Code generation for AMReX with applications to numerical relativity

Peterson, A.J.; Willcox, D.; Mösta, P.

DOI
10.1088/1361-6382/ad0b37
Publication date
2023
Document Version
Final published version
Published in
Classical and Quantum Gravity
License
Article 25fa Dutch Copyright Act (https://www.openaccess.nl/en/in-the-netherlands/you-share-we-take-care)
Link to publication

Citation for published version (APA):
Peterson, A. J., Willcox, D., & Mösta, P. (2023). Code generation for AMReX with applications to numerical relativity. *Classical and Quantum Gravity*, *40*(24), Article 245013. https://doi.org/10.1088/1361-6382/ad0b37

Class. Quantum Grav. **40** (2023) 245013 (28pp) https://doi.org/10.1088/1361-6382/ad0b37

Code generation for AMReX with applications to numerical relativity

Adam J Peterson[1,2,*], **Don Willcox**[1] **and Philipp Mösta**[3]

[1] Center for Computational Science and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States of America
[2] Strategic Deterrence Directorate, Lawrence Livermore National Laboratory, Livermore, CA 94550, United States of America
[3] GRAPPA, Anton Pannekoek Institute for Astronomy, Institute of High-Energy Physics, and Insitute of Theoretical Physics University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

E-mail: peterson129@llnl.gov

Received 1 March 2023; revised 25 October 2023
Accepted for publication 9 November 2023
Published 21 November 2023

Abstract
We present a new python/SymPy based code generator for producing executable numerical expressions for partial differential equations in AMReX-based applications. We demonstrate the code generator capabilities for the case of $3+1$ ADM formulations of numerical relativity for the constraint damped, conformal Z4 formulations (Z4c and CCZ4). The generated spacetime solvers are examined for stability and accuracy using a selection of checks from the standard Apples with Apples testbeds for numerical relativity applications. We also explore physically interesting vacuum spacetimes including head-on and inspiraling black hole binary collisions, and investigate the simulated gravitational waveforms from such events with the Newman–Penrose formulation of waveform extraction.

Keywords: code generation, adaptive mesh refinement, numerical relativity

(Some figures may appear in colour only in the online journal)

1. Introduction

One of the most exciting developments in astrophysics and cosmology in the last decade has been the arrival of gravitational wave astronomy. Observations from LIGO [1], VIRGO [2], and other gravitational wave observatories have opened up a new probe into the Universe [3],

[*] Author to whom any correspondence should be addressed.

and in particular, new testing into the strong field limit of general relativity via black hole/neutron star collisions. For both observational and testing purposes there comes the necessity to accurately simulate relativistic spacetime physics in the regions of large curvature (non-perturbative), such as those in the vicinity of black holes and neutron stars. Also of interest is the dynamical evolution of stellar core collapse supernovae, which for proper understanding, requires accurate evolution of general relativistic/magnetohydrodynamic systems.

For these reasons (and many others) a market has developed for large scale numerical simulations capable of evolving general relativistic systems in both vacuum and matter filled spacetimes. In the past decade many codes have been developed that successfully evolve spacetimes for both black hole and neutron star systems, in particular those in compact binary evolutions. It has also become clear that such complex systems involve highly non-trivial partial differential equations associated with the $3+1$ decomposition of Einstein's equations. Since no single code can capture all of the physics of interest for all studies of astrophysical and cosmological systems, in many cases one is required to develop codes from scratch that are tailored for the particular physics of interest. For each study, one is required to translate tensor symbolic expressions to computational numerical language (finite-differencing, index expansions, etc) for each separate project. For such complex numerical systems, this is both impractical and inconvenient for the researcher interested in modeling multiple physical phenomena in the context of high spacetime curvature.

It has also become clear in the past few years that $3+1$ formulations of general relativity provide excellent testing grounds for new PDE code generators, as they are well studied systems with numerous examples in the literature to compare results. Formulations such as the BSSNOK [4, 5] and Z4 [6–8] approaches provide particularly rigorous tests of code generation due to their large number of evolved variables, non-linearity of their equations of motion, and multiple finite-difference stenciling for both forward, backward, and centered indexing. They are also known to be numerically stable [9] if formulated correctly, and thus provide a sensitive test of the correctness of the generated code [10].

The goal of this project is to develop and demonstrate the utility of code generation to facilitate the translation of symbolic tensor formulations of equations to discretized formulas written in low level programming languages such as C++ or Fortran. Python based symbolic manipulation tools such as SymPy, NumPy, and SciPy are excellent applications for performing calculations on symbolic/tensor objects. However, as numerical solvers for large systems increasingly rely on parallelization and adaptive mesh refinement (AMR) based applications that are typically written in low level languages, it is necessary to develop translators that can generate executable expressions for such languages.

To this end we develop a code generator (which will be made available for public use) capable of producing executable expressions for AMReX based PDE solvers from symbolic manipulation tools written in SymPy and NumPy. Previous projects have produced packages designed specifically for mathematica [11] and python [12] interfaces with more examples found in the Einstein Toolkit, and we wish to extend these interests to a python-to-AMReX code generation. Specifically, for this project, we demonstrate code generation for the conformal Z4 (Z4c) [6–8] and conformal covariant Z4 (CCZ4) [13, 14] formulations of general relativity and evaluate the resulting system using selected examples of Apples with Apples tests for numerical spacetime solvers. AMReX provides the necessary infrastructure for developing massively parallel block structured AMR applications. In particular, AMReX provides the necessary tools to evolve hyperbolic PDEs associated with $3+1$ formulations of general relativity. The code generator is designed to read textbook expressions of symbolic equations written in SymPy and produce the necessary finite differenced equivalents written as AMReX executable lines.

We organize our presentation as follows: We begin in section 2 with a review of the Z4c formulations of general relativity. In section 3 we discuss numerical methods for finite differencing and AMReX's algorithms for AMR. Section 4 is devoted to the specifics of code generation for AMReX applications. In section 5 we demonstrate some of the standard tests for our numerical solvers, and apply the code to black hole binary systems in section 6. We conclude with a discussion and summary in section 7.

2. Review of $3+1$ formulation of Einstein equations

For the bulk of the simulations performed in this project, we used the $3+1$ Z4c formulation [6] of numerical relativity with constraint damping [7]. The Z4c formulation is based on the extension of Einstein's equations to include constraint damping terms for stability purposes. More specifically, the Z4c formulation is given as follows:

$$G_{ab} = 8\pi T_{ab} - 2\nabla_{(a}Z_{b)} + g_{ab}\nabla_c Z^c + \kappa_1\left[2n_{(a}Z_{b)} + \kappa_2 g_{ab}n_c Z^c\right], \tag{1}$$

with the Einstein tensor:

$$G_{ab} = R_{ab} - \frac{1}{2}g_{ab}R. \tag{2}$$

Here, R_{ab} is the Ricci tensor and R the Ricci scalar. ∇_a is the covariant derivative compatible with the metric g_{ab}. Z_a is a vector field of constraints. The field n_a is the timelike unit normals to the spacelike hypersurfaces in the $3+1$ decomposition of the spacetime manifold. The constraint Z_a is decomposed into the normal projection $\Theta \equiv -n^a Z_a$, and $Z_i = \perp_i^a Z_a$ orthogonal to the normal n_a. Clearly, if $Z_a = 0$, solutions to (1) are also solutions of the Einstein equations.

The introduction of the fields Z_a in (1) introduce additional evolution equations in the Z4c system. Thus in addition to the dynamical evolution of the spatial metric γ_{ij} and extrinsic curvature K_{ij},

$$\begin{aligned}\partial_t\gamma_{ij} &= -2\alpha K_{ij} + \mathcal{L}_\beta\gamma_{ij}\\ \partial_t K_{ij} &= -D_iD_j\alpha + \alpha\Bigg[R_{ij} + KK_{ij} - 2K_{ik}K^k_j + 2\hat{D}_{(i}Z_{j)}\\ &\qquad - \kappa_1(1+\kappa_2)\Theta\gamma_{ij}\Bigg] + 4\pi\alpha\left[\gamma_{ij}(S-\rho_{\mathrm{ADM}}) - 2S_{ij}\right] + \mathcal{L}_\beta K_{ij},\end{aligned} \tag{3}$$

additional equations for Θ and Z_i are included in the dynamical evolution:

$$\begin{aligned}\partial_t\Theta &= \frac{1}{2}\alpha\left[H + 2\hat{D}^i Z_i - 2\kappa_1(2+\kappa_2)\Theta\right] + \mathcal{L}_\beta\Theta\\ \partial_t Z_i &= \alpha\left[M_i + D_i\Theta - \kappa_1 Z_i\right] + \gamma^{1/3}Z^j\partial_t\left[\gamma^{-1/3}\gamma_{ij}\right] + \beta^j\hat{D}_j Z_i,\end{aligned} \tag{4}$$

with the definition

$$\hat{D}_i Z_j \equiv \gamma^{1/3}\gamma_{kj}\partial_i\left[\gamma^{-1/3}Z^k\right]. \tag{5}$$

Here $\gamma \equiv \det \gamma_{ij}$. Additionally, the Hamiltonian and momentum constraints are written as:

$$H = R - K_{ij}K^{ij} + K^2 - 16\pi\,\rho_{\mathrm{ADM}} \tag{6}$$
$$M_i = D^j\left[K_{ij} - \gamma_{ij}K\right] - 8\pi\,S_i\,. \tag{7}$$

The Z4c system is further developed with a conformal transformation of the spatial metric, along with additional conformal transformations of the dynamical variables. We define the following:

$$\tilde{\gamma}_{ij} = \gamma^{-1/3}\gamma_{ij}, \qquad \phi = \frac{1}{12}\log\gamma \tag{8}$$
$$\tilde{A}_{ij} = \gamma^{-1/3}\left(K_{ij} - \frac{1}{3}\gamma_{ij}K\right), \qquad \hat{K} = K - 2\Theta \tag{9}$$
$$\tilde{\Gamma}^i = 2\tilde{\gamma}^{ij}Z_j + \tilde{\gamma}^{ij}\tilde{\gamma}^{kl}\partial_l\tilde{\gamma}_{jk}, \qquad \tilde{\Gamma}^i_{\mathrm{d}} = \tilde{\Gamma}^i_{jk}\tilde{\gamma}^{jk}. \tag{10}$$

With these definitions the Z4c system (3) and (4) takes the form

$$\partial_t\phi = -\frac{1}{6}\alpha\left(\hat{K} + 2\Theta\right) + \beta^i\partial_i\phi + \frac{1}{6}\partial_i\beta^i \tag{11}$$
$$\partial_t\tilde{\gamma}_{ij} = -2\alpha\tilde{A}_{ij} + 2\tilde{\gamma}_{k(i}\partial_{j)}\beta^k - \frac{2}{3}\tilde{\gamma}_{ij}\partial_k\beta^k + \beta^k\partial_k\tilde{\gamma}_{ij} \tag{12}$$
$$\begin{aligned}\partial_t\hat{K} = {}& -D_iD^i\alpha + \alpha\left[\tilde{A}_{ij}\tilde{A}^{ij} + \frac{1}{3}\left(\hat{K} + 2\Theta\right)^2 + \kappa_1\left(1-\kappa_2\right)\Theta\right] \\ & + 4\pi\,\alpha\left(S + \rho_{\mathrm{ADM}}\right) + \beta^i\,\partial_i\hat{K}\end{aligned} \tag{13}$$
$$\begin{aligned}\partial_t\tilde{A}_{ij} = {}& e^{-4\phi}\left[-D_iD_j\alpha + \alpha\left(R_{ij} - 8\pi\,S_{ij}\right)\right]^{\mathrm{tf}} + \alpha\left[\left(\hat{K} + 2\Theta\right)\tilde{A}_{ij} - 2\tilde{A}_{ik}\tilde{A}^k_j\right] \\ & + 2\tilde{A}_{k(i}\partial_{j)}\beta^k - \frac{2}{3}\tilde{A}_{ij}\partial_k\beta^k + \beta^k\partial_k\tilde{A}_{ij}\end{aligned} \tag{14}$$
$$\begin{aligned}\partial_t\Theta = {}& \frac{1}{2}\alpha\Bigg[R - \tilde{A}_{ij}\tilde{A}^{ij} + \frac{2}{3}\left(\hat{K} + 2\Theta\right)^2 - 16\pi\,\rho_{\mathrm{ADM}} \\ & - 2\kappa_1\left(2+\kappa_2\right)\Theta\Bigg] + \beta^i\,\partial_i\Theta\end{aligned} \tag{15}$$
$$\begin{aligned}\partial_t\tilde{\Gamma}^i = {}& \tilde{\gamma}^{jk}\partial_j\partial_k\beta^i + \frac{1}{3}\tilde{\gamma}^{ij}\partial_j\partial_k\beta^k - 2\tilde{A}^{ij}\partial_j\alpha + 2\alpha\Bigg[\tilde{\Gamma}^i_{jk}\tilde{A}^{jk} + 6\tilde{A}^{ij}\partial_j\phi \\ & - \frac{1}{3}\tilde{\gamma}^{ij}\partial_j\left(2\hat{K} + \Theta\right) - \kappa_1\left(\tilde{\Gamma}^i - \tilde{\Gamma}^i_{\mathrm{d}}\right) - 8\pi\tilde{\gamma}^{ij}S_j\Bigg] \\ & + \frac{2}{3}\tilde{\Gamma}^i_{\mathrm{d}}\,\partial_j\beta^j - \tilde{\Gamma}^j_{\mathrm{d}}\partial_j\beta^i + \beta^j\partial_j\tilde{\Gamma}^i.\end{aligned} \tag{16}$$

