A hp-adaptive discontinuous Galerkin solver for elliptic equations in numerical relativity

Trevor Vincent,[1, 2] Harald P. Pfeiffer,[3, 1] and Nils L. Fischer[3]

[1]*Canadian Institute for Theoretical Astrophysics, University of Toronto, Toronto, Ontario M5S 3H8, Canada*
[2]*Department of Physics, University of Toronto, Toronto, Ontario, M5S 3H5, Canada*
[3]*Max Planck Institute for Gravitational Physics (Albert Einstein Institute), Am Mühlenberg 1, Potsdam 14476, Germany*

(Dated: July 4, 2019)

A considerable amount of attention has been given to discontinuous Galerkin methods for hyperbolic problems in numerical relativity, showing potential advantages of the methods in dealing with hydrodynamical shocks and other discontinuities. This paper investigates discontinuous Galerkin methods for the solution of elliptic problems in numerical relativity. We present a novel hp-adaptive numerical scheme for curvilinear and non-conforming meshes. It uses a multigrid preconditioner with a Chebyshev or Schwarz smoother to create a very scalable discontinuous Galerkin code on generic domains. The code employs compactification to move the outer boundary near spatial infinity. We explore the properties of the code on some test problems, including one mimicking Neutron stars with phase transitions. We also apply it to construct initial data for two or three black holes.

I. INTRODUCTION

Discontinuous Galerkin (DG) methods [1–6] have matured into standard numerical methods to simulate a wide variety of partial differential equations. In the context of numerical relativity [7], discontinuous Galerkin methods have shown advantages for relativistic hyperbolic problems over traditional discretization methods such as finite difference, finite volume and spectral finite elements [8–22] by combining the best aspects of all three methods. As computers reach exa-scale power, new methods like DG are needed to tackle the biggest problems in numerical relativity such as realistic supernovae and binary neutron-star merger simulations on these very large machines [21].

DG efforts in numerical relativity have so far targeted evolutionary problems [8–12, 16–20, 22]. Radice and Rezzolla [12] showed for spherically symmetric problems that the discontinuous Galerkin method could handle strong relativistic shock waves while maintaining exponential convergence rates in smooth regions of the flow. Teukolsky [16] showed how to develop discontinuous Galerkin methods for applications in relativistic astrophysics. Bugner et al. [17] presented the first three-dimensional simulations of general relativistic hydrodynamics with a fixed spacetime background using a discontinuous Galerkin method coupled with a WENO algorithm. Kidder et al. [21] showcased the first discontinuous Galerkin code to use a task-based parallelism framework for applications in relativisitic astrophysics. Kidder et al. tested the scalability and convergence of the code on relativistic magnetohydrodynamics problems. Miller et al. [20] developed a discontinuous Galerkin operator method for use in finite difference codes and used it to solve gauge wave problems involving the BSSN formulation of the Einstein field equations. Hebert et al. [23] presented the first discontinuous Galerkin method for evolving neutron stars in full General Relativity. Finally, Fambri et al. [22] used an ADER discontinuous Galerkin scheme to solve general relativistic ideal magnetohydrodynamics problems in fixed spacetimes. They compared their DG method to a finite volume scheme and showed that DG is much more efficient.

This paper explores DG for elliptic problems in numerical relativity. We develop an elliptic solver with the following primary features: (i) It operates on curved meshes, with non-conforming elements. (ii) It supports adaptive h and p refinement, driven by a posteriori error estimators. (iii) It employs multi-grid for efficient solution of the resulting linear systems. (iv) It uses compactified domains to treat boundary conditions at infinity. While each of these features has appeared in the literature before [2, 24–29], to our knowledge, our solver for the first time combines all these elements simultaneously and demonstrate their effectiveness on difficult numerical problems.

Specifically, the present article is a step toward a solver for the Einstein constraint equations, which must be solved to construct initial data for the evolution of compact binary systems [7, 30, 31]. The constraint equations are generally rewritten as elliptic equations, and depending on detailed assumptions, this results in one or more coupled non-linear elliptic partial differential equations. Construction of initial data is arguably the most important elliptic problem in numerical relativity, but not the only one: Elliptic equations also occur in certain gauge conditions [7] or for implicit time-stepping to alleviate the computational cost of high-mass-ratio binaries [32, 33] .

The motivation for developing a new solver is multi-fold. First, current spectral methods have difficulty obtaining certain initial data sets, such as binaries at short separation containing a neutron star, where the neutron star has high compactness and a realistic equation of state [34]. Furthermore, there is a need for a solver which can routinely obtain high-accuracy. Errors from inaccurate initial data sets creep into the evolutions with sizeable effect: Ref. [35] shows that despite only global (local) differences of 0.02% (1%) in the initial data of the two codes COCAL and LORENE for irrotational neutron-star binaries, the gravitational wave phase at the merger time differed by 0.5 radians after 3 orbits. The design of a more accurate code requires adaptive mesh refinement, load-balancing and scalability which a DG code potentially can provide. The present work also complements the DG evolution code presented in Ref. [21], leading to a complete framework for solving both elliptic and hyperbolic PDEs in numerical relativity.

The organization of the paper is as follows: Section II presents the components of our discontinuous Galerkin code.

Section III showcases our hp-adaptive multigrid solver on increasingly challenging test-problems, each illustrating the power of the discontinuous Galerkin method. This section includes a solution for the Einstein constraint equations in the case of a constant density star. This problem has a surface discontinuity which mimicks Neutron stars with phase transitions. Lastly, this section also presents solutions for initial-data of two orbiting non-spinning black-holes to showcase and compare it with the elliptic solver Spells [36] as well as solutions for the initial data of three black-holes with random locations, spins and momenta. We close with a discussion in Sec. IV.

II. NUMERICAL ALGORITHM

A. DGFEM discretization

We base our nomenclature on [24, 27–29, 37, 38], which we summarize here for completeness.

1. Mesh

Our purpose is to solve elliptic equations in a computational domain $\Omega \subset \mathbb{R}^d$ in d dimensions. To allow for nontrivial topology and shape of Ω, we introduce a *macro-mesh* (also known as a multi-block mesh) $\mathbb{E}_0$ consisting of macro-elements (also known as blocks) $e_0 \in \mathbb{E}_0$ such that (1) the macro-elements cover the entire domain, i.e., $\cup_{e_0 \in \mathbb{E}_0} e_0 = \Omega$, (2) the macro-elements touch each other at complete faces and do not overlap; and (3) each $e_0 \in \mathbb{E}_0$ is the image of the reference cube $[-1, 1]^d$ under a mapping $\Phi_{e_0} : [-1, 1]^d \rightarrow e_0$. As an example, Fig. 1 shows a macro-mesh of five elements covering a two-dimensional disk. The macro-mesh represents the coarsest level of subsequent mesh-refinement.

The macro-mesh $\mathbb{E}_0$ is refined by subdividing macro-elements into smaller elements along faces of constant reference cube coordinates. Refinement can be multiple levels deep (i.e. refined elements can be further subdivided), and we do not assume uniform refinement. The refined mesh $\mathbb{E}$ is referred to as the *fine mesh*, which is exemplified in panel (b) of Fig. 1. In contrast to the macro-mesh, the fine mesh will generally be non-conforming at element boundaries, both within one macro-element and at boundaries between macro-elements. We assume that there is at most a 2:1 refinement difference at any element boundary, i.e. the boundary of a coarse element faces at most two smaller elements (per dimension). Each face of each element $e \in \mathbb{E}$ is endowed with an outward-pointing unit-normal $\hat{n}$. The map Φ_{e_0} in macro-element $e_0 \in \mathbb{E}_0$ induces a map on each fine element e within e_0, denoted

$$\Phi_e : [-1, 1]^d \rightarrow e. \tag{1}$$

The reference coordinates of Φ_e are linearly related to the reference coordinates of Φ_{e_0}.

Turning to boundaries, we define the set of all element boundaries (internal and external), $\Gamma = \cup_{e \in \mathbb{E}} \partial e$, where ∂e is the boundary of element e of the fine mesh. Γ is called the *mortar*, and it decomposes into a finite set of $(d-1)$-dimensional mortar elements $m \in \mathbb{M}$, arising from the faces of each mesh element $e \in \mathbb{E}$, s.t. $\Gamma = \cup_{m \in \mathbb{M}} m$ and each mortar element intersects at most at the boundary of two elements. $\mathbb{M}$ splits into *interior mortar elements* $\mathbb{M}_\mathrm{I}$ and *exterior mortar elements* $\mathbb{M}_\mathrm{B}$, with $\mathbb{M}_\mathrm{I} \cap \mathbb{M}_\mathrm{B} = 0$. Similarly, Γ splits into *interior mortar* $\Gamma_\mathrm{I} = \cup_{m \in \mathbb{M}_I} m$ and *exterior mortar* $\Gamma_\mathrm{B} = \cup_{m \in \mathbb{M}_\mathrm{B}} m$, cf. Fig. 1, which intersect at $(d-2)$-dimensional edges where the interior mortar touches the exterior boundary. We partition the exterior mortar elements further into elements where we apply Dirichlet boundary conditions, $\mathbb{M}_\mathrm{D}$, Neumann boundary conditions, $\mathbb{M}_\mathrm{N}$, and Robin boundary conditions, $\mathbb{M}_\mathrm{R}$, respectively. This induces sets of boundary points via $\Gamma_\mathrm{D} = \cup_{m \in \mathbb{M}_\mathrm{D}} m$, $\Gamma_\mathrm{N} = \cup_{m \in \mathbb{M}_\mathrm{N}} m$ and $\Gamma_\mathrm{R} = \cup_{m \in \mathbb{M}_\mathrm{R}} m$. We assume Γ_D, Γ_N and Γ_R are disjoint, i.e. one type of boundary condition is employed on each connected part of the boundary. Finally, we introduce two definitions to help us simplify equations later in the text. Firstly, because internal boundaries and external Dirichlet boundaries are often treated similarly, it is convenient to define $\mathbb{M}_\mathrm{ID} = \mathbb{M}_\mathrm{I} \cup \mathbb{M}_\mathrm{D}$ and $\Gamma_\mathrm{ID} = \Gamma_\mathrm{I} \cup \Gamma_\mathrm{D}$. Secondly, we refer to the set of mortar elements surrounding an element e as $\mathbb{M}_e$.

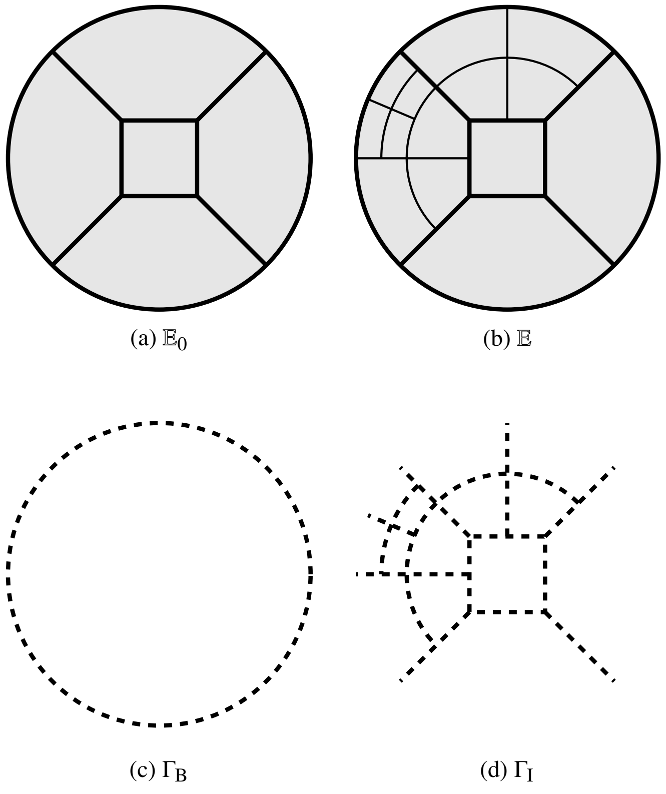

FIG. 1. Ingredients into the setup of the domain-decomposition: (a) The macro-mesh $\mathbb{E}_0$ of a 2-dimensional disk consisting of five macro-elements. (b) A representative mesh derived from the macro-mesh by refining once in the left-most macro-element. (c) The boundary mortar Γ_B. (d) The interior mortar Γ_I.

2. Weak Equations

On the fine mesh $\mathbb{E}$, we wish to discretize the following model elliptic problem with the symmetric interior penalty discontinuous Galerkin method [24]:

$$\partial_i\partial_i u(\boldsymbol{x}) + f(\boldsymbol{x})u = g(\boldsymbol{x}) \qquad \boldsymbol{x}\in\Omega, \tag{2a}$$

$$u = g_D(\boldsymbol{x}) \qquad \boldsymbol{x}\in\Gamma_{\rm D}, \tag{2b}$$

$$\hat{n}_i\partial_i u = g_N(\boldsymbol{x}) \qquad \boldsymbol{x}\in\Gamma_{\rm N}, \tag{2c}$$

$$\hat{n}_i\partial_i u + \gamma u = g_R(\boldsymbol{x}) \qquad \boldsymbol{x}\in\Gamma_{\rm R}. \tag{2d}$$

In Eqs. (2) and in the following, we employ the Einstein sum convention, so that the first term in Eq. (2a) represents $\sum_{i=1}^{d}\partial_i\partial_i u$, and similar for the left hand side of Eqs. (2c) and (2d). In accordance with the underlying ideas of DG, we seek to approximate the solution of Eqs. (2) with polynomials on each element $e\in\mathbb{E}$, without strictly enforcing continuity across elements. Let V_{h,e,p_e} denote the set of polynomials on the reference cube $[-1,1]^d$ up to degree p_e mapped to element $e\in\mathbb{E}$. We assume the same maximum polynomial order along each dimension; it is straightforward to extend to different polynomial order along different dimensions. The functions in V_{h,e,p_e} are understood to be extended by 0 outside of e (i.e. on other elements). The global function space is the direct sum of the per-element polynomial spaces,

$$V_h = \bigoplus_{e\in\mathbb{E}} V_{h,e,p_e}, \tag{3}$$

where the polynomial order may vary between elements. Because neighboring element touch, on internal boundaries $\Gamma_{\rm I}$ the discretized solution $u_h\in V_h$ will be represented twice on touching elements with generally different values on either element (this is origin of the term 'discontinuous' in DG).

