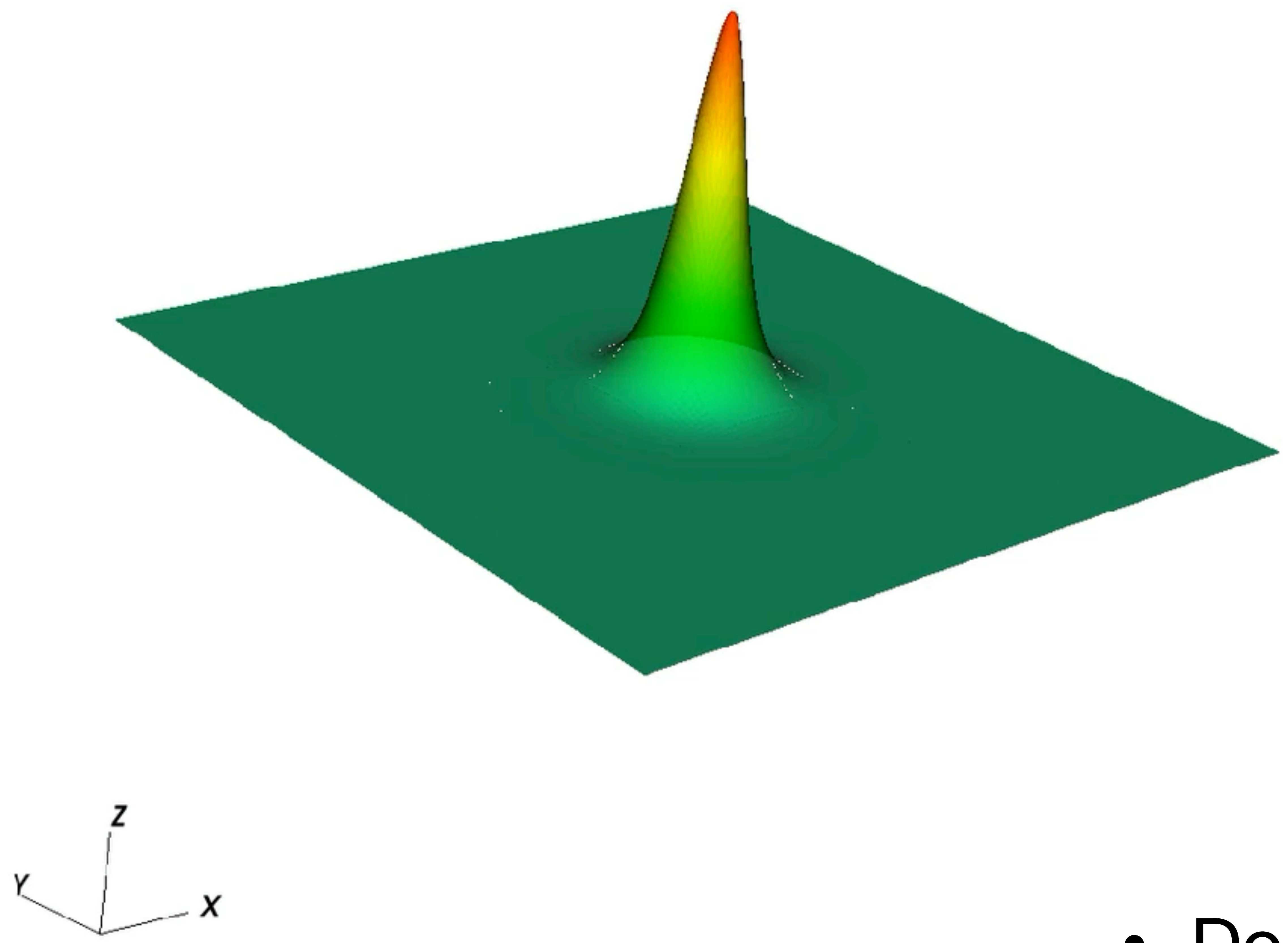


Matter waves extract energy and angular momentum from the binary and cause it to inspiral faster

Simulation credit: KC / J Bamber

BabyGRChombo problem - oscillatons

See Helfer et. al. 2016 (https://arxiv.org/abs/1609.04724)



- Do we need NR for this?

Q2: How do I choose my initial data?

=

What initial data can I (fairly easily) solve for?

```
#set gauge
lapse = 1.0
FOR(i) { shift[i] = 0.0 }

#set metric
chi = 1.0
FOR(i) { bar_gamma[i][i] = 1 }

#set matter
u = 10.0 * exp(-r*r)
        * r2

v = 0.0
```

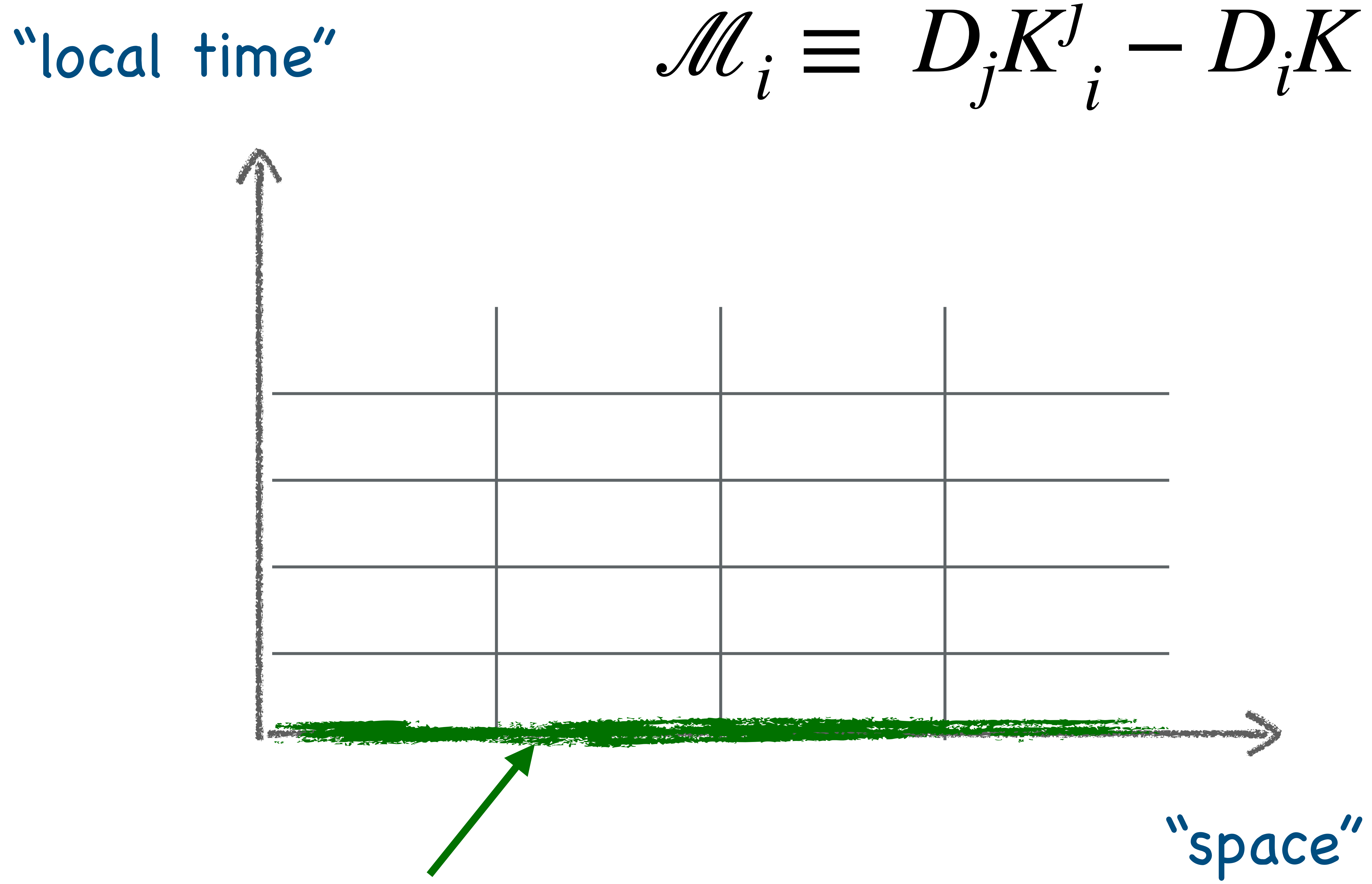
What is wrong here?

Initial conditions

Given (ρ, S_i) configuration, choose (γ_{ij}, K_{ij}) such that

$$\mathscr{H} \equiv {}^{(3)}R + K^2 + K_{ij}K^{ij} - 16\pi\rho = 0$$

$$\mathscr{M}_i \equiv D_j K^j{}_i - D_i K - 8\pi S_i = 0$$



initial data $(\partial_t g_{ab},\, g_{ab},\, T_{ab})$ satisfying

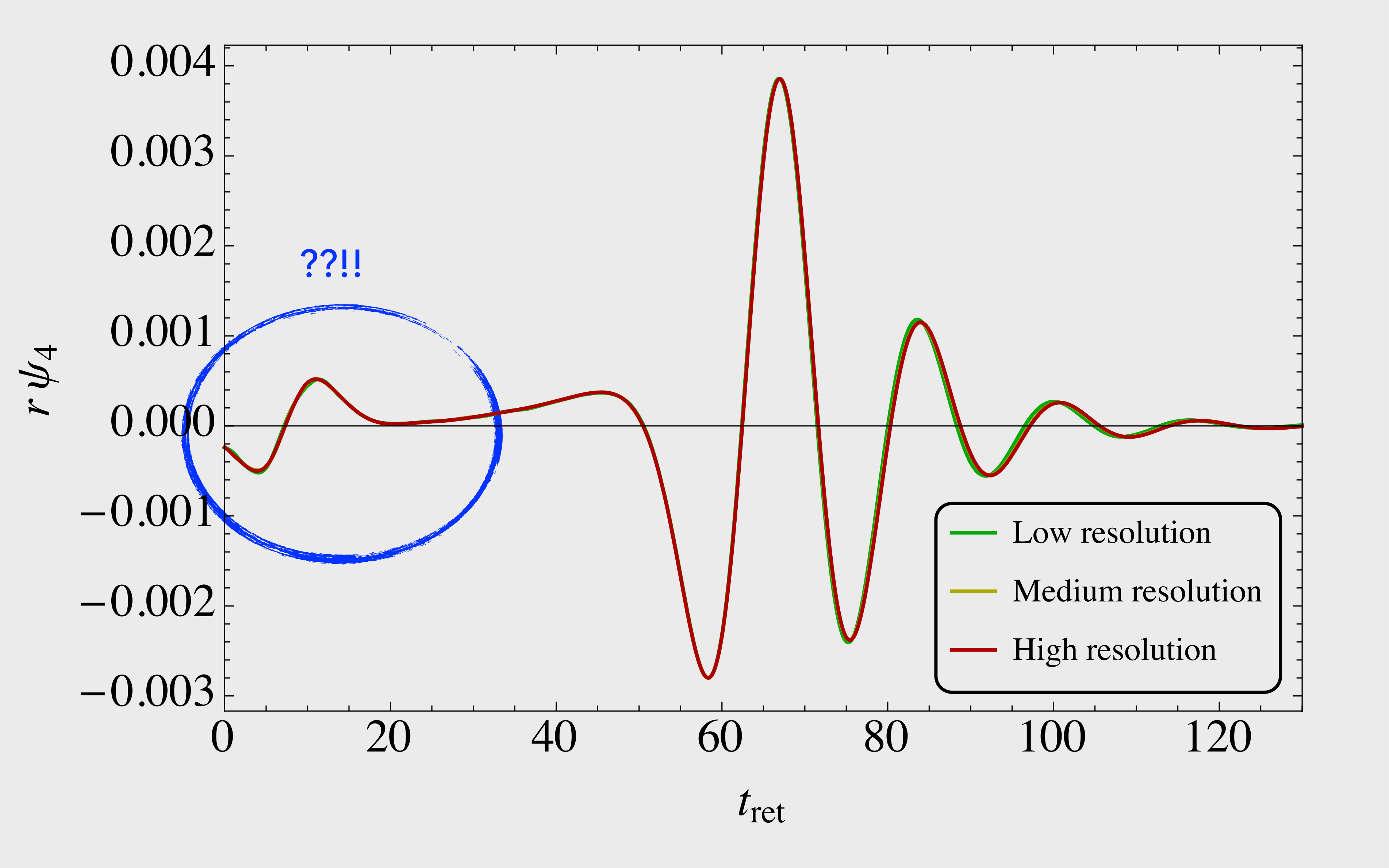
$$\frac{\partial^2 g}{\partial x^2} + \text{non linear terms} = f(\text{energy, momentum})$$

