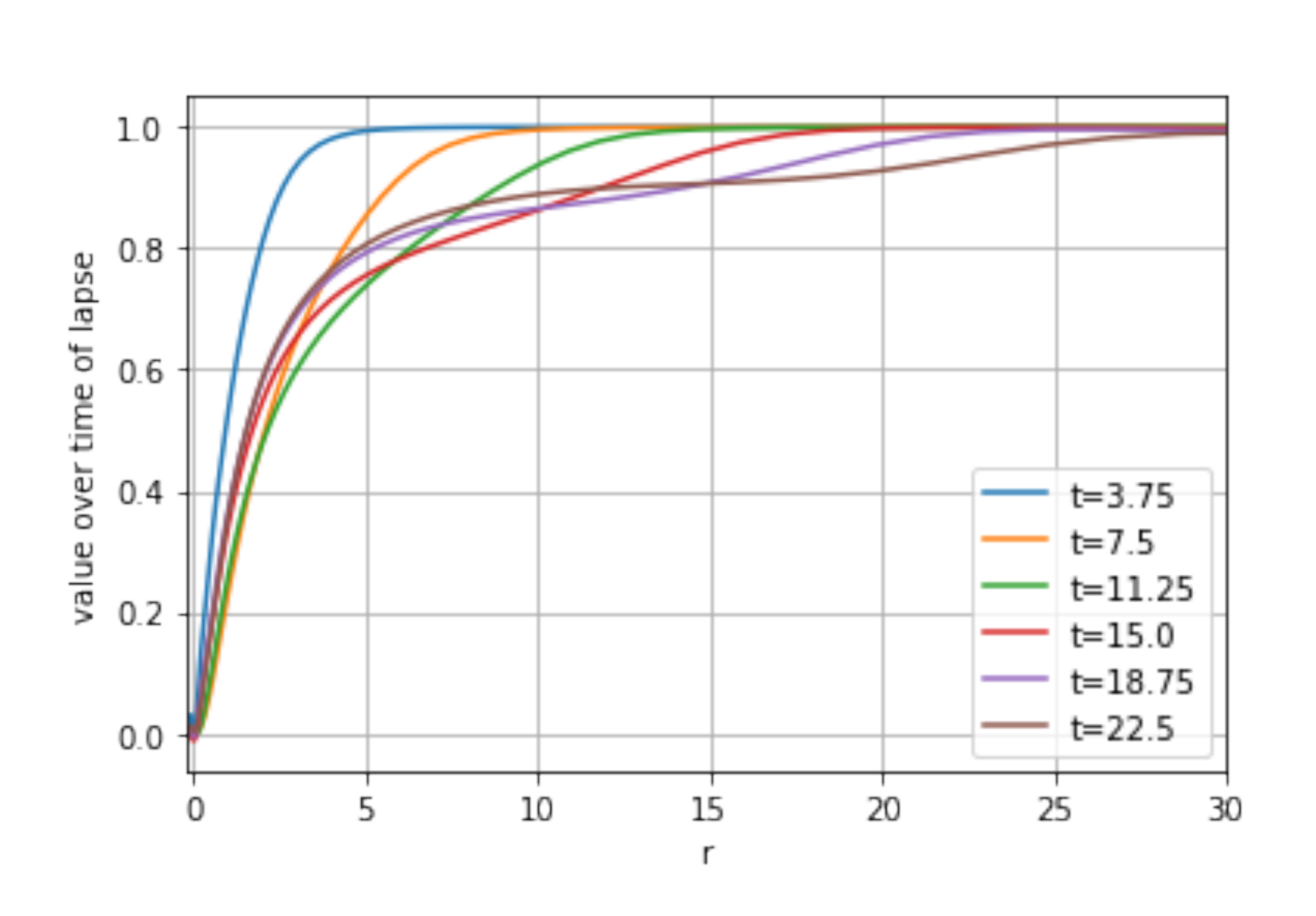
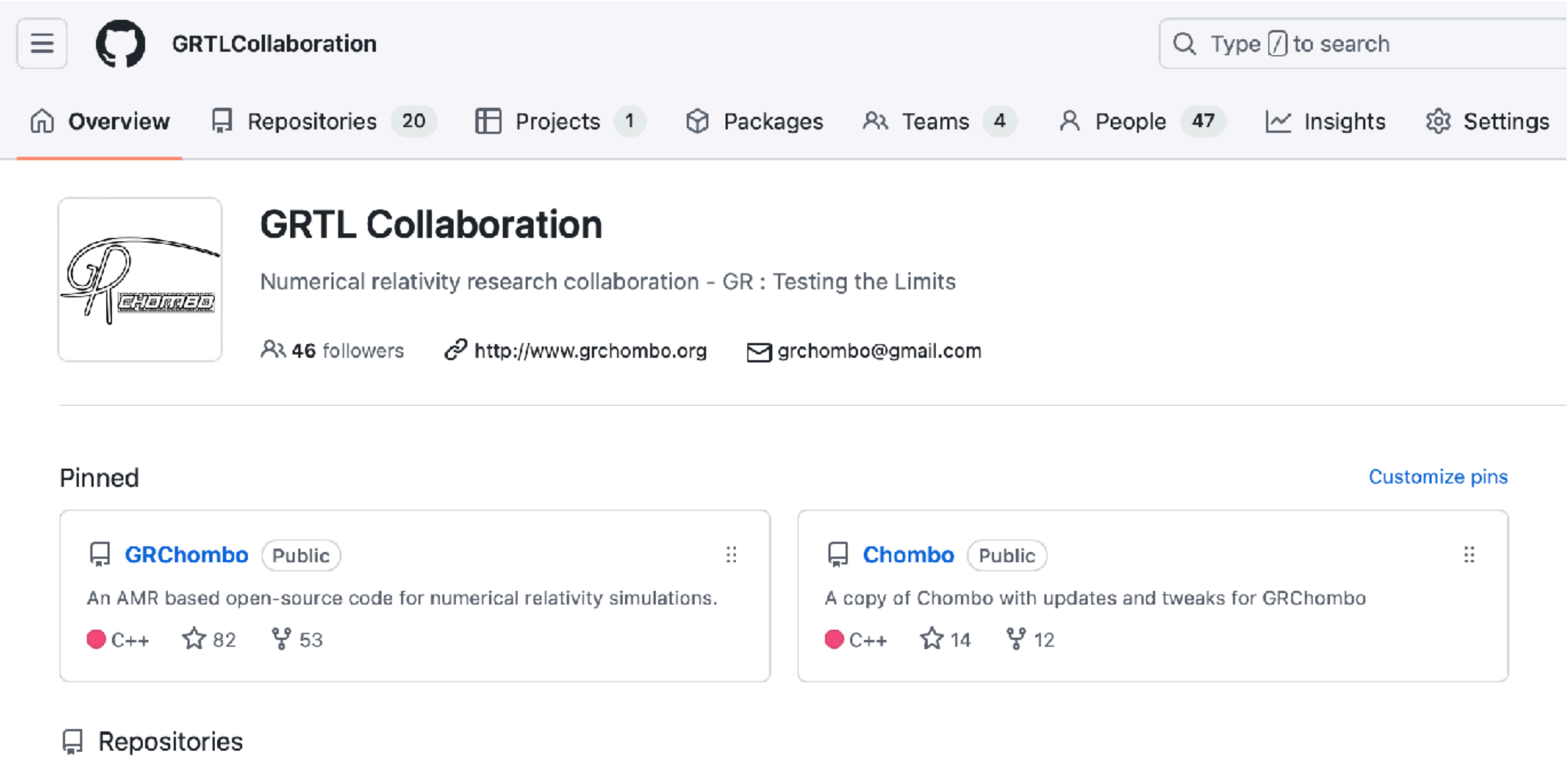
Numerical Relativity with engrenage



Dr Katy Clough
Queen Mary University of London

GR : Testing the Limits = GRTL



https://github.com/GRTLCollaboration

GRTL Collaboration

GRDzhadzha Public
A code to evolve matter on curved spacetimes with an analytic time and space dependence, e.g. black holes
C++ 10 BSD-3-Clause 6 0 1 Updated 5 hours ago

Fixed metric backgrounds

GRChombo Public
An AMR based open-source code for numerical relativity simulations.
C++ 82 BSD-3-Clause 53 33 11 Updated yesterday

Standard NR code

GRTeclyn Public
Port of GRChombo to AMReX - under development!
C++ 4 BSD-3-Clause 2 25 2 Updated last week

GPU supported code (under construction)

engrenage Public
A spherically symmetric BSSN code used for teaching NR
Jupyter Notebook 24 BSD-3-Clause 67 1 0 Updated 3 weeks ago

Spherically symmetric code

Chombo Public
A copy of Chombo with updates and tweaks for GRChombo
C++ 14 12 1 0 Updated on Jul 17

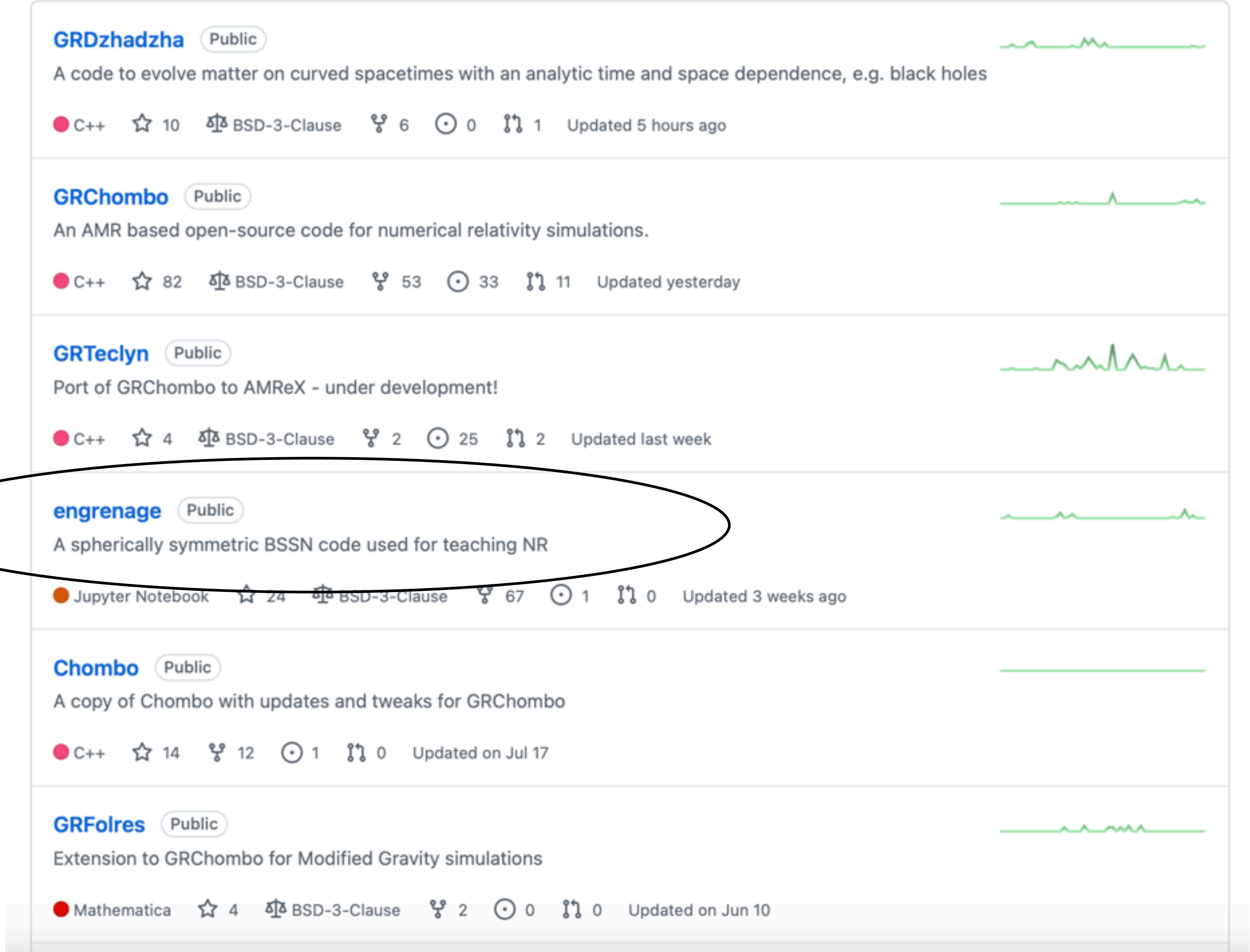
Underlying AMR PDE solver

GRFolres Public
Extension to GRChombo for Modified Gravity simulations
Mathematica 4 BSD-3-Clause 2 0 0 Updated on Jun 10

Modified gravity

https://github.com/GRTLCollaboration

GRTL Collaboration



Spherically symmetric code
- plan to make it spherically adapted

https://github.com/GRTLCollaboration/engrenage

GRTL Collaboration

Acknowledgements

This code is based on a private spherically adapted (but not spherically symmetric) code by Thomas Baumgarte, and the NRpy code of Zac Etienne, in particular the formalism used closely follows that described in the papers Numerical relativity in spherical polar coordinates: Evolution calculations with the BSSN formulation and SENR/NRPy+: Numerical relativity in singular curvilinear coordinate systems.

This code has also benefitted from input from Nils Vu @nilsvu ("You don't use python environments? I don't even know where to start..."), Leo Stein @duetosymmetry ("Why wouldn't you use the existing numpy functions for that?") and bug spotting from Cristian Joana @cjoana and Cheng-Hsin Cheng @chcheng3 when this code debuted at the ICERM Numerical Relativity Community Summer School in August 2022, and David Sola Gil @David-Sola-Gil in the London LTCC course February 2023. Thanks also to Marcus Hatton @MarcusHatton for the addition of animation to the figures.

The main developer of engrenage is Katy Clough, who is supported by a UK STFC Ernest Rutherford Fellowship ST/V003240/1.