Here the Ricci tensor R_{ij} of γ_{ij} can be decomposed into a conformal part R^{ϕ}_{ij} and the Ricci tensor $\tilde{R}_{ij}$ associated with $\tilde{\gamma}_{ij}$. Additionally, the constraints associated with Z_i are absorbed into the definition of $\tilde{R}_{ij}$:

$$R_{ij} = R^{\phi}_{ij} + \tilde{R}_{ij} \tag{17}$$

$$R^{\phi}_{ij} = -2\tilde{D}_i\tilde{D}_j\phi - 2\tilde{\gamma}_{ij}\tilde{D}_k\tilde{D}^k\phi + 4\left(\tilde{D}_i\phi\tilde{D}_j\phi - \tilde{\gamma}_{ij}\tilde{D}_k\phi\tilde{D}^k\phi\right) \tag{18}$$

$$\tilde{R}_{ij} = -\frac{1}{2}\tilde{\gamma}^{lm}\partial_i\partial_j\tilde{\gamma}_{lm} + \tilde{\gamma}_{k(i}\partial_{j)}\tilde{\Gamma}^k + \tilde{\Gamma}^k_{\mathrm{d}}\tilde{\Gamma}_{(ij)k} + \tilde{\gamma}^{lm}\left(2\tilde{\Gamma}^k_{l(i}\tilde{\Gamma}_{j)km} + \tilde{\Gamma}^k_{im}\tilde{\Gamma}_{klj}\right). \tag{19}$$

This formulation also introduces the algebraic constraints:

$$\det\tilde{\gamma} = 1, \quad \mathrm{Tr}A \equiv \tilde{\gamma}^{ij}\tilde{A}_{ij} = 0. \tag{20}$$

Finally, to close the system of PDEs, we must introduce gauge fixing of the lapse α, and shift β. For most of the systems considered in this project we will implement the puncture gauge conditions composed of the Bona–Masso lapse [15] and the Gamma-Driver condition on the shift [9, 16]:

$$\partial_t\alpha = -\mu_{\mathrm{L}}\alpha^2\hat{K} + \beta^i\partial_i\alpha \tag{21}$$

$$\partial_t\beta^i = \mu_{\mathrm{S}}\alpha^2\tilde{\Gamma}^i - \eta\beta^i + \beta^j\partial_j\beta^i. \tag{22}$$

For most purposes the $1+\log$ variant is used with $\mu_L = 2/\alpha$ and $\mu_S = 1/\alpha^2$.

For completion of the Apples with Apples tests, we perform a gauge wave test below where we used the CCZ4 formulation described in the appendix. In this case we will also employ the harmonic lapse condition $\mu_L = 1$ with zero shift $\beta = 0$.

All of the dynamical tests considered in this project are performed in vacuum spacetimes with $\rho_{\mathrm{ADM}} = S_{ij} = 0$.

3. Numerical methods

In this section we will describe in detail our numerical methods for evaluating the Z4 systems described in the previous section. We intend to make our formulations and explanations as detailed as possible so other research efforts may compare with our results with no ambiguities in methods appearing. This section will be devoted to the general numerical methods used for our simulations below. Additional information specific to the particular problem will be described in the appropriate sections below.

3.1. Discretization

For spacetime solver applications we mostly employ the cell-centered data approach, however we occasionally make use of mixed nodal and cell centered data for certain one dimensional tests where determining convergence behavior was of primary interest.

For all problems considered, evolution is carried out using fourth order Runge–Kutta (RK) time integration. The algebraic constraints (20) are enforced at each of the substeps of the RK time step. Spatial discretization is carried out using both second order and fourth order accurate finite differencing depending on the problem of interest. For second order accurate problems centered discretization is performed as:

$$\partial_i \to D_{0i}, \quad \partial_i\partial_j \to \begin{cases} D_{0i}D_{0j}, & \text{if } i \neq j \\ D_{+i}D_{-i} & \text{if } i = j \end{cases}, \tag{23}$$

where

$$\begin{aligned}
D_+ v_j &\equiv \frac{v_{j+1} - v_j}{\Delta x} \\
D_- v_j &\equiv \frac{v_j - v_{j-1}}{\Delta x} \\
D_0 v_j &\equiv \frac{v_{j+1} - v_{j-1}}{2\Delta x} \\
D_+ D_- v_j &\equiv \frac{v_{j+1} - 2v_j + v_{j-1}}{\Delta x^2}.
\end{aligned} \tag{24}$$

For fourth order accurate problems the spatial derivatives are defined for centered discretization as:

$$\begin{aligned}
\partial_i &\to D_i^{(4)} \equiv D_{0i}\left(1 - \frac{\Delta x^2}{6} D_{+i} D_{-i}\right) \\
\partial_i \partial_j &\to D_{0i}^{(4)} \equiv \begin{cases} D_{0i}^{(4)} D_{0j}^{(4)}, & \text{if } i \neq j \\ D_{+i} D_{-j}\left(1 - \frac{\Delta x^2}{12} D_{+i} D_{-i}\right) & \text{if } i = j \end{cases}.
\end{aligned} \tag{25}$$

To accommodate the advection terms $\beta^i \partial_i f$ appearing in the Z4c system (16) and gauge conditions (22), we make use of upwinded discretization:

$$\partial_i \to D_i^{(\mathrm{up})} = \begin{cases} D_{+i} - \frac{1}{2}\Delta x_i D_{+i} D_{+i} & \text{if } \beta^i > 0 \\ D_{-i} + \frac{1}{2}\Delta x_i D_{-i} D_{-i} & \text{if } \beta^i < 0 \end{cases}. \tag{26}$$

Finally, to stabilize high frequency modes which are excited in the numerical evolution, we employ Kreiss-Oliger dissipation whereby the right hand sides (RHS) of the equations of motion are modified:

$$\partial_t \mathbf{f} = \mathrm{RHS}(\mathbf{f}) \to \partial_t \mathbf{f} = \mathrm{RHS}(\mathbf{f}) + Q\mathbf{f}, \tag{27}$$

where the operator Q is defined as $Q = Q_x + Q_y + Q_z$, with

$$Q_x \equiv \sigma_{\mathrm{KO}} (-1)^{r+1} \Delta x^{2r-1} (D_+)^r (D_-)^r / 2^{2r}, \tag{28}$$

and correspondingly for Q_y and Q_z. Here r is defined as the $r \equiv \mathcal{O}/2 + 1$ for an order $\mathcal{O}$ accurate finite differencing scheme. For our purposes we usually set $\sigma_{\mathrm{KO}} \sim 0.1$, however for CCZ4 problems a higher value $\sigma_{\mathrm{KO}} \sim 0.5$ is necessary.

3.2. *AMR*

To support the multiple length scales within black hole binary simulations (evolution near the event horizon, and waveform extraction in the linear region), as well as ensuring that boundary conditions do not interfere with the physical region of interest, most spacetime evolution simulations rely on either fixed (FMR) or AMR techniques. For this project we have built our code in the AMReX software framework.

AMReX is a software framework for providing massively parallel, block-structured AMR applications. AMReX uses a nested hierarchy of logically rectangular grids as illustrated in figure 1. Here refined grid domains are always bounded by their parent grid domain, though they may intersect multiple blocks within that parent grid. Each grid block is also formulated with appropriate ghost cells which are either filled with boundary conditions (if the block

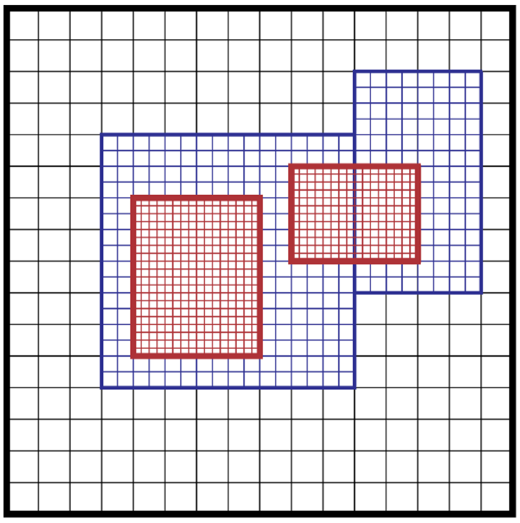

Figure 1. Illustration of AMR grids with two levels of factor 2 refinement. The coarsest grid covers the domain with 16^2 cells. Bold lines represent grid boundaries. The two intermediate resolution grids are at level 1 and the cells are a factor of two finer than those at level 0. The two finest grids are at level 2 and the cells are a factor of 2 finer than the level 1 cells. Note that the level 2 grids are properly nested within the union of level 1 grids, but there is no direct parent-child connection.

intersects the domain boundary) or are filled by interpolation from the next coarser grid. The interpolation procedure is described in the following paragraphs.

Refinement of grids takes place simultaneously in both space and time. Our particular AMReX applications use the subcycling-in-time approach, whereby finer grids are evolved multiple times with smaller time steps than the coarser levels. Figure 2 illustrates the sequence of time step evolution at specific refinement levels.

For all FMR and AMR applications used in this project, during regridding, refined levels are filled using the cell conservative quartic method. In this method a 4th order polynomial is used to fit the data. For each cell involved in constructing the polynomial, the average of the polynomial inside that cell is equal to the cell averaged value of the original data on the coarser level. We find that this approach (of the standard interpolations available in the AMReX toolkit) leads to the best stability and minimizes reflections of constraint violations at refinement boundaries. Coarse level data that is covered by a finer grid is updated by averaging down the fine level data after each RK stage.

To maintain 4th-order time and spatial accuracy at coarse-fine boundaries we perform a 4th-order accurate time interpolation of fine ghost cells at each stage of the RK procedure. In this procedure approximations for fine level ghost data at each intermediate stage of the RK procedure are determined by interpolating solutions and derivatives at the coarse level using the RK stage values $k_1, \ldots, k_4$. The specifics of this procedure are illustrated in great detail in section 3.1 of [17].

3.3. *Specifics of AMReX*

The AMReX framework includes several parameters for the user to select in order to optimize refinement and regridding, as well as optimizing memory distribution for parallel processes.

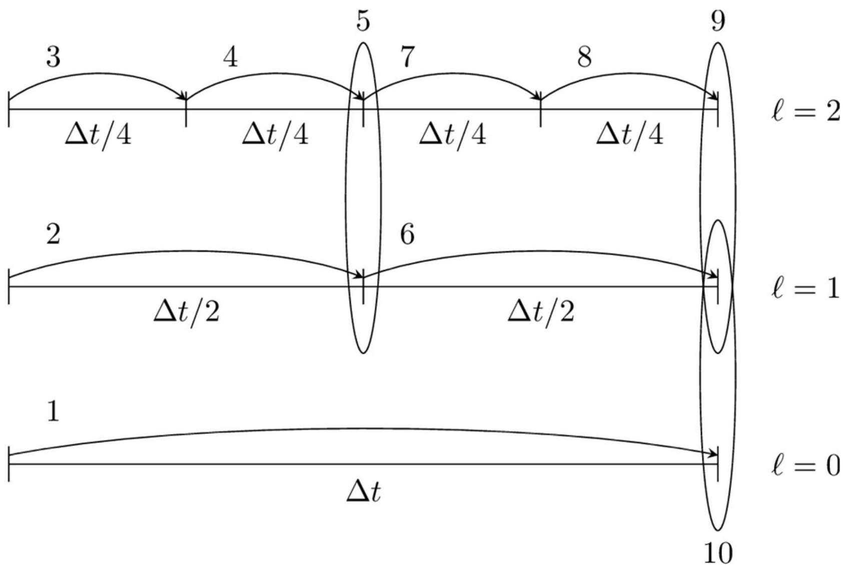

Figure 2. The subcycling approach is illustrated as a nested sequence of time substeps in each level. Here the sequence of level advancement is shown for the example of 2 refinement levels (in addition to the coarse level) and can be extrapolated to any max level. Higher levels are advanced with smaller time steps and performed in the order shown (labeled as steps 1 through 9).

