PAPER

The new discontinuous Galerkin methods based numerical relativity program Nmesh

IOP Publishing Classical and Quantum Gravity
Class. Quantum Grav. **40** (2023) 025004 (39pp) https://doi.org/10.1088/1361-6382/acaae7

The new discontinuous Galerkin methods based numerical relativity program `Nmesh`

Wolfgang Tichy[1,*], **Liwei Ji**[2], **Ananya Adhikari**[1], **Alireza Rashti**[3,4] **and Michal Pirog**[1]

[1] Department of Physics, Florida Atlantic University, Boca Raton, FL 33431, United States of America
[2] Rochester Institute of Technology, Rochester, NY 14623, United States of America
[3] Institute for Gravitation & the Cosmos, The Pennsylvania State University, University Park, PA 16802, United States of America
[4] Department of Physics, The Pennsylvania State University, University Park, PA 16802, United States of America

E-mail: wolf@fau.edu

Received 24 August 2022; revised 17 November 2022
Accepted for publication 12 December 2022
Published 23 December 2022

Abstract

Interpreting gravitational wave observations and understanding the physics of astrophysical compact objects such as black holes or neutron stars requires accurate theoretical models. Here, we present a new numerical relativity computer program, called `Nmesh`, that has the design goal to become a next generation program for the simulation of challenging relativistic astrophysics problems such as binary black hole or neutron star mergers. In order to efficiently run on large supercomputers, `Nmesh` uses a discontinuous Galerkin method together with a domain decomposition and mesh refinement that parallelizes and scales well. In this work, we discuss the various numerical methods we use. We also present results of test problems such as the evolution of scalar waves, single black holes and neutron stars, as well as shock tubes. In addition, we introduce a new positivity limiter that allows us to stably evolve single neutron stars without an additional artificial atmosphere, or other more traditional limiters.

Keywords: discontinuous Galerkin method, numerical relativity, neutron stars, positivity limiter, general relativistic hydrodynamics, black holes

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

* Author to whom any correspondence should be addressed.

1. Introduction

In August 2017, a binary neutron star merger has been observed by detecting its gravitational wave signal [1] together with an electromagnetic counterpart (across the whole electromagnetic spectrum) [2, 3]. This and similar observations [4–6] have started a new era of multi-messenger astronomy [7] and have opened a new window to the Universe, that allows us to measure and understand phenomena related to the equation of state (EoS) at supranuclear densities, the production of heavy elements via rapid neutron capture (r-process) nucleosynthesis, and cosmological constants [7–12].

Accurate theoretical models are required for creating gravitational wave and electromagnetic templates to interpret the observations, and to extract all the information contained in such signals about the properties of the binary. While there are analytical models to describe compact object coalescence, as long as the objects are well separated [13], the highly non-linear regime around the moment of merger is only accessible through simulations employing full numerical relativity (NR). To carry out such simulations, various computer programs have been developed, e.g. BAM [14–16], Einstein Toolkit [17, 18], NRPy+ [19], SACRA-MPI [20], and SpEC [21, 22]. Given that current detectors have a relatively high noise level, the numerical errors in these computer programs are not the main limiting factor when comparing observations and simulations.

However, the arrival of a new generation of detectors in the near future, like Cosmic Explorer [23], the DECi-hertz Interferometer Gravitational-wave Observatory (DECIGO) [24], Einstein Telescope [25], LIGO Voyager [26], the Laser Interferometer Space Antenna (LISA) [27], NEMO [28], and TianQin [29], will allow for observations with much higher signal-to-noise ratios. Therefore, in order to not bias the interpretation of the observed data, future NR computer programs will be required to better model micro-physics and also to deliver simulations with a higher accuracy. In principle higher accuracy can be achieved by current computer programs by simply increasing the resolution, while at the same time using more computational cores. This, however, will likely raise the computational cost to a level that is no longer affordable, because conventional NR computer programs do not scale well enough when we want to use hundreds of thousands of computational cores.

Consequently, a new campaign in the NR community is taking place to upgrade and develop computer programs that scale well and thus have a chance to achieve the accuracy needed for future observations. Examples of such next generation programs are BAMPS [30, 31], GRaM-X [32, 33], Dendro-GR [34], ExaHyPE [35] GR-Athena++ [36] GRChombo [37, 38] SpECTRE [39, 40], and SPHINCS_BSSN [41].

In this work, we present a new computer program, called `Nmesh`, that aims to be one of these next generation programs. One of the main features of `Nmesh` is its use of discontinuous Galerkin (DG) methods. The DG method for hyperbolic conservation laws has been introduced in [42–46]. Only in recent years it has been explored by the NR community to evolve the Einstein equations and the equations of general relativistic hydrodynamics [30, 39, 40, 47–52]. It has two main advantages when compared to more traditional finite difference or finite volume methods. First, when the evolved fields are smooth, a DG method can be exponentially convergent and thus much more efficient than traditional methods. Second, because of the way boundary conditions between adjacent domains are imposed within a DG method, there is less communication overhead when domains are distributed across many computational cores. This is expected to result in better scalability in a future where many groups will want to use hundreds of thousands or possibly even millions of computational cores.

The main purpose of this paper is to describe and test `Nmesh`. As will be discussed below, the main novelties when compared to other programs such as SpECTRE (as described in [40])

are a simplified treatment of the normal vectors and induced metric on domain boundaries, as well as certain positivity limiters that allow us to evolve single neutron stars without any additional limiters.

In section 2 we describe the DG method and other numerical methods we use. This is followed by a discussion of the effectiveness of our parallelization in section 3. In section 4 we test how well `Nmesh` can evolve various systems such as scalar waves, black holes, neutron stars, as well as some shock waves. We summarize and discuss our results in section 5. Throughout the article, we use geometric units, in which $G = c = 1$, as well as $M_\odot = 1$. Indices from the middle of the Latin alphabet, such as i, run from 1 to 3 and denote spatial indices, while indices from the beginning of the Latin alphabet, like a, and also Greek indices, such as μ, run from 0 to 3 and denote spacetime indices.

2. Numerical methods

In this section, we present the various numerical methods we use to perform simulations with hyperbolic evolution equations.

2.1. The discontinuous Galerkin method

In `Nmesh`, we use a DG method to discretize evolution equations. Often, these evolution equations come from general relativistic conservation laws of the form:

$$\nabla_\mu J^\mu = S, \tag{1}$$

where S is a possible source term. The covariant divergence on the left hand side can be written in terms of coordinate derivative using $\nabla_\mu J^\mu = \frac{1}{\sqrt{|g|}}\partial_\mu(\sqrt{|g|}J^\mu)$, where g is the determinant of the 4-metric $g_{\mu\nu}$. In terms of the standard 3+1 decomposition [53], the 4-metric is written as:

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

where γ_{ij} is the spatial metric on $t = const$ slices, and α and β^i are called lapse and shift. We can show that $\sqrt{|g|} = \alpha\sqrt{\gamma}$, where γ is the determinant of the 3-metric γ_{ij}. Thus, equation (1) is equivalent to:

$$\partial_t(\sqrt{\gamma}\alpha J^t) + \partial_i(\sqrt{\gamma}\alpha J^i) = \sqrt{\gamma}\alpha S. \tag{3}$$

Usually, we introduce the new variables:

$$u = \sqrt{\gamma}\alpha J^t, \quad f^i = \sqrt{\gamma}\alpha J^i, \quad s = \sqrt{\gamma}\alpha S, \tag{4}$$

so that equation (3) finally yields:

$$\partial_t u + \partial_i f^i = s. \tag{5}$$

Note that the flux vector f^i is usually a function $f^i(u)$, that depends on u. For brevity, we omit this dependence in most equations.

To discretize the spatial derivatives in equation (5), we first integrate against a test function ψ with respect to the coordinate volume d^3x over a certain region Ω. We obtain:

$$\int(\psi\partial_t u + \psi\partial_i f^i)d^3x = \int \psi s d^3x. \tag{6}$$

The second term is now integrated by parts using:

$$\int \psi \partial_i f^i d^3x = \oint \psi f^i n_i d^2\Sigma - \int f^i \partial_i \psi d^3x, \tag{7}$$

where $d^2\Sigma$ is the surface element for integrating over the boundary $\partial\Omega$, and n_i is normal to $\partial\Omega$. In the surface integral, the flux $f^i n_i$ at the boundary appears. To incorporate numerical boundary conditions, $f^i n_i$ in the surface integral over the boundary is replaced by the so-called numerical flux $(f^i n_i)^*$ (see section 2.3 below). It contains any information we need from the other side of the boundary (such as incoming characteristic modes). The replacement of $f^i n_i$ by $(f^i n_i)^*$ in the surface integral yields:

$$\begin{aligned}\int \psi \partial_i f^i d^3x &\to \oint \psi (f^i n_i)^* d^2\Sigma - \int f^i \partial_i \psi d^3x \\ &= \oint \psi [(f^i n_i)^* - f^i n_i] d^2\Sigma + \int \psi \partial_i f^i d^3x,\end{aligned} \tag{8}$$

where in the last step we have used integration by parts again to eliminate derivatives of ψ. With this replacement, equation (6) becomes:

$$\int (\psi \partial_t u + \psi \partial_i f^i) d^3x = \int \psi s d^3x - \oint \psi [(f^i n_i)^* - f^i n_i] d^2\Sigma. \tag{9}$$

We now introduce a coordinate transformation:

$$x^i = x^i(x^{\bar{i}}), \tag{10}$$

such that the volume we integrate over extends from -1 to $+1$ for all three $x^{\bar{i}}$ coordinates. In these coordinates, equation (9) reads:

$$\int \psi \left(\partial_t u + \frac{\partial x^{\bar{i}}}{\partial x^i} \partial_{\bar{i}} f^i \right) J d^3\bar{x} = \int \psi s J d^3\bar{x} - \oint \psi F d\bar{A}, \tag{11}$$

where we have defined:

$$F := (f^i n_i)^* - f^i n_i, \tag{12}$$

$$J = \left| \det \left(\frac{\partial x^i}{\partial x^{\bar{i}}} \right) \right|, \tag{13}$$

where J is called the Jacobian, and $d\bar{A}$ is the surface element on one of the six surfaces $x^{\bar{i}} = \pm 1$, but now expressed in $x^{\bar{i}}$ coordinates. For example on $x^{\bar{3}} = -1$,

$$d\bar{A} = \sqrt{{}^{(2)}\bar{\gamma}} dx^{\bar{1}} dx^{\bar{2}}, \tag{14}$$

where ${}^{(2)}\bar{\gamma}$ is the determinant of the 2-metric induced on this coordinate surface by the flat 3-metric δ_{ij}. The flat δ_{ij} results from our choice to integrate over the coordinate volume d^3x in equation (6), without including $\sqrt{\gamma}$. Below these x^i coordinates will be chosen to be Cartesian-like, so that they can cover all numerical domains.

Next, we expand both u and f^i in terms of Lagrange's characteristic polynomials:

$$l_q(\bar{x}) = \prod_{r=0, r\neq q}^{N} \frac{\bar{x}-\bar{x}_r}{\bar{x}_q-\bar{x}_r}, \tag{15}$$

so that, e.g.

$$u(\bar{x}) = \sum_{r_1=0}^{N}\sum_{r_2=0}^{N}\sum_{r_3=0}^{N} u_{r_1 r_2 r_3} l_{r_1}(x^{\bar{1}}) l_{r_2}(x^{\bar{2}}) l_{r_3}(x^{\bar{3}}). \tag{16}$$

The $\bar{x}_r$ are $N+1$ grid points that we choose in the interval $[-1,1]$. For the test function ψ, we use these same basis polynomials, i.e.:

$$\psi = l_{q_1}(x^{\bar{1}}) l_{q_2}(x^{\bar{2}}) l_{q_3}(x^{\bar{3}}). \tag{17}$$

The final step is to approximate all integrals in equation (11) using Gaußian quadrature (specifically Lobatto's Integration formula on p 888 of [54]), which, in one dimension, is given by:

$$\int_{-1}^{1} d\bar{x}\; g(\bar{x}) \approx \sum_{q=0}^{N} w_q g(\bar{x}_q), \tag{18}$$

here we use the $N+1$ Legendre Gauß-Lobatto grid points $\bar{x}_q$, that are defined as the extrema of the standard Legendre polynomial $P_N(\bar{x})$ in $\bar{x}\in[-1,1]$. The integration weights are then given by [54]:

$$w_q = \frac{2}{N(N+1)P_N(\bar{x}_q)^2}. \tag{19}$$

The integrals in equation (11) then turn into sums over products of $l_p(\bar{x})$, or products of $l_p(\bar{x})$ and its derivative. If we define:

$$(u,v) := \sum_{r=0}^{N} w_r u(\bar{x}_r) v(\bar{x}_r), \tag{20}$$

the products we encounter are (l_q, l_r) and $(l_q, \partial_{\bar{x}} l_r)$. Since,

$$l_q(\bar{x}_r) = \delta_{qr}, \tag{21}$$

we find:

$$(l_q, l_r) = w_q \delta_{qr}, \tag{22}$$

and

$$(l_q, \partial_{\bar{x}} l_r) = w_q \partial_{\bar{x}} l_r(\bar{x}_q) =: w_q D^{\bar{x}}_{qr}. \tag{23}$$

The differentiation matrix is given by:

$$D^{\bar{x}}_{qr} = \frac{c_r}{c_q}\frac{1}{\bar{x}_q-\bar{x}_r}, \tag{24}$$

where the Lagrange interpolation weights are:

$$c_q = \left[\prod_{s=0, s\neq q}^{N} (\bar{x}_q - \bar{x}_s)\right]^{-1}. \tag{25}$$

Putting all this together equation (11) becomes:

$$
\begin{aligned}
&w_{q_1} w_{q_2} w_{q_3} J_{q_1 q_2 q_3} \left(\partial_t u_{q_1 q_2 q_3} + \frac{\partial x^{\bar{1}}}{\partial x^i} \sum_{r=0}^{N} D^{\bar{1}}_{q_1 r} f^i_{r q_2 q_3} + \frac{\partial x^{\bar{2}}}{\partial x^i} \sum_{r=0}^{N} D^{\bar{2}}_{q_2 r} f^i_{q_1 r q_3} + \frac{\partial x^{\bar{3}}}{\partial x^i} \sum_{r=0}^{N} D^{\bar{3}}_{q_3 r} f^i_{q_1 q_2 r} \right) \\
&\quad = w_{q_1} w_{q_2} w_{q_3} J_{q_1 q_2 q_3} s_{q_1 q_2 q_3} - w_{q_2} w_{q_3} F_{q_1 q_2 q_3} \sqrt{{}^{(2)}\bar{\gamma}_{q_1 q_2 q_3}} (\delta_{q_1 0} + \delta_{q_1 N}) \\
&\qquad - w_{q_1} w_{q_3} F_{q_1 q_2 q_3} \sqrt{{}^{(2)}\bar{\gamma}_{q_1 q_2 q_3}} (\delta_{q_2 0} + \delta_{q_2 N}) - w_{q_1} w_{q_2} F_{q_1 q_2 q_3} \sqrt{{}^{(2)}\bar{\gamma}_{q_1 q_2 q_3}} (\delta_{q_3 0} + \delta_{q_3 N}),
\end{aligned}
\tag{26}
$$

where the Kronecker deltas on the right hand side come from $l_q(+1) = \delta_{qN}$ and $l_q(-1) = \delta_{q0}$. We now divide equation (26) by $w_{q_1} w_{q_2} w_{q_3} J_{q_1 q_2 q_3}$ and use the fact that:

$$
\sqrt{{}^{(2)}\bar{\gamma}}/J = \sqrt{\bar{\gamma}^{\bar{i}\bar{i}}}, \tag{27}
$$

holds on the surface $x^{\bar{i}} = const.$ Here $\bar{\gamma}^{\bar{i}\bar{i}}$ is a diagonal component of the inverse 3-metric obtained by transforming the flat 3-metric δ_{ij} from x^i-coordinates to $x^{\bar{i}}$-coordinates. This results in:

$$
\begin{aligned}
&\partial_t u_{q_1 q_2 q_3} + \sum_{r=0}^{N} \left(\frac{\partial x^{\bar{1}}}{\partial x^i} D^{\bar{1}}_{q_1 r} f^i_{r q_2 q_3} + \frac{\partial x^{\bar{2}}}{\partial x^i} D^{\bar{2}}_{q_2 r} f^i_{q_1 r q_3} + \frac{\partial x^{\bar{3}}}{\partial x^i} D^{\bar{3}}_{q_3 r} f^i_{q_1 q_2 r} \right) \\
&\quad = s_{q_1 q_2 q_3} - \frac{\sqrt{\bar{\gamma}^{\bar{1}\bar{1}}_{q_1 q_2 q_3}}}{w_{q_1}} F_{q_1 q_2 q_3} (\delta_{q_1 0} + \delta_{q_1 N}) - \frac{\sqrt{\bar{\gamma}^{\bar{2}\bar{2}}_{q_1 q_2 q_3}}}{w_{q_2}} F_{q_1 q_2 q_3} (\delta_{q_2 0} + \delta_{q_2 N}) \\
&\qquad - \frac{\sqrt{\bar{\gamma}^{\bar{3}\bar{3}}_{q_1 q_2 q_3}}}{w_{q_3}} F_{q_1 q_2 q_3} (\delta_{q_3 0} + \delta_{q_3 N}),
\end{aligned}
\tag{28}
$$

which is the version we use in `Nmesh`'s DG method. Notice that the derivation of this DG method has mostly followed the one introduced by Teukolsky in [48, 55] and tested extensively in [40], except for one important difference. Since we integrate over d^3x without including the determinant of the physical metric, we use the flat metric δ_{ij} when we construct $\bar{\gamma}^{\bar{i}\bar{i}}$ or when we normalize the normal vector n_i. This same n_i also enters the calculation of the fluxes and eigenvalues discussed in subsection 2.3. Recall that n_i arose in equation (7) after using Gauß's theorem. As shown in appendix A, Gauß's theorem can be used with any metric, as long as we normalize n_i with this same metric. The advantage of δ_{ij} is that it is constant and thus cannot have any discontinuities. Our approach simplifies the formalism as one does not have to worry about possible discontinuities in the physical metric or the normal vector across domain boundaries, which is an issue in the other approach [40]. In appendix B we discuss the difference between both approaches for the well understood case of an advection equation. We find that our approach together with the flux of equation (33) yields the correct upwind result, while the approach in [40] does not, if the physical metric is discontinuous across the boundary. However, the true physical solution of the Einstein equations is a continuous metric, even in the presence of shocks in the matter. Thus we expect that both approaches will converge to the same result, because any discontinuities in the metric should converge to zero. We have also tried both approaches by evolving a black hole, where the physical metric is far from flat, and where discontinuities in the physical metric arise due to numerical errors. We find no

important differences in the numerical solution or its rate of convergence. The discontinuities in the physical metric for this black hole case are described in section 4.2 and shown in figure 8.