Counting degrees of freedom

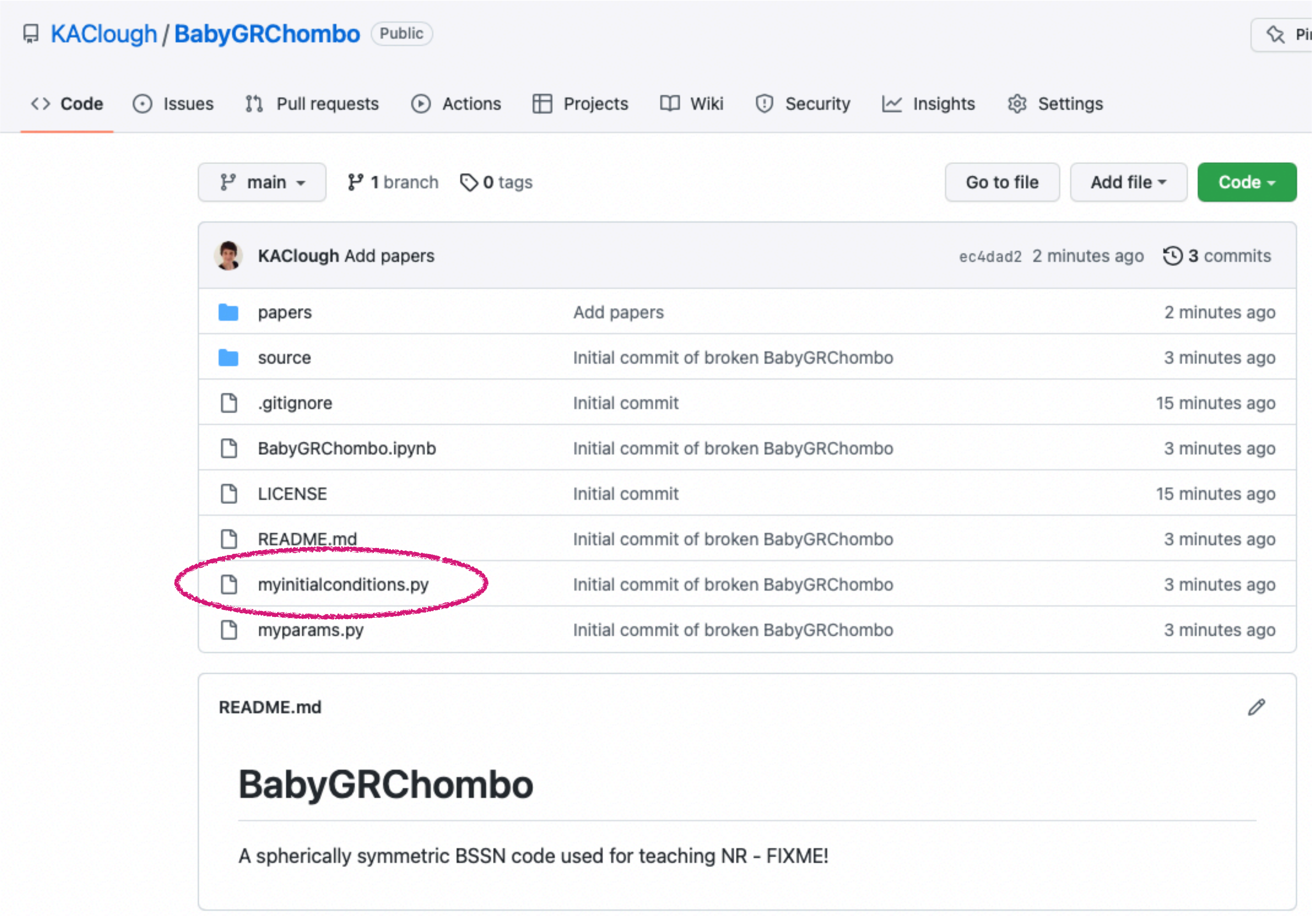
Given (ρ, S_i) configuration, choose (γ_{ij}, K_{ij}) - for now treat lapse and shift as gauge params

- 6 + 6 components to be chosen
- 4 constraints
- 8 remaining = 2 x 2 physical degrees of freedom (GW polarisations) + 4 coordinate choices

Not usually obvious how to separate these…



BabyGRChombo initial conditions



BabyGRChombo initial conditions

110 lines (88 sloc) | 4.34 KB

```python
# myinitialconditions.py

# set the initial conditions for all the variables

from myparams import *
from source.uservariables import *
from source.tensoralgebra import *
from source.fourthorderderivatives import *
import numpy as np
from scipy.interpolate import interp1d

def get_initial_vars_values() :

    initial_vars_values = np.zeros(NUM_VARS * N)

    # Use oscilloton data to construct functions for the vars
    grr0_data    = np.loadtxt("source/initial_data/grr0.csv")
    lapse0_data  = np.loadtxt("source/initial_data/lapse0.csv")
    v0_data      = np.loadtxt("source/initial_data/v0.csv")

    # set up grid in radial direction in areal polar coordinates
    dR = 0.01;
    length = np.size(grr0_data)
    R = np.linspace(0, dR*(length-1), num=length)
    f_grr   = interp1d(R, grr0_data)
    f_lapse = interp1d(R, lapse0_data)
    f_v     = interp1d(R, v0_data)

    for ix in range(num_ghosts, N-num_ghosts) :

        # position on the grid
        r_i = r[ix]
```

- Interpolates from some (constraint satisfying) data which was solved for using a shooting method
- This data is generated in areal polar gauge $ds^2 = \alpha^2 dt^2 + g_{rr} dr^2 + r^2 d\Omega^2$ but has to be converted into the appropriate form for the reference metric
- The scalar field is initially at u=0 everywhere but with non zero conjugate momentum v
- What could be wrong here?

Q3: What is the right formulation?

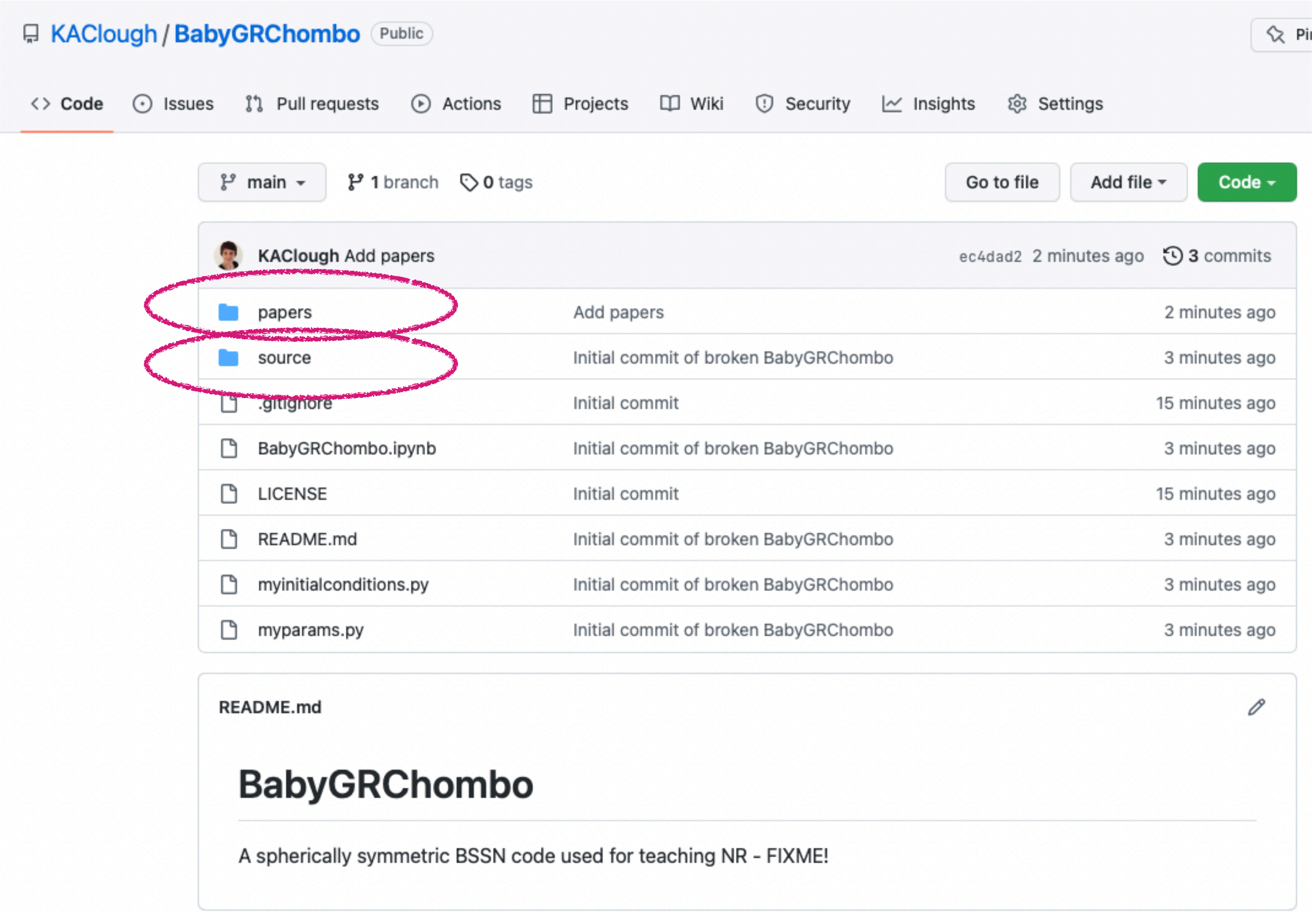
=

What is the formulation in the code I am using? Can I read the code?

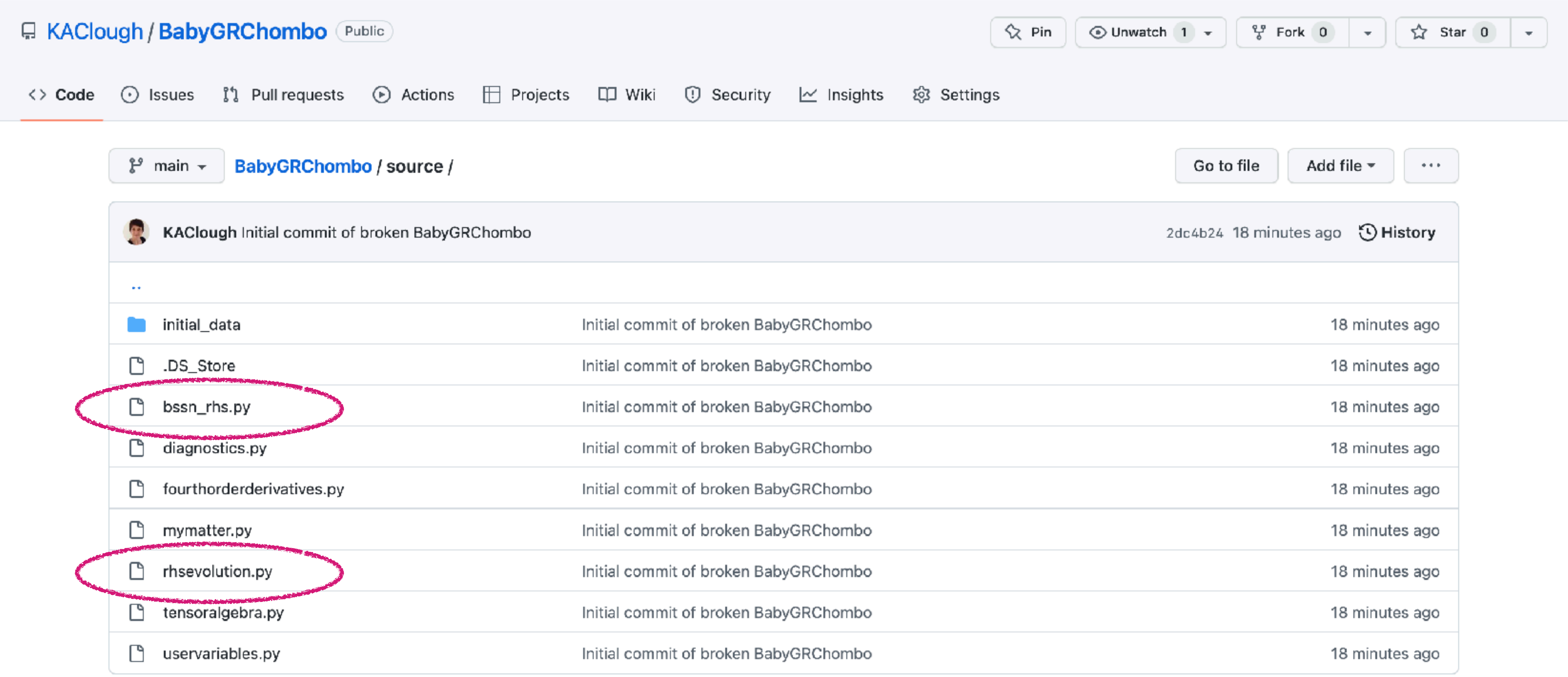
```
#Calculate bg eqn 4
a = 2.0 * K -
        3.0/2.0 * st

b = {1.0, 0, pi/3.0}

out = a * b[2]
      - a * b[0]
      - 3.0 * b[1]

out2 = myfunction
        (a, b, 67.3)
```

What is wrong here?

