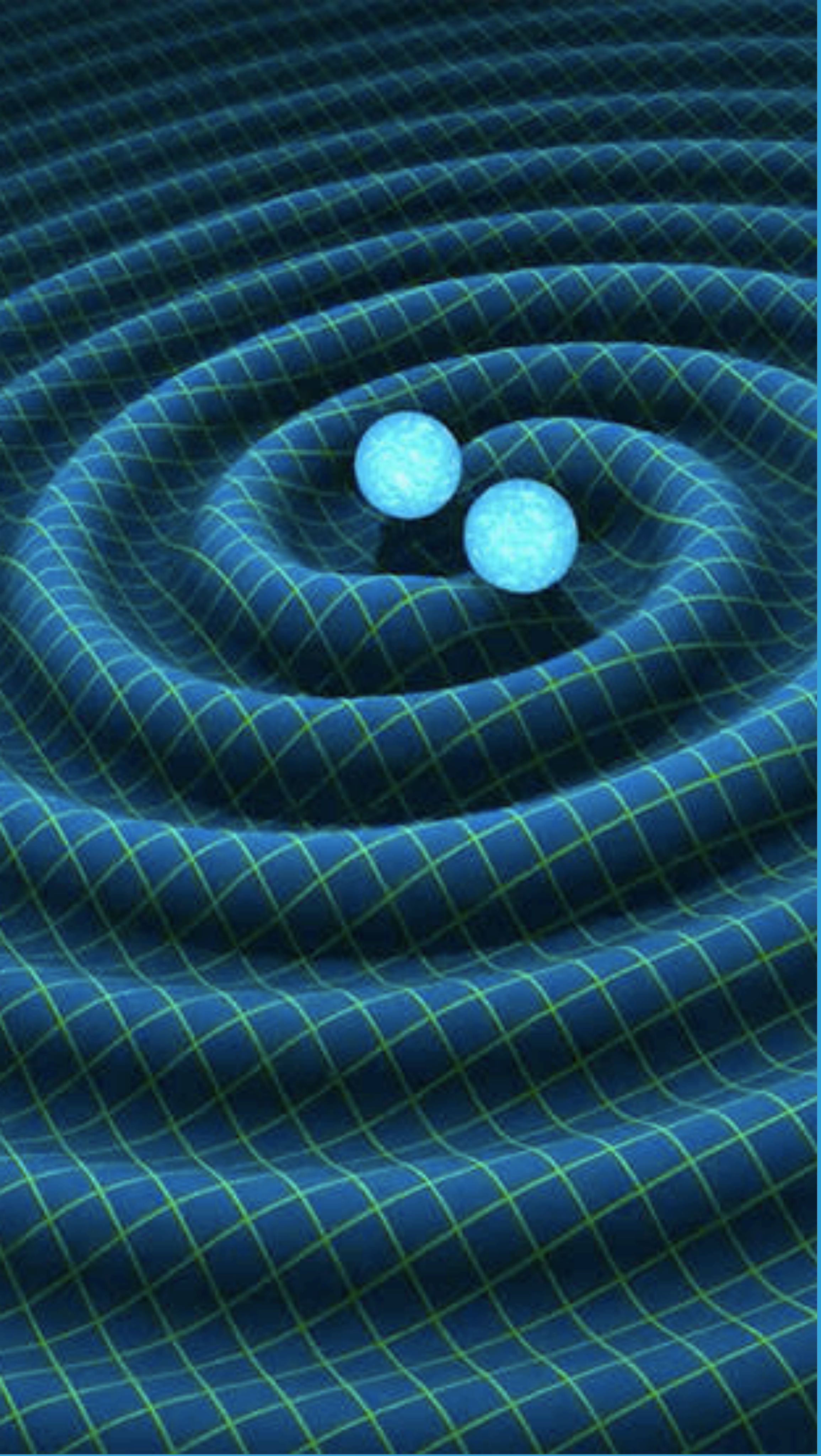
KATY CLOUGH

INTRO TO GRCHOMBO: THE BIG PICTURE

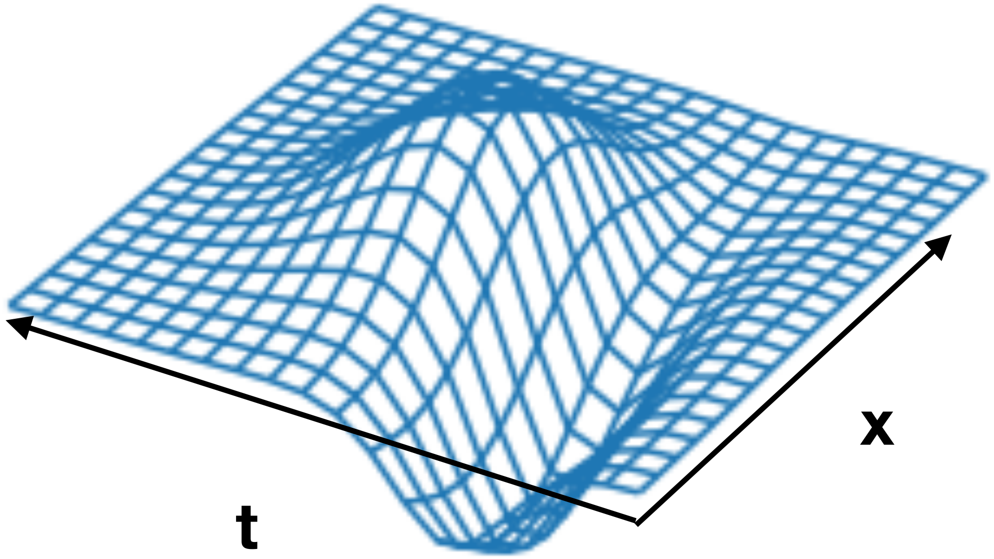


NUMERICAL RELATIVITY: BIG PICTURE

GR IN 2 MINUTES

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

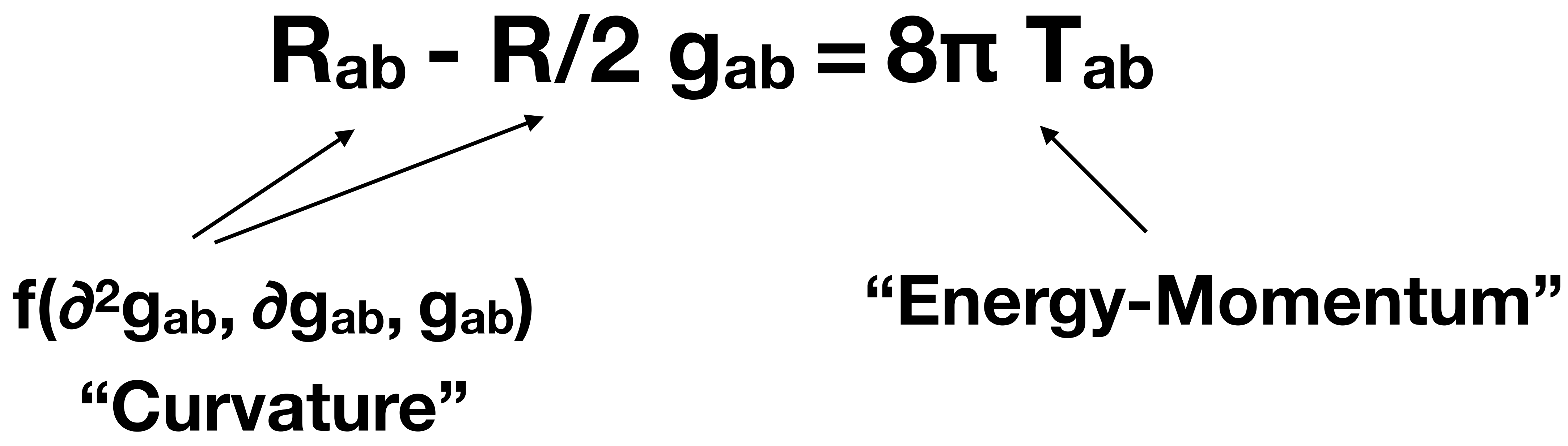
$$2\ h(x,t)\ dt\ dx$$



$$ds^2 = \begin{pmatrix} dt & dx \end{pmatrix} \underbrace{\begin{pmatrix} f(x,t) & h(x,t) \\ h(x,t) & g(x,t) \end{pmatrix}}_{\text{“The spacetime metric”}\quad g_{ab}(t,\vec{x})} \begin{pmatrix} dt \\ dx \end{pmatrix}$$

GR IN 2 MINUTES

"Matter tells spacetime how to curve…"



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

$f(\partial^2 g_{ab}, \partial g_{ab}, g_{ab})$

"Curvature"

"Energy-Momentum"

Can rearrange into form (using ADM decomposition):

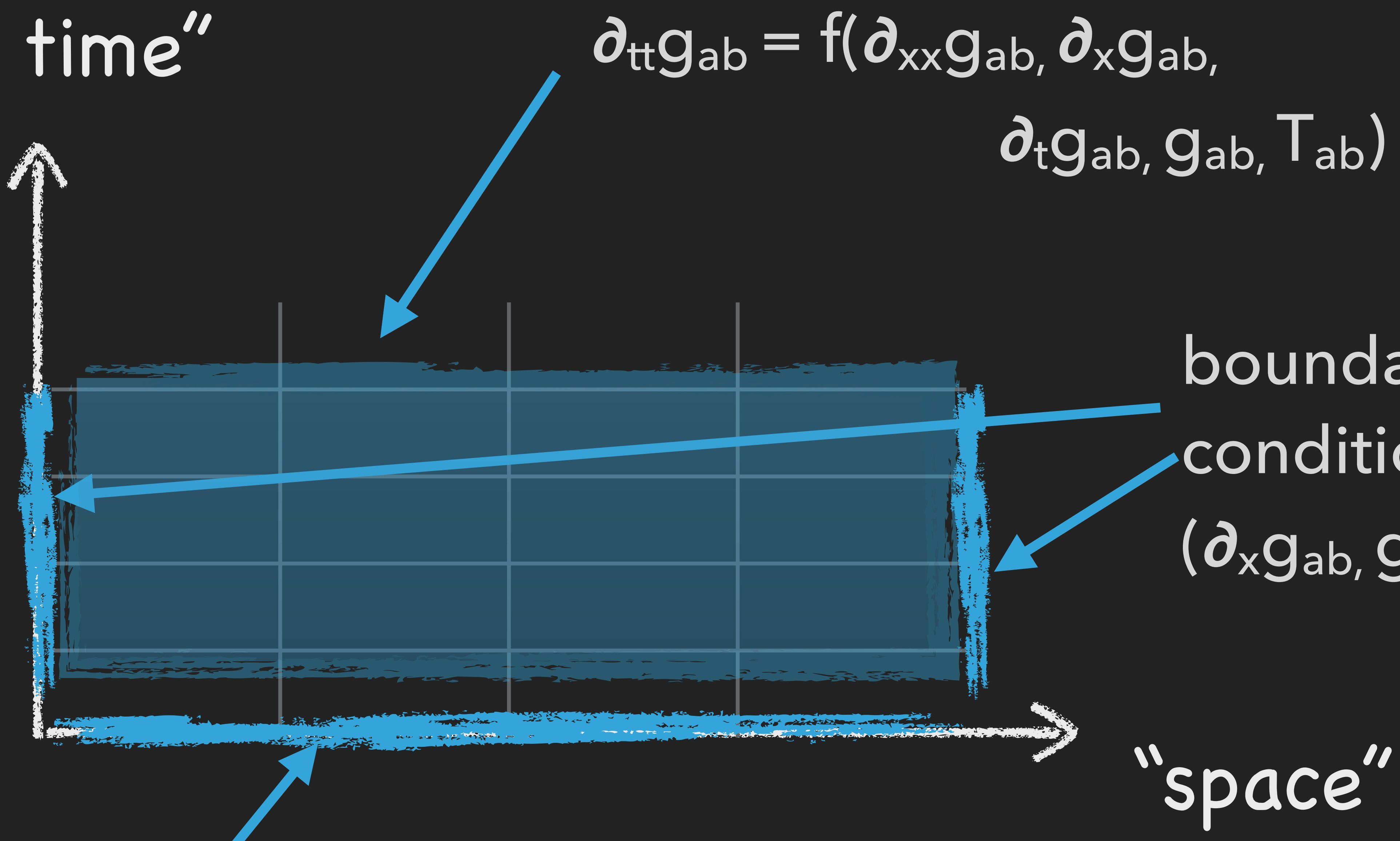
$$\partial_t(\partial_t g_{ab}) = f(\partial_{xx} g_{ab}, \partial_x g_{ab}, \partial_t g_{ab}, g_{ab}, T_{ab})$$

where $\partial_t g_{ab} \sim K_{ab}$

NR IN 2 MINUTES

Fill using Einstein equation

$\partial_{tt}g_{ab} = f(\partial_{xx}g_{ab}, \partial_x g_{ab}, \partial_t g_{ab}, g_{ab}, T_{ab})$

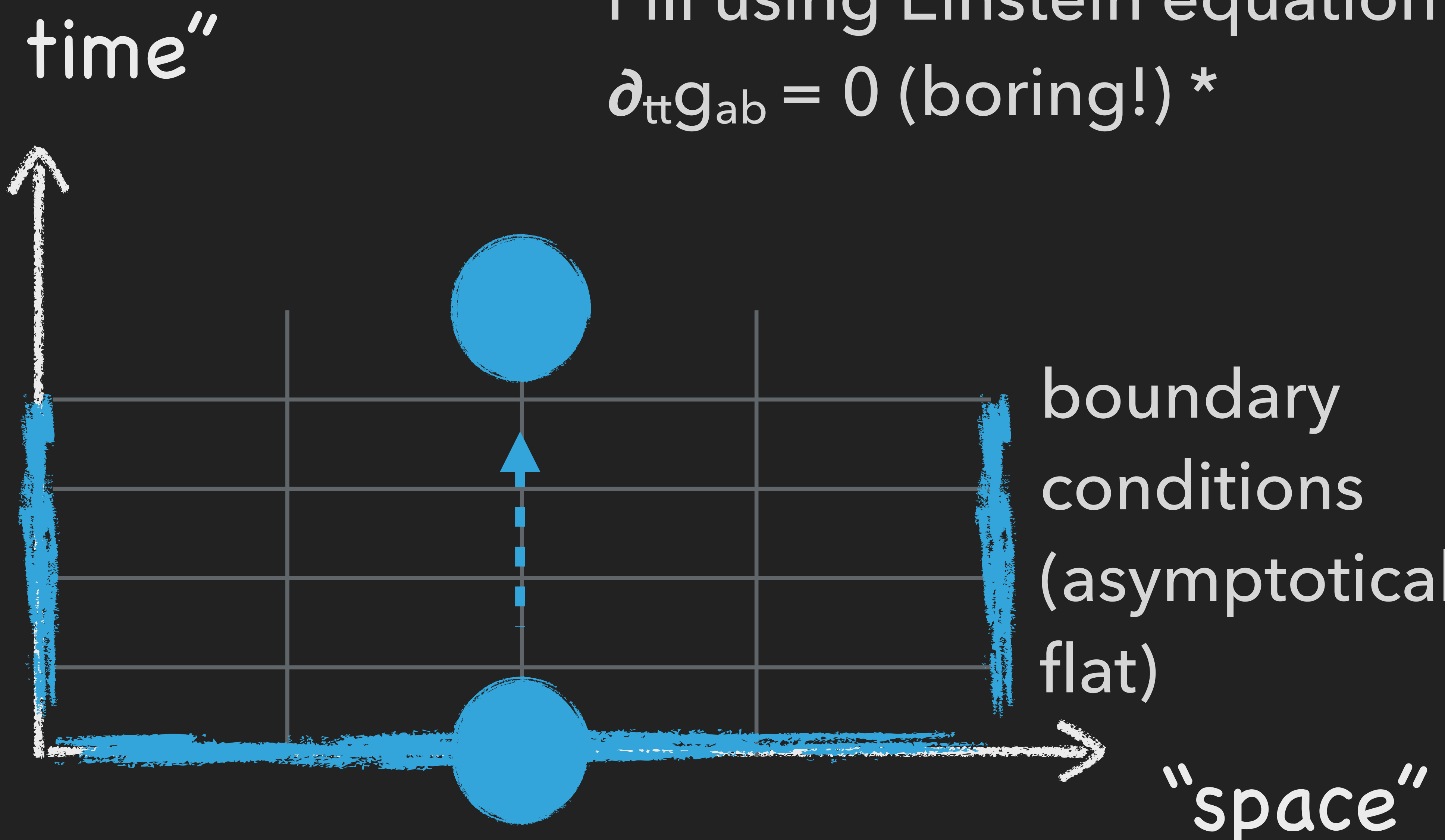


boundary conditions ($\partial_x g_{ab}, g_{ab}$)

