The Astrophysical Journal Supplement Series, 278:50 (12pp), 2025 June https://doi.org/10.3847/1538-4365/adcf96

Performance-portable Numerical Relativity with AthenaK

Hengrui Zhu (朱恒锐)[1,2], Jacob Fields[3,4], Francesco Zappa[5], David Radice[3,4,6,8], James M. Stone[7], Alireza Rashti[3,4], William Cook[5], Sebastiano Bernuzzi[5], and Boris Daszuta[5]
[1] Department of Physics, Princeton University, Jadwin Hall, Washington Road, NJ 08544, USA; hz0693@princeton.edu
[2] Princeton Gravity Initiative, Princeton University, Princeton, NJ 08544, USA
[3] Department of Physics, The Pennsylvania State University, University Park, PA 16802, USA
[4] Institute for Gravitation & the Cosmos, The Pennsylvania State University, University Park, PA 16802, USA
[5] Theoretisch-Physikalisches Institut, Friedrich-Schiller-Universität Jena, 07743, Jena, Germany
[6] Department of Astronomy & Astrophysics, The Pennsylvania State University, University Park, PA 16802, USA
[7] School of Natural Sciences, Institute for Advanced Study, 1 Einstein Drive, Princeton, NJ 08540, USA
Received 2024 September 25; revised 2025 April 21; accepted 2025 April 21; published 2025 June 6

Abstract

We present the numerical relativity module within AthenaK, an open-source performance-portable astrophysics code designed for exascale computing applications. This module employs the Z4c formulation to solve the Einstein equations. We demonstrate its accuracy through a series of standard numerical relativity tests, including convergence of the gravitational waveform from binary black hole coalescence. Furthermore, we conduct scaling tests on OLCF Frontier, NERSC Perlmutter, and ALCF Aurora, where AthenaK exhibits excellent weak-scaling efficiency of 80% on up to 65,536 AMD MI250X GPUs on Frontier (relative to four GPUs) and 67% on Aurora up to 24,576 Intel Data Center Max Series GPUs (relative to 12 GPUs) and strong-scaling efficiencies of 84% and 77% on AMD MI250X and NVIDIA A100 GPUs on Frontier and Perlmutter, respectively. Additionally, we observe a significant performance boost, with 2 orders of magnitude speedup ($\gtrsim$200×) on a GPU compared to a single CPU core, affirming that AthenaK is well suited for exascale computing, and thereby expanding the potential for breakthroughs in numerical relativity research.

Unified Astronomy Thesaurus concepts: Compact objects (288); Black holes (162); Gravitational waves (678); Gravitational wave sources (677)

1. Introduction

The inherent nonlinearity of the Einstein equations necessitates the use of numerical methods to generate dynamical solutions, a field known as numerical relativity (NR; T. W. Baumgarte & S. L. Shapiro 1999, 2010; M. A. Miller 2000; L. Lehner 2001; L. Lehner & F. Pretorius 2014; M. D. Duez & Y. Zlochower 2019; D. Hilditch 2024). This approach has been crucial for modeling gravitational waveforms from binary compact object mergers (M. Shibata & K. Uryu 2000; F. Pretorius 2005; M. Campanelli et al. 2006; J. G. Baker et al. 2007), particularly in light of the increasing detections from the LIGO, Virgo, and KAGRA collaborations (LVK; B. P. Abbott et al. 2016a, 2016b, 2017a; T. Akutsu et al. 2021). With LVK's ongoing sensitivity enhancements (B. P. Abbott et al. 2020) and the upcoming next-generation detectors (M. Punturo et al. 2010; B. P. Abbott et al. 2017b; P. Amaro-Seoane et al. 2017; D. Reitze et al. 2019), there is a pressing need for more accurate gravitational waveform catalogs that cover larger parameter spaces, including eccentric, precessing, and high-mass-ratio black hole binaries (C. O. Lousto et al. 2010; H. Nakano et al. 2011; G. Lovelace 2021).

One of the central challenges in NR is resolving the wide range of spatial and temporal scales present in simulations. To handle this, NR codes often employ adaptive mesh refinement (AMR). A widely used implementation is the Carpet thorn (E. Schnetter et al. 2004) on top of the Cactus framework (T. Goodale et al. 2003), which leverages the Berger–Oliger method (M. J. Berger & J. Oliger 1984; M. J. Berger & P. Colella 1989), where nested patches of grid with increasing resolution are used to resolve the binary constituents and track their evolution in time. NR codes building on top of this framework include Llama (D. Pollney et al. 2011; C. Reisswig et al. 2013), McLachlan (D. Brown et al. 2009), and LEAN (U. Sperhake 2007). Other NR codes, including BAM (B. Brügmann et al. 2008), AMSS-NCKU (Z. Cao et al. 2008), and GRChombo (K. Clough et al. 2015), also use the Berger–Oliger AMR approach.

This approach, however, incurs heavy performance penalties due to the overhead for synchronization of data between patches, more complicated bookkeeping, memory locality, and other factors, thereby decreasing the scaling performance on modern high-performance computing systems (Q. F. Stout et al. 1997). This leads to the adoption of block-based octree AMR in codes like Dendro-GR (M. Fernando et al. 2019, 2023) and GR-Athena++ (B. Daszuta et al. 2021, 2024; B. Daszuta & W. Cook 2024; W. Cook et al. 2025). Both of these codes have demonstrated excellent scaling on large CPU clusters.

All of the aforementioned codes use finite differencing with Cartesian coordinates. Other methods include pseudospectral techniques, exemplified by SpEC (B. Szilagyi et al. 2009), where the domain is decomposed into patches of topological spheres and cylinders. Another example is the bamps code, which uses the pseudospectral method for spacetime variables but discontinuous Galerkin (DG) for matter (M. Bugner et al. 2016; D. Hilditch et al. 2016). A novel hybrid finite-difference and DG method is used in SpECTRE (L. E. Kidder et al. 2017;

[8] Alfred P. Sloan Fellow.

N. Deppe et al. 2024) and Nmesh (W. Tichy et al. 2023). For the Einstein equations, pseudospectral, DG, and finite-difference methods may all be expressed as summation-by-parts penalty methods (J. M. Miller & E. Schnetter 2017). A more recent innovation aims to eliminate the need for AMR by generating problem-specific curvilinear grids, as seen in SENR/NRPy+ (V. Mewes et al. 2018, 2020; I. Ruchlin et al. 2018).

The emergence of exascale computing platforms provides unprecedented amounts of computational power, but to fully harness this, NR codes must efficiently run on GPU architectures and scale well across thousands of nodes. Most existing codes are limited by either their reliance on CPU-based architectures or their poor scaling on large clusters. As a result, next-generation NR codes capable of operating on exascale systems are essential to further scientific progress.

To address these challenges, we introduce AthenaK, a performance-portable extension of the Athena++ astrophysics code (C. J. White et al. 2016; K. G. Felker & J. M. Stone 2018; J. M. Stone et al. 2020), developed using the Kokkos library (C. R. Trott et al. 2022). Kokkos abstracts machine-specific dependencies, allowing AthenaK to fully utilize the computational power of exascale clusters for a broad range of astrophysical simulations.

In this paper, we describe the implementation of the vacuum Einstein equations in Z4c formulation within AthenaK, using octree AMR. We demonstrate the accuracy of the code through a series of tests, including the convergence of gravitational waveforms from binary black hole coalescence. Additionally, we introduce new AMR criteria that enable the simulation of binary black holes with large mass ratios. Strong- and weak-scaling results from the Frontier supercomputers show AthenaK achieves an 80% weak-scaling efficiency on 65,536 GPUs, providing a clear path toward exascale NR. This work is part of a three-paper series describing AthenaK's key functionalities (J. M. Stone et al. 2024; J. Fields et al. 2025), available on GitHub.[9]

AthenaK is not the only, nor the first, effort to leverage GPUs for NR applications. AsterX and GRaM-X are notable other examples (S. Shankar et al. 2023; J. V. Kalinani et al. 2025). Both of these codes solve the GRMHD equations coupled with the Einstein equations in the Z4c formulation, using the CarpetX AMR driver (E. Schnetter et al. 2004) within the Einstein Toolkit (S. R. Brandt et al. 2024), which is built on AMReX (W. Zhang et al. 2019, 2021). However, the convergence and waveform accuracy of the Z4c solvers have yet to be demonstrated. Additionally, Dendro-GR offers a GPU extension as presented in M. Fernando et al. (2022), but scaling results have been limited to 16 GPUs. The code is primarily written in CUDA, which restricts its compatibility to NVIDIA systems, although there is preliminary support for AMD GPUs. A similar effort to AthenaK is ongoing to develop an NR module in the Phoebus code (Phoebus 2022), which is based on the Parthenon framework (P. Grete et al. 2022).

In this paper, we show that AthenaK is performance-portable, shows better scaling, and gives accurate and convergent waveforms. The paper is organized as follows: In Section 2, we recap the choice of Z4c formulation for solving the Einstein equations. In Section 3, we describe the numerical implementation for vacuum general relativity, focusing on algorithmic differences from GR-Athena++. We present a series of test problems, including convergence of binary black hole waveforms, in Section 4. All tests were run on GPUs. We then present performance and scaling results in Section 5. Lastly, we conclude in Section 6.

[9] The code is available at https://github.com/IAS-Astrophysics/athenak.

2. Z4c Formulation

As in GR-Athena++, we evolve the Einstein equation in the conformally decomposed Z4 formulation (C. Bona et al. 2003), or Z4c (S. Bernuzzi & D. Hilditch 2010; D. Hilditch et al. 2013). The continuum form of our equations and gauge choices are exactly the same as those in GR-Athena++ (B. Daszuta et al. 2021), to which we refer interested readers for details. Here, we briefly summarize the advantages of the Z4c formulation.

