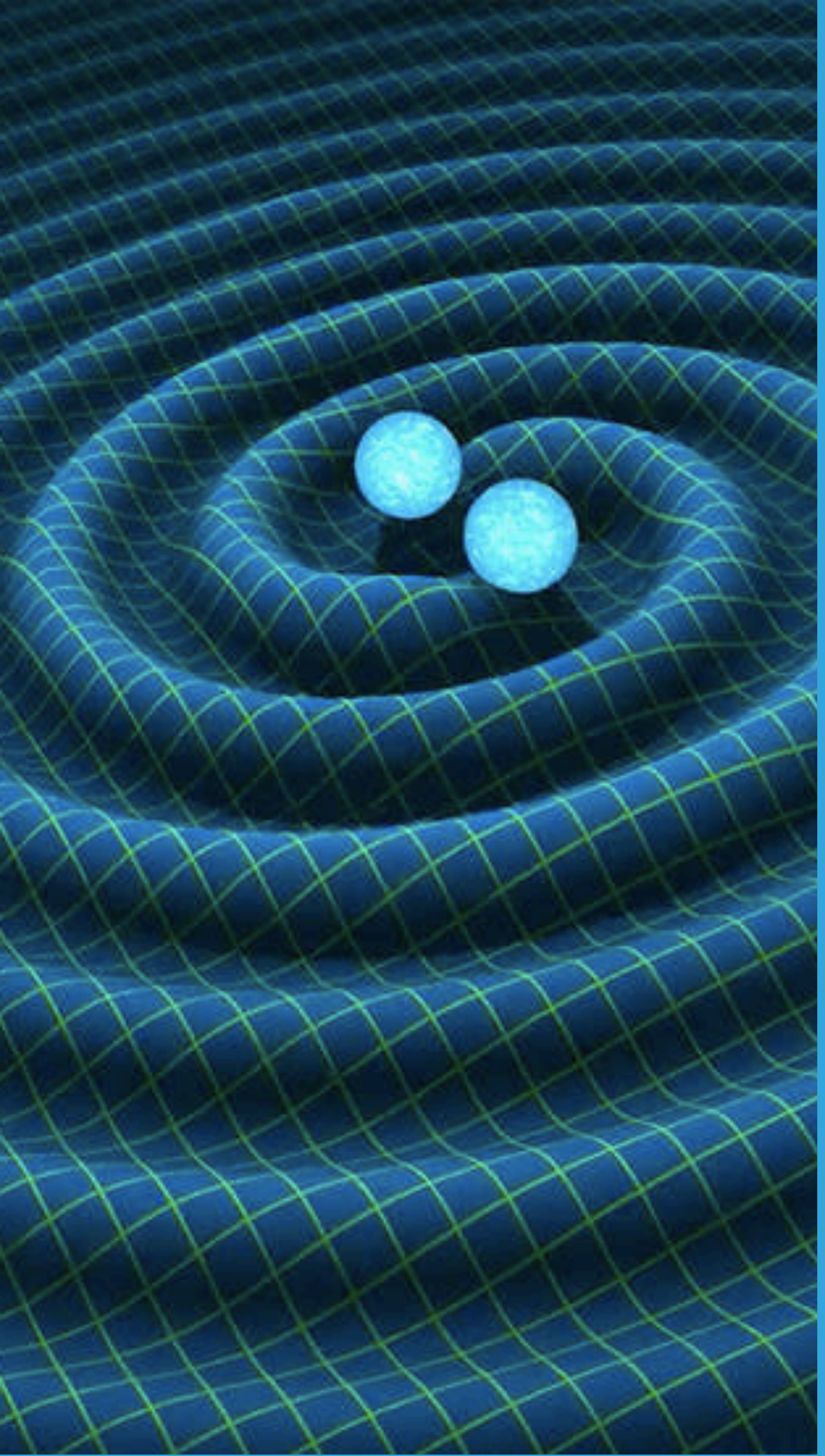
KATY CLOUGH

INTRO TO GRCHOMBO: THE BIG PICTURE

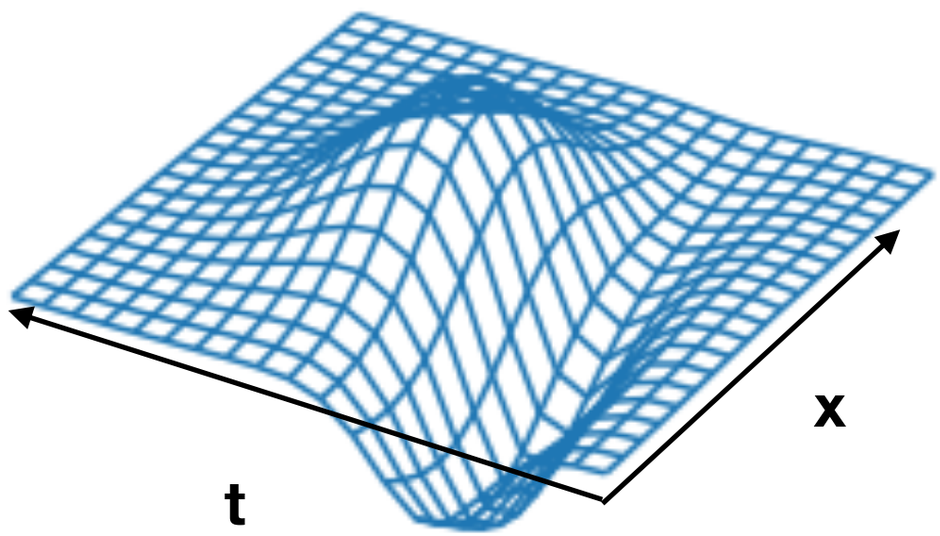


NUMERICAL RELATIVITY: BIG PICTURE

GR IN 2 MINUTES

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

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



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

“The spacetime metric” $g_{ab}(t,\vec{x})$

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"

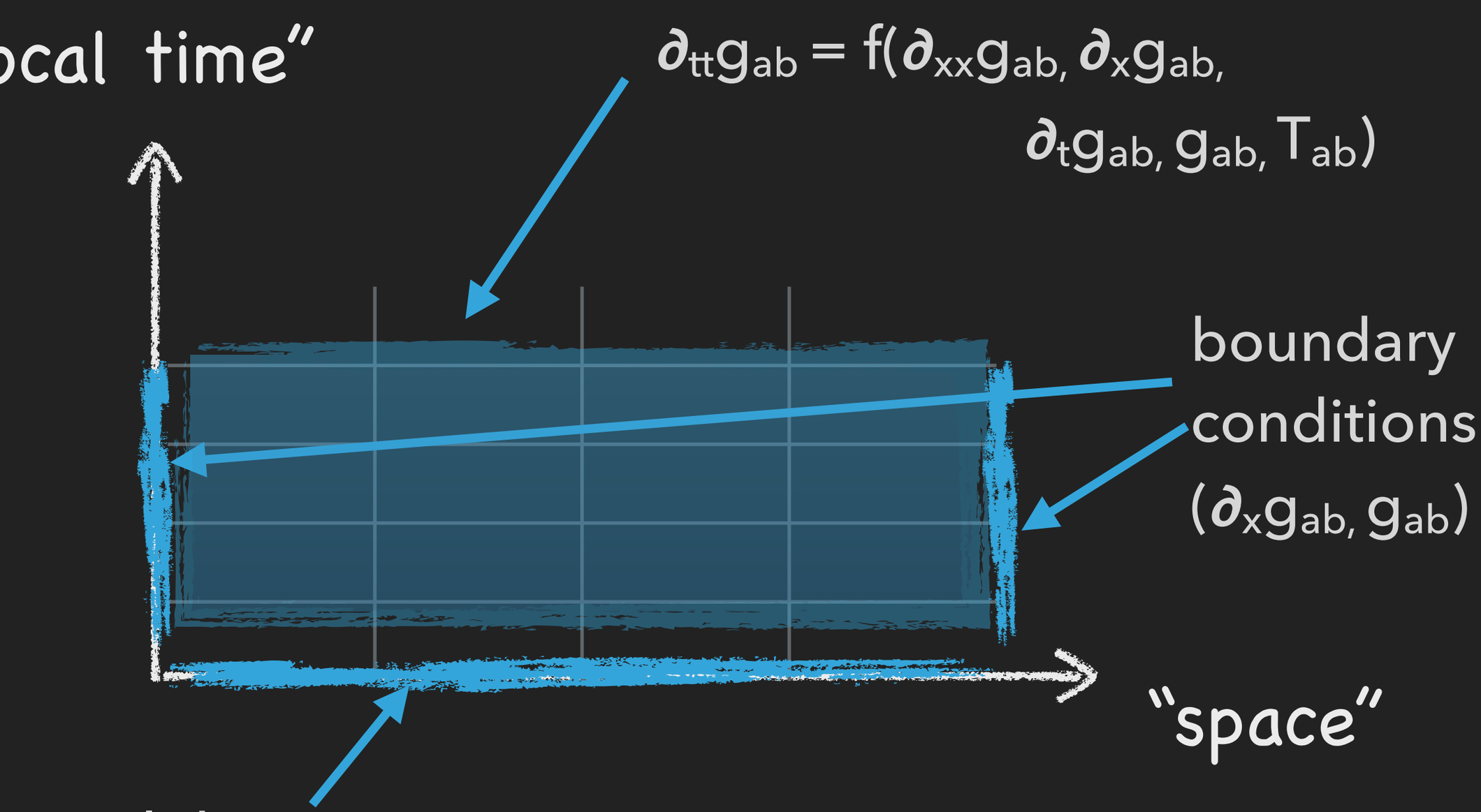
"Energy-Momentum"