Let us also consider the field u from equation (5) for the case where $s=0$. Then $\int u d^3x$ is exactly conserved. In formulations that directly use equation (3), and do not include $\sqrt{\gamma}$ in the field definition, the conserved quantity is $\int U\sqrt{\gamma}d^3x$, where $U=\alpha J^t$. In fact, for the exact solution of equation (5) both integrals are identical. However, once u and U are expressed in terms of basis functions (or equivalently in terms of values at grid points), the numerical (Gaußian quadrature) integrals over u and U will yield answers that differ at the level of the numerical truncation error. Nevertheless, a correct DG formulation will preserve these numerical integrals. Furthermore, any differences between the two numerical integrals will converge away with increasing resolution. Thus at any finite resolution the two approaches conserve different quantities, but in the continuum limit both converge to the same result.

2.2. *Evolution equations in non-conservative form*

So far, we have only considered evolution equations that can be written in conservative form as in equation (5), i.e. in terms of a flux vector f^i. However, the equations describing general relativistic gravity are often not available in this form. Rather they take the form:

$$\partial_t u + A^i(u)\partial_i u = s. \tag{29}$$

Note that here the matrix $A^i(u)$ depends on u itself. In this case, we can still integrate against a test function ψ, as before. The crucial introduction of a numerical flux in equation (8) now takes the form:

$$\begin{aligned}\int \psi A^i \partial_i u d^3x &\to \oint \psi\,(n_i A^i u)^* d^2\Sigma - \int u\partial_i(A^i\psi)d^3x \\ &= \oint \psi\,[(n_i A^i u)^* - n_i A^i u]d^2\Sigma + \int \psi A^i\partial_i u d^3x.\end{aligned} \tag{30}$$

Thus, the surface integral has almost the same form, with $(n_i A^i u)^*$ playing the role of the numerical flux. If we again expand in Lagrange's characteristic polynomials, and retrace our previous steps, we find the equivalent of equation (28). We obtain:

$$\begin{aligned}&\partial_t u_{q_1q_2q_3} + \sum_{r=0}^{N} A^i_{q_1q_2q_3}\left(\frac{\partial x^{\bar{1}}}{\partial x^i}D^{\bar{1}}_{q_1r}u_{rq_2q_3} + \frac{\partial x^{\bar{2}}}{\partial x^i}D^{\bar{2}}_{q_2r}u_{q_1rq_3} + \frac{\partial x^{\bar{3}}}{\partial x^i}D^{\bar{3}}_{q_3r}u_{q_1q_2r}\right) \\ &= s_{q_1q_2q_3} - \frac{\sqrt{\bar{\gamma}^{\bar{1}\bar{1}}_{q_1q_2q_3}}}{w_{q_1}}G_{q_1q_2q_3}(\delta_{q_10}+\delta_{q_1N}) - \frac{\sqrt{\bar{\gamma}^{\bar{2}\bar{2}}_{q_1q_2q_3}}}{w_{q_2}}G_{q_1q_2q_3}(\delta_{q_20}+\delta_{q_2N}) \\ &\quad - \frac{\sqrt{\bar{\gamma}^{\bar{3}\bar{3}}_{q_1q_2q_3}}}{w_{q_3}}G_{q_1q_2q_3}(\delta_{q_30}+\delta_{q_3N}),\end{aligned} \tag{31}$$

where G is defined by:

$$G := (n_i A^i u)^* - n_i A^i u. \tag{32}$$

When we compare with the surface term F defined in equation (12), appearing in the analog equation (28), we see that G can be obtained from F if we replace f^i by $A^i u$.

2.3. The numerical flux

In the interior of the domain, the flux vector $f^i(u)$ is simply computed from the field values u in the interior. However, the numerical flux that is used in the surface integral over the boundary is computed from the field values on both sides of the boundary. In many cases, we will use the Rusanov or local Lax-Friedrichs (LLF) flux. It is given by:

$$(f^i n_i)^* = \frac{1}{2}\left[f^i(u_{\rm in})n_i + f^i(u_{\rm adj})n_i + |\lambda|_{\rm max}\,(u_{\rm in} - u_{\rm adj})\right], \tag{33}$$

here n_i is the outward pointing normal to the boundary, $u_{\rm in}$ is the field value at the boundary using grid points that belong to the domain enclosed by the boundary, $u_{\rm adj}$ is the field value at the boundary using grid points that belong to the adjacent domain on the other side of the boundary, and $|\lambda|_{\rm max}$ is the absolute value of the eigenvalue of the characteristic mode with the largest eigenvalue magnitude, considering eigenvalues from both sides.

In the case where our system of equations takes the form of equation (29), we often also use another numerical flux, called the upwind flux. It is constructed from the orthonormalized eigenvectors and eigenvalues of the matrix $A^i n_i$ appearing in the surface term (32). Let the matrix S contain the eigenvectors as its columns. Then we can write:

$$A^i n_i = S\Lambda S^{-1}, \tag{34}$$

where Λ is a diagonal matrix that contains the corresponding eigenvalues. The eigenvectors with positive eigenvalues correspond to modes going along the direction on n_i, while the ones with negative eigenvalues correspond to modes going in the opposite direction. This means positive and negative eigenvalues are associated with modes that are outgoing and incoming through the boundary of the domain. As is usually the case, we wish to impose conditions only on the incoming modes. So, we define the upwind numerical flux that appears in equation (32), as:

$$(n_i A^i u)^* = (S(\Lambda^+ + \Lambda^-)S^{-1}u)^* := S(\Lambda^+ S^{-1}u_{\rm in} + \Lambda^- S^{-1}u_{\rm adj}), \tag{35}$$

here $\Lambda = \Lambda^+ + \Lambda^-$ with Λ^+ and Λ^- containing the positive and negative eigenvalues, and $u_{\rm in}$ and $u_{\rm adj}$ are the field values from the current domain and the adjacent domain.

2.4. Patches

To write equation (5) or (29), we use particular coordinates that are chosen to be Cartesian-like, and we call them $x^i = (x, y, z)$ in `Nmesh`. As already explained before we map these globally used Cartesian coordinates to local coordinates $x^{\bar{i}}$ via equation (10). This mapping is usually carried out in two steps. We first map them into a particular region or patch via:

$$x^i = x^i(X^j). \tag{36}$$

For example, we can use standard spherical coordinates $X^i = (r, \theta, \varphi)$ with a range $r \in [r_{\rm min}, r_{\rm max}]$, $\theta \in [\theta_{\rm min}, \theta_{\rm max}]$, $\varphi \in [\varphi_{\rm min}, \varphi_{\rm max}]$, so that we cover a certain section of a shell. Next we use:

$$X^i = \frac{1}{2}\left[(X^i_{\rm max} - X^i_{\rm min})\bar{X}^i + X^i_{\rm max} + X^i_{\rm min}\right], \tag{37}$$

to map each X^i into an $\bar{X}^i$ that has the standard range $\bar{X}^i \in [-1, 1]$. These $\bar{X}^i$ are what have been denoted by $x^{\bar{i}}$ in equation (10). Each patch is thus described by the particular transformation (36) and range we use for the X^i coordinates. In some cases, we only need Cartesian

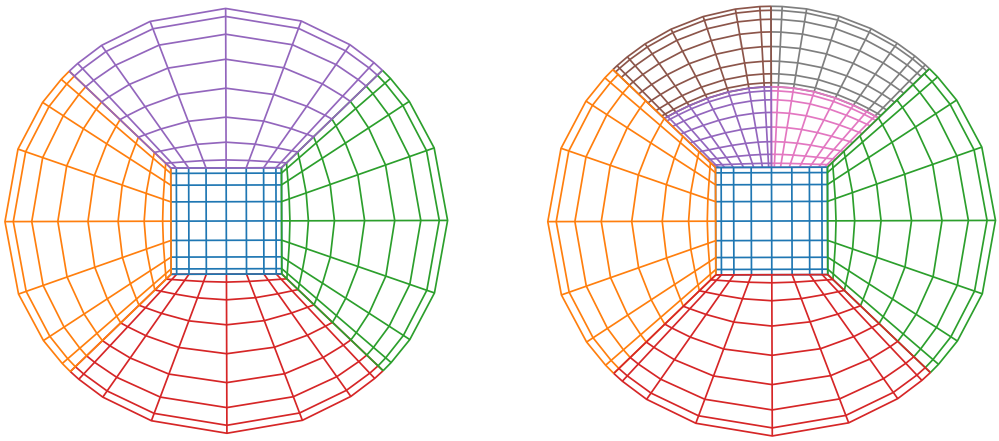

Figure 1. On the left side, we show a mesh which is made up of six cubed sphere patches that are arranged around one central cube. Five of these seven patches intersect the *xy*-plane and are shown in the picture. The right side shows the same patches as on the left. However, the top root node (that covers the entire top patch) has been h-refined so that this patch is now covered by eight child nodes, of which we show four in the *xy*-plane.

coordinates so that we use the identity transformation in equation (36), but we have also implemented the transformation to the cubed sphere coordinates $X^i = (\lambda, A, B)$ described in [56]. We then arrange our various patches such that they touch and cover the region of interest. An example is shown on the left side of figure 1. Here we have one central cube that is covered by Cartesian coordinates. This cube is surrounded by six cubed sphere patches. Five of these seven patches intersect the *xy*-plane and are shown on the left of figure 1. Note that each of the six cubed sphere patches shares one face with the central cube. The face on the opposite side is curved and arises by deforming one side of a larger cube into a spherical surface via a coordinate transformation of the form in equation (36). It thus comprises one sixth of the spherical outer boundary. The remaining faces of each cubed sphere patch touch four other cubed sphere patches, along flat surfaces. All patches are touching each other without any overlap, so that all interior patch faces have their entire face in common with one other patch face. In the next section we explain how information is exchanged via numerical fluxes between adjacent patches.

2.5. Adaptive mesh refinement

The patches described before can be directly covered with the Legendre Gauß-Lobatto grid points introduced above. Yet, to gain more flexibility, we can further refine each patch in `Nmesh`. This is achieved by identifying each patch with a so-called root node that can be further refined. When we refine this root node, we cut the original ranges of all three X^i-coordinates in half so that we end up with eight touching child nodes that now cover the original root node or patch. Each of these new nodes can then be further refined by again dividing it into eight child nodes. In this way we can refine each patch as often as we want. This can be done in an irregular way, where we further refine only the nodes in certain regions of interest. We end up with a node tree called an octree, where each node has either zero or eight child nodes. The nodes without children are called leaf nodes. Together, these leaf nodes cover the entire patch

and are thus the nodes in which we perform any calculations. For this reason, the leaf nodes are often called computational elements or just elements. However, in this paper, we will simply call them leaf nodes or just nodes. We note that, in the context of finite volume methods, the word 'node' is also sometimes used in the literature to denote a grid point. Yet, in this paper the word 'node' will always refer to a node in our octree.

The X^i-range of each leaf node is covered by grid points that correspond to the Legendre Gauß-Lobatto points discussed above. This means that we have grid points on each node face. This simplifies any calculations that depend on the values of fields on both sides of a node boundary. The number of grid points in each node can be freely chosen. When we increase it, we obtain higher order accuracy if the fields are smooth within the node. This is called p-refinement because increasing the number of grid points corresponds to an increase in the number of basis polynomials we use to represent a field within a node. Of course, p-refinement is most useful for smooth fields. For non-smooth fields it is often better to refine a node by splitting it into eight child nodes, which is known as h-refinement as it refines the resolution even if each child node has still the same number of grid points as the parent node. In `Nmesh` both p- and h-refinement can be performed whenever desired. Together we call this adaptive mesh refinement (AMR).

On the right of figure 1, we show an example where we h-refine the top node from the left side of figure 1. The resulting child nodes now cover the top patch. As we can see, the grid points of the h-refined nodes along, e.g. the left patch boundary no longer all coincide with the grid points of the unrefined node covering the left patch. This is a general phenomenon, whenever two neighboring nodes differ in their h- or p-refinement, many of the surface grid points of one node do not coincide with the surface grid points of the touching adjacent node. Furthermore, the surface of one node may be touching several adjacent nodes, as is the case for the node covering the left (orange) patch. This complicates the calculation of numerical fluxes such as equation (33) or (35) in one of our nodes, because we need both the fields u_{in} and u_{adj} at every surface grid point of the current node. We already have u_{in} at every point of the current node. However, u_{adj} only exists at the grid points of the adjacent nodes, which in general do not coincide with the surface grid points of the current node. To obtain u_{adj} at one of the surface grid points of the current node, we interpolate the u_{adj} data from the adjacent node onto this point. For this we currently use Lagrange interpolating polynomials constructed from the 2-dimensional surface data of the adjacent node. To easily find adjacent neighbors each node has an associated data structure that keeps track of all adjacent neighbor nodes. When p-refinement is applied to a node this data structure can remain unchanged since the size of the nodes and therefore the number of neighbors does not change. However, the data structure has to be updated whenever a node is h-refined. In this case the structure gets updated on this one node and on all of its neighbors. In this way `Nmesh` is able to accommodate arbitrary levels of h-refinement. For example, it is possible to h-refine a node into eight child nodes, and to then repeat this as often as desired with any of the child nodes, without at the same time refining any of the original neighbor nodes. In each case the final result is a number of touching leaf nodes that cover each patch. Since the patches themselves are also touching, the collection of all leaf nodes from all patches forms the mesh on which we perform our calculations.

The word Adaptive in AMR usually also implies an algorithm that automatically chooses p- or h-refinement. We currently have implemented only one such algorithm. Within each node, it expands a chosen quantity, such as the matter density ρ_0, in terms of Legendre polynomials. The coefficients in front of each of the Legendre polynomials can then be used to judge the smoothness of ρ_0. If ρ_0 is perfectly smooth, we expect the coefficients to fall off exponentially with increasing polynomial order. In our algorithm we then fit the logarithm of the coefficient magnitudes to a linear function. If the slope values b_i in all three directions ($i = 1, 2, 3$) of this

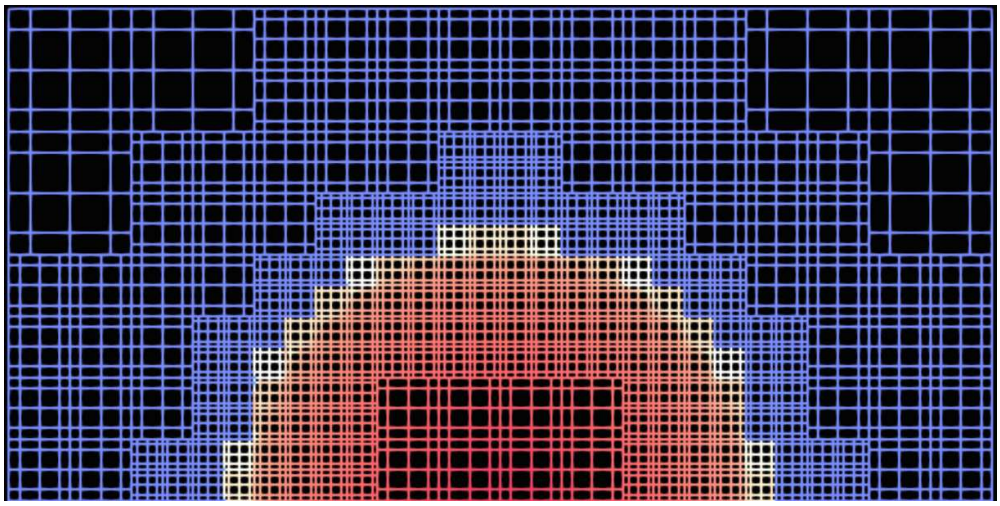

Figure 2. Leaf nodes near the neutron star surface are most refined. Inside the star (red region) and outside the star (blue region) less refinement is used.

linear function are not negative enough, we consider ρ_0 to be not smooth. We then h-refine the node. This algorithm is in principle geared toward dealing with the non-smooth behavior of matter across a neutron star surface. We have, however, not had any real success with this algorithm yet. We can turn it on and evolve neutron stars with it, but we have not been able to tune the parameters, that decide when the b_i are not negative enough, to values that work well after some matter leaves the star surface. We mention this particular algorithm here only to show that `Nmesh` has AMR capabilities in principle. We note, however, that these capabilities were not used in the simulations discussed below, where we use uniform h-refinement. Figure 2 shows the mesh for a single neutron star in a plane through its center. The different levels of h-refinement shown here are obtained from the coefficient drop off based algorithm mentioned above. It also refines neighbor nodes if their level of refinement is more than one below the just refined node. The purpose of the latter is to avoid abrupt changes in resolution, but is technically not required by `Nmesh`.