Our choice of the Z4c formulation for solving the Einstein field equations (EFEs) is driven by its ability to integrate the strengths of two prominent approaches (C. Gundlach et al. 2005): the BSSNOK formulation (T. Nakamura et al. 1987; M. Shibata & T. Nakamura 1995; T. W. Baumgarte & S. L. Shapiro 1999) and the generalized harmonic gauge (GHG) formulation (H. Friedrich 1985; F. Pretorius 2005; L. Lindblom et al. 2006). First, like the GHG formulation, Z4c removes the zero-speed characteristic in the constraint subsystem, which could cause constraint violation to build up especially in the presence of matter; furthermore, it admits a natural constraint damping scheme (Z. Cao & D. Hilditch 2012; A. Weyhausen et al. 2012; D. Hilditch et al. 2013). Second, like BSSNOK, Z4c is compatible with the puncture scheme for evolving black holes, where coordinate singularities are explicitly advected in the computational domain. The puncture scheme avoids the need for excision, where the region interior to the apparent horizons is removed. Due to the nonlocality of the horizon finding routine, avoiding excision would greatly improve parallel efficiency.

We refer readers to Equations (8)–(13) of B. Daszuta et al. (2021) for the exact form of the equations used in our implementation. In addition to these equations, one must also specify the gauge conditions to close the system. As in GR-Athena++, we use the Bona–Másso lapse (C. Bona et al. 1996) and the gamma-driver shift (M. Alcubierre et al. 2003; Equation (22) in B. Daszuta et al. 2021).

3. Numerical Implementation

We refer interested readers to J. M. Stone et al. (2020, 2024) for details regarding the numerical infrastructure of AthenaK. Here we briefly recap key numerical features in AthenaK for evolving the vacuum Einstein equations, highlighting the difference in design choice from its predecessor GR-Athena++.

As in Athena++, the computational domain over which a set of physics is evolved is abstracted in a class named Mesh. Mesh contains rectangular blocks known as MeshBlocks. In contrast to the structure of Athena++, the MeshBlocks are further organized into MeshBlockPacks to reduce the number of kernel launches, which greatly improves performance on GPUs (P. Grete et al. 2022).

Each MeshBlock consists of an active region and a ghost region. Data in the ghost region must be communicated between MeshBlocks and is used to set boundary conditions.

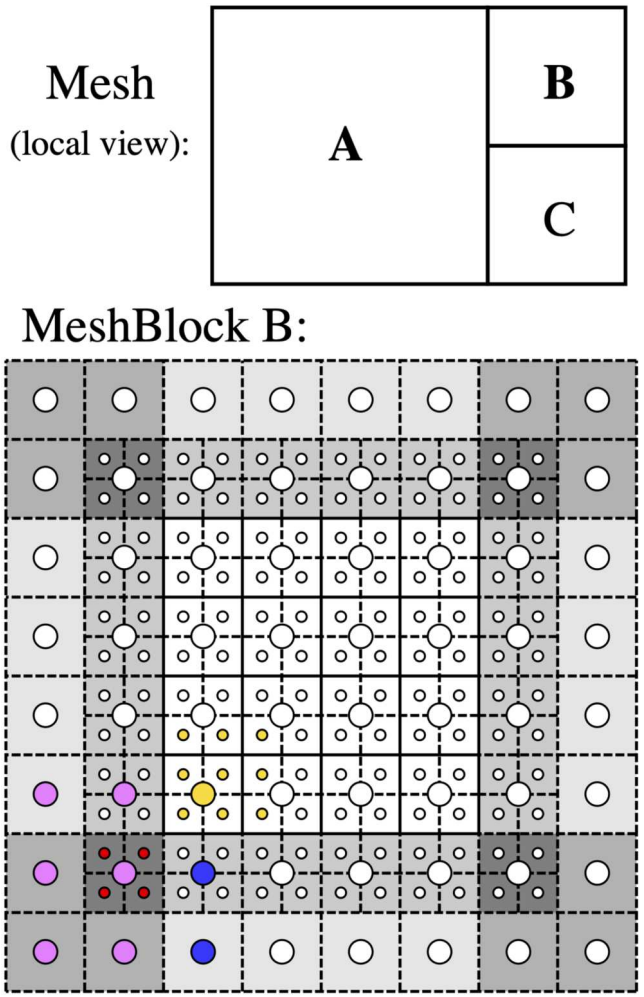

Figure 1. Top: schematic for `MeshBlock`s at a refinement boundary. The squares show the active region of each `MeshBlock`. Bottom: nodal structure inside `MeshBlock` B. The smaller circles represent the nodes in the `MeshBlock`, and the larger circles represent the coarse buffers used in AMR communication. The active region is unshaded, and the ghost region is shaded with different gray levels. We also draw schematics for the prolongation and restriction stencils. The smaller yellow circles show the restriction stencil to fill the coarse buffer node highlighted in yellow, and all colored large circles make up the prolongation stencil to fill in the red little circles in the fine ghost zone.

At each stage of the time integration process, where we employ a family of explicit Runge–Kutta methods, the ghost zones are filled by either the active region of neighboring `MeshBlock`s or the boundary conditions. When adaptive or static mesh refinement is used, neighboring `MeshBlock`s may have different resolutions, restricted to factors of 2 in `AthenaK`. On these refinement boundaries, a set of prolongation and restriction operations are performed to fill the ghost cells with the required accuracy. Since we use higher-order finite differencing to evolve the Z4c equations—unlike fluid evolution, which uses the second-order finite-volume approach —the boundary communication strategy at refinement boundaries must be modified. We detail these modifications below.

3.1. Modifications to Refinement Boundary Communication

As in the fluid sector, the boundary communication at the mesh refinement boundary is done with a set of prolongation and restriction operators. The top panel of Figure 1 illustrates three `MeshBlock`s at a refinement boundary.

From the perspective of `MeshBlock` A, the communication is the same as that for the fluid sector:

1. Pack and send active nodes that overlap with the coarse buffer of nearby finer `MeshBlock`s, for example, the large pink dots in Figure 1.
2. Receive nodes from the active region of coarse buffers of neighboring finer `MeshBlock`s, e.g., the large yellow dot in Figure 1.

See Figure 4 and discussion in Section 2.1.3 of J. M. Stone et al. (2020) for detailed illustrations.

The perspective for `MeshBlock` B is slightly modified:

1. Restrict the data in the active region to fill the coarse buffer.
2. Send a coarse buffer to neighboring `MeshBlock`s, to fill either their ghost zone (for neighbors with one less level of refinement, i.e., `MeshBlock` A) or the ghost region for their coarse buffer (for neighbors with the same refinement level, i.e., `MeshBlock` C).
3. Send a ghost zone to neighboring `MeshBlock`s at the same level, i.e., `MeshBlock` C.
4. Receive a coarse buffer and ghost zones from neighbors to fill the ghost zone and parts of the ghost zone for the coarse buffers that overlaps with the prolongation stencils.
5. Prolongate the coarse buffer to fill the fine ghost zones.

The main distinction of the above procedure from that for fluid, as described in J. M. Stone et al. (2020), is that the coarse buffers must be passed between neighboring `MeshBlock`s at the same level (namely B and C), as the prolongation stencil now overlaps with these nodes. We now discuss the change in the restriction and prolongation operators.

3.1.1. Restriction

For evolving fluid with a finite volume, each node stores the cell average, so the restriction operation becomes a trivial averaging. This is no longer the case with finite differencing. To fill the coarse buffers for the fine region, we use high-order Lagrange polynomial interpolation, which calculates the value for the desired node with coordinate x as a weighted sum, given a set of $\{(x_i, f(x_i))\}$ in the interpolation stencil. In one dimension, the weights are given by

$$\ell_i(x) = \prod_{\substack{0\leqslant m\leqslant k \\ m\neq j}} \frac{x - x_m}{x_i - x_m}. \tag{1}$$

Then, the value at x is given by

$$f(x) = \sum_{0\leqslant i\leqslant k} l_i(x) f(x_i). \tag{2}$$

This procedure can be generalized to two and three spatial dimensions by taking the outer product between the interpolation weights for each dimension. These weights for three dimensions are precomputed to save floating point operations at runtime for both the restriction and prolongation operations.

The order of convergence for Lagrange interpolation is $n + 1$, where n is the number of points in the stencil for each spatial dimension. In `AthenaK`, we choose the interpolation stencil size to be $n = N_g + 1$, where N_g is the number of ghost cells,

so that the error term converges away at third order for two ghosts, and fifth order for four ghosts.[10]

In the lower panel for Figure 1, we show the restriction stencil for two ghost cells. To fill the coarse buffer in the corner of the active region (unshaded), the large yellow dot as an example, we use the stencil consisting of nine small yellow dots (this becomes 27 nodes in three spatial dimensions). We note that the stencil is always within the active region. Such a choice is made to improve locality for the restriction step. Despite the asymmetry in the interpolation stencil, we have yet to find any sacrifice in accuracy.

3.1.2. Prolongation

The prolongation operator also uses a Lagrange interpolation polynomial. In Figure 1, we highlight the four fine ghost cells (little red dots) in the corner of `MeshBlock` B (or edges in three dimensions) that need to be filled during boundary communication. The prolongation stencil consists of nine colored large dots: the yellow one filled from restriction in the active region of `MeshBlock` B, the blue ones from the coarse buffer of `MeshBlock` C, and the pink ones from fine nodes in `MeshBlock` A.

3.2. Cell-centered versus Vertex-centered Finite Differencing

The aforementioned AMR strategies are also different from those in `GR-Athena++`, because here we use a cell-centered (CC) finite-differencing scheme as opposed to a vertex-centered (VC) one.[11] In a VC scheme, the finite-differencing nodes are located at cell vertices, so that the coarse nodes overlap with some fine nodes, making the restriction step trivial. However, this comes with some complications when coupling the Einstein equation with fluid equations, which are typically solved using finite-volume methods: one must interpolate the metric components and the gauge variables to the cell center, which serves as effective source terms for the fluid equations, and the stress–energy tensor for the fluid back to the cell vertices for evolving the spacetime. These extra interpolation operations add to the total computational cost when evolving spacetime with matter (B. Daszuta et al. 2024). In our CC scheme, the fluid variables are stored at the same location as the Z4c variables, making these interpolation operations unnecessary.