The discretized solution $u_h\in V_h$ is determined, such that the residual of Eqs. (2a) is orthogonal to the function space V_h. Within the symmetric interior penalty discontinuous Galerkin discretization [24, 38], this yields

$$L_h(u_h,v_h) = F_h(v_h) \qquad \forall v_h\in V_h, \tag{4}$$

where

$$\begin{aligned}L_h(u,v) = &\int_{\mathbb{E}}\partial_i v\partial_i u\,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_{\rm ID}} [\![u]\!]_i\,\{\!\{\partial_i v\}\!\}\,\mathrm{d}\boldsymbol{s}\\ &- \int_{\Gamma_{\rm ID}} [\![v]\!]_i\,\{\!\{\partial_i u\}\!\}\,\mathrm{d}\boldsymbol{s} + \int_{\Gamma_{\rm ID}}\sigma[\![u]\!]_i[\![v]\!]_i\mathrm{d}\boldsymbol{s}\\ &+ \int_{\mathbb{E}} fuv\,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_R}\gamma uv\,\mathrm{d}\boldsymbol{s},\end{aligned} \tag{5}$$

and

$$\begin{aligned}F_h(v) = &\int_{\mathbb{E}} gv\mathrm{d}x - \int_{\Gamma_D} g_D\hat{n}_i\partial_i v\mathrm{d}\boldsymbol{s} + \int_{\Gamma_D}\sigma g_D v\mathrm{d}\boldsymbol{s}\\ &+ \int_{\Gamma_N} g_N v\mathrm{d}\boldsymbol{s} - \int_{\Gamma_R} g_R v\mathrm{d}\boldsymbol{s}.\end{aligned} \tag{6}$$

For internal boundaries, the operators $[\![\,.\,]\!]$ and $\{\!\{\,.\,\}\!\}$ are defined by

$$[\![q]\!] = q^+\boldsymbol{n}^+ + q^-\boldsymbol{n}^- \quad \text{on } \Gamma_I, \tag{7}$$

$$\{\!\{q\}\!\} = \frac{1}{2}\left(q^+ + q^-\right) \quad \text{on } \Gamma_I. \tag{8}$$

Here q is a scalar function, '+' and '-' is an arbitrary labeling of the two elements e^+ and e^- touching at the interface, and $q^\pm$ and $\boldsymbol{n}^\pm$ are the function values and the outward pointing unit normal on the two elements that share the interface. These operators are extended to external boundaries by

$$[\![q]\!] = q\boldsymbol{n} \quad \text{on } \Gamma_B, \tag{9}$$

$$\{\!\{q\}\!\} = q \quad \text{on } \Gamma_B. \tag{10}$$

Breaking up the integrals in Eqs. (5) and (6) into integrals over individual mesh– and mortar–elements, one finds

$$\begin{aligned}L_h(u,v) = &\sum_{e\in\mathbb{E}}\int_e \partial_i v\partial_i u\,\mathrm{d}\boldsymbol{x} - \sum_{m\in\mathbb{M}_{\rm ID}}\int_m [\![u]\!]_i\,\{\!\{\partial_i v\}\!\}\,\mathrm{d}\boldsymbol{s}\\ &- \sum_{m\in\mathbb{M}_{\rm ID}}\int_m [\![v]\!]_i\,\{\!\{\partial_i u\}\!\}\,\mathrm{d}\boldsymbol{s} + \sum_{m\in\mathbb{M}_{\rm ID}}\int_m \sigma[\![u]\!]_i[\![v]\!]_i\mathrm{d}\boldsymbol{s}\\ &- \sum_{m\in\mathbb{M}_R}\int_m \gamma uv\mathrm{d}\boldsymbol{s} + \sum_{e\in\mathbb{E}}\int_e fuv\,\mathrm{d}\boldsymbol{x},\end{aligned} \tag{11}$$

and

$$\begin{aligned}F_h(v) = &\sum_{e\in\mathbb{E}}\int_e gv\mathrm{d}x - \sum_{m\in\mathbb{M}_{\rm D}}\int_m g_D\hat{n}_i\partial_i v\mathrm{d}\boldsymbol{s} + \sum_{m\in\mathbb{M}_{\rm D}}\int_m \sigma g_D v\mathrm{d}\boldsymbol{s}\\ &+ \sum_{m\in\mathbb{M}_{\rm N}}\int_m g_N v\mathrm{d}\boldsymbol{s} - \sum_{m\in\mathbb{M}_{\rm R}}\int_m g_R v\mathrm{d}\boldsymbol{s}.\end{aligned} \tag{12}$$

For later reference, we will refer to the first four integrals in Eq. (11) as the stiffness, consistency, symmetry and penalty integrals respectively and denote them $L_h^{\rm stiff}(u,v)$, $L_h^{\rm con}(u,v)$, $L_h^{\rm sym}(u,v)$ and $L_h^{\rm pen}(u,v)$ so that we may write

$$\begin{aligned}L_h(u,v) = &L_h^{\rm stiff}(u,v) + L_h^{\rm con}(u,v) + L_h^{\rm sym}(u,v) + L_h^{\rm pen}(u,v)\\ &+ \sum_{e\in\mathbb{E}}\int_e fuv\,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_R}\gamma uv\,\mathrm{d}\boldsymbol{s}.\end{aligned} \tag{13}$$

3. Basis Functions

So far, we have not yet specified a concrete basis for the polynomial spaces V_{h,e,p_e} introduced in Sec. II A 2. We do so now. Recall that curvilinear elements $e\in\mathbb{E}$ are mapped to reference cubic elements $[-1,1]^d$. That is, each point $\boldsymbol{x}\in e$ corresponds to a reference cube coordinate $\boldsymbol{\xi} = \Phi_e^{-1}(\boldsymbol{x})$, cf. Eq. (1).

Along each dimension $\xi_i\in[-1,1]$, where the subscript $i\in\{1,\ldots,d\}$ denotes dimension, we first choose N_i+1

Legendre-Gauss-Lobatto collocation points $\xi^{\text{LGL}}_{\alpha_i}$. The index $\alpha_i \in \{1, \ldots, N_i + 1\}$ identifies the point along dimension i. We take $N_i \geq 1$ so that the points with $\xi_i = -1$ and $\xi_i = 1$, that lie on the faces of the cube, are always collocation points. The collocation points in all d dimensions form a tensor product grid of $\prod_{i=1}^{d}(N_i + 1)$ d-dimensional collocation points $\boldsymbol{\xi}^{\text{LGL}}_{\alpha} = (\xi^{\text{LGL}}_{\alpha_1}, \ldots, \xi^{\text{LGL}}_{\alpha_d})$ that we index by $\alpha \in \{1, \ldots, \prod_{i=1}^{d}(N_i + 1)\}$ that identifies a point regardless of dimension. With respect to these collocation points we can now construct the one-dimensional interpolating Lagrange polynomials

$$l_{\alpha_i}(\xi) := \prod_{\substack{\beta_i=1 \\ \beta_i \neq \alpha_i}}^{N_i+1} \frac{\xi - \xi^{\text{LGL}}_{\beta_i}}{\xi^{\text{LGL}}_{\alpha_i} - \xi^{\text{LGL}}_{\beta_i}}, \quad \xi \in [-1, 1] \tag{14}$$

and employ their tensor product to define the d-dimensional basis functions

$$\psi_{\alpha}(\boldsymbol{\xi}) = \prod_{i=1}^{d} l_{\alpha_i}(\xi_i). \tag{15}$$

When evaluating ψ_α in the physical coordinates $\boldsymbol{x} \in e$, one uses $\Phi_e^{-1} : \boldsymbol{x} \longrightarrow \boldsymbol{\xi}$ to map the function argument. For instance, the set of test-functions on element $e \in \mathbb{E}$ becomes

$$V_{h,e,p_e} = \text{span}\left\{ \Phi_e^{-1} \circ \psi_\alpha \right\}. \tag{16}$$

Furthermore, because the ψ_α form a nodal basis, the expansion coefficients are the function values at the nodal points $\boldsymbol{\xi}^{\alpha}$, and each test-function can be written as

$$u_h^e = \sum_{\alpha} u_h^e(\boldsymbol{\xi}^{\text{LGL}}_{\alpha})\psi_\alpha = \sum_{\alpha} u_\alpha^e \psi_\alpha, \tag{17}$$

where $u_\alpha^e := u_h^e(\boldsymbol{\xi}^{\text{LGL}}_{\alpha})$.

4. *Semi-Discrete Global Matrix Equations*

The global solution over the entire mesh is the direct sum of the solutions on each element, that is

$$u_h = \bigoplus_{e \in \mathbb{E}} u_h^e = \bigoplus_{e \in \mathbb{E}} \sum_{\alpha} u_\alpha^e \psi_\alpha^e. \tag{18}$$

Thus, with a chosen global ordering of elements $e \in \mathbb{E}$ and local ordering of basis functions ψ_α^e, we may prescribe a global index α' to each of the local expansion coefficients u_α^e. With this, we may now turn Eqs. (11) and (12) into a global linear system of equations. Let $\mathbb{L} = L_h(\psi_{\alpha'}, \psi_{\beta'}) =: L_{\alpha'\beta'}$, $\boldsymbol{u} = u_{\beta'}$ and $\boldsymbol{F} = F(\psi_{\alpha'}) =: F_{\alpha'}$, then the global linear system is

$$\mathbb{L}\boldsymbol{u} = \boldsymbol{F}. \tag{19}$$

Instead of forming the full global matrix $\mathbb{L}$ and performing the matrix-vector operator $\mathbb{L}\boldsymbol{u}$ over the global space, we can restrict the integrals in (11) and (12) to those pertaining to element e and the mortar elements m of ∂e, and perform elemental matrix-vector operations.

Equation 19 contains integrals and therefore represents only a semi-discrete set of equations. In the next section we show how to make Eq. (19) fully discrete by using quadrature to approximate the integrals.

5. *Quadrature*

The integrals in Eqs. (11) and (12) depend on various geometric objects: We define the Jacobian matrix with respect to the mapping $\boldsymbol{\phi}_e : \boldsymbol{\xi} \longrightarrow \boldsymbol{x}$ on an element as $e \in \mathbb{E}$

$$(\boldsymbol{J})^i_j = \frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{\xi}_j}, \tag{20}$$

with an inverse given by

$$(\boldsymbol{J}^{-1})^i_j = \frac{\partial \boldsymbol{\xi}_i}{\partial \boldsymbol{x}_j}. \tag{21}$$

The Jacobian determinant is

$$J = \det \boldsymbol{J} \tag{22}$$

Similarly, the surface Jacobian for a mortar element $m \in \mathbb{M}$ is

$$S^m = J\sqrt{\frac{\partial \xi_j}{\partial \boldsymbol{x}} \cdot \frac{\partial \xi_j}{\partial \boldsymbol{x}}}, \tag{23}$$

where j is the index of the reference coordinate which is constant on the surface under consideration (no sum-convention). For a mortar on a macro-element boundary, S^m may change depending on which of the two macro-element mappings are used to compute Eqn. (23). This ambiguity is not problematic as long as all quantities in a mortar integrand are transformed to the macro-element coordinate system in which S^m is being computed. The normal is computed by

$$\hat{\boldsymbol{n}} = \text{sgn}(\xi_j)\frac{J}{S^m}\frac{\partial \xi_j}{\partial \boldsymbol{x}}, \tag{24}$$

where sgn is the signum function, where j labels the dimension that is constant on the face under consideration (no sum-convention).

Let us now consider the stiffness integral in Eq. (11). Because basis-functions are local to each element, we can consider each element e individually, and we will omit the superscript e to lighten the notational load. Substituting the expansion of the solution $u_h = u_\alpha \psi_\alpha$, as well as $v = \psi_\beta$, the stiffness integral becomes

$$L^{\text{stiff}}_{h,e}(u_h, \psi_\beta) = \int_e u_\alpha \frac{\partial \psi_\alpha}{\partial x_k}\frac{\partial \psi_\beta}{\partial x_k}\,\mathrm{d}\boldsymbol{x}. \tag{25}$$

We recall that we employ the Einstein sum convention, i.e. Eq. (25) has implicit sums over α and k. The derivatives $\partial\psi_\alpha/\partial\xi_l$ are just polynomials in $\boldsymbol{\xi}$, therefore, they can be re-expanded in our nodal basis as $\partial\psi_\alpha/\partial\xi_l = D^l_{\alpha\beta}\psi_\beta$ with $D^l_{\alpha\beta} := \partial\psi_\alpha/\partial\xi_l(\xi^\beta)$. The physical derivative $\frac{\partial\psi_\alpha}{\partial x_k}$ is then $\frac{\partial\psi_\alpha}{\partial x_k} = (J^{-1})^l_k D^l_{\alpha\beta}\psi_\beta$. The tensor-grid in Eq. (15) implies that the matrices $D^l_{\alpha\beta}$ are sparse for $d > 1$. Using these expressions, and converting to an integral in the reference coordinates, we obtain

$$L^{\text{stiff}}_{h,e}(u_h, v_h) = u_\alpha \int\limits_{[-1,1]^d} (J^{-1})^l_k D^l_{\alpha\gamma}\psi_\gamma (J^{-1})^m_k D^m_{\beta\delta}\psi_\delta \, J \, \mathrm{d}\boldsymbol{\xi} \quad (26)$$

$$= u_\alpha L^{\text{stiff},e}_{\alpha\beta}. \quad (27)$$

We evaluate the integral in Eq. (26) with Gauss-Legendre (GL) quadrature. This choice follows [39] in the use of a stronger set of quadrature points (higher order GL grids) to decrease the error in geometric aliasing. We denote the GL abscissas and weights by $\xi^{(c)}_{\text{GL}}$ and $w^{(c)}_{\text{GL}}$, respectively, where $c = 1, \ldots, N_{\text{GL}}$. In multiple dimensions these will be a tensor product of the 1-d GL abscissas and weights denotes by $\boldsymbol{\xi}^\sigma_{\text{GL}}$ and w^σ_{GL}, respectively. All non-polynomial functions, including geometric quantities such as $J, (J^{-1})^i_j, S^m, \hat{\boldsymbol{n}}$, are evaluated directly at $\boldsymbol{\xi}^\sigma_{\text{GL}}$, whereas polynomial functions like the trial function u_h and the test function ψ_h —which naturally are represented on the Legendre-Gauss-Lobatto grid— are interpolated to the GL–quadrature points. Denoting the interpolation matrix by $I_{\sigma\alpha} := \psi_\alpha(\boldsymbol{\xi}^\sigma_{\text{GL}})$, we find

$$L^{\text{stiff},e}_{\alpha\beta} \approx \sum_\sigma w_\sigma \left(J\,(\boldsymbol{J}^{-1})^l_k D^l_{\alpha\gamma}\psi_\gamma (\boldsymbol{J}^{-1})^m_k D^m_{\beta\delta}\psi_\delta \right)\Bigg|_{\xi^\sigma_{GL}} \quad (28)$$

$$= D^l_{\alpha\gamma} I_{\sigma\gamma} (\boldsymbol{J}^{-1}_\sigma)^l_k \, w_\sigma J_\sigma (\boldsymbol{J}^{-1}_\sigma)^m_k I_{\sigma\delta} D^m_{\beta\delta}. \quad (29)$$

Here $I_{cm} = \psi_m(\xi^\zeta_{GL})$ is an interpolation matrix from the GLL points to the GL points. The other volume integrals in Eq. (13) are computed in a similar manner.