BabyGRChombo formulation



BabyGRChombo formulation



BabyGRChombo formulation

97 lines (72 sloc) | 4.23 KB

Raw | Blame

```python
# bssn_rhs.py
# as in Etienne https://arxiv.org/abs/1712.07658v2
# see also Baumgarte https://arxiv.org/abs/1211.6632 for the eqns with matter

import numpy as np
from myparams import *
from source.tensoralgebra import *

def get_rhs_phi(lapse, K, bar_div_shift) :

    dphidt = (- one_sixth * lapse * K
              + one_sixth * bar_div_shift)

    return dphidt

def get_rhs_h(r_here, r_gamma_LL, lapse, traceA, bar_div_shift, hat_D_shift, a) :

    dhdt = np.zeros_like(rank_2_spatial_tensor)
    inv_scaling = np.array([1.0, 1.0/r_here , 1.0/r_here/sintheta])
    for i in range(0, SPACEDIM):
        for j in range(0, SPACEDIM):

            # note that trace of \bar A_ij = 0 is enforced dynamically using the first term
            # as in Etienne https://arxiv.org/abs/1712.07658v2 eqn (11a)
            # recall that h is the rescaled quantity so we need to scale
            dhdt[i][j] += ( two_thirds * r_gamma_LL[i][j] * (lapse * traceA - bar_div_shift)
                           + inv_scaling[i] * inv_scaling[j] * (hat_D_shift[i][j] + hat_D_shift[j][i])
                           - 2.0 * lapse * a[i][j])

    return dhdt
```

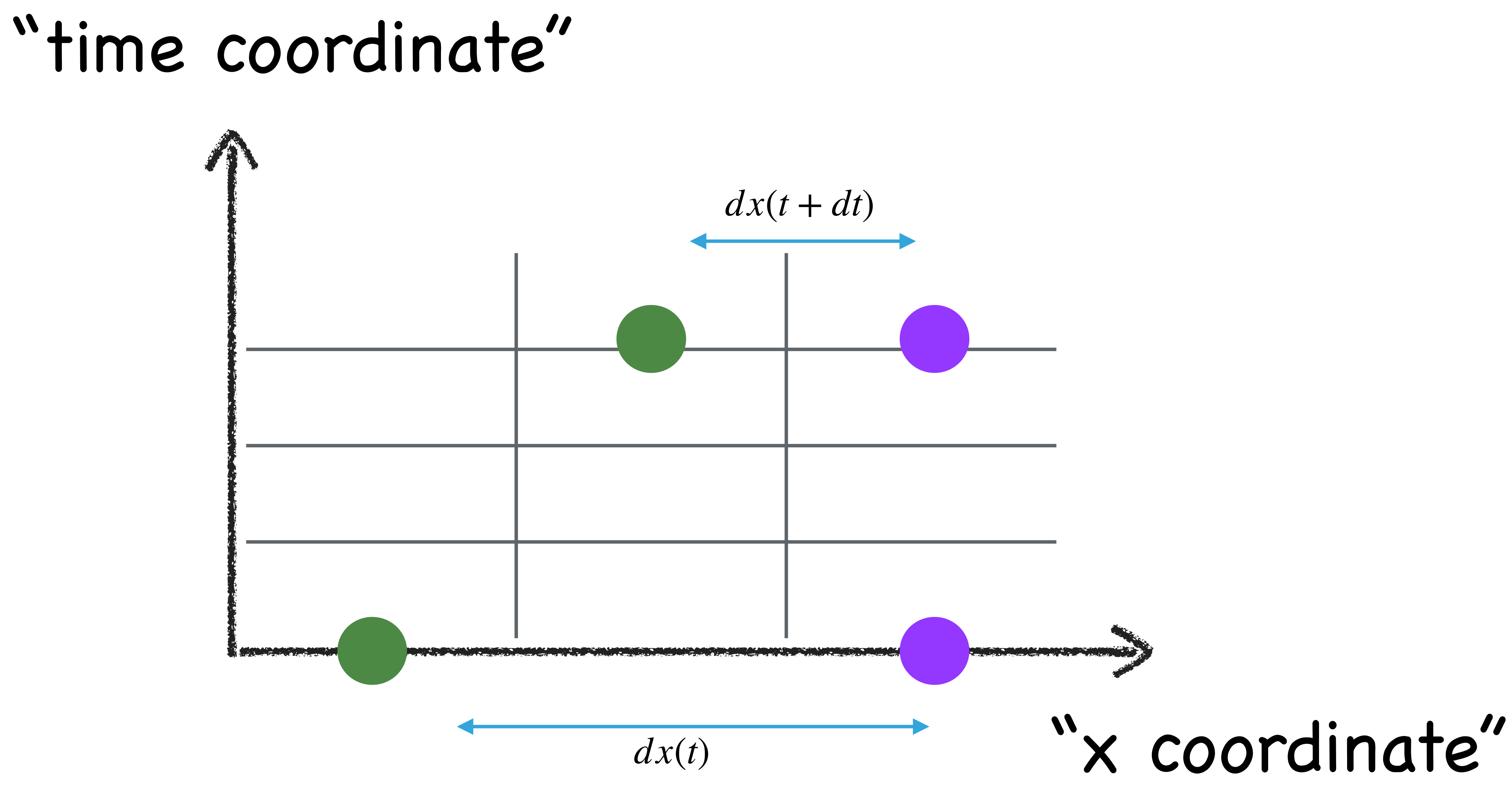
- This is probably where the bugs are!
- Will need to check the equations to the papers provided
- Uses the reference metric approach for spherical symmetry which scales the tensors to remove singularities (don't worry too much about this, just check the equations!)
- Do you find the naming helpful? What would you change?

Q4: What is the right gauge?

=

Do I understand my evolving coordinates?

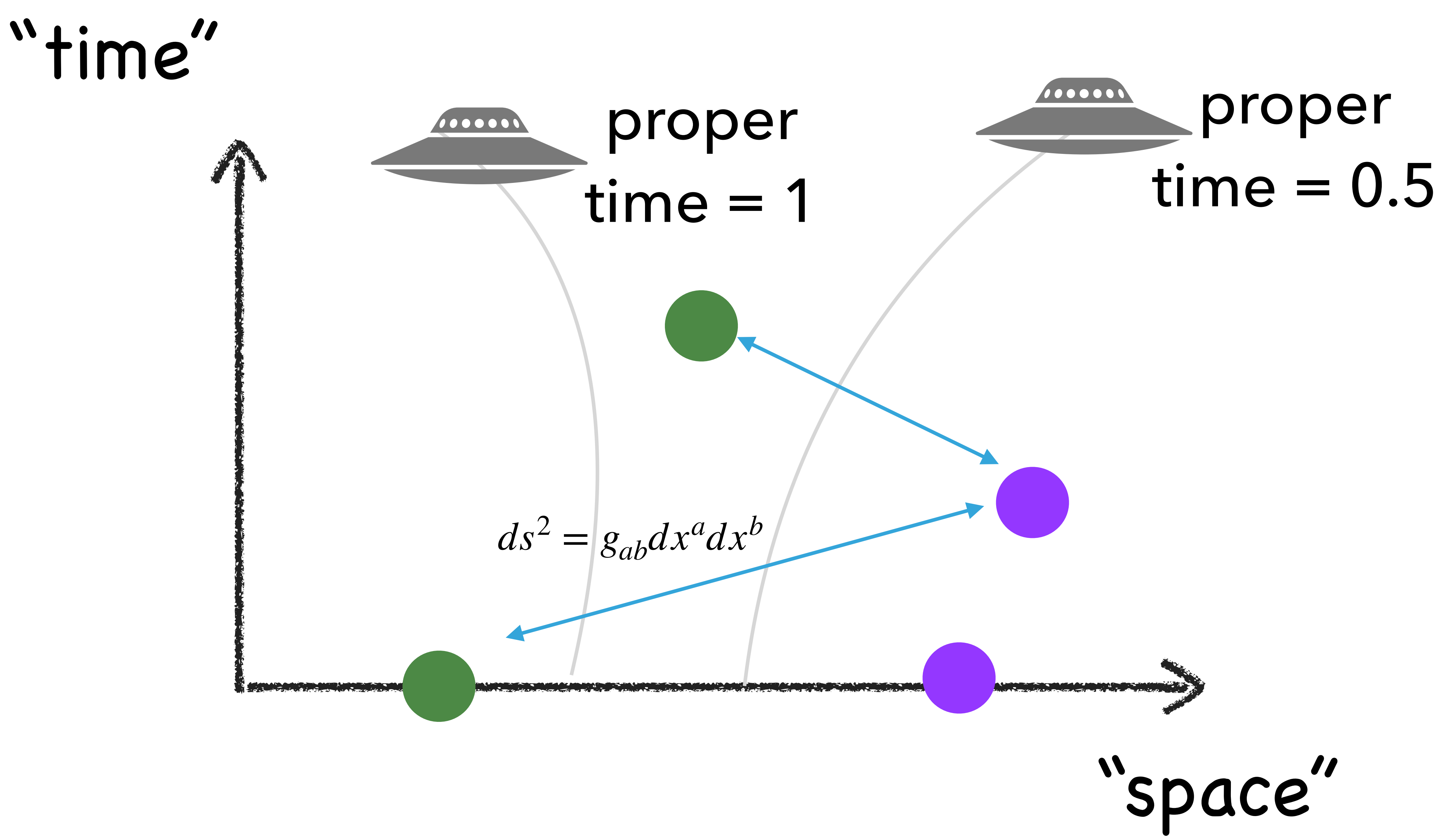
CODE VIEW



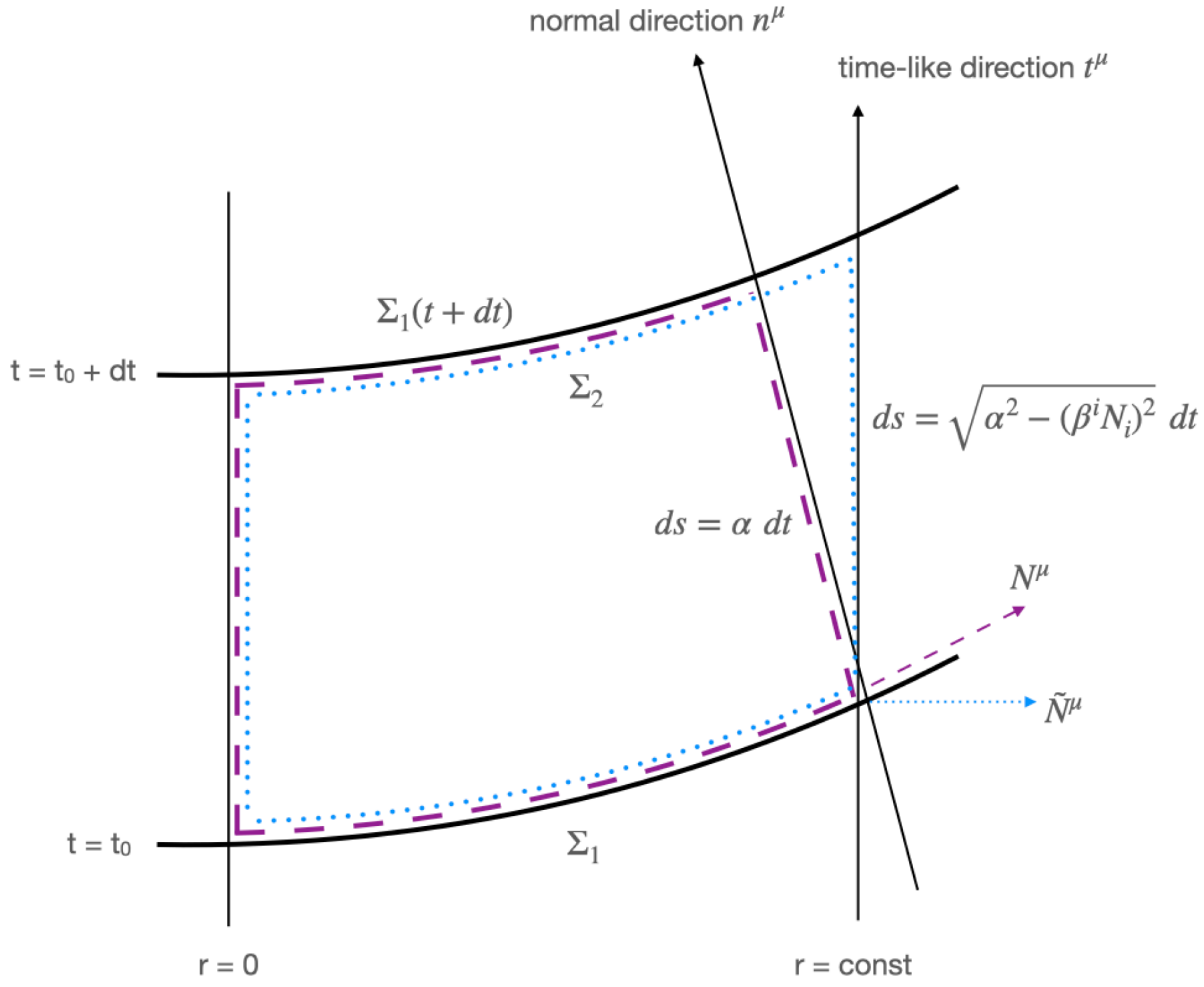
What is wrong here?

“My two objects have got closer together over time”

"PHYSICAL" VIEW?



What do “fixed" coordinates mean in a puncture-like gauge?



BabyGRChombo gauge

```
        rhs_a           = get_rhs_a(a, bar_div_shift, lapse[ix], K[ix], em4phi, bar_Rij,
                                    r_here, Delta_ULL, bar_gamma_UU, bar_A_UU, bar_A_LL,
                                    d2phidx2[ix], dphidx[ix], d2lapsedx2[ix], dlapsedx[ix],
                                    h, dhdr, d2hdr2, matter_Sij)

        rhs_lambdar[ix] = get_rhs_lambdar(hat_D2_shift, Delta_U, Delta_ULL, bar_div_shift,
                                          bar_D_div_shift, bar_gamma_UU, bar_A_UU, lapse[ix], dlapsedx[ix],
                                          dphidx[ix], dKdx[ix], matter_Si)

        # Add advection to time derivatives
        rhs_phi[ix]     += shiftr[ix] * dphidx[ix]
        rhs_hrr[ix]     = rhs_h[i_r][i_r] + shiftr[ix] * dhrrdx[ix] - 2.0 * hrr[ix] * dshiftrdx[ix]
        rhs_htt[ix]     = rhs_h[i_t][i_t] + shiftr[ix] * dhttdx[ix]
        rhs_hpp[ix]     = rhs_h[i_p][i_p] + shiftr[ix] * dhppdx[ix]
        rhs_K[ix]       += shiftr[ix] * dKdx[ix]
        rhs_arr[ix]     = rhs_a[i_r][i_r] + shiftr[ix] * darrdx[ix] - 2.0 * arr[ix] * dshiftrdx[ix]
        rhs_att[ix]     = rhs_a[i_t][i_t] + shiftr[ix] * dattdx[ix]
        rhs_app[ix]     = rhs_a[i_p][i_p] + shiftr[ix] * dappdx[ix]
        rhs_lambdar[ix] += shiftr[ix] * dlambdardx[ix] - lambdar[ix] * dshiftrdx[ix]

        # Set the gauge vars rhs
        rhs_br[ix]      = 0.75 * rhs_lambdar[ix] - eta * br[ix]
        rhs_shiftr[ix] = br[ix]
        rhs_lapse[ix]  = - 2.0 * lapse[ix] * K[ix] + shiftr[ix] * dlapsedx[ix]
```

- Using standard puncture gauge
- Will we see any gauge evolution?
- What could be wrong here?