3.3. New AMR Criteria

In addition to the AMR algorithms implemented in `GR-Athena++` (see, e.g., A. Rashti et al. 2024), we implement two additional AMR criteria that are agnostic to the puncture location, based on the conformal factor χ. The conformal factor χ goes to zero toward the punctures, naturally tracking the location of the black holes. Since it is crucial to adequately resolve the horizon to prevent constraint violation from propagating outward, we require more resolution near the punctures. Therefore, χ could be used as a good proxy for specifying the region of the computational domain to refine for binary black hole problems. Here, we present two χ-based AMR criteria.

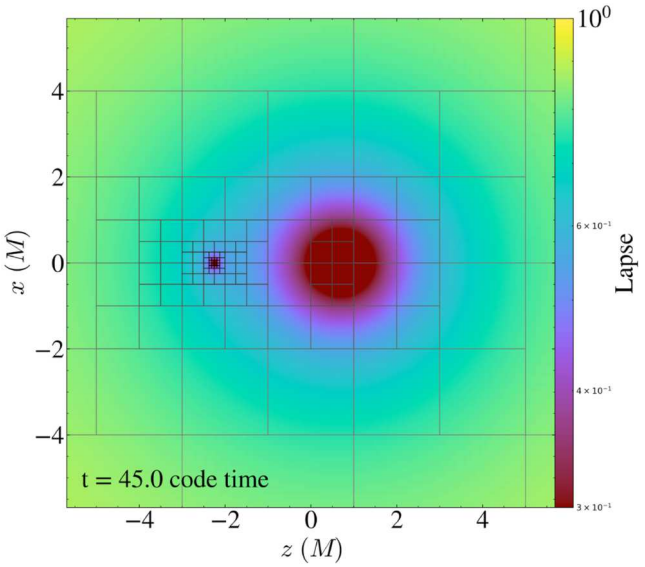

Figure 2. Mesh refinement structure of a black hole binary with a mass ratio of 20 using the $d\chi$ − max refinement criterion. The smaller hole automatically receives four more levels of mesh refinement due to its larger gradient in the conformal factor.

3.3.1. χ-min Criterion

For the χ-min criterion, we assess whether the value of χ within each `MeshBlock` falls below a specified threshold. If it does, the `MeshBlock` is refined, provided it has not already reached the maximum level of mesh refinement. The `AthenaK` AMR framework maintains a two-to-one refinement condition between neighboring `MeshBlocks`, allowing the resolution to gradually decrease as one moves away from the vicinity of the punctures. In the BAM calibration binary black hole run, we observed that this criterion typically results in fewer `MeshBlocks` being created or destroyed compared to the commonly used puncture-tracker-based L^2 criterion. A snapshot of the refinement structure in the strong field for the calibration run is shown in Figure 9. A detailed comparison with the L^2 criterion is deferred to future work.

3.3.2. dχ-max Criterion

The χ criterion works well for equal-mass ratios, but in asymmetric binaries, it tends to overresolve the interior of the larger black hole. To address this, we also implement a $d\chi$-max criterion. At each refinement step, we check whether the gradient of χ exceeds a certain threshold within a `MeshBlock`. If it does, the `MeshBlock` is refined, unless it has already reached the maximum refinement level. Since this criterion is based on the gradient of the conformal factor rather than on χ itself, it naturally results in higher levels of refinement around the smaller black hole. We illustrate the resulting grid structure during a head-on collision with a reference mass ratio of 20 in Figure 2. The $d\chi$ criterion effectively tracks the black holes and naturally produces four additional levels of mesh refinement near the smaller puncture, avoiding overresolving the larger black hole's interior.

[10] By default, we use second-order spatial differencing when using two ghost cells, and sixth-order with four ghosts, while the time integrator is kept at fourth order. In practice we never find the prolongation and restriction operators to dominate over the spatial differencing error.

[11] A recent CC extension, referred to as CX in B. Daszuta et al. (2024), has been added to `GR-Athena++`. We defer a comparison between the two schemes to future work.

3.3.3. Derefinement at the Wave Zone

A common issue for both of the AMR criteria above is that the resolution could fall off fast away from the black holes. This is problematic since the gravitational waveform is usually extracted at large radii ($\gtrsim 100 r_g$), and a low resolution at the wave zone could significantly impact its accuracy. To address this, we implement an additional radius criterion, where all meshblocks within a certain radius of a given point are kept above a specified refinement level.

3.4. Other Diagnostics

As in GR-Athena++, we implement two key diagnostics for binary black hole evolution: puncture trackers and wave extraction. The puncture trackers are initialized at the puncture location in the initial data and advected across the domain by integrating the shift vector at each time step. To extract gravitational waves, we compute the outgoing Weyl scalar Ψ_4 using a coordinate tetrad, then interpolate Ψ_4 onto geodesic spheres at various radii and decompose it into spin-weighted spherical harmonics. Ongoing work includes a horizon finder and outputting worldtube data for Cauchy characteristic extraction (N. T. Bishop et al. 1996; C. Reisswig et al. 2010; N. T. Bishop & L. Rezzolla 2016; K. Barkett et al. 2020; J. Moxon et al. 2020, 2023).

4. Code Tests

To test the correctness and accuracy of our implementation, we perform a series of tests and compare the results to those of BAM and GR-Athena++ in this section. All tests in this section are done on NVIDIA A100 GPUs.

4.1. "Apples with Apples" Tests

We first perform two tests from the "apples with apples" test-bed problems (M. Alcubierre et al. 2005; M. Babiuc et al. 2009).

The tests performed here are effectively one-dimensional, on a three-dimensional mesh with periodic boundary conditions. The size of the mesh is set to 1 across all dimensions. We fix the number of points in the trivial directions to 4, and adjust the resolution in the nontrivial dimension by number of points $N = 50\rho$, with $\rho \in \mathbb{N}$. Following Z. Cao & D. Hilditch (2012), we set the Z4c parameters as follows: constraint damping $\kappa_1 = 0.02$ and $\kappa_2 = 0$, and shift damping $\eta = 2$. The Kreiss–Oliger (KO) dissipation parameter is chosen as $\sigma = 0.02$ and the Courant–Friedrichs–Lewy (CFL) condition as $1/2$. Lastly, we use the puncture gauge evolution equations in GR-Athena++, with the initial gauge condition set to

$$\alpha|_{t=0} = 1, \ \beta^i|_{t=0} = 0. \tag{3}$$

4.1.1. Robust Stability

The robust stability test aims at detecting instabilities within numerical implementations regarding the principal part of the evolution equation. At the initial slice, we add resolution-dependent uniform random noise to the Arnowitt–Deser–Misner (ADM) metric and extrinsic curvature $\epsilon \in (-10^{-10}\rho^{-2}, 10^{-10}\rho^{-2})$. Then, we monitor the constraint violation through the L^2 norm of the collective constraint (see Equation (21) from B. Daszuta et al. 2021), and the deviation

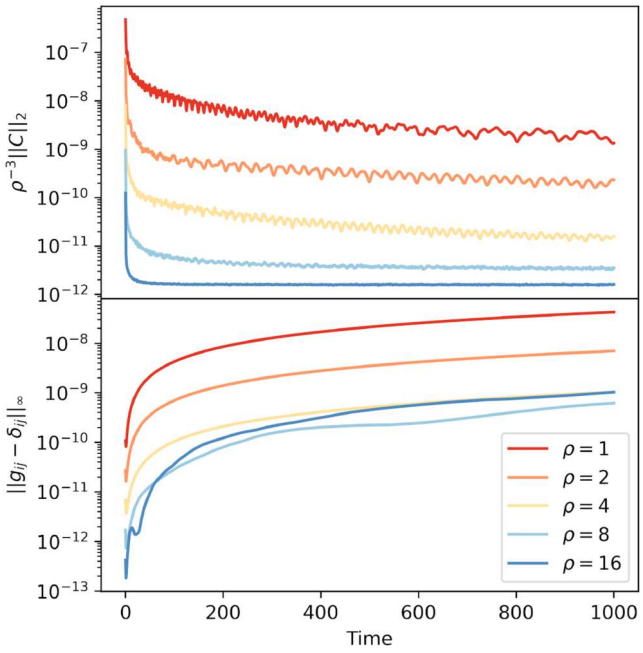

Figure 3. Robust stability test at increasing resolution: L^2 norm of the Z4c collective constraints (top) and L^∞ norm for deviation from flat metric (bottom). Both shown in logarithmic scale.

of the ADM metric from a flat metric by $||g_{ij} - \delta_{ij}||_\infty$ as suggested in D. Daverio et al. (2018). This test is run with $N_g = 2$ (second-order spatial differencing) and the RK4 time integrator. The results in Figure 3 demonstrate robust numerical stability and efficacy of the constraint damping scheme, consistent with GR-Athena++.

4.1.2. Gauge Wave

We further perform an unshifted gauge wave test to test the robustness of the evolution system as well as the convergence properties of the high-order spatial differencing algorithms. A coordinate transformation of the Minkowski spacetime yields the following induced metric:

$$\gamma_{xx} = 1 - H(x, t), \ \gamma_{yy} = \gamma_{zz} = 1, \tag{4}$$

with the nontrivial extrinsic curvature component

$$K_{xx} = \frac{\partial_t H(x, t)}{2\sqrt{1 - H(x, t)}}, \tag{5}$$

where $H(x, t)$ is a freely specifiable smooth function. Here, we choose a periodic function:

$$H(x, t) = A \sin\left(\frac{2\pi(x - t)}{d}\right), \tag{6}$$

with $A = 0.01$ and $d = 1$.