The mortar integrals are a bit more involved owing to the extra book-keeping arising from the two elements (named '+' and '-') that touch at the boundary $m \in \mathbb{M}$, each with their own local basis-functions, denoted by ψ^-_α and ψ^+_α. Taking the penalty-integral as an example, for test-functions v_h that are a basis-function of the '-' element, the definition Eq. (7) yields

$$L^{\text{pen}}_{m,h}(u_h, \psi^-_\alpha) = \int_m \sigma [\![u_h]\!]_k [\![\psi^-_\alpha]\!]_k \mathrm{d}\boldsymbol{s} \quad (30)$$

$$= \int_{[-1,1]^{d-1}} \sigma \left(u^-_h - u^+_h\right) \psi^-_\alpha \, S^m \, \mathrm{d}\boldsymbol{\xi}. \quad (31)$$

Equation (31) contains both u^-_h and u^+_h; therefore, when substituting in their respective local expansions, $u^\pm_h = u^\pm_\alpha\psi^\pm_\alpha$, and pulling the coefficients outside the integral, we see that the penalty term will result in entries of the global matrix equation (19) that couple the two elements $e^\pm$. Once the coefficients $u^\pm_\alpha$ are moved outside the integral, the remaining integral only depends on basis-functions $\psi^\pm_\alpha$ and geometric quantities. These integrals are evaluated with GL-quadrature, which is expressed in terms of interpolation matrices $I^\pm_{\alpha\rho} := \psi^\pm_\alpha(\boldsymbol{\xi}^\rho_{\text{GL}})$, where ρ labels the GL collocation points on the boundary $[-1, 1]^{d-1}$ and α labels the local basis-functions in $e^\pm$, which are to be evaluated in the suitably mapped $\boldsymbol{\xi}_{\text{GL}}$ locations.

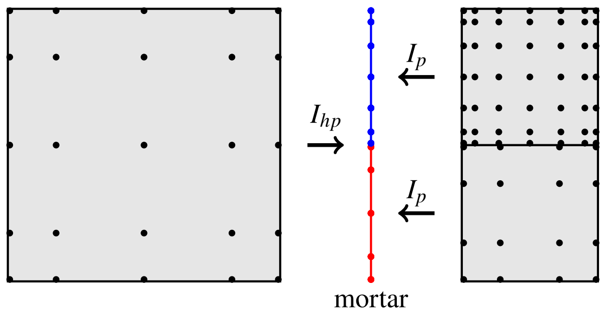

FIG. 2. A representation of a 2:1 interface in two dimensions. On the left we have a 4th-order element and on the right we have two elements of degree 8 and degree 2 respectively. The data on the faces of these three elements will be used in the mortar integrals of equation 5, but since the interface is non-conforming, we need to interpolate the face data to a shared broken polynomial space on the mortar. To ensure that the face data on each element is in a polynomial space which a subset of the polynomial space on the mortar, we demand that the polynomial degree p on the mortar is the maximum of the polynomial degrees on either side. For the lower mortar face (red) this would be max(4, 3) = 4 and for the upper mortar face (green) this would be max(4, 6) = 6. Since the left element must interpolate it's face data to a space containing two faces, we use the hp-interpolation operator $\mathcal{I}_{hp}$. On the right side, each element has to map its data from one face to one face, so we use the p-interpolation operator $\mathcal{I}_p$.

B. Penalty Function

The penalty parameter σ in Eqs. (5) and (11) is a spatially dependent function on boundaries $m \in \mathbb{M}$, defined by

$$\sigma = C\frac{p^2_m}{h_m}, \quad (32)$$

where p_m and h_m represent a typical polynomial degree and a typical length-scale of the elements touching at the boundary, respectively, and where the parameter $C > 0$ is large enough such that $L(\cdot,\cdot)$ is coercive. We choose $p_m = \max(p^+, p^-)$ and we set $h_m = \min(J^+/S^m, J^-/S^m)$ as this has yielded the best results empirically. Here $J^\pm$ is the volume Jacobian on $e^\pm$ mapping $[0, 1]^d$ to $e^\pm$, and S^m is the surface Jacobian mapping $[0, 1]^{d-1}$ to the boundary face of $e^\pm$.

C. Norms

Throughout the following sections we will use the energy norm,

$$||v||_* = \left(\sum_{e\in\mathbb{E}} \int_e |\nabla v|^2 \mathrm{d}\boldsymbol{x} + \sum_{m\in\Gamma} \int_m |\sqrt{\sigma}[\![v]\!]|^2 \mathrm{d}\boldsymbol{s} \right)^{\frac{1}{2}}, \quad (33)$$

Algorithm 1 Multigrid Preconditioned Newton-Krylov: Solves Eq. (35).$N_{\text{iter,NK}}$ and tol_{NK} are user-specified parameters.

```
1: procedure NK($u_0$)
2:     $k \leftarrow 0$
3:     while $k \leq N_{\text{iter,NK}}$ or $||R(u_k)|| \geq \text{tol}_{\text{NK}}$ do
4:         $\delta u_k \leftarrow \text{LSOLVE}(u_k, R(u_k), \text{J}_R(u_k))$
5:         $u_{k+1} \leftarrow u_k + \delta u_k$
6:         $k \leftarrow k + 1$
7:     end while
8:     return $u_k$
9: end procedure
```

and the L_2 norm,

$$||v||_2 = \left(\int_\Omega v^2 \mathrm{d}\boldsymbol{x}\right)^{\frac{1}{2}} = \left(\sum_{e\in\mathbb{E}} \int_e v^2 \mathrm{d}\boldsymbol{x}\right)^{\frac{1}{2}}. \qquad (34)$$

Here, v is a scalar function on Ω.

D. Multigrid-Preconditioned Newton-Krylov

1. Newton-Krylov

In general, a set of elliptic partial differential equations can be written in the form

$$R(\boldsymbol{u}) = 0, \qquad (35)$$

where $\boldsymbol{u}$ is the solution and R is a non-linear elliptic operator. Defining the Jacobian of the system by $\text{J}_R(\boldsymbol{u}) \equiv \frac{\partial R}{\partial \boldsymbol{u}}(\boldsymbol{u})$, Newton-Raphson iteratively refines an initial guess for the solution $\boldsymbol{u}^{(k)}$ by solving the following linear system for the correction $\delta\boldsymbol{u}^{(k)} = \boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)}$:

$$\text{J}_R(\boldsymbol{u}^{(k)})\delta\boldsymbol{u}^{(k)} = -R(\boldsymbol{u}^{(k)}). \qquad (36)$$

Upon discretization as described in Sec. II A, Eq. (36) results in a N × N linear system. Once Eq. (36) is solved, the improved solution is given by $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \delta\boldsymbol{u}^{(k)}$. The Newton-Raphson iterations are continued until the residual $R(\boldsymbol{u}^{(k)})$ is sufficiently small. At each step, the linear system is solved by a linear solver we abbreviate as $LSOLVE$. The full algorithm is described in Algorithm 1.

For three-dimensional problems with a large number N of degrees of freedom, the Jacobian J is too large to form fully and partition across processes. For building a scalable solver, we are therefore limited to iterative solvers, the most popular class of which are the Krylov solvers [40] such as the Conjugate Gradient method or GMRES, which only involve matrix-vector operations. In this paper we use the flexible conjugate gradient Krylov [41] solver at each Newton-Raphson step, as summarized in Algorithm 2, where u_0 denotes the initial guess of the linear solver, and N_{its} is the number of iterations.

Algorithm 2 Flexible Conjugate Gradients: Solves Eq. (36), with the variable u representing $\delta u^{(k)}$. $N_{\text{iter,FCG}}$ and tol_{FCG} are user-specified parameters.

```
1: function LSOLVE($u_0$,$R$,J)
2:     $r_0 \leftarrow \text{J}u_0 + R$
3:     $k \leftarrow 0$
4:     while $k{+}{+} \leq N_{\text{iter,FCG}}$ or $||r_k|| \geq \text{tol}_{\text{FCG}}$ do
5:         $v_k \leftarrow \text{VCYCLE}(r_k, \text{J}v_k)$
6:         $w_k \leftarrow \text{J}v_k$
7:         $\alpha_k \leftarrow v_k \cdot r_k$
8:         $\beta_k \leftarrow v_k \cdot w_k$
9:         if k = 0 then
10:            $d_k \leftarrow v_k$
11:            $q_k \leftarrow w_k$
12:            $\rho_k \leftarrow \beta_k$
13:        else
14:            $\gamma_k \leftarrow v_k \cdot q_{k-1}$
15:            $d_k \leftarrow v_k - (\gamma_k/\rho_{k-1})d_{k-1}$
16:            $q_k \leftarrow w_k - (\gamma_k/\rho_{k-1})q_{k-1}$
17:            $\rho_k \leftarrow \beta_k - \gamma_k^2/\rho_{k-1}$
18:        end if
19:        $u_{k+1} \leftarrow u_k + (\alpha_k/\rho_k)d_k$
20:        $r_{k+1} \leftarrow r_k - (\alpha_k/\rho_k)q_k$
21:    end while
22:    return $u_k$
23: end function
```

A typical Krylov solver will take $\text{O}(\sqrt{\kappa})$ iterations to reach a desired accuracy, where the condition number κ is defined as the ratio of the maximum eigenvalue and minimum eigenvalue of J_R. For the discontinuous galerkin method, the discretized Laplacian matrix has a condition number that grows with p-refinement and h-refinement (see [2, 42] for example) and therefore the number of iterations will grow with each AMR step unless Eq. (36) is preconditioned.

2. Multigrid preconditioner

We use a multigrid V-cycle [43] as a preconditioner for each FCG solve (called on line 4 in Algorithm 2). Multigrid is an multi-level iterative method aimed at achieving mesh-independent error reduction rates through a clever method of solving for the error corrections on coarser grids and prolonging the corrections to the finer grids. For a detailed overview of multigrid methods see [43]. The main drawback of multigrid is its complexity. In this section we will briefly describe the components of the multigrid algorithm employed in solving problems in this paper with hp-grids in parallel with the interior penalty discontinuous Galerkin method.

3. Multigrid Meshes

Multigrid uses a hierarchy of coarsened meshes, labeled with $l = 0, \ldots L$, where $l = L$ represents the fine mesh $\mathbf{E}$ on which the solution is desired, and $l = 0$ represents the coarsest

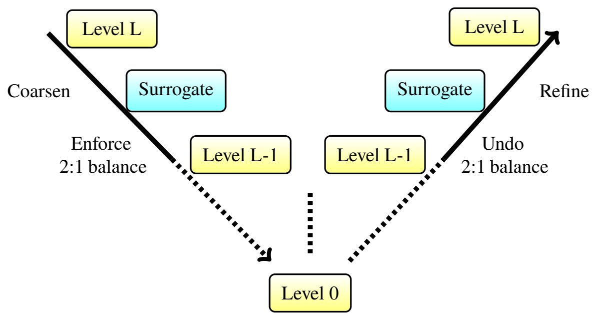

FIG. 3. A structural representation of a multigrid v-cycle. The nodes in yellow are actual grid-levels, while the nodes in blue represent surrogate grids which are not necessarily 2:1 balanced. In order for a multigrid v-cycle to represent a symmetric operation, the grid levels along the down arrow are exactly the same as the grid-levels along the up arrow. Level 0 represents the coarsest possible mesh.

mesh. One V-cycle proceeds from the fine grid to the coarsest grid, and back to the fine grid, as indicated in Fig. 3. We construct the coarse meshes $l = L - 1, \dots 0$ by successively coarsening. Coarsening by combining 2^d elements into one element can lead to interfaces with a 4:1 balance, even if the original mesh is 2:1 balanced. We avoid such 4:1 balance with surrogate meshes, which have been used before in large-scale FEM multigrid solvers, see for example [44, 45]. A surrogate mesh is the naively h-coarsened mesh, as indicated in blue in Fig. 3. If the surrogate mesh has indeed interfaces with 4:1 balance, the desired 2:1 balance is enforced by refining at the unbalanced interfaces. This results in the coarsened mesh on Level $l = L - 1$, and iteratively down to $l = 0$. It is easy to show that at each coarsening the coarse level $l - 1$ mesh is strictly coarser than the level l fine mesh. Following this, we must make a decision as to what polynomial degree we choose for the coarsened elements. We take the minimum polynomial degree of the 2^d children elements for each parent element of the coarsened mesh, ensuring that functions on the coarse grid can be represented exactly on the fine mesh, a property we will utilize below in Eq. (42). Lastly, it is important to note that we never purely p-coarsen on any of the multigrid levels. We do not store the multigrid meshes because we have empirically shown that the coarsening, refining and balancing is not the bottleneck for the multigrid algorithm.

4. *Multigrid algorithm*

Having constructed the coarse meshes, we can now turn to the multigrid algorithm, summarized in Algorithm 3. It consists of several important components most importantly smoothing and inter-mesh operators like coarsening, balancing of the mesh, restriction and prolongation. We now look at each of these in turn.