Q5: How should I set the parameters

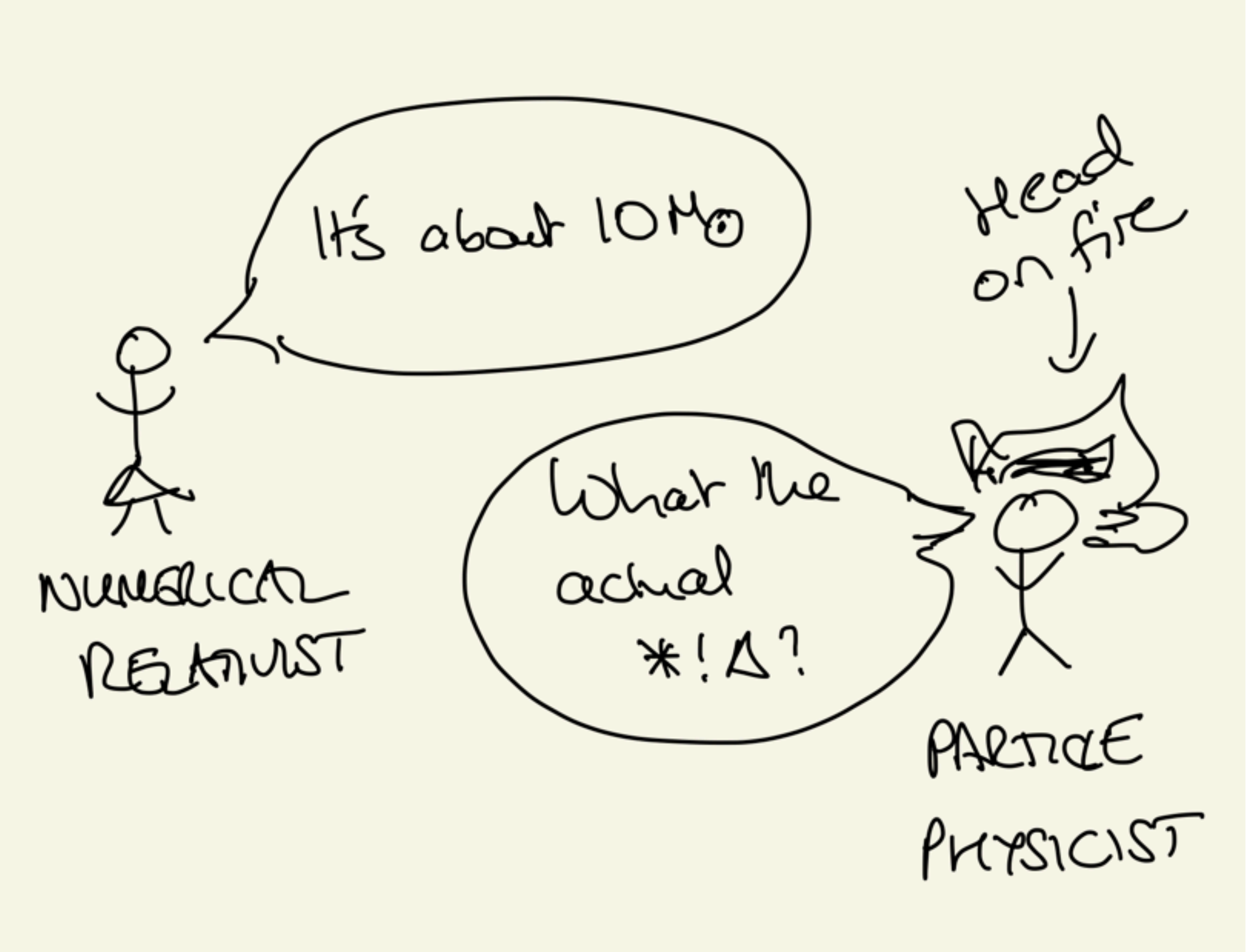
=

What are the units?

```
# My params
Mass_BH1 = 0.5
Mass_BH2 = 0.5
scalar_field_mass = 1.0

# gauge params
eta = 2e3
kappa1 = 0.5
```

What is wrong here?

What is the separation of the two black holes in your simulation?

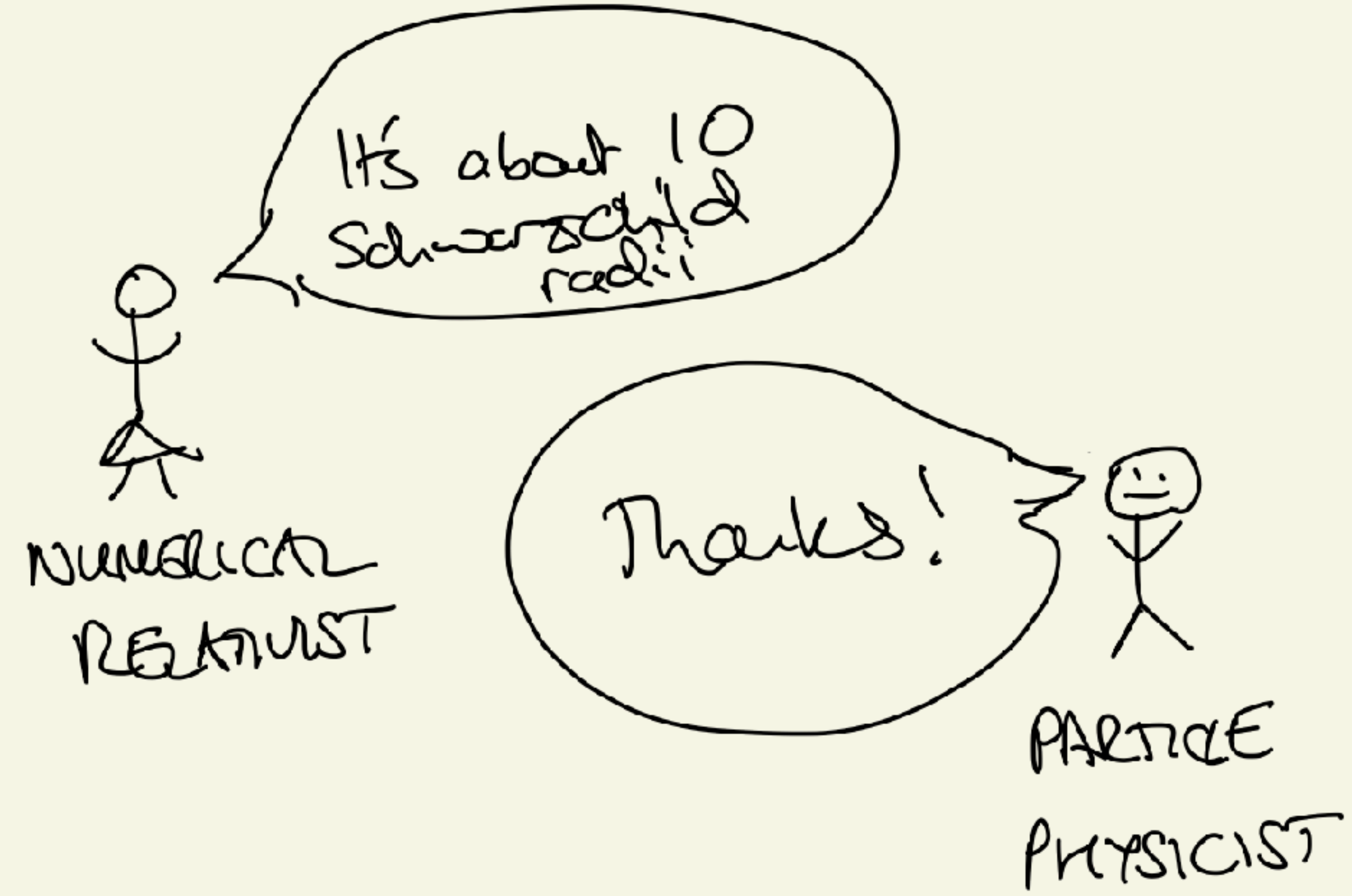


$G = c = 1$ for NR

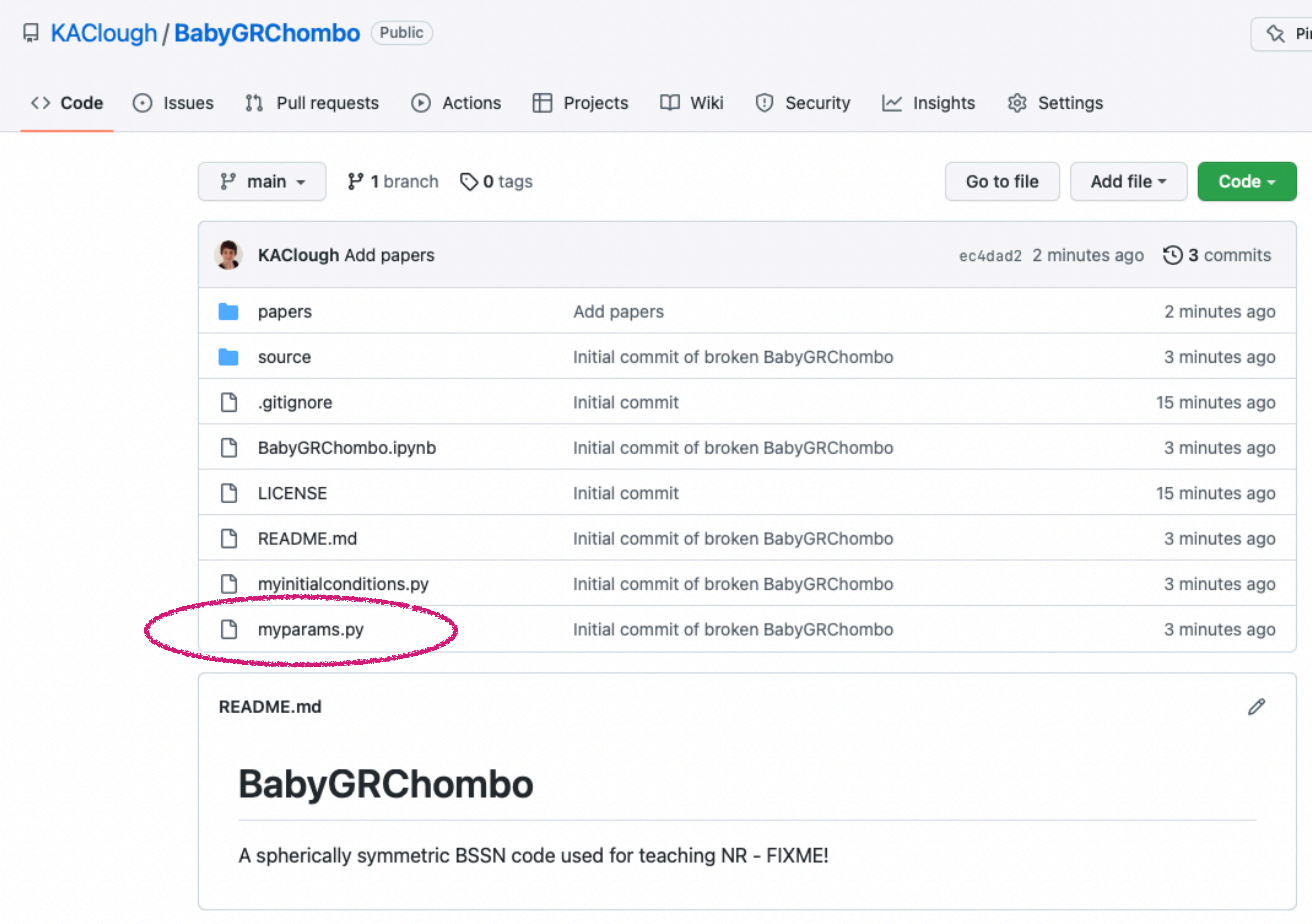
THERE IS NO $\hbar$ in GR

If we set $M = M_{pl}$ then $\hbar$ is 1, but usually $\hbar \neq 1$, because $M = M_\odot$.

Usually we are describing a "curvature radius" not a mass.



BabyGRChombo params



BabyGRChombo params

28 lines (23 sloc) | 922 Bytes

Raw Blame

```
# myparams.py

# specify the params that are fixed throughout the evolution

import numpy as np

# Input parameters for grid and evolution here
N_r = 120 # num points on physical grid
N_t = 101 # time resolution (only for outputs, not for integration)
R = 60 # Maximum outer radius
T = 3.0 # Maximum evolution time

# coefficients for bssn and gauge evolution
eta = 1.0 # 1+log slicing damping coefficient
sigma = 1.0 # kreiss-oliger damping coefficient
eight_pi_G = 8.0 * np.pi * 1.0 # Newtons constant, we take G=c=1
scalar_mu = 1.0 # this is an inverse length scale related to the scalar compton wavelength

# These values are hardcoded or calculated from the inputs above
# so should not be changed
dx = R/N_r
dt = T/N_t
num_ghosts = 3
N = N_r + num_ghosts * 2
r = np.linspace(-(num_ghosts-0.5)*dx, R+(num_ghosts-0.5)*dx, N)
t = np.linspace(0, T-dt, N_t)
oneoverdx  = 1.0 / dx
oneoverdxsquared = oneoverdx * oneoverdx
```

- Check you know what all of these are…
- Hard coding should be avoided!
- Are the values reasonable?
- Why do we predefine some values like oneoverdx?