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

NR IN 2 MINUTES

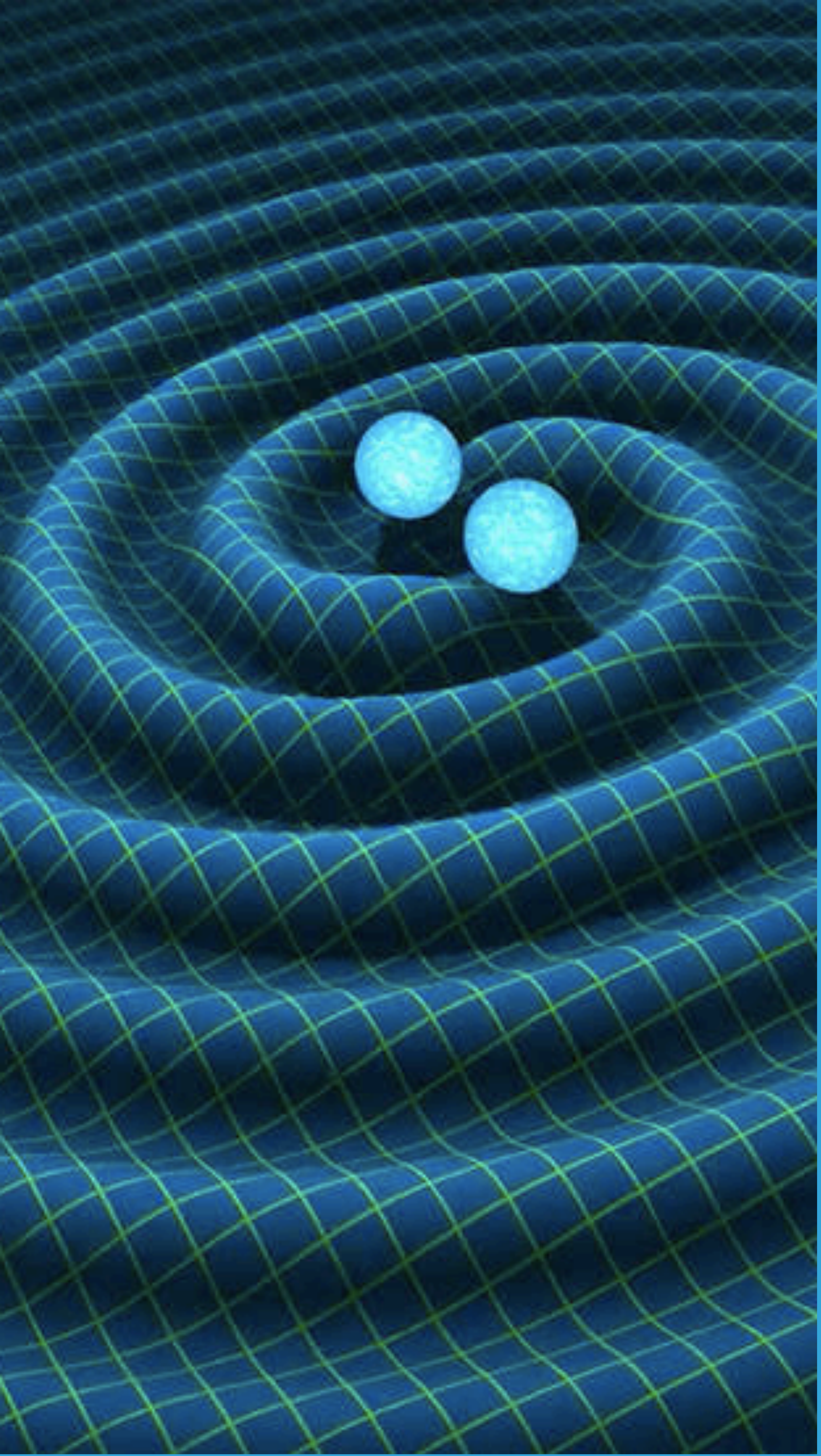
Fill using Einstein equation
$\partial_{tt} g_{ab} = 0$ (boring!) *



boundary conditions (asymptotically flat)

initial data ($\partial_t g_{ab}$= 0, g_{ab}= Schwarzschild metric)

* in practise since the NR coordinates are not typically the Schwarzschild ones, we see some gauge evolution



GRCHOMBO: BIG PICTURE (FOCUS ON PROGRAM FLOW)

THREE LEVELS : CHOMBO / GRCHOMBO / BINARYBH

- Chombo - overall program flow relevant to any initial value problem - AMR, AMRLevel, ChomboParameters

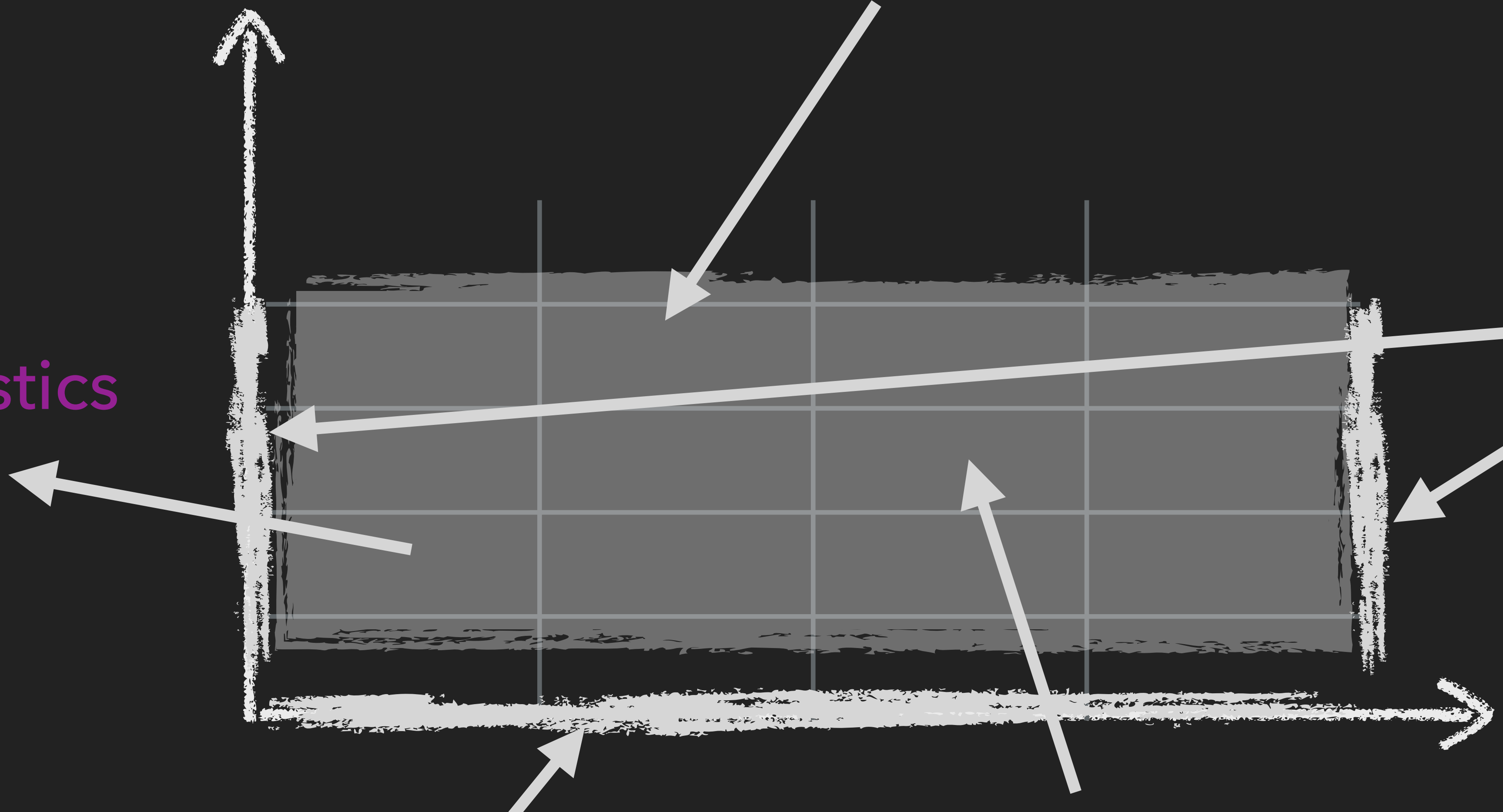
↓ INHERITANCE

- GRChombo - specific physics actions common to most GR problems - GRAMR, GRAMRLevel, SimulationParametersBase

↓ INHERITANCE

- BinaryBH - specific actions relevant to the Binary BH example - BHAMR, BinaryBHLevel, SimulationParameters

CHOMBO / GRCHOMBO / BINARYBH



Fill using rhs

field(t+dt) = df/dt(field) x dt

boundary conditions (boundary type -> fields in ghost cells)

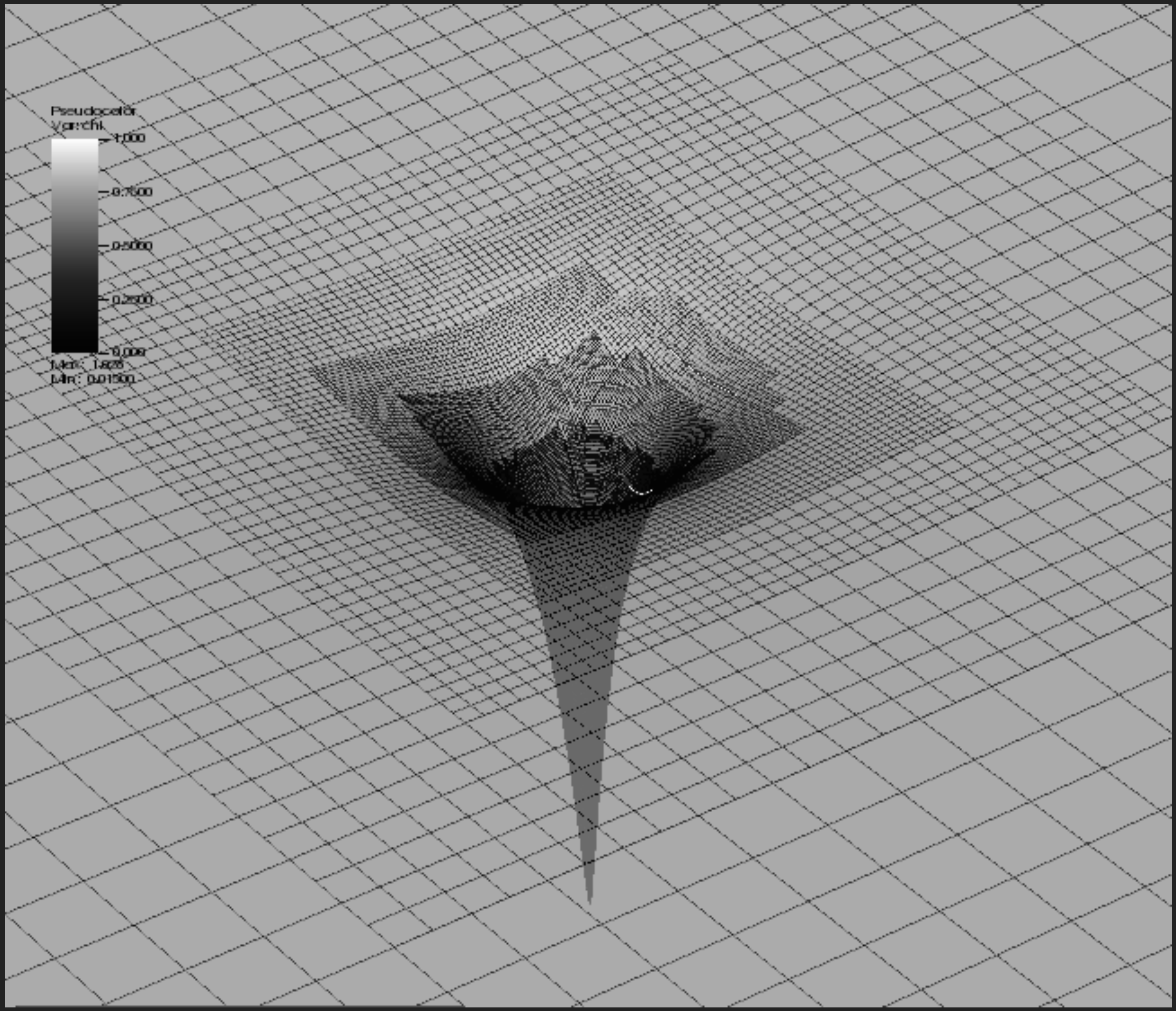
diagnostics

initial data (fields)

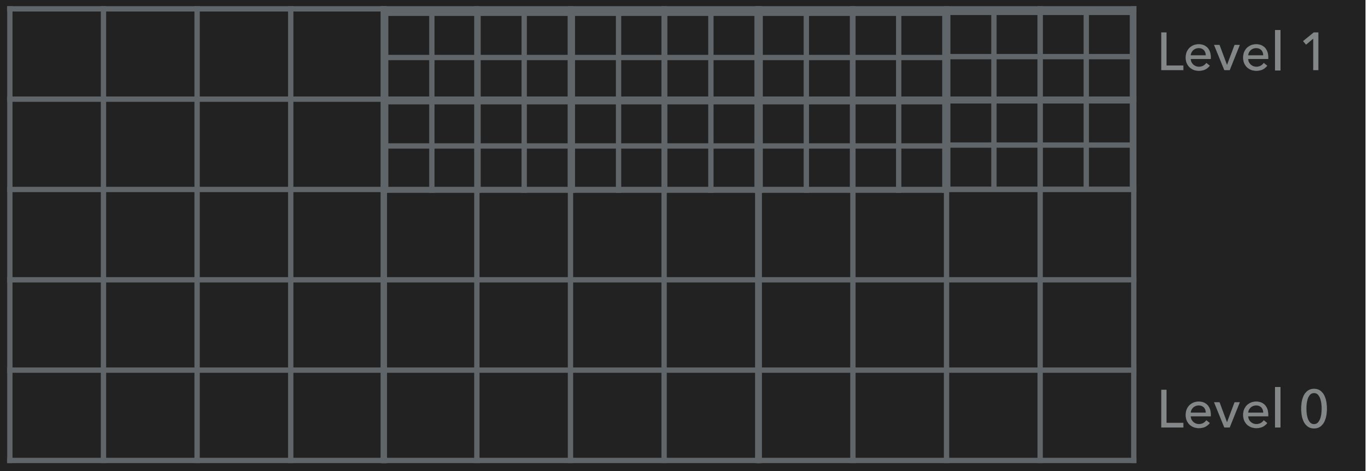
Calculate derivatives using finite differences, e.g.

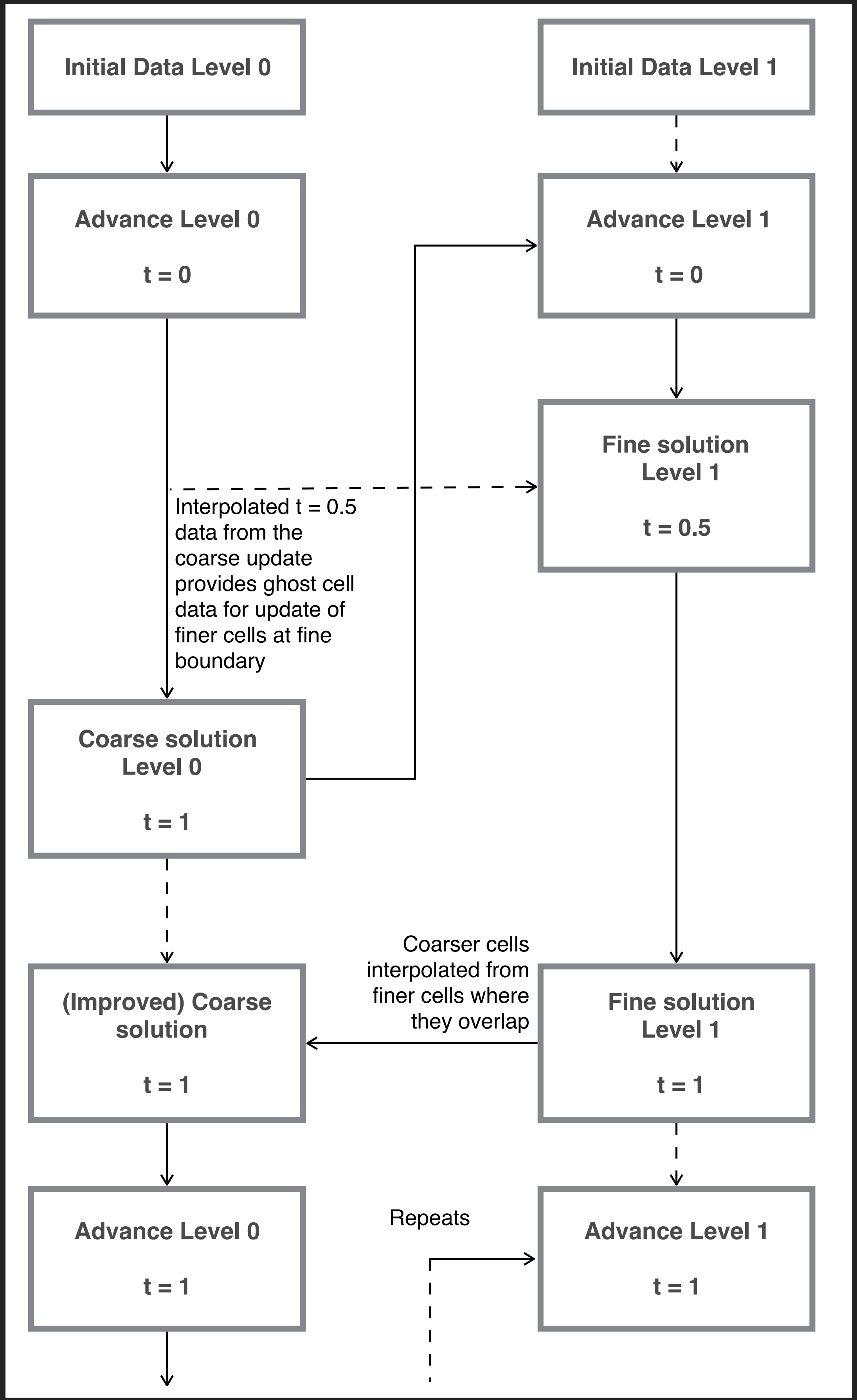
df/dx(field) = [f(x + dx) - f(x)] / dx

CHOMBO DEALS WITH THE ADAPTIVE MESH REFINEMENT (AMR)



AMR TIME STEPPING





Initial Data Level 0

Advance Level 0 t = 0

Interpolated t = 0.5 data from the coarse update provides ghost cell data for update of finer cells at fine boundary