2.6. Time integration

Note that equations (28) or (31) still contain time derivatives, since up to this point we have only discretized spatial derivatives. This means that these equations represent a set of coupled ordinary differential equations for the fields $u_{q_1q_2q_3}$ at the grid points. In this paper, we use standard Runge–Kutta time integrators to find the solution of these ODEs from the $u_{q_1q_2q_3}$ at the initial time. Such Runge–Kutta methods are only stable when the Courant–Friedrichs–Lewy (CFL) condition is satisfied, i.e. if the time step Δt is small enough. In this work we use:

$$\Delta t = \Delta x_{\min}/v, \tag{38}$$

where $\Delta x_{\min}$ is the distance between the two closest grid points, and v is a number that needs to be greater than the largest characteristic speed.

2.7. Filters that can improve stability

Even when the time step satisfies the CFL condition, instabilities can still occur in some cases, e.g. on non-Cartesian patches. To combat such instabilities, we filter out high frequency modes in the evolved fields. This is achieved by first computing the coefficients $c_{l_1 l_2 l_3}$ in the expansion:

$$u(\bar{x})J = \sum_{l_1=0}^{N}\sum_{l_2=0}^{N}\sum_{l_3=0}^{N} c_{l_1 l_2 l_3} P_{l_1}(x^{\bar{1}}) P_{l_2}(x^{\bar{2}}) P_{l_3}(x^{\bar{3}}), \tag{39}$$

of each field u, where the $P_l(\bar{x})$ are Legendre polynomials. The coefficients are then replaced by:

$$c_{l_1 l_2 l_3} \to c_{l_1 l_2 l_3} e^{-\alpha_f (l_1/N)^s} e^{-\alpha_f (l_2/N)^s} e^{-\alpha_f (l_3/N)^s}, \tag{40}$$

and u is recomputed using these new coefficients. Note that we typically use $\alpha_f = 36$ and $s = 32$, so that the coefficients with the highest $l = N$ are practically set to zero, while all others are mostly unchanged.

3. Parallelization strategy

Modern supercomputers are made of thousands of compute nodes, each with on the order of 100 central processing unit (CPU) cores. Each compute node has its own separate memory (typically on the order of 100 GB), and cannot directly access data on other compute nodes. However, all compute nodes are connected by a network that allows data transfers between them. The by now traditional way to parallelize programs on such supercomputers is to use the message passing interface (MPI) library. With MPI, we start multiple processes (i.e. programs), each using its own piece of memory. Typically each process then works on a part of the problem that we wish to solve. The only way to exchange data is via messages sent between the different processes, hence the name MPI. Since no direct memory access occurs, MPI works very well if the processes run on different compute nodes that do not share any memory. Nevertheless, it is also possible to start multiple MPI processes within one compute node to take advantage of the presence of multiple CPU cores.

Systems consisting of black holes or neutron stars are governed by partial differential equations. To discretize them, we use the DG method together with AMR, as described above, so that the region of interest is covered by a number of leaf nodes (as described in section 2.5). We typically use several levels of h-refinement so that we end up with a large number of leaf nodes, possibly hundreds of thousands or even millions. The parallelization strategy of `Nmesh` is then to distribute these leaf nodes (referred to simply as nodes below) among the available MPI processes. To take one time step, we need to evaluate the various terms in equation (28) or (31). Notice that F and G in these equations depend on field values from the surface points of adjacent nodes via the numerical flux. Hence, MPI messages need to be sent to obtain these surface values. All other terms in equations (28) and (31) depend only on field values local to each node. Thus, we instruct MPI to start the transfer of the surface values. While this transfer is ongoing, we locally calculate all the terms in equation (28) or (31) besides F or G. This allows us to overlap communication and calculation, i.e. we avoid waiting for network transfers to arrive. Also note that the amount of data that needs to be sent via MPI is quite small, as the only values that need to be exchanged are from points on the surfaces of adjacent nodes. This is a significant advantage of DG methods compared to more traditional

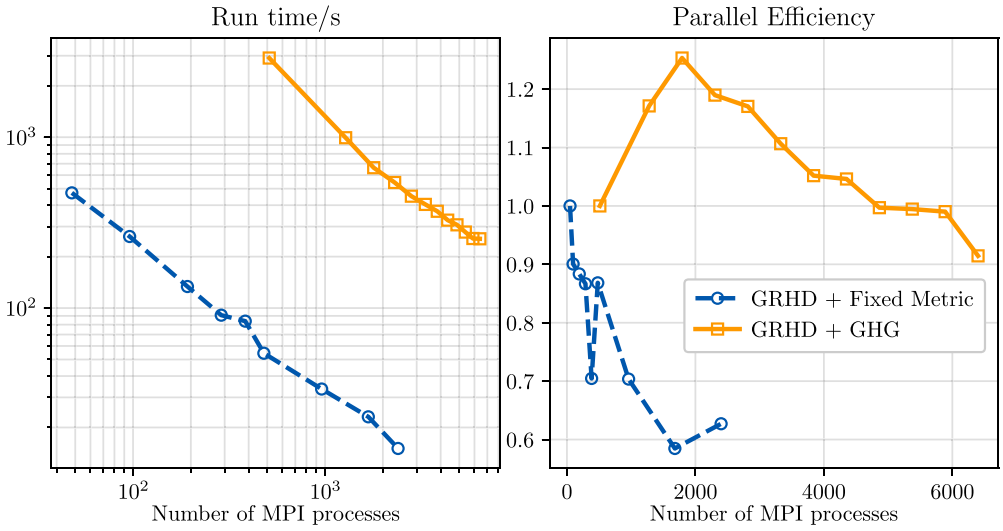

Figure 3. Strong scaling tests for the evolution of a neutron star using 262 144 leaf nodes with $5 \times 5 \times 5$ points. Circles indicate results obtained on the Cartesius supercomputer for a fixed spacetime metric. Squares show results on the Bridges-2 supercomputer, where the metric is evolved as well.

finite difference or finite volume methods. The latter two require transfer of data from a layer several points deep. The depth of this layer even increases when one increases the order of accuracy of the finite difference or finite volume method. Furthermore, if one uses coordinate patches, such as the cubed spheres as discussed above, one even needs data from more than just the six directly adjacent neighbor nodes (see [57]), because if we go several points deep in a curved coordinate direction, we may end up in yet another node. We thus expect our DG method to be more efficient.

To demonstrate the efficiency of `Nmesh` we have performed two strong scaling tests. On the left side of figure 3 we show the run time vs the number of MPI processes used. The circles correspond to the simulation of a single neutron star on a fixed spacetime metric on the Cartesius supercomputer. The squares are from the simulation of a single neutron star together with an evolving spacetime metric, that was performed on the Bridges-2 supercomputer. Perfect scaling corresponds to a linear speedup as this fixed size problem is evolved with more MPI processes. As we see on the left side of figure 3, our run time measurements almost follow a straight line, and thus indicate good scaling. To study the scaling further we also show the parallel efficiency on the right side of figure 3. This efficiency is computed from $t_r(1)/[nt_r(n)]$, where $t_r(n)$ is the run time measured using n MPI processes. Since a single MPI process cannot obtain enough memory for the simulations, $t_r(1)$ is estimated from $t_r(1) = n_{\min} t_r(n_{\min})$, where $n_{\min}$ is the run with the lowest number of MPI processes performed in each case.

Perfect scaling would correspond to a constant parallel efficiency. However, this is typically not achieved by real programs. In the case of `Nmesh` any sort of scaling will definitely stop once the number of MPI processes becomes comparable to the number of leaf nodes (here 262 144). In fact, we expect it to stop even before this, due to the growing communication overhead when more parallelization is used. The fact that the run with the evolving spacetime metric has a higher efficiency might be related to the fact that the evolution of the metric is time consuming, so that every MPI process does more work before communication is needed

again. On the other hand, Bridges-2 had newer hardware and MPI libraries than Cartesius, which could also account for part of the difference. The fact that our curves end at 2400 and 6400 MPI processes, is not due to any particular limitation of `Nmesh`. Rather, we currently do not have access to a machine that would allow us to use more cores. It thus remains to be seen up to which number of cores `Nmesh` will scale well. Nevertheless we consider our results encouraging. They confirm the expectation that a program based on a DG method should have good scaling.

4. Evolution system tests and results

In this section, we perform tests with several different evolution systems to validate our new `Nmesh` program. We also explain in detail which methods we use for our simulations of general relativistic hydrodynamics, and then show our results.

4.1. Scalar wave equation

One of the simplest systems one can evolve is a scalar wave. Here we consider a single scalar field obeying the wave equation:

$$\partial_t^2 \phi = \delta^{ij} \partial_i \partial_j \phi. \tag{41}$$

The DG method described earlier cannot be applied directly to systems with second order derivatives. We therefore introduce the extra variables:

$$\Pi := \partial_t \phi, \tag{42}$$

and

$$\chi_i := \partial_i \phi. \tag{43}$$

This results in the following system of first order equations:

$$\begin{aligned} \partial_t \Pi + \partial_j f^j_\Pi &= 0, \\ \partial_t \chi_i + \partial_j f^j_{\chi_i} &= 0, \\ \partial_t \phi &= \Pi. \end{aligned} \tag{44}$$

Here we have defined the flux vectors:

$$\begin{aligned} f^j_\Pi &:= -\chi_j, \\ f^j_{\chi_i} &:= -\Pi \delta^j_i, \\ f^j_\phi &:= 0. \end{aligned} \tag{45}$$

As we can see, the system in equation (44), consists of two coupled partial differential equations and one ordinary differential equation.

To evolve this system with our DG method, we also need to provide initial values and boundary conditions. Since,

$$\phi = \sin(k_i x^i - \omega t), \tag{46}$$

is a solution for any k_i and $\omega = \sqrt{\delta^{ij} k_i k_j}$, we initialize the system according to this equation at $t = 0$. We also use equation (46) at the outer boundary so that this sine wave is continuously entering through the outer boundary.

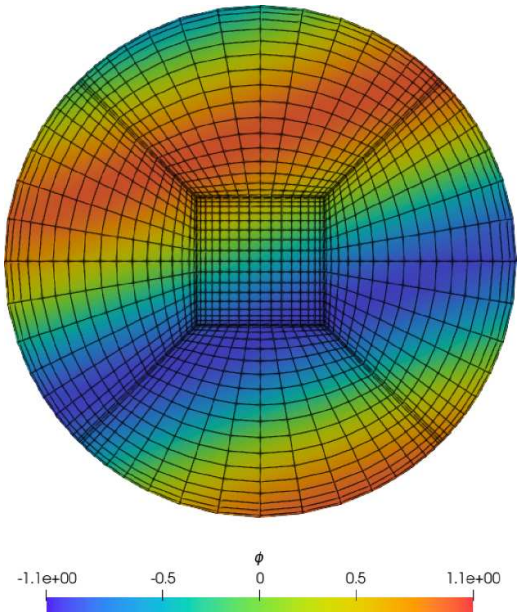

Figure 4. The plot shows the scalar field ϕ in terms of a colormap together with the mesh (in black) at evolution time $t = 4.5$. The mesh is made up of seven leaf nodes, five of which intersect the xy-plane shown here.

The DG method requires numerical fluxes. We have successfully used both the LLF flux of equation (33) as well as the upwind flux of equation (35). To impose equation (46) at the outer boundary, we use the same numerical flux as in the interior, but we set $u_{\rm adj}$ to the value coming from equation (46).

The wave vector k_i in equation (46) is arbitrary. For the test results presented here, we choose $k_i = (0.7, -2, 4.3)$, so that it represents the general case where k_i is not aligned with any coordinate direction. As we can see in figure 4, we get a sinusoidal wave that propagates through our numerical domain, with this k_i vector. In this case we have used seven patches and the figure shows the scalar field ϕ in the xy-plane after evolving up to time $t = 4.5$. For the test case depicted in figure 4, we have used an equal number of grid points ($19 \times 19 \times 19$) in all directions, without any h-refinement applied to the root nodes. However, we have also performed tests with an unequal number of grid points and with the root nodes h-refined. We have obtained stable evolution for the system for all these cases with both the LLF and upwind numerical fluxes. The choice of numerical flux did not have any significant effect in any of the scalar wave evolution tests, as both cases yield results that have errors of the same order.

To demonstrate the convergence of our new `Nmesh` program in this scalar wave evolution test, we have applied p-refinement and h-refinement separately. In figure 5 we plot the L2-norm of the error in ϕ for both. For the case of p-refinement (top) we increase the number of points n in all directions, setting them to $n = 10, 12, 14, 16, 18, 20$, with no h-refinement applied to the root nodes. We observe an exponential drop in the L2-norm error, as expected in this case. For the case of h-refinement (bottom), we apply $l = 0, 1, 2, 3, 4, 5$ levels of refinement to the root nodes, and have $n = 10$ points in each direction in each node. Again, we observe convergent behavior for the L2-norm of the error, when increasing the number of h-refinement levels. Figure 5 only shows the results for the LLF numerical flux, since the results for the

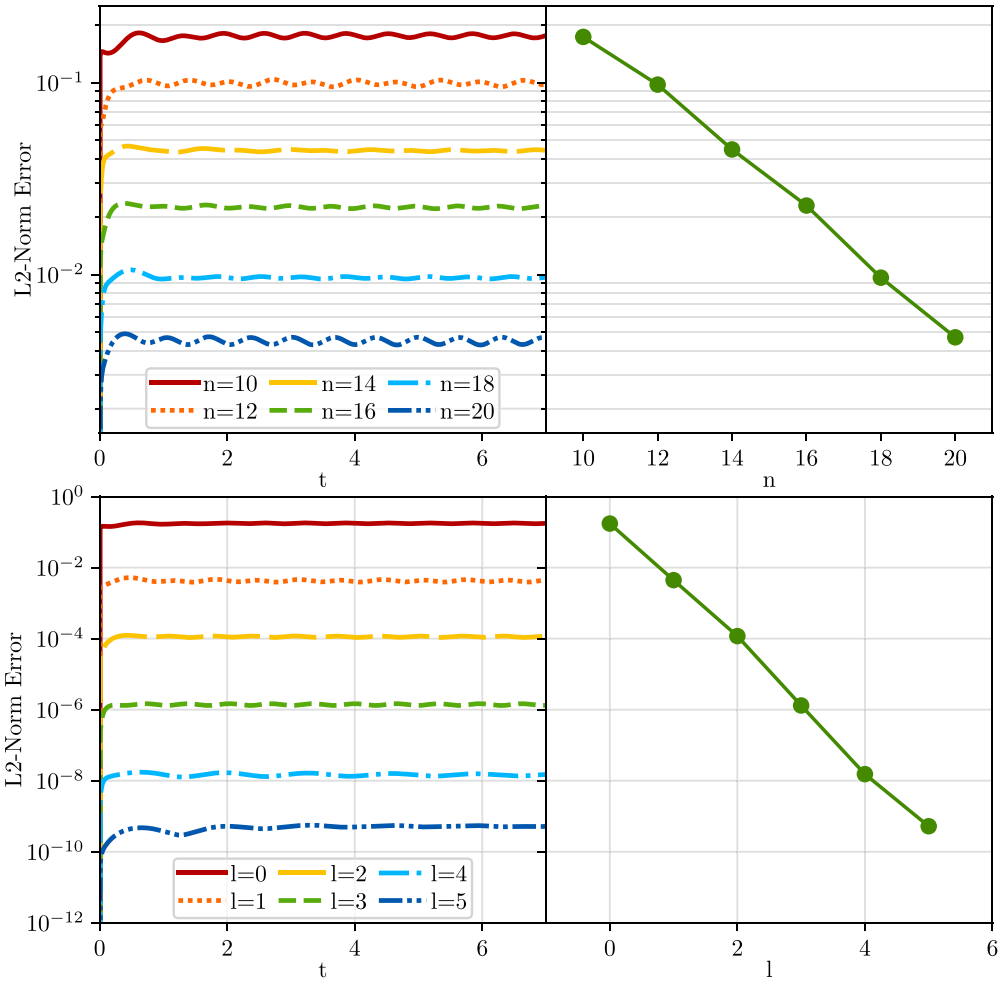

Figure 5. This plot shows the change in the L2-norm of the error in the scalar field ϕ for two different cases when using the LLF numerical flux. First, p-refinement is applied (without any h-refinement) and we plot the error vs time (top left) and then the error vs number of points n along each axis in a node at time $t = 3$ (top right). Next, h-refinement is applied, and we plot the error vs time (bottom left) and then the error vs levels of refinement l applied to the root node at time $t = 3$ (bottom right). In this case, we keep the number of points in each direction in each node fixed at $n = 10$.

upwind flux are very similar. For all the runs shown in figures 4 and 5, the time step was set to $\Delta t = \Delta x_{\min}/5$ in accordance with equation (38).