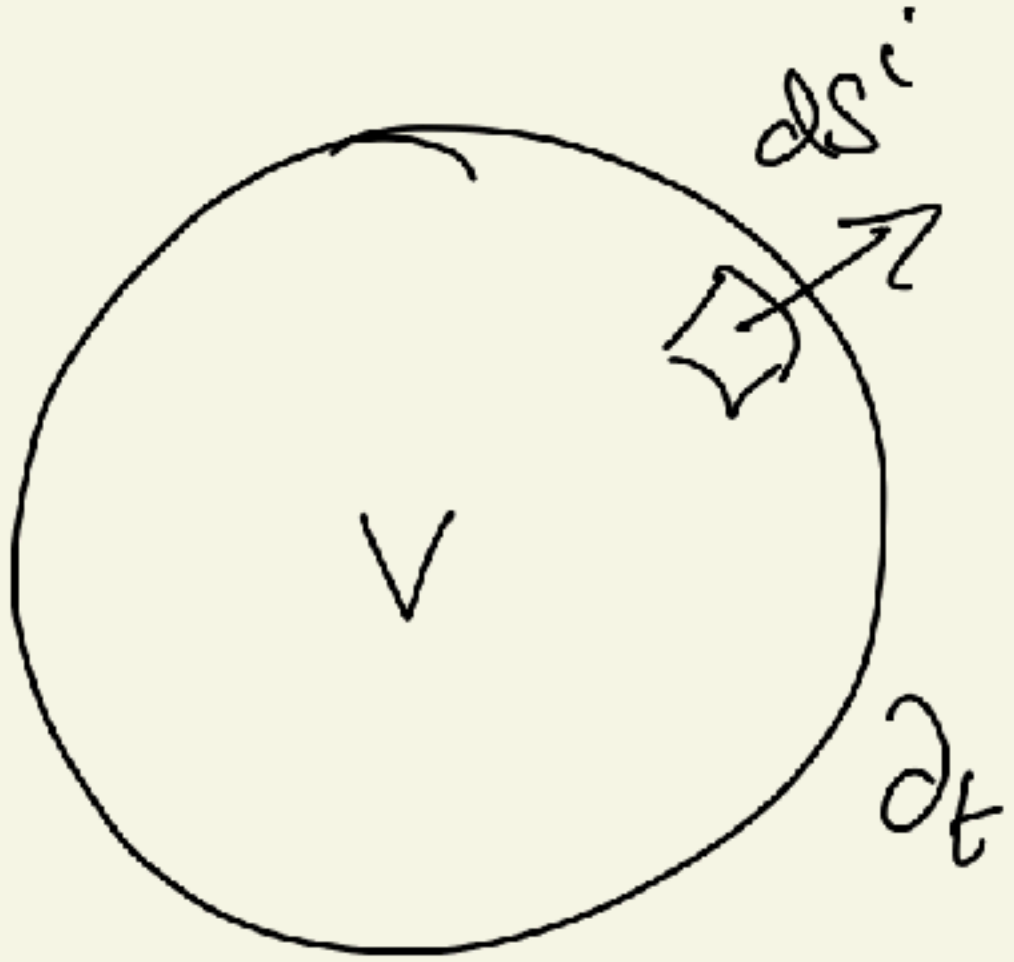
Q6: What diagnostics should I extract?

=

What does this all mean?

Diagnostics

$$\mathrm{diag}(t) = \int_0^L\int_0^L\int_0^L \rho(t)\, dx\, dy\, dz$$

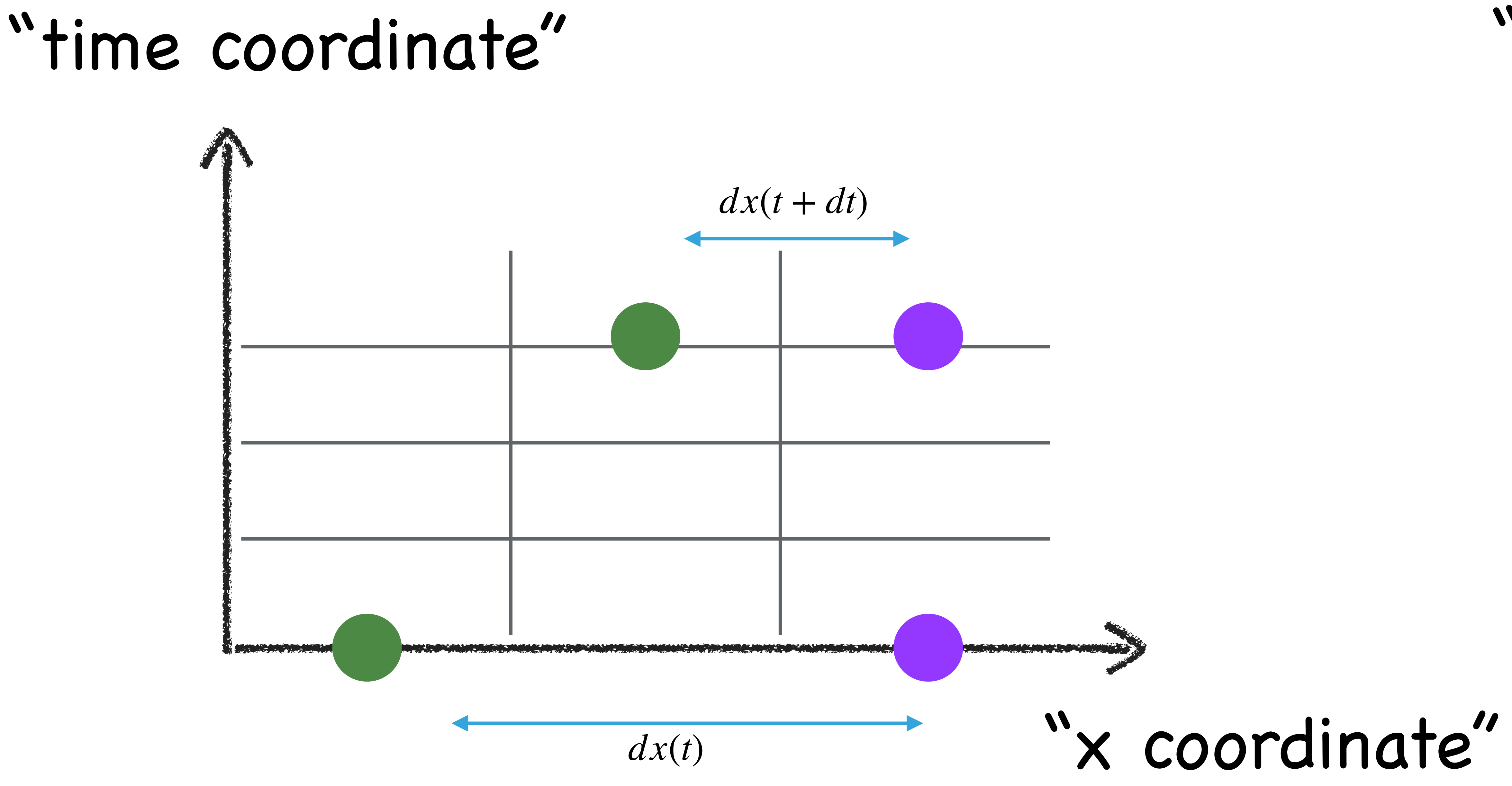


$$\int T_i^{\,0}\, dS^i = 0$$

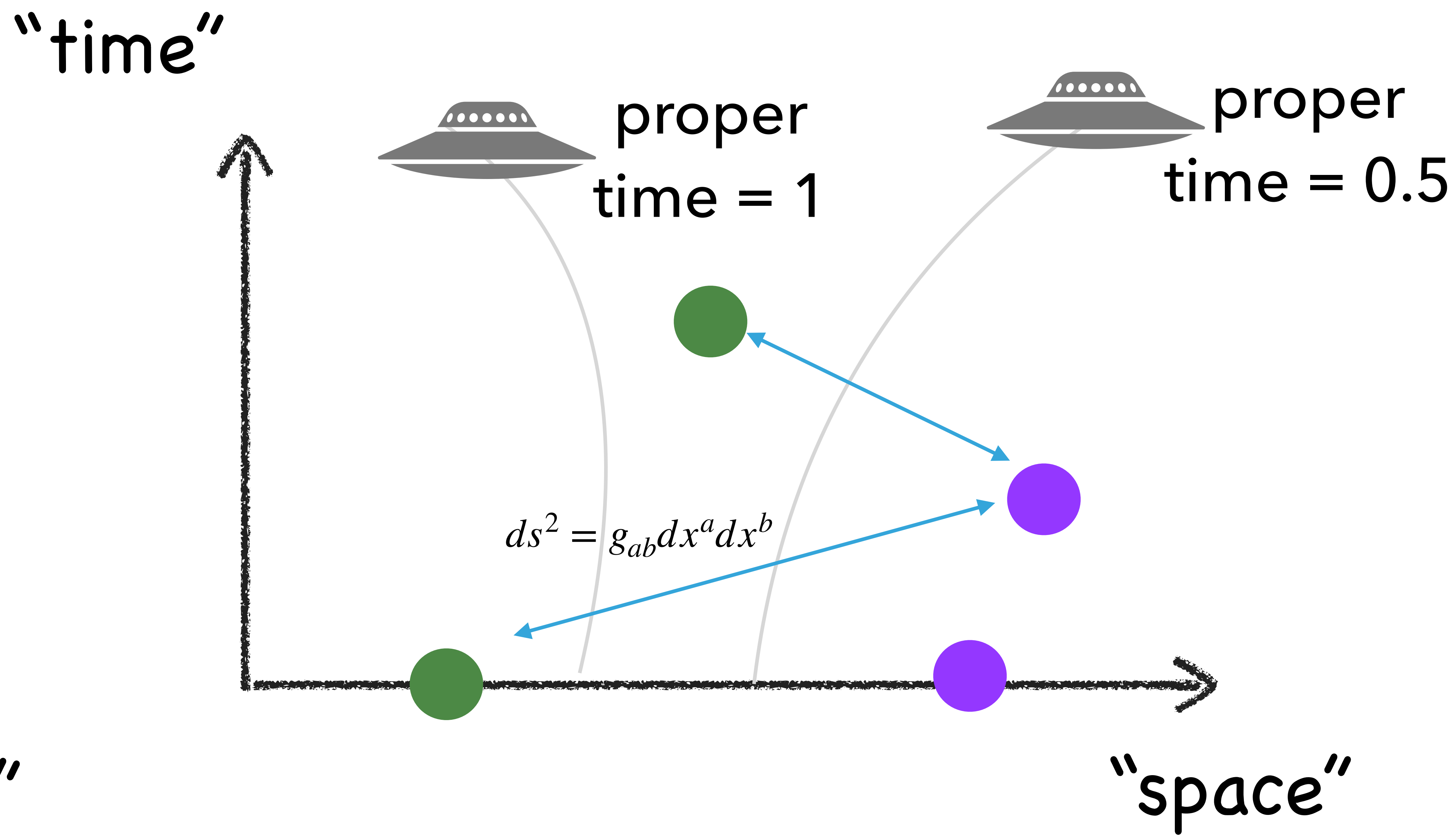
$$\partial_t \int \rho\, dV = 0$$

What is wrong here?

CODE VIEW



"PHYSICAL" VIEW?



Useful diagnostics

Classical and Quantum Gravity

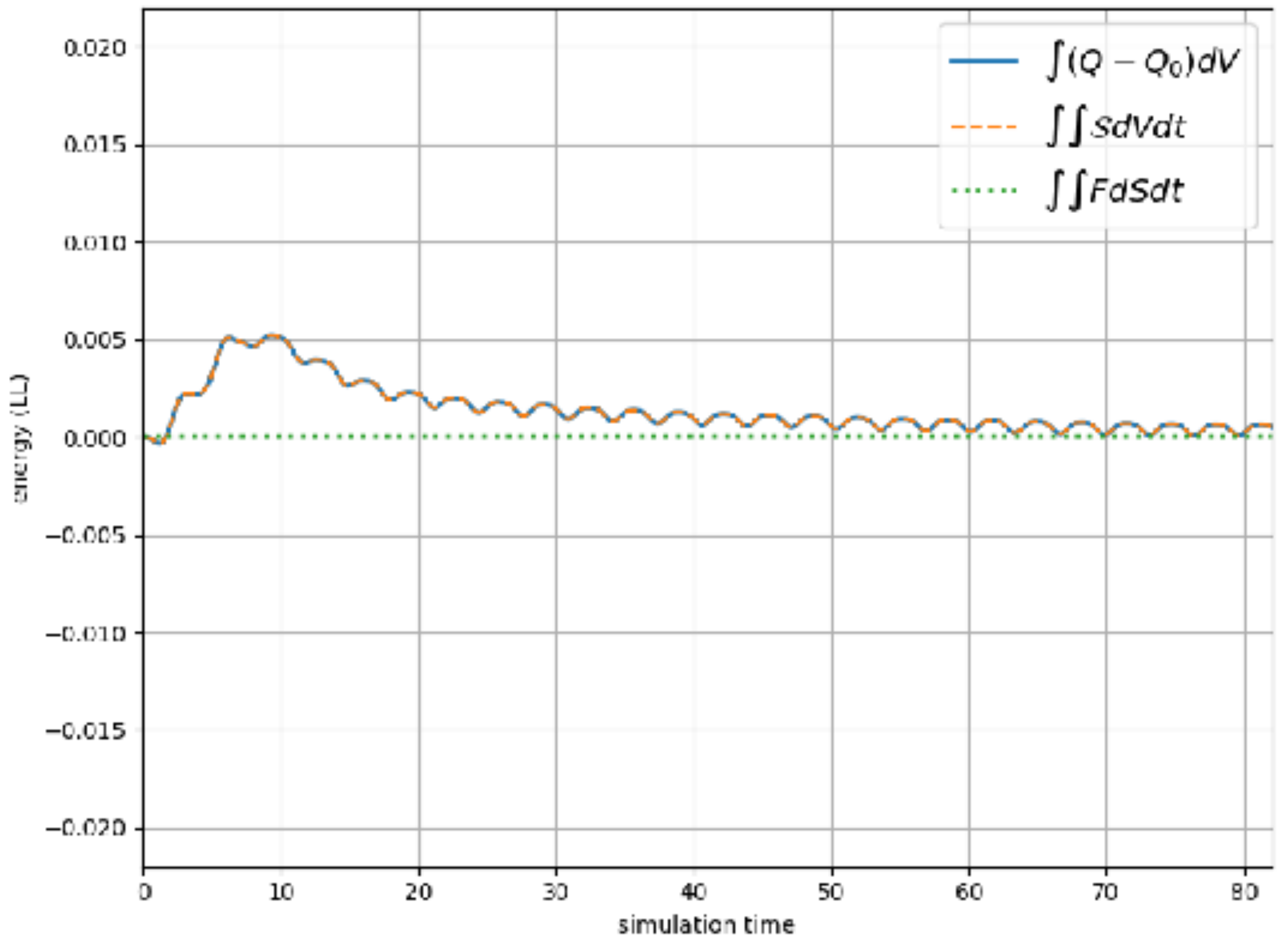
NOTE

Continuity equations for general matter: applications in numerical relativity

Katy Clough[2,1]