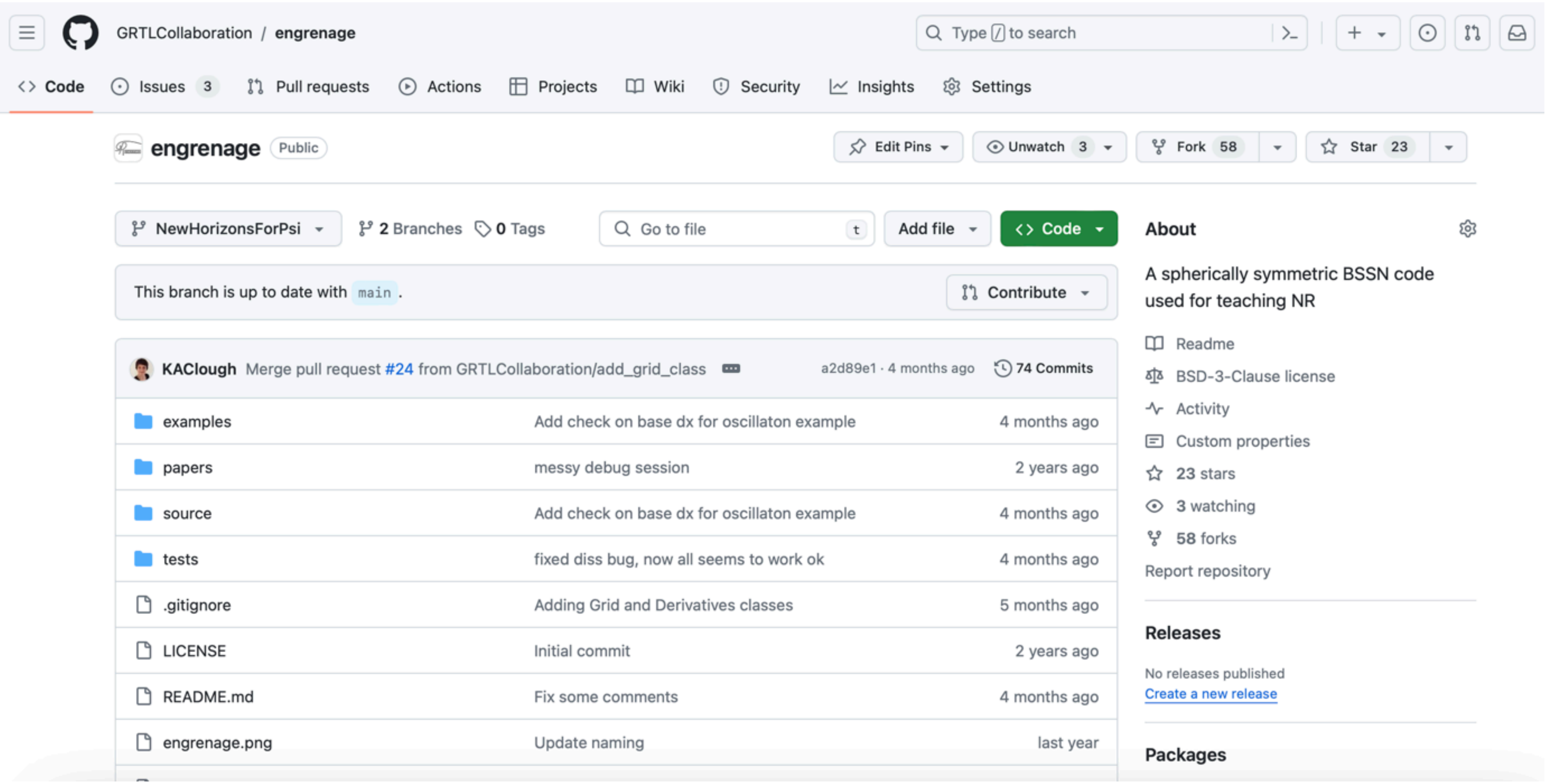
https://github.com/GRTLCollaboration/engrenage

Why did you write a spherically symmetric code in python when you have a full 3+1D code GRChombo?

Why engrenage?

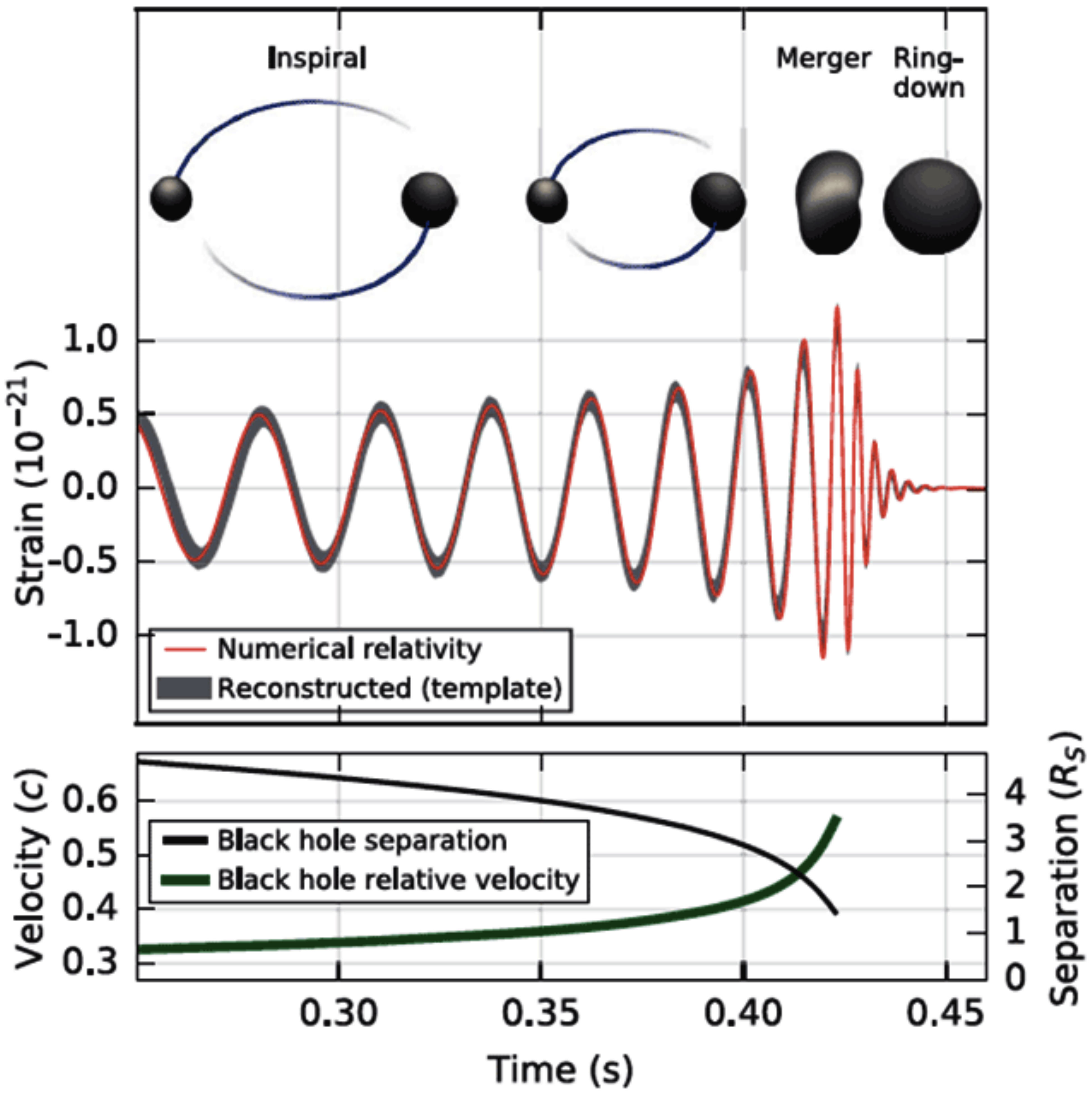
- Very useful for hands on sessions like this!
- Very useful for masters student projects (more on this from Maxence Gérard)
- Very useful for introducing new PhD students to NR concepts
- (Wasn't the original goal, but actually) very useful for actual research problems - reasonably fast so can do parameter scans, so easy to develop and modify, good testbed/comparison for 3+1D simulations
- Now being actively developed by our collaboration and friends, so more features coming… you are welcome to join us!

engrenage aka Baby GRChombo is born



https://github.com/GRTLCollaboration/engrenage

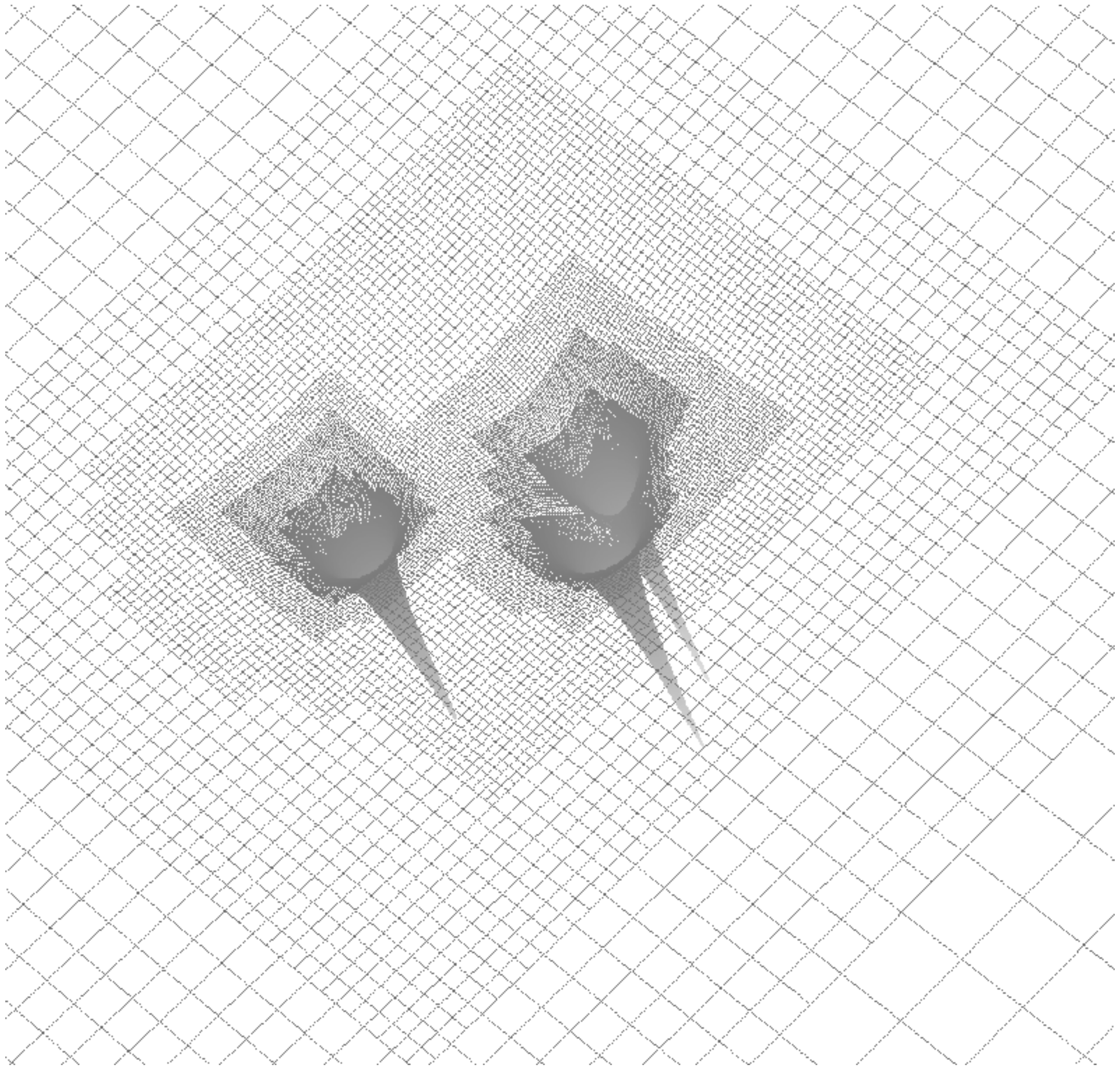
A very brief introduction to numerical relativity



GW150914

t=14 September 2015, x = LIGO, Earth

Curved spacetime



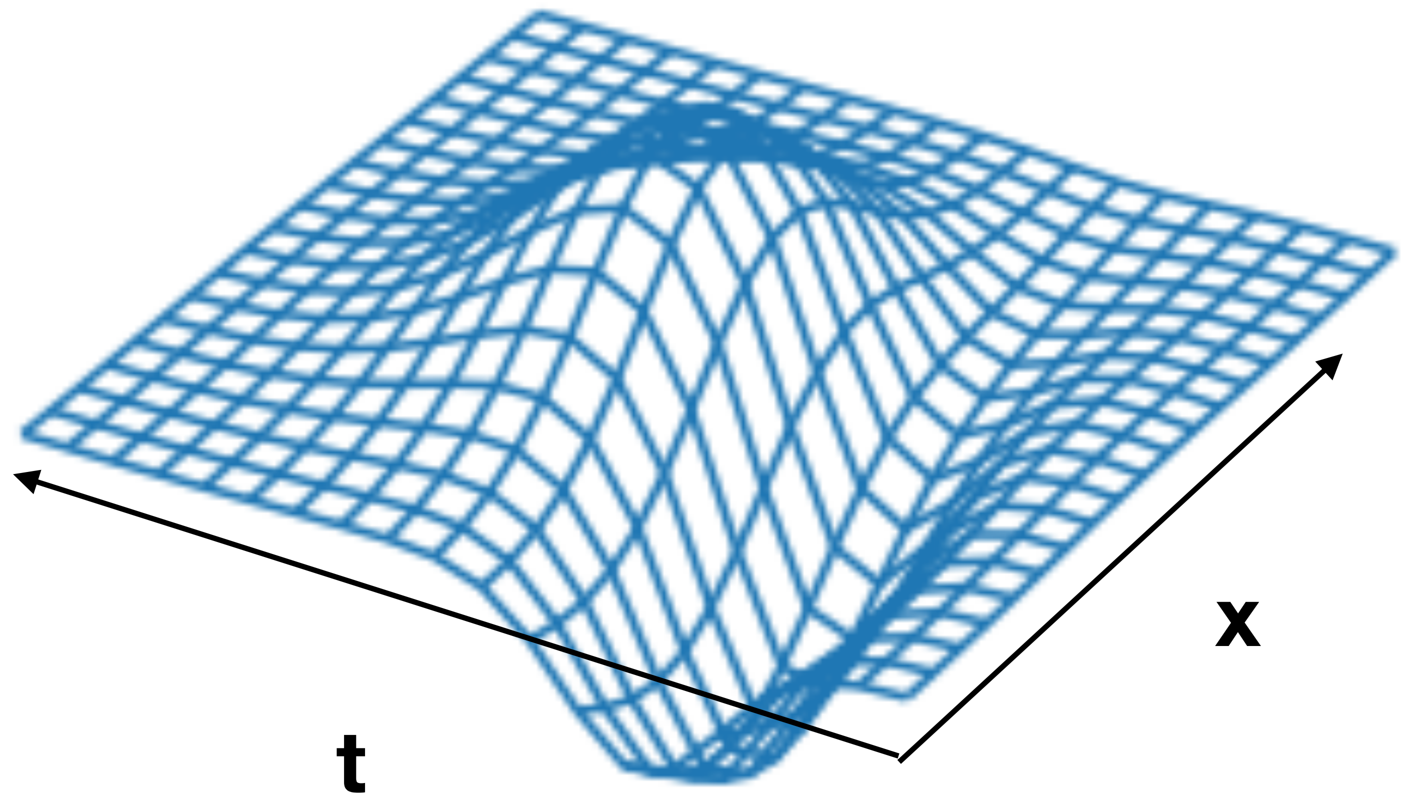
$$ds^2 = \begin{pmatrix} dt & dx & dy & dz \end{pmatrix} \underbrace{\begin{pmatrix} g_{00} & g_{01} & g_{02} & g_{03} \\ g_{10} & g_{11} & g_{12} & g_{13} \\ g_{20} & g_{21} & g_{22} & g_{23} \\ g_{30} & g_{31} & g_{32} & g_{33} \end{pmatrix}} \begin{pmatrix} dt \\ dx \\ dy \\ dz \end{pmatrix}$$