Algorithm 3 Multigrid Preconditioner Vcycle.
L denotes the number of multigrid Levels.

```
1: function VCYCLE($f_L$, $J_L$)
2:     for $l = L, 1$ do
3:         $v_l \leftarrow 0$
4:         $v_l \leftarrow$ SMOOTHER($v_l, f_l, J_l$)
5:         Coarsen grid
6:         Balance grid
7:         $f_{l-1} \leftarrow I_l^T (f_l - J_l v_l)$                ▷ Restriction
8:     end for
9:     $v_0 \leftarrow 0$
10:    $v_0 \leftarrow$ SMOOTHER($v_0, f_0, J_0$)
11:    for $l = 1, L$ do
12:        Refine grid
13:        $v_l \leftarrow I_l(v_{l-1}) + v_l$                ▷ Prolongation
14:        $v_l \leftarrow$ SMOOTHER($v_l, f_l, J_l$)
15:    end for
16:    return $v_L$
17: end function
```

Algorithm 4 Chebyshev Smoothing
$N_{\text{iter,Cheb}}$ and Λ are user-defined parameters.

```
1: function SMOOTHER($x, b, J$)
2:     $p \leftarrow 0$
3:     $\lambda_{max}, x \leftarrow$ CGEIGS($x, b, J$)
4:     $\lambda_{min} \leftarrow \lambda_{max}/\Lambda$
5:     $c \leftarrow (\lambda_{max} - \lambda_{min})/2$
6:     $d \leftarrow (\lambda_{max} + \lambda_{min})/2$
7:     for $k = 1 \dots N_{\text{iter,Cheb}}$ do
8:         $r \leftarrow b - Jx$
9:         $\alpha \leftarrow \begin{cases} d^{-1} & k = 1 \\ 2d(2d^2 - c^2)^{-1} & k = 2 \\ (d - \alpha c^2/4)^{-1} & k \neq 1, 2 \end{cases}$
10:        $\beta \leftarrow \alpha d - 1$
11:        $p \leftarrow \alpha r - \beta p$
12:        $x \leftarrow x + p$
13:    end for
14:    return $x$
15: end function
```

5. *Multigrid Smoother*

We explore two different smoothers, Chebyshev smoothing and a Schwarz smoother. Both of these avoid explicit storage of the matrix, as required by smoothers like the Gauss-Seidel method [46].

Chebyshev smoothing [47], a type of polynomial smoother [40] requires only matrix-vector operations and has been shown to work satisfactorily in scalable multigrid solvers [48]. Our implementation of Chebyshev smoothing [47] is presented in Algorithm 4. For this algorithm there are two user-defined parameters: $N_{\text{iter,Cheb}}$ and Λ. Typical values we use for $N_{\text{iter,Cheby}}$ are in the range $8 - 15$ and Λ is usually set in the range $10 - 30$. Chebyshev polynomial smoothers require a spectral bound on the eigenvalues of the linear oper-

Algorithm 5 CG Spectral Bound Solver
$N_{\text{iter,eigs}}$ is a user-defined parameter.

```
 1: function CGEIGS($x_1, b, J$)
 2:     $r_1 \leftarrow b - Jx_1$
 3:     $p_1 \leftarrow r_1$
 4:     $\lambda_{\max} \leftarrow 0$
 5:     for $k = 1, N_{\text{iter,eigs}}$ do
 6:         $\alpha_k \leftarrow (r_k \cdot r_k)/(p_k \cdot Jp_k)$
 7:         $x_{k+1} \leftarrow x_k + \alpha_k p_k$
 8:         $r_{k+1} \leftarrow r_k - \alpha_k J p_k$
 9:         $\beta_k \leftarrow (r_{k+1} \cdot r_{k+1})/(r_k \cdot r_k)$
10:         $p_{k+1} \leftarrow r_{k+1} + \beta_k p_k$
11:         $\lambda_k \leftarrow \begin{cases} \frac{1}{\alpha_1} + \frac{\sqrt{\beta_1}}{a1} & \text{if } k = 1 \\ \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}} + \frac{\sqrt{\beta_{k-1}}}{\alpha_{k-1}} & \text{if } k \neq 1 \end{cases}$
12:         $\lambda_{\max} = \max(\lambda_k, \lambda_{\max})$
13:     end for
14: return $\lambda_{\max}, x_{k+1}$
15: end function
```

ator. We use conjugate gradients to estimate the eigenvalues of a general symmetric linear operator, as implemented in Algorithm 5. It can be shown [49] that each iteration of conjugate gradients obtains one row of the underlying linear operator in tri-diagonal form:

$$\begin{pmatrix} \frac{1}{\alpha_1} & -\frac{\sqrt{\beta_2}}{\alpha_1} & 0 & \cdots & 0 \\ -\frac{\sqrt{\beta_2}}{\alpha_1} & \frac{1}{\alpha_2} + \frac{\beta_2}{\alpha_1} & -\frac{\sqrt{\beta_3}}{\alpha_2} & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & -\frac{\sqrt{\beta_{k-1}}}{\alpha_{k-2}} & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-1}}{\alpha_{k-2}} & -\frac{\sqrt{\beta_k}}{\alpha_{k-1}} \\ 0 & \cdots & 0 & \frac{-\sqrt{\beta_k}}{\alpha_{k-1}} & \frac{1}{\alpha_k} + \frac{\beta_k}{\alpha_{k-1}} \end{pmatrix} \tag{37}$$

The values α_k and β_k in this expression are obtained from the conjugate gradients algorithm 5 on lines 6 and 9. Furthermore, the eigenvalues of each of the sub tri-diagonal matrices is a subset of the eigenvalues of the full matrix. So at each iteration we can get an estimate of the bound by using the Gershgorin circle theorem [50]. Our Alg. 5 combines the CG steps with the estimation of the bound $\lambda_{\max}$ of the largest eigenvalue.

Besides estimating the spectral radius, we also use the CG iterations to further smooth the solution. This adds robustness to multigrid algorithms [51].

While the Chebyshev smoother is easy to implement, it is reliant on a robust estimate of the largest eigenvalue and this may not be always true in our case. Thus we also implement an additive Schwarz smoother in the manner of [25], which is much more robust. The Schwarz smoother works by performing local solves on element-centered subdomains and then adding a weighted sum of these local solves to obtain the smoothed solution. A simple example of one of these subdomains (where the grid is both p and h-uniform) is shown in Figure 4. More generally, the Schwarz subdomain centered on element e_c of the mesh is constructed as follows: Starting with all collocation points on e_c, one adds all boundary points of neighboring elements which coincide with faces, vertices or corners of e_c. Around these collocation points, one then adds $N_{\text{overlap}} - 1$ layers of additional collocation points (in Fig. 4, $N_{\text{overlap}} = 2$). If the mesh has non-uniform h or p refinement, the resulting set of collocation points will have ragged boundaries.

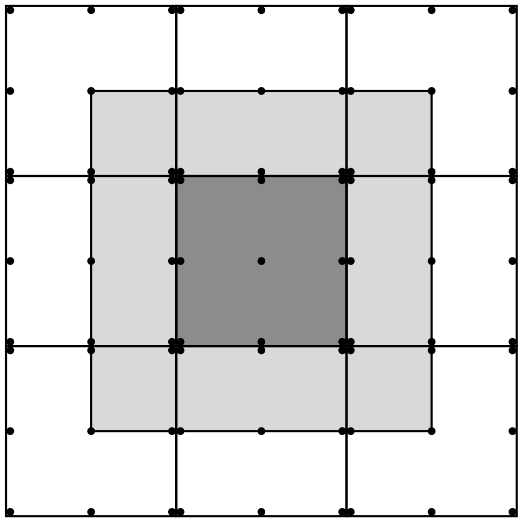
FIG. 4. A simple 2-D Schwarz subdomain, with no h-nonconforming or p-nonconforming boundaries. In grey is the element e which is the center of the subdomain. The light grey area is the overlap (of size δ_ξ) into the other elements. The subdomain is composed of everything in light and dark grey.

The solutions on the individual Schwarz subdomains are combined as a weighted sum. The weights differ with each collocation point of each subdomain. In 1-D, for a Schwarz subdomain centered on element e_c with left and right neighbours e_{c-1} and e_{c+1} we define an extended LGL coordinate ξ_{ext} for a collocation point x as follows:

$$\xi_{\text{ext}}(\boldsymbol{x}) = \begin{cases} \xi^* & \boldsymbol{x} \in e_c, \\ \xi^* \pm 2 & \boldsymbol{x} \in e_{c\pm1}, \end{cases} \tag{38}$$

where ξ^* is the LGL coordinate of the collocation point x in the reference coordinate system of the macro-element containing e_c. This definition takes care of the case when a Schwarz subdomain contains a face that is on a tree boundary.

We denote the overlap size as δ_ξ and compute it as the width of the Schwarz subdomain overlap in the coordinate ξ_{ext}. With these definitions we compute the weights for this 1-D subdomain using the function $w_h : \mathbb{R} \longrightarrow \mathbb{R}$ defined as

$$w_h(\xi_{\text{ext}}) = \frac{1}{2}\left(\phi\left(\frac{\xi_{\text{ext}} + 1}{\delta_\xi}\right) - \phi\left(\frac{\xi_{\text{ext}} - 1}{\delta_\xi}\right)\right), \tag{39}$$

where the ϕ function is given by

$$\phi(\xi) = \begin{cases} \text{sgn}(\xi), & |\xi| > 1, \\ \frac{1}{8}(15\xi - 10\xi^3 + 3\xi^5), & |\xi| \leq 1. \end{cases} \tag{40}$$

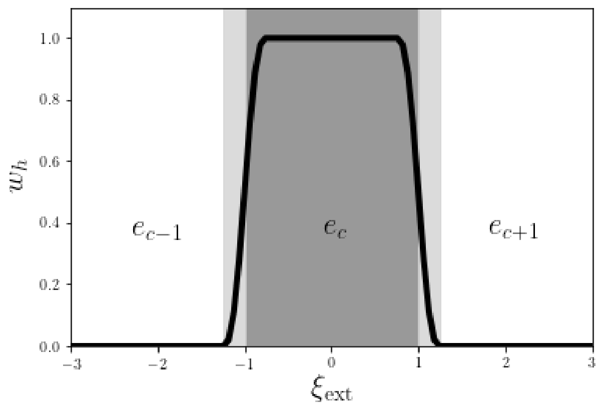

FIG. 5. A plot of the Schwarz weighting function defined by Eq. (39). The dark grey shaded region is the center element of the Schwarz subdomain e_c and the light grey shaded region is the overlap of width δ_ξ. For this plot, we set $\delta_\xi = .25$.

Algorithm 6 Schwarz Smoother
$N_{\rm iter,Sch}$ is a user-defined parameter.

```
1:  function SMOOTHER($x, b$, J)
2:      for $k = 1...N_{\rm iter,Sch}$ do
3:          $r \leftarrow b - {\rm J}x$
4:          for $s = 1...N_{\rm subs}$ do
5:              $r_s \leftarrow R_s r$
6:              Solve $R_s {\rm J} R_s^T \delta x_s = r_s$
7:          end for
8:          $x \leftarrow x + \sum_s R_s^T W_s \delta x_s$
9:      end for
10:     return $x$
11: end function
```

We have plotted the weighting function w_h in Figure 5.

For Schwarz subdomains in d-dimensions, the weights are computed as a product of 1-dimensional weights along each dimension,

$$W = \prod_{i=1}^{d} w_h(\xi^i_{ext}). \tag{41}$$

For the Schwarz subdomain solves we need to define the following operators: R_s is the restriction operator for a Schwarz subdomain, it reduces the data on the mesh to the nodes of the subdomain. R_s^T is the transpose of this operator. W_s are the weights for a Schwarz subdomain, computed by evaluating Eq. (39) on the nodes of the subdomain. With these definitions, the Schwarz smoother algorithm is listed in Alg. 6. In Alg. 6, N_s indicates the number of smoothing cycles (typically, $N_s = 3$), N_{subs} indicates the number of subdomains which for our implementation is equal to the number of elements. The linear system on line 6 of Alg. 6 is of the size of the Schwarz-subdomain; we solve it with conjugate gradients to a relative tolerance of 10^{-3}.

6. Multigrid Inter-mesh Operators

Let us now turn to the implementation of restriction and prolongation operators. For clarity, we use the following notation: a superscript lowercase h refers to the children elements and an uppercase H refers to the parent element.

We recall our requirement that the polynomial order of a coarse element is smaller than or equal to the polynomial orders of its children elements (cf. Sec. II D 3). This ensures that a parent element's Lagrange polynomial space is always embedded in the broken Lagrange polynomial space of its children. In particular, the coarse-mesh basis-functions can be written, exactly, as

$$\psi^H_\alpha = (I^H_h)_{\beta\alpha}\psi^h_\beta, \tag{42}$$

where $(I^H_h)_{\beta\alpha} = \psi^H_\alpha(\boldsymbol{\xi}^\beta_h)$ is the interpolation-matrix of the coarse-mesh basis-functions to the fine-mesh collocation points, and where we utilized our choice of a *nodal* basis.

For h-refinement, each ψ^h_β has support in only one of the child-elements and, consequently, the implicit sum over β in Eq. (42) goes over the basis-elements of all children elements.

The preceding paragraph immediately suggests the natural definition of the prolongation operator from a coarse-grid function $u^H = u^H_\alpha\psi^H_\alpha$ to a fine-grid function $u^h = u^h_\beta\psi^h_\beta$ such that $u^h \equiv u^H$. This yields

$$u^h_\beta = u^H(\boldsymbol{\xi}^\beta_h) = (I^H_h)_{\beta\alpha}u^H_\alpha, \tag{43}$$

so that (I^H_h) represents the prolongation operator.

Because the left hand side operator for the weak equations —e.g. $L(\cdot,\cdot)$ in Eqn. (13)— is a bilinear form, the residual will also be a bilinear form, which we will denote $B(\cdot,\cdot)$; on the coarse grid. Equation (42) therefore implies

$$B(\,\cdot\,,\psi^H_\alpha) = (I^H_h)_{\beta\alpha}B(\,\cdot\,,\psi^h_\beta), \tag{44}$$

which defines our coarse grid residual.