Throughout this project we include specifications of these parameters so future researchers may recreate our results with as little ambiguity as possible. We describe the meaning and use of these additional parameters here.

During the gridding and regridding processes, at each level of refinement the domain is divided into logically rectangular blocks of cells based on the parameters specified at run time. When dividing the region, cells in a block must be divisible by the blocking factor (in each direction, though we use one value for all directions in this project), and blocks may contain no more cells than specified by the max grid size.

Finally, when tagging cells for refinement a box is grown in each direction by a minimum of the number of cells specified. Throughout this work we use one buffer cell when tagging and regridding.

4. AMReX code generator

The complexity of the Einstein equations presents a major hurdle towards writing efficient code in low level languages, where the barrier is significant between symbolic tensorial expressions and the executable blocks of code. Many high level languages such as python and mathematica provide tensor manipulation packages that can be used to perform index manipulation within tensor calculations.

However, symbolic outputs from high level languages need significant manipulation to make equivalent low level executable blocks. One example occurs when one wishes to write finite differenced equivalents of partial derivatives especially for higher order accuracy to fourth order and beyond. Equally prevalent is the case of tensor index expansion.

The case of writing executable code blocks for Einstein's equations are particularly involved when one considers the requirement of fourth (or higher) order finite differencing with multiple centering schemes, and the complexity of the tensor expressions themselves. Additionally, one is required to write right hand side expressions for upwards of 20 components, as well as perform post-updating corrections after each RK time step in the integration.

Many packages have been developed (such as Kranc, nrpy+, and Cpi) to overcome this hurdle, by automating the process of translating symbolic (textbook) expressions to executable blocks in low level languages such as C/C++ or Fortran. One goal for this project was to develop an equivalent method of code generation to translate the symbolic Z4c expressions (16) into executable lines of code for AMReX, which itself is built on C++ and Fortran. The AMReX Code Generator is built using SymPy to ease the manipulation and expansion of tensors and finite difference derivatives, as well as make use of general definitions of Christoffel symbols and curvature tensors from their formulations from the metric tensor. Thus the user has to spend less effort expanding such definitions, and instead may rely on the code generator to produce the correctly expanded expressions.

The logical architecture of the code generator is designed around symbolic objects that contain information that can be accessed via member functions. To elaborate, a symbolic object such as the scalar curvature K or a component of the spatial metric tensor γ_{12} contains specific data that can be accessed and arranged to output expressions AMReX compilable code . Each symbolic object contains member variables such as the symbol name (e.g. ψ), the AMReX indexed expression (for referencing grid data), and the symbolic equation referenced as one might wish to do with intermediate symbols such as the spatial Christoffel symbols Γ^i_{jk} which contain expressions written in spatial derivatives of the spatial metric tensor γ_{ij}.

The pieces of data contained in the symbolic objects can be accessed and used in subsequent expressions as SymPy symbolic objects. The user may, for example, reference symbolic objects and their symbolic derivatives when building a 'right hand side' expression as will be demonstrated below. Addtionally, member functions can be used to output expresssions in AMReX code such as assigning a variable to specific grid data, or intermediate expression.

As an example, if one is considering a simple Klein–Gordon system written in first order in time derivatives:

$$\begin{aligned}\partial_t\psi &= \pi\\ \partial_t\pi &= m^2\psi + \nabla^2\psi.\end{aligned} \tag{29}$$

In a python script one would write:

```
psi = stvar('psi', state = True) \\Declaration of evolved variables
pi = stvar('pi', state = True)

dtpsi = stvar('RHS_psi', rhs = True) \\Declaration of right hand
sides
dtpi = stvar('RHS_pi', rhs = True)

m = stvar('m') \\Constant mass term

dDDpsi_LL.expr = psi.diff('dDD', Accuracy = 2) \\2nd order, 2nd
Derivative

dtpsi.expr = pi.symb
```

```
dtpi.expr = m**2*psi.symb**2 + dDDpsi_LL[0][0].symb
+ dDDpsi_LL[1][1].symb
```

One may then output these variables in executable form, such as:

```
print(psi.symb2isymb())
print(dtpsi.symb2expr())
```

with the resulting output:

```
psi = state(i, j, k, Idx::psi);
rhs(i, j, k, Idx::psi) = std::pow(m,2)*psi + dDDpsi_LL_00
+ dDDpsi_LL_11;
```

Additionally, one may output appropriately expanded finite differencing formulas for numerical derivatives to arbitrary accuracy and with offsetting for up/down-winding. For example, the 4th order central differenced first derivatives of ψ can be accessed using:

```
print(dDpsi_L.symb2expr())
```

with output

```
amrex::Real dDpsi_L_0 = ((2.0/3.0)*state_fab(i + 1, j, k, Idx::psi)
                       - 1.0/12.0*state_fab(i + 2, j, k, Idx::psi)
                       - 2.0/3.0*state_fab(i - 1, j, k, Idx::psi)
                       + (1.0/12.0)*state_fab(i - 2, j, k,
                        Idx::psi))/dx[0];
amrex::Real dDpsi_L_1 = ((2.0/3.0)*state_fab(i, j + 1, k, Idx::psi)
                       - 1.0/12.0*state_fab(i, j + 2, k, Idx::psi)
                       - 2.0/3.0*state_fab(i, j - 1, k, Idx::psi)
                       + (1.0/12.0)*state_fab(i, j - 2, k,
                        Idx::psi))/dx[1];
```

In this case, the domain dimension has been defined as $\mathrm{dim} = 2$, but up to $\mathrm{dim} = 3$ is supported. Here arbitrary orders of finite differencing are accessed by determining the finite difference coefficient matrix using the Lagrange polynomial expression for the specific stencil being considered.

Additionally, as a matter of convenience for the numerical relativist, libraries of functions have been written to automatically define symbolic versions of complex objects such as the Riemann tensor or Christoffel symbols in terms of a previously defined spatial metric. Assuming proper spatial derivatives have been previously defined the user may simply invoke various functions to perform the proper expansion of the symbolic expressions containing the metric derivatives, etc without having to write out expressions by hand.

Finally, the code generator implements symmetrization of tensors to avoid redundant definitions for symbolic tensor components. When defining the metric tensor and referencing specific components, the code generator automatically redefines the three redundant components (for a 3×3 symmetric tensor) of the tensor object in terms of their symmetric counterparts. When assigning the tensor object for AMReX expressions, only the independent components will be written. Future iterations of the code generator will also implement sub-expression elimination processes to further optimize generated code.

5. AwA tests and results

In order to validate the spacetime solver built by the code generator discussed in the previous section, we perform a selected set of tests from the standard Apples with Apples testbeds for Einstein solvers. To this end we perform tests designed to validate the accuracy and convergence of the solvers under refinements of the numerical grid.

We include the standard robust stability test to probe the constraint damping behavior in both 2nd and 4th order finite differencing. We also perform one dimensional linear wave (2nd order finite differencing) and gauge wave tests (4th order finite differencing) for both convergence testing and stability for long time evolutions. These tests are also useful in that they require no numerically constructed initial data, and instead involve exact initial data.

Finally, we perform black hole binary simulations for both head on collisions and binary inspirals for equal mass black holes. Head on collisions also involve exact initial data via the puncture approach. They also provide a straightforward method for testing convergence behavior in waveform extractions. Binary inspirals are more difficult to asses for accuracy, however it is useful to compare both the waveforms and black hole trajectories with other previously constructed spacetime solvers.

5.1. Robust stability test

The Robust stability test serves to diagnose the constrain violation behavior of a particular method of solving Einstein's equations. It has most notably served to characterize stability behavior for solvers based on the $3+1$ ADM, BSSN, Z4, etc. It is well known that the ADM solvers suffer from rapid undesirable growth of constraint violation that effectively renders solver based on this method useless. It has also been demonstrated that the BSSN methods develop moderate but bounded growth of constraint violations, and the Z4c methods result in decreasing constraint violations.

Here we simply wish to validate our solver based on code generation in the Z4c scheme with puncture gauge conditions in both 2nd order and 4th order finite differencing. To this end we expect to observe decreasing constraint violation behavior.

For this test we perform the simulations on a cell centered grid of $n_x = 64\rho$ points in the x direction. For 2nd order tests we use $n_y = n_z = 4\rho$, and for 4th order tests $n_y = n_z = 8\rho$. We perform the test on a domain of $x \in (-0.64, 0.64)$. For 2nd order tests y and $z \in (-0.04, 0.04)$. For 4th order we have y and $z \in (-0.08, 0.08)$. All 22 variables are initialized randomly $\varepsilon = (-10^{-10}/\rho^2, 10^{-10}/\rho^2)$. Evolution is performed with $\eta = 2$, $\kappa_1 = \kappa_2 = 0$, $\sigma_{\mathrm{KO}} = 0.1$. We perform the test for a CFL factor of 0.5 for 1000 light crossing times ($t_{\mathrm{final}} = 1280$) on the domain.

The constraint violations are monitored using the suggested diagnostic variable [18]:

$$C \equiv \sqrt{H^2 + \gamma_{ij} M^i M^j + \Theta^2 + 4\gamma_{ij} Z^i Z^j}, \tag{30}$$

where H and M^i are the Hamiltonian and Momentum constraints defined in (7).

The results for both 2nd and 4th order finite differencing are shown in figures 3 and 4 respectively. In each case we observe the expected constraint damping behavior of the diagnostic variable C.

5.2. Linear wave test

We test numerical stability and precision of our generated solver by evolving a linear wave as a perturbation on the Minkowski background. In this case a linear wave is an exact solution of the

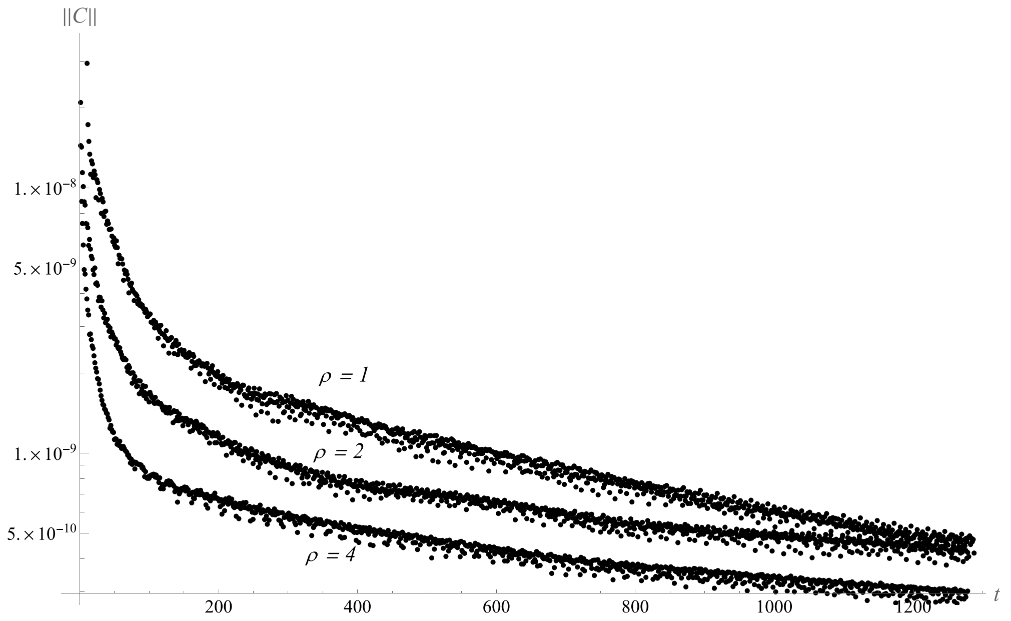

Figure 3. Here the norm of the constraint monitor $||C||$ is graphed for the robust stability test. Second order finite differencing is used for spatial derivatives in the evolution equations. Three values for ρ were considered for amplitudes $\varepsilon = (-10^{-10}/\rho^2, 10^{-10}/\rho^2)$.