Coarse solution Level 0 t = 1

(Improved) Coarse solution t = 1

Advance Level 0 t = 1

Initial Data Level 1

Advance Level 1 t = 0

Fine solution Level 1 t = 0.5

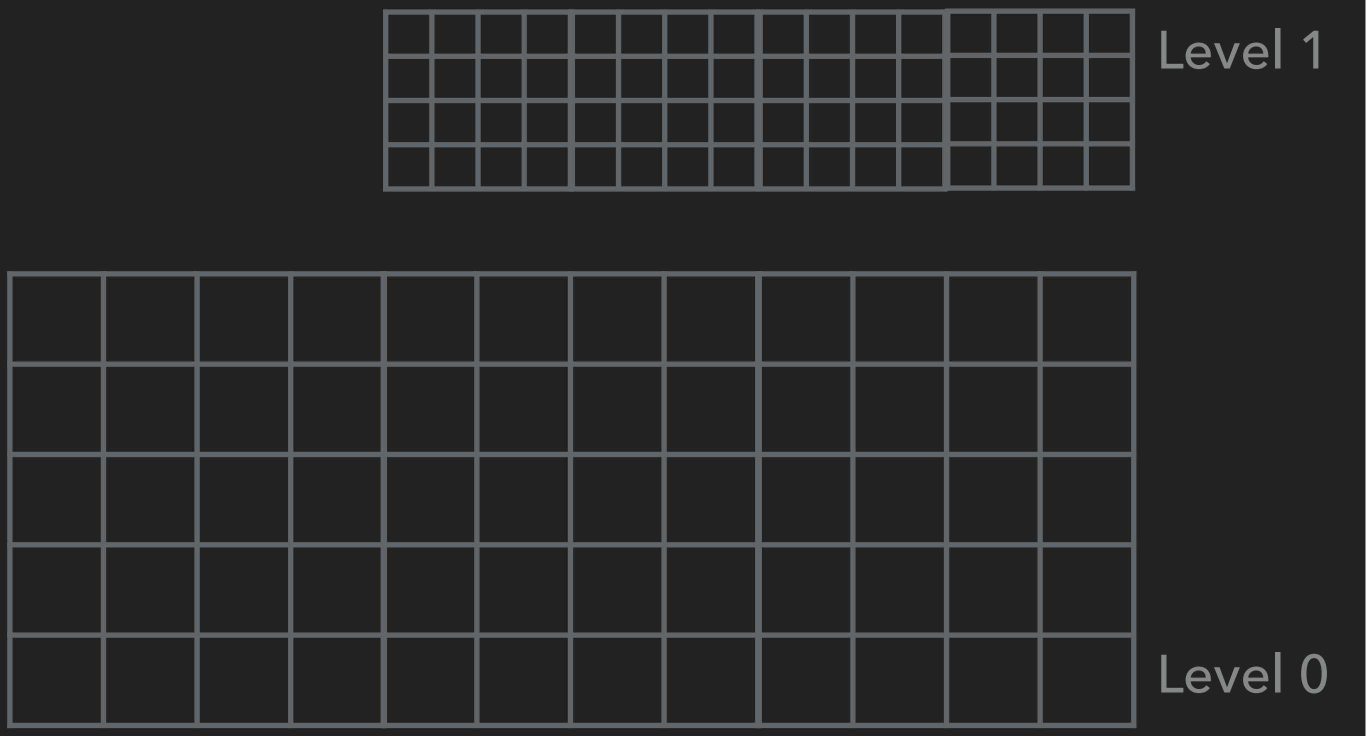
Coarser cells interpolated from finer cells where they overlap

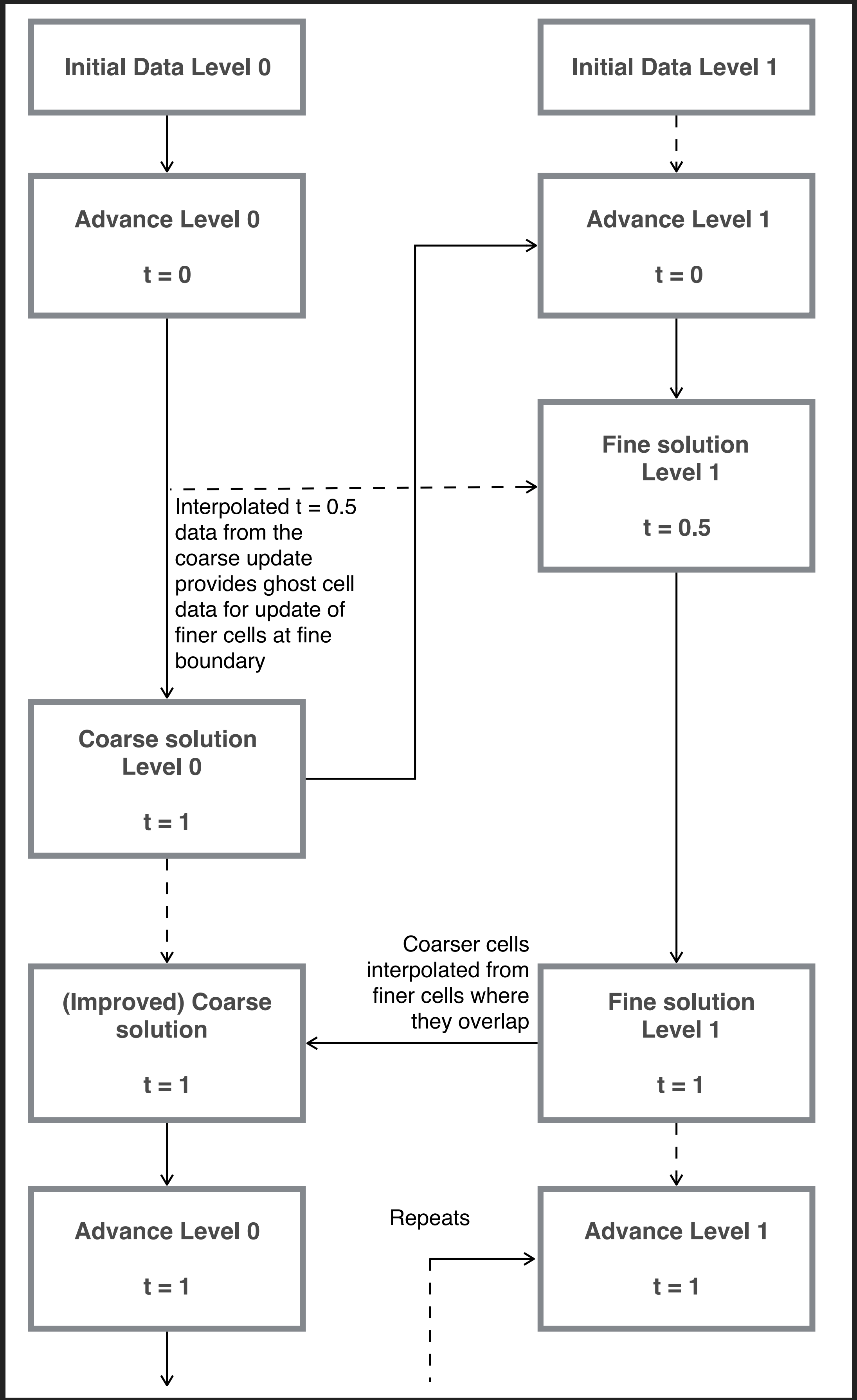
Fine solution Level 1 t = 1

Repeats

Advance Level 1 t = 1

AMR TIME STEPPING





Initial Data Level 0

Advance Level 0

t = 0

Interpolated t = 0.5 data from the coarse update provides ghost cell data for update of finer cells at fine boundary

Coarse solution Level 0

t = 1

(Improved) Coarse solution

t = 1

Advance Level 0

t = 1

Initial Data Level 1

Advance Level 1

t = 0

Fine solution Level 1

t = 0.5

Coarser cells interpolated from finer cells where they overlap

Fine solution Level 1

t = 1

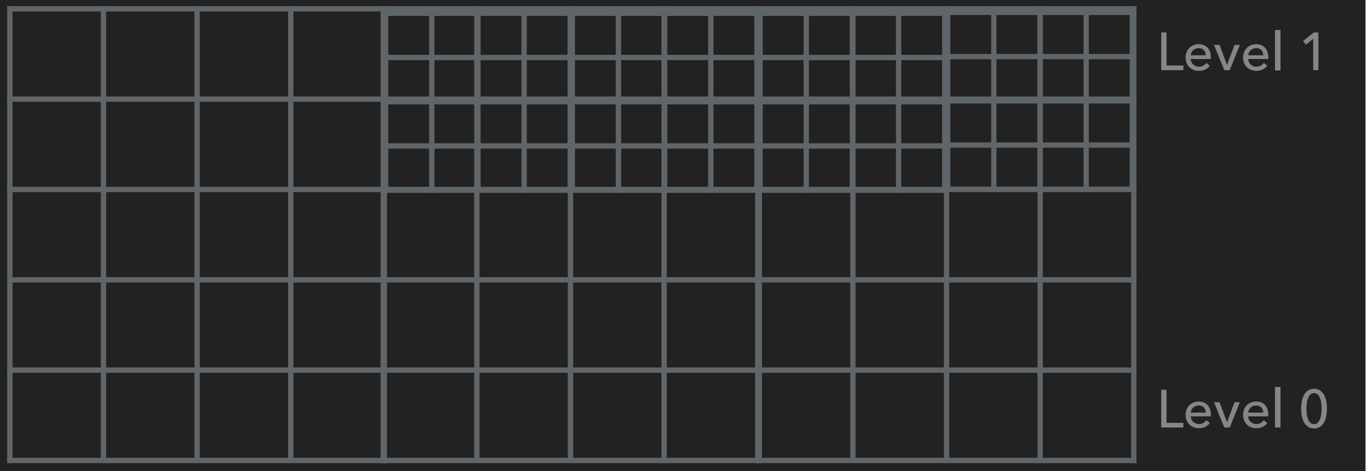
Repeats

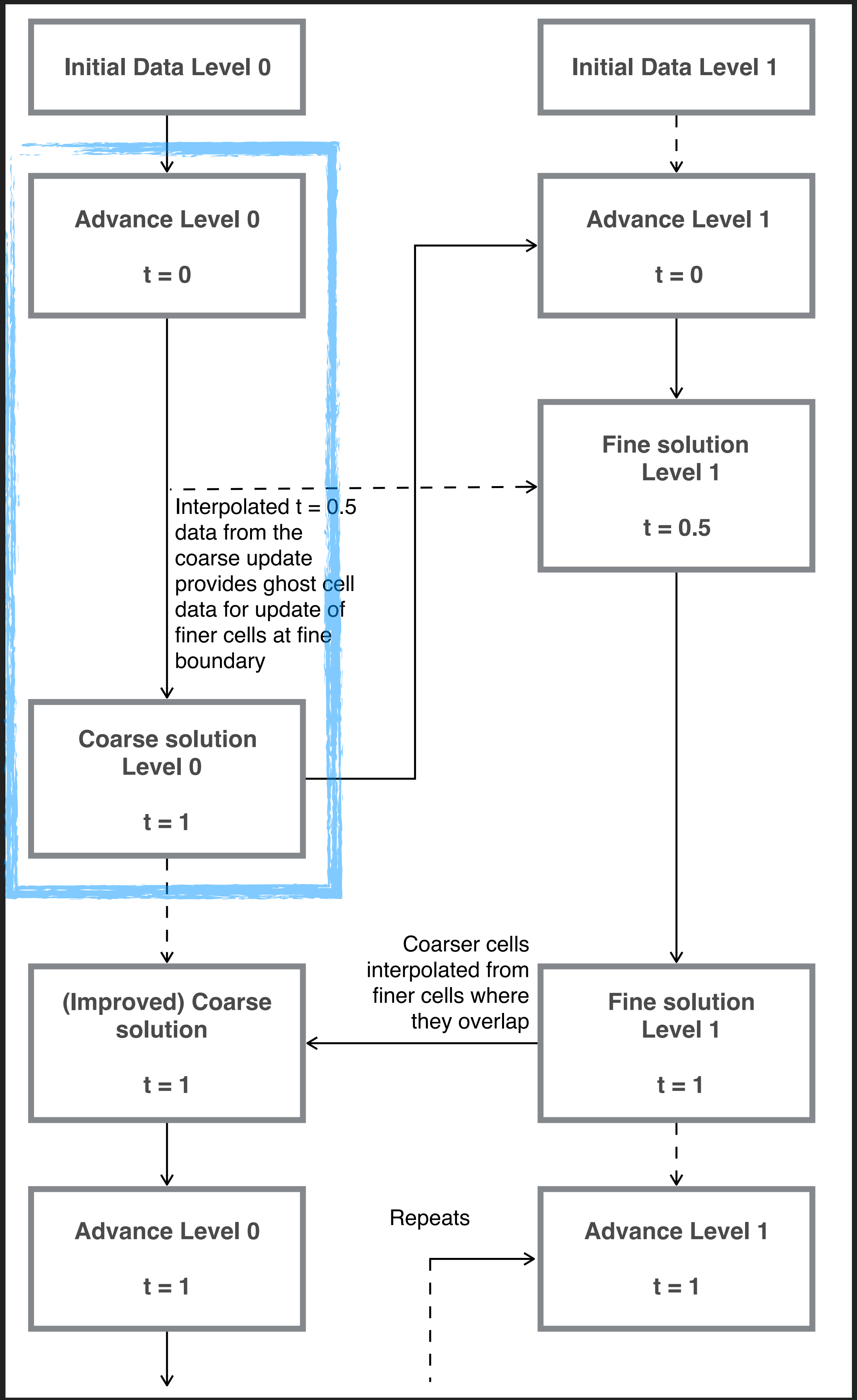
Advance Level 1

t = 1

AMR TIME STEPPING

- Each step is not really a single step but a series of Runge Kutta (RK4) substeps





Initial Data Level 0

Initial Data Level 1

Advance Level 0

t = 0

Advance Level 1

t = 0

Fine solution Level 1

t = 0.5

Interpolated t = 0.5 data from the coarse update provides ghost cell data for update of finer cells at fine boundary

Coarse solution Level 0

t = 1

Coarser cells interpolated from finer cells where they overlap

(Improved) Coarse solution

t = 1

Fine solution Level 1

t = 1

Advance Level 0

t = 1

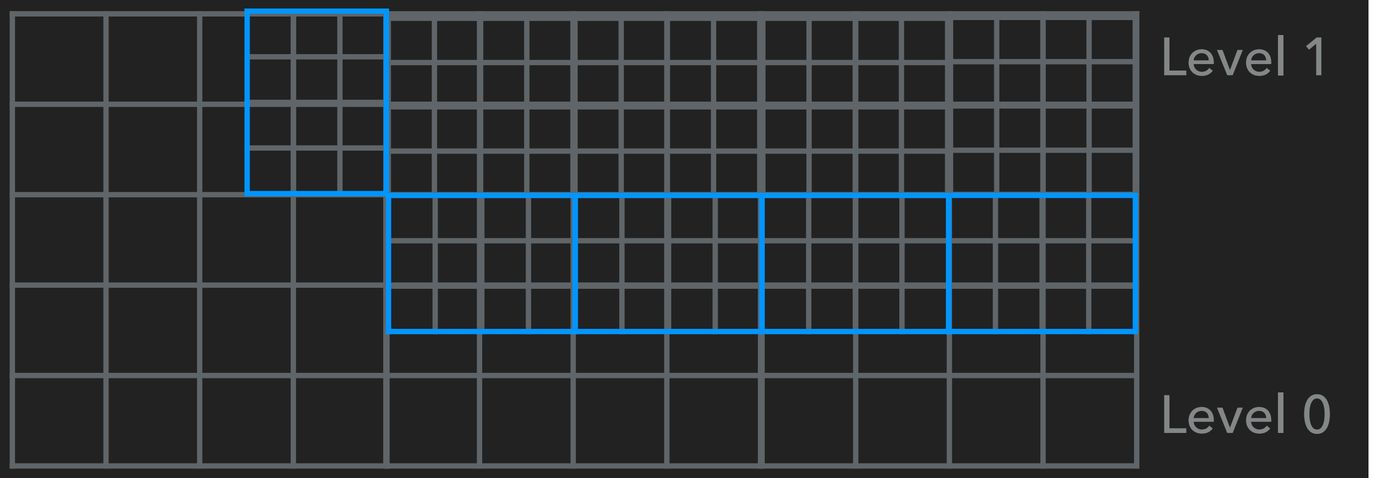
Repeats

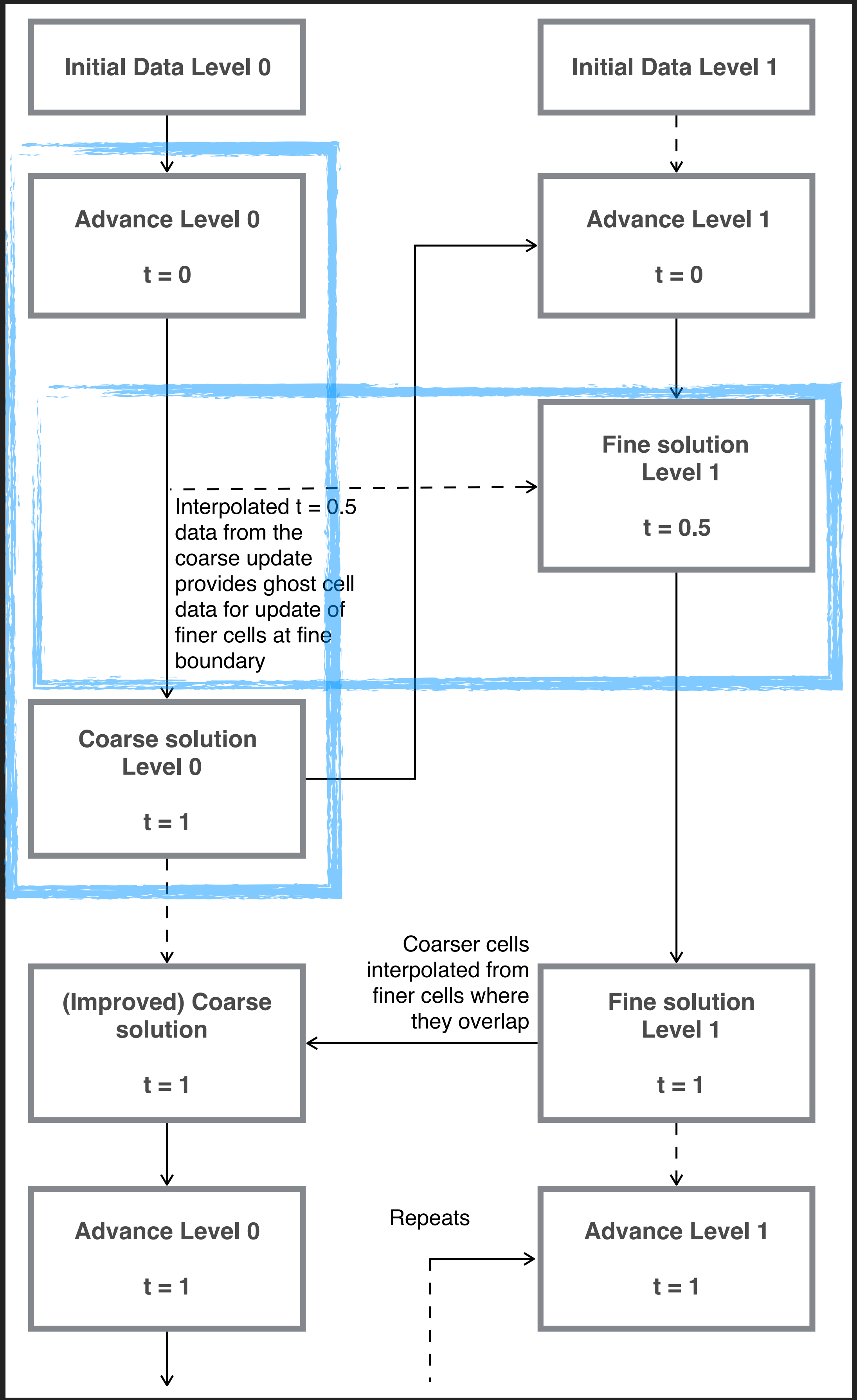
Advance Level 1

t = 1

AMR TIME STEPPING

- Each step is not really a single step but a series of Runge Kutta (RK4) substeps
- Data from coarser level is interpolated in both space and time to fill finer level ghost cells at level boundaries