Can rearrange into form (using ADM decomposition):

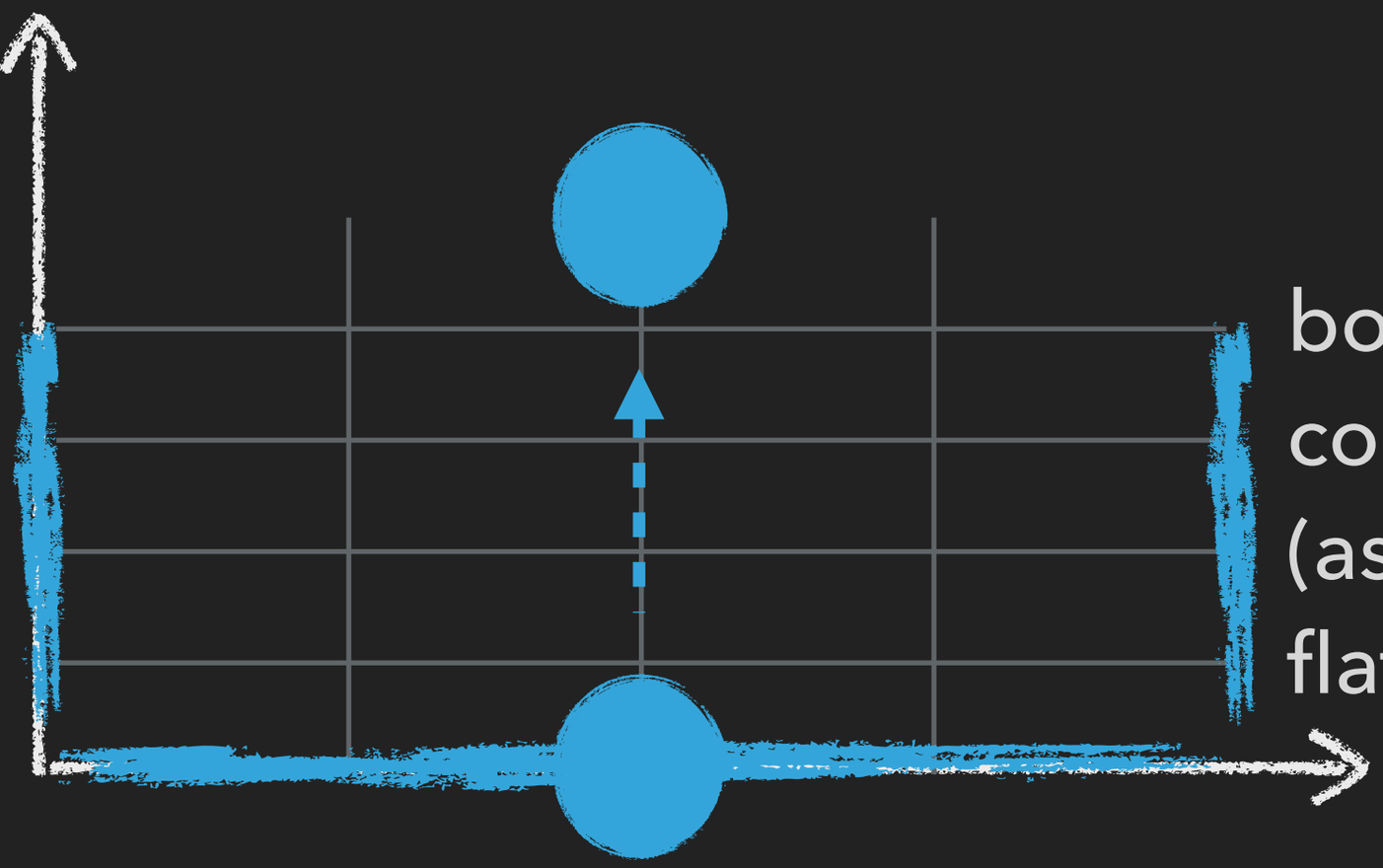
$$\partial_t(\partial_t g_{ab}) = f(\partial_{xx} g_{ab}, \partial_x g_{ab}, \partial_t g_{ab}, g_{ab}, T_{ab})$$