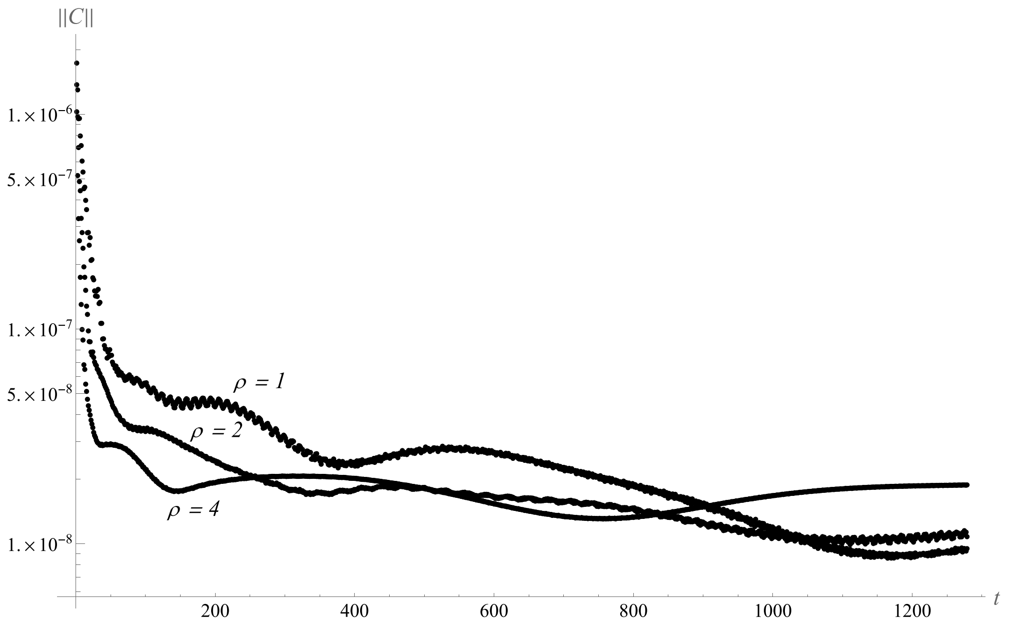

Figure 4. Here the norm of the constraint monitor $||C||$ is graphed for the robust stability test. Fourth order finite differencing is used for spatial derivatives in the evolution equations. Three values for ρ were considered for amplitudes $\varepsilon = (-10^{-8}/\rho^2, 10^{-8}/\rho^2)$.

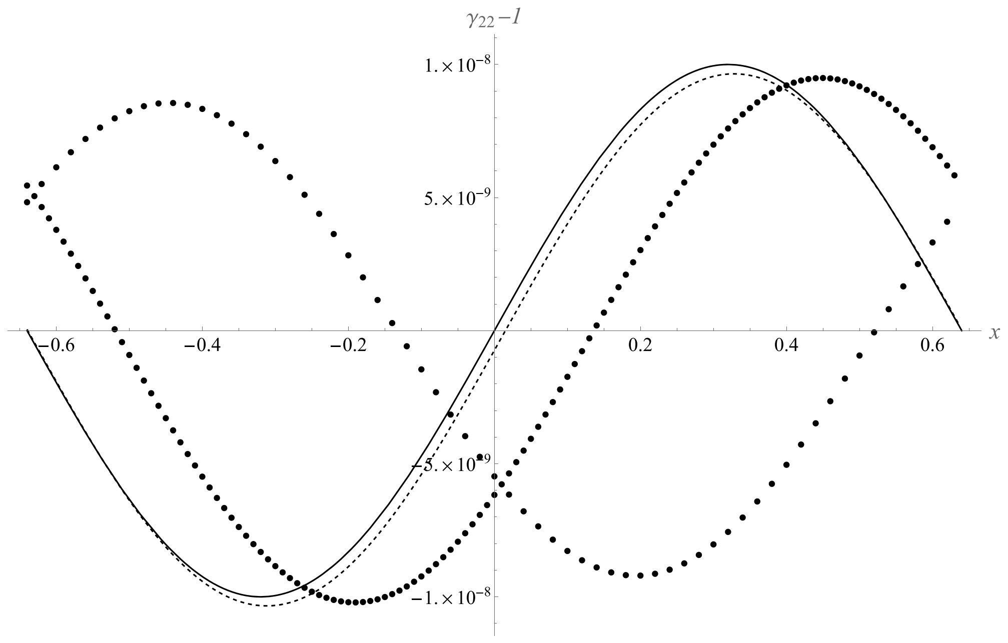

Figure 5. Comparison of wave forms for linear wave initial data at $t = 1000$ crossing times. Here we compare the wave forms for $\gamma_{11} - 1$. Resulting forms for $\rho = 1,2$ are shown (sparsely dotted, and dotted respectively). The resulting form for $\rho = 8$ is also shown (dashed line), along with the exact solution (solid line). Drifting of solutions for lower resolutions is observed, though higher resolutions show convergence to the exact case (see figure 6).

linearized Einstein equations which are valid for small amplitudes where interaction terms are below threshold for generating transitions between modes. Thus a linear wave of sufficiently small amplitude should maintain its structure up to numerical accuracy and machine precision.

To this end we use analytic initial data of the form:

$$\tilde{\gamma}_{xx} = 1, \qquad \gamma_{yy} = 1 + b, \qquad \gamma_{zz} = 1 - b \tag{31}$$

$$\alpha = 1, \qquad K_{yy} = \frac{1}{2}\partial_t b, \qquad K_{zz} = -\frac{1}{2}\partial_t b, \tag{32}$$

with

$$b = A\sin\left(\frac{2\pi\,(x-t)}{d}\right), \tag{33}$$

where $A = 10^{-8}$ and $d = 1.28$. The amplitude A is chosen such that non-linear terms in (16) are below machine precision and can thus be numerically neglected in the evolution.

The test is performed for 2nd order on a (node, cell, cell) centered grid with $n_x = 64\rho$, $n_y = n_z = 4\rho$ on the domain $x \in (-0.64, 0.64)$, and $y, z \in (-0.04, 0.04)$. We perform the test for $\rho = 1, 2, 4$, and 8. Here we choose $\eta = 2$, $\kappa_1 = 0.02$ $\kappa_2 = 0$, with dissipation factor $\sigma_{\mathrm{KO}} = 0.1$. We chose the Courant factor as $\lambda = 0.5$.

Figure 5 illustrates the γ_{yy} wave forms for a linear wave at $t = 1000$ crossing times. The convergence of solutions is apparent in these plots. The convergence of increasing resolutions

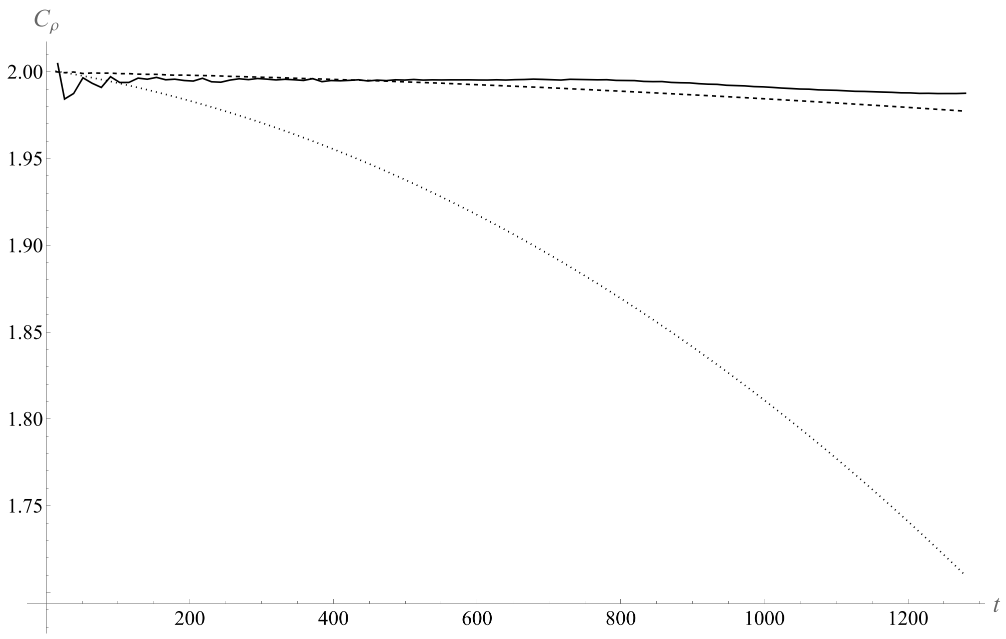

Figure 6. Convergence of linear wave forms for 2nd order finite differencing using the convergence formula in (34). Here we compare waveforms for $\rho = 1, 2, 4, 8$, and 16 using the formulas for $C_{\rho=4}$ (dotted), $C_{\rho=8}$ (dashed), and $C_{\rho=16}$ (solid).

is further illustrated in figure 6. Here the convergence is defined by comparing results at three different resolutions via the formula:

$$C_\rho \equiv \log \frac{||u_\rho - u_{\rho/2}||}{||u_{\rho/2} - u_{\rho/4}||}, \tag{34}$$

where the norm $||u||$ is defined as the sum of squares of the 22 evolved variables in (16) and (22):

$$||u||^2 = \sum_i u_i^2 = \phi^2 + \tilde{\gamma}_{ij}^2 + \hat{K}^2 + \tilde{A}_{ij} + \ldots \tag{35}$$

and the logarithm is base 2 or 4 depending on the order of finite differencing we consider. We will use this definition for both the linear wave and gauge wave tests below (with appropriate variables redefined when the CCZ4 formulation is used).

5.3. Gauge wave test

As a final quantitative test for our code generated solvers, we employ the gauge wave test with 4th order finite differencing. Gauge wave tests serve to illustrate several key characteristics of convergence of Einstein solvers. In particular, gauge wave tests allow one to probe the numerical stability and accuracy of the solver in a technically non-perturbative manor (arbitrary wave amplitude), by introducing an exact time dependent solution of Einstein's equations that are gauge equivalent to vacuum spacetime. Thus one is able to illustrate how well the solver maintains gauge invariance. Typically this is achieved by monitoring the Hamiltonian and momentum constraints (7) for long time evolutions.

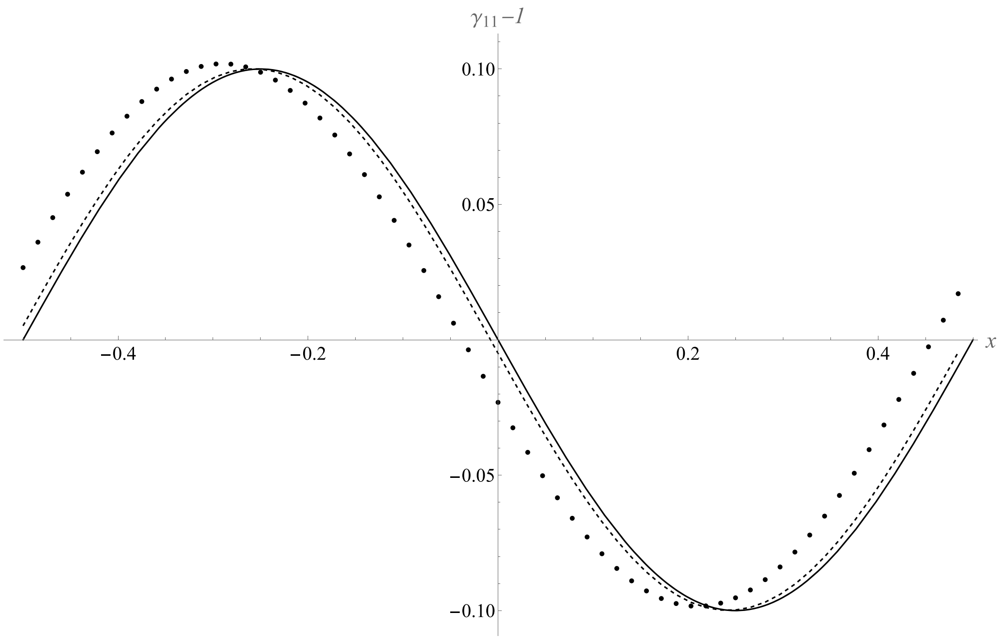

Figure 7. Comparison of gauge waves at $t = 1000$ crossing times, for $\rho = 4$ (dotted) and $\rho = 8$ (dashed). Here we compare $\gamma_{11} - 1$. The exact solutions is plotted (solid line). Evolution is done in CCZ4 formulation with harmonic lapse and zero shift.

For this test we initialize the variables according to an exact gauge wave:

$$\tilde{\gamma}_{xx} = 1 - b, \qquad \gamma_{yy} = 1, \qquad \gamma_{zz} = 1 \tag{36}$$

$$\alpha = \sqrt{1-b}, \qquad K_{xx} = \frac{\partial_t b}{2\sqrt{1+b}}, \qquad \hat{\Gamma}^x = -\frac{2\partial_x b}{3\,(1-b)^{5/3}}, \tag{37}$$

where b is defined as in (33) with $d = 1$. Here we choose an amplitude $A = 0.1$. Such an amplitude allows for non-linear terms in the equations of motion to affect the solution, and also allows for testing at 4th order finite differencing.

For this test we again use a (node, cell, cell)-centered grid, with $n_x = 64\rho$, $n_y = n_z = 4\rho$ on the domain $x \in (-0.5, 0.5)$, and $y, z \in (-0.03125, 0.03125)$. However, we choose to evolve the system in a CCZ4 scheme, with harmonic lapse condition ($\mu_L = 1$), and zero shift. Additionally, we set $\eta = 2$, $\kappa_1 = 1$, $\kappa_2 = 0$, and $\kappa_3 = 1$, with dissipation factor $\sigma_{\text{KO}} = 0.3$. We evolve with a CFL factor of $\lambda = 0.5$.