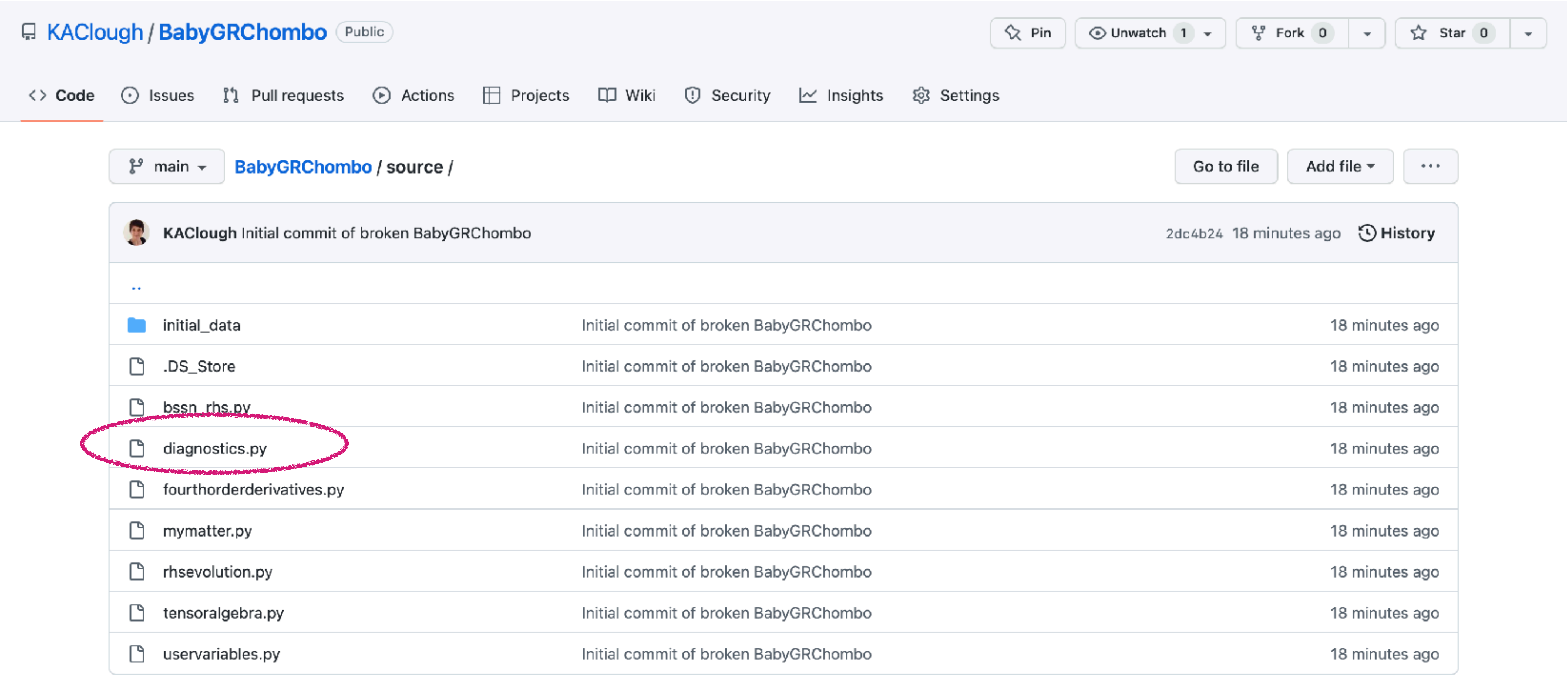
Published 23 July 2021 • © 2021 IOP Publishing Ltd
Classical and Quantum Gravity, Volume 38, Number 16
Citation Katy Clough 2021 *Class. Quantum Grav.* **38** 167001

References ▾ Open science ▾



- Anything extracted at asymptotically flat infinity!
- Constraint violation in the "*area of physical interest*"
- All the contributions to the conserved quantities in matter charges (helps identify gravitational "forces")
- …

BabyGRChombo diagnostics



BabyGRChombo diagnostics

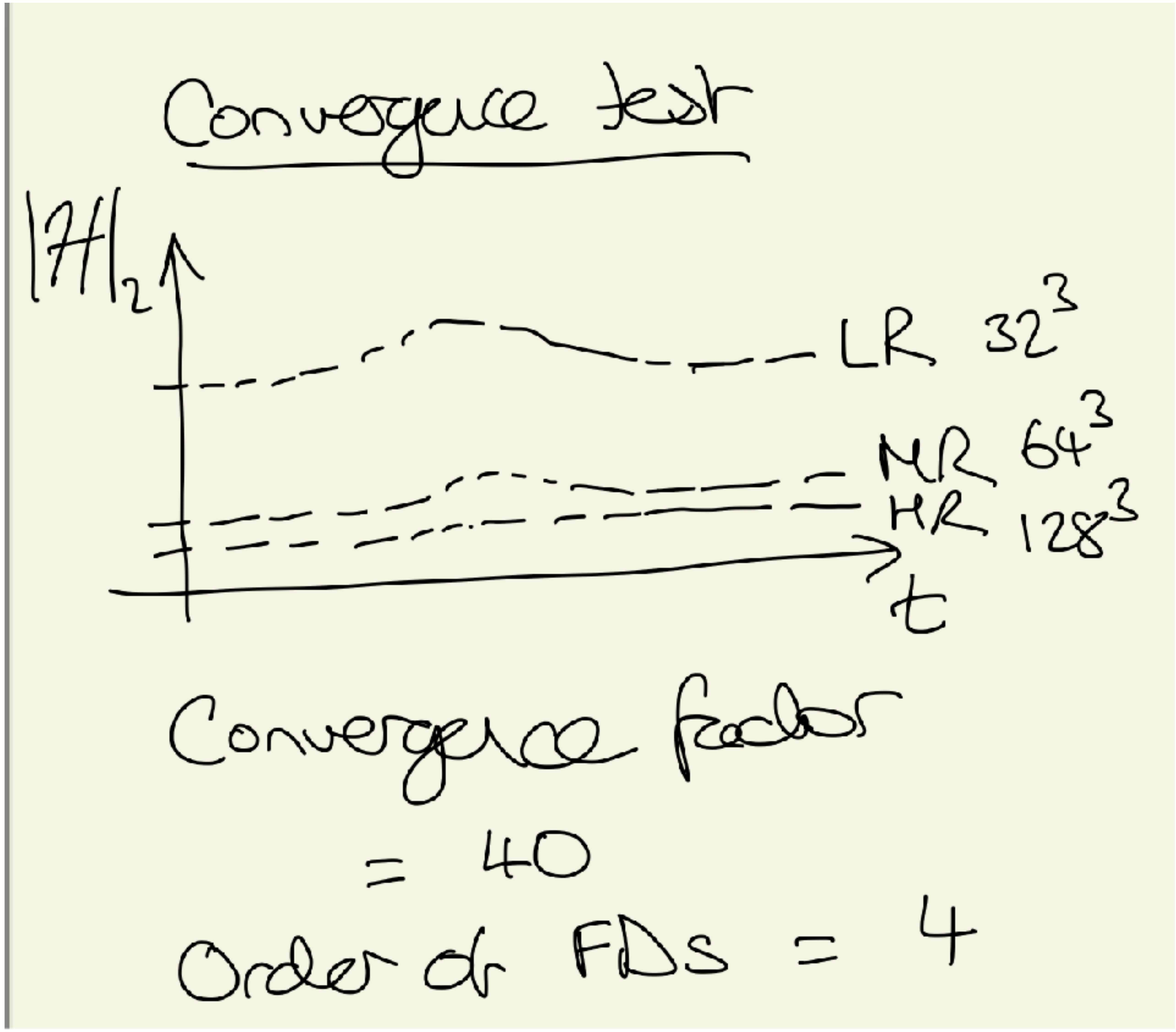
```
# The connections Delta^i, Delta^i_jk and Delta_ijk
Delta_U, Delta_ULL, Delta_LLL  = get_connection(r_here, bar_gamma_UU, bar_gamma_LL, h, dhdr)
bar_Rij = get_ricci_tensor(r_here, h, dhdr, d2hdr2, lambdar[ix], dlambdardx[ix],
                           Delta_U, Delta_ULL, Delta_LLL, bar_gamma_UU, bar_gamma_LL)
bar_R   = get_trace(bar_Rij, bar_gamma_UU)

# Matter sources
matter_rho            = get_rho( u[ix], dudx[ix], v[ix], bar_gamma_UU, em4phi )
matter_Si             = get_Si(  u[ix], dudx[ix], v[ix], bar_gamma_UU, em4phi )
matter_S, matter_Sij  = get_Sij( u[ix], dudx[ix], v[ix], bar_gamma_UU, em4phi,
                                 bar_gamma_LL)

# End of: Calculate some useful quantities, now start diagnostic
#################################################################

# Get the Ham constraint eqn (13) of Baumgarte https://arxiv.org/abs/1211.6632
Ham_i[ix] = (  two_thirds * K[ix] * K[ix] - trace_A2
             + em4phi * ( bar_R
                         - 8.0 * bar_gamma_UU[i_r][i_r] * (dphidx[ix] * dphidx[ix]
                                                          + d2phidx2[ix])
                         + 8.0 * bar_gamma_UU[i_t][i_t] * flat_chris[i_r][i_t][i_t] * dphidx[ix]
                         + 8.0 * bar_gamma_UU[i_p][i_p] * flat_chris[i_r][i_p][i_p] * dphidx[ix]
                         + 8.0 * Delta_U[i_r] * dphidx[ix])
             - 2.0 * eight_pi_G * matter_rho )

#print("bar_R is ", bar_R)
```

- Currently only the Hamiltonian constraint is implemented
- What other quantities would be useful?
- You will see that these are done in *postprocessing* and not during the evolution. Advantages / disadvantages?

Q7: How can I be sure my simulation is correct?

=

How small is small?
(Have I done a convergence test?)

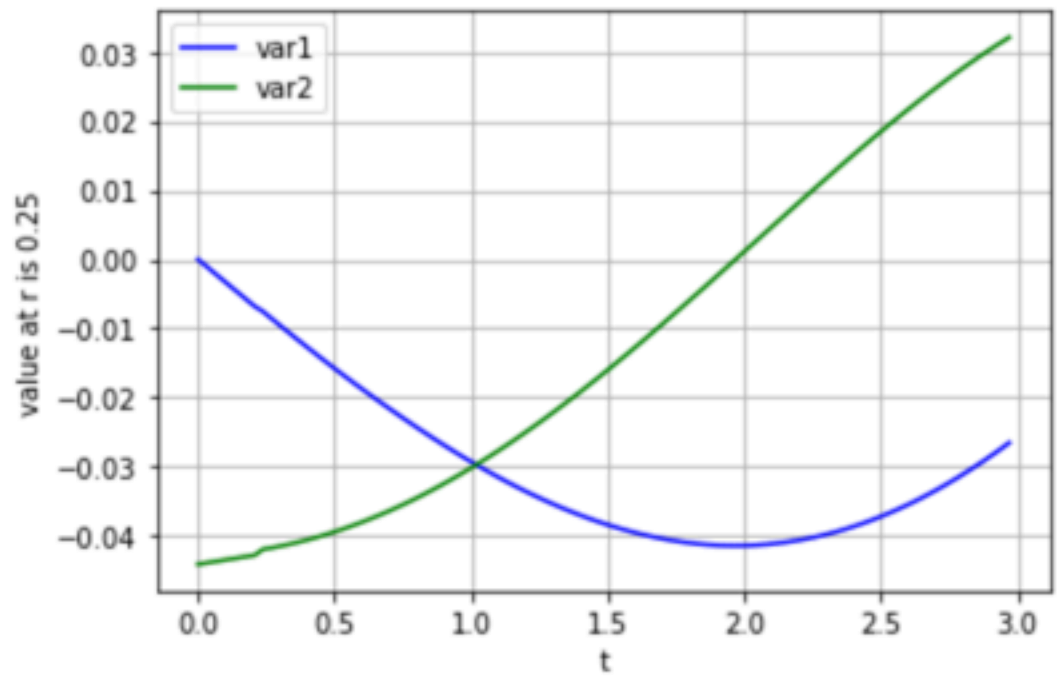


What is wrong here?