Initial Data Level 0

Initial Data Level 1

Advance Level 0

t = 0

Advance Level 1

t = 0

Fine solution Level 1

t = 0.5

Interpolated t = 0.5 data from the coarse update provides ghost cell data for update of finer cells at fine boundary

Coarse solution Level 0

t = 1

(Improved) Coarse solution

t = 1

Coarser cells interpolated from finer cells where they overlap

Fine solution Level 1

t = 1

Advance Level 0

t = 1

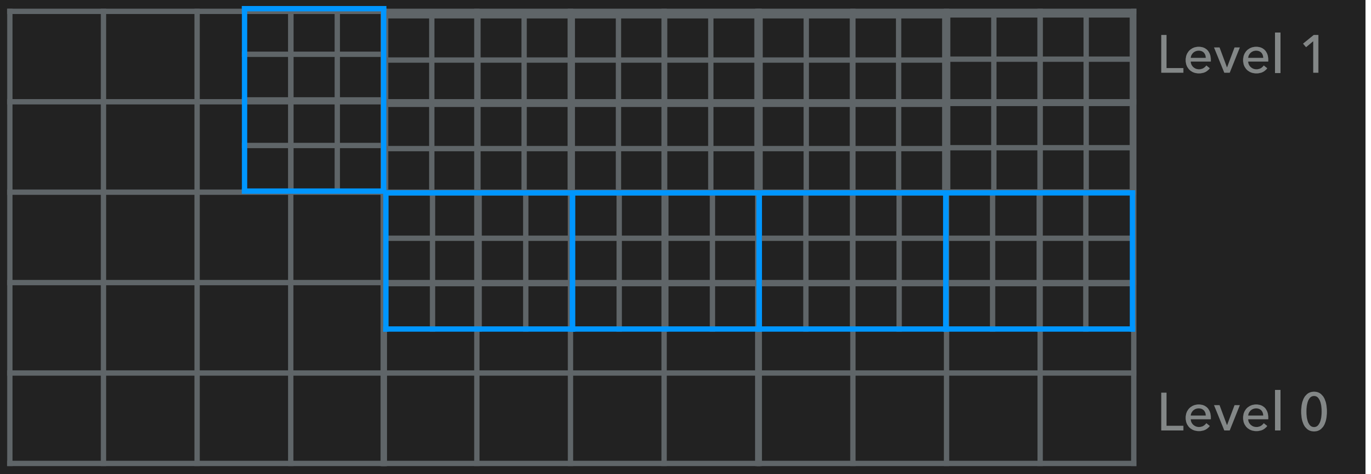
Repeats

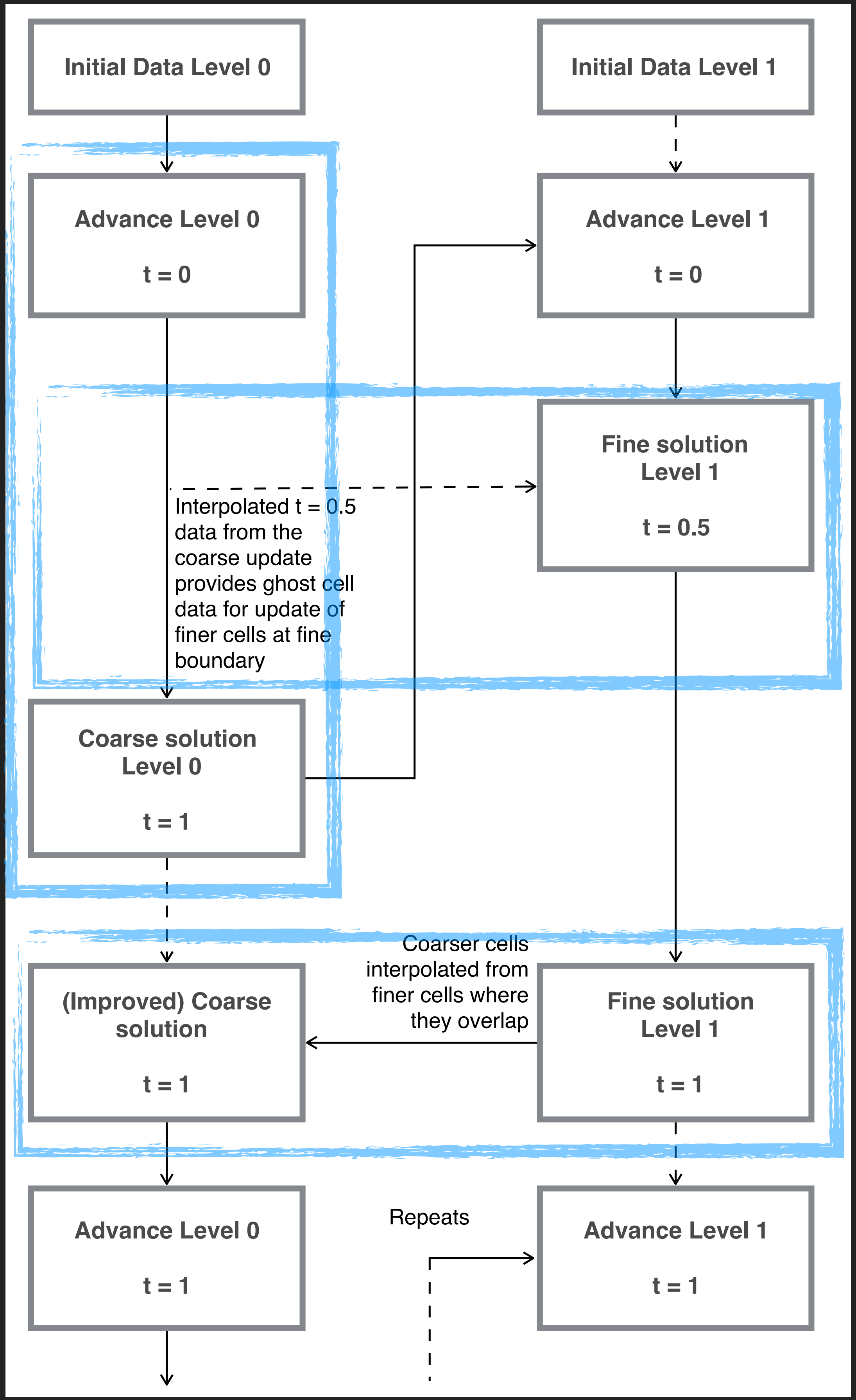
Advance Level 1

t = 1

AMR TIME STEPPING

- Each step is not really a single step but a series of Runge Kutta (RK4) substeps
- Data from coarser level is interpolated in both space and time to fill finer level ghost cells at level boundaries
- Level 0 is not finalised until Level 1 is => coarser levels have to wait for finer ones to end, so each level is processed in serial





Initial Data Level 0

Initial Data Level 1

Advance Level 0

t = 0

Advance Level 1

t = 0

Interpolated t = 0.5 data from the coarse update provides ghost cell data for update of finer cells at fine boundary

Fine solution Level 1

t = 0.5

Coarse solution Level 0

t = 1

Coarser cells interpolated from finer cells where they overlap

(Improved) Coarse solution

t = 1

Fine solution Level 1

t = 1

Advance Level 0

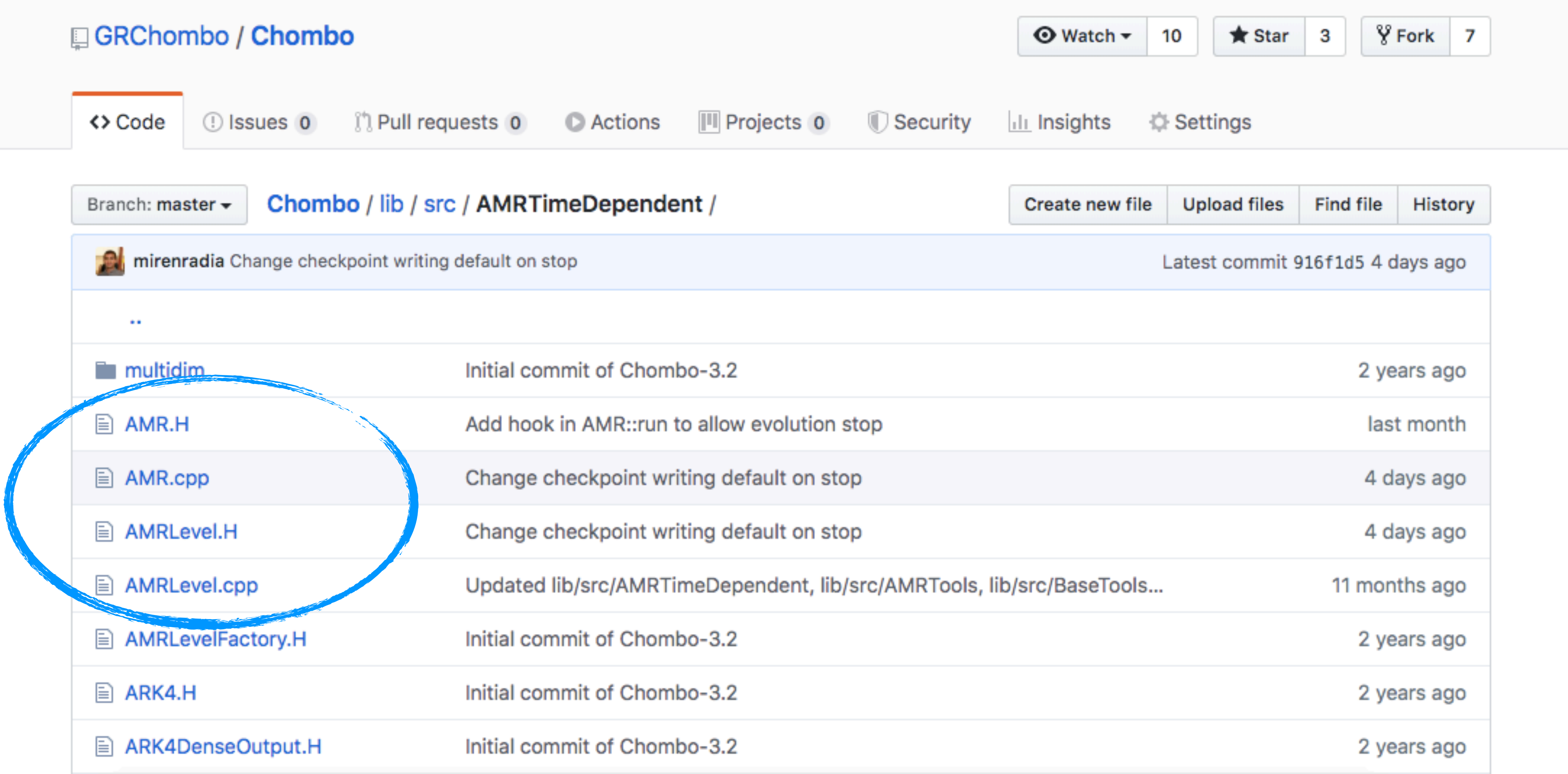
t = 1

Repeats

Advance Level 1

t = 1

WHERE ARE THE KEY CHOMBO FILES?



GRChombo / Chombo

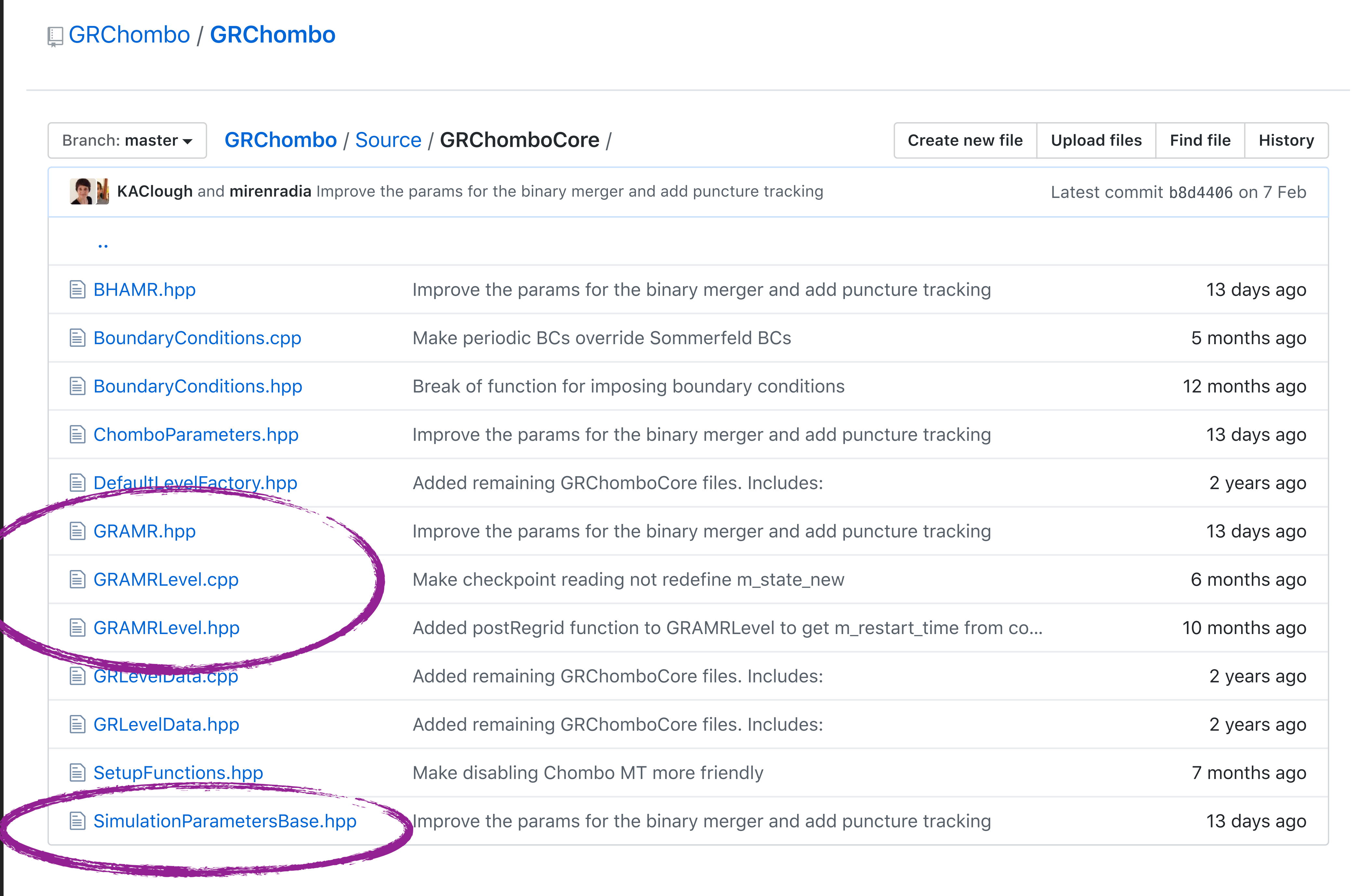
Chombo / lib / src / AMRTimeDependent /

mirenradia Change checkpoint writing default on stop — Latest commit 916f1d5 4 days ago

..

multidim	Initial commit of Chombo-3.2	2 years ago
AMR.H	Add hook in AMR::run to allow evolution stop	last month
AMR.cpp	Change checkpoint writing default on stop	4 days ago
AMRLevel.H	Change checkpoint writing default on stop	4 days ago
AMRLevel.cpp	Updated lib/src/AMRTimeDependent, lib/src/AMRTools, lib/src/BaseTools...	11 months ago
AMRLevelFactory.H	Initial commit of Chombo-3.2	2 years ago
ARK4.H	Initial commit of Chombo-3.2	2 years ago
ARK4DenseOutput.H	Initial commit of Chombo-3.2	2 years ago

WHERE ARE THE KEY GRCHOMBO FILES?