Figure 7 illustrates the convergence of wave forms at $t = 1000$ crossing times for increasing ρ to the exact solution. The waveform convergence as a function of time is shown quantitatively in figure 8. Finally, we also monitor the Hamiltonian constraint (7) as a function of time for all value of ρ considered. Shown in figure 9, is the integrated value of H as a function of time. The convergence of H at 4th order is shown in figure 10.

6. Black hole binary systems

As a final set of tests for the code generated spacetime solver, we wish to demonstrate numerical simulations of black hole binary mergers. Such tests serve both as interesting physical

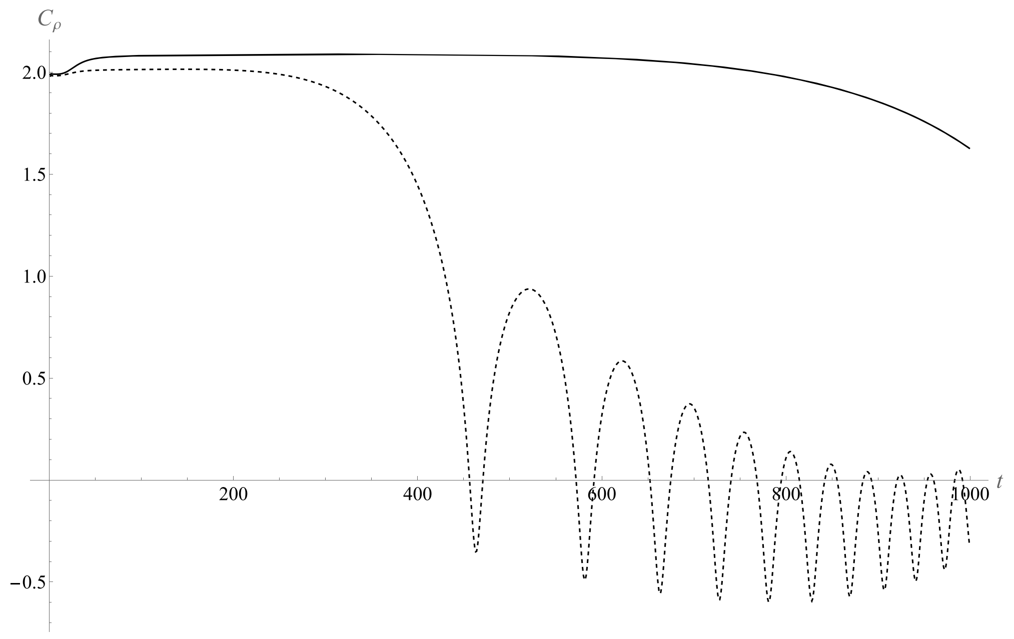

Figure 8. Convergence of waveforms for gauge waves with 4th order finite differencing. Comparisons are performed for $\rho = 8$, $\rho = 4$, $\rho = 2$, and $\rho = 1$, using the formulas for $C_{\rho=4}$ (dashed) and $C_{\rho=8}$ (solid) given in (34).

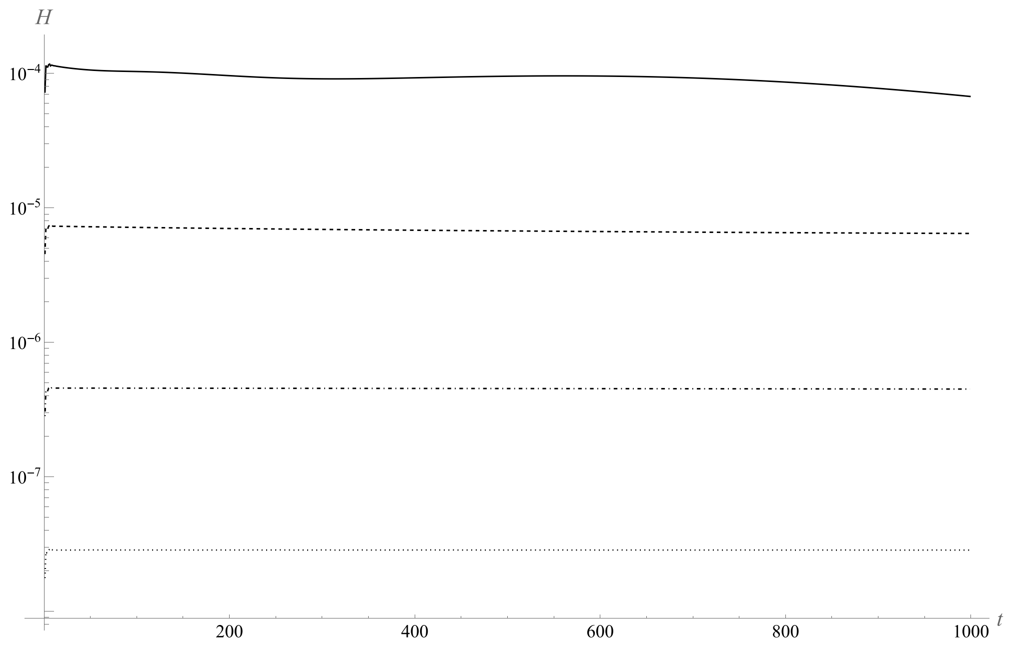

Figure 9. Shown is the H constraint (7) as a function of crossing times for $\rho = 1, 2, 4,$ and 8 (solid, dashed, dot-dashed, and dotted respectively) for gauge waves with 4th order finite differencing.

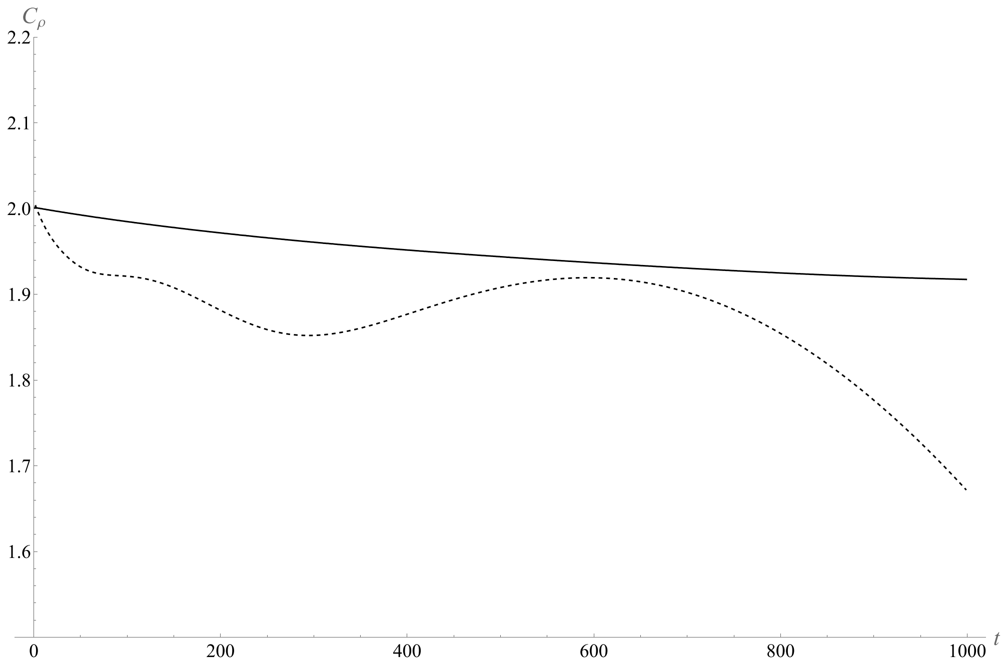

Figure 10. Convergence of H constraint for gauge waves with 4th order finite differencing. Here the convergence of H is performed with (34) using the single variable $u \equiv H$ for $C_{\rho=4}$ (dashed) and $C_{\rho=8}$ (solid).

systems, as well as a strong test of numerical stability for three spatial dimensions. We are also able to observe the behavior of physical simulations in the presence of FMR and AMR.

To perform these tests we will consider both a head on binary collision, as well as a binary inspiral. In both cases we observe the gravitational radiation via the Weyl scalar ψ_4 in the Newman–Penrose tetrad formulation for waveform extraction.

The initial data for black hole binaries is generated in a separate solver from the evolution system (16).

6.1. Head on collision

The goal of the equal mass black hole head on collision test is to quantitatively demonstrate the convergence behavior of the extracted wave forms for several resolutions. Here multiple resolutions can be tested easily since the solution of the initial data [19–21] is exact for puncture black holes initially at rest on the initial spatial hypersurface. In particular, it is clear from the appendix that an exact solution of the constraints is achieved by simply setting $A_{ij} = K = 0$ and $u = 0$ in (45). Thus, initial data for head on collisions is analytic, and no comparison of numerical initial data is necessary.

For the initial data the equation (47) is solved analytically with $u = 0$, and the definition of ξ given in (46). Here we place two equal mass black hole punctures at rest with bare masses $M_\pm = 0.5$ at coordinate positions $\vec{x}_\pm = (0, \pm 1.1515, 0)$.

In this case we use a cell centered numerical grid with $(n_x, n_y, n_z) = (64\rho, 64\rho, 64\rho)$ for $\rho = 1, 2, 4,$ and 8 at the coarsest level, on the domain $x, y, z \in (-128, 128)$. Here the spatial derivatives are approximated using 4th order finite differencing, however it should be noted that convergence will only be available to 2nd order as explained below. We perform the evolution

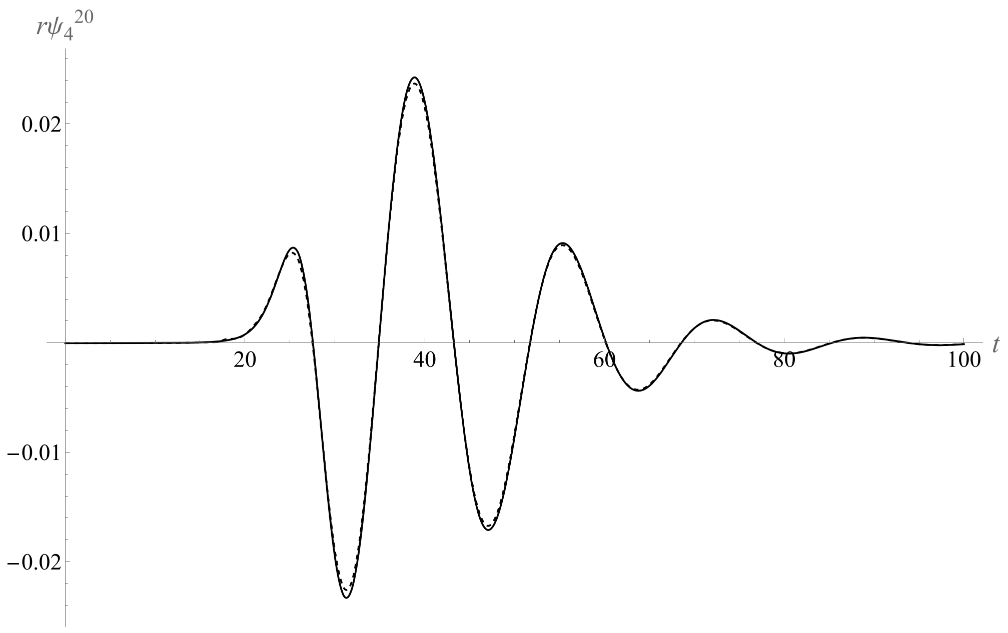

Figure 11. The ψ_4^{20} waveform for the head-on black hole collision at $r = 20$ is shown for both the lowest resolution $n = 64$ (dashed) and the highest resolution $n = 256$ (solid).

with six levels of FMR (with level $l = 0$ as the coarsest level) with a refinement factor of 2 at each level. Cells are tagged for refinement with the criterion $r < 64, 32, 16, 8, 4, 2$, for $r = (x^2 + y^2 + z^2)^{1/2}$ the standard radius on a Cartesian grid. We choose a blocking factor of 16 for $\rho = 1, 2$ and 32 for $\rho = 4, 8$. The max grid size is set to 32 cells. One buffer cell is used. Regridding takes place every ten timesteps. The systems is evolved with the Z4c scheme, with puncture gauge conditioning. For this case we set $\eta = 2$, $\kappa_1 = 0.02$, and $\kappa_2 = 0$. We choose a dissipation factor of $\sigma_{\text{KO}} = 0.1$. The Courant factor is chosen to be $\lambda = 0.5$.