For this test, we use second-, fourth-, and sixth-order spatial differencing algorithms, corresponding to numbers of ghost points $N_g = 2, 3, 4$, respectively. In Figure 4, we show the collective constraint violations for the three algorithms at two different resolutions ($\rho = 1$ and 2), where expected decreases in error are seen with the higher-order methods.

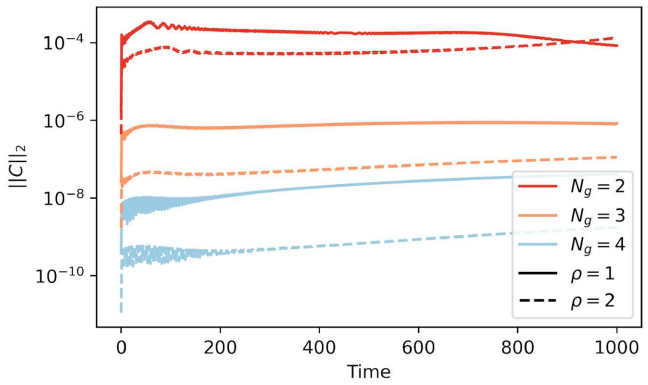

Figure 4. Constraint violation monitoring for the gauge wave test. The spatial differencing algorithms are labeled by the number of ghost points N_g, and the dashed lines correspond to runs with twice the resolution. As expected, the error decreases drastically with the order of the spatial algorithms and with the resolution.

We further measure the convergence factor n by comparing constraint violation at different resolutions, given by

$$n_\rho = \log_2\left(\frac{||C||_2^\rho}{||C||_2^{2\rho}}\right), \tag{7}$$

where ρ labels the resolution.

The results are shown in Figure 5. For the second-order method, the convergence factor approaches the expected value as the resolution increases, as expected. For the fourth-order method, the convergence factor first converges toward the theoretical value with increasing resolution, then decreases at very high resolution, since the accumulation of roundoff error becomes dominant. For the sixth-order method, this roundoff error becomes the primary source of error even at the lowest resolution, yielding a convergence factor slightly above 4. We therefore conclude that `AthenaK` robustly passes the gauge wave test.

4.2. Linear Wave Convergence

We next test the code's ability to accurately propagate the gravitational wave by performing a linear wave convergence test. In contrast to the implementation in the "apples with apples" tests, we perform the test in full three spatial dimensions by propagating the wave along the diagonal of the `Mesh`, with an amplitude of 10^{-8}. We then calculate the $L^1_{\rm rms}$ error after one cycle of propagation at time T, given by

$$L^1_{\rm rms} = \sqrt{\sum_{a,b\in\{x,y,z\}}\left(\int_{\rm mesh} dv|g_{ab}(T) - g_{ab}(0)|/{\rm Vol.}\right)^2}, \tag{8}$$

where Vol. is the volume of the `Mesh`.

We try the evolution with three different combinations of spatial differencing order and time integrator. As shown in Figure 6, we find all the algorithms converge at the expected order. For the combination of $N_g = 4$ and RK4, we find it to converge at sixth order, indicating that in this regime the spatial differencing error dominates over that of the time integrator, and therefore the convergence order of the scheme is higher than that of the time integrator.

To test the accuracy for the prolongation and restriction operators, we perform the same linear wave evolution but now with a corner of the `Mesh` refined at one level higher. We plot

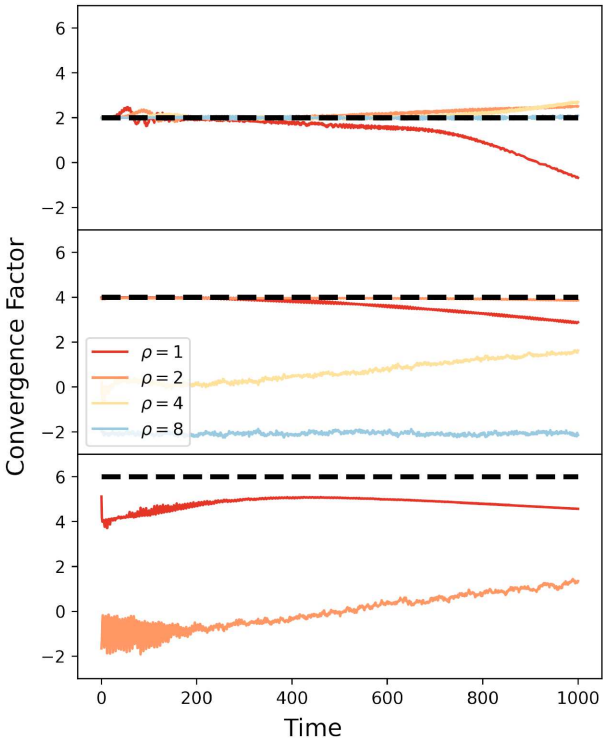

Figure 5. Order of convergence from the gauge wave test. The expected orders of convergence are shown as a bold dashed line for the second-order (top), fourth-order (mid), and sixth-order (bottom) algorithms. At high resolution and for high-order differencing, the accumulation of roundoff error quickly dominates and reduces the order of convergence.

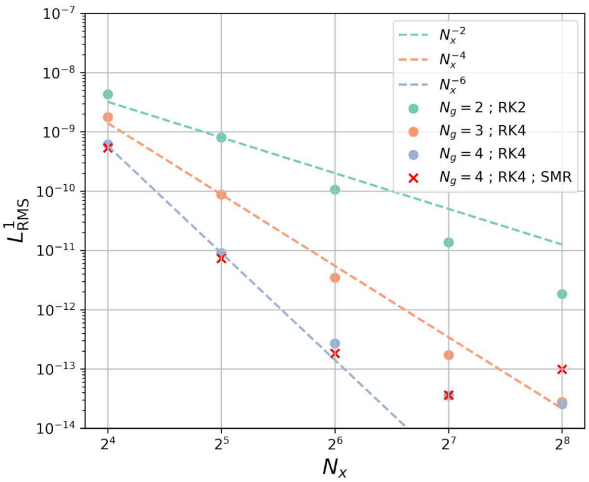

Figure 6. $L^1_{\rm rms}$ error of linear gravitational wave after one cycle of propagation using different numerical schemes. The red crosses are from runs with a quadrant of the `Mesh` refined, to test the accuracy for the prolongation and restriction operators described in Section 3.

the results as red crosses in Figure 6. We find that the rms error is consistently lower than that in the same run without mesh refinement in the corner, except for the last data point, where the added number of operations causes accumulation of roundoff error and therefore an overall increase in the rms error.

At high resolution (lower-right corner of Figure 6), we find that the rms error plateaus. This is due to nonlinear effects (coming from the fact that the solution of the EFE is not the same as that of the advection equation). To further explore

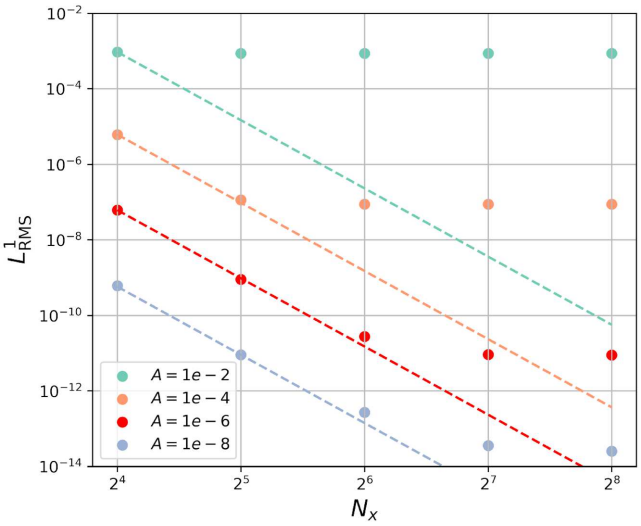

Figure 7. Linear wave tests for different initial amplitudes reveal a nonlinear effect. The dashed lines show the expected rate of convergence. Note that the plateau in the rms errors becomes quadratically spaced in the wave amplitude, indicating that they arise from quadratic effects.

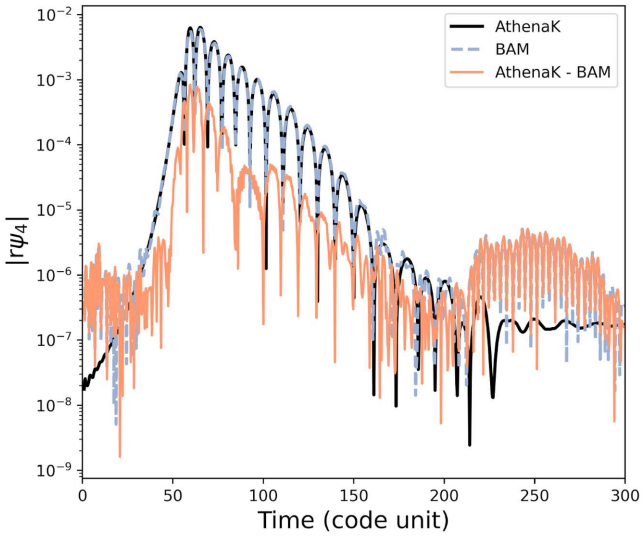

Figure 8. AthenaK vs. BAM waveform for a single spinning puncture. Here, we plot the $l = 2$, $m = 0$ component of the Weyl scalar extracted at a coordinate radius of 50 M. The difference between the two codes remains small during the course of the evolution.

this, we run the same linear wave test with sixth-order spatial differencing and RK4 but with a different wave amplitude. The results are shown in Figure 7, where we find that the plateau in the rms error becomes quadratically spaced in the wave amplitude, indicating the cause of this error is indeed a quadratic effect, with a coefficient of order 10.