“The spacetime metric”

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

The Einstein Equation

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



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

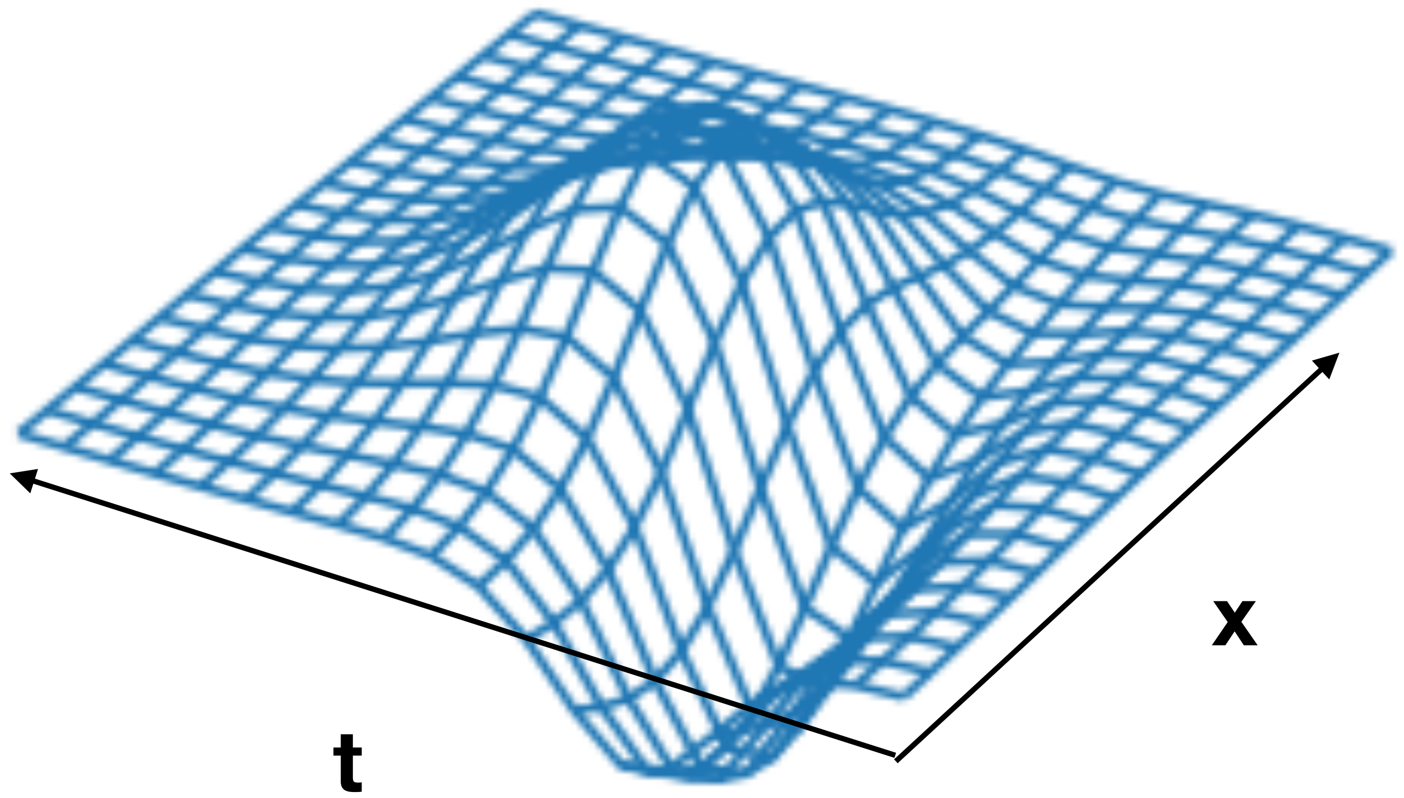
$$\frac{\partial^2 g}{\partial x^2} + \text{non linear terms} = f(\text{energy, momentum})$$

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

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

The Einstein Equation

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



$$\nabla^a T_{ab} = 0$$

Klein Gordon equation for the scalar field u

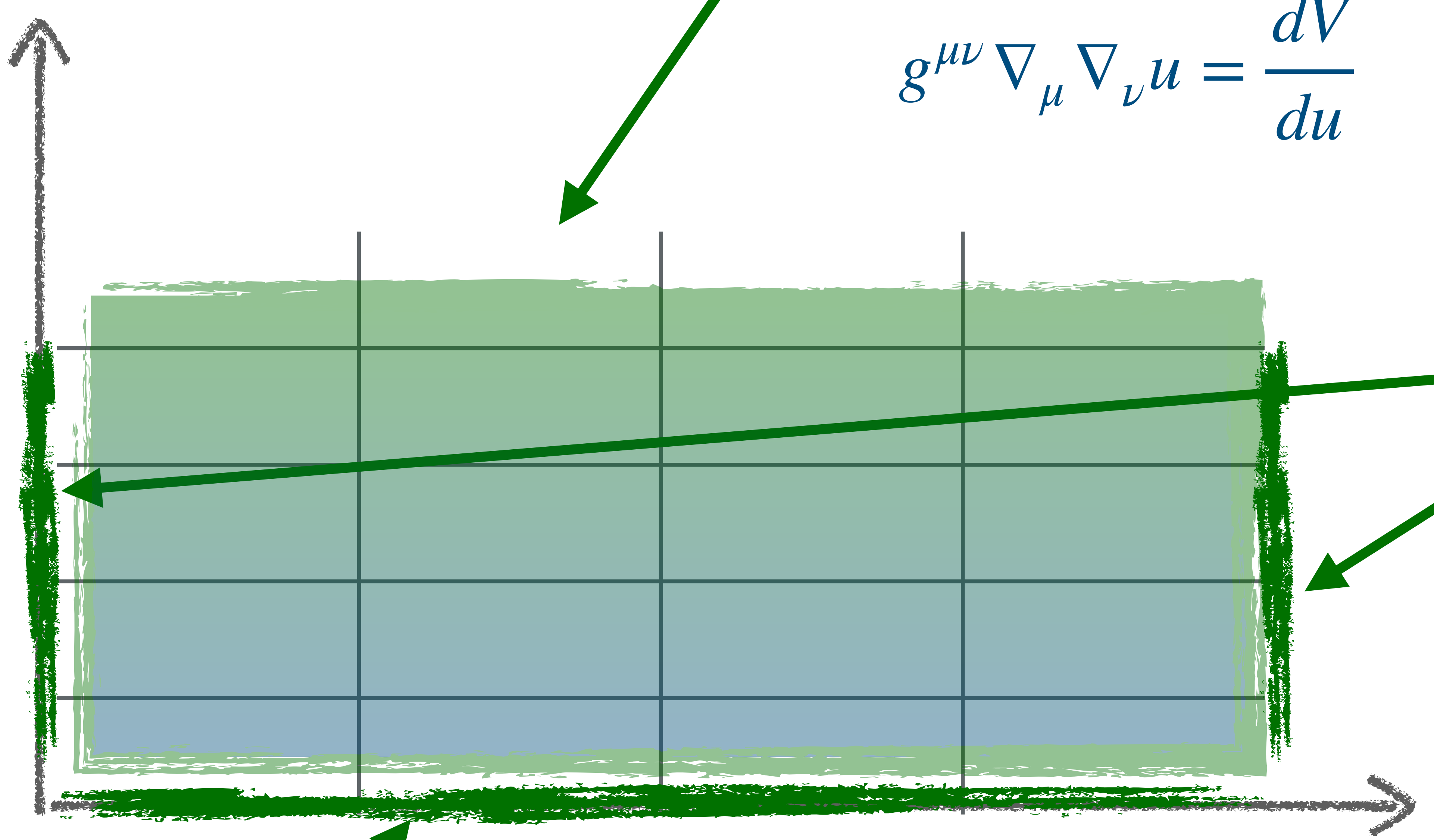
$$g^{\mu\nu} \nabla_\mu \nabla_\nu u = \frac{dV}{du}$$

Numerical relativity

Fill using Einstein equation and continuity for matter

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

$$g^{\mu\nu}\nabla_\mu \nabla_\nu u = \frac{dV}{du}$$



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

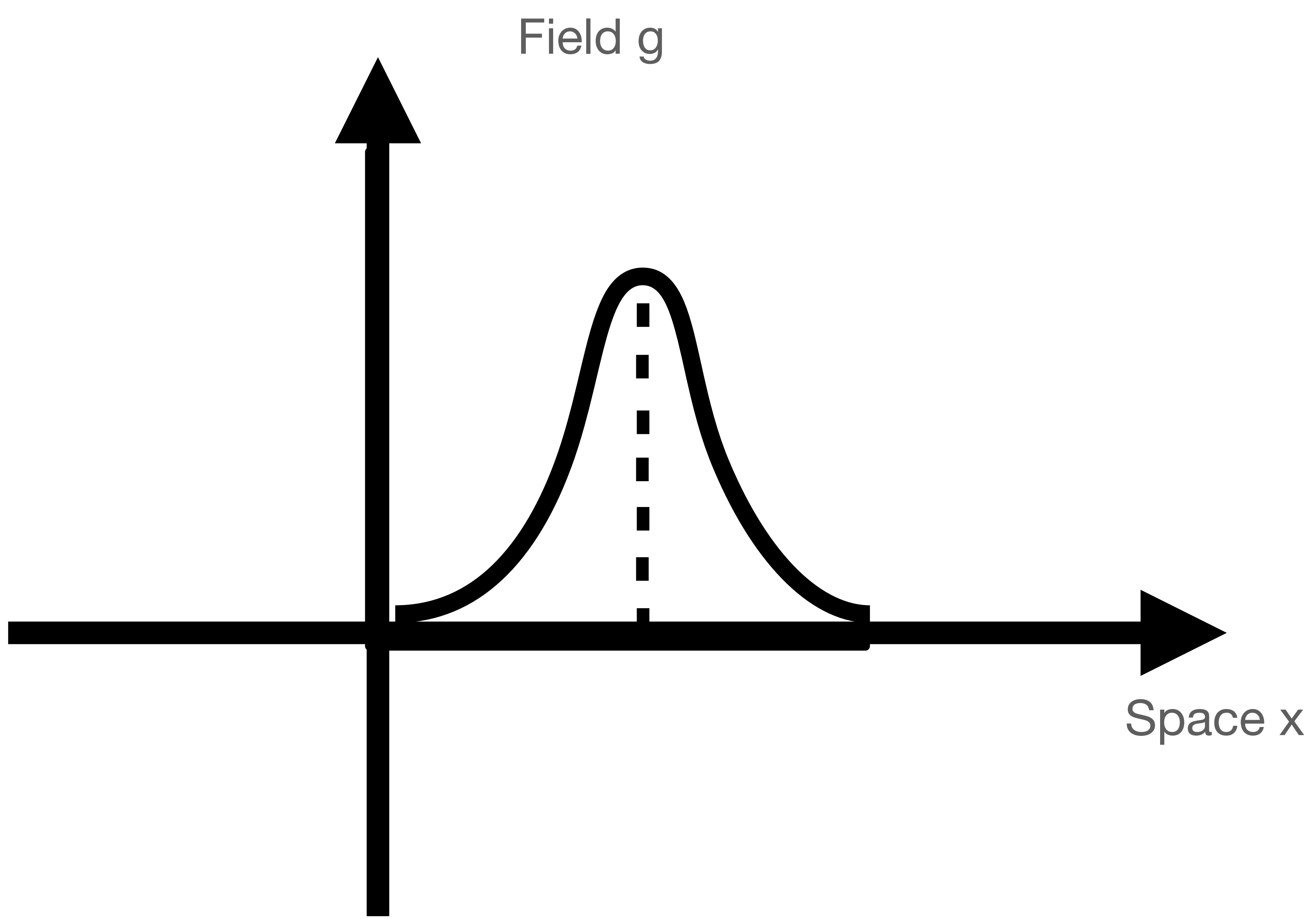
$$\frac{\partial^2 g}{\partial x^2} + \text{non linear terms} = f(\text{energy, momentum})$$