To analyze the simulation we perform a waveform extraction at various coordinate radii in the near linear regime $r = 20, 30, 40, 50$. The wave form is constructed using the Newman–Penrose formalism (reviewed in appendix C), which is additionally generated using the code generator. Following the construction of the Weyl scalar ψ_4, we determine the $l = 2$, $m = 0$ spherical harmonic amplitude ψ_4^{20}. Since the data is native to Cartesian coordinates, we perform a trilinear interpolation to the sphere at the radius of extraction.

As stated previously, evolution is performed using 4th order spatial finite differencing. However, it can be shown that numerical interpolation and integration of cell centered Cartesian data on a spherical shell leads to an expected 2nd order convergence of the waveform for increasing resolution. This is indeed observed as shown in figures 11 and 12. The robustness of the wave form is shown in figure 13 showing the waveform at several times corresponding to the travel distance of the wave.

6.2. *Black hole inspiral*

The final test we wish to present is that of an equal mass black hole binary inspiral (see [22]) as one might expect to compare with observational data from gravitational wave interferometer observatories. This last test has the added requirement of numerically generated initial data, that is fed into the Z4c solver, which will then serve as a test of both solvers.

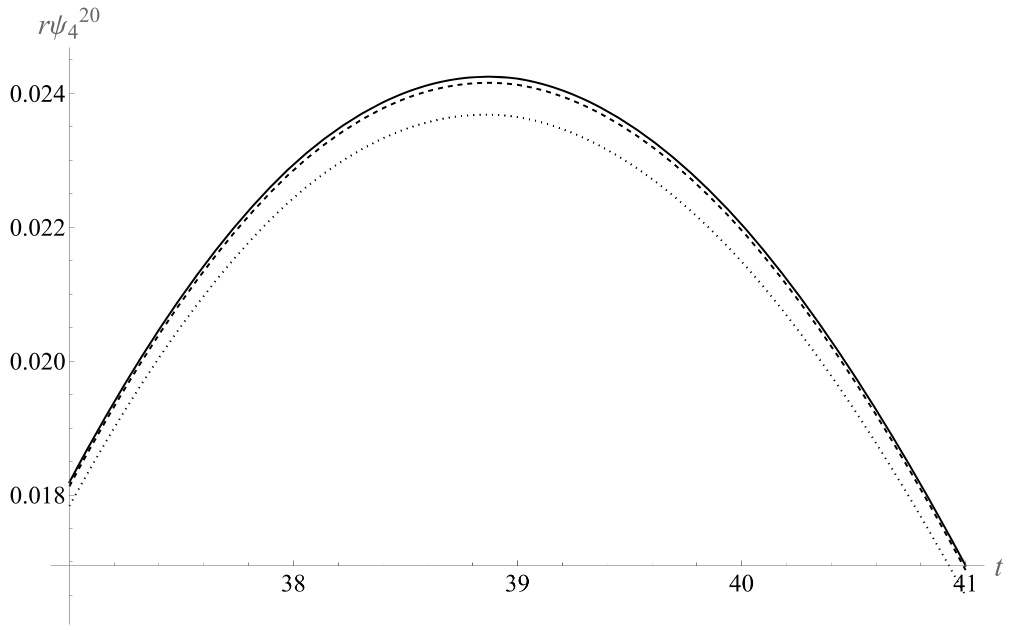

Figure 12. To illustrate the convergence of the waveform for a head on collision, we zoom in on the highest peak in figure 11. All three resolutions are shown; lowest (dotted), medium (dashed), and highest (solid).

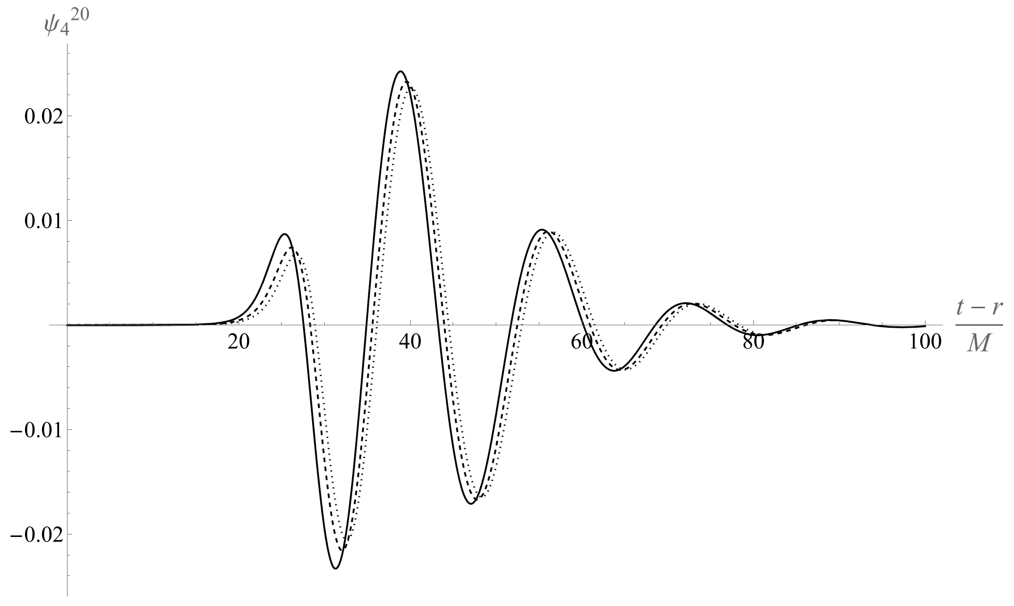

Figure 13. The robustness of the waveform is illustrated by plotting the waveform at 3 radii $r = 20$ (solid), $r = 30$ (dashed), and $r = 40$ (dotted). The $r = 30$ and $r = 40$ waveforms are plotted with a $\Delta t = 10$ and $\Delta t = 20$ delay respectively and overlaid to illustrate the evolution of the waveform.

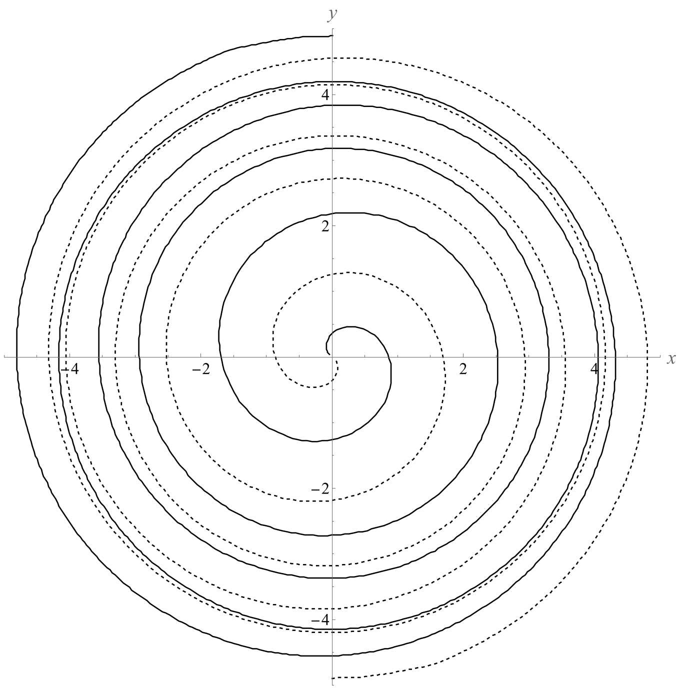

Figure 14. Blackhole binary trajectories. The black hole 'centers' are tracked by following the minimized lapse (solid and dashed lines denote each black hole center in the trajectory).

For the initial data we solve (47) using the solution for the momentum constraint (42), and the definition of ξ given in (46). Here we place two equal mass black hole punctures with bare masses $M_{\pm} = 0.4856$ at coordinate positions $\vec{x}_{\pm} = (0, \pm 4.891, 0)$. Here the punctures are given initial momenta $\vec{P}_{\pm} = (\mp 0.0969, 0, 0)$. The initial data system is solved using relaxation until the Hamiltonian constraint is satisfied to the order $H \lesssim 10^{-6}$ outside the black hole horizons.

Following the initial data solving procedure, we initialize the gauge variables as $\alpha = \psi^{-2}$, $\beta^i = 0$ as in Brill–Lindquist initial data. Additionally, we initially set $\phi = \log\psi$ per the definition, and $\tilde{\gamma}_{ij} = \delta_{ij}$. Additionally, we set $\hat{K} = \theta = \tilde{\Gamma}^i = 0$.

For this case we performed the simulation on a coarse grid of $(n_x, n_y, n_z) = (256, 256, 256)$, for a domain $x, y, z \in (-512, 512)$ with 4th order spatial finite differencing. In this case we use a mixed FMR/AMR grid with 8 levels of refinement, and a refinement factor of 2. The first 4 levels are refined on a fixed grid as in the head on collision case with $r < 256, 128, 64, 32$. The last 4 levels are tagged using AMR on the lapse $\alpha < 0.8, 0.7, 0.6, 0.5$. This type of mixed FMR/AMR allows for consistent interpolation and integration of waveforms to spherical shells at the radius of the extraction for some $r > 32$. Here the blocking factor is chosen to be 8 with a max grid size of 16 with one buffer cell. The AMR grid is updated after every 5 time steps in the RK4 evolution. We evolve the equations of motion in the Z4c formalism in the puncture gauge with$\eta = 0.25$, $\kappa_1 = 0.02$, $\kappa_2 = 0$, and a dissipation factor $\sigma_{\mathrm{KO}} = 0.1$. We choose a Courant factor of $\lambda = 0.1$.

Figure 14 illustrates the trajectory of the black holes as the simulation progresses. The results are similar to trajectories for equal mass binaries as obtained by previous analysis

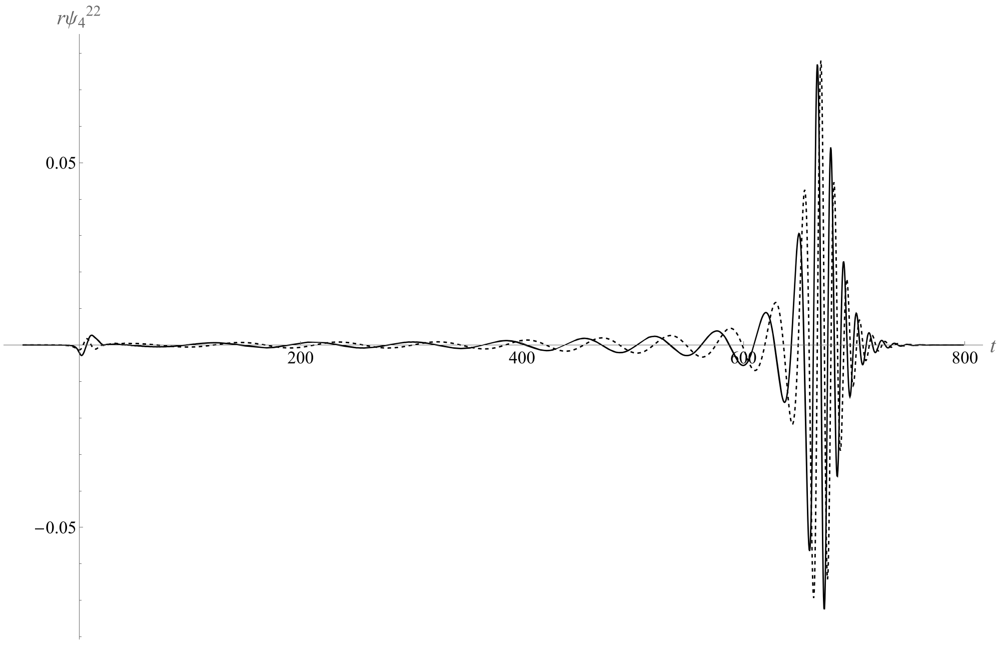

Figure 15. Blackhole binary $(l = 2,\ m = 2)$ waveforms for both the real (solid) and imaginary (dashed) parts of $r\psi_4^{22}$.

[23]. Note however, one should be aware that differences in gauge conditioning and evolution schemes can lead to quantitative differences in the plotted trajectories.

As in the previous case of the head on collision, the simulation is analyzed by extracting the wave forms at fixed radii in the linear regime. Here we extract the wave forms at coordinate radius $r = 50$ outside the FMR region (defined for $r > 32$, where regridding does not take place). In this case we calculate the $l = 2, m = 2$ spherical harmonic amplitude of the Weyl scalar ψ_4^{22}. Figure 15 illustrates the time dependent values of the spherical wave forms at the extraction radius.

7. Summary and discussion

To summarize our presentation in this paper, we have developed and demonstrated a code generation method for constructing complex PDE solvers for AMReX applications. The method of code generation significantly reduces the time spent analyzing and debugging large lines of code. Additionally code generation allows one to generate executable code for low level (fast) languages from symbolic tensor manipulation packages in higher level languages found in python and Mathematica applications.