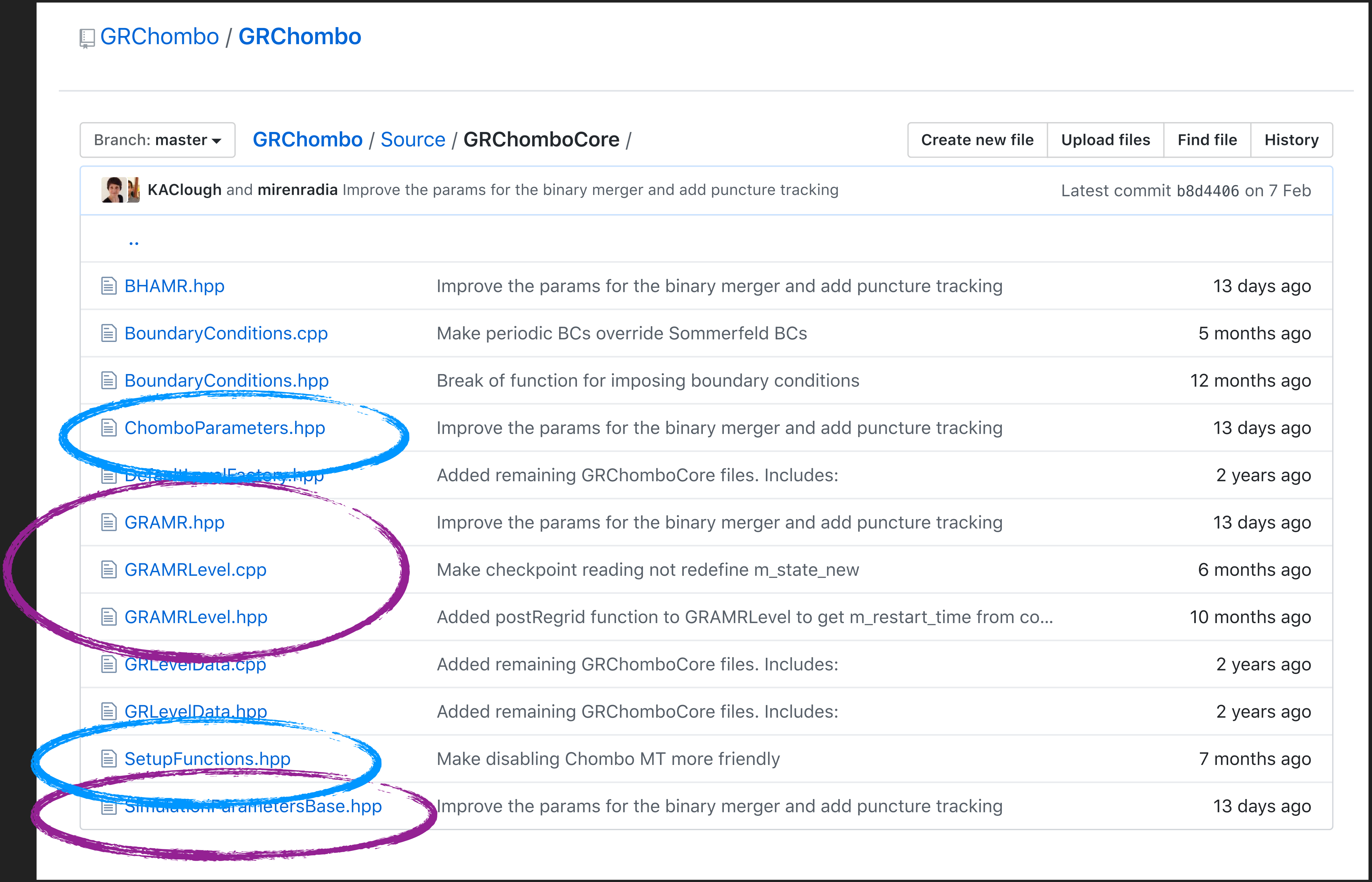
GRChombo / **GRChombo**

GRChombo / Source / GRChomboCore /

KAClough and **mirenradia** Improve the params for the binary merger and add puncture tracking — Latest commit `b8d4406` on 7 Feb

..		
BHAMR.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
BoundaryConditions.cpp	Make periodic BCs override Sommerfeld BCs	5 months ago
BoundaryConditions.hpp	Break of function for imposing boundary conditions	12 months ago
ChomboParameters.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
DefaultLevelFactory.hpp	Added remaining GRChomboCore files. Includes:	2 years ago
GRAMR.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
GRAMRLevel.cpp	Make checkpoint reading not redefine m_state_new	6 months ago
GRAMRLevel.hpp	Added postRegrid function to GRAMRLevel to get m_restart_time from co…	10 months ago
GRLevelData.cpp	Added remaining GRChomboCore files. Includes:	2 years ago
GRLevelData.hpp	Added remaining GRChomboCore files. Includes:	2 years ago
SetupFunctions.hpp	Make disabling Chombo MT more friendly	7 months ago
SimulationParametersBase.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago

WHERE ARE THE KEY GRCHOMBO FILES?



GRChombo / **GRChombo**

Branch: **master**

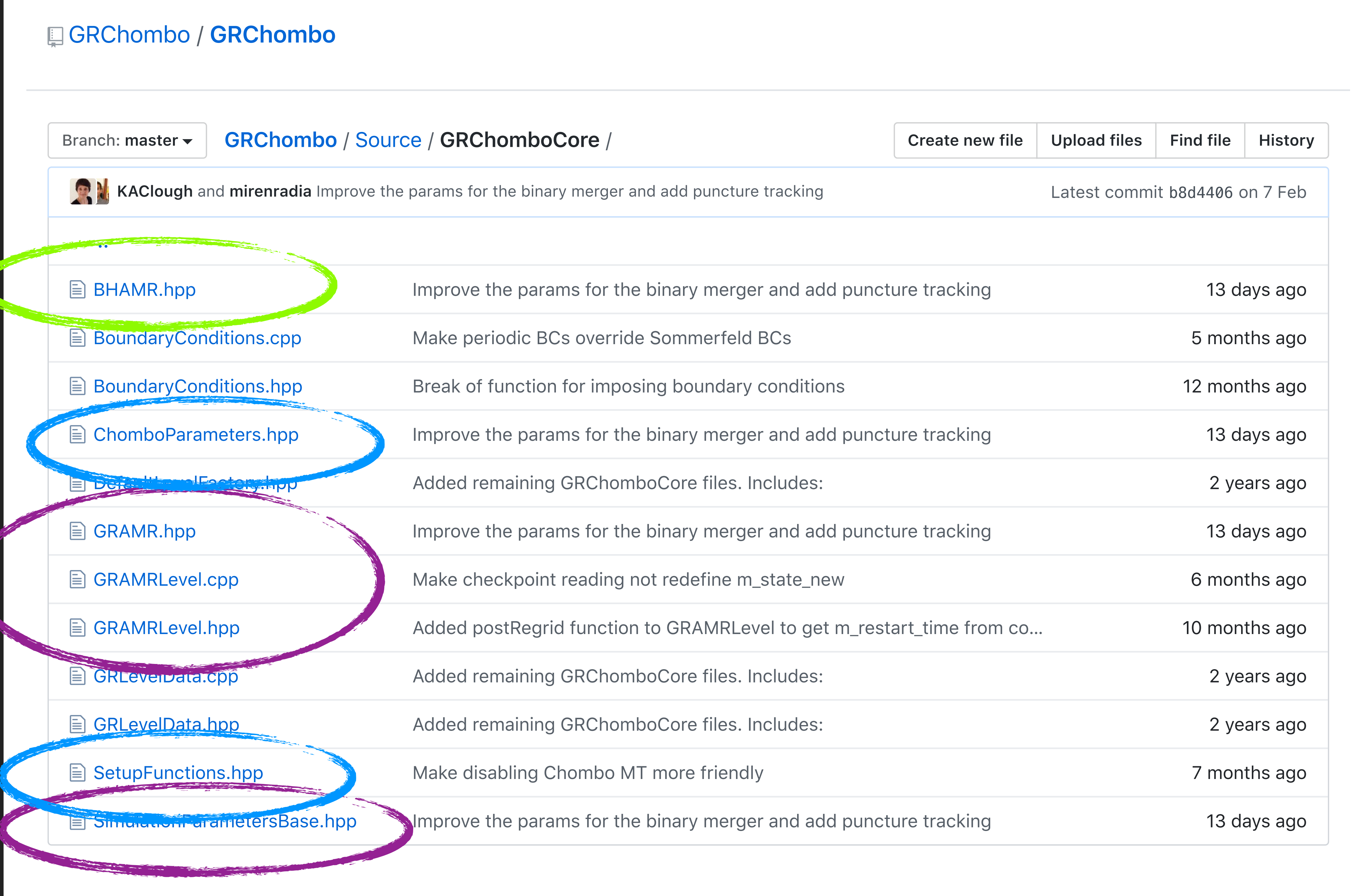
GRChombo / **Source** / **GRChomboCore** /

KAClough and **mirenradia** Improve the params for the binary merger and add puncture tracking — Latest commit `b8d4406` on 7 Feb

..

BHAMR.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
BoundaryConditions.cpp	Make periodic BCs override Sommerfeld BCs	5 months ago
BoundaryConditions.hpp	Break of function for imposing boundary conditions	12 months ago
ChomboParameters.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
DefaultLevelFactory.hpp	Added remaining GRChomboCore files. Includes:	2 years ago
GRAMR.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
GRAMRLevel.cpp	Make checkpoint reading not redefine m_state_new	6 months ago
GRAMRLevel.hpp	Added postRegrid function to GRAMRLevel to get m_restart_time from co…	10 months ago
GRLevelData.cpp	Added remaining GRChomboCore files. Includes:	2 years ago
GRLevelData.hpp	Added remaining GRChomboCore files. Includes:	2 years ago
SetupFunctions.hpp	Make disabling Chombo MT more friendly	7 months ago
SimulationParametersBase.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago

WHERE ARE THE KEY GRCHOMBO FILES?



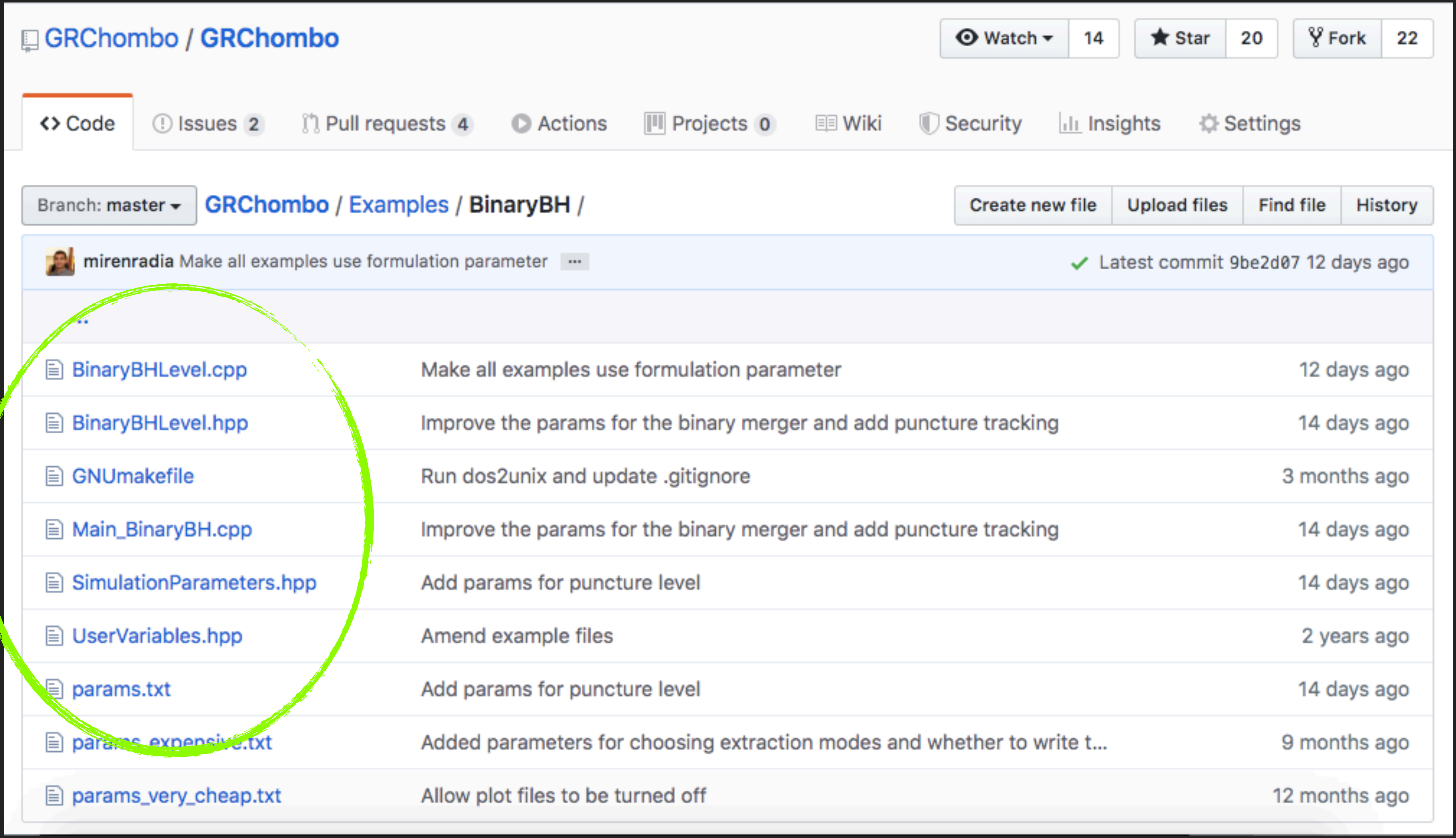
GRChombo / **GRChombo**

Branch: master ▾ | GRChombo / Source / GRChomboCore / | Create new file | Upload files | Find file | History

KAClough and **mirenradia** Improve the params for the binary merger and add puncture tracking | Latest commit `b8d4406` on 7 Feb

..		
BHAMR.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
BoundaryConditions.cpp	Make periodic BCs override Sommerfeld BCs	5 months ago
BoundaryConditions.hpp	Break of function for imposing boundary conditions	12 months ago
ChomboParameters.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
DefaultLevelFactory.hpp	Added remaining GRChomboCore files. Includes:	2 years ago
GRAMR.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago
GRAMRLevel.cpp	Make checkpoint reading not redefine m_state_new	6 months ago
GRAMRLevel.hpp	Added postRegrid function to GRAMRLevel to get m_restart_time from co…	10 months ago
GRLevelData.cpp	Added remaining GRChomboCore files. Includes:	2 years ago
GRLevelData.hpp	Added remaining GRChomboCore files. Includes:	2 years ago
SetupFunctions.hpp	Make disabling Chombo MT more friendly	7 months ago
SimulationParametersBase.hpp	Improve the params for the binary merger and add puncture tracking	13 days ago

WHERE ARE THE KEY BINARYBH FILES?



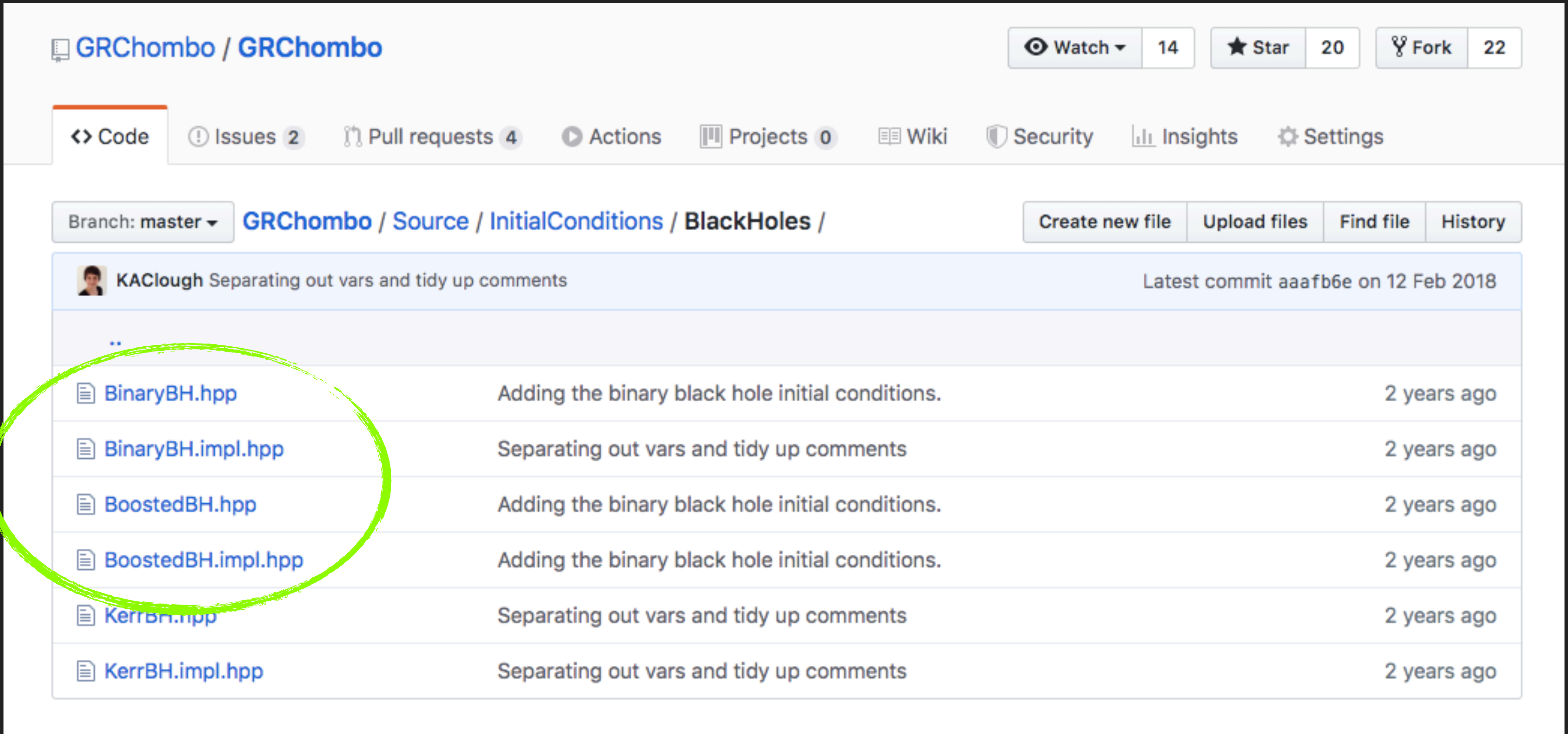
GRChombo / Examples / BinaryBH /

mirenradia Make all examples use formulation parameter … ✓ Latest commit 9be2d07 12 days ago

..