4.3. Single-puncture Test

Before simulating a full binary black hole coalescence, we first demonstrate the robustness of AthenaK for evolving a single spinning puncture. Using initial data generated by the TwoPunctures library, as employed in similar tests of the BAM code in D. Hilditch et al. (2013) and the GR-Athena++ code in B. Daszuta et al. (2021), we simulate the evolution of a single spinning puncture, representing a rotating black hole with a dimensionless spin parameter $a = 0.5$. We initialize two black holes: one with a target mass of $1M$ and spin $a = 0.5$, and another with a negligible mass of $10^{-12}M$ and zero spin, separated initially by $10^{-5}M$. The smaller black hole is not numerically resolved by the evolution code and only sources a small perturbation. Therefore this initial data can be treated as a target rotating black hole with some small perturbation. We utilize static mesh refinement to construct a grid around the puncture, extending to $\pm 1024M$ in each spatial dimension. The resolution is set to be consistent with the BAM evolution both at the puncture ($\delta x = 0.08333M$) and in the wave zone ($\delta x = 0.66667M$). As shown in Figure 8, the $l = 2$, $m = 0$ harmonic of the gravitational waveform from AthenaK using sixth-order spatial differencing is consistent with that from BAM, which used fourth-order finite differencing.

4.4. Binary Black Hole Evolution

Lastly, we test the correctness and convergence of the gravitational waveform from binary black hole coalescence. For this we use the BAM calibration binary black hole problem (B. Brügmann et al. 2008), which consists of an equal-mass, nonspinning quasi-circular binary that merges in three orbits with the TwoPunctures initial data (M. Ansorg et al. 2004).

Here, we compare the waveform from AthenaK with that from GR-Athena++.

As in GR-Athena++ (B. Daszuta et al. 2021), we choose a KO dissipation of 0.5, constraint damping parameters $\kappa_1 = 0.02$ and $\kappa_2 = 0$, and a CFL number of 0.25. We use sixth-order spatial differencing and the RK4 time integrator.

4.4.1. Self-convergence

We first test the self-convergence of the gravitational waveform measured with the Weyl scalar Ψ_4. We use a Mesh extending from -1024 to 1024 M in all three spatial dimensions, with M measured in the ADM mass of the initial data. The root grid consists of 4^3 MeshBlocks. To resolve the horizons, we use 10 levels of mesh refinement with the $\chi -$ min refinement criterion, with the $\chi_{\rm min}$ threshold set to 0.3.

We run the calibration runs at three different resolutions. To demonstrate convergence with AMR enabled, we uniformly increase the resolution by factors of 2 by changing the number of points in each MeshBlock (32^3, 64^3, and 128^3) but without changing the overall structure of the mesh (a snapshot of which is shown in Figure 9), so that the structure of the computational domain is maximally consistent between runs with different resolutions. The resulting resolutions at the puncture are 0.03125, 0.015625, and 0.0078125, respectively.

The numerical waveform consists of the continuum limit $\psi_{\rm cont}(t)$ and an error term that scales polynomially with resolution Δx, namely

$$\psi_{\rm NR}(t, \Delta x) = \psi_{\rm cont}(t) + \xi(\Delta x)^n, \tag{9}$$

where n is the convergence order for the algorithm, and ξ is an error coefficient dependent on the discretization schemes as well as the physical problem. When comparing waveforms at different resolutions, the residual ϵ between the two runs is then given by

$$\begin{aligned}\epsilon(\Delta x_1, \Delta x_2) &= \psi(t, \Delta x_1) - \psi(t, \Delta x_2) \\ &= \xi((\Delta x_1)^n - (\Delta x_2)^n).\end{aligned} \tag{10}$$

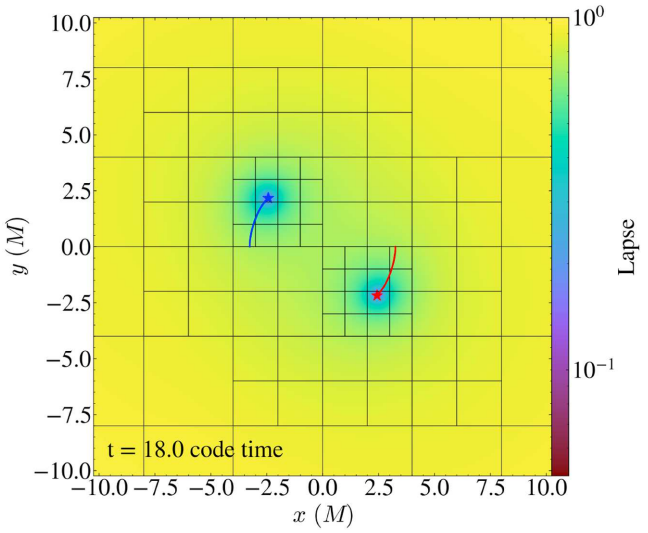

Figure 9. Mesh refinement structure during the calibration binary black hole run using the χ criterion. The history of punctures' location (through puncture trackers) is overplotted as blue and red curves.

To show the order of convergence, we then show that

$$\frac{\epsilon\,(\Delta x_1,\,\Delta x_2)}{\epsilon\,(\Delta x_2,\,\Delta x_3)} \simeq \frac{(\Delta x_1)^n - (\Delta x_2)^n}{(\Delta x_2)^n - (\Delta x_3)^n} = :Q_n, \tag{11}$$

for an appropriate order n.

In Figure 10, we plot the waveform as well as the residual between different resolutions in terms of the Weyl scalar Ψ_4, extracted at a coordinate radius of 100 M. The residuals for both the waveform amplitude and the phase are plotted in logarithmic scale to better demonstrate convergence. The residual between the high and mid resolution when scaled up by the Q factor is consistent with that between the low and mid resolution, suggesting a fourth-order convergence.[12]

4.4.2. Comparison with GR-Athena++

In addition to self-convergence, we compare the waveforms with those from GR-Athena++ using the same initial data. The available GR-Athena++ runs are done using the L^2 AMR criteria, and with a KO dissipation of 0.02 (as opposed to the 0.5 used here). The size (number of points along each dimension) of the MeshBlocks is also significantly different, as smaller (16^3) MeshBlocks are used to improve scaling on CPUs. Furthermore, the GR-Athena++ runs are done with a VC scheme, as opposed to the CC scheme for AthenaK. We choose GR-Athena++ and AthenaK simulations with matching resolution at the puncture, and compare the residual difference between the two codes at two different resolutions (low and mid). Despite the aforementioned differences, we find that the difference between the codes still converges away with resolution (see Figure 11).

4.4.3. Accuracy and Resolution Requirement with AMR

In this section we do a brief exploration of how the grid structure affects the accuracy of the waveform. Starting with a low-resolution run, to increase the accuracy of the simulation, one can perform a global refinement, where the resolution throughout the Mesh is uniformly increased. Alternatively, one can increase the maximum level of mesh refinement, which will only increase the resolution around the punctures, leaving the wave zone resolution untouched. The advantage of the later strategy is due to performance: while a global increase in resolution, say, by a factor of 2, increases computational cost by a factor of 16, changing the maximum level of mesh refinement by 1 only increases the computational cost by a factor of roughly 2. For the later strategy, the number of MeshBlocks and therefore of gridpoints only changes by around 10% typically, so the increase in computational cost is primarily due to the CFL condition. The resulting grid structure yields matching resolution at the puncture but only half the resolution in the wave zone.

In Figure 12, we plot the error in waveform for the calibration run with several different configurations. In addition to the low and mid resolution with 10 levels of mesh refinement used in the convergence test, we now add an additional run using the same root grid as the low-resolution run but now with 11 levels of mesh refinement. The error on

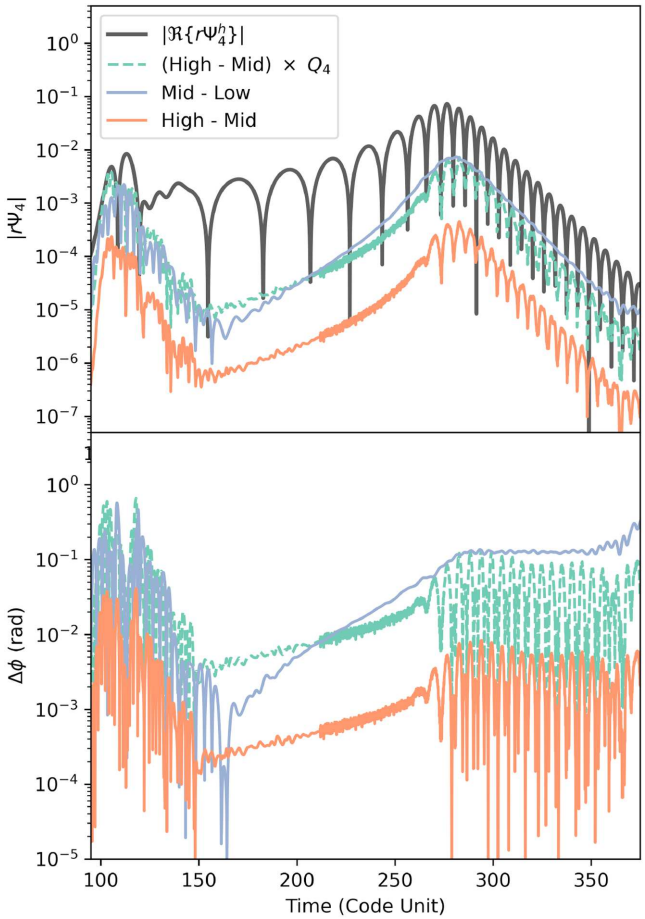

Figure 10. Convergence of gravitational waveform for the calibration binary black hole run. Top: we show the real part of the quadrupolar ($\ell = m = 2$) component of Ψ_4 from the high-resolution run, and in colored lines we show the residual between mid and low resolution $|\Psi_4^{\rm mid} - \Psi_4^{\rm low}|$ (in purple), and between high and mid resolution $|\Psi_4^{\rm high} - \Psi_4^{\rm mid}|$ (in orange). To demonstrate fourth-order convergence, we scale up the residual between the high and mid resolution by the Q factor defined in Equation (11) and plot it as a dashed cyan curve. When this curve is compared with the purple curve, clear fourth-order convergence is seen. Bottom: the same but for the phase difference between the waveforms.