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

NR IN 2 MINUTES



NR IN 2 MINUTES

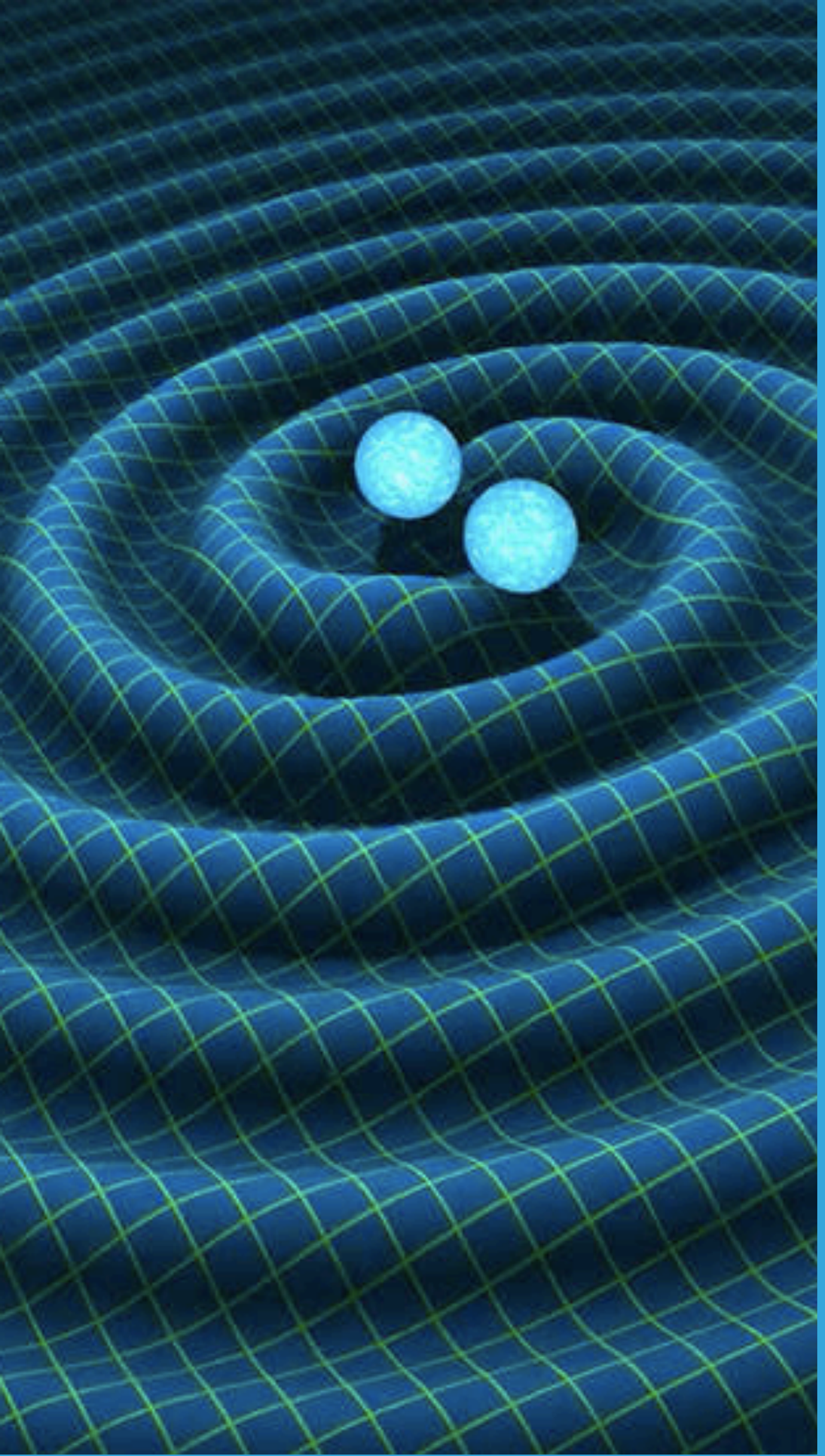


Fill using Einstein equation
$\partial_{tt} g_{ab} = 0$ (boring!) *

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)

GRCHOMBO IS A BIG CODE

GRChombo / GRChombo Public

Unpin | Watch 17 | Fork 31 | Star 40

Code | Issues 33 | Pull requests 7 | Discussions | Actions | Projects | Wiki | Security | Insights | Settings

main | 46 branches | 1 tag | Go to file | Add file | Code

KAClough Merge pull request #211 from GRChombo/bugfix/210 ... 231753c 6 days ago 355 commits

.github/workflows	Add github action to build [GR]Chombo with clang	4 months ago
Doxygen	Add GitHub action to update Doxygen documentation (#195)	6 months ago
Examples	Add fancier FOR macro replacing FOR1,FOR2,...	10 months ago
InstallNotes	Add github action to build [GR]Chombo with clang	4 months ago
Source	Fix absolute restart_file paths bug	7 days ago
Tests	Alternative fix for segfault in SphericalExtractionTest	4 months ago
.clang-format	Added clang-format configuration file.	4 years ago
.gitignore	Ignoring files from more then one simultaenous vim edits	2 years ago
.pre-commit-config.yaml	Add pre-commit configuration file (#198)	5 months ago
GNUmakefile	Decrease make verbosity	2 years ago
LICENSE	Initial commit	4 years ago
README.md	Update README.md	3 months ago
run_clang_format	Added clang-format configuration file.	4 years ago

About

An AMR based open-source code for numerical relativity simulations.

Readme
BSD-3-Clause License
40 stars
17 watching
31 forks

Releases 1

GRChombo: An adaptable nu... Latest
on 10 Dec 2021

Packages

No packages published
Publish your first package

Contributors 14

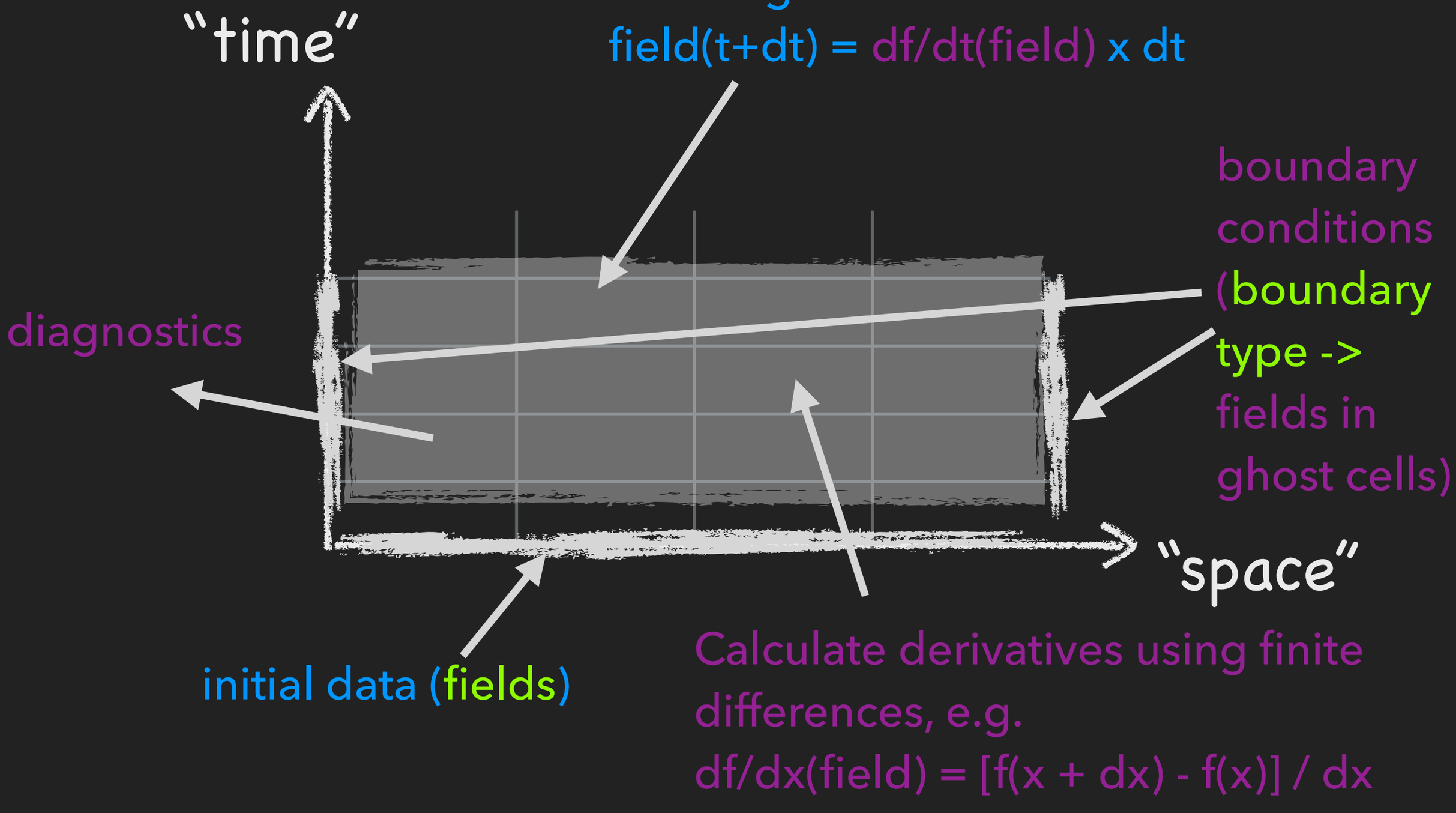
THREE LEVELS : CHOMBO / GRCHOMBO / BINARYBH