Even though all the convergence results stated above have been obtained for a mesh using six cubed sphere patches that surround one central cube, we also obtain convergence for a mesh covered by a single cubic Cartesian patch. The key difference between these is that we need the filters of equation (40) to stabilize the evolution on meshes that contain both Cartesian and cubed sphere patches, while this is not necessary on a purely Cartesian patch, even if it is h-refined. For the runs of figures 4 and 5 we set the filter parameters of equation (40) to the values $\alpha_{\mathrm{f}} = 36$ and $s = 32$.

4.2. Convergence tests with the GHG system for a single excised black hole

For the gravitational part, we have implemented the first-order reduction of generalized harmonic gauge (GHG) formulation [31, 58]:

$$\partial_t g_{ab} - (1+\gamma_1)\beta^k \partial_k g_{ab} = -\alpha \Pi_{ab} - \gamma_1 \beta^k \Phi_{kab}, \tag{47}$$

$$\begin{aligned}
&\partial_t \Pi_{ab} - \beta^k \partial_k \Pi_{ab} + \alpha \gamma^{ki} \partial_k \Phi_{iab} - \gamma_1 \gamma_2 \beta^k \partial_k g_{ab} \\
&\quad = 2\alpha g^{cd} \left(\gamma^{ij} \Phi_{ica} \Phi_{jdb} - \Pi_{ca} \Pi_{db} - g^{ef} \Gamma_{ace} \Gamma_{bdf} \right) - 2\alpha \nabla_{(a} H_{b)} - \frac{1}{2} \alpha n^c n^d \Pi_{cd} \Pi_{ab} \\
&\qquad - \alpha n^c \Pi_{ci} \gamma^{ij} \Phi_{jab} + \alpha \gamma_0 \left[2\delta^c{}_{(a} n_{b)} - g_{ab} n^c \right] (H_c + \Gamma_c) - \gamma_1 \gamma_2 \beta^k \Phi_{kab} \\
&\qquad - 16\pi\alpha \left(T_{ab} - \frac{1}{2} g_{ab} g^{cd} T_{cd} \right),
\end{aligned} \tag{48}$$

$$\begin{aligned}
&\partial_t \Phi_{iab} - \beta^k \partial_k \Phi_{iab} + \alpha \partial_i \Pi_{ab} - \alpha \gamma_2 \partial_i g_{ab} \\
&\quad = \frac{1}{2} \alpha n^c n^d \Phi_{icd} \Pi_{ab} + \alpha \gamma^{jk} n^c \Phi_{ijc} \Phi_{kab} - \alpha \gamma_2 \Phi_{iab},
\end{aligned} \tag{49}$$

here g_{ab} is the spacetime metric, n_a is the unit normal to the hypersurface of constant coordinate time t, and $\Gamma_a = g^{bc}\Gamma_{abc}$ is the contracted Christoffel symbol. The equations are written in terms of the extra variables $\Pi_{ab} := -n^c \partial_c g_{ab}$ and $\Phi_{iab} := \partial_i g_{ab}$, that have been introduced to make the original second-order GHG system first-order in both time and space. Gauge conditions in the GHG system are specified by prescribing the gauge source function H_a. The lapse α, shift β^i and spatial metric γ_{ij} come from the $3+1$ decomposition in equation (2). The GHG evolution equations also contain extra terms that are multiplied with the parameters γ_0, γ_1, and γ_2. In this paper we set $\gamma_1 = -1$, and choose $\gamma_2 = \gamma_0 = 1$ for the constraint damping parameters.

To test the gravitational part of `Nmesh`, we evolve a black hole spacetime. As initial data, we choose the metric of a single Schwarzschild black hole in Kerr-Schild coordinates [59],

$$g_{ab} = \eta_{ab} + \frac{2M}{r} l_a l_b, \tag{50}$$

where η_{ab} is the Minkowski metric, and M is the mass of the black hole. In the Cartesian coordinates, $r = (x^2+y^2+z^2)^{1/2}$, and $l_a = (1, x/r, y/r, z/r)$. The gauge source function is initialized based on the above metric (50), and is left constant during the simulation [60],

$$H_a(t=0) = -\Gamma_a(t=0), \quad \partial_t H_a = 0. \tag{51}$$

With this initial condition, the analytic solution of the evolution equations is simply the static Schwarzschild metric, so that all evolved fields should be constant. Of course evolution will lead to some amount of numerical errors. Thus we test here if `Nmesh` can stably evolve this setup and whether the numerical evolution will settle down to a stable state.

As computational domain, we choose a spherical shell that extends from $r = 1.8M$ to $11.8\,M$, and is covered by six cubed sphere patches. The inner boundary is thus inside the black hole horizon of $r = 2M$. The speeds of all characteristic modes at the inner boundary are such that every mode is moving toward the black hole center and thus leaving the computational domain. We therefore do not impose any boundary conditions at the inner boundary. The situation at the outer boundary is different as we have both incoming and outgoing modes. We impose no condition on the outgoing modes, but we keep the incoming modes constant at their initial values (consistent with the static analytic solution). This is done using equation (35),

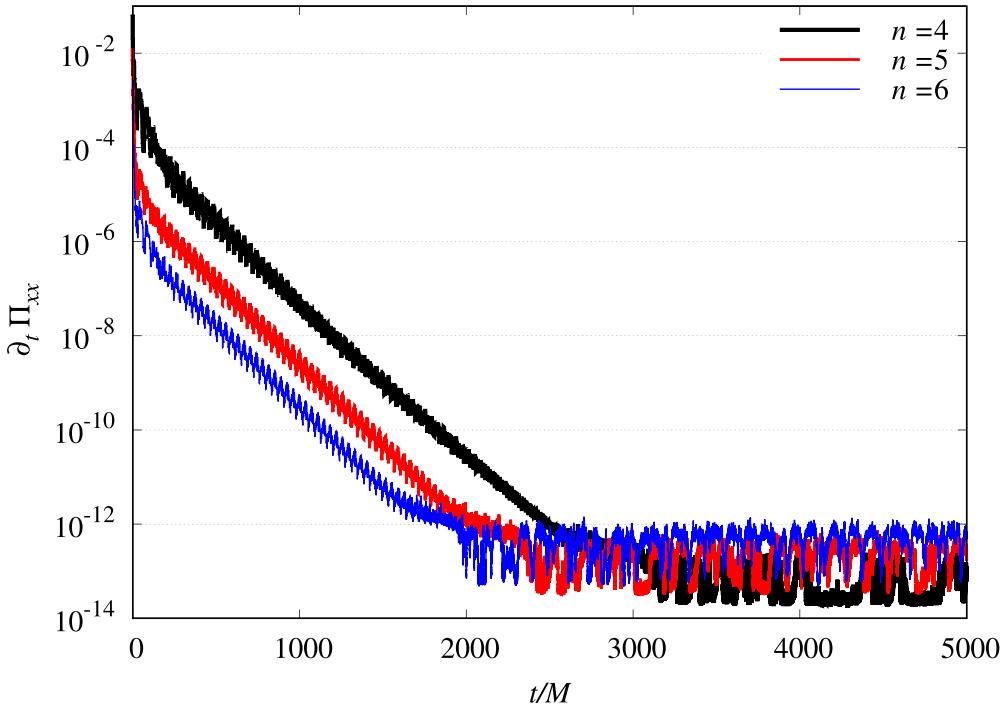

Figure 6. Time evolution of the time derivative of Π_{xx} for a black hole in Kerr–Schild coordinates. Here, we use n grid points in each direction in each leaf node, and have h-refined each of the root nodes three times. We use the upwind flux.

where $u_{\rm adj}$ is set to the analytic Schwarzschild solution, $u_{\rm in}$ are the evolved fields at the boundary, and S, Λ^+, and Λ^- come from the characteristic modes and their eigenvalues [58], calculated from normals that are normalized with respect to the flat metric. To improve the accuracy we use either 2 or 3 levels of h-refinement in each patch. We choose the time step according to equation (38), with $\nu = 4$. We find that, with this setup, no filters are necessary to stabilize our runs. As in the scalar field test cases discussed above, filters only become necessary when both Cartesian and cubed sphere patches are present. In the latter case, the filter of equation (40) is again sufficient for obtaining stable runs. We have evolved this setup using both the LLF and upwind fluxes of equations (33) and (35) at inter domain boundaries. As described below, both fluxes work about equally well.

In order to demonstrate stability of our runs at high resolution, we have evolved the black hole with three levels of h-refinement for three different numbers of grid points. We find that the system of equations reaches a state where the time derivatives $\partial_t g_{ab}$, $\partial_t \Pi_{ab}$, and $\partial_t \Phi_{iab}$ all approach zero (up to machine precision), as expected for a static black hole. As an example, we show $\partial_t \Pi_{xx}$ in figure 6 when evolved with $4 \times 4 \times 4$, $5 \times 5 \times 5$, and $6 \times 6 \times 6$ grid points in each node. As we can see, this time derivative falls exponentially until it settles down to below 10^{-12}. Beyond this point, the terms that determine the time derivatives in the GHG evolution equations (47)–(49) add up to almost zero, and deviate from zero only because of roundoff errors due to the use of floating point numbers. As expected, at higher resolution this steady state is reached earlier, because our numerical method then has smaller discretization errors.

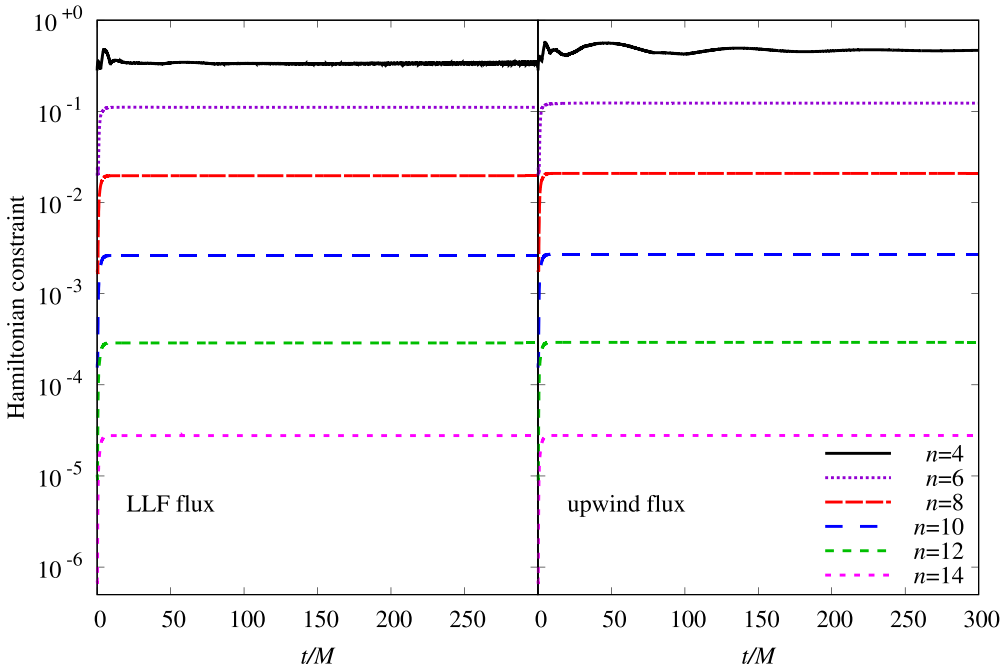

Figure 7. The infinity norm of the Hamiltonian constraint, when evolving a black hole in Kerr-Schild coordinates. On the left, the constraint evolution is shown, when using the LLF flux. On the right are the corresponding results for the upwind flux. All the simulations on both sides use grids with 6 patches, and each patch is refined uniformly twice. The different lines correspond to different numbers of grid points n in each direction in each leaf node. The Hamiltonian constraint converges to zero exponentially as we increase n.

The Hamiltonian and momentum constraints of general relativity read as:

$$H = R - K_{ij}K^{ij} + K^2 - 16\pi\rho_{\text{ADM}}, \tag{52}$$

$$M^i = D_j(K^{ij} - \gamma^{ij}K) - 8\pi\, j^i, \tag{53}$$

where R and D_j are the Ricci scalar and derivative operator associated with the 3-metric γ_{ij}, $K_{ij} = -\frac{1}{2}\pounds_n\gamma_{ij}$, $\rho_{\text{ADM}} = T_{ab}n^a n^b$, and $j^i = -T_{ab}n^a\gamma^{bi}$ (see e.g. [61]). General relativity dictates $H = M^i = 0$ for all time. In figure 7, we show the infinity norm of H over the grid when we evolve the black hole with 2 levels of h-refinement for various numbers of grid points per node. As we can see, H stabilizes after a short time and then stays practically constant, which again indicates stability. As expected H converges to zero exponentially as we increase the number of grid points. We also see that the results for the LLF flux (on the left) and the upwind flux (on the right) are almost the same. In our simulations the momentum constraint M^i behaves just like H and also converges to zero exponentially as we increase the number of grid points.

Since the initial data is given by the Kerr-Schild metric, the analytic solution is a static black hole. The numerical solution, however, evolves for a while until it settles down into a stable configuration (see figure 6). This happens because the analytic solution does not exactly satisfy the discretized GHG equations. As already mentioned in section 2 we use domain normals that are normalized with respect to the flat metric for our black hole evolutions. In figure 8

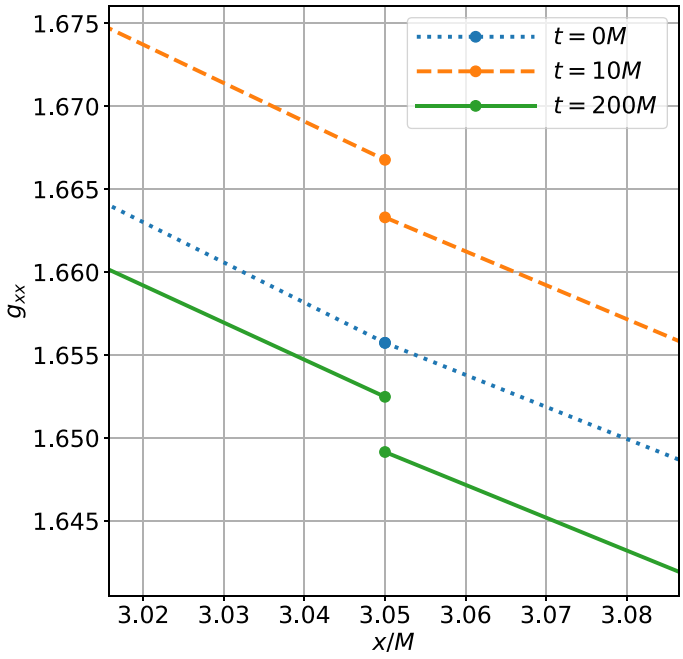

Figure 8. The plot shows the metric component g_{xx} at different times, along a portion of the x-axis near a domain boundary located at $x = 3.05\,M$. The initial data at $t = 0M$ are continuous (dotted line). By $t = 10\,M$ a discontinuity (dashed line) has developed. The metric still rapidly evolves for another $100\,M$ and then stabilizes around $t = 200M$, while keeping the size of the discontinuity roughly constant. Any further metric changes after $t = 200\,M$ are so small that they are practically indistinguishable from the solid line.

we plot the metric component g_{xx} at different times (for the $n = 4$ case of figure 6). We have zoomed in onto a region close to a domain boundary that is in the strong field region close to the horizon at $x = 2M$. We find that the metric rapidly evolves away from the continuous Kerr–Schild initial data. During this rapid evolution discontinuities develop across domain boundaries due to numerical errors. These discontinuities persist throughout the evolution, but do not negatively affect its stability or convergence, even though dynamic evolution takes place. This indicates that such discontinuities in the physical metric are not a problem for a DG method that uses numerical fluxes and eigenvalues, which are computed from normals that are normalized with respect to the flat metric.