[12] This order of convergence could in principle be improved by adopting the techniques developed in Z. B. Etienne (2024), which showed that the numerical error in puncture evolution is dominated by the sharp lapse features at the start of the evolution. We will investigate this in future work.

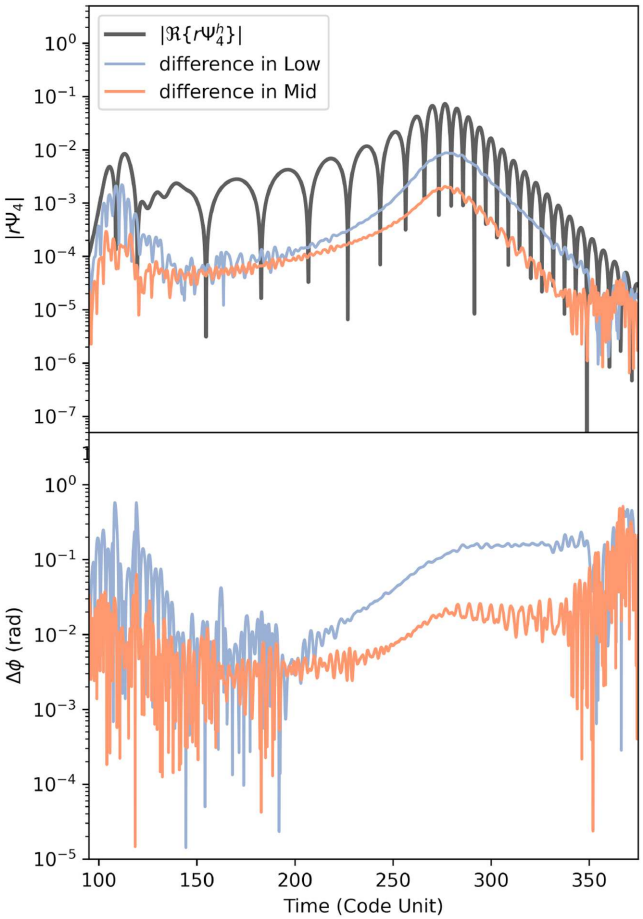

Figure 11. Waveform comparison between `AthenaK` and `GR-Athena++`. As in Figure 10, the black curve shows the waveform amplitude at high resolution from `AthenaK`. The purple and orange curves show the difference between the two codes for the low- and mid-resolution runs, respectively.

the waveform is then estimated by the difference from the high-resolution waveform. We find that, during the inspiral phase, as marked by the blue shaded region in Figure 12, the error in the new Low Lev 11 run is practically the same as that for the mid-resolution run, despite being 8 times cheaper. However, this is no longer the case during the early nor late part of the waveform, corresponding to the junk radiation and merger ringdown phases, respectively. This is primarily due to the presence of higher-frequency signals, which are not as well resolved in the wave zone.

In typical binary black hole simulations, the inspiral phase is an order of magnitude longer than both the junk radiation and merger ringdown phases. Therefore, increasing the level of refinement instead of a global refinement would save a tremendous amount of computing resources. The junk radiation phase does not need to be resolved, as it is typically cut out from the waveform (see, e.g., I. G. Pretto et al. 2025). To resolve the merger ringdown phase, one can simply increase the wave zone resolution when the coordinate separation of the two punctures is below a certain threshold. We defer a detailed diagnostic to future work.

5. Performance

As `AthenaK` aims at solving exascale problems, we demonstrate the performance portability and scaling of the vacuum NR module on large GPU clusters. In this section, we

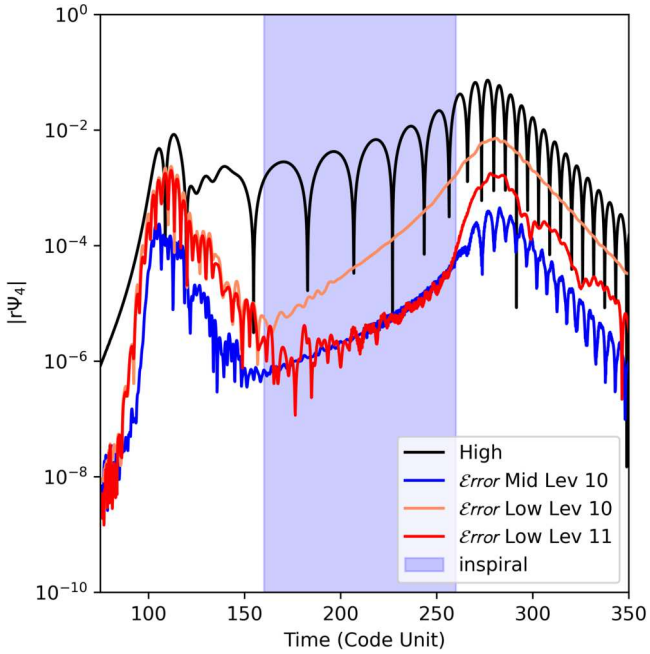

Figure 12. Comparison of error in gravitational waveform with different grid structures. The error is estimated by taking the difference between the waveform of a given run and that for the high-resolution run. The Mid (blue) and Low (orange) Lev 10 runs share the same mesh refinement structure, yet the Mid has twice the resolution within each `MeshBlock`. Low Lev 11 (red) has the same root grid as Low Lev 10, yet one more level of mesh refinement, yielding the same resolution at the puncture as the mid-resolution run. We find that during the inspiral phase Low Lev 11 yields an error consistent with the Mid Lev 10 run, despite being roughly 8 times cheaper.

first show the performance portability across a wide range of hardware architectures. Furthermore, we show its scaling on Frontier and Perlmutter, which employ `AMD` and `NVIDIA` GPUs, respectively. Throughout this section, performance is quantified by zone cycle per second (ZCPS), or the number of cell updates per wall clock second, and all tests are done with sixth-order spatial differencing and the RK4 time integrator.

5.1. Performance Portability

Modern high-performance computers employ a wide range of hardware architectures. Therefore, performance portability is arguably the most crucial feature for a code aiming at exascale applications. To this end, we test the Z4c module in `AthenaK` on a wide range of hardware, ranging from ARM and X86 CPUs to GPUs by different vendors. To measure performance, we run the linear gravitational-wave problem with 64^3 `MeshBlocks`. To make sure the GPUs are properly saturated, we load them with the largest number of `MeshBlocks` that could fit into the video random-access memory (VRAM), whereas on CPUs we only use a single `MeshBlock` as resource saturation is not a concern. We find that both `AMD` and `NVIDIA` GPUs offer a speedup of 2 orders of magnitude when compared to a modern CPU core. The performance numbers are summarized in Table 1.

5.2. Weak Scaling

It is also important to demonstrate that the code can saturate resource on large computing clusters through weak-scaling tests. For this test, we evolve a single black hole in a puncture gauge with 10 levels of static mesh refinement on Frontier. To

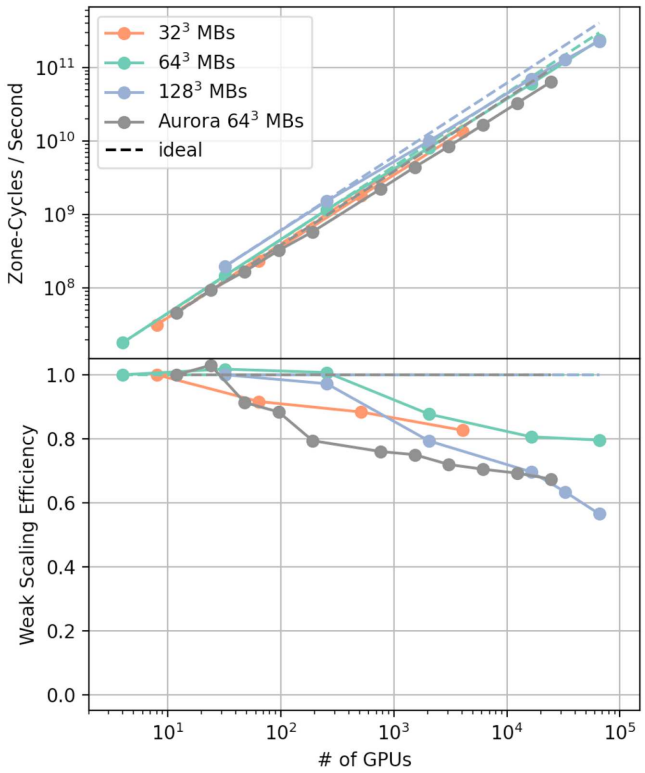

Figure 13. Weak-scaling performance (top) and efficiency (bottom) for evolving a single black hole on Frontier and Aurora with varying `MeshBlock` sizes.