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

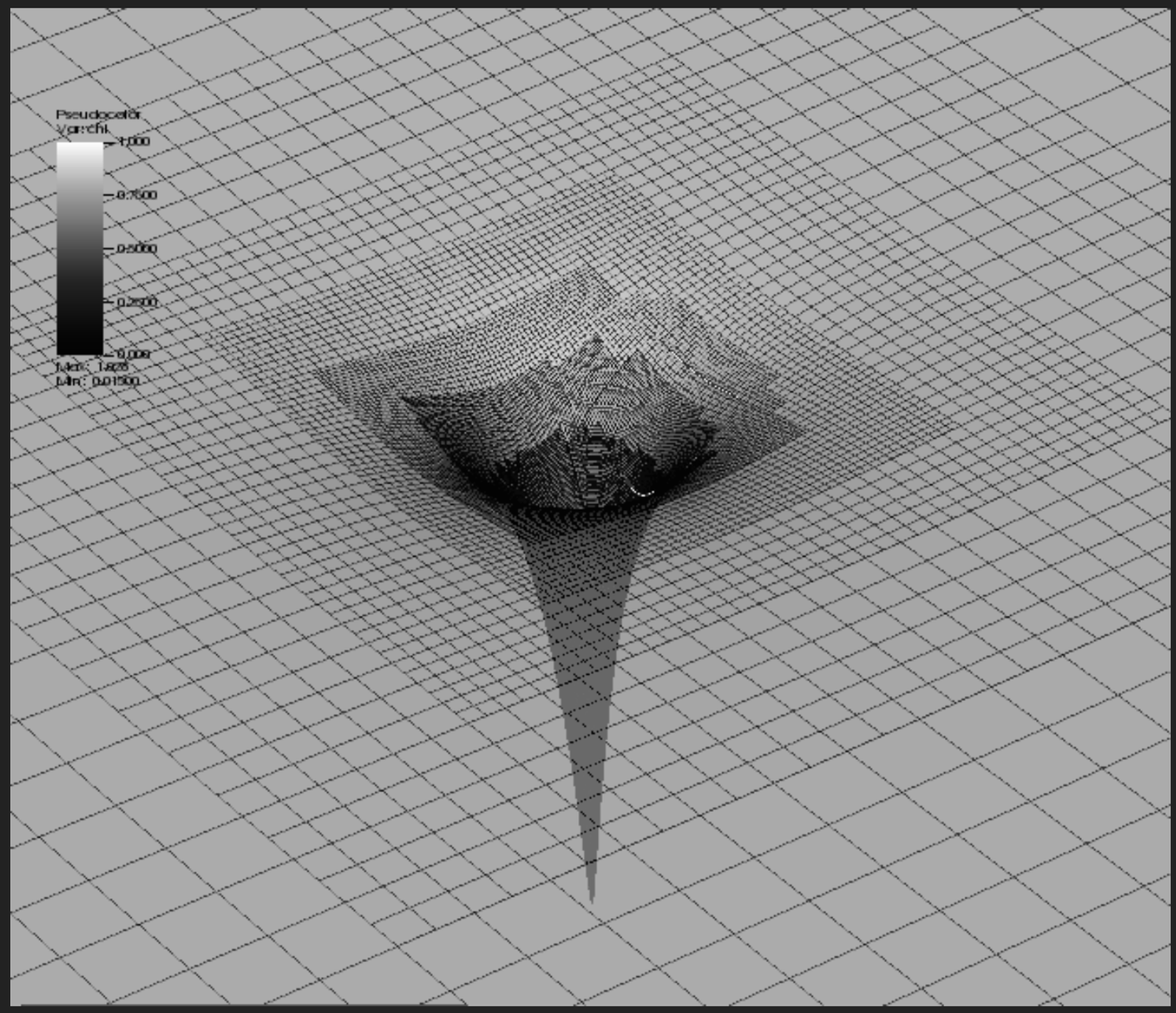
- GRChombo - specific physics actions common to most GR problems - GRAMR, GRAMRLevel, SimulationParametersBase

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

CHOMBO / GRCHOMBO / BINARYBH

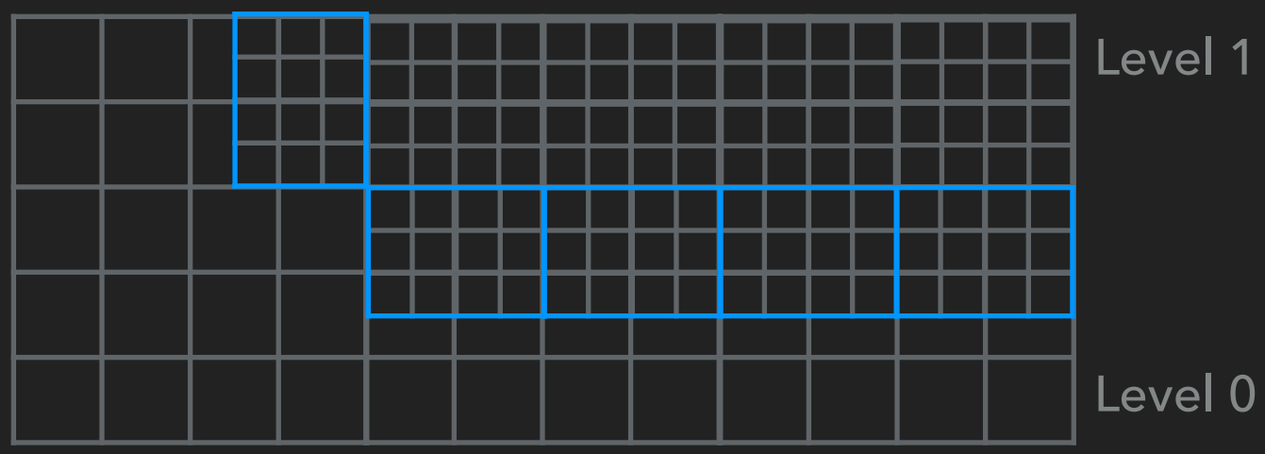


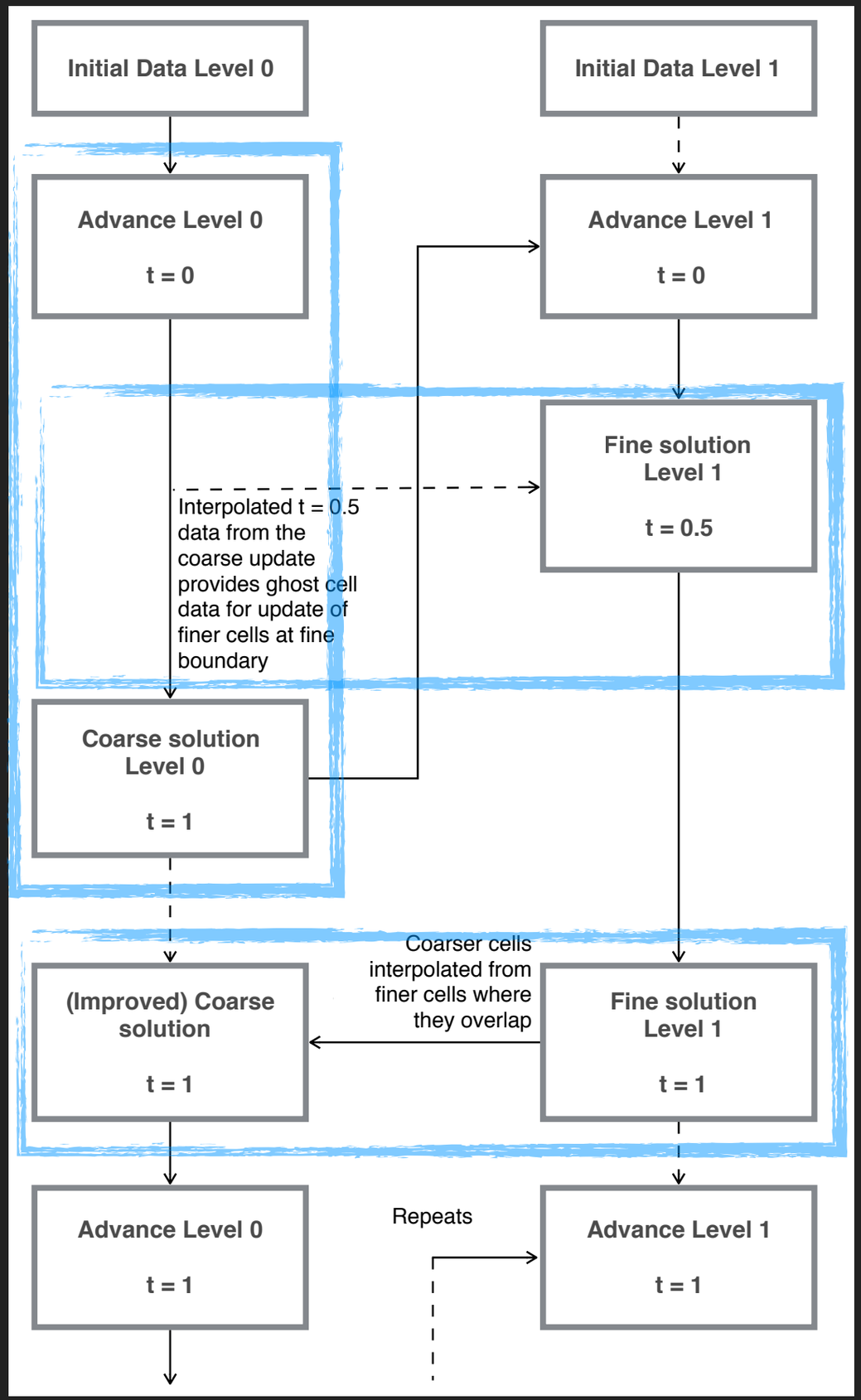
CHOMBO DEALS WITH THE ADAPTIVE MESH REFINEMENT (AMR)



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





WHERE ARE THE KEY CHOMBO FILES?