BinaryBHLevel.cpp	Make all examples use formulation parameter	12 days ago
BinaryBHLevel.hpp	Improve the params for the binary merger and add puncture tracking	14 days ago
GNUmakefile	Run dos2unix and update .gitignore	3 months ago
Main_BinaryBH.cpp	Improve the params for the binary merger and add puncture tracking	14 days ago
SimulationParameters.hpp	Add params for puncture level	14 days ago
UserVariables.hpp	Amend example files	2 years ago
params.txt	Add params for puncture level	14 days ago
params_expensive.txt	Added parameters for choosing extraction modes and whether to write t…	9 months ago
params_very_cheap.txt	Allow plot files to be turned off	12 months ago

WHERE ARE THE KEY BINARYBH FILES?



GRChombo / Source / InitialConditions / BlackHoles /

KAClough Separating out vars and tidy up comments — Latest commit aaafb6e on 12 Feb 2018

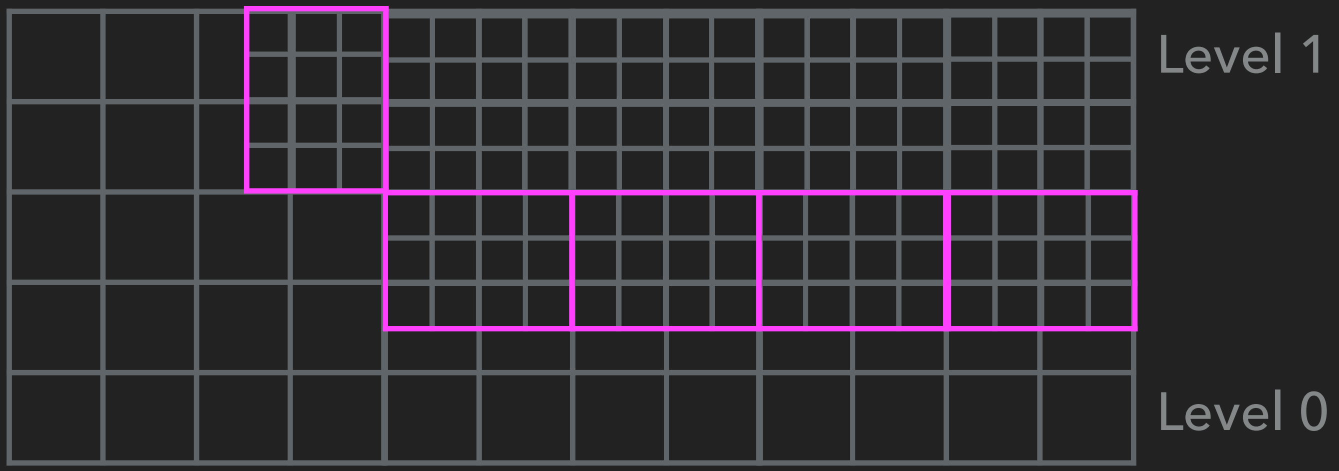
..

BinaryBH.hpp	Adding the binary black hole initial conditions.	2 years ago
BinaryBH.impl.hpp	Separating out vars and tidy up comments	2 years ago
BoostedBH.hpp	Adding the binary black hole initial conditions.	2 years ago
BoostedBH.impl.hpp	Adding the binary black hole initial conditions.	2 years ago
KerrBH.hpp	Separating out vars and tidy up comments	2 years ago
KerrBH.impl.hpp	Separating out vars and tidy up comments	2 years ago

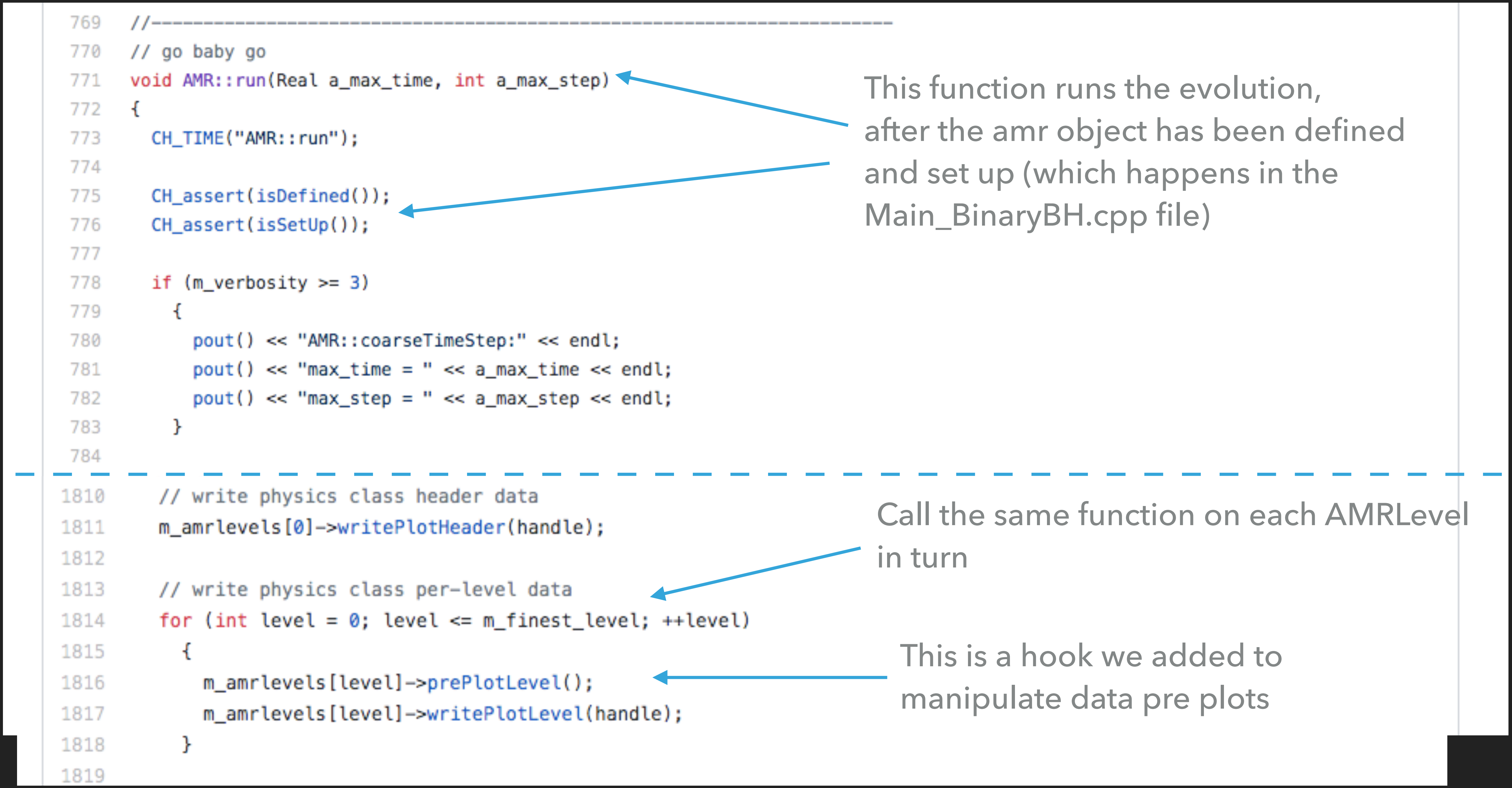
NB: These initial conditions are in “Source” as they are likely to be used for many examples ***without modification***. If you are using something very problem specific, you may want to put it in the Example folder.

STRUCTURE OF AMR

- Does setup (for restart or using initial data) and runs evolution
- Knows about all of the levels, each function generally cycles through each level from coarse to fine
- Contains hooks for physics class actions (occurring in GRAMRLevel / BinaryBHLevel)



E.G. AMR::RUN() DOES THE EVOLUTION



```
//-----------------------------------------------------------------------
// go baby go
void AMR::run(Real a_max_time, int a_max_step)
{
  CH_TIME("AMR::run");

  CH_assert(isDefined());
  CH_assert(isSetUp());

  if (m_verbosity >= 3)
    {
      pout() << "AMR::coarseTimeStep:" << endl;
      pout() << "max_time = " << a_max_time << endl;
      pout() << "max_step = " << a_max_step << endl;
    }

```

This function runs the evolution, after the amr object has been defined and set up (which happens in the Main_BinaryBH.cpp file)

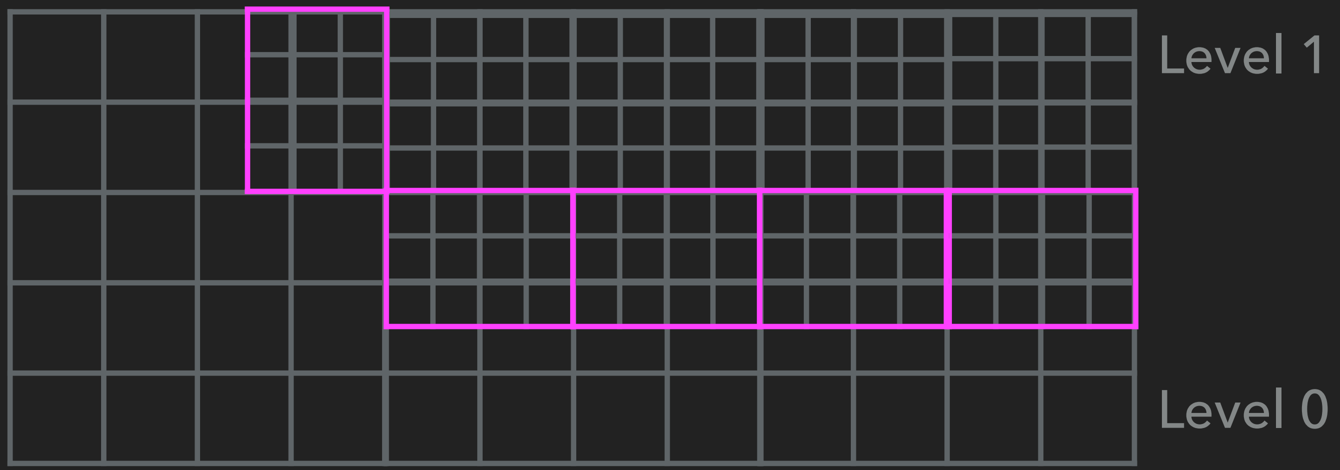
```
  // write physics class header data
  m_amrlevels[0]->writePlotHeader(handle);

  // write physics class per-level data
  for (int level = 0; level <= m_finest_level; ++level)
    {
      m_amrlevels[level]->prePlotLevel();
      m_amrlevels[level]->writePlotLevel(handle);
    }

```

Call the same function on each AMRLevel in turn

This is a hook we added to manipulate data pre plots

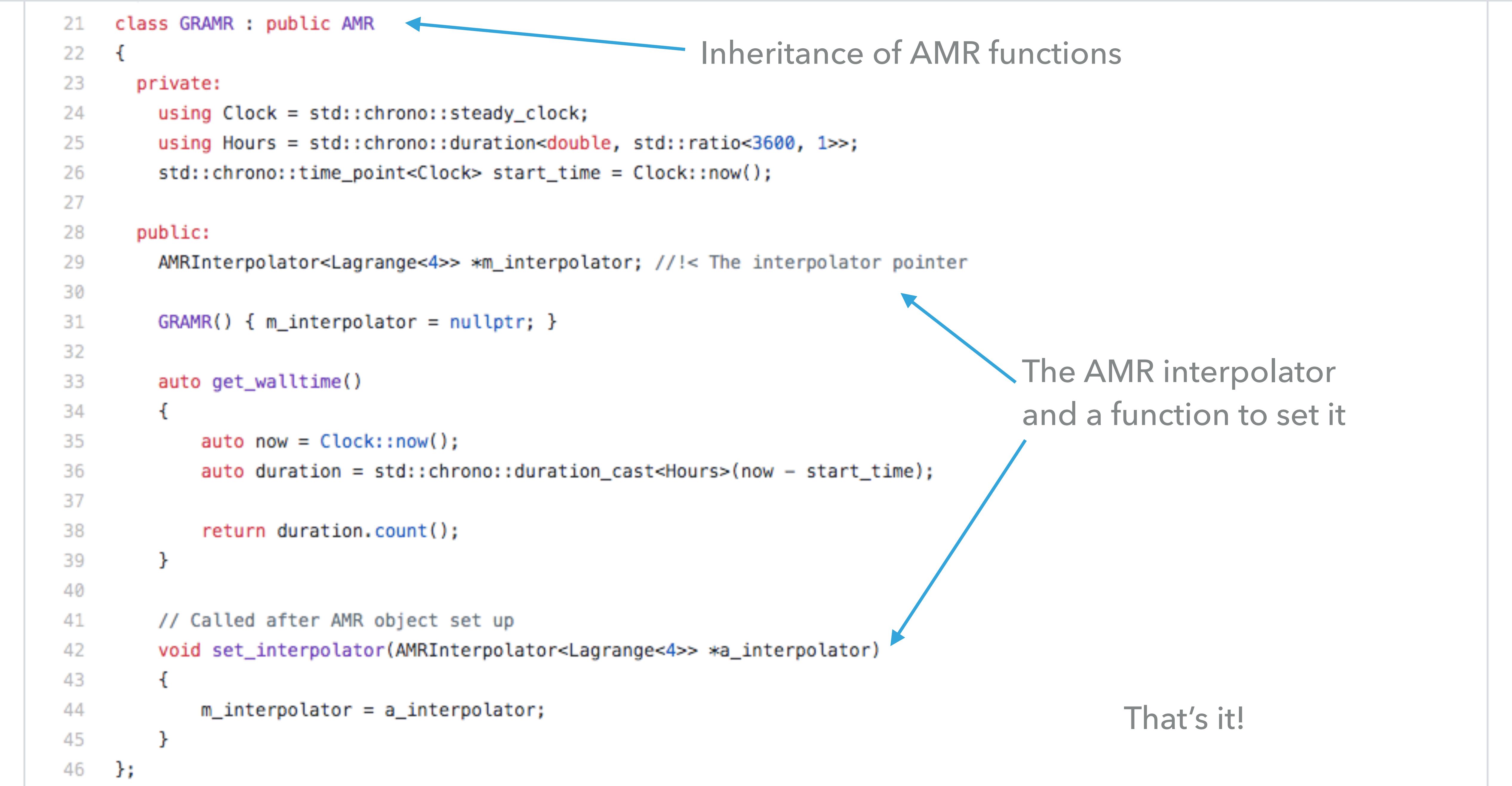
STRUCTURE OF GRAMR

- Inherits all functionality from AMR
- Adds in our GR specific tools, e.g. AMRInterpolator*
- Only contains things that happen globally across the grid, so actually not that much. Most actions are local to a level.



(*OK, so this is not GR specific, but it did not exist in Chombo so we built it, and now it lives in GRChombo because we don't want to hack the Chombo code too much.)

GRAMR CLASS



```
class GRAMR : public AMR
{
 private:
  using Clock = std::chrono::steady_clock;
  using Hours = std::chrono::duration<double, std::ratio<3600, 1>>;
  std::chrono::time_point<Clock> start_time = Clock::now();

 public:
  AMRInterpolator<Lagrange<4>> *m_interpolator; //!< The interpolator pointer

  GRAMR() { m_interpolator = nullptr; }

  auto get_walltime()
  {
    auto now = Clock::now();
    auto duration = std::chrono::duration_cast<Hours>(now - start_time);

    return duration.count();
  }

  // Called after AMR object set up
  void set_interpolator(AMRInterpolator<Lagrange<4>> *a_interpolator)
  {
    m_interpolator = a_interpolator;
  }
};
```