BabyGRChombo convergence

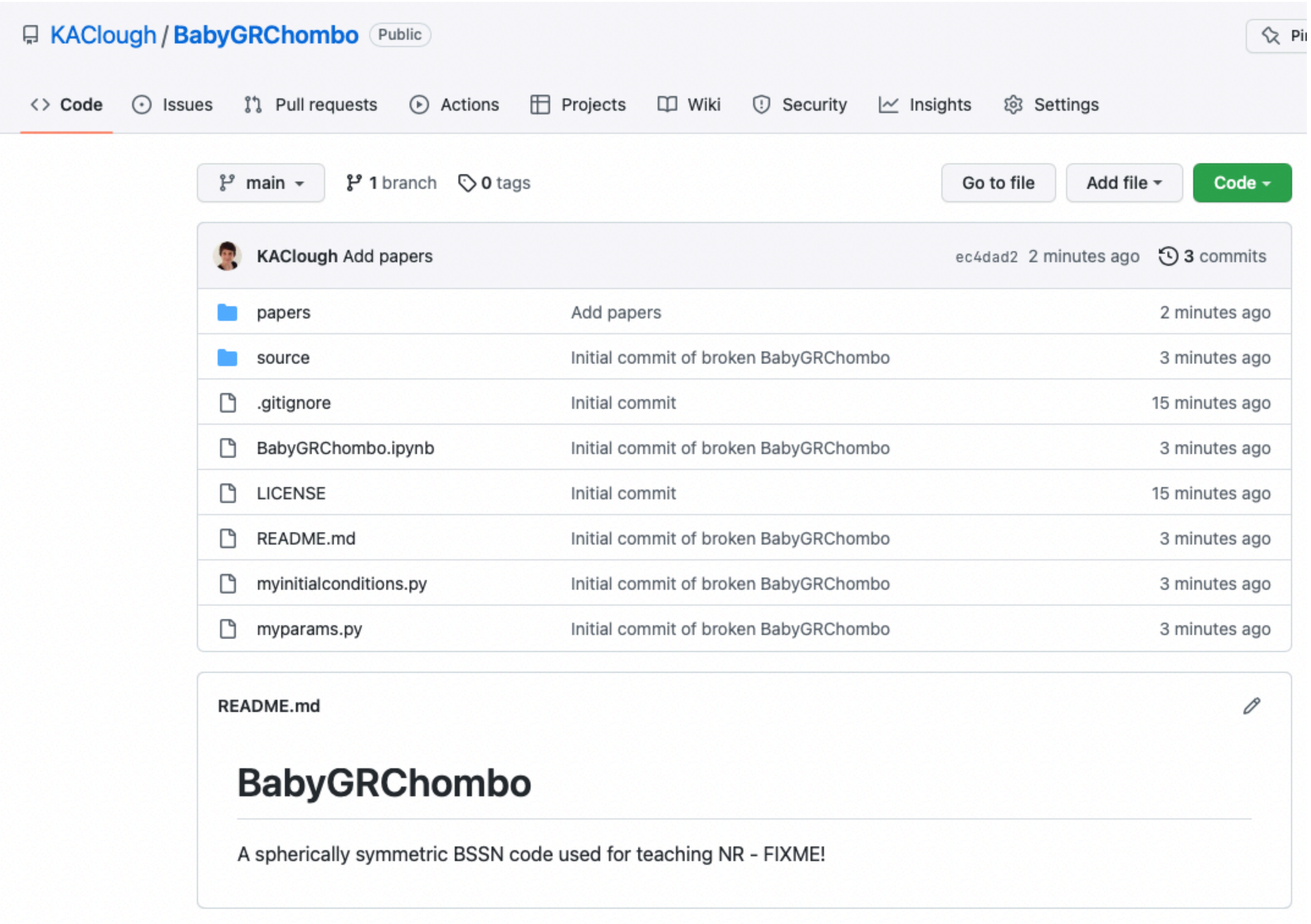
In [5]:
```
#solve for the solution
solution = odeint(get_rhs, initial_vars_values, t, args=(0, 0), atol=1e-3, rtol=1e-3) # hmin=1e-2, mxstep=100
```

In [6]:
```
# Plot a single point versus time

var1 = idx_u
var2 = idx_v

idx = num_ghosts
r_i = np.round(r[idx],2)
u_of_t = solution[0:N_t, var1 * N + idx]
plt.plot(t, u_of_t, 'b-', label='var1')
v_of_t = solution[0:N_t, var2 * N + idx]
plt.plot(t, v_of_t, 'g-', label='var2')
plt.legend(loc='best')
plt.xlabel('t')
plt.ylabel('value at r is '+str(r_i))
plt.legend(loc='best')
plt.grid()
```



- Uses odeint which is a python routine plus 4th order finite difference stencils
- What should the order of convergence be?
- Need to add a convergence test!

Day 2 of Advanced NR: Fix BabyGRChombo!



Get BabyGRChombo

1. Navigate to https://github.com/KAClough/BabyGRChombo

2. Create your own fork of the code (you will need a GitHub account). Now you can change things without breaking the main code :-)

3. In your laptop terminal git clone your fork to your laptop:

   ```
   >> git clone https://github.com/KAClough/BabyGRChombo.git
   ```

4. Navigate to the folder and open jupyter notebooks

   ```
   >> cd BabyGRChombo
   >> jupyter notebook
   ```

5. Look at the file BabyGRChombo.ipynb and run it

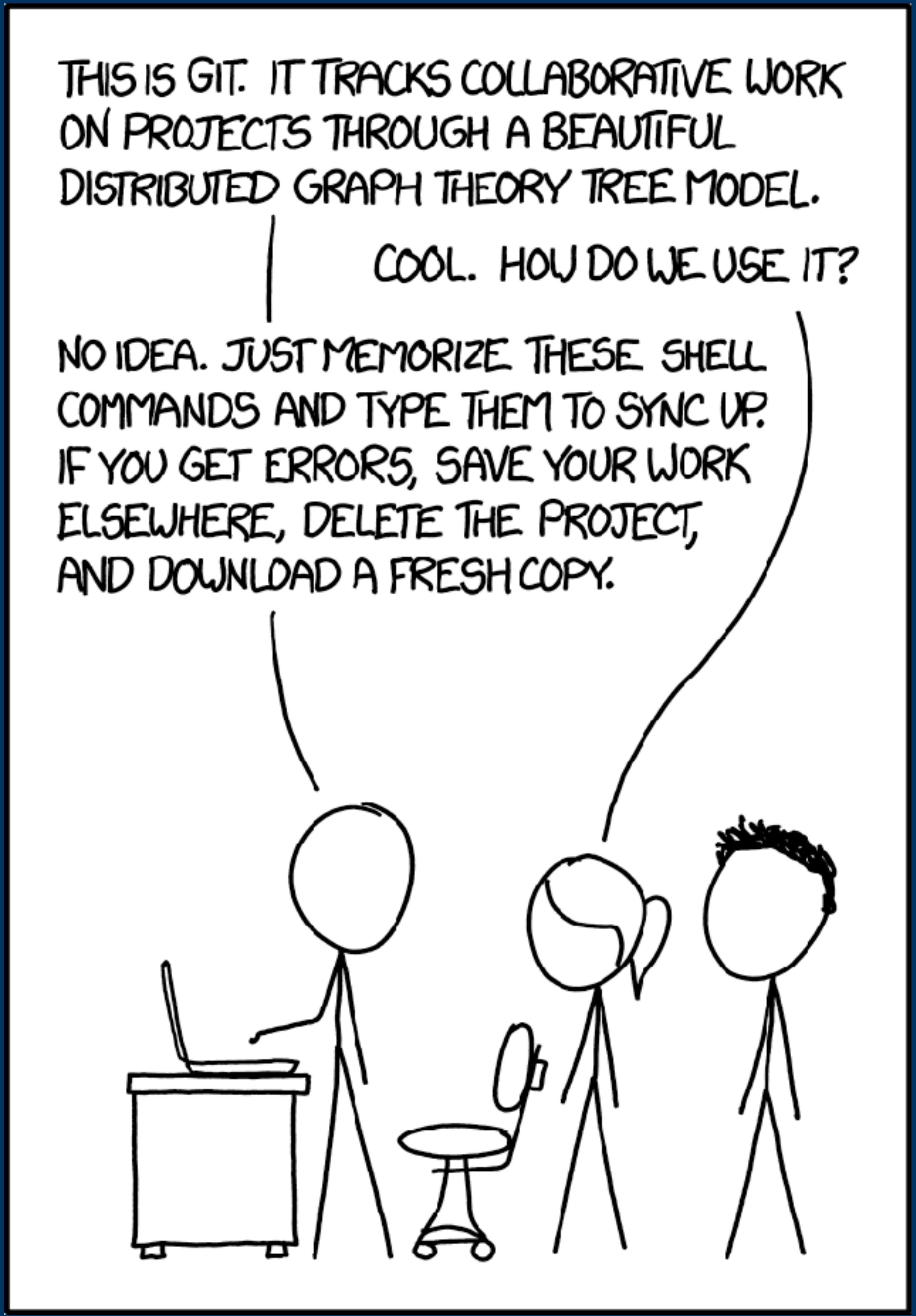
Questions?

www.grchombo.org
You can follow us on Twitter!
@GRChombo

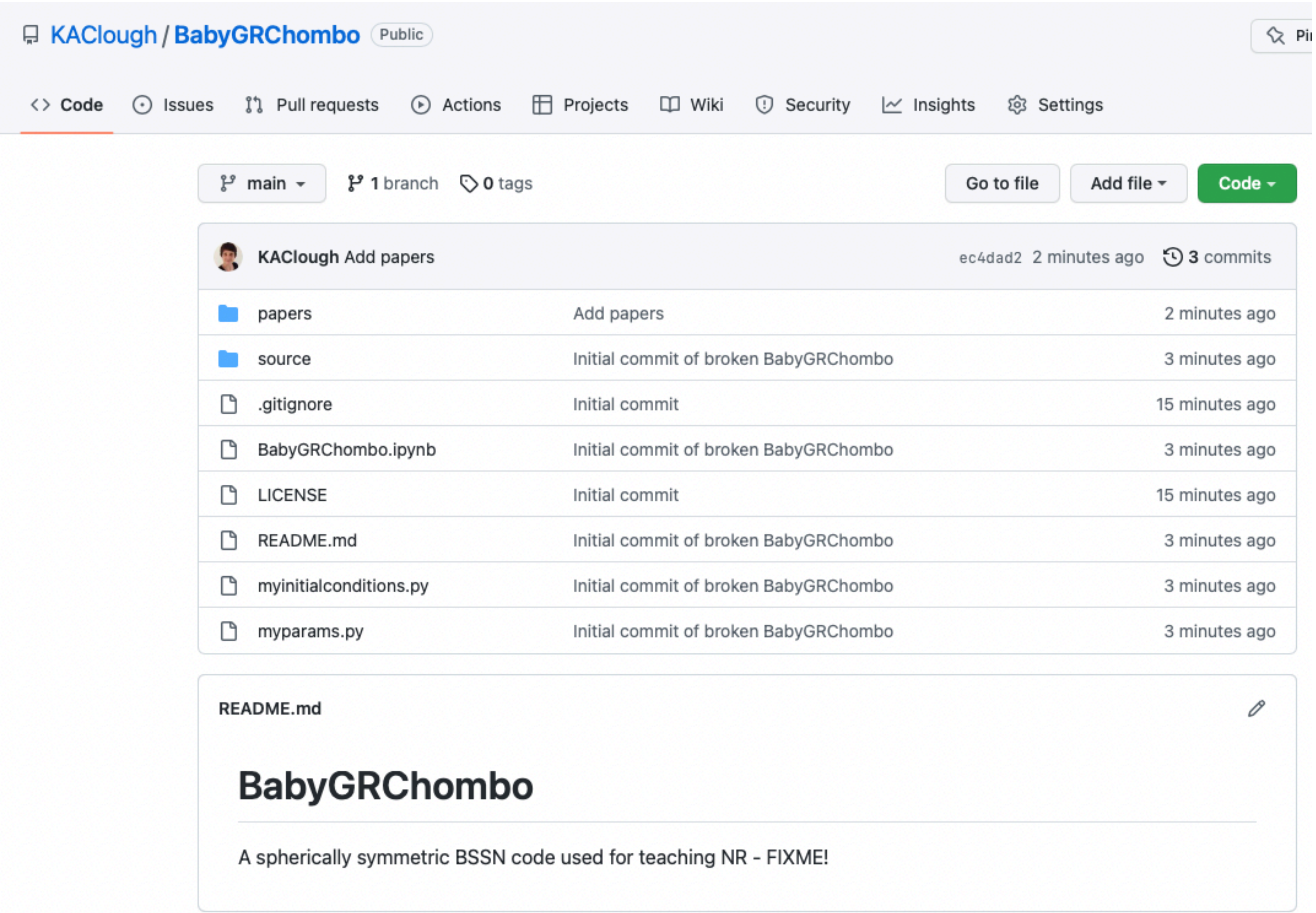
End of Day 1 lectures

Practical ~~Advanced~~ Numerical Relativity : Day 2

Katy Clough



BabyGRChombo is broken!