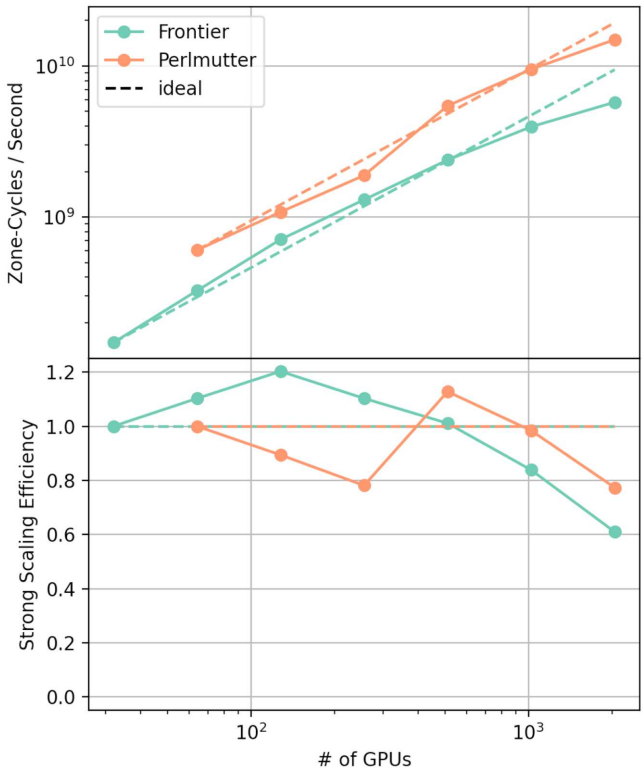

Figure 14. Strong-scaling performance (top) and efficiency (bottom) for evolving a single black hole with 64^3 `MeshBlock` on both Frontier and Perlmutter.

Table 1
Performance Portability of `AthenaK` Z4c Module across a Wide Range of Hardware Architectures

Configuration	ZCPS	1/Speedup
`NVIDIA` A100	1.87×10^7	1
`NVIDIA` Grace Hopper	3.74×10^7	0.5
`NVIDIA` RTX 3070 Ti	4.32×10^6	4.33
`AMD` MI250X (single die)	5.08×10^6	3.68
`AMD` 7950x Zen 3 (single core)	4.53×10^4	412
`Intel` Cascade Lake (single core)	1.71×10^4	1093
`Apple` M3 (single CPU core; ARM)	6.53×10^4	286

Note. Here, ZCPS is measured either per device or per CPU core. We also show the inverse speedup of each computing device as compared against a mainstream `NVIDIA` A100 GPU.

also show the dependence on the size of the `MeshBlock`, we perform the same test with three different `MeshBlock` sizes, namely 32^3, 64^3, and 128^3.

We then double the resolution uniformly throughout the `Mesh` and increase the resource by factors of 8 simultaneously, except for the largest run, which uses 65,536 GPUs, where a factor of 8 increase on the resource exceeds the size of Frontier. For the last run, we only double the resolution along the x- and y-axis and increase the resource by a factor of 4 instead. We find that, with 64^3 `MeshBlocks`, scaling up from four to 65,536 GPUs retains an 80% weak-scaling efficiency, indicating that `AthenaK` can efficiently run on exascale machines (see Figure 13). The performance and scaling efficiency are shown in Table 2. On Aurora, jobs are limited to 2048 nodes with 12 GPUs per node due to stability issues.

For the runs with 32^3 `MeshBlocks`, we use twice the number of GPUs for the same problem because it cannot be fitted onto the same number of GPUs due to the increased ratio of ghost zones to active zones. In general, we observe roughly a factor of 2 increase in performance when changing the size of the `MeshBlock` from 32^3 to 128^3 on `AMD` MI250X.

5.3. Strong Scaling

To enable a rapid scan of the binary black hole parameter space, excellent strong scaling is required to shrink the wall clock for each simulation. For this test, we perform the same test with 4544 64^3 `MeshBlocks`. The size of this test problem is tuned to match those for high-resolution binary black hole runs with a high mass ratio or high spin. The resulting resolution at the puncture is $\sim$0.008 M, same as that of the high-resolution binary black hole calibration run discussed in Section 4.4.

We perform the test on both Frontier and Perlmutter. In contrast to the CPU computing case, the size of the VRAM is the limiting factor for how many GPUs are needed to evolve a given problem. Frontier utilizes AMD MI250X, each with two dies (two logical GPUs) and a total of 128 GB VRAM, whereas Perlmutter uses the 40 GB variant of NVIDIA A100. Due to the limitation of VRAM size per device, the problem can be fitted onto 32 logical GPUs on Frontier, whereas on Perlmutter we use 64 GPUs at a minimum.

The performance numbers are shown in Figure 14 and Table 2. We find that `AthenaK` still maintains 84% and 77% efficiency on Frontier and Perlmutter, respectively, when scaling up the computational resource by a factor of 32, though the number drops significantly to 61% on Frontier when we

Table 2
Strong- and Weak-scaling Data with 64^3 `MeshBlocks`

	Weak Scaling (Frontier)		Strong Scaling (Frontier)		Strong Scaling (Perlmutter)	
No. GPUs	ZCPS	Efficiency	ZCPS	Efficiency	ZCPS	Efficiency
4	1.81×10^7	1.00	⋯	⋯	⋯	⋯
32	1.48×10^8	1.02	1.48×10^8	1.00	⋯	⋯
64	⋯	⋯	3.26×10^8	1.10	6.04×10^8	1.00
128	⋯	⋯	7.10×10^8	1.20	1.08×10^9	0.89
256	1.17×10^9	1.01	1.30×10^9	1.10	1.89×10^9	0.78
512	⋯	⋯	2.39×10^9	1.01	5.45×10^9	1.13
1024	⋯	⋯	3.96×10^9	0.84	9.51×10^9	0.98
2048	8.14×10^9	0.88	5.76×10^9	0.61	1.49×10^{10}	0.77
16,384	5.99×10^{10}	0.81	⋯	⋯	⋯	⋯
65,536	2.36×10^{11}	0.80	⋯	⋯	⋯	⋯

Note. The scaling efficiency is computed by the run with the smallest number of GPUs for all cases.

increase the resource further by a factor of 64. This is not unexpected on GPUs, as the minimum amount of workload to saturate a GPU is much larger than that for CPUs. We also report a factor of 2 increase in performance per logical device for `AthenaK` when running on `NVIDIA` A100 compared with `AMD` MI250X (single die).

6. Discussion and Conclusion

In this paper, we introduce and validate the NR module within `AthenaK`, a reimplementation of `GR-Athena++` optimized for exascale applications using the `kokkos` library. Our focus is on solving the vacuum Einstein equations in the Z4c formulation with octree AMR, addressing the need for high-accuracy gravitational waveforms with large parameter space coverage, which is crucial for next-generation gravitational-wave detectors.

We demonstrate `AthenaK`'s accuracy and scalability through convergence of linear gravitational waves, accuracy of spinning puncture evolution, and convergence of gravitational waveforms from equal-mass black hole binaries, by cross-code validation with `BAM` and `GR-Athena++`. These results confirm `AthenaK`'s reliability for precision waveform modeling and its ability to produce consistent, accurate results.

We further demonstrate performance portability across all major high-performance computing architectures. In particular, scaling tests on Frontier and Perlmutter, using AMD and NVIDIA GPUs, respectively, demonstrate AthenaK's excellent performance on exascale platforms. The code achieves 80% weak-scaling efficiency up to 65,536 GPUs on Frontier and approximately 80% strong-scaling efficiency when increasing the number of GPUs by a factor of 32 on both Frontier and Perlmutter. These results underscore AthenaK's capacity to leverage the computational power of emerging exascale computers, making it a powerful tool for large-scale NR simulations.

Acknowledgments

We thank the many contributors to the `Athena++` code project. We thank Patrick Mullen and Frans Pretorius for valuable conversations, and Matthew Cawood for performance numbers on NVIDIA's Grace Hopper chip. We thank Rossella Gamba for running the weak-scaling test on Aurora. H.Z. thanks Frans Pretorius and Nils Vu for providing comments on the draft. D.R. was supported by funding from the U.S. Department of Energy, Office of Science, Division of Nuclear Physics under award numbers DE-SC0021177 and DE-SC0024388, by NASA under award No. 80NSSC21K1720, and by the National Science Foundation under grants No. PHY-2011725, PHY-2020275, PHY-2116686, and AST-2108467. J.M.S. was supported by a subcontract from the Texas Advanced Computing Center from National Science Foundation grant 2139536, and by the Eric and Wendy Schmidt Funds for Strategic Innovation. F.Z., S.B., and B.D. acknowledge funding from the European Union H2020 program under ERC Starting Grant No. BinGraSp-714626. S.B. and B.D. acknowledge funding from the European Union Horizon program under ERC Consolidator Grant No. InspiReM-101043372.

A portion of this work was carried out during the TACC Open Hackathon, organized as part of the Open Hackathons program. We gratefully acknowledge the support of OpenACC (https://www.openacc.org/). We are especially thankful to Matthew Cawood, Victor Eijkhout, and Forrest Glines for their valuable insights and discussions during the optimization of `AthenaK`.

Simulations were performed on OLCF's Frontier, NERSC's Perlmutter, Pennsylvania State University's Institute for Computational and Data Sciences' Roar Collab supercomputer, and Princeton University's Della and Stellar supercomputers. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under contract No. DE-AC02-05CH11231. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract No. DE-AC05-00OR22725. The simulations presented in this article were partly performed on computational resources managed and supported by Princeton Research Computing, a consortium of groups including the Princeton Institute for Computational Science and Engineering (PICSciE) and the Office of Information Technology's High Performance Computing Center and Visualization Laboratory at Princeton University.