Lastly, let us consider restriction of the operators itself. After linearization around the current solution u_0, the problems we consider here take the form

$$\nabla^2\delta u + f(u_0)\delta u = 0, \tag{45}$$

for some function f. The associated weak form reads

$$L_h(\delta u,\psi_\alpha) + \int f(u_0)\psi_\alpha\delta u\,{\rm d}\boldsymbol{x} = 0 \qquad \forall\ \psi_\alpha. \tag{46}$$

The weak Laplacian operator $(L_h(\cdot,\cdot))$ can be computed on any coarse grid via Eqn. (13) without any need for a restriction operator. The second term will require either a restriction operator on $f(u_0)$, or a restriction operator on $O^h_{\alpha\beta} \equiv \int f(u_0)\psi^h_\beta\psi^h_\alpha{\rm d}\boldsymbol{x}$. We choose the latter. By Eq. (42), it holds that

$$\int f(u_0)\psi^H_\alpha\psi^H_\beta\,{\rm d}\boldsymbol{x} = (I^H_h)_{\gamma\alpha}(I^H_h)_{\delta\beta}\int f(u_0)\psi^h_\gamma\psi^h_\delta\,{\rm d}\boldsymbol{x}, \tag{47}$$

so that the restricted operator in matrix form becomes

$$O^H = (I^H_h)^T\ O^h\ (I^H_h). \tag{48}$$

E. hp-Adaptivity

One of the great advantages of the discontinuous Galerkin method is that it naturally allows for two different methods of refining the grid, h-adaptivity, where one subdivides the element into smaller sub-elements, and p-adaptivity, where one increases the order p_e of the polynomial basis on an element. The combination of both h-adaptivity and p-adaptivity is called hp-adaptivity. If one strategically p-refines in smooth regions of the mesh and h-refines in discontinuous regions of the mesh, then it is possible to regain exponential convergence in particular error norms for problems with non-smooth functions. In the next four sections we will discuss the four components of our hp-adaptive scheme. These are: (1) The expected convergence of the solution on a series of adaptively refined hp-meshes (2) an a posteriori error estimator to decide which elements will be refined; (3) a driving strategy that determines based on the convergence of the error estimator whether to h-refine or p-refine; (4) an efficient method to apply discrete operators in multi-dimensions on an hp-grid. We discuss these four items in order in the follow subsections.

1. Expected Convergence

For quasi-linear problems with piecewise-analytic solutions u on polygonal domains, the convergence of the energy norm (Eq. 33) of the analytical error for the numerical solution u_h is

$$||u - u_h||_* \leq C \left(\sum_{e \in \mathbb{E}} \frac{h_e^{2s_e-2}}{p_e^{2k_e-3}} ||u||^2_{H^{k_e}} \right)^{1/2}. \tag{49}$$

Here H^{k_e} is the Sobolev space on element e with Sobolev order k_e and $1 \leq s_e \leq \min(p_e + 1, k_e)$ [52–55]. For uniform refinement and uniform Sobolev order k across elements, we expect $||u-u_h||_* \leq Ch^{\min(p+1,k)-1}$. Thus, for smooth problems we expect $||u - u_h||_* \leq Ch^p$.

One can show [56] that there exists a series of hp-adaptive refinement steps where the convergence of the energy norm is asymptotically bounded by

$$||u - u_h||_* \leq C_1 \exp(-C_2 \mathrm{DOF}^{1/(2d-1)}). \tag{50}$$

Here C_1 and C_2 are constants, d is the dimension of the mesh and DOF is the number of degrees of freedom, in other words the number of grid points on the mesh.

2. A Posteriori Error Estimator

Ref. [55] derives a local a posteriori error estimator for an interior penalty hp-DG discretization of the following class of strongly nonlinear elliptic problems with Dirichlet boundary conditions

$$-\nabla \cdot \boldsymbol{a}(\boldsymbol{x}, u, \nabla u) + f(\boldsymbol{x}, \nabla u) = 0. \tag{51}$$

The error estimator can be computed locally on each element by the following prescription. Given a discretized solution u_h, first define quantities on each element e or mortar element m by

$$\begin{aligned}
R_e &\equiv f(u_h, \nabla u_h) - \nabla \cdot \boldsymbol{a}(u_h, \nabla u_h), && e \in \mathbb{E}, && \text{(52a)}\\
J_{m,1} &\equiv [\![\boldsymbol{a}(u_h, \nabla u_h)]\!]_m, && m \in \mathbb{M}_{\mathbb{I}}, && \text{(52b)}\\
J_{m,2} &\equiv [\![u_h]\!]_m, && m \in \mathbb{M}_{\mathbb{I}}, && \text{(52c)}\\
J_{m,3} &\equiv [\![u_h - g_D]\!]_m, && m \in \mathbb{M}_{\mathrm{D}}. && \text{(52d)}
\end{aligned}$$

From these quantities we compute the followings integrals on an element e and mortar m:

$$\begin{aligned}
\eta_e^2 &= h_e^2 p_e^2 ||R_e||^2_{0,e}, && \text{(53a)}\\
\eta_{m,1}^2 &= h_m p_m^{-1} ||J_{m,1}||^2_{0,m}, && \text{(53b)}\\
\eta_{m,2}^2 &= \sigma h_m^{-1} p_m^2 ||J_{m,2}||^2_{0,m}, && \text{(53c)}\\
\eta_{m,3}^2 &= \sigma h_m^{-1} p_m^2 ||J_{m,3}||^2_{0,m}, && \text{(53d)}
\end{aligned}$$

Here, $||.||^2_{0,e} = \int_e (.)^2 \mathrm{d}\boldsymbol{x} = \int_{[-1,1]^d} J(.)^2 \mathrm{d}\boldsymbol{\xi}$ and $||.||^2_{0,m} = \int_m (.)^2 \mathrm{d}\boldsymbol{s} = \int_{[-1,1]^{d-1}} S^m(.)^2 \mathrm{d}\boldsymbol{\xi}$. Furthermore, h_e is the diameter of the element e, p_m, h_m and σ are defined in Eq. (32) and J, S^m are defined in Eq. (22) and Eq. (23).

Following [55], we take the local estimator on the element to be

$$\eta^2(e) = \eta_e^2 + \sum_{m \subset \mathbb{M}_e} \eta_{m,1}^2 + \sum_{m \subset \mathbb{M}_e \setminus \mathbb{M}_D} \eta_{m,2}^2 + \sum_{m \subset \mathbb{M}_e \cap \mathbb{M}_D} \eta_{m,3}^2, \tag{54}$$

where $\mathbb{M}_e$ and $\mathbb{M}_d$ are the sets of mortar elements touching the element e and $\partial\Omega_d$ respectively (see Sec.II A 1). The estimator provides an estimate of the error in the energy norm, $||u - u_h||_*$, see Eq. (33). Similar error estimators for various classes of linear and non-linear elliptic PDEs can be found in [52, 57–62]. Equations (53) are typically computed on the physical grid on which the elliptic PDE is solved. For highly stretched grids (for instance, when a mapping inverse in radius is used to push the outer boundary to very large radius $R \sim 10^3 \cdots 10^{10}$), the geometric factors in Eqs. (53) can distort the error estimates. For those cases, we will sometimes introduce a fiducial grid of identical structure and connectivity, that avoids or mitigates excessive grid-stretching. Once a fiducial grid is chosen, its geometric properties can be used in Eqs. (53) and the corresponding norms. Below, we demonstrate that such a “fiducial-grid” error-estimator allows efficient hp-refinement even in highly stretched computational domains. This problem is encountered below in Sec. III C and discussed there in greater detail.

3. Driving Strategy

The a posteriori estimator Eq. (54) indicates which elements to refine. Then the choice must be made for each cell, whether to h-refine or p-refine elements with a large error $\eta(e)$. In the survey paper [63], Mitchell and McClain looked at 15 different hp-adaptive strategies with a finite elements scheme. Their

Algorithm 7 hp-AMR Driving Strategy
γ_h and γ_p are user-defined parameters.

```
1:  procedure SMOOTH-PRED
2:      if $\eta^2(e)$ is large then
3:          if $\eta^2(e) > \eta^2_{\text{pred}}(e)$ then
4:              h-refine element
5:              $\eta^2_{\text{pred}}(e_{\text{children}}) \leftarrow \gamma_h \left(\frac{1}{2}\right)^d \left(\frac{1}{2}\right)^{2p_e} \eta^2(e)$
6:          else
7:              p-refine element
8:              $\eta^2_{\text{pred}}(e) \leftarrow \gamma_{p_e} \eta^2(e)$
9:          end if
10:     end if
11: end procedure
```

results indicate that the strategy known as smooth-pred, first introduced in [64] performs quite well, if not the best under the example problems they tested. For this reason alone we use it, but our code is flexible enough to use any of strategies studied in Mitchell and McClain's paper.

The idea behind the smooth-pred strategy is based on the observation that for a locally smooth solution, the energy norm Eq. (33) $\eta(e)$ will converge as h^p, (see Sec. II E 1). To take advantage of this observation, we first predict the error in the next refinement step and test if this prediction satisfies the smooth error convergence law, if so we p-refine, otherwise we h-refine. The full algorithm is given in Algorithm 7.

In Algorithm 7, $\eta^2(e)$ is the square of the error estimator, Eq. (54) of the element under consideration, and $\eta^2_{\text{pred}}(e)$ is the predicted error estimator from the last AMR step assuming the solution on the element is smooth. We always start with $\eta_{\text{pred}}(e) = 0$, so that each element will first be p-refined, before h-refinement is considered.

The parameters γ_p and γ_h influence the behavior of Alg. 7 as follows: As long as the actual $\eta^2(e)$ is *small* compared to the predicted $\eta^2_{\text{pred}}(e)$, Line 3 implies continued p-refinement. Since Line 8 reduces the predicted η^2_{pred} by a factor γ_p, this means, that p-refinement continues as long as each increment in p_e reduces the error-estimator $\eta^2(e)$ by a factor γ_p, i.e. as long as exponential convergence is obtained with convergence rate better than $\sqrt{\gamma_p}$. If this convergence-rate is not observed, the driver switches to h-refinement, and γ_h begins to matter. In this case, a large γ_h will preferably switch back to p-refinement, whereas a small γ_h will prefer continued h-refinement. All runs shown in the remainder of this paper use $\gamma_p = 0.1$. The runs differ in γ_h, which is used to tune h- vs p-refinement, and in how many elements are refined in each AMR-iteration, which is used to control how quickly AMR increases the number of degrees of freedom. However for the majority of the runs we find that $\gamma_h = 0.25$ is a good choice.

In the numerical examples of this paper we use the following two alternatives for the conditional on line 2 of Alg. 7, i.e. to decide which elements should be refined: In the first test-problem, we refine if η^2 is greater than some constant factor times the mean $\bar{\eta}^2$ across all elements e. This criterion was used in the original description of the hp-AMR scheme (See [64]). In the remaining problems we refine a percentage of elements with the largest $\eta(e)$. We change criterion because the estimator may vary over orders of magnitude and this variation may change at each refinement step. In such cases, the percentage criterion robustly captures more of the elements with a large estimator than thresholding on a constant times the average estimator). However, the constant factor times the mean method has the advantage that it is computationally cheap and does not require a global sort of the estimator over all cores like the percentage method does. When we use the percentage indicator, we make use of the highly parallel global sort outlined in [65].

4. *On-the-fly hp Tensor-Product Operations*

To solve the elliptic equations with multigrid on grids that are hp-adaptively refined, a large number of different operators must be generated at run-time. In no specific order, we need at least the following linear operators

- interpolation operators on faces, edges and volumes of size $(p' + 1) \times (p + 1)$ in 1-D, where p is the order of the original data and p' is the order of the new interpolated data
- restriction operators on faces, edges and volumes of size $(p + 1) \times (p' + 1)$ in 1-D, where p' is the order of the original data and p is the order of the new restricted data
- derivative operator of order p on the reference element, which has size $(p + 1) \times (p + 1)$ in 1-D.

We can take advantage of the tensor product nature of the reference element in order to efficiently generate these operators on demand. All of the operators listed can be reduced to a tensor product of 1-D operators when applied in $d \geq 2$ dimensions. Upon requiring a certain operator in a calculation, the process is then as follows

1. Check if the 1-D version of the operator has already been computed in the database, if it has been computed, retrieve and goto step 3, if not goto step 2.
2. Compute the 1-D version of the operator and store in the MPI process-local database.
3. Apply the 2-D or 3-D version of the operator on a vector by using optimized Kronecker product matrix vector operation on the nodal vector v: $(A_{1D} \otimes B_{1D})\vec{v}$ for $d = 2$, or $(A_{1D} \otimes B_{1D} \otimes C_{1D})\vec{v}$ for $d = 3$. Here A, B, C denote the 1-D matrices of the respective operator, as applied to the first, second, and third dimension.

A Kronecker product matrix vector operation can be efficiently performed using a series of BLAS matrix-multiply calls, see [66] for example.

F. Implementation

To implement the multi-block adaptive mesh refinement we use the p4est library, which has been shown to scale to $O(100{,}000)$ cores [67]. We use PETSc [68] for the Krylov subspace linear solves and the Newton Raphson iterations. The components of the multigrid algorithm are written by the authors and do not use PETSc.

III. TEST EXAMPLES

We examine the components of our code through three test examples, the first a linear Poisson problem solved on a square grid where the solution is only C^2-smooth at the $(0,0)$ grid point. The second example is a non-linear elliptic problem, where we solve the Einstein constraint equations for the initial data of a constant density star. We then end the test examples section with a linear problem on a cubed-sphere with stretched outer boundary and a solution that falls off as r^{-1} with radial coordinate r as $r \longrightarrow \infty$. Each test is aimed at isolating different aspects of the puncture black-hole problem, whose solution contains points that are C^2-smooth and falls off as r^{-1}. The puncture black-hole problem further requires us to solve non-linear Einstein constraint elliptic equations on a cubed-sphere mesh.

A. Poisson with $H^{4-\epsilon}$ solution

The first test problem we will investigate numerically is taken from [69], where the authors solve $\nabla^2 u = f$ on $\Omega = (0,1)^2$ with the solution chosen as

$$u = x(1-x)\,y(1-y)\left[\left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2\right]^{3/2}. \quad (55)$$

Here $u \in H^{4-\epsilon}$, where $\epsilon > 0$ so from Eq. (49) we expect third order convergence for uniform refinement. We have confirmed this numerically. However, with hp-adaptivity, it is possible to achieve $||u-u_h||_* \sim \exp(\text{DOF}^{\frac{1}{3}})$, as we show in Fig. 6. Notice also the close agreement between the estimator and energy norm $||u-u_h||_*$. The final mesh is shown in Fig. 7. Of note is the fact that the elements with the lowest polynomial order (p=4) are only in the vicinity of the C^2-smooth $(0,0)$ grid-point and the refinement level of the mesh is locally much higher here as well. Here we used AMR parameters $\gamma_h = 10$, $\gamma_p = 0.1$; at each AMR iterations all cells were refined which have $\eta^2(e)$ larger than $1/4$ of the overall mean of all $\eta^2(e)$. For this problem, we performed a survey of different choices for γ_h and γ_p. We found rather modest dependence on γ_h within $0.25 \lesssim \gamma_h \lesssim 10$, with $\gamma_h = 10$ leading to the good balance between $h-$ and $p-$refinement shown in Fig. 7. Our preferred value for the rest of the paper ($\gamma_h = 0.25$) also yields good convergence, but leads to a more uniform polynomial degree through the mesh.