GRChombo / Chombo

Watch 10 | Star 3 | Fork 7

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Security | Insights | Settings

Branch: master ▾ | Chombo / lib / src / AMRTimeDependent / | Create new file | Upload files | Find file | History

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?

main GRChombo / Source GRChomboCore / Go to file Add file ···

mirenradia Fix absolute restart_file paths bug ··· ✓ 738e549 7 days ago History

..		
BoundaryConditions.cpp	Add fancier FOR macro replacing FOR1,FOR2,...	10 months ago
BoundaryConditions.hpp	Allow building GRChombo with NAMESPACE=TRUE (#151)	15 months ago
ChomboParameters.hpp	Fix absolute restart_file paths bug	7 days ago
DefaultLevelFactory.hpp	Allow building GRChombo with NAMESPACE=TRUE (#151)	15 months ago
EmptyDiagnosticVariables.hpp	Allow building GRChombo with NAMESPACE=TRUE (#151)	15 months ago
GRAMR.cpp	Move AMRInterpolator::fill_ghosts to GRAMR::fill_multilevel_ghosts	14 months ago
GRAMR.hpp	Move AMRInterpolator::fill_ghosts to GRAMR::fill_multilevel_ghosts	14 months ago
GRAMRLevel.cpp	Tidy up parameters files	13 months ago
GRAMRLevel.hpp	Tidy up parameters files	13 months ago
GRLevelData.cpp	Rewrite GRLevelData::plus with OpenMP and SIMD (#148)	14 months ago
GRLevelData.hpp	Rewrite GRLevelData::plus with OpenMP and SIMD (#148)	14 months ago
SetupFunctions.hpp	Fix absolute restart_file paths bug	7 days ago
SimulationParametersBase.hpp	Add fancier FOR macro replacing FOR1,FOR2,...	10 months ago
UserVariables.inc.hpp	Allow building GRChombo with NAMESPACE=TRUE (#151)	15 months ago
VariableType.hpp	Enable AMRInterpolator to interp diagnostic vars	2 years ago

WHERE ARE THE KEY BINARYBH FILES?

GRChombo / GRChombo

Watch 14 | Star 20 | Fork 22

Code | Issues 2 | Pull requests 4 | Actions | Projects 0 | Wiki | Security | Insights | Settings

Branch: master ▾ GRChombo / Examples / BinaryBH / | Create new file | Upload files | Find file | History

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 / GRChombo

Watch 14 | Star 20 | Fork 22

Code | Issues 2 | Pull requests 4 | Actions | Projects 0 | Wiki | Security | Insights | Settings

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

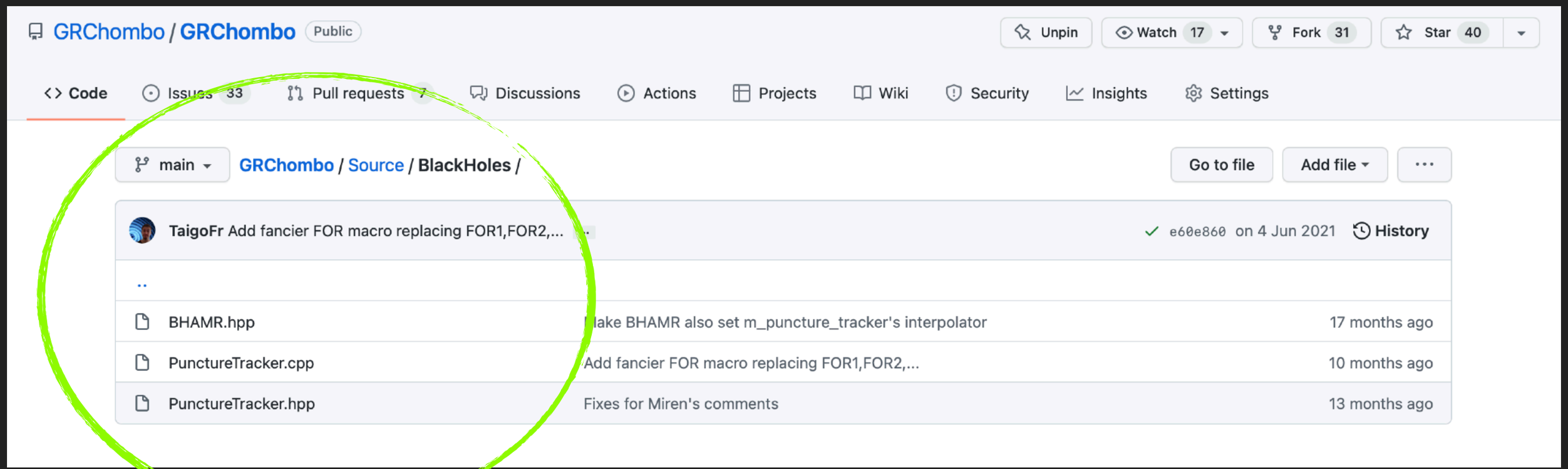
KAClough Separating out vars and tidy up comments — Latest commit aaafb6e on 12 Feb 2018

..

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 files 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.

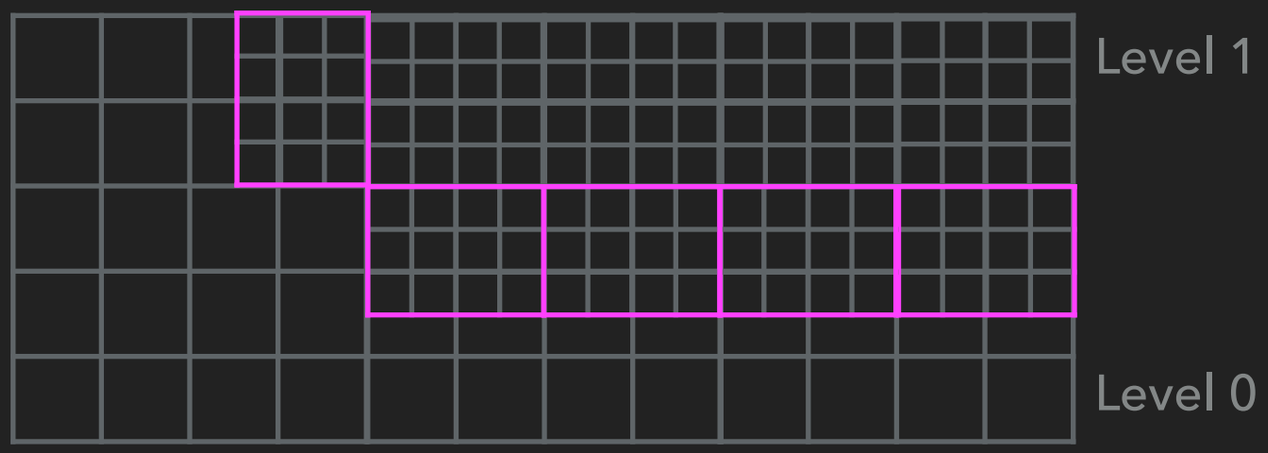
WHERE ARE THE KEY BINARYBH FILES?