4.3. General relativistic hydrodynamics

To treat neutron star matter we use the Valencia formulation [62]. Matter is thus described as a perfect fluid, where the stress-energy tensor is given by:

$$T^{\mu\nu} = (\rho + P)\, u^{\mu} u^{\nu} + P g^{\mu\nu}, \tag{54}$$

here ρ is the energy density, P is the pressure, and u^{μ} is the four-velocity. The total energy density is written as:

$$\rho = \rho_0 (1 + \epsilon), \tag{55}$$

where ρ_0 is the rest-mass energy density, and ϵ is the specific internal energy. We express the four-velocity u^μ in terms of the three-velocity given by:

$$v^\mu = u^\mu/W - n^\mu, \tag{56}$$

and also introduce the Lorentz factor:

$$W = -n_\mu u^\mu = \alpha u^0. \tag{57}$$

Together $(\rho_0, \epsilon, Wv^i, P)$ are known as the primitive variables.

The matter equations follow from the conservation law for the energy-momentum tensor and the conservation law for the baryon number. In order to obtain flux conservative evolution equations of the form (5), one introduces the conserved variables:

$$D = \rho_0 W, \tag{58}$$

$$\tau = \rho_0 h W^2 - P - \rho_0 W, \tag{59}$$

$$S_i = \rho_0 h W^2 v_i, \tag{60}$$

here D is rest-mass density, τ the internal energy density, S_i the momentum density as seen by Eulerian observers. The last two equations also contain the specific enthalpy given by:

$$h = 1 + \epsilon + P/\rho_0. \tag{61}$$

The conserved variables are then:

$$u = \sqrt{\gamma}\begin{pmatrix} D \\ \tau \\ S_l \end{pmatrix}. \tag{62}$$

They satisfy equation (5), with the flux vectors and sources given by:

$$f^i = \sqrt{\gamma}\begin{pmatrix} (\alpha v^i - \beta^i)D \\ (\alpha v^i - \beta^i)\tau + \alpha P v^i \\ (\alpha v^i - \beta^i)S_l + \alpha P \delta^i_l \end{pmatrix} \tag{63}$$

and

$$\frac{s}{\sqrt{\gamma}} = \begin{pmatrix} 0 \\ T^{00}(\beta^i\beta^j K_{ij} - \beta^i\partial_i\alpha) + T^{0i}(2\beta^j K_{ij} - \partial_i\alpha) + T^{ij}K_{ij} \\ T^{00}(\frac{\beta^i\beta^j}{2}\partial_l\gamma_{ij} - \alpha\partial_l\alpha) + T^{0i}\beta^j\partial_l\gamma_{ij} + T^0_i\,\partial_l\beta^i + \frac{T^{ij}}{2}\partial_l\gamma_{ij} \end{pmatrix}. \tag{64}$$

The components of the stress-energy tensor appearing here can be expressed in terms of the primitive variables as:

$$T^{00} = (W^2 h\rho_0 - P)/\alpha^2, \tag{65}$$

$$T^{0i} = Wh\rho_0 u^i/\alpha + P\beta^i/\alpha^2, \tag{66}$$

$$T^{ij} = h\rho_0 u^i u^j + P(\gamma^{ij} - \beta^i\beta^j/\alpha^2), \tag{67}$$

$$T^0_i = h\rho_0 W^2 v_i/\alpha, \tag{68}$$

where

$$u^i = Wv^i - W\frac{\beta^i}{\alpha}. \tag{69}$$

To close the evolution system, we have to specify an EoS for the fluid, i.e. an equation of the form:

$$P = P(\rho_0, \epsilon), \tag{70}$$

that allows us to obtain the pressure for a given rest-mass energy density and the specific internal energy, as well as the sound speed squared c_s^2. If $c_s^2 < 0$ or $c_s^2 > 1$, we set it to zero. We also set it to zero if $\rho_0 = 0$ or $h = 0$.

As numerical flux we use the LLF flux of equation (33). For this, we need the eigenvalues of the characteristic modes given by [62]:

$$\lambda_1 = \alpha \frac{v^n(1-c_s^2) + \sqrt{C^2}}{1 - v^2 c_s^2} - \beta^n, \tag{71}$$

$$\lambda_2 = \alpha \frac{v^n(1-c_s^2) - \sqrt{C^2}}{1 - v^2 c_s^2} - \beta^n, \tag{72}$$

$$\lambda_3 = \lambda_4 = \lambda_5 = \alpha v^n - \beta^n, \tag{73}$$

where $C^2 = c_s^2(1-v^2)[\gamma^{mn}(1-v^2c_s^2) - v^n v^n(1-c_s^2)]$, $v^n = v^i n_i$, and n_i is the normal to the interface, normalized with respect to the flat metric. At points where $1 - v^2 c_s^2 = 0$ or $C^2 < 0$, we simply set $\lambda_1 = \lambda_2 = 0$.

4.3.1. Converting conserved to primitive variables. As already mentioned, we formulate the matter equations in the flux conservative form of equation (5) in terms of the conserved variables in equation (62). However, the flux vectors and sources in equations (63) and (64) also depend on the primitive variables ρ_0, ϵ, Wv^i, P. Thus we need a way to compute the primitive variables form the conserved variables. This is done with the help of a root finder that we will describe next. Note that we use Wv^i as our primitive velocity variable instead of v^i. The advantage is that Wv^i is allowed to take any real value, while v^i is bounded by the speed of light. The latter is inconvenient in numerical calculations as numerical inaccuracies can often violate the light speed bound.

The method we use closely follows the approach in appendix C of [63], i.e. we will try to find:

$$Wv := \sqrt{Wv^i\, Wv_i}, \tag{74}$$

with the help of a root finder. This root is given by the zero of the function:

$$f(Wv) = Wv - \frac{\sqrt{S_i S^i}}{D h(Wv)}. \tag{75}$$

Here, in order to find $h(Wv)$, we first need to compute the following:

$$W = \sqrt{1 + (Wv)^2}, \tag{76}$$

$$\rho_0(Wv) = \frac{D}{W}, \tag{77}$$

$$\epsilon(Wv) = W\frac{\tau}{D} - Wv\frac{\sqrt{S_i S^i}}{D} + \frac{(Wv)^2}{1+W}, \tag{78}$$

$$P(Wv) = P(\rho_0(Wv), \epsilon(Wv)) \tag{79}$$

$$a(Wv) = \frac{P(Wv)}{\rho_0(Wv) + \rho_0(Wv)\epsilon(Wv)}, \tag{80}$$

$$h(Wv) = [1 + \epsilon(Wv)][1 + a(Wv)]. \tag{81}$$

Note that our implementation of the EoS $P(\rho_0, \epsilon)$ gracefully handles cases where ϵ is slightly negative. Nevertheless if $\epsilon(Wv) < 0$ we set it to zero when calculating $a(Wv)$ and $h(Wv)$.

After we have obtained the primitive variables, we calculate $Z^i = S^i/(Wh\rho_0)$ and $Z = \sqrt{Z^i Z_i}$. According to equation (60), we should have $Z^i = Wv^i$. However, due to numerical errors, the latter equality will only hold up to the accuracy goal specified for the root finder (typically, the root finder has a relative error of 10^{-10}). If $Z \leqslant Wv$, we accept this small discrepancy, but if $Z > Wv$, we scale both S_i and Wv^i by a factor of Wv/Z.

4.3.2. A positivity limiter for low density regions. We use a strong stability preserving third order Runge–Kutta scheme [64] to evolve the conserved variables. It is possible that the conserved variables become unphysical after a Runge–Kutta substep due to numerical errors. By unphysical, we mean points where the mass density D or the energy density τ is negative, or where $S > D + \tau$, with $S = \sqrt{S_i S^i}$. If this happens, it also becomes impossible to then find the primitive variables needed for the next Runge–Kutta substep. To combat this problem we use so-called positivity limiters after each substep. The idea of these limiters is to scale each conserved variable u, that we desire to limit, towards its node average $\bar{u}$ using:

$$u \to \bar{u} + \theta_u \cdot (u - \bar{u}), \tag{82}$$

here $0 \leqslant \theta_u \leqslant 1$, and u can be D, τ or S_i. For each we try to find the maximum θ_u, such that u satisfies certain criteria. For D, the criterion is $D \geqslant 10^{-12}\rho_{0,\max}$, where $\rho_{0,\max}$ is the maximum mass density. For τ, we simply demand $\tau \geqslant 0$, while the S_i criterion is $S < D + \tau$. All three criteria have to hold at each point of the node. Of course even with the lowest allowed value of $\theta_u = 0$, it is possible that some of the three criteria are still not met at some points. This occurs if $\bar{D} < 10^{-12}\rho_{0,\max}$ or $\bar{\tau} < 0$. In this case we replace $\bar{D}$ or $\bar{\tau}$ in equation (82) by these limits. If $\bar{S} > \bar{D} + \bar{\tau}$ we reduce the magnitude of the vector S_i by a factor of $(\bar{D} + \bar{\tau})/\bar{S}$ to meet this criterion. Notice that we do not use an artificial atmosphere as, e.g. in [15, 40, 63, 65–70]. Rather the positivity limiters described above ensure that $D \geqslant 10^{-12}\rho_{0,\max}$, $\tau \geqslant 0$, and $S < D + \tau$. In some sense that gives us an atmosphere as well, as D can never drop below this minimum. Yet, since in most cases scaling towards the average suffices to satisfy all three criteria, we do not violate mass, energy or momentum conservation in most cases. And even in cases where we reset D, τ or S_i in some node, and thus violate conservation, we usually need to modify only one of these conserved variables, while the usual artificial atmosphere treatment would set D to an atmosphere value and also zero both τ and S_i, thus removing any velocity that the atmosphere naturally might have had. As shown in [71], resetting as little as possible can be an advantage in simulations with orbiting stars when we wish to accurately track lower density mass ejecta.

4.3.3. Star surfaces. Since the matter fields are not smooth across neutron star surfaces, we observe Gibbs phenomena (i.e. high frequency noise) in the nodes that contain a piece of the star surface. Here, we use a simple solution to this problem and apply the filter of equation (40) to damp this noise after each full time step. This filter changes the fields at every point by a

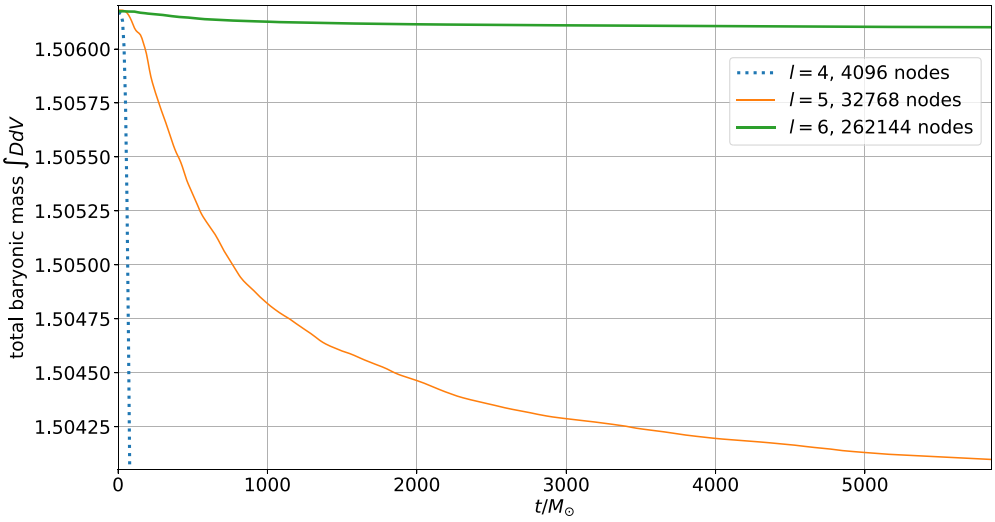

Figure 9. The plot shows the total baryonic mass in our computational domain versus time for $l=5$ and $l=6$ levels of h-refinement. As one can see, mass conservation is quite good at the highest resolution. Here $dV = \sqrt{\gamma}d^3x$.

typically small amount. Nevertheless this can still cause trouble in low density regions by, e.g. making D or τ slightly negative or by violating $S < D + \tau$. Thus after filtering we reapply the positivity limiters discussed above. For the neutron star tests described below, we use the filter parameters $\alpha_{\rm f} = 36$ and $s = 32$.

4.3.4. Tests with single neutron stars. To test our general relativistic hydrodynamics implementation, we have performed simulations of a single neutron star for a fixed spacetime metric. As already mentioned, we use units where $G = c = M_\odot = 1$. To convert to SI units, a dimensionless length has to be multiplied by $L_0 = 1476.6250$ m, a time by $T_0 = 4.9254909 \times 10^{-6}$ s, a mass by $M_\odot = 1.9884099 \times 10^{30}$ kg, and a mass density by 6.1758285×10^{20} kg m^{13}.

The first test starts with initial data for a static Tolman–Oppenheimer–Volkoff (TOV) star with a central density of $\rho_0 = 0.00128$. To setup the initial data, we use a polytropic EoS, where pressure and specific internal energy are given by $P = \kappa\rho_0^{1+1/n}$ and $\epsilon = n\kappa\rho_0^{1/n}$, with $\kappa = 100$ and $n = 1$. This results in star with a baryonic mass (i.e. rest-mass) of $m_0 = 1.5061762M_\odot$ and an ADM mass of $m = 1.4001597M_\odot$. For the subsequent evolution we adopt a gamma-law EoS of the form $P = \rho_0\epsilon/n$ with $n = 1$.

We evolve this star on a single cubic patch with side length 32. The patch is centered on the star and covered by Cartesian coordinates. To better resolve the star surface, where the matter fields are not smooth, we use either 4, 5, or 6 levels of h-refinement, so that we end up with 4096, 32 768, or 262 144 leaf nodes. In each node, we use $5 \times 5 \times 5$ grid points. The star is then evolved for more than $5000M_\odot$, with a time step of 0.1, 0.05, or 0.025. As we can see in figure 9, baryonic mass conservation improves with increasing resolution. The reason why mass is not exactly conserved is twofold. First, as already mentioned, our positivity limiters are conservative only if the node average is above the limits we impose. Yet this is not always the case, so that the limiter can cause conservation violations. Second, the outer boundary is relatively close, so that mass can escape from our numerical domain. Nevertheless baryonic mass conservation improves with increasing resolution.

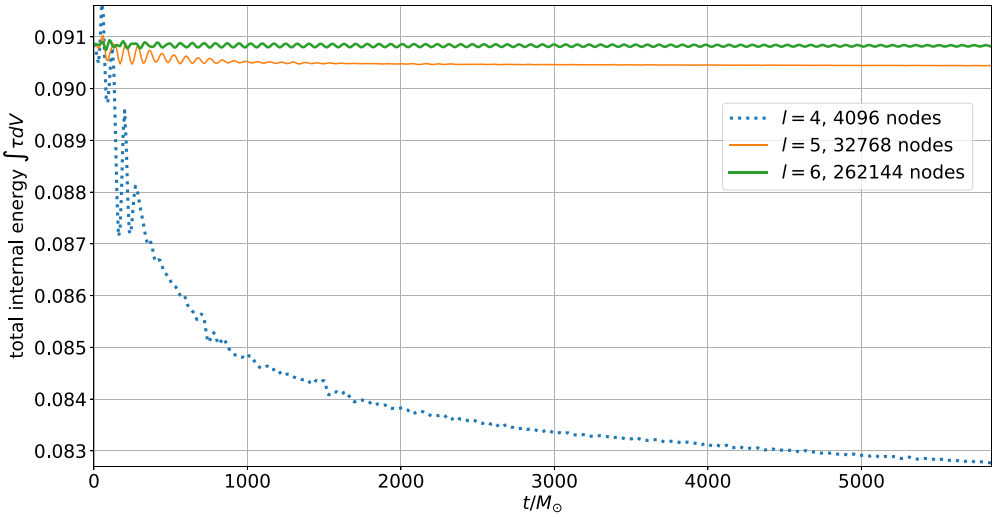

Figure 10. The plot shows the total internal energy versus time for $l = 4$, $l = 5$, and $l = 6$ levels of h-refinement. Since τ is not strictly conserved in general relativity, we can see oscillations in it. For $l = 5$ and $l = 6$, the oscillation period of $75M_\odot$ is easily visible and agrees with the fundamental oscillation frequency of the star.

In figure 10 we show the integral over the internal energy density τ. This quantity is conserved in special relativity, but has a source term in general relativity. Thus, it is not expected to be strictly conserved during an evolution. In fact, for the two higher resolutions, one can clearly see oscillations in it that are slowly damped out. The period of these oscillations is about $75M_\odot$ which corresponds to a frequency of 2.7 kHz, which is in agreement with the known fundamental oscillation frequency of this star [72, 73]. These oscillations are also visible for the highest resolution in figure 11, which shows the maximum of D versus time. For the lower two resolutions, however, these oscillations are swamped by noise that is caused by Gibbs phenomena at the star surface. The reason why oscillations due to Gibbs phenomena are more prominent in figure 11 than in figures 9 and 10 is that the maximum of D is determined at a single point, while the integrals over D and τ represent an average over the entire domain that is less sensitive to Gibbs phenomena. It is clear from figure 11 that if we are interested in values at particular points, we need high resolution to get results where the expected physical oscillations dominate over the oscillations due to Gibbs phenomena. Nevertheless, our approach, that only uses positivity limiters together with filters, is capable of stabilizing the evolution of the star for all three resolutions.