ORCID iDs

Hengrui Zhu (朱恒锐) https://orcid.org/0000-0001-9027-4184
Jacob Fields https://orcid.org/0000-0001-5705-1712
Francesco Zappa https://orcid.org/0000-0002-0351-6518

David Radice https://orcid.org/0000-0001-6982-1008
James M. Stone https://orcid.org/0000-0001-5603-1832
Alireza Rashti https://orcid.org/0000-0003-3558-7684
William Cook https://orcid.org/0000-0003-2244-3462
Sebastiano Bernuzzi https://orcid.org/0000-0002-2334-0935
Boris Daszuta https://orcid.org/0000-0001-6091-2827

References

Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2016a, PhRvL, 116, 061102
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2016b, PhRvL, 116, 241102
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2017a, PhRvL, 119, 161101
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2017b, CQGra, 34, 044001
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2020, LRR, 23, 3
Akutsu, T., Ando, M., Arai, K., et al. 2021, PTEP, 2021, 05A102
Alcubierre, M., Brügmann, B., Diener, P., et al. 2005, PhRvD, 72, 044004
Alcubierre, M., Corichi, A., González, J. A., Nunez, D., & Salgado, M. 2003, CQGra, 20, 3951
Amaro-Seoane, P., Audley, H., Babak, S., et al. 2017, arXiv:1702.00786
Ansorg, M., Brügmann, B., & Tichy, W. 2004, PhRvD, 70, 064011
Babiuc, M., Bishop, N., Szilagyi, B., & Winicour, J. 2009, PhRvD, 79, 084011
Baker, J. G., Campanelli, M., Pretorius, F., & Zlochower, Y. 2007, CQGra, 24, S25
Barkett, K., Moxon, J., Scheel, M. A., & Szilágyi, B. 2020, PhRvD, 102, 024004
Baumgarte, T. W., & Shapiro, S. L. 1999, PhRvD, 59, 024007
Baumgarte, T. W., & Shapiro, S. L. 2010, Numerical Relativity: Solving Einstein's Equations on the Computer (Cambridge: Cambridge Univ. Press),
Berger, M. J., & Colella, P. 1989, JCoPh, 82, 64
Berger, M. J., & Oliger, J. 1984, JCoPh, 53, 484
Bernuzzi, S., & Hilditch, D. 2010, PhRvD, 81, 084003
Bishop, N. T., Gomez, R., Lehner, L., & Winicour, J. 1996, PhRvD, 54, 6153
Bishop, N. T., & Rezzolla, L. 2016, LRR, 19, 2
Bona, C., Ledvinka, T., Palenzuela, C., & Zacek, M. 2003, PhRvD, 67, 104005
Bona, C., Massó, J., Stela, J., & Seidel, E. 1996, in The Seventh Marcel Grossmann Meeting: On Recent Developments in Theoretical and Experimental General Relativity, Gravitation, and Relativistic Field Theories, ed. R. T. Jantzen, G. M. Keiser, & R. Ruffini (Singapore: World Scientific)
Brandt, S. R., Haas, R., Diener, P., et al. 2024, The Einstein Toolkit, the "Lev Landau" Release, ET_2024_05, Zenodo, doi:10.5281/zenodo.12588764
Brown, D., Diener, P., Sarbach, O., Schnetter, E., & Tiglio, M. 2009, PhRvD, 79, 044023
Bugner, M., Dietrich, T., Bernuzzi, S., Weyhausen, A., & Brügmann, B. 2016, PhRvD, 94, 084004
Brügmann, B., Gonzalez, J. A., Hannam, M., et al. 2008, PhRvD, 77, 024027
Campanelli, M., Lousto, C. O., Marronetti, P., & Zlochower, Y. 2006, PhRvL, 96, 111101
Cao, Z., & Hilditch, D. 2012, PhRvD, 85, 124032
Cao, Z., Yo, H.-J., & Yu, J.-P. 2008, PhRvD, 78, 124011
Clough, K., Figueras, P., Finkel, H., et al. 2015, CQGra, 32, 245011
Cook, W., Daszuta, B., Fields, J., et al. 2025, ApJS, 277, 3
Daszuta, B., & Cook, W. 2024, arXiv:2406.05126
Daszuta, B., Cook, W., Hammond, P., et al. 2024, arXiv:2406.09139
Daszuta, B., Zappa, F., Cook, W., et al. 2021, ApJS, 257, 25
Daverio, D., Dirian, Y., & Mitsou, E. 2018, arXiv:1810.12346
Deppe, N., Throwe, W., Kidder, L. E., et al. 2024, SpECTRE v2024.08.03, Zenodo, doi:10.5281/zenodo.13207712
Duez, M. D., & Zlochower, Y. 2019, RPPh, 82, 016902
Etienne, Z. B. 2024, PhRvD, 110, 064045
Felker, K. G., & Stone, J. M. 2018, JCoPh, 375, 1365
Fernando, M., Neilsen, D., Hirschmann, E., et al. 2022, in SC22: Int. Conf. for High Performance Computing, Networking, Storage and Analysis (Piscataway, NJ: IEEE),
Fernando, M., Neilsen, D., Lim, H., Hirschmann, E., & Sundar, H. 2019, SJSC, 41, C97
Fernando, M., Neilsen, D., Zlochower, Y., Hirschmann, E. W., & Sundar, H. 2023, PhRvD, 107, 064035
Fields, J., Zhu, H., Radice, D., et al. 2025, ApJS, 276, 35
Friedrich, H. 1985, CMaPh, 100, 525
Goodale, T., Allen, G., Lanfermann, G., et al. 2003, in Vector and Parallel Processing - VECPAR'2002, 5th International Conference, ed. J. M. L. M. Palma (Berlin: Springer), 197
Grete, P., Dolence, J. C., Miller, J. M., et al. 2022, Int. J. High Perform. Comput. Appl., 37, 465
Gundlach, C., Martin-Garcia, J. M., Calabrese, G., & Hinder, I. 2005, CQGra, 22, 3767
Hilditch, D. 2024, arXiv:2405.06035
Hilditch, D., Bernuzzi, S., Thierfelder, M., et al. 2013, PhRvD, 88, 084057
Hilditch, D., Weyhausen, A., & Brügmann, B. 2016, PhRvD, 93, 063006
Kalinani, J. V., Ji, L., Ennoggi, L., et al. 2025, CQGra, 42, 025016
Kidder, L. E., Field, S. E., Foucart, F., et al. 2017, JCoPh, 335, 84
Lehner, L. 2001, CQGra, 18, R25
Lehner, L., & Pretorius, F. 2014, ARA&A, 52, 661
Lindblom, L., Scheel, M. A., Kidder, L. E., Owen, R., & Rinne, O. 2006, CQGra, 23, S447
Lousto, C. O., Nakano, H., Zlochower, Y., & Campanelli, M. 2010, PhRvD, 82, 104057
Lovelace, G. 2021, Nat. Comput. Sci., 1, 450
Mewes, V., Zlochower, Y., Campanelli, M., et al. 2018, PhRvD, 97, 084059
Mewes, V., Zlochower, Y., Campanelli, M., et al. 2020, PhRvD, 101, 104007
Miller, J. M., & Schnetter, E. 2017, CQGra, 34, 015003
Miller, M. A. 2000, arXiv:gr-qc/0008017
Moxon, J., Scheel, M. A., & Teukolsky, S. A. 2020, PhRvD, 102, 044052
Moxon, J., Scheel, M. A., Teukolsky, S. A., et al. 2023, PhRvD, 107, 064013
Nakamura, T., Oohara, K., & Kojima, Y. 1987, PThPS, 90, 1
Nakano, H., Zlochower, Y., Lousto, C. O., & Campanelli, M. 2011, PhRvD, 84, 124006
Phoebus, 2022 Phoebus: Phifty one ergs blows up a star, GitHub, https://github.com/lanl/phoebus
Pollney, D., Reisswig, C., Schnetter, E., Dorband, N., & Diener, P. 2011, PhRvD, 83, 044045
Pretorius, F. 2005, PhRvL, 95, 121101
Pretto, I. G., Scheel, M. A., & Teukolsky, S. A. 2025, CQGra, 42, 075004
Punturo, M., Abernathy, M., Acernese, F., et al. 2010, CQGra, 27, 194002
Rashti, A., Bhattacharyya, M., Radice, D., et al. 2024, CQGra, 41, 095001
Reisswig, C., Bishop, N. T., Pollney, D., & Szilagyi, B. 2010, CQGra, 27, 075014
Reisswig, C., Haas, R., Ott, C. D., et al. 2013, PhRvD, 87, 064023
Reitze, D., Adhikari, R. X., Ballmer, S., et al. 2019, BAAS, 51, 35
Ruchlin, I., Etienne, Z. B., & Baumgarte, T. W. 2018, PhRvD, 97, 064036
Schnetter, E., Hawley, S. H., & Hawke, I. 2004, CQGra, 21, 1465
Shankar, S., Mösta, P., Brandt, S. R., et al. 2023, CQGra, 40, 205009
Shibata, M., & Nakamura, T. 1995, PhRvD, 52, 5428
Shibata, M., & Uryu, K. 2000, PhRvD, 61, 064001
Sperhake, U. 2007, PhRvD, 76, 104015
Stone, J. M., Mullen, P. D., Fielding, D., et al. 2024, arXiv:2409.16053
Stone, J. M., Tomida, K., White, C. J., & Felker, K. G. 2020, ApJS, 249, 4
Stout, Q. F., De Zeeuw, D. L., Gombosi, T. I., et al. 1997, in 1997 ACM/IEEE Conf. on Supercomputing, SC '97 (New York: Association for Computing Machinery), 1
Szilagyi, B., Lindblom, L., & Scheel, M. A. 2009, PhRvD, 80, 124010
Tichy, W., Ji, L., Adhikari, A., Rashti, A., & Pirog, M. 2023, CQGra, 40, 025004
Trott, C. R., Lebrun-Grandié, D., Arndt, D., et al. 2022, IEEE Trans. Parallel Distrib. Syst., 33, 805
Weyhausen, A., Bernuzzi, S., & Hilditch, D. 2012, PhRvD, 85, 024038
White, C. J., Stone, J. M., & Gammie, C. F. 2016, ApJS, 225, 22
Zhang, W., Almgren, A., Beckner, V., et al. 2019, JOSS, 4, 1370
Zhang, W., Myers, A., Gott, K., Almgren, A., & Bell, J. 2021, Int. J. High Perform. Comput. Appl., 35, 508