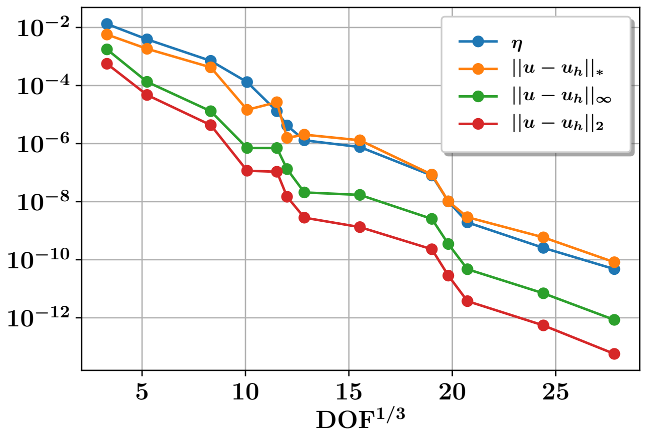

FIG. 6. Problem A, Eq. (55): Convergence of the energy norm estimator η^2 and the error between the numerical solution u_h and the analytic solution u in the energy norm $||\cdot||_*$ (Eq. 33) and the L_2 norm $||\cdot||_2$ (Eq. 34).

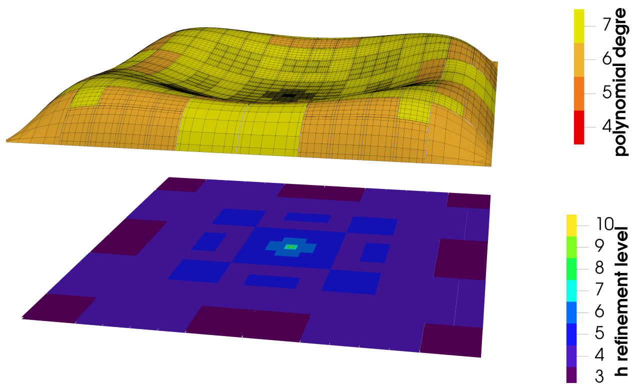

FIG. 7. Problem A, Eq. (55): Visualization of the solution and the hp-refined computational mesh. **Top portion:** The computational grid, color-coded by polynomial degree, with height representing the solution u. **Bottom portion:** Color-coded by h-refinement level. A cell on level l has size 2^{-l} of the overall computational domain.

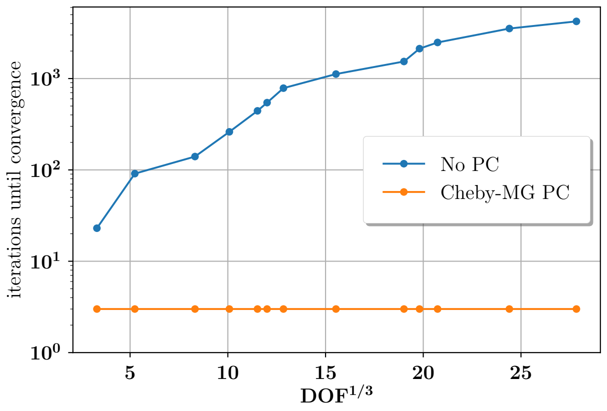

FIG. 8. Problem A, Eq. (55): Iteration-count of the flexible conjugate gradient (FCG) Krylov subspace solver versus the Multigrid preconditioned FCG solver (MG-FCG). The MG-FCG completes all solves in ≤ 3 iterations.

B. Constant density star

For the next test problem, taken from [70], we solve the Einstein constraints in the simplest possible scenario, a constant density star. The goal of this test problem is to investigate how the elliptic solver behaves for problems that contain surface discontinuities that mimick surface and phase transition discontinuities in a Neutron star. The Einstein constraint equations for the case of a constant density star reduce to

$$\nabla^2\psi + 2\pi\rho\psi^5 = 0, \tag{56}$$

where ρ is the density of the star and ψ is the conformal factor which describes the deviation of the space from flat space. In [70] the authors solve the above equation for the case of a star with radius r_0 and mass-density

$$\rho = \begin{cases} \rho_0 & r \le r_0 \\ 0 & r > r_0, \end{cases} \tag{57}$$

where r is the radial spherical polar coordinate. Since the star is in isolation, the boundary condition at infinity is $\psi = 1$, corresponding to a asymptotically-flat space. For such a problem, there is an analytic solution given by

$$\psi = \begin{cases} Cu_\alpha(r) & r \le r_0 \\ \frac{\beta}{r} + 1 & r > r_0. \end{cases} \tag{58}$$

with $C = (2\pi\rho_0/3)^{-1/4}$ and

$$u_\alpha(r) \equiv \frac{(\alpha r_0)^{1/2}}{(r^2 + (\alpha r_0)^2)^{1/2}}. \tag{59}$$

The parameters α and β are determined from the continuity of ψ and it's first derivative at the surface of the star, and are given by

$$\beta = r_0(Cu_\alpha(r_0) - 1) \tag{60}$$

$$\rho_0 r_0^2 = \frac{3}{2\pi}\frac{\alpha^{10}}{(1+\alpha^2)^3} \tag{61}$$

We solve the above problem on a cubic domain with $\rho_0 = 0.001$ and analytic Dirichlet boundary conditions on a boundary at $8r_0$. The non-linear term in Eqn. 56 is handled by first interpolating ψ onto the GL quadrature points of an element, evaluating ψ^5 at the GL quadrature points and then performing the necessary Gaussian quadrature sum on each element. Figure 9 showcases the convergence of the solution and the nice agreement between the energy norm estimator η^2 and the energy norm of the analytic error. To achieve this convergence we used the following AMR parameters, $\gamma_h = 0.25$, $\gamma_p = 0.1$. At each AMR refinement iteration, 10% of the elements are refined. In Figure 10 we show a comparison between multigrid-preconditioned FCG iterations and unpreconditioned FCG iterations. We notice that for the preconditioned case, the iteration count is roughly constant with increases in DOF, whereas the unpreconditioned iterations grow with DOF. Finally, in Figure 11 we show the mesh after the last AMR step. This mesh showcases the highest h-refinement around the star boundary (yellow circle) as expected, since this area is not smooth.

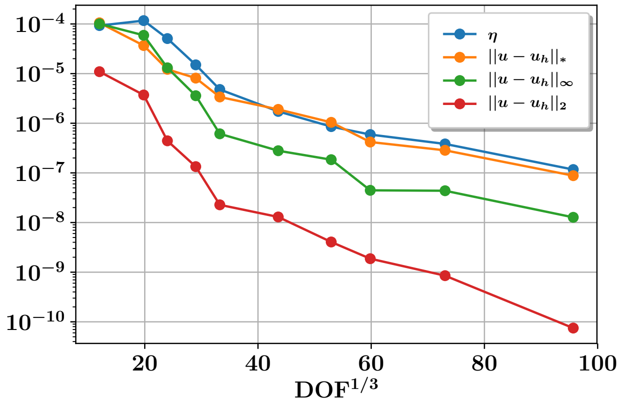

FIG. 9. Problem B, Eq. (56): Convergence of the energy norm estimator η^2 and the error between the numerical solution u_h and the analytic solution u in the energy norm $|| \cdot ||_*$ and the $|| \cdot ||_2$ norm

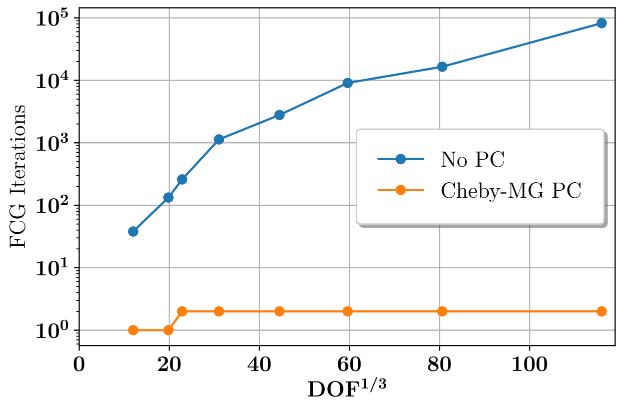

FIG. 10. Comparison of the average number of iterations per Newton-Raphson step when using a Chebyshev smoothed Multigrid preconditioner and no preconditioner.

C. Cubed sphere Meshes and Stretched Boundary Elements

So far, we have only investigated meshes with a regular, Cartesian structure and with a rectangular outer boundary at close distance. We will now investigate scenarios with a spherical outer boundary, where we use a macro-mesh arising from the 3-dimensional generalization of Fig. 1 (a). We will also place the outer boundary at very large radius, typically 10^9, to approximate boundary conditions at infinity. Typically, such problems have solutions which fall off as a power series in $1/r$.

Figure 12 shows the structure of the mesh we will use: a central cube, surrounded by *two* layers of six deformed macro-elements each. The inner layer interpolates from the cubical inner region to a spherical outer region. The outer layer has

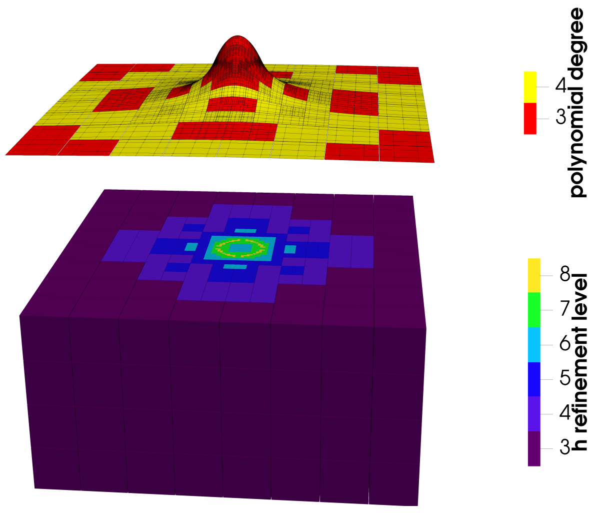

FIG. 11. Visualization of hp-refined computational mesh for Problem B, Eq. (56). **Top portion:** The $z = 0$ cross-section of the computational grid, color-coded by the polynomial degree and with height representing the solution ψ. **Bottom portion:** Volume rendering of the $z \le 0$ part of the computational domain, color-coded by the h-refinement level. A cell on level l has size 2^{-l} of the overall computational domain.

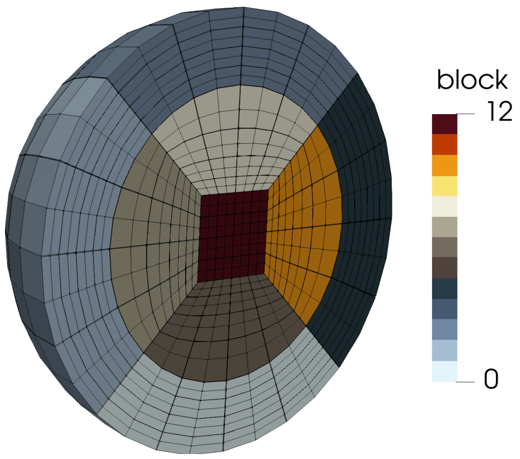

FIG. 12. The mesh structure for a computational domain with spherical outer boundary. This structure consists of 13 macro-meshes (shown in different colors). For clarity, only the $z \le 0$ part of the mesh is shown.

spherical boundaries both at its inner and outer surface, and thus radial coordinate lines that are always orthogonal to the angular coordinate lines. This allows to apply a *radial coordinate transformation* in the outer layer, to move its outer boundary to near infinity. Because we know the solution will fall off as $1/r$ we use an inverse mapping in the outer six spherical wedges of the cubed-sphere (blocks 0-5 in Fig. 12). This mapping is defined as follows. Denote the physical grid variable with $r \in [r_1, r_2]$ and the collocation-point integration variable as $x \in [x_1, x_2]$, then the inverse mapping is defined by

$$r = \frac{m}{x - t}, \tag{62}$$

where

$$m = \frac{x_2 - x_1}{\frac{1}{r_2} - \frac{1}{r1}}, \qquad t = \frac{x_1 r_1 - x_2 r_2}{r_1 - r_2}. \tag{63}$$

We will investigate the discontinuous Galerkin method on the following test problem which captures many of the above features:

$$\nabla^2 u = 3(1 + x^2 + y^2 + z^2)^{-\frac{5}{2}}, \tag{64}$$

with $\Omega = \{(x, y, z) : x^2 + y^2 + z^2 < R\}$ and Dirichlet boundary conditions given by the analytical solution, which is a Lorentzian function $u = (1 + r^2)^{-1/2}$. The Lorentzian function falls off as a power series in $1/r$ as $r \longrightarrow \infty$.

To solve this test problem we run two schemes: uniform p-refinement and adaptive p-refinement. We run only with p-AMR because the underlying solution is smooth everywhere so there is very little to no benefit in running with hp-AMR for this problem (which we also found to be the case empirically). For the uniformly refined run, we start with refinement level $l = 4$, i.e. with $2^{3l} = 4096$ elements in each of the 13 macro-elements, and increase p from 2 to 11.

To achieve pure p-AMR with the hp-AMR scheme outlined in Sec. II E, we start with refinement level $l = 2$, i.e. with $2^{3l} = 64$ elements in each of the 13 macro-elements, and with polynomial order $p_e = 1$ in all elements. We set $\gamma_h = 10^6$, $\gamma_p = 10^6$ and in each AMR iteration, we refine the 25% elements with the largest estimator. Figure 13 shows the convergence of the p-AMR scheme versus the p-uniform scheme. We stop at the maximum polynomial order currently allowed in the code, $p = 19$. Comparing the L_∞ norm between the p-AMR and p-uniform runs, the number of degrees of freedom per dimension, DOF$^{1/3}$, is roughly cut in half. The L_2 norm is slower to converge because it is dominated by the contributions of the outer stretched wedges, and this region is less aggressively refined by the AMR. Figure 14 shows the final mesh for the problem. The AMR algorithm increases the polynomial order p predominantly in the centre, where the solution has the most structure.