The oscillations described so far originate purely from numerical errors. To test the robustness of our approach, we have also evolved perturbed stars. In this case, we use the same analytic TOV solution as above, but we add a perturbation of the form:

$$\delta P = \lambda \cdot (P + \rho_0 + \rho_0 \epsilon) \sin(\pi r / r_{\text{surf}}) Y_2^0(\theta, \varphi), \tag{83}$$

to the pressure. Here (r, θ, φ) are the standard spherical coordinates, $r_{\text{surf}} = 8.1251439$ is the radius of the unperturbed star in isotropic coordinates, and $Y_2^0(\theta, \varphi)$ is the $l = 2, m = 0$ spherical harmonic. We use the above polytropic EoS to then recalculate the initial ρ_0 and ϵ from the perturbed pressure $P + \delta P$. All metric variables are kept at their unperturbed TOV values. For the simulations in this paper, we have used a fairly strong perturbation with $\lambda = 0.05$.

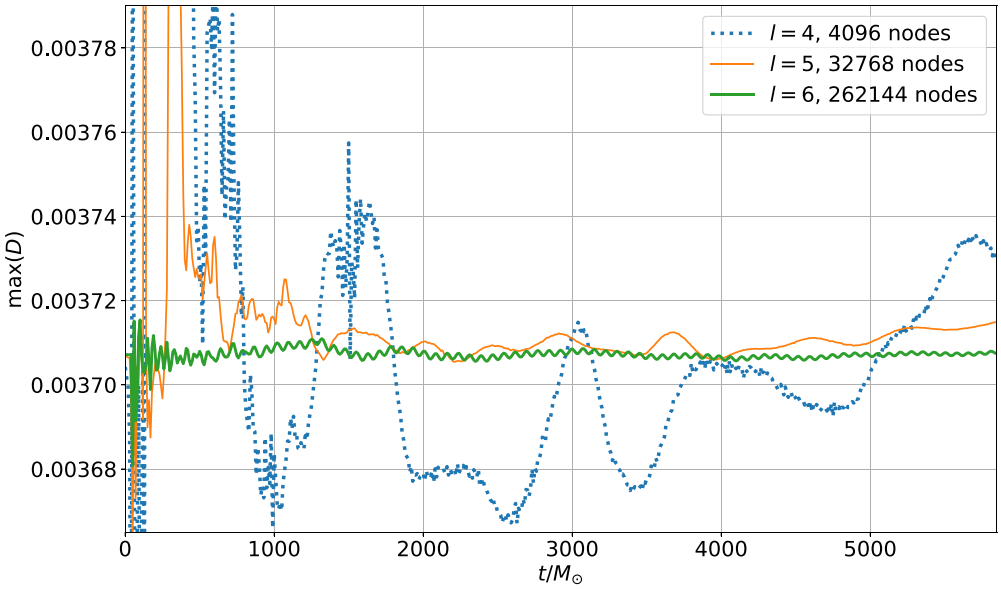

Figure 11. The plot shows the maximum of D versus time for $l=4$, $l=5$, and $l=6$ levels of h-refinement. Gibbs phenomena emanating from the star surface lead to noisy oscillations. Only for the highest resolution, these oscillations clearly exhibit the fundamental oscillation frequency of this star.

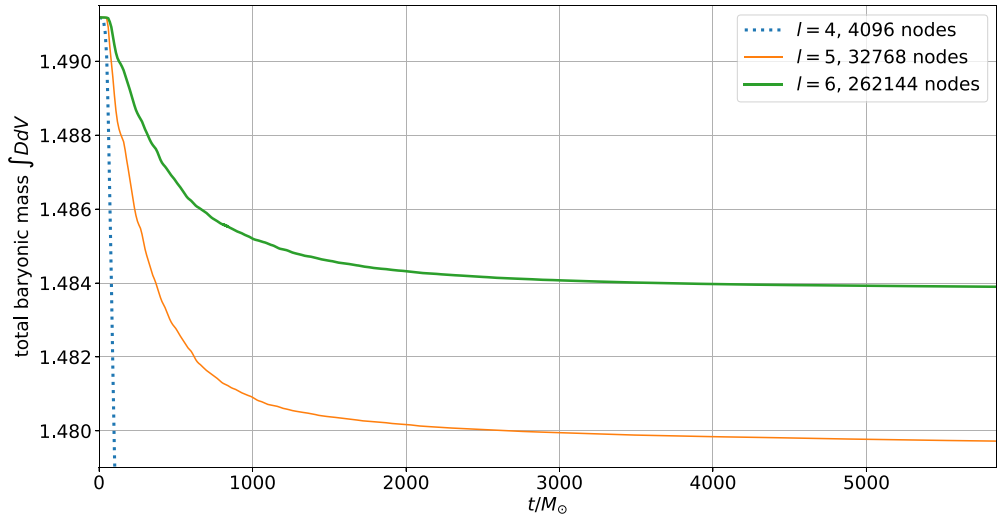

Figure 12. The total baryonic mass for the perturbed star versus time for $l=5$ and $l=6$ levels of h-refinement. Strong star pulsations cause material to leave through the outer boundary and are thus responsible for the initial drop in the mass.

In figures 12–14, we show the total mass, the total internal energy, and the maximum of the density D for the perturbed star. Since the perturbation is relatively strong, the star oscillations are now much bigger, so that the oscillations in the total internal energy are now much larger.

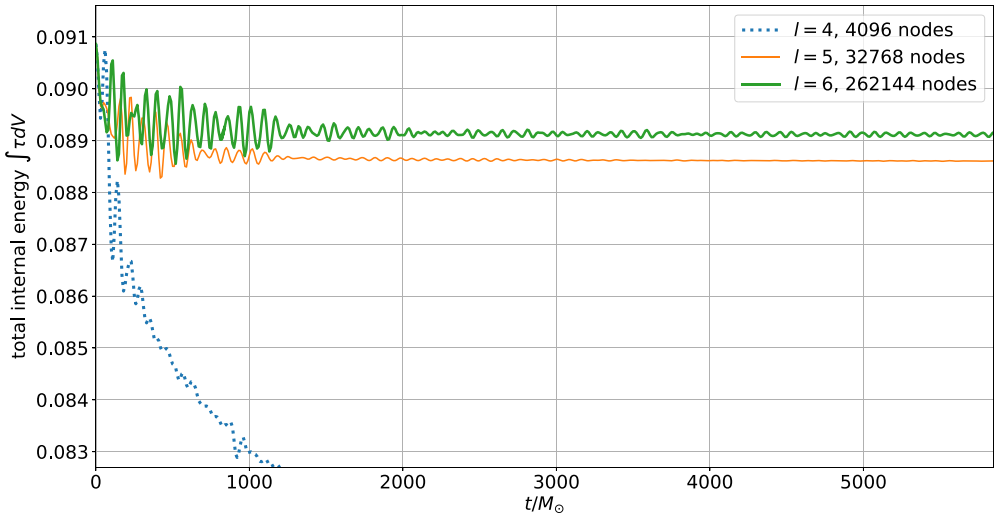

Figure 13. The total internal energy for the perturbed star. Due to the strength of the perturbation the oscillation amplitude is much larger than for the unperturbed star in figure 10.

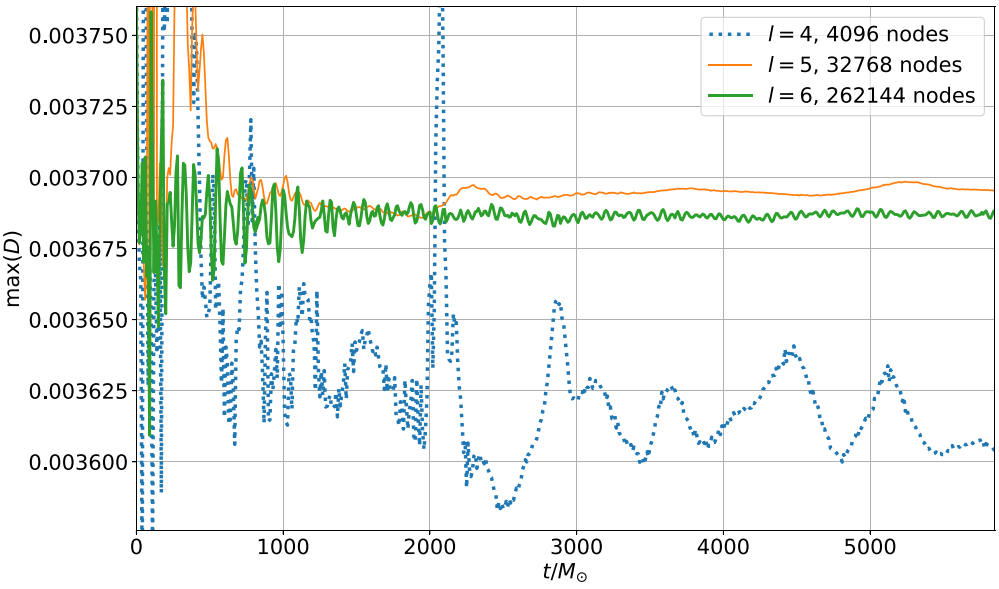

Figure 14. The maximum D for the perturbed star is qualitatively similar to the unperturbed case, but the oscillation amplitudes are larger.

This is clearly visible for the two higher resolutions ($l = 5$, $l = 6$) in figure 13. In fact, the star pulsations are now so strong that much more material leaves through the outer boundary. This leads to the initial drop in the mass seen in figure 12. The maximum of D also oscillates stronger, but as for the unperturbed case, the expected oscillation frequency is only readily discernible at the highest resolution ($l = 6$) in figure 14.

When comparing the oscillations in the internal energies in figures 10 and 13, for both perturbed and unperturbed stars, we can see that in both cases the star oscillations are more strongly damped for low resolutions.

The main takeaway is thus that our approach is robust since it still works for strongly perturbed stars. We have also seen that, at the lowest resolution, oscillations due to Gibbs phenomena can easily dominate the expected physical oscillations. Such Gibbs phenomena will become only worse once we have to deal with true shocks, e.g. if two stars collide. We thus expect to need additional limiters once we have to deal with shocks.

We also wish to comment on the work in [40] where single neutron stars are simulated using a DG method together with various limiters (such as e.g. WENO, HWENO, or Krivodonova), and also using a hybrid scheme, that switches to finite differences (FD) in non-smooth regions (e.g. near the star surfaces). The main result of [40] is that their hybrid DG-FD scheme works better than any of the many limiters tested, and that in fact the evolution of a single neutron star failed with many of the limiters tested. Since our new positivity limiter is not expected to be sufficient to deal with true shocks, using such a hybrid DG-FD scheme may very well be the best way forward. However, it is possible we are at least able to obtain stable evolutions with our limiter if we combine it with an additional limiter that deals with shocks. In the next section we will therefore test several limiters that are designed to treat shocks.

4.3.5. Limiters for the treatment of shocks. Since general relativistic hydrodynamics allows for the development of shocks in the fluid, we need to be prepared to deal with them. A general way to handle spurious oscillations due to Gibbs phenomena, occurring in these situations, is to apply limiters to the hydrodynamic fields. We try out two types of limiters in this paper. The first is the so-called total variation bounded minmod or minmodB slope limiter, which has been developed, demonstrated and utilized in multiple articles, such as [43, 46, 74, 75], including methods compatible with the DG evolution scheme. We follow closely the formalism in [75] and apply the minmodB limiter to the conserved variables. The other one is the limiter proposed by Moe *et al* [76], dubbed henceforth as the MRS limiter. In this work, we apply the MRS limiter to either the conserved variables (MRS(cons.)) or the primitive variables (MRS(prim.)). The case of MRS(cons.) is straightforward, as we can directly apply the limiter to the variables we actually evolve. However, in case of MRS(prim.), a problem arises since we first have to recover the primitive variables from the evolved conserved variables, which can fail if, e.g. the momentum density is too high. To address this, we perform a procedure of prelimiting similar to what is described in [55], to a copy of the conserved variables. Through this prelimiting, we ensure that the strong condition $S_i S^i < \tau(\tau + 2D)$ holds for this copy of conserved variables. Once we have calculated the primitive variables from the prelimited copy of conserved variables, we compute the rescaling factor θ_i for the MRS limiter, as described in [76], using the primitive variables ρ_0, Wv^i, and P. However, after we have obtained this θ_i, we apply it to rescale the original non-prelimited conserved variables that we are evolving, which is then our actual limiting procedure.

To test how well `Nmesh` handles shocks, we implement test cases in both 1D and 2D, where we have an initial discontinuity in density and pressure, as in a Riemann problem. We then evolve this initial discontinuity using the full general relativistic hydrodynamic evolution system of equations on a fixed Minkowski metric. The mesh is composed of adjacent Cartesian domains. For these tests, the time step was set to be $\Delta t = \Delta x_{\min}/4$.

1D Test: We use the special relativistic blast wave test from [77], and also use the analytic solution code from the same article to compare with the numerical result from `Nmesh`. The initial data in this case is such that we have two different values on the left and right halves of

Table 1. Initial data for 1D special relativistic blast wave for primitive variables (ρ_0, P, v_x).

Left, $x \leqslant 0.5$	(10.00, 13.33, 0.00)
Right, $x > 0.5$	(1.00, 0.00, 0.00)

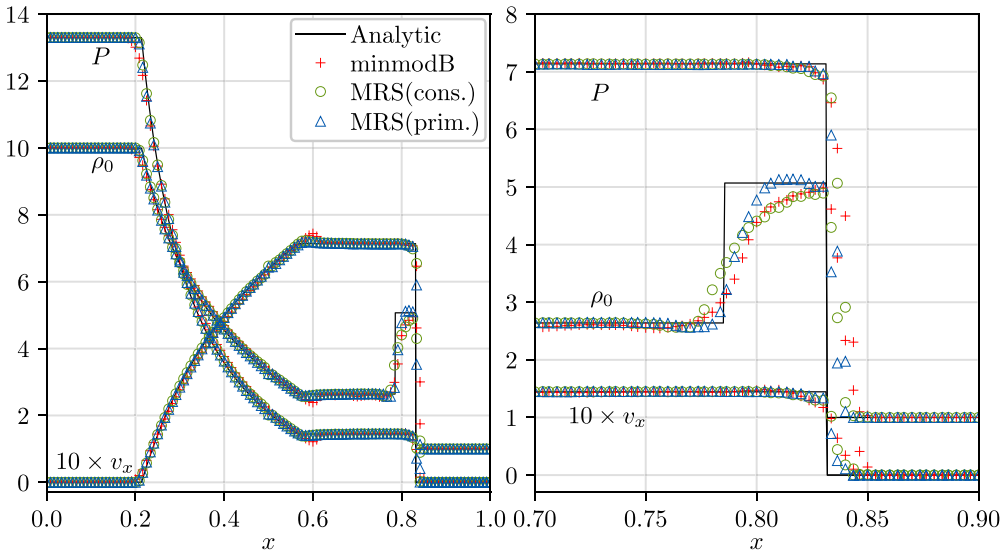

Figure 15. The plots show blast wave profiles for pressure P, rest-mass density ρ_0 and speed v_x (times 10) at $t = 0.4$ after evolving the initial shocks in P and ρ_0. There are 200 domains, with 4 points per domain. We show results for minmodB, MRS(cons.), and MRS(prim.) limiters. The left plot shows the whole domain, while the right one focuses on the contact discontinuity and shock fronts. The legend on the left plot also holds for the right plot.

the mesh, for the primitive variables ρ_0 and P. The values of the primitive variables ρ_0, P, and v_x are as stated in table 1.

In figure 15 we show the profiles for the primitive variables over the entire mesh (left plot), as well as the contact discontinuity and the shock front (right plot), after evolving the initial data to time $t = 0.4$. The plots contain the numerical results obtained from `Nmesh` as well as the analytic solution from [77]. For the numerical results, we have used 200 adjacent Cartesian domains along the x-axis, with 4 points in each domain. For minmodB, referring to the formalism in [75], we set $\beta = 0.6$ and $\alpha_{\rm lim} := \tilde{M} = 5$. For MRS(cons.) and MRS(prim.), we set the α from [76] to $\alpha = \alpha_{\rm lim} L^{3/2}$ with $\alpha_{\rm lim} = 25$, where L is the size of the node. While the exact meaning of $\alpha_{\rm lim}$ is different in minmodB and MRS, in both cases lower $\alpha_{\rm lim}$ makes the limiter more aggressive. From the plots, it appears that the result with MRS(prim.) adheres closest to the analytic result, whereas the one with minmodB seems to deviate the most from it. This is true for the plot on the left, that shows the behavior across the entire mesh, but is clearer from the plot on the right, that focuses on the problematic region of the contact discontinuity and the shock front.

2D Test: The 2D test we perform is an extension of the 1D test Riemann problem, that can be found in [78]. The initial data in the primitive variables (ρ_0, P, v_x, v_y) for the 2D test over the mesh is as stated in table 2.

Table 2. Initial data for 2D special relativistic blast wave for primitive variables (ρ_0, P, v_x, v_y).