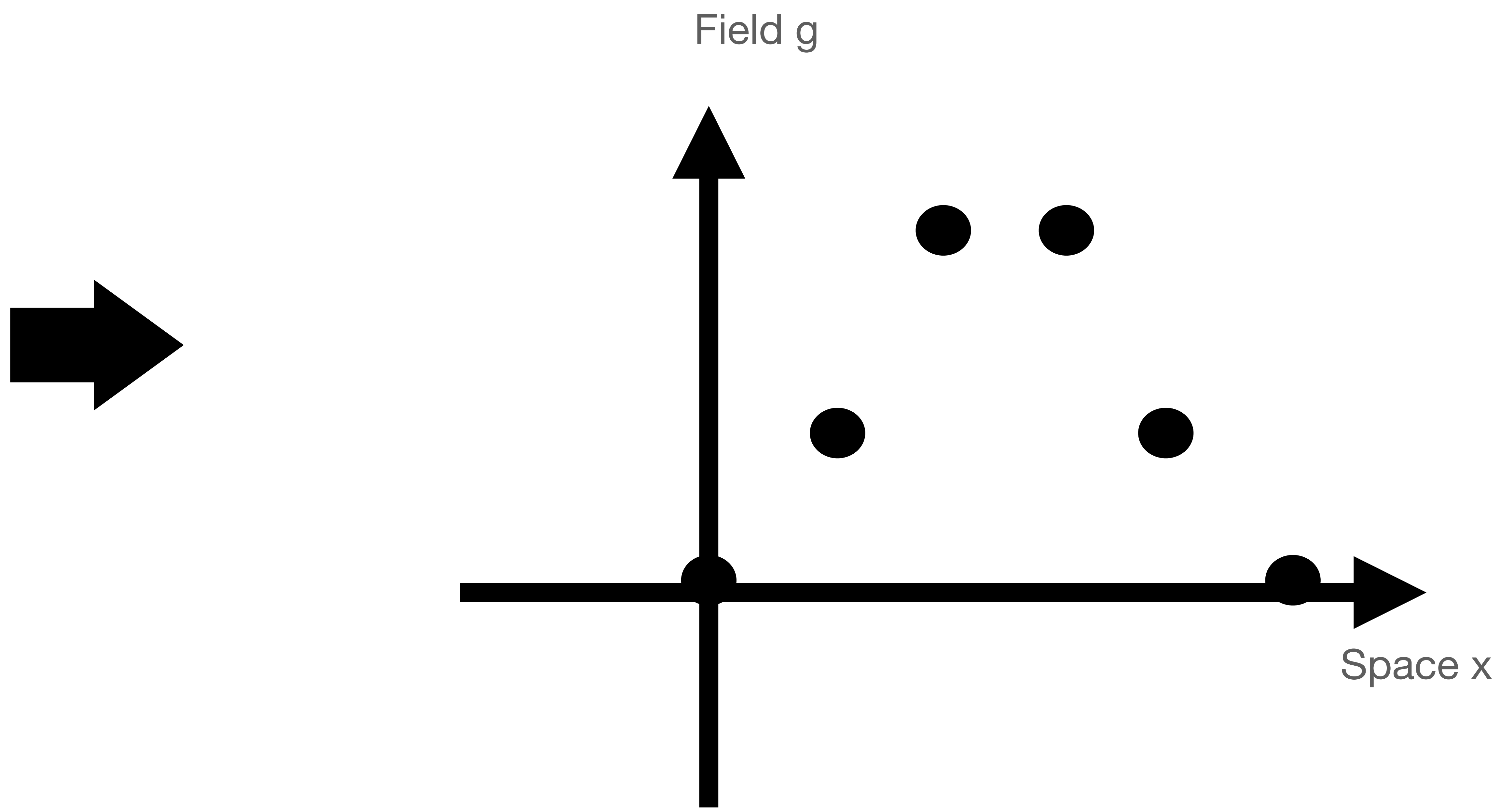
“[Nature] does not care about our mathematical difficulties; [it] integrates [numeri]cally.”

- Albert Einstein (roughly said this)

How do I represent a continuous function on a computer?

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	3	1	0





Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

$\Delta x = 0.5$

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dg/dx						

Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

↑

-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dg/dx			**1**			

Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

↑

-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dg/dx			**1**			

Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

↑

-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dg/dx		**3**	**1**			

Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

-1	0	1

dg/dx		**3**	**1**	**-2**		

Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

-1	0	1

dg/dx		**3**	**1**	**-2**	**-2**	

What about the end points?

Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

Use one sided stencil - doesn't have to be centralised

-2	2

dg/dx		**3**	**1**	**-2**	**-2**	**-2**

Spatial derivatives use finite differencing

We can see it as the convolution of ***a stencil*** with the ***current state vector***.

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

*OR use a **boundary condition** -*
some knowledge about the function
- e.g. maybe its derivative goes to zero here

dg/dx		**3**	**1**	**-2**	**-2**	**0**

Finite differencing - matrix representation

We can also represent this convolution in matrix form:

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0

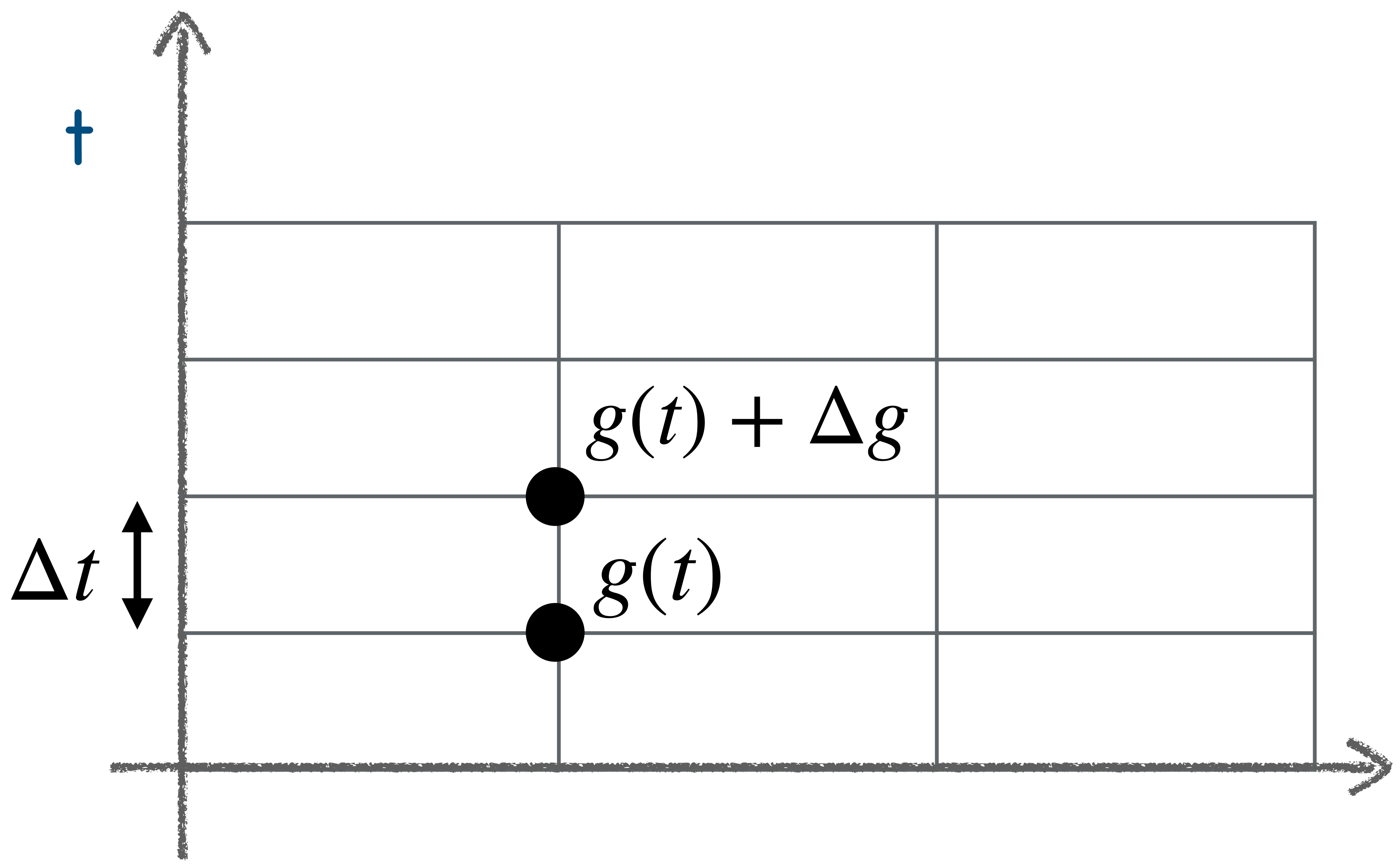
dg/dx = ***Matrix D*** ***g***

$$\begin{bmatrix} 2 \\ 3 \\ 1 \\ -2 \\ -2 \\ -2 \end{bmatrix} = \begin{bmatrix} -2 & 2 & & & & \\ -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & -1 & 0 & 1 & \\ & & & -1 & 0 & 1 \\ & & & & -2 & 2 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 1 \\ 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

All blank entries zero

First time derivatives simple to integrate

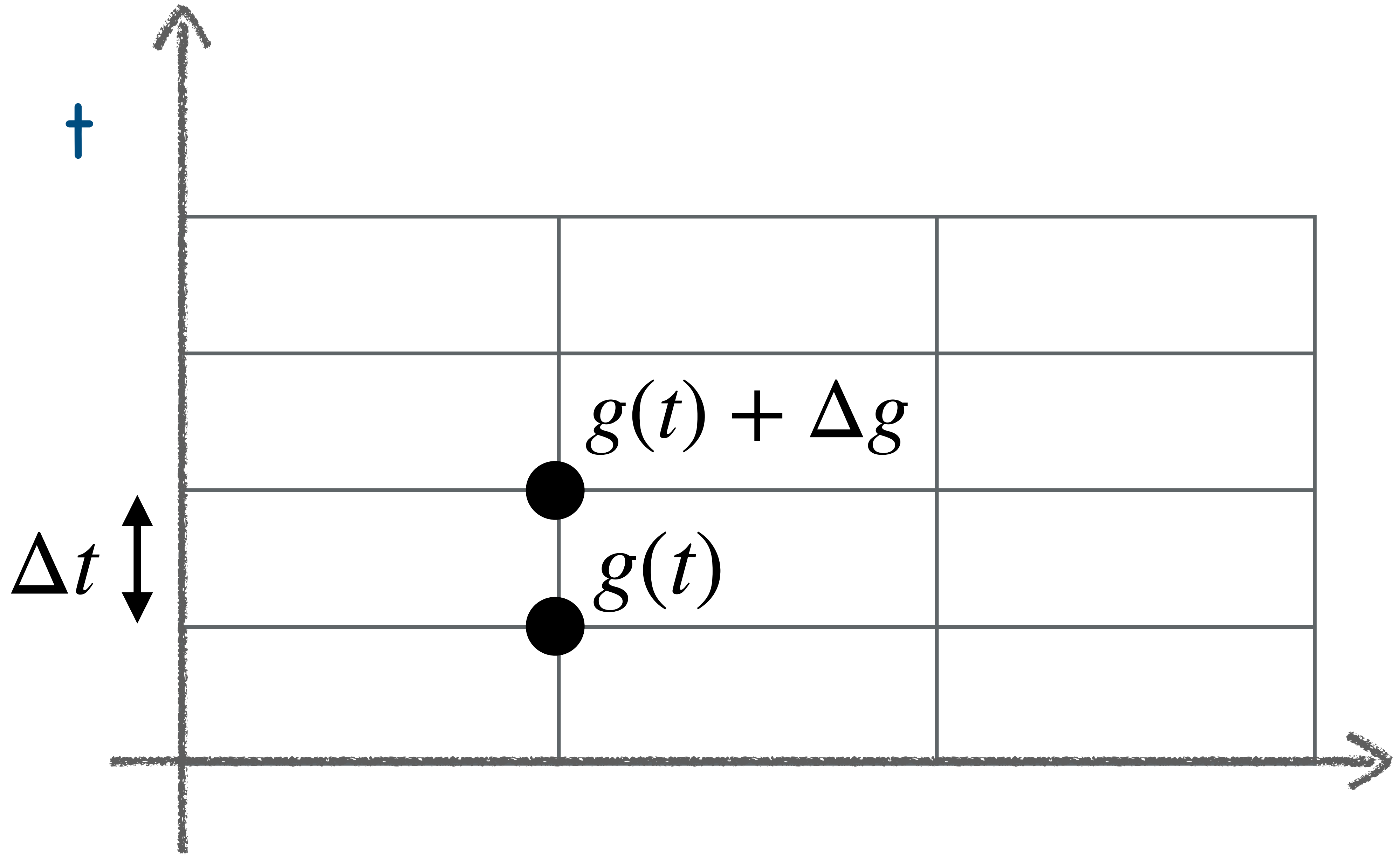
(In practise we use higher order methods, in python we use solve_ivp() which is RK4/5)



$$\Delta g = \frac{dg}{dt}\ \Delta t$$

Second time derivatives = 2 x first time derivatives

Decompose second order time derivative into two first order equations



$$\frac{\partial^2 g}{\partial t^2} - \frac{\partial^2 g}{\partial x^2} = \text{Source}$$

$$\begin{cases} \dfrac{\partial K}{\partial t} = \dfrac{\partial^2 g}{\partial x^2} + \text{Source} \\[2ex] \dfrac{\partial g}{\partial t} = K \end{cases}$$

Matrix implementation of time evolution

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0
Field K	0	2	1	1	1	0

dgdt = ***K***

$$\begin{bmatrix} 0 \\ 2 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

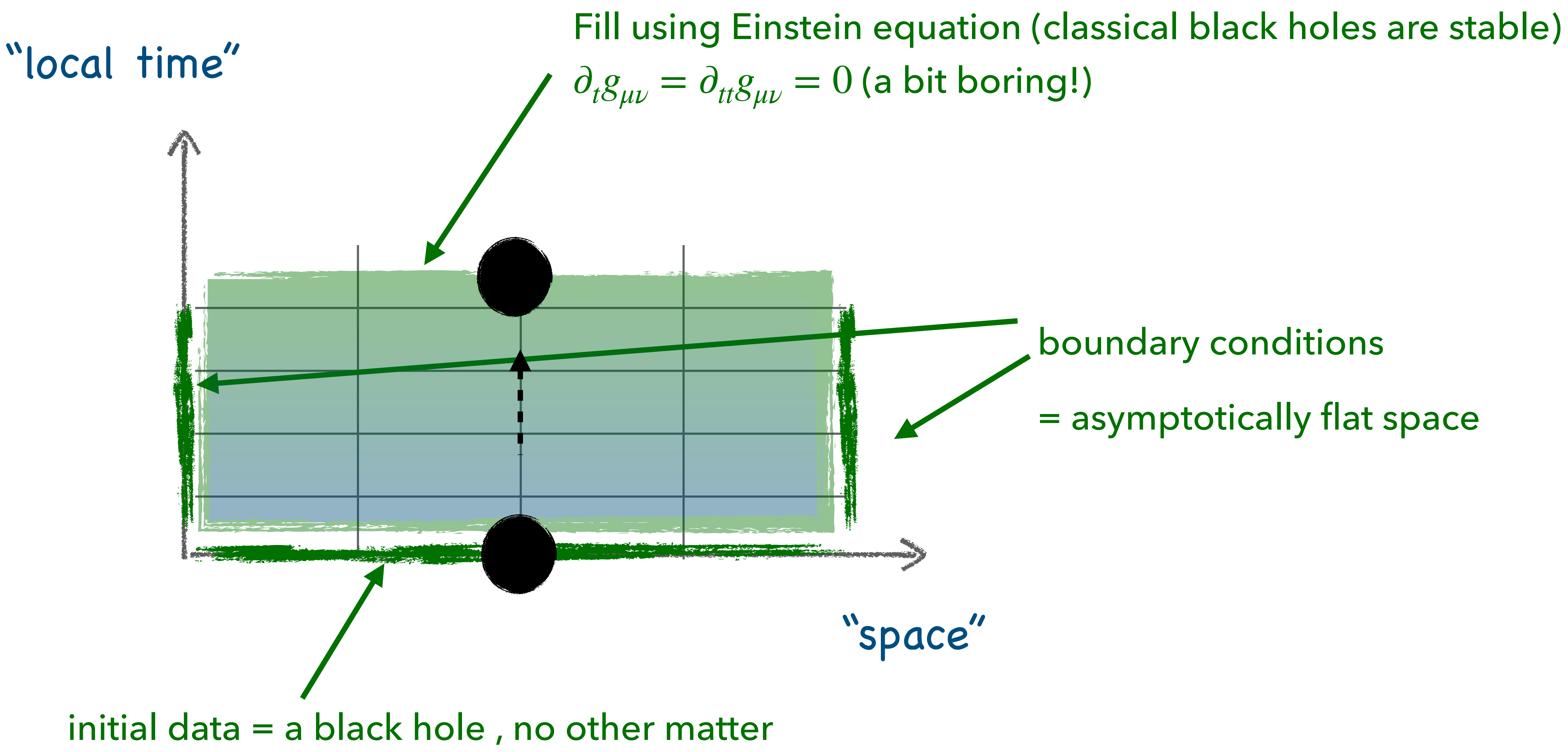
Matrix implementation of time evolution

Position x	0	0.5	1	1.5	2	2.5
Field g	0	1	3	2	1	0
Field K	0	2	1	1	1	0

dKdt **=** ***Matrix D^2*** ***g*** ***Source***

$$
\begin{bmatrix} 2 \\ 3 \\ 1 \\ -2 \\ -2 \\ -2 \end{bmatrix}
=
\begin{bmatrix}
X & X & & & & \\
X & X & X & & & \\
 & X & X & X & & \\
 & & X & X & X & \\
 & & & X & X & X \\
 & & & & X & X
\end{bmatrix}
\bullet
\begin{bmatrix} 0 \\ 1 \\ 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}
+
\begin{bmatrix} 0 \\ 3 \\ 4 \\ 5 \\ 7 \\ 0 \end{bmatrix}
$$