Strong coordinate stretching, as performed via the inverse map Eq. (62) leads to the following problem: The error estimator $\eta^2(e)$ utilizes integrals in physical coordinates in Eqs. (52). For strongly stretched grids (e.g. with inverse mappings where R is orders of magnitude larger than other length scales in the problem) these volume integrals will place a large emphasis on the regions at large distance. AMR will then aggressively refine in the stretched region despite the pointwise errors (as measured by the L_∞ norm for individual elements) being very small. Such regions tend to have a very low L_∞ because the numerical solution is very accurate there, but a high L_2 because of the large volume in the stretched region, thus also explaining why the L_2 norm of the p-AMR run in Fig. 13 fails to converge as smoothly as the L_∞ norm.

We solve this problem by introducing a "compactified grid", which has the same structure as the physical grid, but without the compactification in the physical grid. We then compute the integrals for the estimator in Eqs. (53) on this compactified grid. The integrands, Eqs. (52), are as before computed on

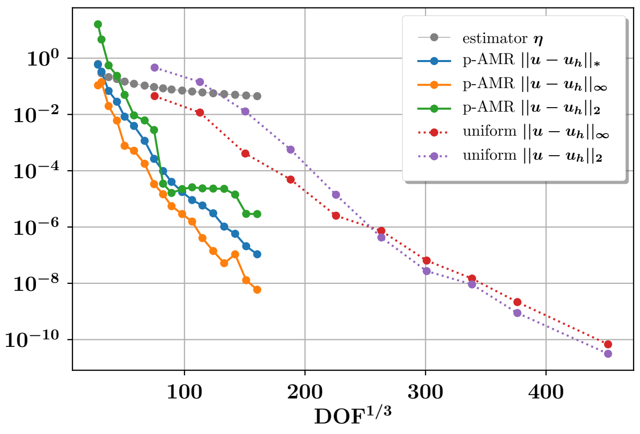

FIG. 13. Problem C, Eq. (64): Convergence of the solution as the degree p is uniformly increased across all elements for uniform case and as the degree p is adaptively increased in the amr case. (Dirichlet BC at R=1000).

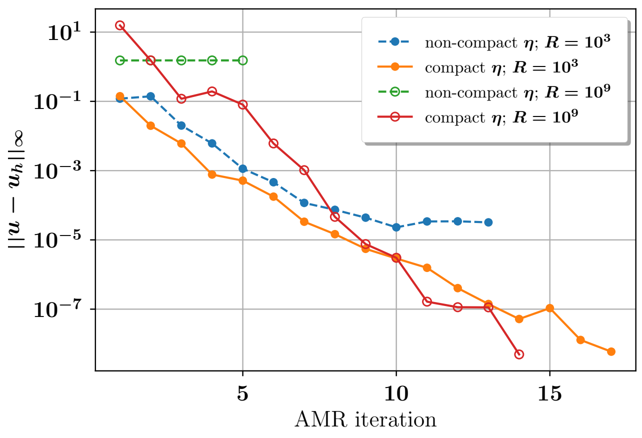

FIG. 15. Problem C, (Eq. 64): Convergence of the solution in the L_2 norm (See Eq. 34) as the degree p is adaptively refined with a compact and non-compact estimator. (Dirichlet BC at R=1000 and $R = 10^9$). All runs use the AMR/dG parameters of the p-amr run in Figure 13.

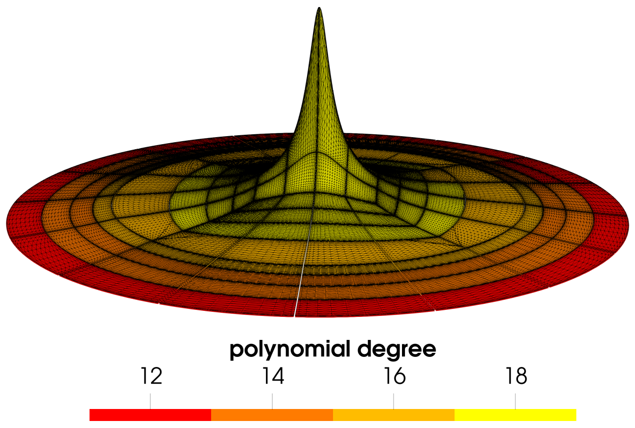

FIG. 14. Visualization of the final mesh for problem C. Shown is a xy-plane slice of the mesh which has been warped so that the height of the surface corresponds to the value u of the solution, color coded by the polynomial degree. For ease of visualization, grid-points are mapped onto the compactified grid on which the estimator η_e is computed; the compactified outer radius $R = 3$ corresponds to the physical outer radius $R = 1000$.

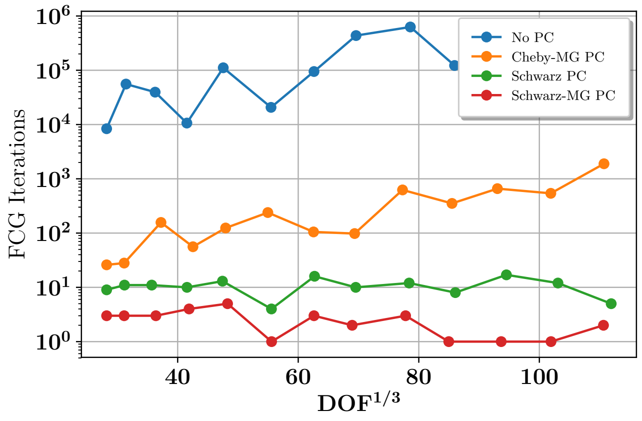

FIG. 16. Comparison of four different methods of preconditioning FCG. All runs use the AMR/dG parameters of the p-amr run in Figure 13.

the physical grid. In essence, this procedure merely changes the weighting of the different regions of the grid via the Jacobians J in the integrals and the parameters h_e, h_m which are now computed on the compactified grid. In practice, we use as compactified grid a cubed-sphere mesh where the outer spherical shell extends from radius 2 to radius 3, and where the middle cube has a side-length of $2/\sqrt{3}$.

Figure 15 shows the convergence of the L_∞ norm using the compactified grid versus the non-compactified grid. For $R = 1000$, the standard estimator ("non-compact η") converges well for the first 10 AMR iterations, but then stalls. For $R = 10^9$, the standard estimator fails to yield convergence of the solution at all. For both cases, the new estimator ("compact η") results in good convergence. We stress that the compactified estimator is only used in driving the AMR refinement. The DG-scheme is always formulated in the physical domain, and also the error plotted in Fig. 15 is computed from the solution on the physical domain with outer boundary $R = 1000$ or 10^9. The p-AMR run shown in Fig. 13 uses the compactified estimator.

Finally, we investigate the efficiency of preconditioners for the p-AMR run with $R = 1000$ and the compactified estimator, i.e. the run plotted in orange in Figs. 13 and 15. Figure 16 presents the iteration counts for four different kinds of preconditioning. Chebyshev-smoothed multigrid preconditioner loses efficiency on cubed spheres with stretched boundaries, possibly due to a poorly estimated upper eigenvalue in Alg. 5. However, using the more powerful domain decomposition additive Schwarz method, we can regain the efficiency of the multigrid-preconditioner seen in the previous two sections. Here, the Schwarz subdomains have $N_{\text{overlap}} = 2$, and each multi-grid iteration employs $N_{\text{iter,Sch}} = 3$ iterations of Schwarz-smoothing with rtol=1e-3. Chebyshev uses $N_{\text{iter,eigs}} = 15$ iterations for the eigenvalue estimate, and $N_{\text{iter,Cheb}} = 15$ iterations in the Chebyshev smoother.

D. Puncture Initial Data

There are various approaches to solving for binary black hole initial data sets and these approaches are primarily distinguished by the initial choice of hypersurface and how the physical singularity inside the black holes is treated. One possibility when considering two black holes is to work on $\mathbb{R}^3$ with two balls excised[71–73]. This approach has been shown to work well with the spectral finite element method (e.g. [74]), but is more problematic for finite-difference codes because special stencils must be created near the curved boundaries. Another popular approach, which is more amenable to finite difference codes is the puncture method [75], where an elliptic equation is solved on $\mathbb{R}^3$, with two points where the solution becomes singular. These points represent the inner asymptotically flat infinity (Brill-Lindquist topology). The puncture method simplifies the numerical method because no special inner boundary condition has to be considered, however the solution is only C^4 smooth at the puncture points [75]. Without using contrived coordinate systems to remove the C^4-smooth nature of the punctures (See [76] for example), spectral methods cannot perform optimally, because they would require the solution to be smooth on the computational domain in order to obtain exponential convergence. We choose to solve for binary black hole initial-data with the puncture method in this paper for two reasons. The first is that the method does not couple well with traditional spectral schemes as discussed above and this allows us to compare the discontinuous Galerkin method to the spectral method. Secondly, the equation we must solve is less complicated than the excision case because it only involves a solve for a single field, the conformal factor, as opposed to the 5 fields one must solve for with the excision method (see e.g. [74]), so it is easier to implement numerically.

Nevertheless, puncture data provides a testing ground for many of the new techniques developed here: Singular points which benefit from h-refinement; smooth regions that benefit from p-refinement; a spherical outer boundary requiring the cubed-sphere domain shown in Fig. 12; and a boundary at infinity (or near infinity) which requires a compactified radial coordinate. Moreover, for testing of adaptivity, it is easy to add arbitrarily many black holes each represented by its own singularity, at arbitrary coordinates with arbitrary spins.

For the case of puncture data, the initial data equations of general relativity reduce down to a single equation [75]:

$$-\nabla^2 u = \frac{1}{8}\bar{A}^{ij}\bar{A}_{ij}\psi^{-7} \tag{65}$$

where $\bar{A}_{ij}$ is a spatially dependent function given by

$$\bar{A}_{ij} = \frac{3}{2}\sum_I \frac{1}{r_I^2}[2P^I_{(i}n^l_{j)} - (f_{ij} - n^l_i n^l_j)P^k_I n^k_I + \frac{4}{r_I}n^l_{(i}\epsilon_{j)kl}S^k_I n^l_I]. \tag{66}$$

Here, n^i_I are the spatially varying components of the radial unit-vector $\hat{n}_I(\vec{x}) = (\vec{x} - \vec{c}_I)/|\vec{x} - \vec{c}_I|$ relative to the position $\vec{c}_I$ of the I-th black hole [75]. The constant vectors $\boldsymbol{P}_I$ and $\boldsymbol{S}_I$ quantify the momentum and spin of the I-th black-hole and $\psi = 1 + \sum_I \frac{m_I}{2r_I} + u$. The boundary condition is

$$u \longrightarrow 0, \quad |\vec{x}| \longrightarrow \infty. \tag{67}$$

We solve the above elliptic PDE using first the spectral code SpEC and then the dG solver presented in this paper. We solve for the case of two orbiting equal mass black holes with momenta $\pm .2$, zero spin and initial positions $(\pm 3, 0, 0)$ in units of total mass M. This is the test case used in [77]. Since there is no analytical solution, we will compare the solutions between refinement levels at four reference points on the x-axis. These are $(0, 0, 0)$, $(3, 0, 0)$, the location of the right-most puncture, $(10, 0, 0)$ and $(100, 0, 0)$.

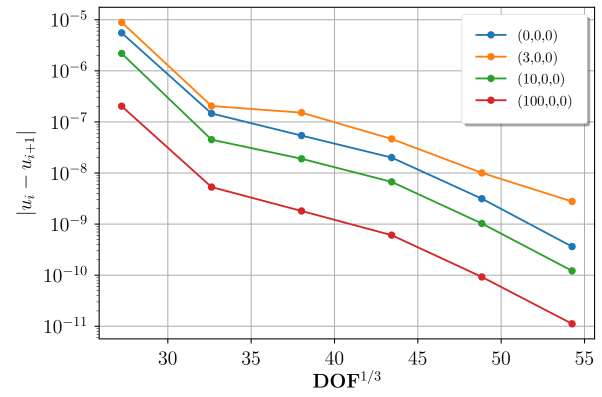

FIG. 17. Convergence of the SpEC–elliptic solver with manually adopting the domain-decomposition and manual adjustment of resolution to compensate for the singularities. Plotted are differences to the next-*lower* resolution at four points in the computational domain.

The SpEC solver was already used to solve for puncture data in [78, 79]. This spectral code is apriori not well suited for puncture data which results in a non-smooth solution $u(\vec{x})$. Because we know where the singularities of the punctures are, this can be overcome manually, by covering the punctures with very small rectangular blocks, at high enough resolution, to compensate for the loss of exponential convergence. Figure 17 shows the difference in the solution at the four reference points as the resolution is manually increased. We emphasize that this solution obtained with SpEC depents on (i) *prior knowledge* of the locations of the singularities; and (ii) tuning of SpEC's mesh and resolution *by hand*.

For the dG solver, we start with a uniformly refined cubed-sphere mesh at level $l = 1$, i.e. 13 blocks, each consisting of eight cells. The outer radius at 10^{11}M and the size of the inner cube is 10M. We start further with elements of polynomial order $p = 2$. The location of the punctures is *not* utilized in the dG code, and all mesh-refinement is automatic, driven by Alg. 7 with parameters $\gamma_h = .25$ and $\gamma_p = .1$ and we refine the top 12.5% of elements. In order to run a Schwarz smoother for this problem we would need to transfer ghost-data for the operator described by Eqn. (48) whenever a Schwarz subdomain contains a ghost element. While this is by no means problematic, we have not yet implemented the infrastructure to do it, so we just use a Chebyshev smoother when we precondition this

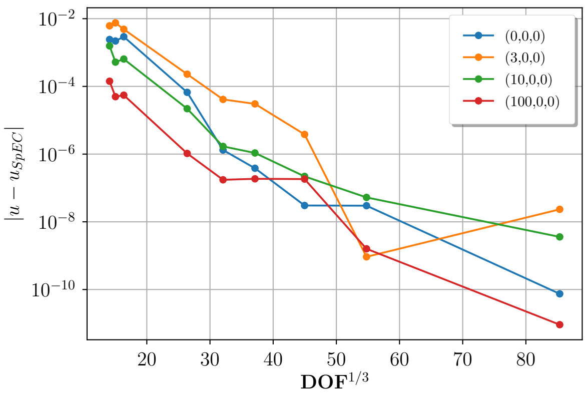

FIG. 18. Problem D: Black hole initial data with two punctures. Convergence of the error between the dG solution and the SpEC solution.