NB: These files 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

```
769  //--------------------------------------------------------------------
770  // go baby go
771  void AMR::run(Real a_max_time, int a_max_step)
772  {
773    CH_TIME("AMR::run");
774
775    CH_assert(isDefined());
776    CH_assert(isSetUp());
777
778    if (m_verbosity >= 3)
779      {
780        pout() << "AMR::coarseTimeStep:" << endl;
781        pout() << "max_time = " << a_max_time << endl;
782        pout() << "max_step = " << a_max_step << endl;
783      }
784
```

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

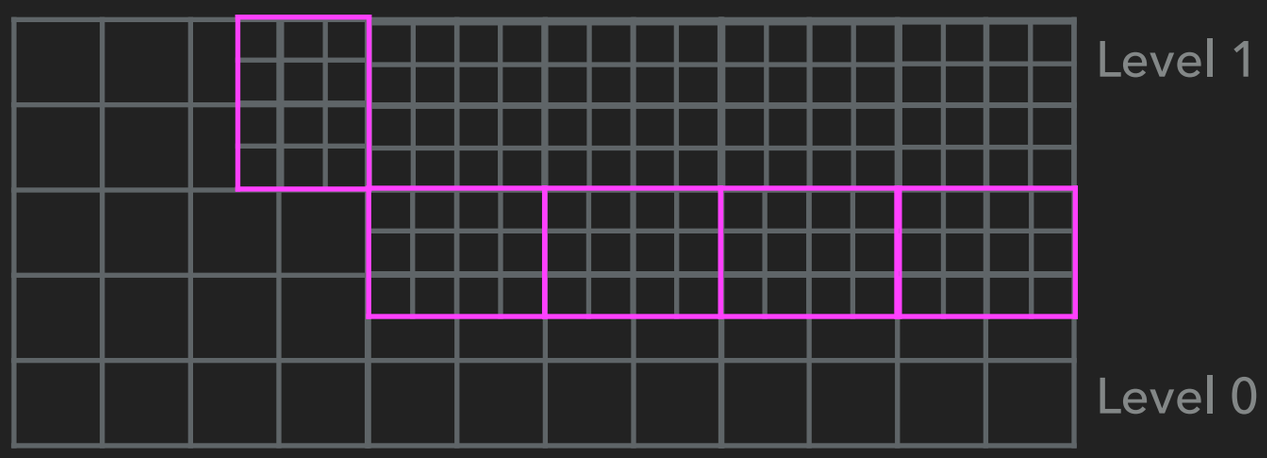
```
1810    // write physics class header data
1811    m_amrlevels[0]->writePlotHeader(handle);
1812
1813    // write physics class per-level data
1814    for (int level = 0; level <= m_finest_level; ++level)
1815      {
1816        m_amrlevels[level]->prePlotLevel();
1817        m_amrlevels[level]->writePlotLevel(handle);
1818      }
1819
```

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

Inheritance of AMR functions

For extracting data on all levels

Function to exchange ghosts on all levels

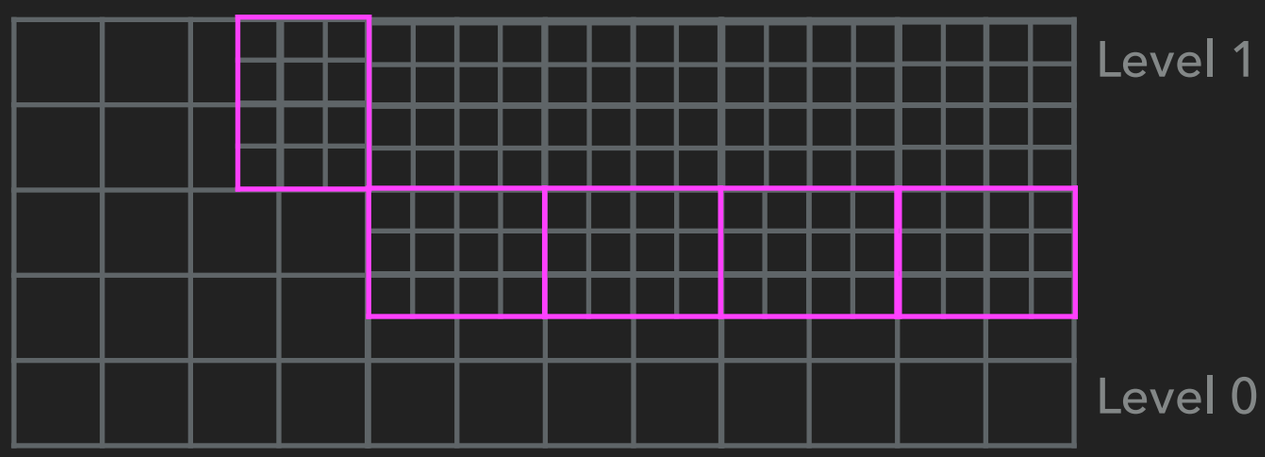
That's it!

```
37  class GRAMR : public AMR
38  {
39    private:
40      using Clock = std::chrono::steady_clock;
41      using Hours = std::chrono::duration<double, std::ratio<3600, 1>>;
42      std::chrono::time_point<Clock> start_time = Clock::now();
43
44    public:
45      AMRInterpolator<Lagrange<4>> *m_interpolator; //!< The interpolator pointer
46
47      GRAMR();
48
49      // defined here due to auto return type
50      auto get_walltime()
51      {
52          auto now = Clock::now();
53          auto duration = std::chrono::duration_cast<Hours>(now - start_time);
54
55          return duration.count();
56      }
57
58      // Called after AMR object set up
59      virtual void set_interpolator(AMRInterpolator<Lagrange<4>> *a_interpolator);
60
61      // returs a std::vector of GRAMRLevel pointers
62      // similar to AMR::getAMRLevels()
63      std::vector<GRAMRLevel *> get_gramrlevels();
64
65      // const version of above
66      std::vector<const GRAMRLevel *> get_gramrlevels() const;
67
68      // Fill ghosts on multiple levels
69      void fill_multilevel_ghosts(
70          const VariableType a_var_type,
71          const Interval &a_comps = Interval(0, std::numeric_limits<int>::max()),
```

Timing the simulation

STRUCTURE OF BHAMR

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



STRUCTURE OF BHAMR

```
5
6   #ifndef BHAMR_HPP_
7   #define BHAMR_HPP_
8
9   #include "GRAMR.hpp"
10  #include "PunctureTracker.hpp"
11
12  /// A child of Chombo's AMR class to interface with tools which require
13  /// access to the whole AMR hierarchy, and those of GRAMR
14  /**
15   * This object inherits from GRAMR and adds tools required for BH spacetimes
16   */
17  class BHAMR : public GRAMR
18  {
19    public:
20      PunctureTracker m_puncture_tracker;
21
22      BHAMR() {}
23
24      void set_interpolator(AMRInterpolator<Lagrange<4>> *a_interpolator) override
25      {
26          GRAMR::set_interpolator(a_interpolator);
27          m_puncture_tracker.set_interpolator(a_interpolator);
28      }
29  };
30
31  #endif /* BHAMR_HPP_ */
```

Inheritance of GRAMR functions (and so also AMR)

BH puncture tracking control added

Setting interpolator now needs to also set it for puncture tracking, (note this **overrides** the one in GRAMR)

ALL THIS COMES TOGETHER IN MAIN_BINARYBH.CPP

```
int runGRChombo(int argc, char *argv[])
{
    // Load the parameter file and construct the SimulationParameter class
    // To add more parameters edit the SimulationParameters file.
    char *in_file = argv[1];
    GRParmParse pp(argc - 2, argv + 2, NULL, in_file);
    SimulationParameters sim_params(pp);

    if (sim_params.just_check_params)
        return 0;

#ifdef USE_TWOPUNCTURES
    TPAMR bh_amr;
    bh_amr.set_two_punctures_parameters(sim_params.tp_params);
    // Run TwoPunctures solver
    bh_amr.m_two_punctures.Run();
#else
    BHAMR bh_amr;
#endif

    // must be before 'setupAMRObject' to define punctures for tagging criteria
    if (sim_params.track_punctures)
    {
        // the tagging criterion used in this example means that the punctures
        // should be on the max level but let's fill ghosts on the level below
        // too just in case
        int puncture_tracker_min_level = sim_params.max_level - 1;
        bh_amr.m_puncture_tracker.initial_setup(
            {sim_params.bh1_params.center, sim_params.bh2_params.center},
            "punctures", sim_params.data_path, puncture_tracker_min_level);
    }
```