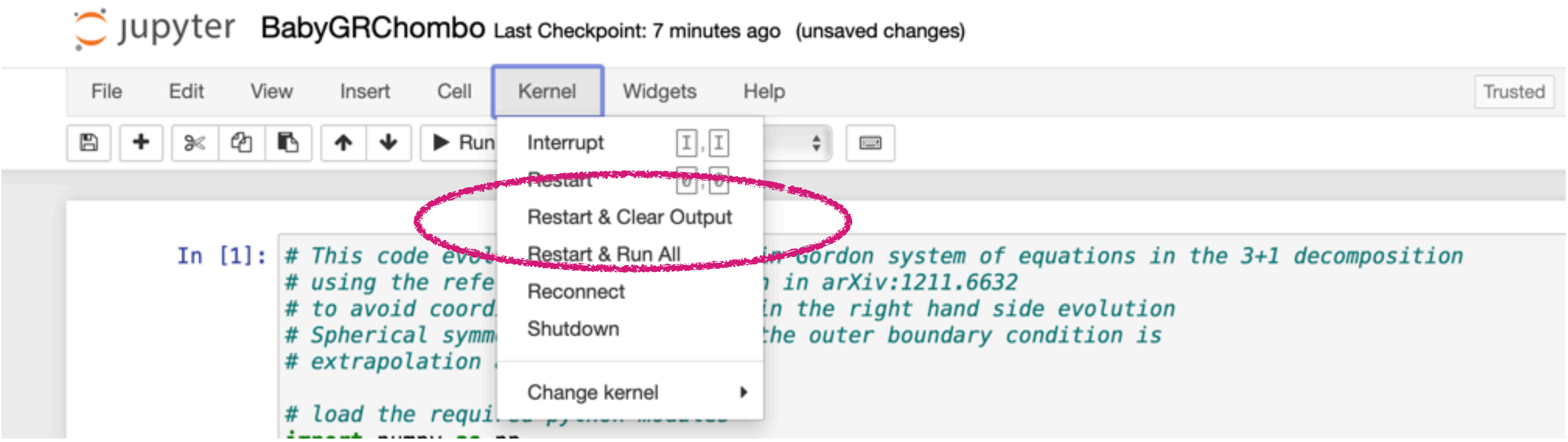


Format for the lectures

Session 2 (now):

- Key points of the reference metric framework
- 4 suggestions for possible error sources

TOP TIP: Remember after changing code to restart kernel:



Key points of the reference metric framework

Numerical Relativity in Spherical Polar Coordinates: Evolution Calculations with the BSSN Formulation

Thomas W. Baumgarte,[1,2] Pedro J. Montero,[1] Isabel Cordero-Carrión,[1] and Ewald Müller[1]
[1]*Max-Planck-Institute für Astrophysik, Karl-Schwarzschild-Str. 1, D-85748, Garching bei München, Germany*
[2]*Bowdoin College, Brunswick, ME 04011, USA*

SENR/NRPy+: Numerical relativity in singular curvilinear coordinate systems

Ian Ruchlin,[1] Zachariah B. Etienne,[1,2] and Thomas W. Baumgarte[3]
[1]*Department of Mathematics, West Virginia University, Morgantown, West Virginia 26506, USA*
[2]*Center for Gravitational Waves and Cosmology, West Virginia University, Chestnut Ridge Research Building, Morgantown, West Virginia 26505, USA*
[3]*Department of Physics and Astronomy, Bowdoin College, Brunswick, Maine 04011, USA*
(Dated: March 30, 2018)

As in the usual BSSN we decompose the spatial metric γ_{ij} such that

$$\gamma_{ij} = e^{4\phi}\bar{\gamma}_{ij}$$

The determinant $\bar{\gamma}$ of the conformal spatial metric must therefore obey

$$e^{4\phi} = (\bar{\gamma}/\gamma)^{-1/3}$$

However, instead of choosing it to be 1 we choose (and enforce at each timestep) that it obeys

$$\partial_t\bar{\gamma} = 0 \quad \bar{\gamma} = \hat{\gamma}$$

Key points of the reference metric framework

What's the hat?

This relates to the reference metric $\hat{\gamma}_{ij}$ - which we choose to be the flat space metric in spherical polar coordinates, i.e.

$$\hat{\gamma}_{ij} = diag(1,\ r^2,\ r^2 \sin^2\theta)$$

Thus

-> $\bar{\gamma} = r^4 \sin^2\theta$ is spatially varying

-> all quantities in the BSSN equations are real tensors (tensor densities of weight 0)

Key points of the reference metric framework

We now decompose the conformal metric into

$$\bar{\gamma}_{ij} = \hat{\gamma}_{ij} + \epsilon_{ij}$$

Where the deviation from the flat metric ϵ_{ij} is ***not necessarily small.***
This deviation is the quantity we want to evolve.

Key points of the reference metric framework

We can also define a related connection ***which is a tensor***

$$\Delta^i_{jk} = \bar{\Delta}^i_{jk} - \hat{\Delta}^i_{jk}$$

And its contracted form

$$\Delta^i = \bar{\gamma}^{ij}\Delta^i_{jk}$$

(Note my adoption of Etienne's naming for this as Delta not DeltaGamma $\Delta^i_{jk} = \Delta\Gamma^i_{jk}$ - it just reduces the number of "gamma"s in the code)

Key points of the reference metric framework

Final clever trick:

We want to evolve just the deviation from the flat metric ϵ_{ij} but many components will scale as 1/r near the origin of the coordinates. Therefore we rescale it (and its time derivative) and evolve the rescaled quantities h and a only:

$$\epsilon_{ij} = \begin{pmatrix} h_{rr} & r h_{r\theta} & r\sin\theta h_{r\phi} \\ r h_{r\theta} & r^2 h_{\theta\theta} & r^2 \sin\theta h_{\theta\phi} \\ r\sin\theta h_{r\phi} & r^2\sin\theta h_{\theta\phi} & r^2\sin^2\theta h_{\phi\phi} \end{pmatrix}. \quad (20)$$

We similarly rescale the extrinsic curvature $\bar{A}_{ij}$ as

$$\bar{A}_{ij} = \begin{pmatrix} a_{rr} & r a_{r\theta} & r\sin\theta a_{r\phi} \\ r a_{r\theta} & r^2 a_{\theta\theta} & r^2 \sin\theta a_{\theta\phi} \\ r\sin\theta a_{r\phi} & r^2\sin\theta a_{\theta\phi} & r^2\sin^2\theta a_{\phi\phi} \end{pmatrix}, \quad (21)$$

and the connection vector $\bar{\Lambda}^i$ as

$$\bar{\Lambda}^i = \begin{pmatrix} \lambda^r \\ \lambda^\theta / r \\ \lambda^\phi / (r\sin\theta) \end{pmatrix}. \quad (22)$$

Simplifications in BabyGRChombo

In spherical symmetry:

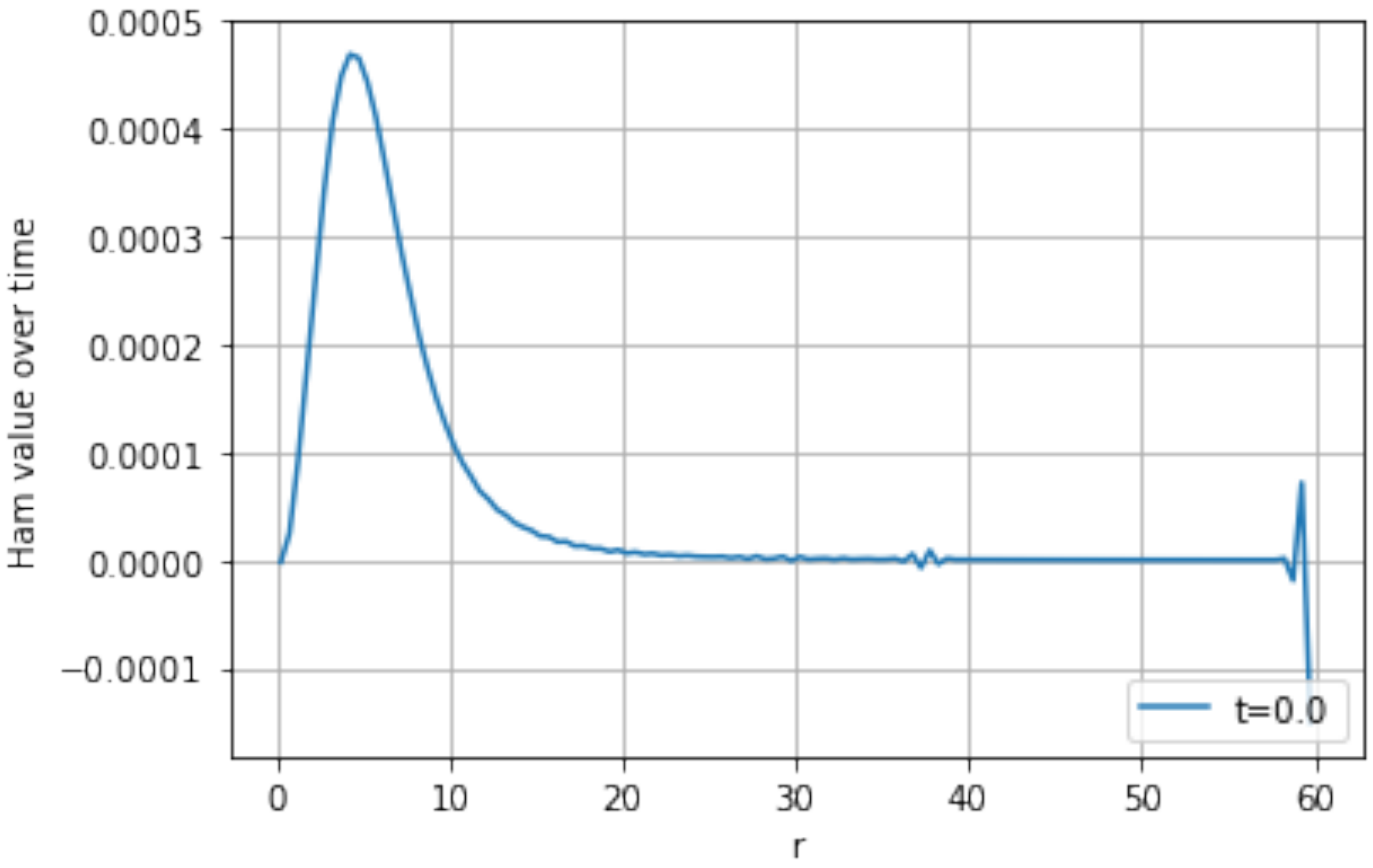
- The metric $\bar{\gamma}_{ij}$ is diagonal

$$
\begin{array}{lll}
\hat{\Gamma}^r_{\theta\theta} = -r & \hat{\Gamma}^r_{\phi\phi} = -r\sin^2\theta & \\
\hat{\Gamma}^\theta_{\phi\phi} = -\sin\theta\cos\theta & \hat{\Gamma}^\theta_{r\theta} = r^{-1} & (18) \\
\hat{\Gamma}^\phi_{r\phi} = r^{-1} & \hat{\Gamma}^\phi_{\phi\theta} = \cot\theta. &
\end{array}
$$

- Therefore so are $\bar{A}$ and a
- Only the *r* component of vectors are non zero
- Only partial derivatives with respect to *r* exist
  (note this does NOT usually mean that only covariant derivatives with respect to r exist due to non zero christoffels)
- We can choose $\sin\theta = 1 \quad \cos\theta = 0$

Possible sources of error

- Initial conditions - are we sure we have satisfied the constraints? (Have I done a convergence test? Of course not!)




```
# Get the Ham constraint eqn (13) of Baumgarte https://arxiv.org/abs/1211.6632
Ham_i[ix] = (  two_thirds * K[ix] * K[ix] - trace_A2
             + em4phi * ( bar_R
                          - 8.0 * bar_gamma_UU[i_r][i_r] * (dphidx[ix] * dphidx[ix]
                                                            + d2phidx2[ix])
                          + 8.0 * bar_gamma_UU[i_t][i_t] * flat_chris[i_r][i_t][i_t] * dphidx[ix]
                          + 8.0 * bar_gamma_UU[i_p][i_p] * flat_chris[i_r][i_p][i_p] * dphidx[ix]
                          + 8.0 * Delta_U[i_r] * dphidx[ix])
             - 2.0 * eight_pi_G * matter_rho )
```

- If wrong this means perhaps the term here multiplying em4phi is wrong as Kij is initially zero
- Could also be the initial setting of phi and h using grr

Possible sources of error

- RHS equations - lots of derived quantities calculated assuming spherical symmetry - are these right?

e.g. here I have used

$$\bar{D}_i\beta^i = \partial_i\beta^i + \frac{1}{2\bar{\gamma}}\beta^i\partial_i\bar{\gamma} \ , \qquad \bar{\gamma} = \hat{\gamma} = r^4 \qquad \Longrightarrow \bar{D}_i\beta^i = \partial_r\beta^r + \frac{2}{r}\beta^r$$

```
# This is the conformal divergence of the shift \bar D_i \beta^i
# We use the fact that the determinant of the conformal metric is
# fixed to that of the flat space metric in spherical coords
bar_div_shift = dshiftrdx[ix] + 2.0 / r_here * shiftr[ix]
# This is D^r (\bar D_i \beta^i) note the raised index of r
bar_D_div_shift = bar_gamma_UU[i_r][i_r] * (d2shiftrdx2[ix]
                                            + 2.0 / r_here * dshiftrdx[ix]
                                            - 2.0 / r_here / r_here * shiftr[ix])
```

Possible sources of error

- The rescaling should ensure factors of 1/r are always treated analytically and not multiplied within terms. Probably some of the tensoralgebra.py code does not respect that. For example this bit looks dodgy:

```python

# Computer the \bar A^ij given A_ij and \bar\gamma^ij
def get_A_UU(A_LL, bar_gamma_UU) :

    A_UU = np.zeros_like(rank_2_spatial_tensor)

    for i in range(0, SPACEDIM):
        for j in range(0, SPACEDIM):
            for k in range(0, SPACEDIM):
                for l in range(0, SPACEDIM):
                    A_UU[i][j] = bar_gamma_UU[i][k]* bar_gamma_UU[j][l] * A_LL[k][l]

    return A_UU

```

Possible sources of error

- What is odeint actually doing? Does it respect the limits on the Courant factor? Might want to set an *hmin* value to respect this.

```
In [*]: #solve for the solution
        solution = odeint(get_rhs, initial_vars_values, t, args=(0, 0), atol=1e-3, rtol=1e-3) # hmin=1e-2, mxstep=100
```

- Reminder - courant factor C relates the timestep to the spatial resolution as

$$C \equiv \Delta t \;/\; \Delta x < 0.5$$

Physically the condition is related to causality so it should really be (ignoring the shift)

$C = \tilde{\alpha} \Delta t \;/\; \Delta x < 0.5$ ($\tilde{\alpha} = \alpha \gamma^{-1/2}$ is the "desensitised lapse")

Exercises with BabyGRChombo

1. Find the bugs!
2. Change gauge so normal observers follow geodesics
3. Add too much (or too little) dissipation
4. Speed up the code using mpi4py, python tricks (maintain readability!)
5. Write momentum constraint diagnostic
6. Write energy conservation diagnostic
7. Add convergence testing
8. Add black hole initial conditions (with zero scalar field)
9. Add an initial condition solver for the metric for arbitrary field configurations
10. Form black hole via collapse of gaussian field configurations
11. Add vector field

More ambitious! ->>