We have demonstrated the application of code generation to the complex equations involved in a $3 + 1$ formulation of numerical relativity. In particular we have adapted our code generator to produce executables for AMReX applications of numerical relativity. We find that the generated systems of equations for Z4c and CCZ4 formulations produce accurate results when probed with the AwA tests for numerical relativity solvers. We thus conclude that the generated code is reliable and accurate, and may be used in more general contexts beyond the construction of numerical relativity solvers.

As an example of future applications, we intend to adapt the code generator to consider more complicated physical systems such as those of neutron star binaries, or core collapse supernovae. Such endeavors will require implementation of proper boundary conditions, such as Sommerfeld conditions for outward propagating waves, which will further require the use of code generation within AMReX. Such applications will further push the development of code generation as one considered hydrodynamics and magnetohydrodynamics in a dynamical spacetime. Additionally, code generation opens the door for consideration of more exotic spacetime solves such as those associated with time evolution in holographic spacetime problems such as those encountered in the AdS/CFT correspondence, or holographic superconductors.

Data availability statement

The data cannot be made publicly available upon publication because they are not available in a format that is sufficiently accessible or reusable by other researchers. The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgment

This research was funded by the ExaStar project within the Exascale Computing Project (17-SC-20-SC). Large scale computations were performed using the Cori supercomputer of the National Energy Research Scientific Computing Center.

A P is very thankful to Erik Schnetter, Vasilli Mewes, and Ann Almgren for useful discussions and advice for this project.

Appendix A. Review of Bowen York initial data

Here we review the method of generating black hole binary initial data as presented in [24] and [25]. The initial data are generated by ensuring that the constraints (7) are satisfied on an initial spatial hypersurface Σ of the spacetime manifold. Following the original conformal decomposition of the spatial metric

$$\gamma_{ij} = \psi^4 \tilde{\gamma}_{ij}, \tag{38}$$

where we switch from ϕ and χ to ψ as the conformal factor for convenience. We may then insert (38) into the Hamiltonian constraint (7):

$$8\tilde{D}^2\psi - \psi R - \psi^5 K^2 + \psi^{-7}\hat{A}^{ij}\hat{A}_{ij} = 0, \tag{39}$$

where we have introduced the conformal decomposition of $\hat{A}_{ij} \equiv \psi^2 A_{ij}$. Hear $\tilde{D}^2 = \tilde{\gamma}^{ij}\tilde{D}_i\tilde{D}_j$ is the covariant Laplacian with $\tilde{D}_i$ the covariant derivative compatible with the spatial conformal metric $\tilde{\gamma}_{ij}$.

With these definitions we rewrite the momentum constraint (7) as:

$$\tilde{D}_j\hat{A}^{ij} - \frac{2}{3}\psi^6\tilde{\gamma}_{ij}\tilde{D}_j K = 0. \tag{40}$$

Analytical solutions to equation (40) compatible with black hole spacetimes are well known. In particular we may chose the conformal transverse traceless approach with an initially flat conformal metric $\tilde{\gamma}_{ij} = \eta_{ij}$ with maximal slicing $K = 0$ (Bowen–York initial data). In this case one is left with

$$\partial_j \hat{A}^{ij} = 0 \tag{41}$$

which is linear, and thus possess simple solutions. Specifically, one may consider a single moving black hole with initial position C^i, and initial momentum P^i in Cartesian coordinates. Solutions possessing these characteristics may be written as:

$$\hat{A}^{ij}_{\mathbf{CP}} = \frac{2}{3r^2_{\mathbf{C}}}\left(P^i n^j_{\mathbf{C}} + P^j n^i_{\mathbf{C}} - \left(\eta^{ij} - n^i_{\mathbf{C}} n^j_{\mathbf{C}}\right) n^k_{\mathbf{C}} P_k\right). \tag{42}$$

The linearity of (41) allows one to add several solutions of (42) together. In particular one may select several values for **C** and **P** and write a general solution

$$\hat{A}^{ij} = \sum_n \hat{A}^{ij}_{\mathbf{C}_n \mathbf{P}_n}. \tag{43}$$

We then turn to the Hamiltonian constraint (39) with conformal flatness and maximal slicing:

$$\nabla^2 \psi = -\frac{1}{8}\psi^{-7}\hat{A}_{ij}\hat{A}^{ij}. \tag{44}$$

In the puncture approach [26] to solving (44) one absorbs the analytic singularities into ψ and considers corrections to ψ that are solved numerically. To this end for a black hole binary we write:

$$\psi = 1 + \frac{1}{\xi} + u, \tag{45}$$

for freely chosen bare masses $M_{1,2}$. Here ξ is defined as:

$$\xi \equiv \frac{M_1}{2r_{\mathbf{C}_1}} + \frac{M_2}{2r_{\mathbf{C}_2}}. \tag{46}$$

In this case the Hamiltonian constraint (44) reduces to:

$$\nabla^2 u = -\frac{1}{8}\hat{A}_{ij}\hat{A}^{ij}\left(\frac{\xi}{\xi\,(1+u)+1}\right)^7. \tag{47}$$

One may then employ standard methods of numerical solutions to elliptic equations to find u. For this particular project we employed a simple relaxation procedure where the initial guess for u was determined via the approximation described in [27].

The grid setup and FMR/AMR algorithm for the initial data is determined by the grid setup of the problem we intend to evolve. Grid and AMR inputs are chosen to match for both initial data and Z4c (or CCZ4) solver. This allows for easy copying from one solver to the next.

Appendix B. Conformal covariant Z4 (CCZ4) system

For certain tests and physical situations it is advantageous to consider an alternate to the Z4c formulation known as the CCZ4 formulation for evolution of the equations of motion. The CCZ4 is based on the same conformal decomposition of the original Z4 system as shown in (1), (3) and (4). The CCZ4 amounts to a rearrangement of terms in the Z4 system with additional adjustable parameters determining the conformal covariance of the evolution of the equations of motion. For particular tests such as the gauge wave test, it has been well documented that the constraint damped CCZ4 leads to improved stability of the evolution. However, the desirability of conformal covariance presents stability issues with black hole binary evolutions. In these cases the conformal covariance of the equations is sacrificed for numerical stability.

We present the final equations of the CCZ4 system with the variable definitions given in section 2. We however adjust our definition of the conformal factor in the spatial metric as:

$$\tilde{\gamma}_{ij} = W^2\gamma_{ij}, \tag{48}$$

and thus we have a redefinition of the conformal variable $W = \exp(-2\phi)$. See [13] and [14] for details and justification for this redefinition.

With this redefinition the CCZ4 system decomposes as:

$$\partial_t W = \frac{1}{3}\alpha W K + \beta^i\partial_i W - \frac{1}{3}W\partial_i\beta^i \tag{49}$$

$$\partial_t\tilde{\gamma}_{ij} = -2\alpha\tilde{A}_{ij} + 2\tilde{\gamma}_{k(i}\partial_{j)}\beta^k - \frac{2}{3}\tilde{\gamma}_{ij}\partial_k\beta^k + \beta^k\partial_k\tilde{\gamma}_{ij} \tag{50}$$

$$\begin{aligned}\partial_t K = &-D_iD^i\alpha + \alpha\left(R + 2D_iZ^i + K^2 - 2\Theta K\right) - 3\alpha\kappa_1\left(1+\kappa_2\right)\Theta \\ &+ 4\pi\alpha\left(S - 3\rho_{\mathrm{ADM}}\right) + \beta^i\partial_i K\end{aligned} \tag{51}$$

$$\begin{aligned}\partial_t\tilde{A}_{ij} = &W^2\left[-D_iD_j\alpha + \alpha\left(R_{ij} + 2D_{(i}Z_{j)} - 8\pi S_{ij}\right)\right]^{\mathrm{tf}} + \alpha\left[(K-2\Theta)\tilde{A}_{ij}\right. \\ &\left.-2\tilde{A}_{ik}\tilde{A}^k_j\right] + 2\tilde{A}_{k(i}\partial_{j)}\beta^k - \frac{2}{3}\tilde{A}_{ij}\partial_k\beta^k + \beta^k\partial_k\tilde{A}_{ij}\end{aligned} \tag{52}$$

$$\begin{aligned}\partial_t\Theta = &\frac{1}{2}\alpha\left[R + 2D_iZ^i - \tilde{A}_{ij}\tilde{A}^{ij} + \frac{2}{3}K^2 - 2\Theta K - 16\pi\rho_{\mathrm{ADM}}\right. \\ &\left.-2\kappa_1\left(2+\kappa_2\right)\Theta\right] - Z^i\partial_i\alpha + \beta^i\partial_i\Theta\end{aligned} \tag{53}$$

$$\begin{aligned}\partial_t\tilde{\Gamma}^i = &\tilde{\gamma}^{jk}\partial_j\partial_k\beta^i + \frac{1}{3}\tilde{\gamma}^{ij}\partial_j\partial_k\beta^k - 2\tilde{A}^{ij}\partial_j\alpha + 2\alpha\left[\tilde{\Gamma}^i_{jk}\tilde{A}^{jk} - 3\tilde{A}^{ij}\frac{\partial_j W}{W}\right. \\ &\left.-\frac{2}{3}\tilde{\gamma}^{ij}\partial_j K - 8\pi\tilde{\gamma}^{ij}S_j\right] + 2\tilde{\gamma}^{ik}\left[\alpha\partial_k\Theta - \Theta\partial_k\alpha - \frac{2}{3}\alpha K Z_k\right] \\ &+\frac{2}{3}\tilde{\Gamma}^i_{\mathrm{d}}\partial_j\beta^j - \tilde{\Gamma}^j_{\mathrm{d}}\partial_j\beta^i + 2\kappa_3\left[\frac{2}{3}\tilde{\gamma}^{ij}Z_j\partial_k\beta^k - \tilde{\gamma}^{jk}Z_j\partial_k\beta^i\right] \\ &-2\alpha\kappa_1\tilde{\gamma}^{ij}Z_j + \beta^j\partial_j\tilde{\Gamma}^i.\end{aligned} \tag{54}$$

Here the Ricci tensor R_{ij} of γ_{ij} can be decomposed into a conformal and the Ricci tensor $\tilde{R}_{ij}$ associated with $\tilde{\gamma}_{ij}$. Additionally, the constraints associated with Z_i are absorbed into the definition of $\tilde{R}_{ij}$:

$$R_{ij} = R^W_{ij} + \tilde{R}_{ij} \tag{55}$$

$$R^W_{ij} = \frac{1}{W^2}\left[W\left(\tilde{D}_i\tilde{D}_jW + \tilde{\gamma}_{ij}\tilde{D}_l\tilde{D}^lW\right) - 2\tilde{\gamma}_{ij}\tilde{D}^lW\tilde{D}_lW\right] \tag{56}$$

$$\tilde{R}_{ij} = -\frac{1}{2}\tilde{\gamma}^{lm}\partial_i\partial_j\tilde{\gamma}_{lm} + \tilde{\gamma}_{k(i}\partial_{j)}\tilde{\Gamma}^k_{\mathrm{d}} + \tilde{\Gamma}^k_{\mathrm{d}}\tilde{\Gamma}_{(ij)k} + \tilde{\gamma}^{lm}\left(2\tilde{\Gamma}^k_{l(i}\tilde{\Gamma}_{j)km} + \tilde{\Gamma}^k_{im}\tilde{\Gamma}_{klj}\right). \tag{57}$$

Note the slight difference in definition of $\tilde{R}_{ij}$ in the second term with that in the Z4c formulation in (19). As in the Z4c formulation we enforce the algebraic constraints as in (20).

Here the puncture gauge conditions for the lapse α, and shift β^i take the form:

$$\partial_t\alpha = -\mu_{\rm L}\alpha^2\left(K-2\Theta\right)+\beta^i\,\partial_i\,\alpha \tag{58}$$

$$\partial_t\beta^i = \mu_{\rm S}\alpha^2\tilde{\Gamma}^i - \eta\beta^i + \beta^j\partial_j\beta^i. \tag{59}$$

Appendix C. Newman–Penrose waveform extraction

To extract the gravitational waveform of a black hole collision, we adopt the Newman–Penrose formulation (see [28, 28–32]). Here the ten components of the Weyl tensor ${}^{(4)}C_{\mu\nu\alpha\beta}$ is decomposed into five complex valued scalars constructed from contractions with a null tetrad $(l^\alpha, k^\alpha, m^\alpha, \bar{m}^\alpha)$. Considering quasi-Kinnersly tetrads it is possible to show that the Weyl scalar ψ_4 defined as

$$\psi_4 \equiv -{}^{(4)}C_{\mu\nu\alpha\beta}k^\mu\bar{m}^\nu k^\alpha m^\beta, \tag{60}$$

has the property that as $r\to\infty$,

$$\psi_4 \to \partial_t^2 h_+ - i\partial_t^2 h_\times, \tag{61}$$

where h_+ and $h_\times$ are the two independent transverse modes of the metric perturbation.