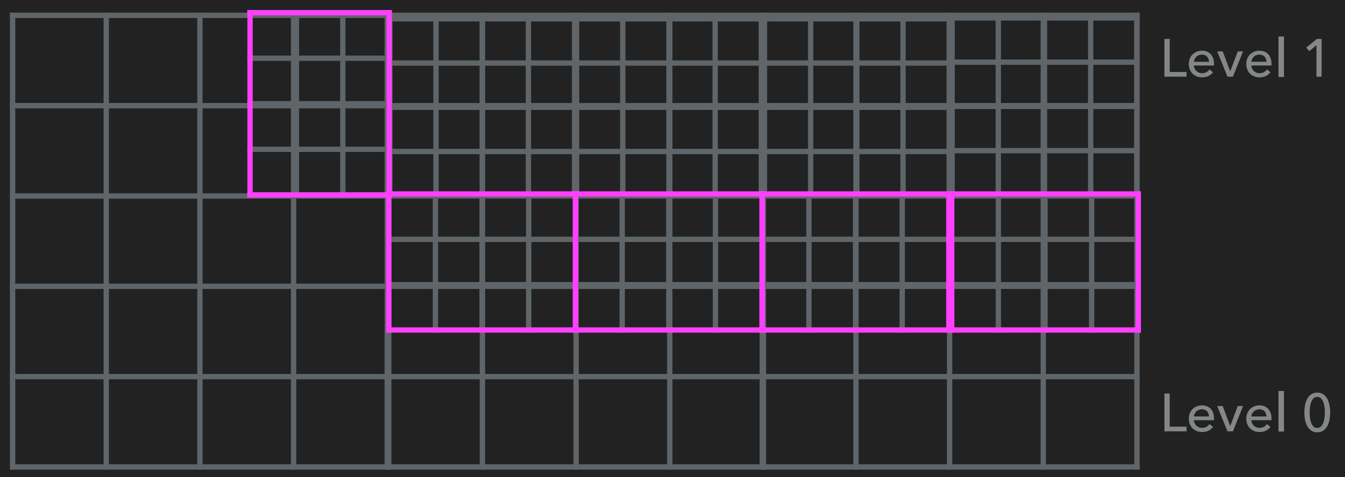
Inheritance of AMR functions

The AMR interpolator
and a function to set it

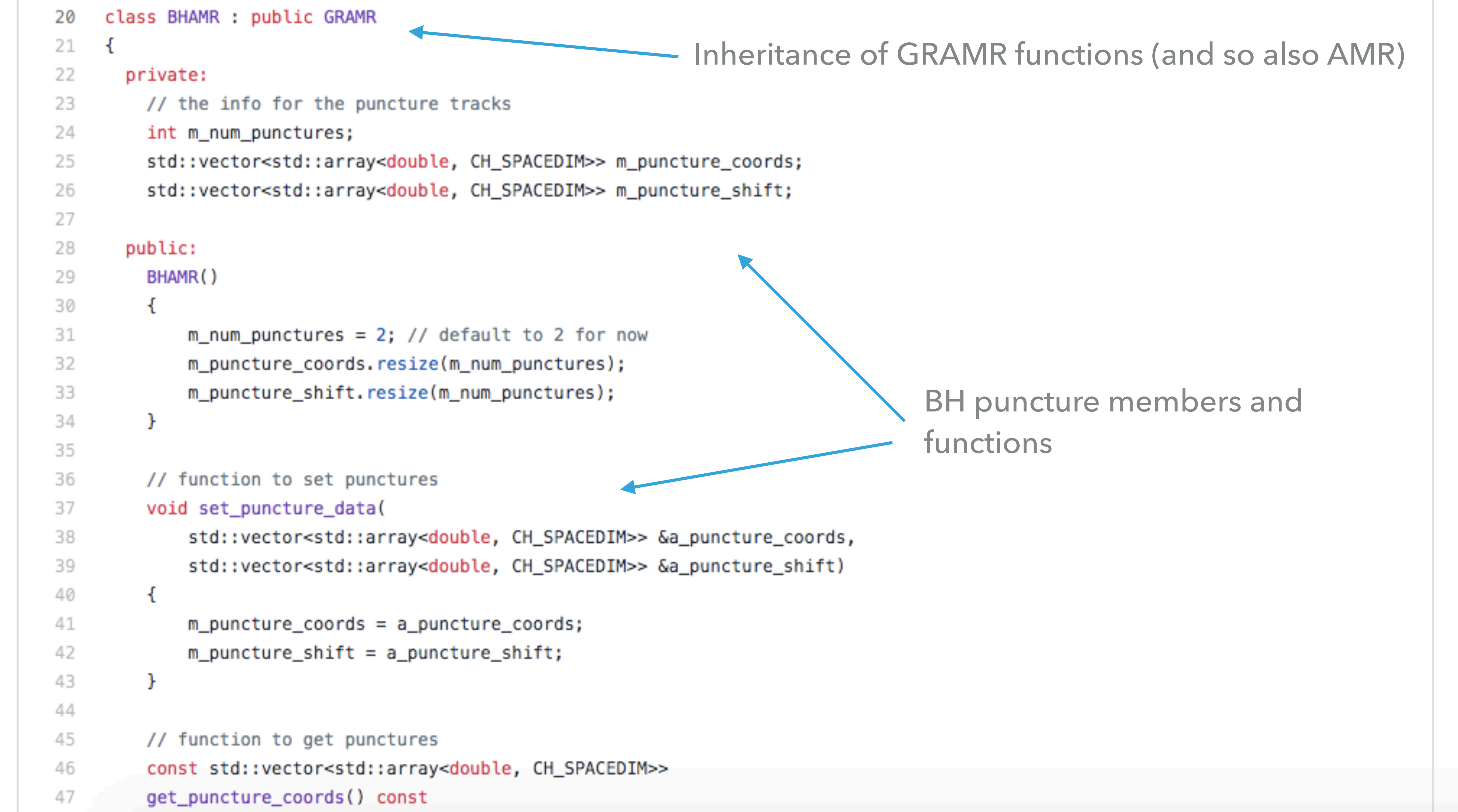
That's it!

STRUCTURE OF BHAMR

- Inherits all functionality from GRAMR
- Adds in BBH specific tools, e.g. Puncture Tracking
- Again not that long!



STRUCTURE OF BHAMR

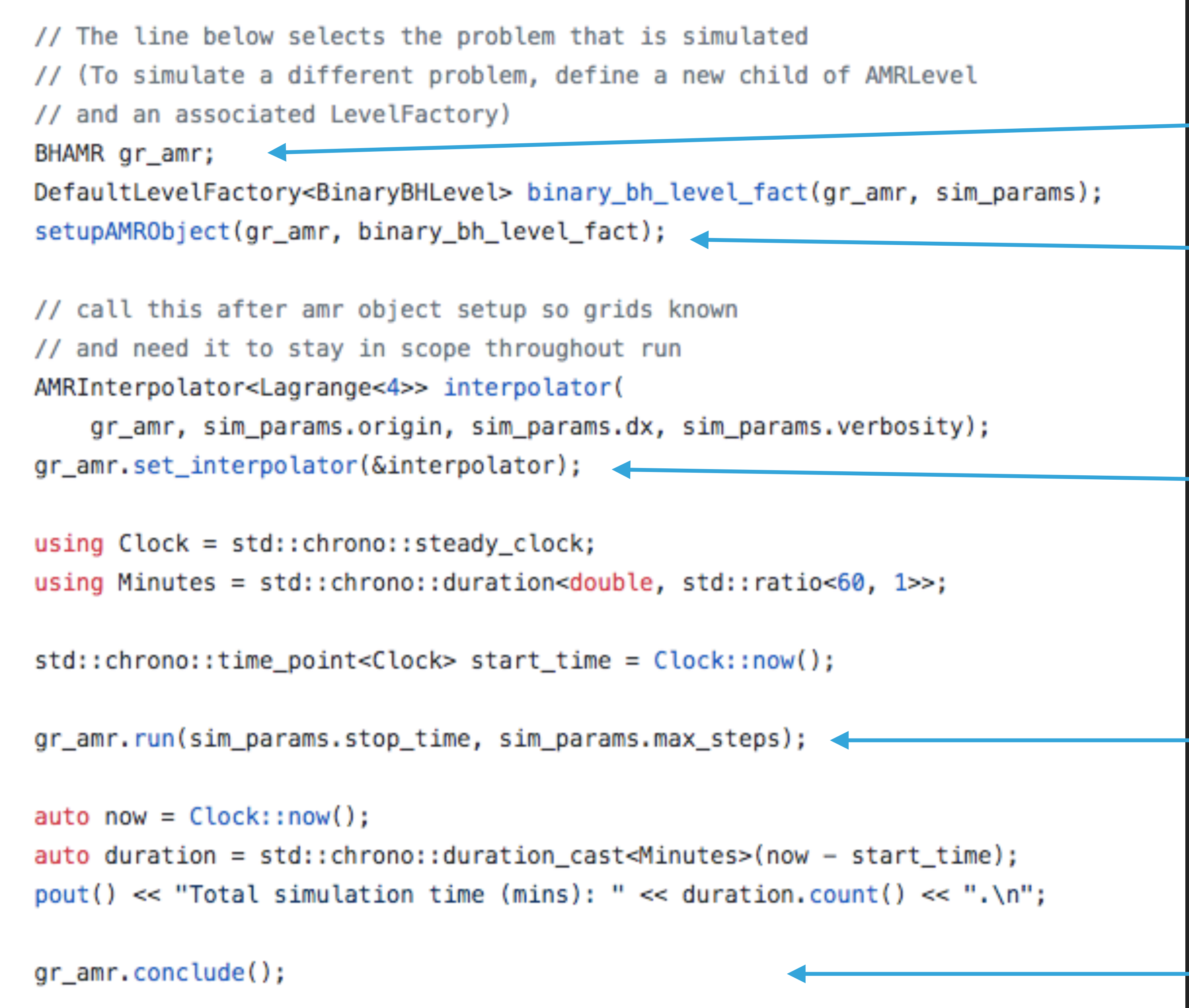


```cpp
class BHAMR : public GRAMR
{
  private:
    // the info for the puncture tracks
    int m_num_punctures;
    std::vector<std::array<double, CH_SPACEDIM>> m_puncture_coords;
    std::vector<std::array<double, CH_SPACEDIM>> m_puncture_shift;

  public:
    BHAMR()
    {
        m_num_punctures = 2; // default to 2 for now
        m_puncture_coords.resize(m_num_punctures);
        m_puncture_shift.resize(m_num_punctures);
    }

    // function to set punctures
    void set_puncture_data(
        std::vector<std::array<double, CH_SPACEDIM>> &a_puncture_coords,
        std::vector<std::array<double, CH_SPACEDIM>> &a_puncture_shift)
    {
        m_puncture_coords = a_puncture_coords;
        m_puncture_shift = a_puncture_shift;
    }

    // function to get punctures
    const std::vector<std::array<double, CH_SPACEDIM>>
    get_puncture_coords() const
```

Inheritance of GRAMR functions (and so also AMR)

BH puncture members and functions

ALL THIS COMES TOGETHER IN MAIN_BINARYBH.CPP



```
// The line below selects the problem that is simulated
// (To simulate a different problem, define a new child of AMRLevel
// and an associated LevelFactory)
BHAMR gr_amr;
DefaultLevelFactory<BinaryBHLevel> binary_bh_level_fact(gr_amr, sim_params);
setupAMRObject(gr_amr, binary_bh_level_fact);

// call this after amr object setup so grids known
// and need it to stay in scope throughout run
AMRInterpolator<Lagrange<4>> interpolator(
    gr_amr, sim_params.origin, sim_params.dx, sim_params.verbosity);
gr_amr.set_interpolator(&interpolator);

using Clock = std::chrono::steady_clock;
using Minutes = std::chrono::duration<double, std::ratio<60, 1>>;

std::chrono::time_point<Clock> start_time = Clock::now();

gr_amr.run(sim_params.stop_time, sim_params.max_steps);

auto now = Clock::now();
auto duration = std::chrono::duration_cast<Minutes>(now - start_time);
pout() << "Total simulation time (mins): " << duration.count() << ".\n";

gr_amr.conclude();
```

Make a BHAMR object

Setup using AMR functions

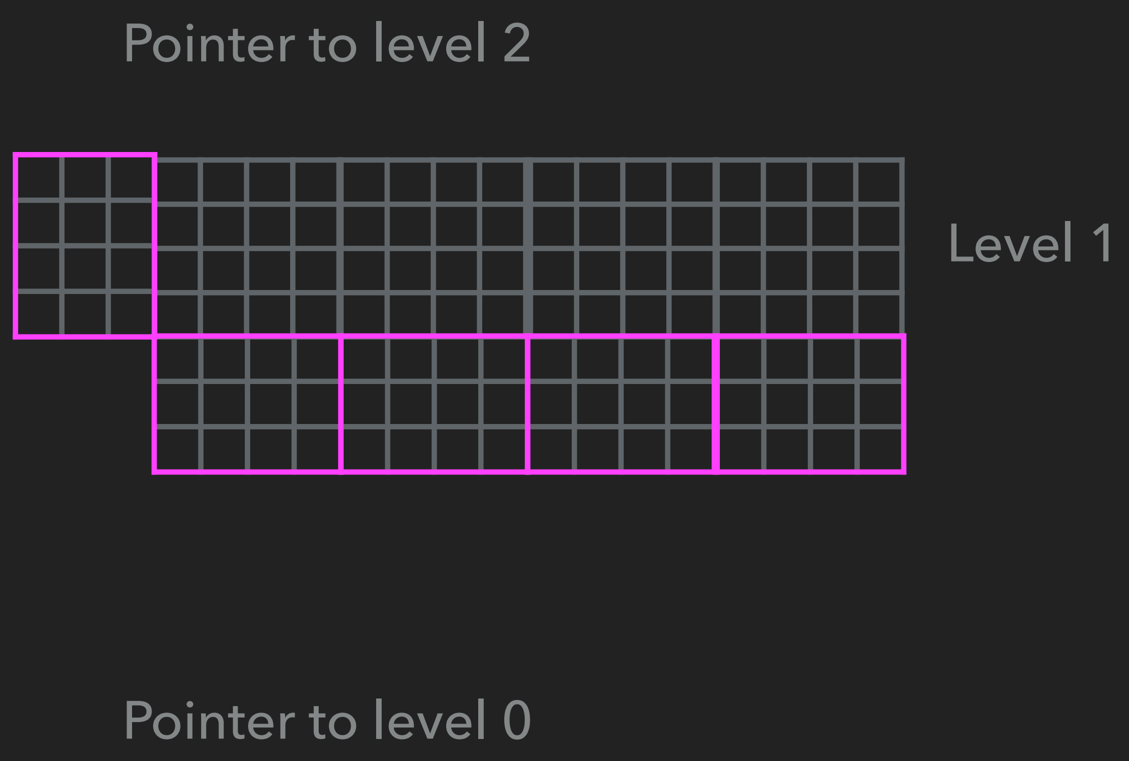
Setup interpolator which lives in GRAMR

AMR run function

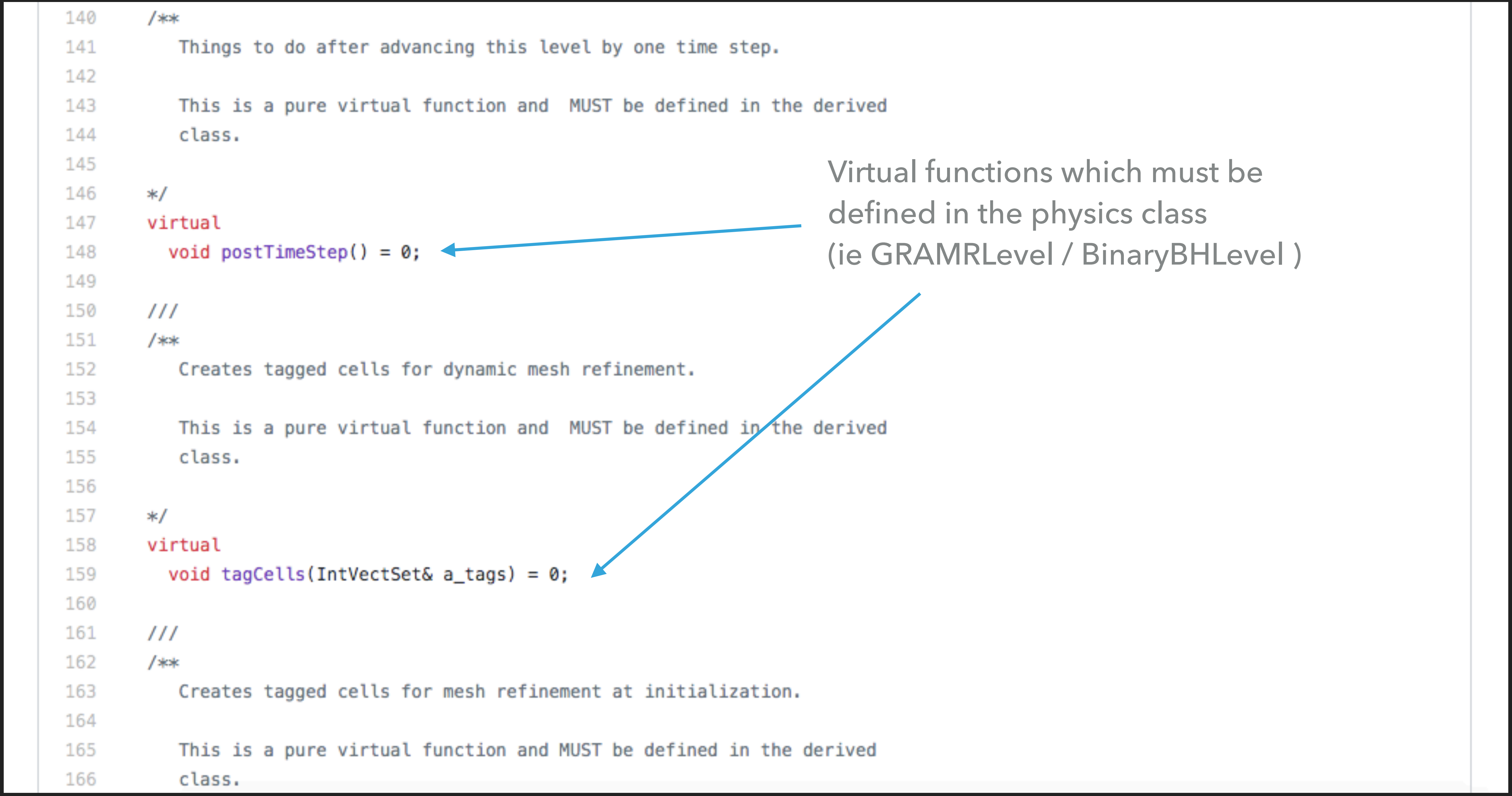
AMR conclude function

STRUCTURE OF AMRLEVEL

- Knows about its own level data, and has a pointer to the coarser and finer levels above and below it
- Abstract base class to be overwritten by a "physics class" i.e. GRAMRLevel / BinaryBHLevel