	$x<0$	$x\geqslant 0$
$y\geqslant 0$	(0.1, 1, 0.7, 0)	(0.03515, 0.163, 0, 0)
$y<0$	(0.5, 1, 0, 0)	(0.1, 1, 0, 0.7)

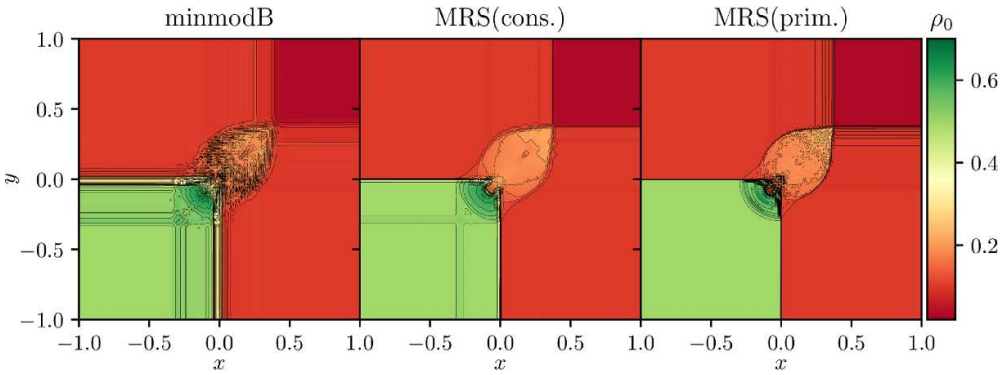

Figure 16. Plots showing rest-mass density profile of 2D blast wave at $t=0.4$ after evolving the initial discontinuity with minmodB, MRS(cons.) and MRS(prim.) limiters in colormap and 30 contour lines spaced evenly between 0.01 and 0.695.

The numerical results obtained from the three different limiter choices from `Nmesh` are shown in figure 16 at time $t=0.4$, after evolving the initial data. We have only plotted the results for ρ_0, as it is arguably the most problematic case. We compare our results with those of Zhao and Tang in [78] and Bugner in [57], while noting that Zhao and Tang have used a finite element DG method with WENO and a special relativistic hydrodynamic system of evolution equations and Bugner used a DG method with WENO and fully general relativistic hydrodynamic system of equations, while `Nmesh` uses DG with the minmodB and MRS limiters and the fully general relativistic hydrodynamic system of equations. In our runs here, the mesh is composed of 100×100 Cartesian domains, with 4 points, i.e., we have 4×4 points in each domain along each direction. Again, for minmodB, we use $\beta=0.6$ and $\alpha_{\rm lim}=5$. For MRS(cons.) and MRS(prim.), we set the parameter $\alpha_{\rm lim}=25$. Also, we use our new positivity limiter to control S_i according to equation (82).

Once again, we see that both MRS(cons.) and MRS(prim.) fare better than minmodB. However, in this case, we cannot draw a clear conclusion as to which one out of MRS(cons.) and MRS(prim.) yields a better result. MRS(cons.) seems to provide overall better results than MRS(prim.), except for the left bottom region, where MRS(prim.) seems to be better at handling the high density region and the so-called 'mushroom cloud' structure around position (0,0). However, overall, the MRS limiter cases are in reasonably good agreement with the results found in [57, 78].

4.3.6. Single TOV star with MRS limiter. Since limiting the conserved variables with the MRS scheme was successful in our shock tube tests, we wanted to know how this limiter would influence the neutron star simulations presented above. As a test we have repeated the runs for the unperturbed TOV star, but this time with the MRS(cons.) limiter turned on with

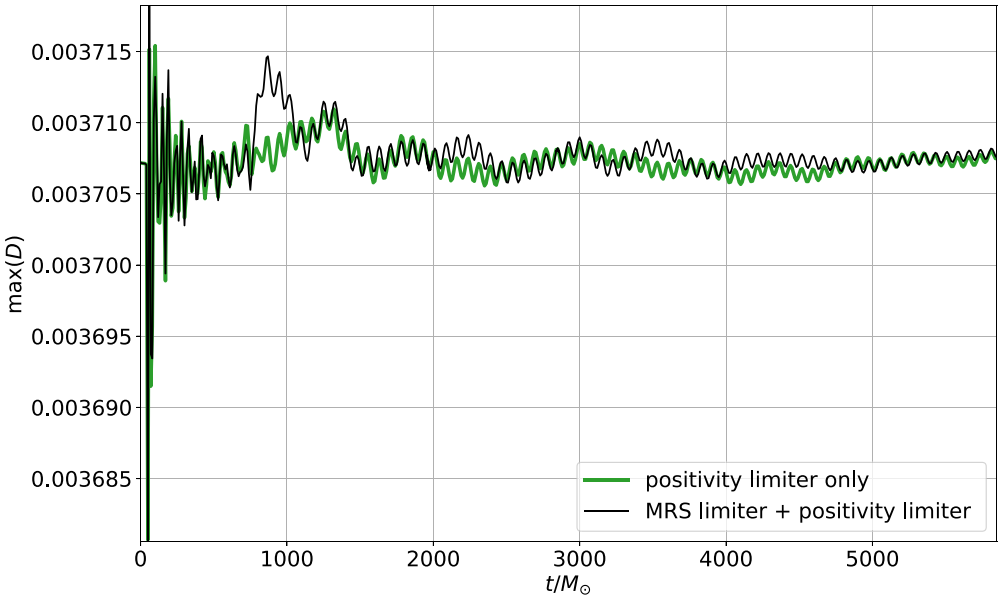

Figure 17. The plot shows the maximum of D versus time for six levels of h-refinement with (thin black line) and without (thick green line) the MRS limiter turned on.

$\alpha_{\text{lim}} = 25$. Note that the positivity limiters are still needed to deal with low density regions. Thus we use both the MRS limiter and the positivity limiters described above, with the MRS limiter running immediately before the positivity limiters. We find that the total baryonic mass and internal energy are almost the same with and without the MRS limiter in the sense that the corresponding plots look almost the same as in figures 9 and 10, even when the MRS limiter is turned on. The biggest difference occurs when we plot the maximum of D. In figure 17 we show the oscillations in the maximum of D for the high resolution with and without the MRS limiter. In both cases we see the expected star oscillations, but there are also longer term drifts in the maximum of D. With the MRS limiter this drift is a bit different and arguably slightly more pronounced. Nevertheless, the MRS limiter does not lead to big changes, which is not too surprising, since the fluid in a single star does not contain any shock fronts. Yet, this run demonstrates that the MRS limiter does not cause any stability problems when added to our previous method.

5. Discussion

In this article, we have presented all the numerical and computational methods used in our new `Nmesh` program to evolve systems of hyperbolic equations. The principal scheme we use for spatial discretization is a DG method. This is then coupled with a Runge–Kutta time integrator to be able to evolve in time. The DG method can easily deal with many domains. We use this to introduce many patches, which can be adaptively refined by splitting them into eight child domains (see e.g. figures 1 and 2), as often as desired. This AMR scheme is then parallelized by distributing the resulting many domains among all available compute cores. For the neutron star test cases shown in figure 3 this approach achieves good strong scaling. As explained in section 3, an advantage of DG methods is that they result in less communication overhead than

traditional finite difference or finite volume methods. In [40], a similar DG method is used, albeit with one crucial difference. To derive our DG method we integrate using coordinate volume elements, and thus do not include the physical metric. This leads to a simplification of the method where one does not have to worry about possible discontinuities in the physical metric or the normal vector across domain boundaries.

We have also carried out simulations of scalar fields and black holes to test the convergence of our new program. Since in this case all evolved fields are smooth, we expect exponential convergence when the number of grid points is increased. Our simulation results conform to this expectation. We find that both, the upwind and LLF fluxes perform equally well, in all cases tested.

A much more complicated case is the evolution of neutron stars, since in this case, the matter fields are not smooth across the star surface. An additional problem arises from the fact that at each Runge–Kutta substep we have to calculate the primitive variables from the evolved conserved variables. The latter can easily fail in low density regions (such as the vacuum outside the star), where numerical errors can cause the conserved variables to become unphysical in the sense that the mass or internal energy densities can become negative, or the momentum density can be become too high. To address this problem, we have developed a new positivity limiter that attempts to reset these variables by scaling toward their node averages in case of trouble. If we use our positivity limiter together with the exponential filters described before, we can stably evolve single neutron stars. These stable evolutions are possible without any extra ingredients, such as an artificial low density atmosphere or additional limiters (like the minmodB or MRS limiters described above), that have been used in other works. We believe that our positivity limiter is an important step, because the more general limiters like minmodB or MRS are really designed to deal with shocks and thus do not help in low density regions near star surfaces. Nevertheless, something like these general limiters is still needed to deal with shocks. As we have shown above, the general limiters can be used in combination with the positivity limiter.

We thus have all necessary ingredients to perform simulations of binary neutron stars and black holes, which is what we plan to do in the future with the `Nmesh` program.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

It is a pleasure to thank Bernd Brügmann for helpful discussions. This work was supported by NSF Grants PHY-1707227 and NSF PHY-2011729. We also acknowledge usage of computer time on the HPC cluster KoKo at Florida Atlantic University, on the Dutch National supercomputer Cartesius, as well as on Bridges-2 and Expanse under XSEDE allocation TG-PHY220018.

Appendix A. About the flat metric in Gauß's theorem

In equation (7) we have used Gauß's theorem in the form:

$$\int \partial_i f^i d^3x = \oint f^i n_i d\bar{A}, \tag{A.1}$$

where n_i was normalized with respect to the flat metric, expressed as δ_{ij} in the global Cartesian-like coordinates x^i that cover all our domains. For example, on $x^{\bar{3}} = 1$ we have:

$$n_i = \frac{1}{\sqrt{\bar{\gamma}^{\bar{3}\bar{3}}}} \frac{\partial x^{\bar{3}}}{\partial x^i}, \tag{A.2}$$

and

$$\bar{\gamma}^{\bar{3}\bar{3}} = \frac{\partial x^{\bar{3}}}{\partial x^i} \frac{\partial x^{\bar{3}}}{\partial x^j} \delta^{ij}. \tag{A.3}$$

Thus we find:

$$n_i d\bar{A} = \frac{1}{\sqrt{\bar{\gamma}^{\bar{3}\bar{3}}}} \frac{\partial x^{\bar{3}}}{\partial x^i} d\bar{A} = \frac{\partial x^{\bar{3}}}{\partial x^i} \frac{J}{\sqrt{{}^{(2)}\bar{\gamma}}} d\bar{A} = \frac{\partial x^{\bar{3}}}{\partial x^i} J dx^{\bar{1}} dx^{\bar{2}}, \tag{A.4}$$

where in the last two steps we have used equations (27) and (14). We see that the flat metric pieces all cancel, and thus do not influence the surface integral.

The analog of equation (27) for the physical metric (denoted by γ_{ij} without overbar) is:

$$J = \frac{\sqrt{{}^{(2)}\gamma}}{\sqrt{\gamma}\sqrt{\gamma^{\bar{i}\bar{i}}}}. \tag{A.5}$$

If we insert the latter into the right hand side of equation (A.4) we find:

$$n_i d\bar{A} = \frac{1}{\sqrt{\gamma}} \frac{1}{\sqrt{\gamma^{\bar{3}\bar{3}}}} \frac{\partial x^{\bar{3}}}{\partial x^i} \sqrt{{}^{(2)}\gamma} dx^{\bar{1}} dx^{\bar{2}} = \frac{1}{\sqrt{\gamma}} N_i dA, \tag{A.6}$$

where N_i is normalized with the physical metric and dA is the physical surface element. Let us now define:

$$F^i := \frac{f^i}{\sqrt{\gamma}}. \tag{A.7}$$

Then the right hand side of equation (A.1) can be written as:

$$\oint f^i n_i d\bar{A} = \oint F^i N_i dA, \tag{A.8}$$

while the left hand side is:

$$\int \partial_i f^i d^3x = \int \frac{1}{\sqrt{\gamma}} \partial_i (\sqrt{\gamma} F^i) \sqrt{\gamma} d^3x = \int D_i F^i \sqrt{\gamma} d^3x, \tag{A.9}$$

where D_i is the covariant derivative operator. Together this yields:

$$\int D_i F^i \sqrt{\gamma} d^3x = \oint F^i N_i dA, \tag{A.10}$$

which is the well-known coordinate independent form of Gauß's theorem.

This shows that we can use other metrics besides the physical one in Gauß's theorem, because all pieces of any metric cancel. Yet, whatever metric we choose to use, must also be used to normalize our normal vector.

Appendix B. On the influence of different normalizations

We now discuss the differences between using the physical metric γ_{ij} (as in [40, 48]) and the flat metric δ_{ij} to normalize the vectors n_i normal to a domain boundary.

To obtain a simple example we start with a 2-dimensional spacetime metric $ds^2 = -\alpha^2 dt^2 + \gamma_{xx} dx^2$. If we retrace the steps that lead from equations (1) to (5), we find:

$$\partial_t u + \partial_x f = s. \tag{B.1}$$

The main step in the DG method consists of integrating the $\partial_x f$ term, which in one spatial dimension becomes:

$$\begin{aligned}\int_a^b dx \psi \, \partial_x f &= [\psi f]_a^b - \int_a^b dx f \partial_x \psi \rightarrow [\psi f^*]_a^b - \int_a^b dx f \, \partial_x \psi \\ &= [\psi (f^* - f)]_a^b + \int_a^b dx \psi \, \partial_x f,\end{aligned} \tag{B.2}$$

where again we have introduced a numerical flux f^*. The term $[\psi (f^* - f)]_a^b$ corresponds to the surface integral in equation (8), and can be written as:

$$[\psi (f^* - f)]_a^b = \psi (f^* - f) n|_b + \psi (f^* - f) n|_a, \tag{B.3}$$

where the outward pointing normals are $n|_b = 1$ and $n|_a = -1$. So far the physical metric γ_{xx} has not appeared. Following Teukolsky [48] we now define a normal vector $N := n/\sqrt{\gamma^{xx}}$ that is normalized with respect to the physical metric. We then obtain:

$$\psi (f^* - f) n = \psi (f^* - f) N \sqrt{\gamma^{xx}}. \tag{B.4}$$

This means we can replace the n that was normalized with respect to the flat metric with an N that is normalized with respect to the physical metric γ_{xx}, provided we include other metric factors. Notice that the factor $\sqrt{\gamma^{xx}}$ in equation (B.4) is equivalent to the γ^{ij} under the root in equation (35) of [40], and that in the case discussed here $\xi \propto x$, so that the J and $\partial \hat{\xi^i}/\partial x^j$ terms in equation (35) of [40] drop out. The fact that the N and $\sqrt{\gamma^{xx}}$ terms in equation (B.4) cancel each other, agrees with the discussion in appendix A of [48] that calls the appearance of the physical metric illusory, and also with our appendix A.

The situation becomes less trivial when one considers how the numerical flux f^* is actually computed, which is related to the point about metric discontinuities being tricky, that is raised in [40]. As an example, let us consider the LLF flux of equation (33). It depends on the field value u_{in} in the current domain and the u_{adj} from the adjacent domain. For n_i equation (33) makes no such distinction because n_i, normalized with the flat metric, is the same on both sides. The analog to equation (33) found in equation (36) of [40] is:

$$(f^i N_i)^* = \frac{1}{2}\left[f^i(u_{\text{in}}) N_i^{\text{in}} + f^i(u_{\text{adj}}) N_i^{\text{adj}} + |\Lambda|_{\max} (u_{\text{in}} - u_{\text{adj}}) \right], \tag{B.5}$$

where N_i^{in} and N_i^{adj} are the normals in the different domains that differ if the physical metric is discontinuous across the domain boundary. Also note that $|\Lambda|_{\max}$ denotes the absolute maximum eigenvalue magnitude, when we consider eigenvalues from both sides. I.e. the numerical flux in [40] differs from our approach if the physical metric is discontinuous across the domain boundary. Note, however, that the physical metric of the true continuum solution will always be continuous, so that such discontinuities will converge to zero with increasing resolution.

Finally, we will compute the numerical flux with both equations (33) and (B.5) for a simple example. Consider the case where we have a single field u with $f = u$ and $s = 0$, so that equation (B.1) becomes:

$$\partial_t u + \partial_x u = 0, \tag{B.6}$$

which is a simple advection equation for u. We wish to solve this equation in a 1-dimensional domain $x \in [a, b]$. If we use n as normal, the eigenvalue $\lambda = 1$ on the right boundary (at $x = b$). At $x = b$ equation (33) then yields:

$$(fn)^* = u_{\text{in}} = fn. \tag{B.7}$$

If we use N as normal, the eigenvalue $\Lambda = N$ on the right boundary (at $x = b$).

Thus equation (B.5) results in:

$$\begin{aligned}(fN)^* &= \frac{1}{2}\left[u_{\text{in}}N^{\text{in}} + u_{\text{adj}}N^{\text{adj}} + |\Lambda|_{\max}\left(u_{\text{in}} - u_{\text{adj}}\right)\right] \\ &= \frac{1}{2}\left[(N^{\text{in}} + |\Lambda|_{\max})u_{\text{in}} + (N^{\text{adj}} - |\Lambda|_{\max})u_{\text{adj}}\right],\end{aligned} \tag{B.8}$$

where $|\Lambda|_{\max} = \max(|N^{\text{in}}|, |N^{\text{adj}}|)$. Unless $N^{\text{in}} = N^{\text{adj}}$, $(fN)^*$ is not equal to fN, and thus the result really differs from $(fn)^* = fn$. An analogous difference also occurs on the left boundary at $x = a$.