Performing the $3+1$ decomposition it can be shown:

$$\begin{aligned}\psi_4 = &\left(R_{ijkl}+2K_{i[k}K_{l]j}\right)n^i\bar{m}^j n^k\bar{m}^l - 8\left(K_{j[k,l]}+\Gamma^p_{j[k}K_{l]p}\right)n^{[0}\bar{m}^{j]}n^k\bar{m}^l\\ &+4\left(R_{jl}-K_{jp}K^p_l+KK_{jl}\right)n^{[0}\bar{m}^{j]}n^{[0}\bar{m}^{l]}.\end{aligned} \tag{62}$$

Following the general expression for the Weyl scalar in (62), we implement the quasi-Kinnersley tetrad as constructed in [Baker, Campanelli, Lousto]. In this procedure an orthogonal set of spatial vectors aligned with the $\hat{(\varphi)}$ and $\hat{(r)}$ directions is selected, along with a third vector constructed from the cross product. These are written in Cartesian coordinates as:

$$\begin{aligned}v_1^a &= [-y, x, 0],\\ v_2^a &= [x, y, z],\\ v_3^a &= \det(\gamma)^{1/2}\gamma^{ad}\epsilon_{dbc}v_1^b v_2^c,\end{aligned} \tag{63}$$

where ϵ_{abc} is the standard Levi-Cevita symbol with $\epsilon_{123}=1$.

An orthonormal basis is then constructed via the Gram–Schmidt procedure. Specifically, orthonormal vectors v_1^a, v_2^a, and v_3^a are manipulated one after the other as follows:

$$\begin{aligned}v_1^a &\to \frac{v_1^a}{\sqrt{\omega_{11}}},\\ v_2^a &\to \frac{v_2^a-\omega_{12}}{\sqrt{\omega_{22}}},\\ v_3^a &\to \frac{v_3^a-\omega_{13}v_1^a-\omega_{23}v_2^a}{\sqrt{\omega_{33}}},\end{aligned} \tag{64}$$

where $\omega_{ij}\equiv v_i^a v_j^b\gamma_{ab}$. Note that for each v_i^a in (64), the numerator is evaluated first, and then the denominator with newly constructed value of v_i^a.

The tetrad is now constructed using the orthonormal basis (64) along with the time-like unit normal u^μ, which in the $3+1$ decomposition takes the form:

$$u^\mu = \frac{1}{\alpha}\left(1, -\beta^i\right). \tag{65}$$

The null tetrad is then constructed as:

$$\begin{aligned} l^\mu &= \frac{1}{\sqrt{2}}\left(u^\mu + r^\mu\right) \\ n^\mu &= \frac{1}{\sqrt{2}}\left(u^\mu - r^\mu\right) \\ m^\mu &= \frac{1}{\sqrt{2}}\left(\theta^\mu + i\varphi^\mu\right). \end{aligned} \tag{66}$$

The Weyl scalar transforms as a spherical tensor with spin weight -2 and is thus decomposed with the spin weighted spherical harmonics Y_{lm}^{-2}:

$$\psi_4\left(t,r,\theta,\varphi\right) = \sum_{l=2}^{\infty}\sum_{m=-l}^{m=l}\psi_4^{lm}\left(t,r\right)Y_{lm}^{-2}\left(\theta,\varphi\right). \tag{67}$$

with spherical amplitude:

$$\psi_4^{lm} = \int \mathrm{d}\Omega\, Y_{lm}^{-2\star}\psi_4. \tag{68}$$

The spin-weighted spherical harmonics are defined as:

$$Y_{lm}^{-2} \equiv \sqrt{\frac{(l-2)!}{(l+2)!}}\left(W_{lm}\left(\theta,\varphi\right) - i\frac{X_{lm}\left(\theta,\varphi\right)}{\sin\theta}\right), \tag{69}$$

with $W_{lm}(\theta,\varphi)$ and $X_{lm}(\theta,\varphi)$ defined as:

$$\begin{aligned} W_{lm}\left(\theta,\varphi\right) &= \left(\partial_\theta^2 - \cot\theta\partial_\theta - \frac{\partial_\varphi^2}{\sin^2\theta}\right)Y_{lm}\left(\theta,\varphi\right), \\ X_{lm} &= 2\partial_\varphi\left(\partial_\theta - \cot\theta\right)Y_{lm}\left(\theta,\varphi\right), \end{aligned} \tag{70}$$

and the standard definition of the spherical harmonics:

$$Y_{lm}\left(\theta,\varphi\right) = \sqrt{\frac{2l+1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}P_m^l\left(\cos\theta\right)e^{im\varphi}, \tag{71}$$

where P_m^l are the associated Legendre polynomials.

For the cases we consider for head on and inspiraling black hole binary collisions we explicitly use the $s=-2$ weighted spherical harmonics:

$$\begin{aligned} Y_{20}^{-2} &= \frac{3}{4}\sqrt{\frac{5}{6\pi}}\sin^2\theta, \\ Y_{22}^{-2} &= \frac{1}{8}\sqrt{\frac{5}{\pi}}\left(1+\cos\theta\right)^2 e^{2i\pi}. \end{aligned} \tag{72}$$

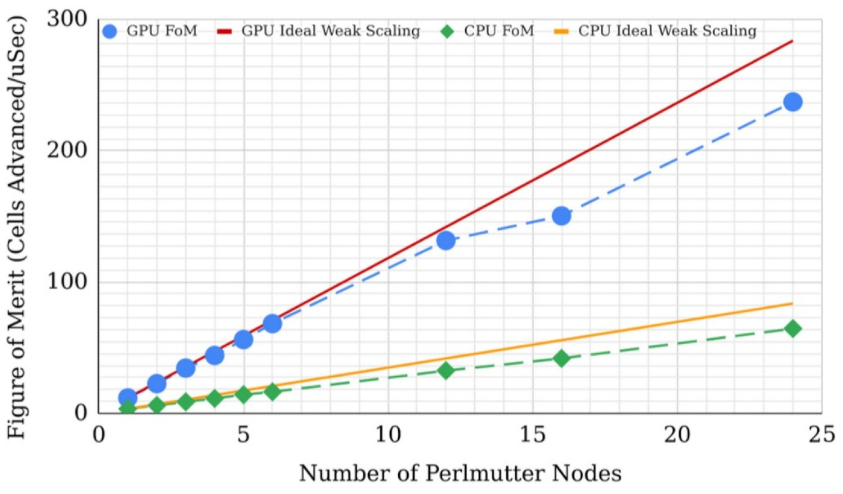

Figure 16. Weak scaling performance of our Z4c code-generated solver with adaptive mesh refinement on the NERSC Perlmutter machine running a black hole collision test problem. Shown is the code's figure of merit, the number of cells advanced per microsecond of simulation walltime. Two weak scaling series are shown in our plot, representing CPU (green diamond) and GPU (blue circles) results along with their ideal weak scaling curves extrapolated from the measured single-node figure of merit (orange and red lines, respectively).

Appendix D. Performance testing results

To assess our code performance on modern supercomputing architectures, we evaluated the weak scaling performance of our Z4c code-generated solver with AMR on the NERSC Perlmutter machine running a black hole collision test problem. The test problem is distributed with the code on github, for reference. We measure a quantity we refer to as the code's figure of merit, the number of cells advanced per microsecond of simulation walltime. Two weak scaling series are shown in our plot, representing CPU (green diamond) and GPU (blue circles) results along with their ideal weak scaling curves extrapolated from the measured single-node figure of merit (orange and red lines, respectively). We measured weak scaling from 1 to 24 nodes on Perlmutter, using the dedicated CPU nodes (2x AMD EPYC 7763) and GPU nodes (1x AMD EPYC 7763 + 4x NVIDIA A100) for the respective tests. For the CPU test, we allocate 1 MPI rank per core, while for the GPU test, we allocate 1 MPI rank per GPU.

As shown in figure 16 We see reasonable weak scaling for this problem, with deviations from the ideal scaling curves around 64 GPUs. We note that the black holes are moving across the domain, and we are using nine total levels of refinement. In this test case, the grid refinement boundaries are likewise moving, resulting in additional communication that would not be present in a static refinement problem, so this is a stress test for scaling performance in a case where regridding changes the grid layout and communication pattern among MPI ranks (and thus, GPUs) dynamically. We ascribe qualitative differences between CPU and GPU scaling to the addition of communication between host and device memory in the latter case. Nevertheless, we have shown that we can effectively get a good speedup from GPUs on a GPU-based supercomputer with 3-4x speedup on dedicated CPU and GPU nodes. We note that if we were to run the CPU-scaling on the GPU nodes instead, there would be only 1 AMD EPYC 7763 per node instead of 2 on the dedicated CPU nodes, likely resulting in an additional factor of 2x GPU speedup over the speedup we show above. To show performance in the mode users are likely to actually use NERSC Perlmutter however, we have chosen to show results using the dedicated CPU and GPU nodes.

ORCID iD

Adam J Peterson https://orcid.org/0000-0003-1121-6782

References

[1] Abramovici A *et al* 1992 *Science* **256** 325–33
[2] Caron B *et al* 1997 *Class. Quantum Grav.* **14** 1461–9
[3] Abbott B P *et al* (LIGO Scientific and Virgo) 2016 *Phys. Rev. Lett.* **116** 061102
[4] Baker J G, Centrella J, Choi D I, Koppitz M and van Meter J 2006 *Phys. Rev. Lett.* **96** 111102
[5] Campanelli M, Lousto C O, Marronetti P and Zlochower Y 2006 *Phys. Rev. Lett.* **96** 111101
[6] Bona C, Ledvinka T, Palenzuela C and Zacek M 2003 *Phys. Rev.* D **67** 104005
[7] Gundlach C, Martin-Garcia J M, Calabrese G and Hinder I 2005 *Class. Quantum Grav.* **22** 3767–74
[8] Bernuzzi S and Hilditch D 2010 *Phys. Rev.* D **81** 084003
[9] Gundlach C and Martin-Garcia J M 2006 *Phys. Rev.* D **74** 024016
[10] Babiuc M C *et al* 2008 *Class. Quantum Grav.* **25** 125012
[11] Husa S, Hinder I and Lechner C 2006 *Comput. Phys. Commun.* **174** 983–1004
[12] Ruchlin I, Etienne Z B and Baumgarte T W 2018 *Phys. Rev.* D **97** 064036
[13] Marronetti P, Tichy W, Bruegmann B, Gonzalez J and Sperhake U 2008 *Phys. Rev.* D **77** 064010
[14] Cao Z, Yo H J and Yu J P 2008 *Phys. Rev.* D **78** 124011
[15] Bona C, Masso J, Seidel E and Stela J 1995 *Phys. Rev. Lett.* **75** 600–3
[16] Alcubierre M, Bruegmann B, Diener P, Koppitz M, Pollney D, Seidel E and Takahashi R 2003 *Phys. Rev.* D **67** 084023
[17] McCorquodale P and Colella P 2011 *Commun. Appl. Math. Comput. Sci.* **6** 1–25
[18] Cao Z and Hilditch D 2012 *Phys. Rev.* D **85** 124032
[19] Brill D R and Lindquist R W 1963 *Phys. Rev.* **131** 471–6
[20] Brandt S R and Bruegmann B 1997 arXiv:gr-qc/9711015 [gr-qc]
[21] Okawa H 2013 *Int. J. Mod. Phys.* A **28** 1340016
[22] Hilditch D, Bernuzzi S, Thierfelder M, Cao Z, Tichy W and Bruegmann B 2013 *Phys. Rev.* D **88** 084057
[23] Etienne Z B, Liu Y T, Shapiro S L and Baumgarte T W 2009 *Phys. Rev.* D **79** 044024
[24] Bowen J M and York J W Jr 1980 *Phys. Rev.* D **21** 2047–56
[25] Kulkarni A, Shepley L and York J W Jr 1983 *Phys. Lett.* A **96** 228–30
[26] Brandt S and Bruegmann B 1997 *Phys. Rev. Lett.* **78** 3606–9
[27] Dennison K A, Baumgarte T W and Pfeiffer H P 2006 *Phys. Rev.* D **74** 064016
[28] Campanelli M, Krivan W and Lousto C O 1998 *Phys. Rev.* D **58** 024016
[29] Campanelli M, Lousto C O, Baker J G, Khanna G and Pullin J 1998 *Phys. Rev.* D **58** 084019 Campanelli M, Lousto C O, Baker J G, Khanna G and Pullin J 2000 *Phys. Rev.* D **62** 069901 (erratum)
[30] Campanelli M and Lousto C O 1999 *Phys. Rev.* D **59** 124022
[31] Baker J G, Campanelli M and Lousto C O 2002 *Phys. Rev.* D **65** 044001
[32] Campanelli M, Kelly B J and Lousto C O 2006 *Phys. Rev.* D **73** 064005