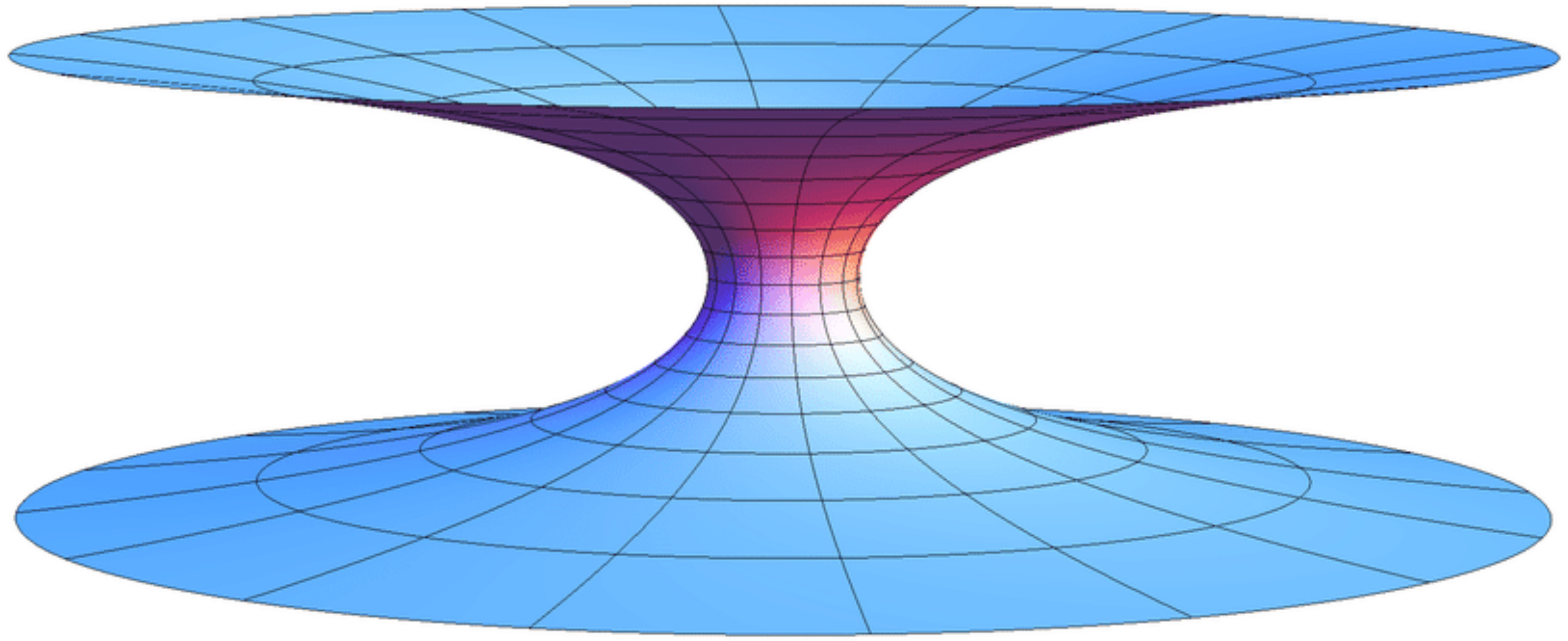
Numerical relativity evolution of a BH

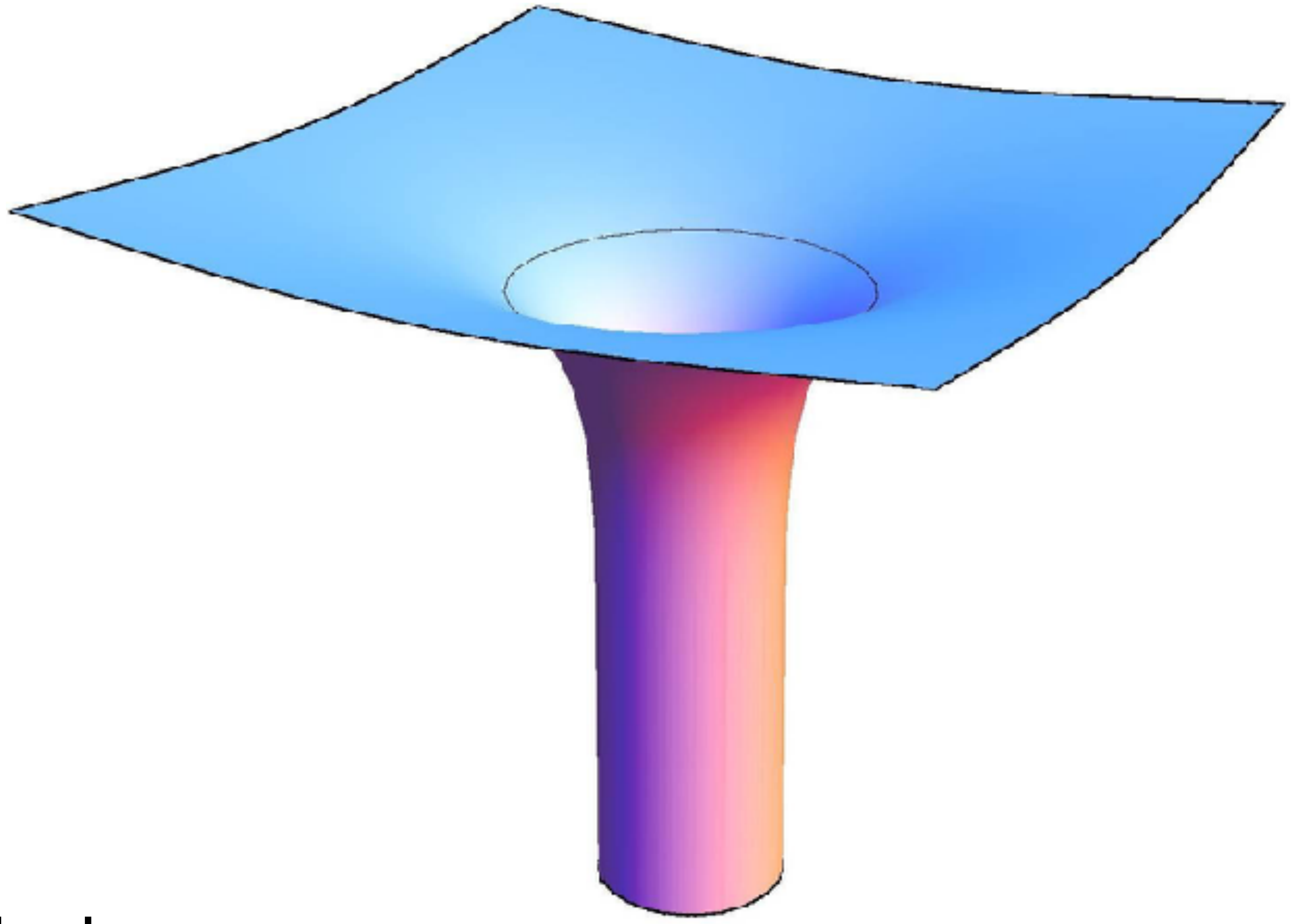


Reality: This won't work! Dynamical coordinates needed

$$ds^2 = -\left(\frac{1 - M/2R}{1 + M/2R}\right)^2 dt^2 + (1 + M/2R)^4(dR^2 + R^2 d\Omega^2)$$

$$ds^2 = -(\alpha^2 - \beta^i \beta_i)dt^2 + 2\beta_i dx^i dt + \gamma_{ij} dx^i dx^j$$

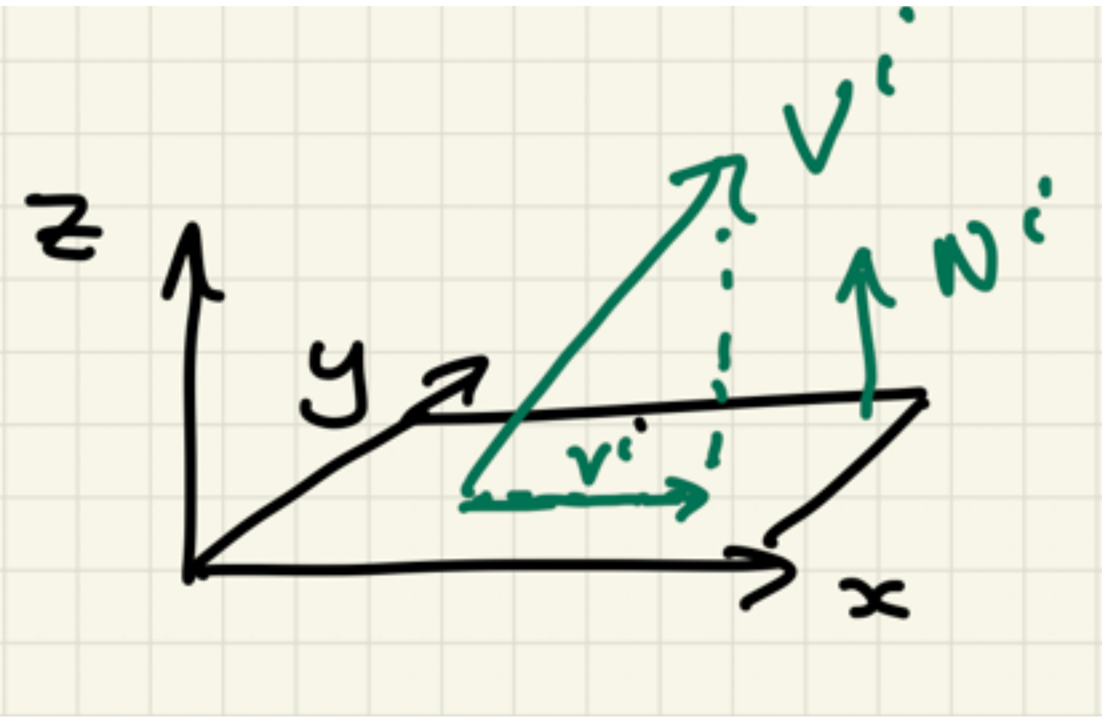




The initial "wormhole" metric evolves into a "trumpet" shape that terminates at a finite radius outside the singularity

ADM decomposition and its representation in engrenage

What is the ADM decomposition?



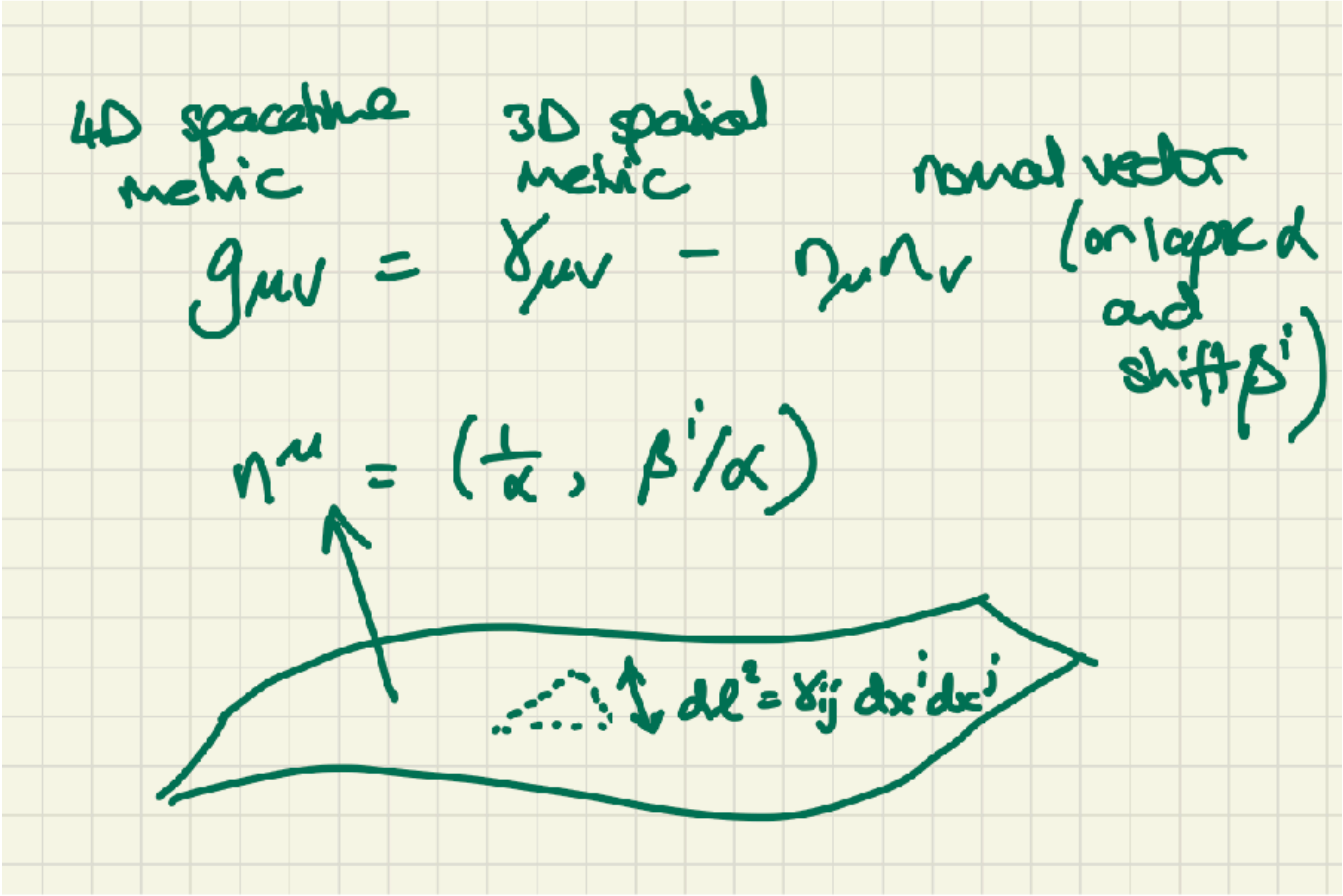
We can decompose a vector into the part that lies in a surface and a part normal to the surface

$$V^i = v^i + aN^i$$

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} + 3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

What is the ADM decomposition?