The question now arises which approach we should use. The analytic solution of the advection equation (B.6) is $u = h(x - t)$, where $h(x)$ is an arbitrary function. I.e. we obtain a profile that moves to the right over time. Thus no boundary condition is needed on the right, because nothing is entering the domain from there. The corresponding numerical flux should thus be computed solely from quantities inside our domain, and hence be given by the upwind flux $f^* = f = u_{\text{in}}$. The latter is exactly what we have obtained from the LLF flux of equation (33), when using the flat metric to normalize our normals. This is expected, as the LLF flux for a single field obeying equation (B.6) is known to be equivalent to the upwind flux. Also notice that the boundary term at $x = b$ in equation (B.3) entirely vanishes for this upwind flux, which is equivalent to not imposing any boundary condition on the right. Yet, we do not obtain these same results if we follow [40] and normalize with the physical metric (unless the physical metric is continuous across the boundary). Nevertheless, we believe that both normalization approaches can work, because the physical metric of the true continuum solution will always be continuous. We thus expect both approaches to converge to the same physical solution. However, we prefer our approach to the one in [40], because it is simpler, and also because it reproduces the correct upwind result for a single advection equation.

We should also note, that in the first paper about the SpECTRE code [39] it is claimed (in the footnote on page 7) that the 'unit normal' is the same on the two sides of the boundary. From the context of this footnote it seems as if the authors mean N_i (normalized with respect to the physical metric), when they write 'unit normal'. This, however, cannot be true because it is precisely n_i (normalized with respect to flat metric), that is the same on both sides of the boundary. This is because n_i denotes the normal expressed in the global Cartesian-like x^i coordinates, which cover all domains (see remark after equation (14)). Thus by definition n_i cannot have any discontinuities, while N_i (obtained by renormalizing n_i with the physical metric) can be discontinuous, if the physical metric is.

Another way of seeing that N_i can be discontinuous, is by recalling that it is normalized with the physical metric and thus:

$$N_i^{\text{in}} N_j^{\text{in}} \gamma_{\text{in}}^{ij} = 1 = N_i^{\text{adj}} N_j^{\text{adj}} \gamma_{\text{adj}}^{ij}. \tag{B.9}$$

Therefore, if γ^{ij}_{in} and γ^{ij}_{adj} differ, N_i^{in} and N_i^{adj} can differ as well. Also notice, that equation (3.16) of [39] has a term with eigenvalues, which is identical to the one in equation (B.5), and also contains an N_i that is different on both sides of the boundary. Hence it seems the authors of [39] agree with us, that N_i can be discontinuous.

In section 4.2 we have tested the evolution of a single black hole using the DG method, where domain normals are normalized with respect to the flat metric. As we have seen, the discontinuities in the physical metric are not a problem for our approach, even though the numerical solution goes through an initial rapid evolution phase.

ORCID iD

Wolfgang Tichy https://orcid.org/0000-0002-8707-754X

References

[1] Abbott B P *et al* 2017 GW170817: observation of gravitational waves from a binary neutron star inspiral *Phys. Rev. Lett.* **119** 161101
[2] Abbott B P *et al* 2017 Gravitational waves and Gamma-rays from a binary neutron star merger: GW170817 and GRB 170817A *Astrophys. J. Lett.* **848** L13
[3] Coulter D A *et al* 2017 Swope supernova survey 2017a (SSS17a), the optical counterpart to a gravitational wave source *Science* **358** 1556–8
[4] Abbott B P *et al* 2019 GWTC-1: a gravitational-wave transient catalog of compact binary mergers observed by LIGO and Virgo during the first and second observing runs *Phys. Rev.* **X9** 031040
[5] Abbott R *et al* 2021 GWTC-2: compact binary coalescences observed by LIGO and Virgo during the first half of the third observing run *Phys. Rev.* X **11** 021053
[6] Abbott R *et al* 2021 Observation of gravitational waves from two neutron starblack hole coalescences *Astrophys. J. Lett.* **915** L5
[7] Abbott B P *et al* 2017 Multi-messenger observations of a binary neutron star merger *Astrophys. J.* **848** L12
[8] Abbott B P *et al* 2017 A gravitational-wave standard siren measurement of the Hubble constant *Nature* **551** 85–88
[9] Radice D, Perego A, Zappa F and Bernuzzi S 2018 GW170817: joint constraint on the neutron star equation of state from multimessenger observations *Astrophys. J.* **852** L29
[10] Most E R, Weih L R, Rezzolla L and Jürgen S-B 2018 New constraints on radii and tidal deformabilities of neutron stars from GW170817 *Phys. Rev. Lett.* **120** 261103
[11] Metzger B D 2020 Kilonovae *Living Rev. Rel.* **23** 1
[12] Tim Dietrich M W Coughlin P T H Pang M B, Heinzel J, Issa L, Tews I and Antier S 2020 Multimessenger constraints on the neutron-star equation of state and the Hubble constant *Science* **370** 1450–3
[13] Blanchet L 2014 Gravitational radiation from post-Newtonian sources and inspiralling compact binaries *Living Rev. Rel.* **17** 2
[14] Bernd Bruegmann J A Gonzalez M H, Husa S, Sperhake U and Tichy W 2008 Calibration of moving puncture simulations *Phys. Rev.* D **77** 024027
[15] Thierfelder M, Bernuzzi S and Bernd B 2011 Numerical relativity simulations of binary neutron stars *Phys. Rev.* **D84** 044012
[16] Dietrich T, Bernuzzi S, Ujevic M and Bernd B 2015 Numerical relativity simulations of neutron star merger remnants using conservative mesh refinement *Phys. Rev.* **D91** 124041
[17] Löffler F *et al* 2012 The einstein toolkit: a community computational infrastructure for relativistic astrophysics *Class. Quantum Grav.* **29** 115001
[18] Haas R *et al* 2020 The Einstein toolkit (available at: http://einsteintoolkit.org)
[19] Ruchlin I, Etienne Z B and Baumgarte T W 2018 SENR/NRPy+: numerical relativity in singular curvilinear coordinate systems *Phys. Rev.* D **97** 064036
[20] Kiuchi K, Kyohei K, Kyutoku K, Sekiguchi Y and Shibata M 2020 Sub-radian-accuracy gravitational waves from coalescing binary neutron stars II: systematic study on the equation of state, binary mass and mass ratio *Phys. Rev.* D **101** 084006

[21] SpEC - Spectral Einstein code (available at: www.black-holes.org/SpEC.html)
[22] Boyle M *et al* 2019 The SXS Collaboration catalog of binary black hole simulations *Class. Quantum Grav.* **36** 195006
[23] Reitze D *et al* 2019 Cosmic explorer: The U.S. contribution to gravitational-wave astronomy beyond LIGO *Bull. Am. Astron. Soc.* **51** 035 (arXiv:1907.04833)
[24] Kawamura S *et al* 2021 Current status of space gravitational wave antenna DECIGO and B-DECIGO *PTEP* **2021** 05A105
[25] Punturo M *et al* 2010 The Einstein telescope: a third-generation gravitational wave observatory *Class. Quantum Grav.* **27** 194002
[26] Adhikari R X *et al* 2019 Astrophysical science metrics for next-generation gravitational-wave detectors *Class. Quantum Grav.* **36** 245010
[27] Amaro-Seoane P *et al* 2017 Laser interferometer space antenna (arXiv:1702.00786 [astro-ph.IM])
[28] Ackley K *et al* 2020 Neutron star extreme matter observatory: a kilohertz-band gravitational-wave detector in the global network *Publ. Astron. Soc. Austral.* **37** e047
[29] Luo J *et al* 2016 TianQin: a space-borne gravitational wave detector *Class. Quantum Grav.* **33** 035010
[30] Bugner M, Dietrich T, Bernuzzi S, Weyhausen A and Bernd B 2016 Solving 3D relativistic hydrodynamical problems with weighted essentially nonoscillatory discontinuous Galerkin methods *Phys. Rev.* D **94** 084004
[31] Hilditch D, Weyhausen A and Bernd B 2016 Pseudospectral method for gravitational wave collapse *Phys. Rev.* D **93** 063006
[32] Shankar S, Mösta P, Brandt S R, Haas R, Schnetter E and de Graaf Y 2022 GRaM-X: a new GPU-accelerated dynamical spacetime GRMHD code for exascale computing with the Einstein Toolkit (arXiv:1702.00786 [astro-ph.IM])
[33] Schnetter E, Brandt S, Cupp S, Haas R, Mösta P and Shankar S 2022 CarpetX (0.1.0) *Zenodo* (https://doi.org/10.5281/zenodo.6131529)
[34] Fernando M, Neilsen D, Lim H, Hirschmann E and Sundar H 2019 Massively parallel simulations of binary black hole Intermediate-Mass-Ratio inspirals *SIAM J. Sci. Comput.* **41** C97–C138
[35] Sven K 2018 Towards an exascale code for GRMHD on dynamical spacetimes *J. Phys.: Conf. Ser.* **1031** 012017
[36] Daszuta B, Zappa F, Cook W, Radice D, Bernuzzi S and Morozova V 2021 GR-Athena++: puncture evolutions on vertex-centered oct-tree adaptive mesh refinement *Astrophys. J. Supp.* **257** 25
[37] Clough K, Figueras P, Finkel H, Kunesch M, Lim E A and Tunyasuvunakool S 2015 GRChombo: numerical relativity with adaptive mesh refinement *Class. Quantum Grav.* **32** 245011
[38] Andrade T *et al* 2021 GRChombo: an adaptable numerical relativity code for fundamental physics *J. Open Source Softw.* **6** 3703
[39] Kidder L E *et al* 2017 SpECTRE: a task-based discontinuous galerkin code for relativistic astrophysics *J. Comput. Phys.* **335** 84–114
[40] Deppe N *et al* 2022 Simulating magnetized neutron stars with discontinuous Galerkin methods *Phys. Rev.* D **105** 123031
[41] Rosswog S and Diener P 2021 SPHINCS_BSSN: a general relativistic Smooth Particle Hydrodynamics code for dynamical spacetimes *Class. Quantum Grav.* **38** 115002
[42] Cockburn B and Shu C-W 1991 The Runge–Kutta local projection P^1-discontinuous-Galerkin finite element method for scalar conservation laws M^2AN **25** 337–61
[43] Cockburn B and Shu C-W 1989 TVB Runge–Kutta local projection discontinuous galerkin finite element method for conservation laws II: general framework *Math. Comput.* **52** 411–35
[44] Cockburn B, Lin S-Y and Shu C-W 1989 TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems *J. Comput. Phys.* **84** 90–113
[45] Cockburn B, Hou S and Shu C-W 1990 The Runge–Kutta local projection discontinuous galerkin finite element method for conservation laws. IV: the multidimensional case *Math. Comput.* **54** 545–81
[46] Cockburn B and Shu C-W 1998 The Runge–Kutta discontinuous galerkin method for conservation laws V: multidimensional systems *J. Comput. Phys.* **141** 199–224
[47] Radice D and Rezzolla L 2011 Discontinuous Galerkin methods for general-relativistic hydrodynamics: formulation and application to spherically symmetric spacetimes *Phys. Rev.* D **84** 024010

[48] Teukolsky S A 2016 Formulation of discontinuous Galerkin methods for relativistic astrophysics *J. Comput. Phys.* **312** 333–56
[49] Dumbser M, Guercilena F, Köppel S, Rezzolla L and Zanotti O 2018 Conformal and covariant Z4 formulation of the Einstein equations: strongly hyperbolic first-order reduction and solution with discontinuous Galerkin schemes *Phys. Rev.* D **97** 084053
[50] Fambri F, Dumbser M, Köppel S, Rezzolla L and Zanotti O 2018 ADER discontinuous Galerkin schemes for general-relativistic ideal magnetohydrodynamics *Mon. Not. R. Astron. Soc.* **477** 4543–64
[51] Cao Z, Pei F, Li-Wei J and Xia Y 2018 Binary black hole simulation with an adaptive finite element method II: application of local discontinuous galerkin method to Einstein equations (arXiv:1805.10640 [gr-qc])
[52] Deppe N, Hébert F, Kidder L E and Teukolsky S A 2022 A high-order shock capturing discontinuous Galerkin-finite-difference hybrid method for GRMHD *Class. Quant. Grav.* **39** 195001
[53] Arnowitt R, Deser S and Misner C W 1962 The dynamics of general relativity *Gravitation: An Introduction to Current Research* ed L Witten (New York: Wiley) pp 227–65 (arXiv: gr-qc/0405109)
[54] Abramowitz M and Stegun I A (eds) 1972 *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables* 10th printing edn (Washington: U S Government Printing Office)
[55] Hébert Fçois, Kidder L E and Teukolsky S A 2018 General-relativistic neutron star evolutions with the discontinuous Galerkin method *Phys. Rev.* D **98** 044041
[56] Tichy W, Rashti A, Dietrich T, Dudi R and Bernd B 2019 Constructing binary neutron star initial data with high spins, high compactnesses and high mass ratios *Phys. Rev.* D **100** 124046
[57] Bugner M 2018 Discontinuous galerkin methods for general relativistic hydrodynamics *PhD Thesis* Jena Dissertation, Friedrich-Schiller-Universität Jena, 2017
[58] Lee Lindblom M A Scheel L E Kidder R O and Rinne O 2006 A New generalized harmonic evolution system *Class. Quantum Grav.* **23** S447–62
[59] Baumgarte T W and Shapiro S L 2010 *Numerical Relativity, Solving Eisntein's Equations on the Computer* (New York: Cambridge University Press)
[60] Bruegmann B 2013 A pseudospectral matrix method for time-dependent tensor fields on a spherical shell *J. Comput. Phys.* **235** 216–40
[61] Tichy W 2017 The initial value problem as it relates to numerical relativity *Rept. Prog. Phys.* **80** 026901
[62] Banyuls F, Font J A, Ibánez J M, Martí J M and Miralles J A 1997 Numerical 3+1 general-relativistic hydrodynamics: a local characteristic approach *Astrophys. J.* **476** 221
[63] Galeazzi F, Kastaun W, Rezzolla L and Font J A 2013 Implementation of a simplified approach to radiative transfer in general relativity *Phys. Rev.* D **88** 064009
[64] Gottlieb S, Shu C-W and Tadmor E 2001 Strong stability-preserving high-order time discretization methods *SIAM Rev.* **43** 89–112
[65] Font J A, Miller M, Suen W M and Tobias M 2000 Three-dimensional numerical general relativistic hydrodynamics: formulations, methods and code tests *Phys. Rev.* D **61** 044011
[66] Dimmelmeier H, Font J A and Müller E 2002 Relativistic simulations of rotational core collapse. I. Methods, initial models and code tests *Astron. Astrophys.* **388** 917–35
[67] Baiotti L, Hawke I, Montero P J, Löffler F, Rezzolla L, Stergioulas N, Font J A and Seidel E 2005 Three-dimensional relativistic simulations of rotating neutron star collapse to a kerr black hole *Phys. Rev.* D **71** 024035
[68] Yamamoto T, Shibata M and Taniguchi K 2008 Simulating coalescing compact binaries by a new code SACRA *Phys. Rev.* D **78** 064054
[69] Rezzolla L and Zanotti O 2013 *Relativistic Hydrodynamics* (Oxford: Oxford University Press)
[70] Baiotti L and Rezzolla L 2017 Binary neutron star mergers: a review of Einstein's richest laboratory *Rept. Prog. Phys.* **80** 096901
[71] Poudel A, Tichy W, Brügmann B and Dietrich T 2020 Increasing the accuracy of binary neutron star simulations with an improved vacuum treatment *Phys. Rev.* D **102** 104014
[72] Font J A, Goodale T, Sai Iyer M A Miller L R, Seidel E, Stergioulas N, Suen W-M and Tobias M 2002 Three-dimensional general relativistic hydrodynamics. 2. long term dynamics of single relativistic stars *Phys. Rev.* D **65** 084024
[73] McDermott P N, van Horn H M and Scholl J F 1983 Nonradial g-mode oscillations of warm neutron stars *Astrophys. J.* **268** 837–48

[74] Shu C-W 1987 TVB Uniformly High-Order Schemes for Conservation Laws *Math. Comput.* **49** 105–21
[75] Schaal K, Bauer A, Chandrashekar P, Pakmor R, Klingenberg C and Springel V 2015 Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement *Mon. Not. R. Astron. Soc.* **453** 4278–300
[76] Moe S A, Rossmanith J A and Seal D C 2015 A simple and effective high-order shock-capturing limiter for discontinuous galerkin methods (arXiv:1507.03024)
[77] Martí J M and Müller E 1999 Numerical hydrodynamics in special relativity *Living Rev. Relativ.* **2** 3
[78] Zhao J and Tang H 2013 Runge–Kutta discontinuous galerkin methods with weno limiter for the special relativistic hydrodynamics *J. Comput. Phys.* **242** 138–68