Make a BHAMR object

Setup puncture tracking which lives in BHAMR

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)
DefaultLevelFactory<BinaryBHLevel> binary_bh_level_fact(bh_amr, sim_params);
setupAMRObject(bh_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(
    bh_amr, sim_params.origin, sim_params.dx, sim_params.boundary_params,
    sim_params.verbosity);
bh_amr.set_interpolator(
    &interpolator); // also sets puncture_tracker interpolator

// must be after interpolator is set
if (sim_params.track_punctures)
    bh_amr.m_puncture_tracker.restart_punctures();

 bh_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";

 bh_amr.conclude();

 CH_TIMER_REPORT(); // Report results when running with Chombo timers.

 return 0;
}
```

Setup using AMR functions

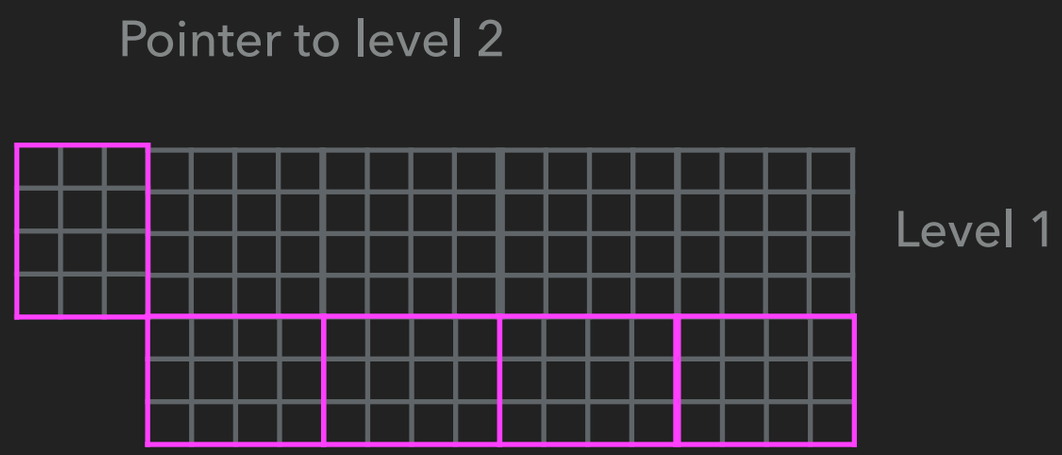
Setup interpolator using function in BHAMR

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



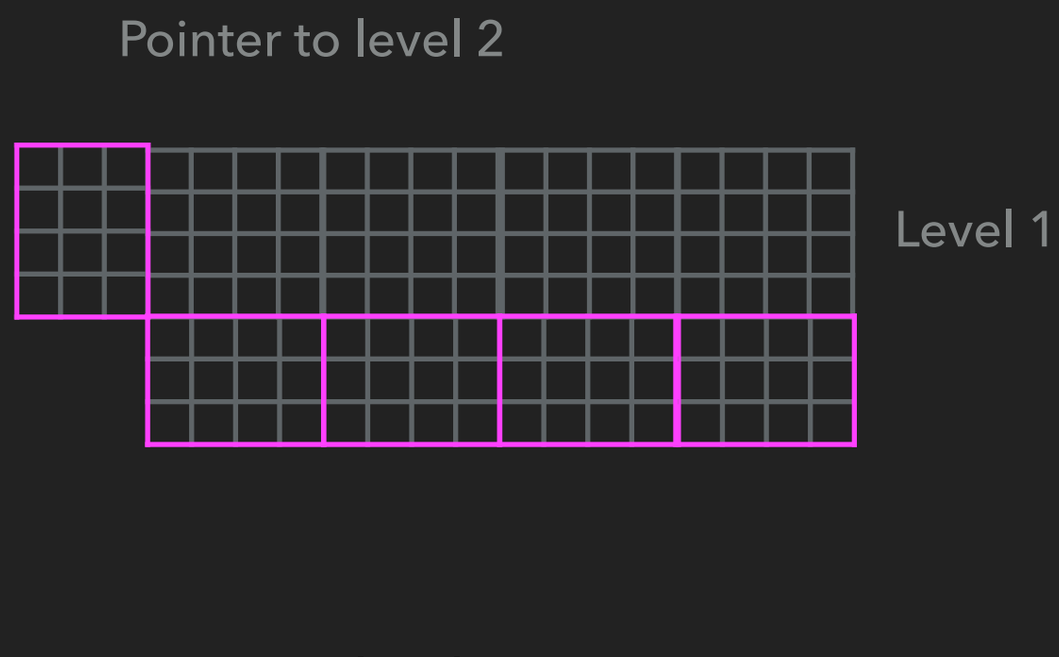
STRUCTURE OF AMRLEVEL

```
140     /**
141        Things to do after advancing this level by one time step.
142
143        This is a pure virtual function and  MUST be defined in the derived
144        class.
145
146     */
147     virtual
148       void postTimeStep() = 0;
149
150     ///
151     /**
152        Creates tagged cells for dynamic mesh refinement.
153
154        This is a pure virtual function and  MUST be defined in the derived
155        class.
156
157     */
158     virtual
159       void tagCells(IntVectSet& a_tags) = 0;
160
161     ///
162     /**
163        Creates tagged cells for mesh refinement at initialization.
164
165        This is a pure virtual function and MUST be defined in the derived
166        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

```
163  // things to do after a timestep
164  void GRAMRLevel::postTimeStep()
165  {
166      if (m_verbosity)
167          pout() << "GRAMRLevel::postTimeStep " << m_level << endl;
168
169      if (m_finer_level_ptr != nullptr)
170      {
171          GRAMRLevel *finer_gr_amr_level_ptr = gr_cast(m_finer_level_ptr);
172          finer_gr_amr_level_ptr->m_coarse_average.averageToCoarse(
173              m_state_new, finer_gr_amr_level_ptr->m_state_new);
174          // Synchronise times to avoid floating point errors for finer levels
175          finer_gr_amr_level_ptr->time(m_time);
176      }
177
178      specificPostTimeStep();
179
180      // enforce symmetric BCs - this is required after the averaging
181      // and postentially after specificPostTimeStep actions
182      fillBdyGhosts(m_state_new);
183
184      if (m_verbosity)
185          pout() << "GRAMRLevel::postTimeStep " << m_level << " finished" << endl;
186  }
187
188  // create tags
189  void GRAMRLevel::tagCells(IntVectSet &a_tags)
190  {
```