We can decompose the 4D spacetime metric into the part that lies in a 3D spatial hypersurface and a part normal to the 3D spatial hypersurface



We can also decompose the Einstein equations themselves into the part that lies in the surface and the part normal to the surface

$$n^\mu n^\nu (G_{\mu\nu} - 8\pi T_{\mu\nu}) \implies \qquad \mathcal{H} \equiv {}^{(3)}R + K^2 + K_{ij}K^{ij} - 16\pi\rho = 0$$

$$P^\mu_i n^\mu (G_{\mu\nu} - 8\pi T_{\mu\nu}) \implies \qquad \mathcal{M}_i \equiv D_j K^j{}_i - D_i K - 8\pi S_i = 0$$

$$P^\mu_i P^\nu_j (G_{\mu\nu} - 8\pi T_{\mu\nu}) \implies \qquad \partial_t K_{ij} = f(\alpha, \beta^i, \gamma^{ij}, K_{ij}, \ \partial_i(\text{variables}), \ \text{matter})$$

Where we defined $\partial_t \gamma_{ij} = -2\alpha K_{ij} + D_i\beta_j + D_j\beta_i$

What is the ADM decomposition?

If we know the metric, we can read off the quantities from the line element in the adapted coordinates

In adapted coordinates, the line element can be written as

$$ds^2 = -(\alpha^2 - \beta^i \beta_i)\, dt^2 + 2\beta_i\, dt\, dx^i + \gamma_{ij}\, dx^i\, dx^j$$

What is the ADM decomposition?

e.g. Schwarzschild

$$ds^2 = -\left(1 - \frac{2M}{r}\right) dt^2 + \left(1 - \frac{2M}{r}\right)^{-1} dr^2 + r^2 d\Omega^2$$

What is ...

$\alpha =$

$\beta^i =$

$\gamma_{ij} =$

?

These initial values must satisfy the first two constraints, and are then evolved using the two evolution equations
PLUS equations for the lapse and shift evolution

$n^\mu n^\nu (G_{\mu\nu} - 8\pi T_{\mu\nu}) \implies$ $$\mathscr{H} \equiv {}^{(3)}R + K^2 + K_{ij}K^{ij} - 16\pi\rho = 0$$

$P_i^\mu n^\mu (G_{\mu\nu} - 8\pi T_{\mu\nu}) \implies$ $$\mathscr{M}_i \equiv D_j K^j{}_i - D_i K - 8\pi S_i = 0$$

$P_i^\mu P_j^\nu (G_{\mu\nu} - 8\pi T_{\mu\nu}) \implies$ $$\partial_t K_{ij} = f(\alpha, \beta^i, \gamma^{ij}, K_{ij}, \partial_i(\text{variables}), \text{matter})$$

$$\partial_t \gamma_{ij} = -2\alpha K_{ij} + D_i\beta_j + D_j\beta_i \quad \text{(definition)}$$

$$\partial_t \alpha \sim -2\alpha K \quad \text{(choice)}$$

$$\partial_t \beta^i \sim \bar{\Gamma}^i \quad \text{(choice)}$$

What relates to the spatial metric γ_{ij} in engrenage?

```
#uservariables.py

# hard code number of ghosts to 3 here
num_ghosts = 3

# This file provides the list of (rescaled) variables to be evolved and
# assigns each one an index and its parity
# For description of the data structure see https://github.com/GRChombo/engrenage/wiki/Useful-code-background

idx_u       = 0    # scalar field
idx_v       = 1    # scalar field conjugate momentum (roughly the time derivative of u)
idx_phi     = 2    # conformal factor of metric, \gamma_ij = e^{4 \phi} \bar \gamma_ij
idx_hrr     = 3    # rescaled \epsilon_rr -> h_rr - deviation of rr component of the metric from flat
idx_htt     = 4    # rescaled \epsilon_tt -> h_tt - deviation of tt component of the metric from flat
idx_hpp     = 5    # rescaled \epsilon_pp -> h_pp - deviation of pp component of the metric from flat
idx_K       = 6    # mean curvature K
idx_arr     = 7    # rescaled \tilde A_rr -> a_rr - (roughly) time derivative of hrr
idx_att     = 8    # rescaled \tilde A_tt -> a_tt - (roughly) time derivative of htt
idx_app     = 9    # rescaled \tilde A_pp -> a_pp - (roughly) time derivative of hpp
idx_lambdar = 10   # rescaled \bar\Lambda -> lambda^r
idx_shiftr  = 11   # rescaled \beta^r -> radial shift - gauge variable for relabelling spatial points
idx_br      = 12   # rescaled B^r -> b^r - time derivative of shift
idx_lapse   = 13   # lapse - gauge variable for time slicing
```

What are phi, h_rr etc?

We perform a conformal decomposition of the spatial metric into a conformal part and an overall conformal factor

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

We also split things into a (flat) background metric and the (not small) deviation from it

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

We also scale the quantities to effectively make the basic vectors orthonormal

$$h_{rr} = \epsilon_{rr} \qquad h_{rt} = \frac{1}{r^2}\epsilon_{rt}$$

What relates to the K_{ij} in engrenage?

```python
#uservariables.py

# hard code number of ghosts to 3 here
num_ghosts = 3

# This file provides the list of (rescaled) variables to be evolved and
# assigns each one an index and its parity
# For description of the data structure see https://github.com/GRChombo/engrenage/wiki/Useful-code-background

idx_u       = 0    # scalar field
idx_v       = 1    # scalar field conjugate momentum (roughly the time derivative of u)
idx_phi     = 2    # conformal factor of metric, \gamma_ij = e^{4 \phi} \bar \gamma_ij
idx_hrr     = 3    # rescaled \epsilon_rr -> h_rr - deviation of rr component of the metric from flat
idx_htt     = 4    # rescaled \epsilon_tt -> h_tt - deviation of tt component of the metric from flat
idx_hpp     = 5    # rescaled \epsilon_pp -> h_pp - deviation of pp component of the metric from flat
idx_K       = 6    # mean curvature K
idx_arr     = 7    # rescaled \tilde A_rr -> a_rr - (roughly) time derivative of hrr
idx_att     = 8    # rescaled \tilde A_tt -> a_tt - (roughly) time derivative of htt
idx_app     = 9    # rescaled \tilde A_pp -> a_pp - (roughly) time derivative of hpp
idx_lambdar = 10   # rescaled \bar\Lambda -> lambda^r
idx_shiftr  = 11   # rescaled \beta^r -> radial shift - gauge variable for relabelling spatial points
idx_br      = 12   # rescaled B^r -> b^r - time derivative of shift
idx_lapse   = 13   # lapse - gauge variable for time slicing
```

What are K, a_rr etc?

We perform a conformal decomposition of the spatial metric into a trace and a trace free part

$$K_{ij} = e^{4\phi}(\bar{A}_{ij} - \frac{1}{3}\bar{\gamma}_{ij}K)$$

We also scale the quantities to effectively make the basic vectors orthonormal

$$a_{rr} = \bar{A}_{rr} \quad a_{rt} = \frac{1}{r^2}\bar{A}_{rt}$$

But roughly speaking

$$a_{rr} \sim \partial_t h_{rr}$$

What relates to the lapse in engrenage?

```
#uservariables.py

# hard code number of ghosts to 3 here
num_ghosts = 3

# This file provides the list of (rescaled) variables to be evolved and
# assigns each one an index and its parity
# For description of the data structure see https://github.com/GRChombo/engrenage/wiki/Useful-code-background

idx_u       = 0    # scalar field
idx_v       = 1    # scalar field conjugate momentum (roughly the time derivative of u)
idx_phi     = 2    # conformal factor of metric, \gamma_ij = e^{4 \phi} \bar \gamma_ij
idx_hrr     = 3    # rescaled \epsilon_rr -> h_rr - deviation of rr component of the metric from flat
idx_htt     = 4    # rescaled \epsilon_tt -> h_tt - deviation of tt component of the metric from flat
idx_hpp     = 5    # rescaled \epsilon_pp -> h_pp - deviation of pp component of the metric from flat
idx_K       = 6    # mean curvature K
idx_arr     = 7    # rescaled \tilde A_rr -> a_rr - (roughly) time derivative of hrr
idx_att     = 8    # rescaled \tilde A_tt -> a_tt - (roughly) time derivative of htt
idx_app     = 9    # rescaled \tilde A_pp -> a_pp - (roughly) time derivative of hpp
idx_lambdar = 10   # rescaled \bar\Lambda -> lambda^r
idx_shiftr  = 11   # rescaled \beta^r -> radial shift - gauge variable for relabelling spatial points
idx_br      = 12   # rescaled B^r -> b^r - time derivative of shift
idx_lapse   = 13   # lapse - gauge variable for time slicing
```

What relates to the shift in engrenage?

```python
#uservariables.py

# hard code number of ghosts to 3 here
num_ghosts = 3

# This file provides the list of (rescaled) variables to be evolved and
# assigns each one an index and its parity
# For description of the data structure see https://github.com/GRChombo/engrenage/wiki/Useful-code-background

idx_u       = 0    # scalar field
idx_v       = 1    # scalar field conjugate momentum (roughly the time derivative of u)
idx_phi     = 2    # conformal factor of metric, \gamma_ij = e^{4 \phi} \bar \gamma_ij
idx_hrr     = 3    # rescaled \epsilon_rr -> h_rr - deviation of rr component of the metric from flat
idx_htt     = 4    # rescaled \epsilon_tt -> h_tt - deviation of tt component of the metric from flat
idx_hpp     = 5    # rescaled \epsilon_pp -> h_pp - deviation of pp component of the metric from flat
idx_K       = 6    # mean curvature K
idx_arr     = 7    # rescaled \tilde A_rr -> a_rr - (roughly) time derivative of hrr
idx_att     = 8    # rescaled \tilde A_tt -> a_tt - (roughly) time derivative of htt
idx_app     = 9    # rescaled \tilde A_pp -> a_pp - (roughly) time derivative of hpp
idx_lambdar = 10   # rescaled \bar\Lambda -> lambda^r
idx_shiftr  = 11   # rescaled \beta^r -> radial shift - gauge variable for relabelling spatial points
idx_br      = 12   # rescaled B^r -> b^r - time derivative of shift
idx_lapse   = 13   # lapse - gauge variable for time slicing
```

What relates to the matter in engrenage?

```python
#uservariables.py

# hard code number of ghosts to 3 here
num_ghosts = 3

# This file provides the list of (rescaled) variables to be evolved and
# assigns each one an index and its parity
# For description of the data structure see https://github.com/GRChombo/engrenage/wiki/Useful-code-background

idx_u       = 0    # scalar field
idx_v       = 1    # scalar field conjugate momentum (roughly the time derivative of u)
idx_phi     = 2    # conformal factor of metric, \gamma_ij = e^{4 \phi} \bar \gamma_ij
idx_hrr     = 3    # rescaled \epsilon_rr -> h_rr - deviation of rr component of the metric from flat
idx_htt     = 4    # rescaled \epsilon_tt -> h_tt - deviation of tt component of the metric from flat
idx_hpp     = 5    # rescaled \epsilon_pp -> h_pp - deviation of pp component of the metric from flat
idx_K       = 6    # mean curvature K
idx_arr     = 7    # rescaled \tilde A_rr -> a_rr - (roughly) time derivative of hrr
idx_att     = 8    # rescaled \tilde A_tt -> a_tt - (roughly) time derivative of htt
idx_app     = 9    # rescaled \tilde A_pp -> a_pp - (roughly) time derivative of hpp
idx_lambdar = 10   # rescaled \bar\Lambda -> lambda^r
idx_shiftr  = 11   # rescaled \beta^r -> radial shift - gauge variable for relabelling spatial points
idx_br      = 12   # rescaled B^r -> b^r - time derivative of shift
idx_lapse   = 13   # lapse - gauge variable for time slicing
```

The lambdar is for numerical stability - can mostly ignore

```python
#uservariables.py

# hard code number of ghosts to 3 here
num_ghosts = 3

# This file provides the list of (rescaled) variables to be evolved and
# assigns each one an index and its parity
# For description of the data structure see https://github.com/GRChombo/engrenage/wiki/Useful-code-background

idx_u       = 0    # scalar field
idx_v       = 1    # scalar field conjugate momentum (roughly the time derivative of u)
idx_phi     = 2    # conformal factor of metric, \gamma_ij = e^{4 \phi} \bar \gamma_ij
idx_hrr     = 3    # rescaled \epsilon_rr -> h_rr - deviation of rr component of the metric from flat
idx_htt     = 4    # rescaled \epsilon_tt -> h_tt - deviation of tt component of the metric from flat
idx_hpp     = 5    # rescaled \epsilon_pp -> h_pp - deviation of pp component of the metric from flat
idx_K       = 6    # mean curvature K
idx_arr     = 7    # rescaled \tilde A_rr -> a_rr - (roughly) time derivative of hrr
idx_att     = 8    # rescaled \tilde A_tt -> a_tt - (roughly) time derivative of htt
idx_app     = 9    # rescaled \tilde A_pp -> a_pp - (roughly) time derivative of hpp
idx_lambdar = 10   # rescaled \bar\Lambda -> lambda^r
idx_shiftr  = 11   # rescaled \beta^r -> radial shift - gauge variable for relabelling spatial points
idx_br      = 12   # rescaled B^r -> b^r - time derivative of shift
idx_lapse   = 13   # lapse - gauge variable for time slicing