STRUCTURE OF AMRLEVEL



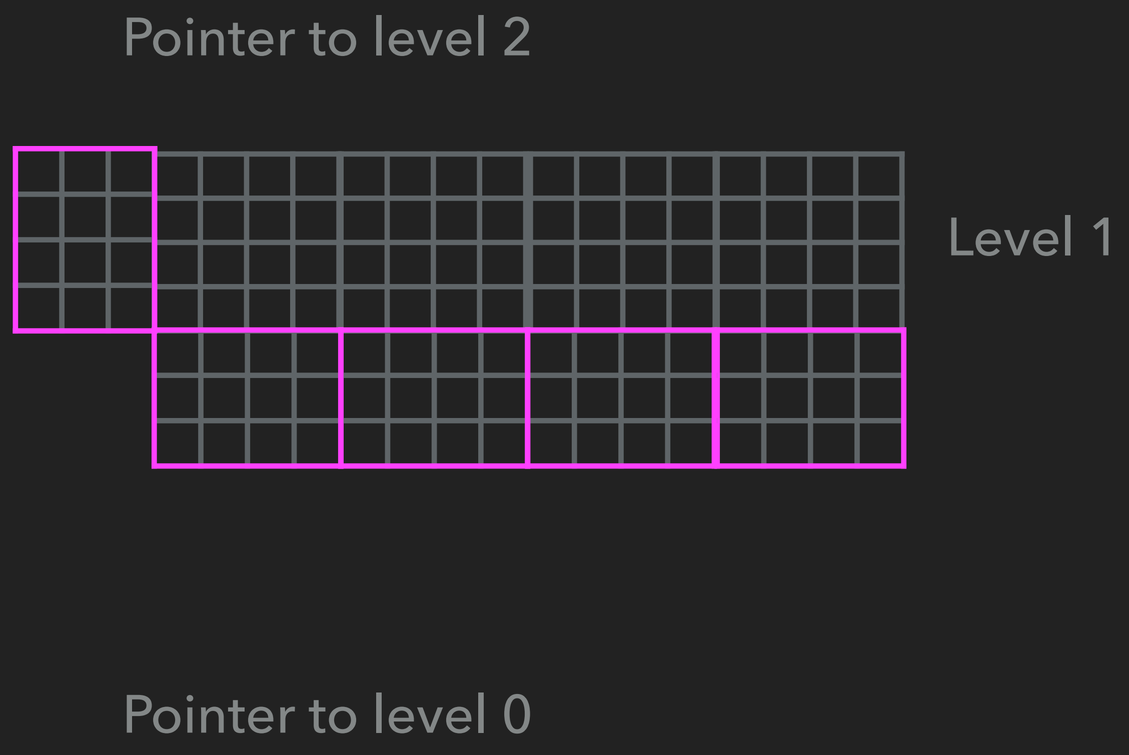
```
/**
   Things to do after advancing this level by one time step.

   This is a pure virtual function and  MUST be defined in the derived
   class.

*/
virtual
  void postTimeStep() = 0;

///
/**
   Creates tagged cells for dynamic mesh refinement.

   This is a pure virtual function and  MUST be defined in the derived
   class.

*/
virtual
  void tagCells(IntVectSet& a_tags) = 0;

///
/**
   Creates tagged cells for mesh refinement at initialization.

   This is a pure virtual function and MUST be defined in the derived
   class.
```

Virtual functions which must be defined in the physics class (ie GRAMRLevel / BinaryBHLevel)

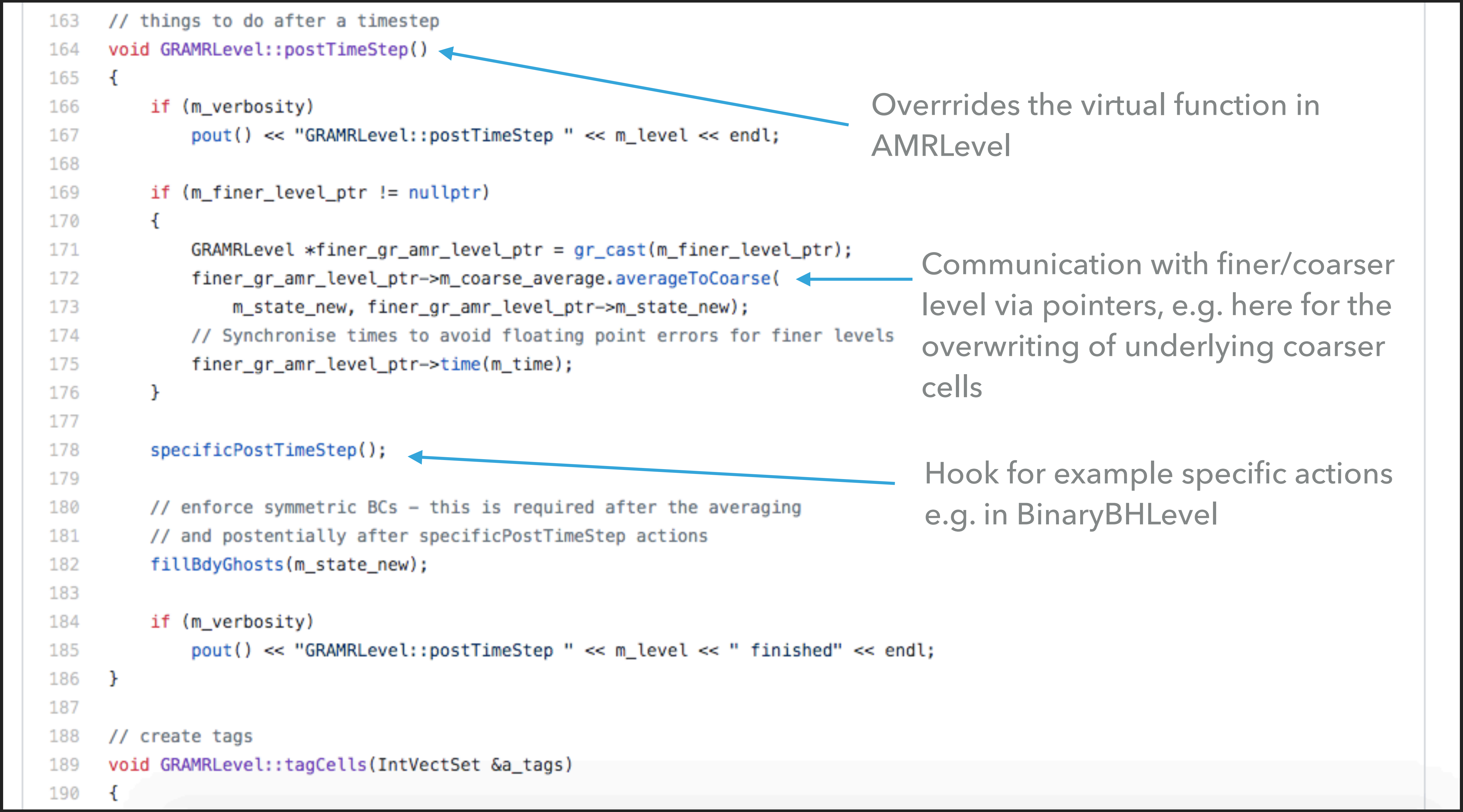
STRUCTURE OF GRAMRLEVEL

- Inherits from AMRLevel and overwrites virtual functions where these are common to most GR simulations

- Contains hooks for example specific actions (occurring in BinaryBHLevel, prefixed by “specific”)



STRUCTURE OF GRAMRLEVEL



```cpp
// things to do after a timestep
void GRAMRLevel::postTimeStep()
{
    if (m_verbosity)
        pout() << "GRAMRLevel::postTimeStep " << m_level << endl;

    if (m_finer_level_ptr != nullptr)
    {
        GRAMRLevel *finer_gr_amr_level_ptr = gr_cast(m_finer_level_ptr);
        finer_gr_amr_level_ptr->m_coarse_average.averageToCoarse(
            m_state_new, finer_gr_amr_level_ptr->m_state_new);
        // Synchronise times to avoid floating point errors for finer levels
        finer_gr_amr_level_ptr->time(m_time);
    }

    specificPostTimeStep();

    // enforce symmetric BCs - this is required after the averaging
    // and postentially after specificPostTimeStep actions
    fillBdyGhosts(m_state_new);

    if (m_verbosity)
        pout() << "GRAMRLevel::postTimeStep " << m_level << " finished" << endl;
}

// create tags
void GRAMRLevel::tagCells(IntVectSet &a_tags)
{
```

Overrrides the virtual function in AMRLevel

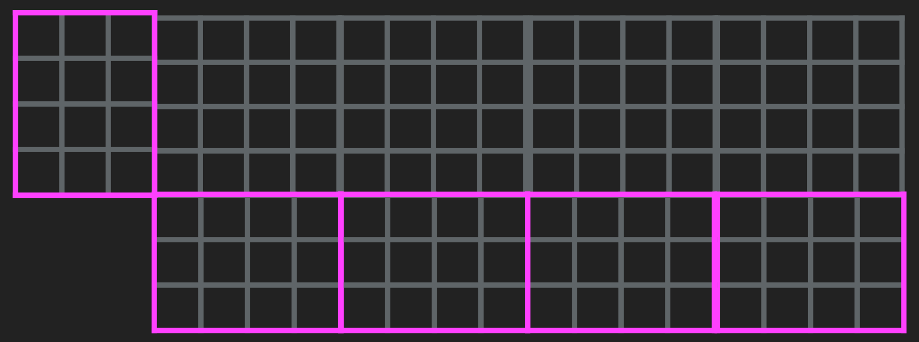
Communication with finer/coarser level via pointers, e.g. here for the overwriting of underlying coarser cells

Hook for example specific actions e.g. in BinaryBHLevel

STRUCTURE OF BINARYBHLEVEL

- Inherits all functionality from GRAMRLevel, overwrites virtual functions where these are specific to BinaryBH example
- Adds in required BBH specific functions via the hooks like specificPostTimeStep()

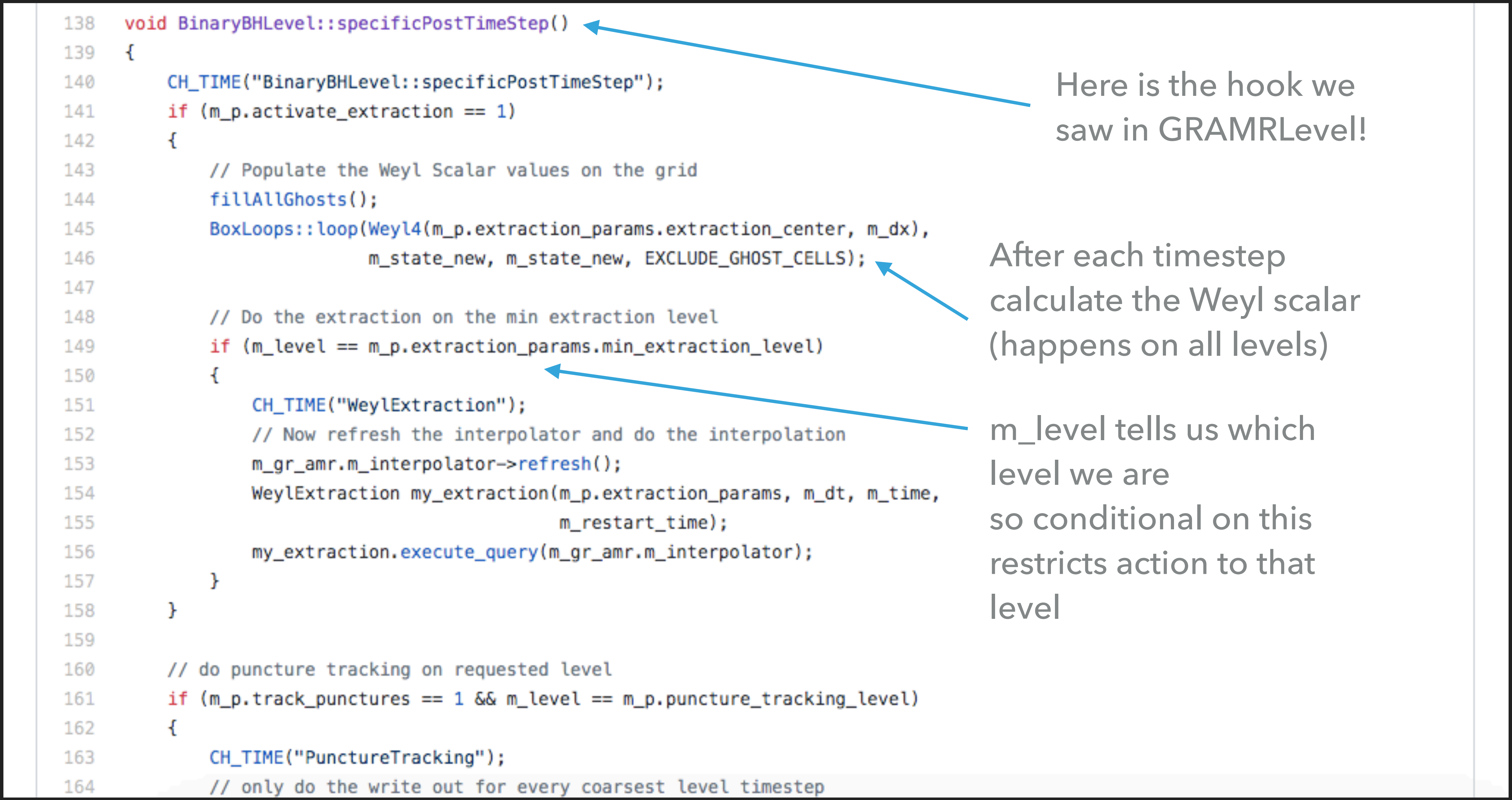
Pointer to level 2



Level 1

Pointer to level 0

STRUCTURE OF BINARYBHLEVEL



```
void BinaryBHLevel::specificPostTimeStep()
{
    CH_TIME("BinaryBHLevel::specificPostTimeStep");
    if (m_p.activate_extraction == 1)
    {
        // Populate the Weyl Scalar values on the grid
        fillAllGhosts();
        BoxLoops::loop(Weyl4(m_p.extraction_params.extraction_center, m_dx),
                       m_state_new, m_state_new, EXCLUDE_GHOST_CELLS);

        // Do the extraction on the min extraction level
        if (m_level == m_p.extraction_params.min_extraction_level)
        {
            CH_TIME("WeylExtraction");
            // Now refresh the interpolator and do the interpolation
            m_gr_amr.m_interpolator->refresh();
            WeylExtraction my_extraction(m_p.extraction_params, m_dt, m_time,
                                         m_restart_time);
            my_extraction.execute_query(m_gr_amr.m_interpolator);
        }
    }

    // do puncture tracking on requested level
    if (m_p.track_punctures == 1 && m_level == m_p.puncture_tracking_level)
    {
        CH_TIME("PunctureTracking");
        // only do the write out for every coarsest level timestep
```

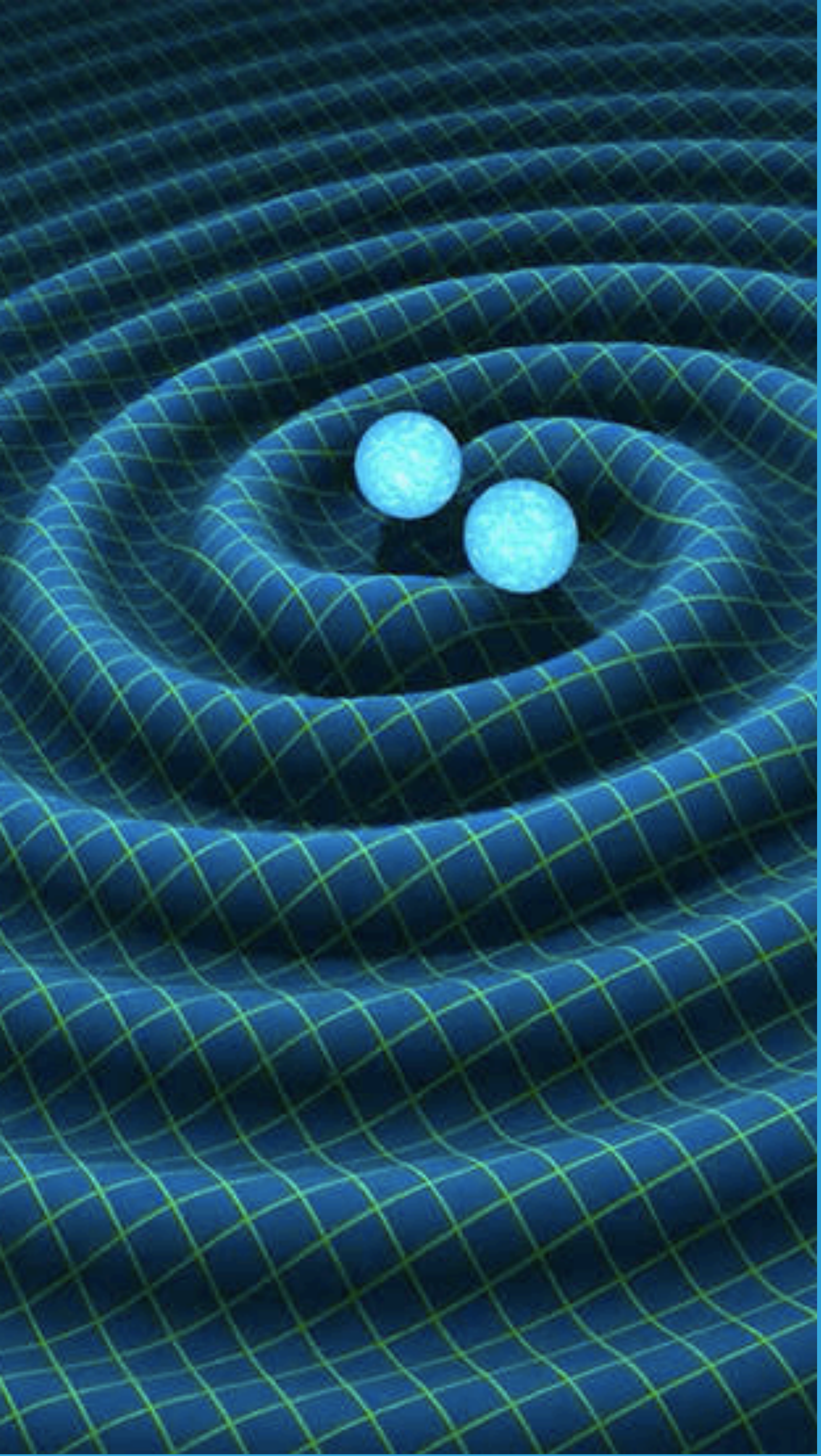
Here is the hook we saw in GRAMRLevel!

After each timestep calculate the Weyl scalar (happens on all levels)

m_level tells us which level we are so conditional on this restricts action to that level

KEY FUNCTIONS THAT WE SPECIFY IN BHBINARYLEVEL

function	required / optional /advised	Comment
initialdata()	required	define metric on initial grid
specificAdvance()	required	happens in RK4 substeps
postRestart()	optional	done after checkpoint restart
preCheckpointLevel()	optional	before output checkpoint
prePlotLevel()	optional	before output plot file
specificWritePlotHeader()	required for plot files	specify plot file variables
specificEvalRHS()	required	happens in RK4 substeps
specificUpdateODE()	advised	happens in RK4 substeps
computeTaggingCriterion()	required for AMR	criterion for refinement
specificPostTimestep()	optional	after level completes dt update



QUESTIONS?