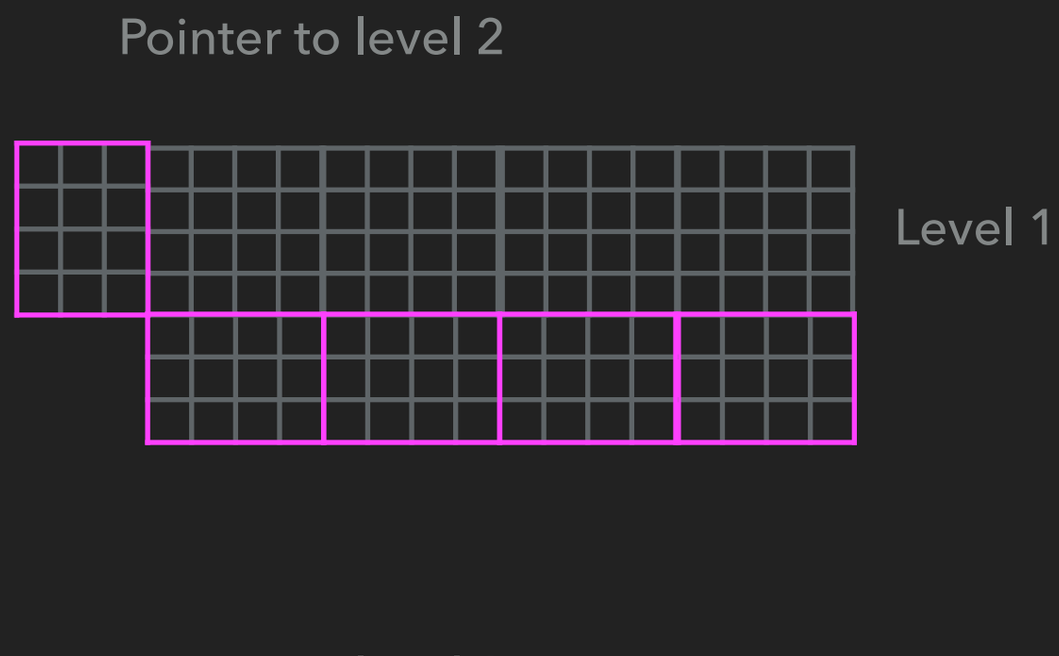
Overrrides the virtual function in AMRLevel

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()



STRUCTURE OF BINARYBHLEVEL

```
138  void BinaryBHLevel::specificPostTimeStep()
139  {
140      CH_TIME("BinaryBHLevel::specificPostTimeStep");
141      if (m_p.activate_extraction == 1)
142      {
143          // Populate the Weyl Scalar values on the grid
144          fillAllGhosts();
145          BoxLoops::loop(Weyl4(m_p.extraction_params.extraction_center, m_dx),
146                         m_state_new, m_state_new, EXCLUDE_GHOST_CELLS);
147
148          // Do the extraction on the min extraction level
149          if (m_level == m_p.extraction_params.min_extraction_level)
150          {
151              CH_TIME("WeylExtraction");
152              // Now refresh the interpolator and do the interpolation
153              m_gr_amr.m_interpolator->refresh();
154              WeylExtraction my_extraction(m_p.extraction_params, m_dt, m_time,
155                                           m_restart_time);
156              my_extraction.execute_query(m_gr_amr.m_interpolator);
157          }
158      }
159
160      // do puncture tracking on requested level
161      if (m_p.track_punctures == 1 && m_level == m_p.puncture_tracking_level)
162      {
163          CH_TIME("PunctureTracking");
164          // 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	Comment
initialdata()	required	define vars on initial grid
computeTaggingCriterion()	required	criterion for refinement
specificEvalRHS()	required	define evolution dvar/dt
specificPostTimestep()	optional	after level completes dt update
specificAdvance()	optional	happens in RK4 substeps
specificUpdateODE()	optional	happens in RK4 substeps
postRestart()	optional	done after checkpoint restart
preCheckpointLevel()	optional	before output checkpoint
prePlotLevel()	optional	before output plot file

FIND THIS AND MORE DETAILS IN THE WIKI!

GRChombo / GRChombo Public

Unpin | Watch 17 | Fork 31 | Star 40

Code | Issues 33 | Pull requests 7 | Discussions | Actions | Projects | Wiki | Security | Insights

Structure of the code (aka where is everything?)

Edit | New Page

K Clough edited this page on 15 Nov 2021 · 23 revisions

Yes, we know, GRChombo is a *big* code. At first the number of files will seem overwhelming, but with time you will start to learn where to find things and the structure will make (some) sense.

On this page we provide some hints on how to find your way around the code, but in the end you just have to dive in and learn as you go.

A useful pdf guide on this topic (with nicer pictures) from the latest GRChombo training day can be found in Useful resources. One should also look at the guides on C++ classes, inheritance and templating, which are used extensively in the code - some basic knowledge of these concepts is assumed below.

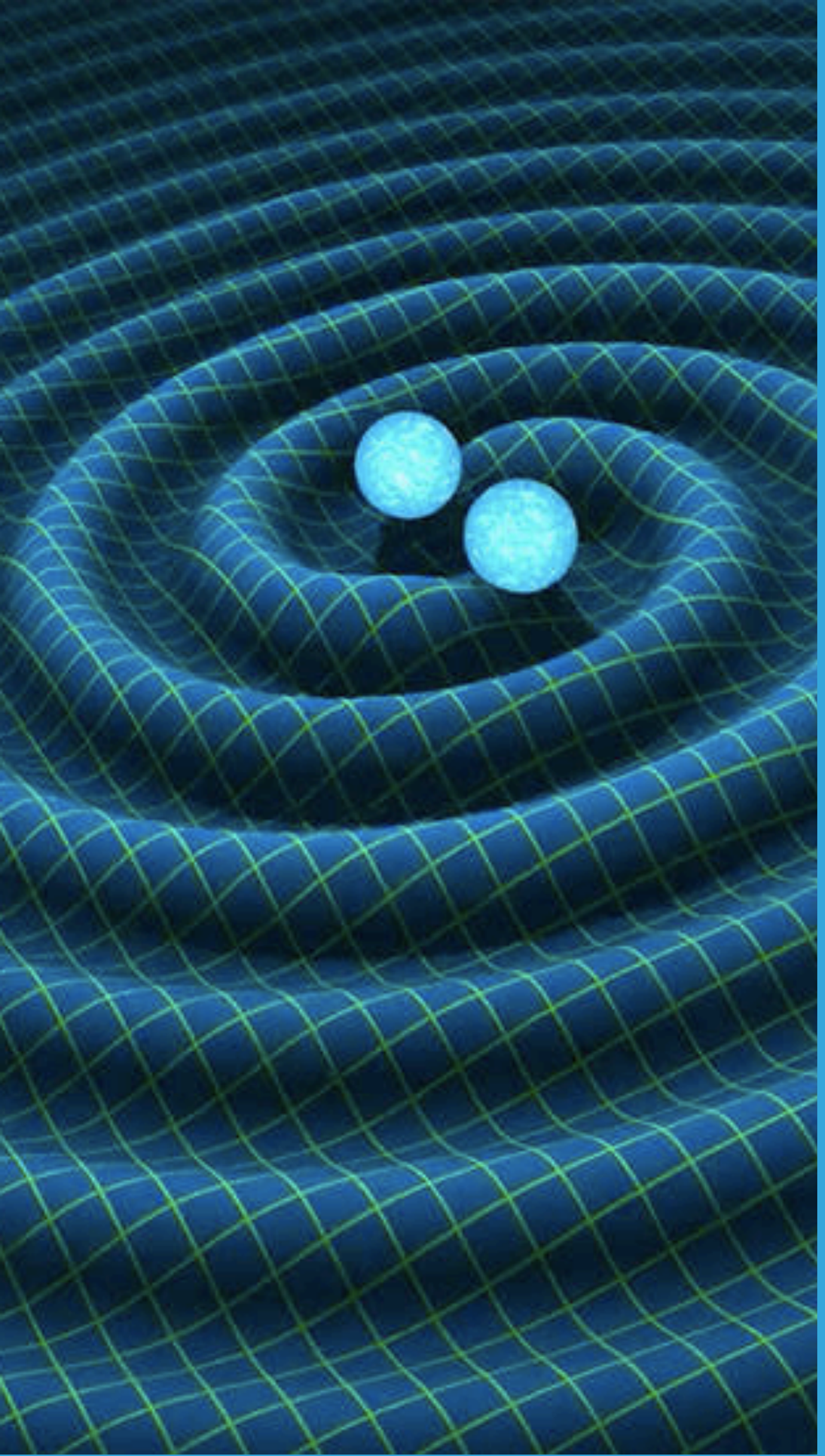
Hierarchy of GRChombo

The code is designed to have 3 main levels in its hierarchy, as follows:

1. Specific Example related files, e.g. for BinaryBH - specific actions relevant to the BinaryBH example - key classes include BHAMR, BinaryBHLevel, SimulationParameters. Also important are the namespaces UserVariables and DiagnosticVariables in the BinaryBH examples folder and BinaryBH (the initial data).

Pages 44

Contents



QUESTIONS?