```

Four practical exercises

- Initial conditions - adding the scalar field to a BH
- Modifying equations of motion for the scalar
- Modifying the dynamical gauge for the metric
- Diagnostics - measuring scalar energy fluxes

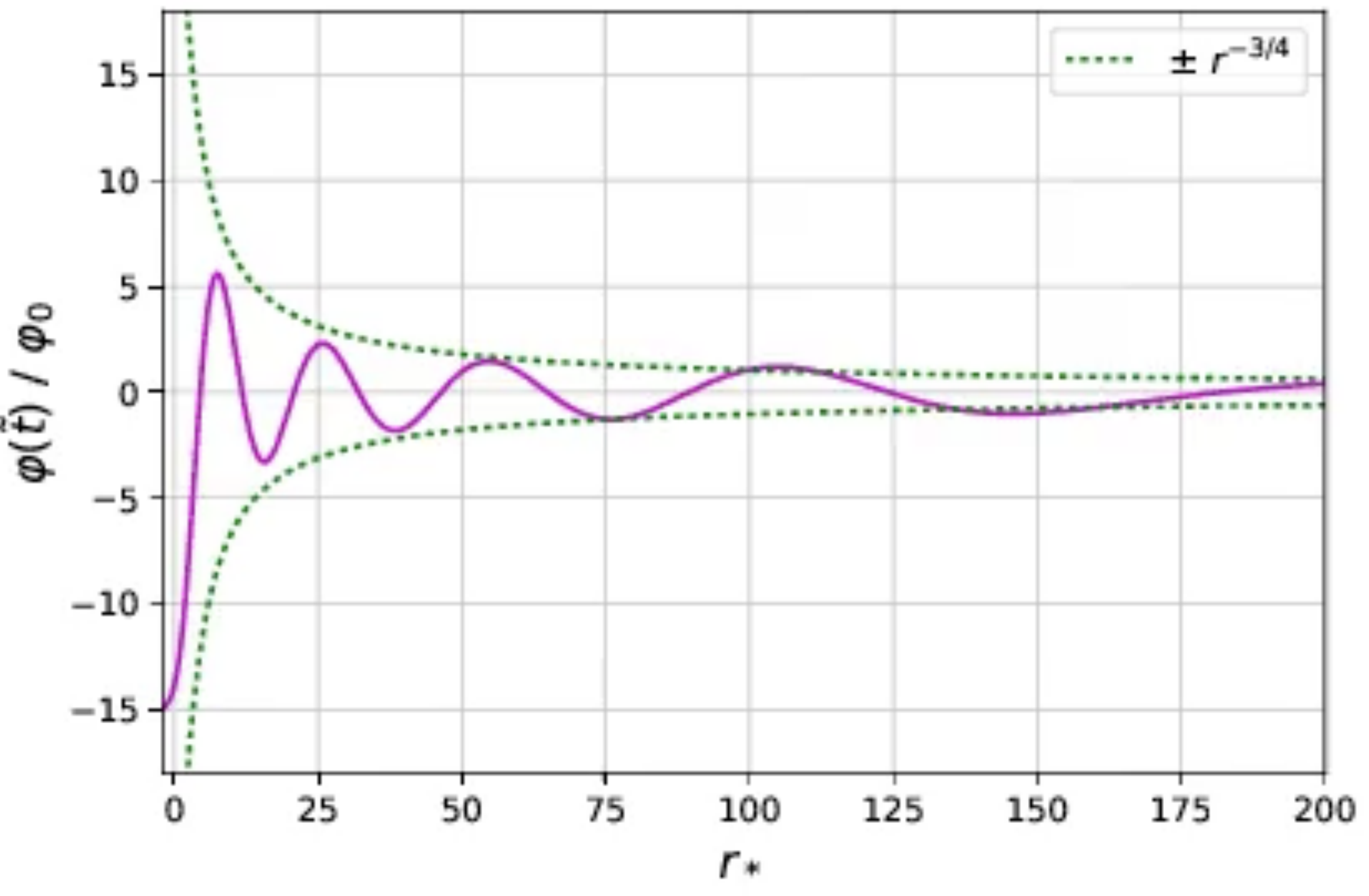
Quick tour of the code

Engrenage exercise 1: initial conditions

- Add a spatially constant scalar field $u_0 = 10^{-6}$ to the black hole initial conditions

- We need to make sure the Hamiltonian constraint is solved, so also set K to achieve this.

- ***Estimated lines of code required: 2-4***



https://inspirehep.net/literature/1731856

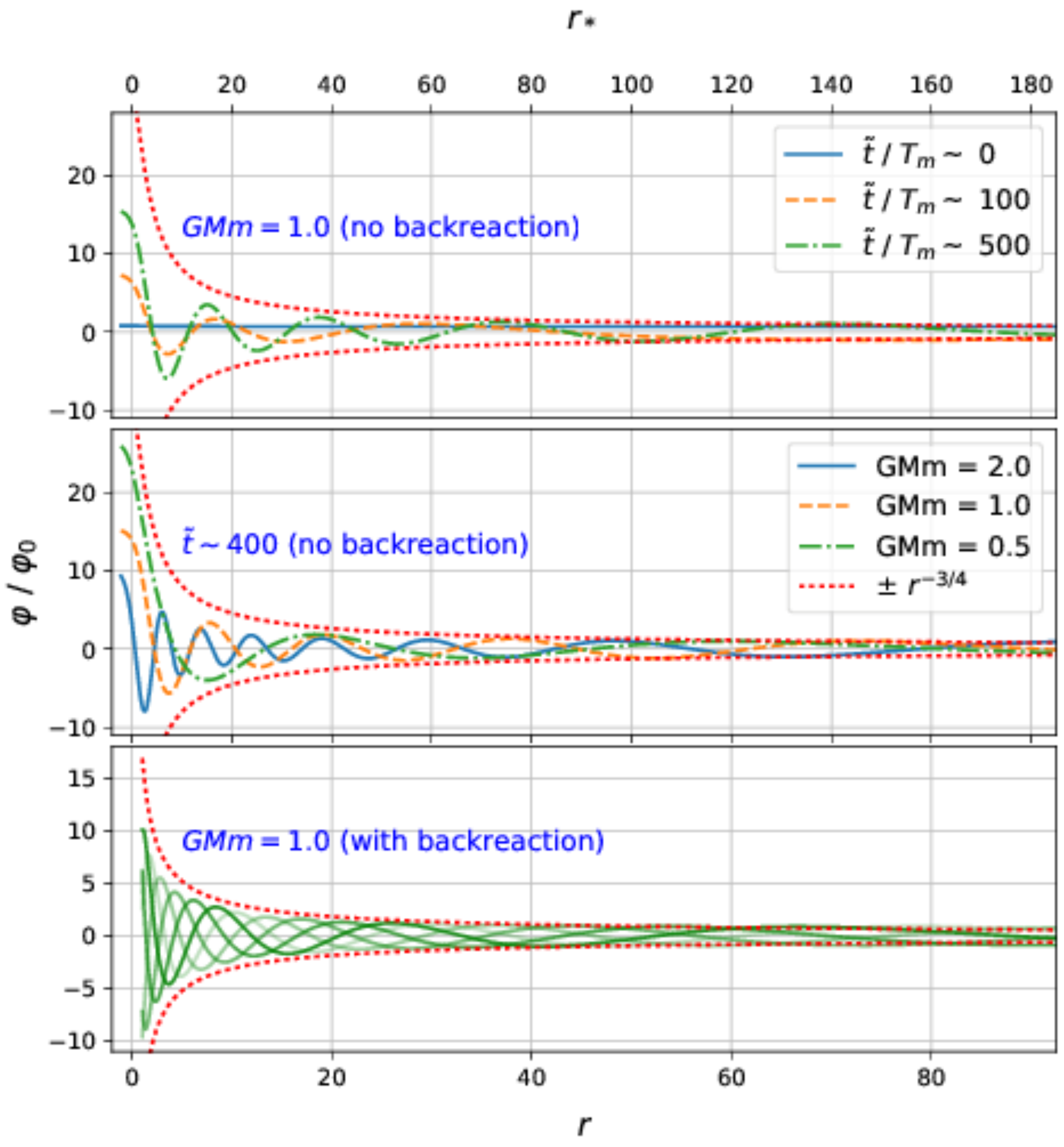
Engrenage exercise 2 - change the scalar eom

- Find and change the potential to:

$$V(u) = \frac{1}{2}\mu^2 u^2 + \frac{1}{4}\mu^2 \lambda u^4$$

Investigate the effect of changing the scalar mass μ and the self interaction λ.

- ***Estimated lines of code required: 2-3***



https://inspirehep.net/literature/1731856

Engrenage exercise 3 - change the gauge

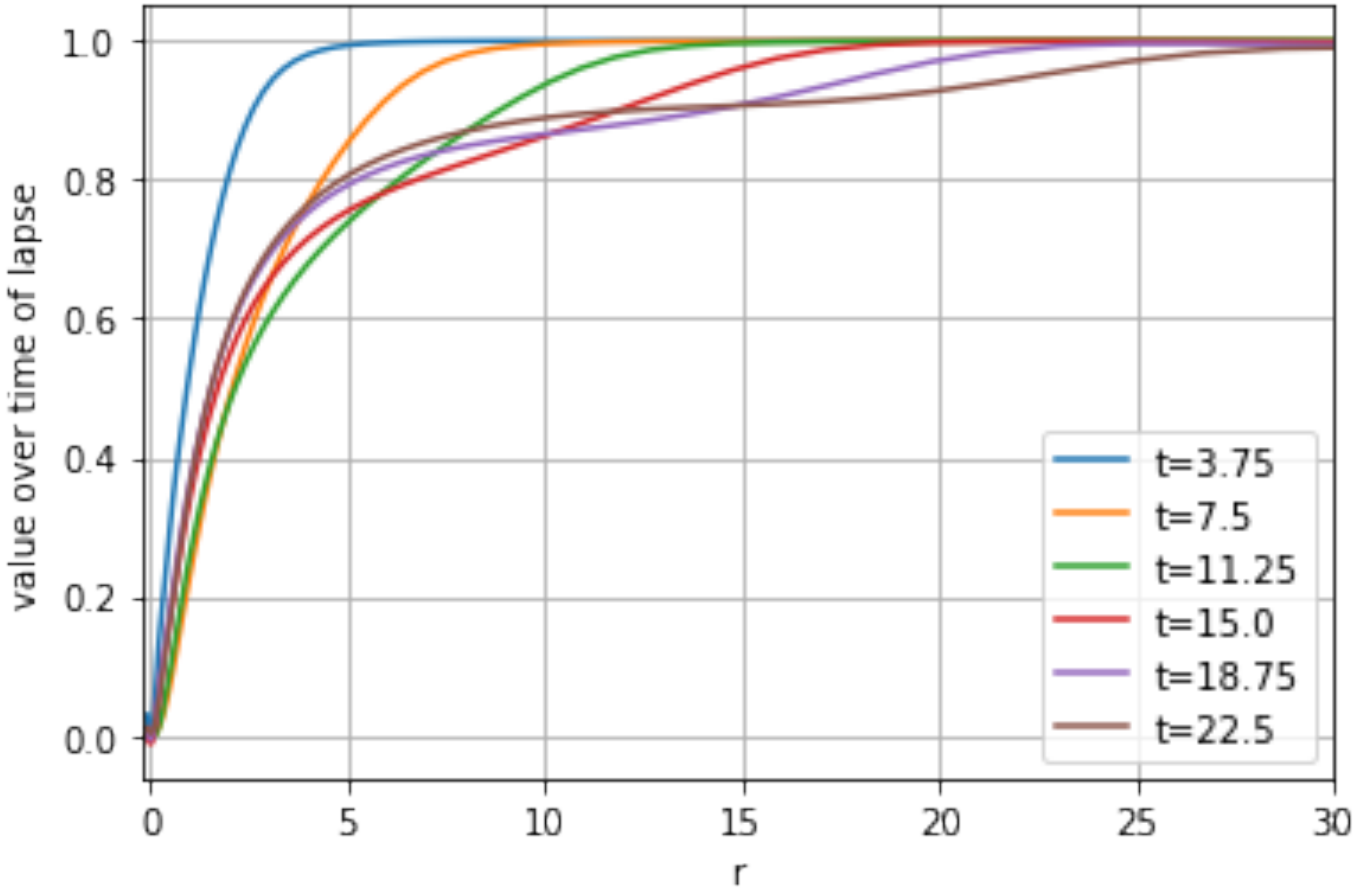
- Implement the shock avoiding gauge in https://inspirehep.net/literature/2111279

  $$\partial_\tau \alpha = -(\alpha^2 + \kappa)\ K$$

  with $\kappa = 0.05$

- What does it change about the evolution of the collapse of the lapse?

- How sensitive is stability to the choice of the parameter kappa?



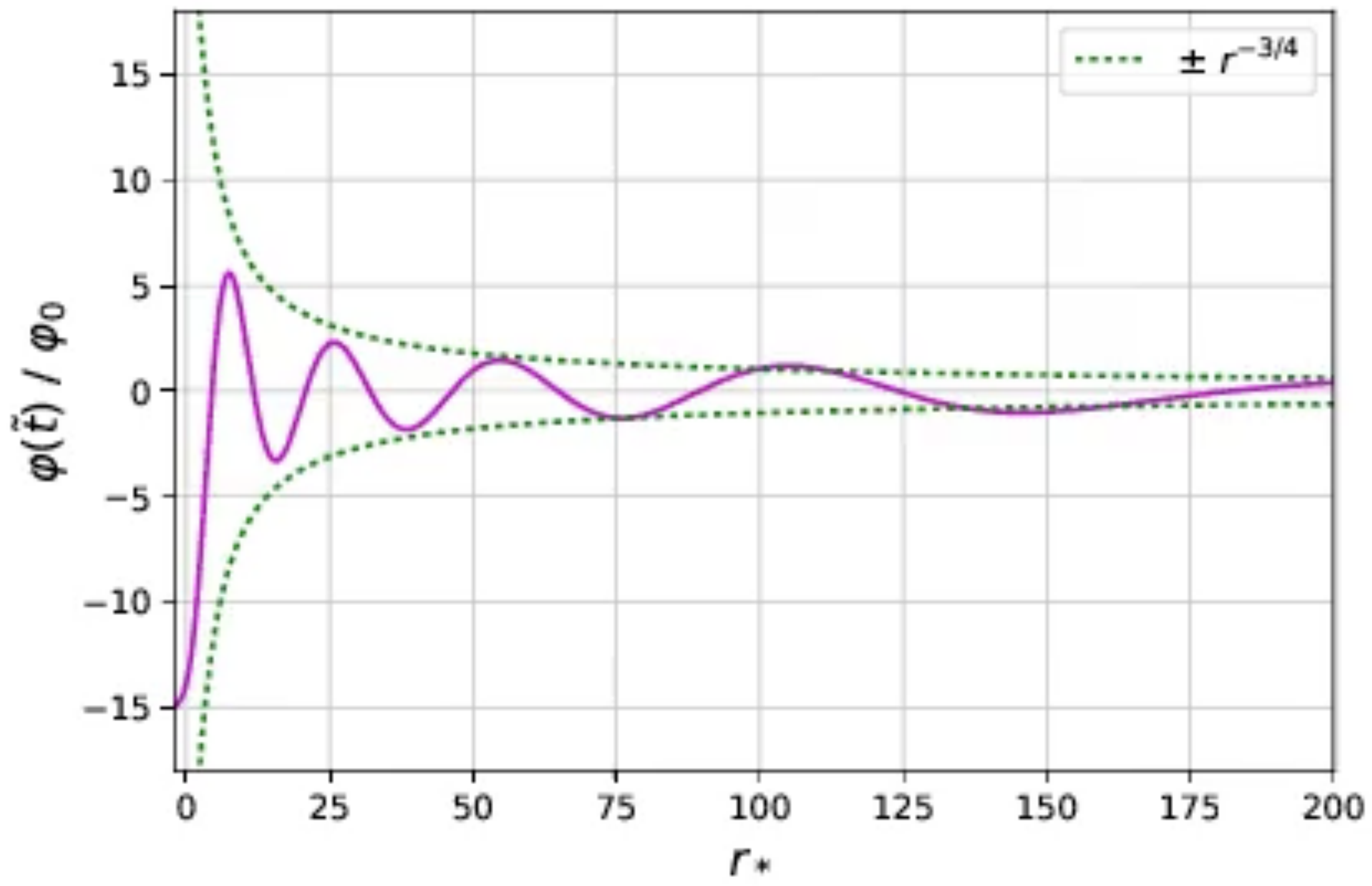
Engrenage exercise 4: diagnostics

- Write a diagnostic to calculate the radial flux across a spherical coordinate surface as a function of radius

$$F = 4\pi r^2 \sqrt{\gamma} S^r$$

- $S_i = -\, v \ \partial_j u$ is the momentum density of the scalar field

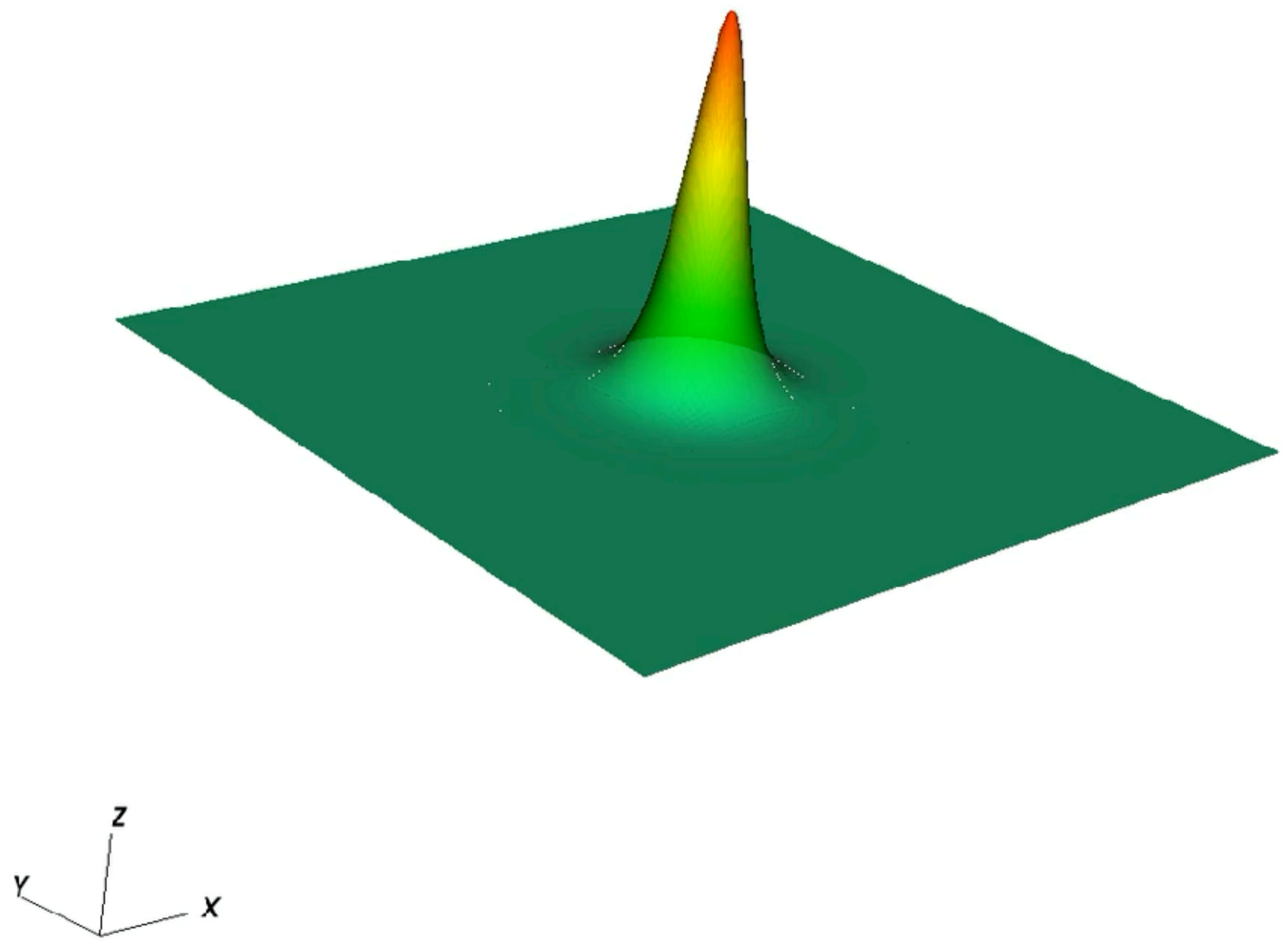
- What happens to the flux at small radii over time?



Extension - oscillaton

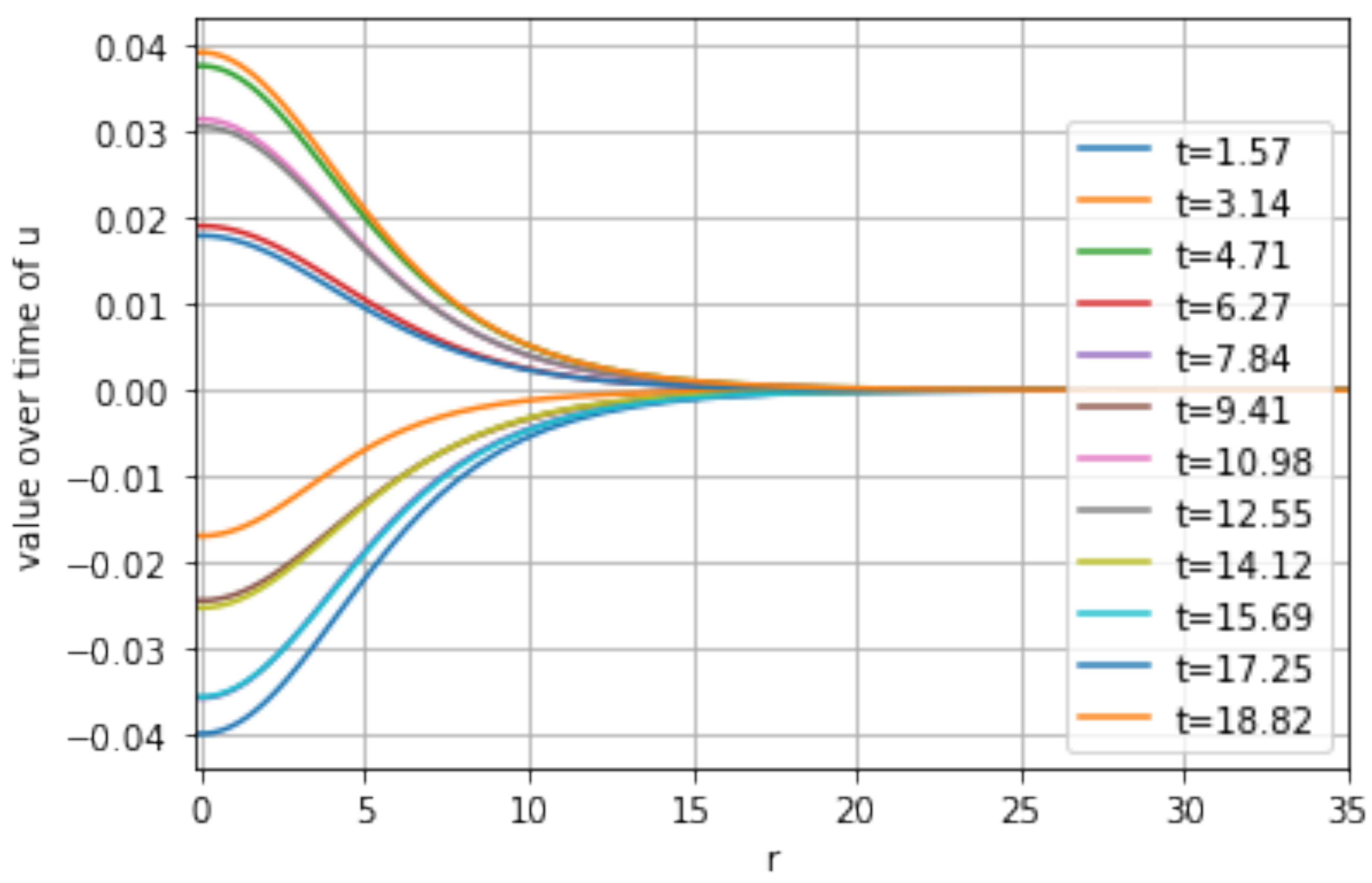
Engrenage oscillaton

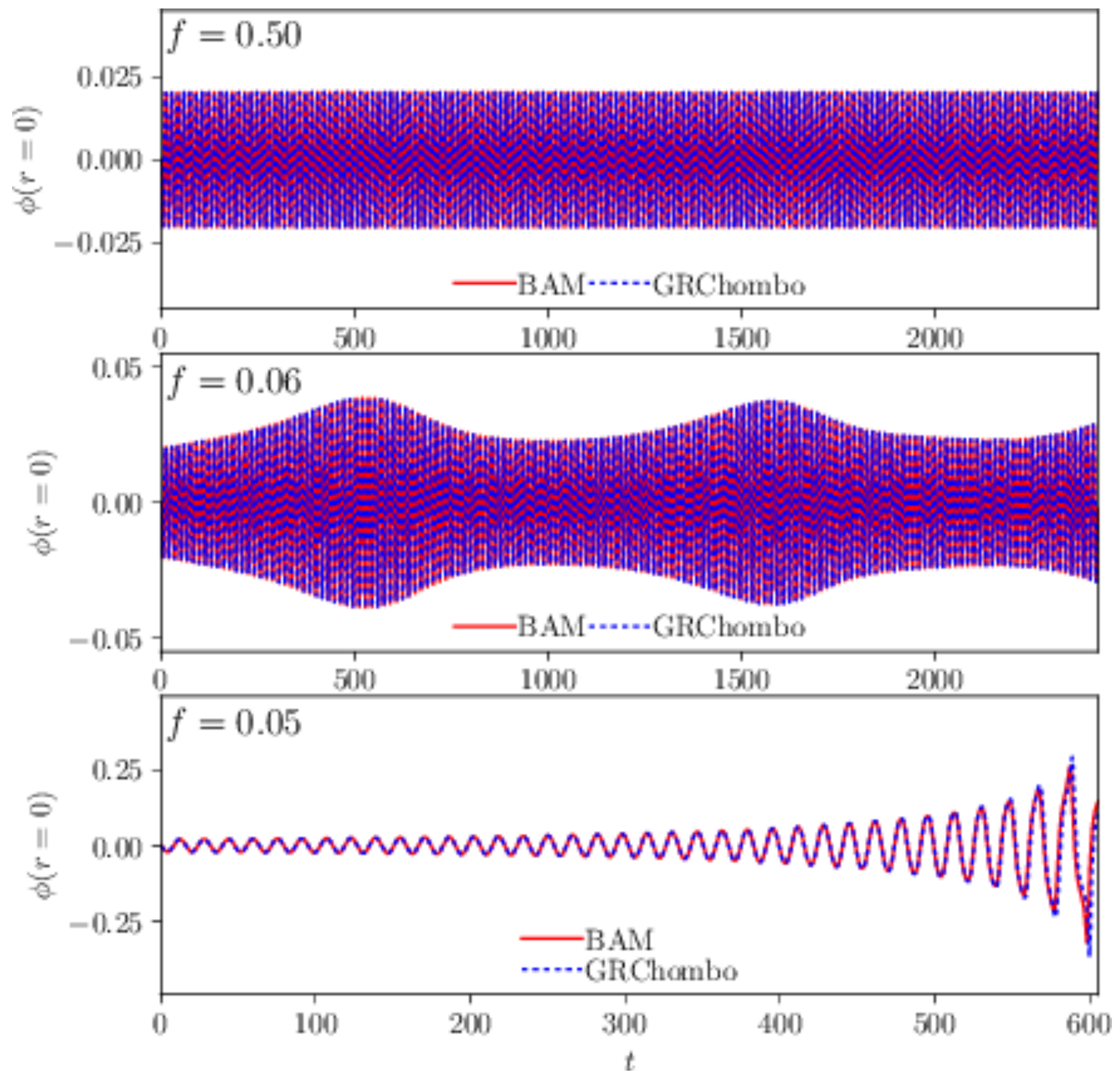
Field obeying massive Klein Gordon equation can have stable solitonic solutions with gravity



Engrenage oscillaton

- Repeat exercises 2-4 from the BH example for the oscillaton





https://inspirehep.net/literature/1687181

Please try it and contribute if you can!
Feedback welcome!