	Puncture 1	Puncture 2	Puncture 3
m	0.2691	0.4063	0.3245
x	0.0152	-2.316	-1.0279
y	-0.6933	1.8274	-2.2711
P_x	0.0585	-0.0284	0.1640
P_y	0.0082	-0.1497	0.0515
S_z	-0.0134	-0.0332	-0.0708

TABLE I. The randomly generated parameters for the three black holes. We list the mass m, the position $(x, y, 0)$, the momentum $(P_x, P_y, 0)$ and the spin $(0, 0, S_z)$ of the black holes.

problem with Multigrid. Figure 18 shows the convergence of the four reference points with respect to the finest grid SpEC solution between AMR levels. We first see that in terms of overall DOF, the dG solver doesn't do much worse than the finely tuned SpEC solver, even though the dG solver has to adaptively find the punctures, has a larger initial error and has no h-or-p coarsening, so mistakes in the refinement cannot be fixed. Thus, taken all of this into account, the convergence is highly satisfactory. The bounce in the $(3, 0, 0)$ at the second to last iteration arises because the dG-solution oscillates around the SpEC solution and coincidentally is shown near a zero-crossing. Figure 20 shows the solution on the final mesh, which has the highest h-refinement exactly at the points of the punctures, as desired.

Next, we solve for the puncture initial data of three black-holes randomly located in the xy-plane, with random spins and random momenta. Spectral solvers such as SpEC cannot perform well when the singular points on the grid are not known in advance. Thus, we end this paper showcasing a problem that our discontinuous Galerkin code can solve, but SpEC cannot. Table III D illustrates the parameters for the randomly placed punctures and their spin and momenta.

Figure 21 shows the convergence of four points, three at the location of the punctures, and one at $(100, 0, 0)$. We use amr parameters $\gamma_h = 0.25$, $\gamma_p = 0.1$, $F_{\text{refined}} = 0.125$ for this run.

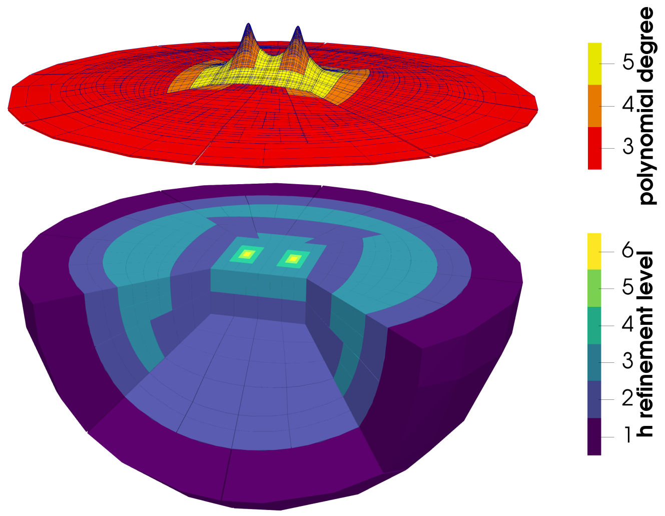

FIG. 19. Problem D (Black hole initial data with two punctures): Visualization of the hp-refined computational mesh. The bottom portion of the image shows a volume rendering of the $z < 0$ portion of the computational domain, with two blocks removed, and color-coded by the h-refinement level. The top portion of the image shows the $z = 0$ cross-section of the computational grid, color-coded by the polynomial degree, with the height representing the solution u. For ease of visualization, grid-points are mapped onto the compactified grid on which the estimator η_e is computed; the compactified outer radius $R = 3$ corresponds to the physical outer radius $R = 10^{11}$.

IV. CONCLUSION AND FUTURE WORK

We presented a new code for solving elliptic equations intended for numerical relativity. The methodology we use differs from other codes in the field in many important respects. In particular, we use a discontinuous Galerkin method to discretize the equations, an hp-adaptive mesh refinement scheme driven by an a posteriori estimator and a matrix-free, scalable Multigrid preconditioned Newton-Krylov solver. Individually, many of the features of our code have been implemented before [2, 25, 28, 29, 45], but they have never been combined together to create a general dG solver. In particular, the combination of curved meshes (cf. Fig. 12) and non-conforming elements is novel and is crucial for generic solution-driven AMR. Moreover, the compactified AMR-driver introduced in Sec. III C is also new, enabling AMR on compactified computational domains with outer boundary near infinity. Lastly, the use of a multigrid-preconditioned solver with a Schwarz (or Chebyshev) smoother on non-polygonal meshes has not been investigated in a dG setting until this paper.

For BBH puncture data, our new code approaches the accuracy of existing, specialized codes like SpEC. In addition, the automatic AMR in the new code does not require manual tuning of the computational mesh, and does not utilize any prior knowledge of features of the solution like the location of black hole punctures. The new code already improves on the more specialized codes by being able to handle an arbitrary number of puncture-black holes. AMR can also automatically resolve discontinuities without prior knowledge of their existence (cf.

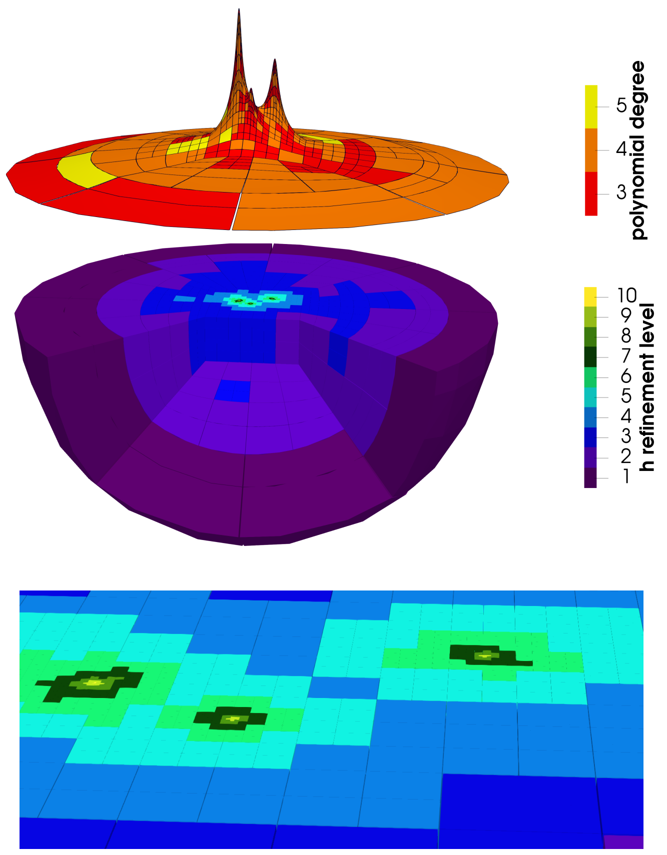

FIG. 20. Problem E (Black hole initial data with three punctures): Visualization of the solution, and the hp-refined computational mesh. **Top portion:** the $z = 0$ cross-section of the computational grid, color-coded by the polynomial degree, with the height representing the solution u. **Middle portion:** volume rendering of the $z < 0$ part of the computational domain, with two blocks removed, and color-coded by the h-refinement level. **Bottom portion:** Zoom into the middle portion, highlighting the region near the three punctures with highest refinement level. For ease of visualization, grid-points are mapped onto the compactified grid on which the estimator η_e is computed; the compactified outer radius $R = 3$ corresponds to the physical outer radius $R = 10^{11}$.

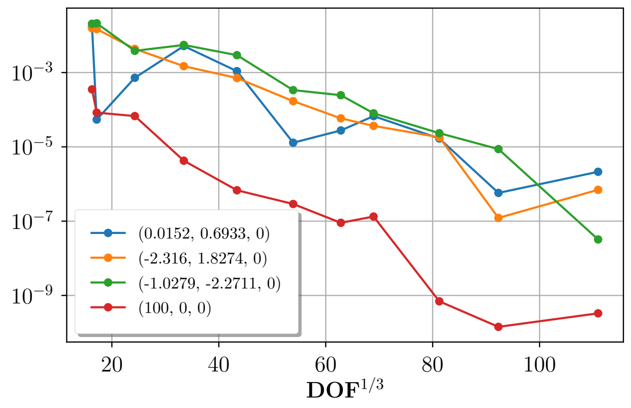

FIG. 21. Problem D: Black hole initial data with three randomly generated punctures. Convergence of the error between AMR steps at four different points, three corresponding to the location of the punctures and one at (100, 0, 0).

Fig. 11).

Moving forward, there are still several areas of improvement our code could possibly benefit from:

1. **Load balancing**: For problems that require adaptive mesh refinement and multi-grid, there will naturally be a unbalanced number of degrees of freedom (DOF) across processors and this can slow down the Krylov iterations substantially. This can be amended by incorporating a task-based parallelism framework for load balancing. The elliptic solver developed in this article will be incorporated task-based parallel code SpECTRE, which is concurrently being developed [21].

2. **Anisotropic refinement**: Most realistic problems have some anisotropy and therefore a solver would benefit from anisotropic mesh refinement. Indeed, most of the problems in this paper could have had better convergence with anisotropic refinement, for instance, Problem C is spherically symmetric, and puncture data is approximately spherically symmetric at large distance. We use the p4est framework for mesh refinement and while it has support for anisotropic refinement, the direction of the anisotropy has to be known a priori. We look to go beyond this and have general refinement in a future edition of our code.

3. **Hybridizable dG**: The discontinuous Galerkin method can be quite expensive in terms of the amount of DOF it requires to converge to a certain error. Recently, a method called Hybridizable dG has been coupled with matrix-free multigrid methods to solve elliptic problems with substantially reduced DOF over the classical dG method [80, 81]. Whether this method can be fully incorporated into the complex scheme presented in this paper, will be an area of further inquiry.

In future work, we plan to use this solver to expand the physics in compact object initial data, for instance, neutron star initial data for equation of state with phase-transitions, neutron stars with very high compactness (where current solvers fail [34]), or compact objects in alternative theories of gravity, or with novel matter fields like boson stars.

ACKNOWLEDGMENTS

The authors would like to acknowledge helpful discussions with Francois Hebert, Jan Hesthaven, Saul Teukolsky and Tim Warburton. This research was supported by NSERC of Canada. Computations were performed on the Minerva cluster at the Max-Planck-Institute for Gravitational Physics, and the GPC and Niagara supercomputers at the SciNet HPC Consortium [82]. SciNet is funded by: the Canada Foundation for Innovation; the Government of Ontario; Ontario Research Fund - Research Excellence; and the University of Toronto.

[1] W. Reed and T. Hill, *Conference: National topical meeting on mathematical models and computational techniques for analysis of nuclear systems, Ann Arbor, Michigan, USA, 8 Apr 1973*, Conference: National topical meeting on mathematical models and computational techniques for analysis of nuclear systems, Ann Arbor, Michigan, USA, 8 Apr 1973 LA-UR–73-479, CONF-730414–2 (1973).

[2] J. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Texts in Applied Mathematics (Springer, 2008).

[3] B. Cockburn, J. Comput. Appl. Math. **128**, 187 (2001).

[4] B. Cockburn and C.-W. Shu, J. Comput. Phys. **141**, 199 (1998).

[5] B. Cockburn, in *An introduction to the Discontinuous Galerkin method for convection-dominated problems; in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations: Lectures given at the 2nd Session of the Centro Internazionale Matematico Estivo (C.I.M.E.) held in Cetraro, Italy, June 23–28, 1997*, Lecture Notes in Physics, edited by A. Quarteroni (Springer, Berlin, New York, 1998) pp. 150–268.

[6] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, in *Discontinuous Galerkin Methods: Theory, Computation and Applications*, Lecture Notes in Computational Science and Engineering, edited by B. Cockburn, G. E. Karniadakis, and C.-W. Shu (Springer, Berlin, New York, 2000) pp. 3–50.

[7] T. W. Baumgarte and S. L. Shapiro, *Numerical relativity: solving Einstein's equations on the computer* (Cambridge University Press, 2010).

[8] S. E. Field, J. S. Hesthaven, S. R. Lau, and A. H. Mroue, Phys. Rev. D **82**, 104051 (2010), arXiv:1008.1820 [gr-qc].

[9] J. D. Brown, P. Diener, S. E. Field, J. S. Hesthaven, F. Herrmann, A. H. Mroué, O. Sarbach, E. Schnetter, M. Tiglio, and M. Wagman, Phys. Rev. D **85**, 084004 (2012), arXiv:1202.1038 [gr-qc].

[10] S. E. Field, J. S. Hesthaven, and S. R. Lau, Class. Quantum Grav. **26**, 165010 (2009), arXiv:0902.1287 [gr-qc].

[11] G. Zumbusch, Class. Quantum Grav. **26**, 175011 (2009), arXiv:0901.0851 [gr-qc].

[12] D. Radice and L. Rezzolla, Phys. Rev. D **84**, 024010 (2011), arXiv:1103.2426 [gr-qc].

[13] P. Mocz, M. Vogelsberger, D. Sijacki, R. Pakmor, and L. Hernquist, Mon. Not. Roy. Astr. Soc. **437**, 397 (2014), arXiv:1305.5536 [physics.comp-ph].

[14] O. Zanotti, M. Dumbser, A. Hidalgo, and D. Balsara, in *8th International Conference of Numerical Modeling of Space Plasma Flows (ASTRONUM 2013)*, Astronomical Society of the Pacific Conference Series, Vol. 488, edited by N. V. Pogorelov, E. Audit, and G. P. Zank (2014) pp. 285–290, arXiv:1401.6448 [astro-ph.IM].

[15] E. Endeve, C. D. Hauck, Y. Xing, and A. Mezzacappa, J. Comput. Phys. **287**, 151 (2015), arXiv:1410.7431 [physics.comp-ph].

[16] S. A. Teukolsky, J. Comput. Phys. **312**, 333 (2016), arXiv:1510.01190 [gr-qc].

[17] M. Bugner, T. Dietrich, S. Bernuzzi, A. Weyhausen, and B. Brügmann, Phys. Rev. D **94**, 084004 (2016), arXiv:1508.07147 [gr-qc].

[18] K. Schaal, A. Bauer, P. Chandrashekar, R. Pakmor, C. Klingenberg, and V. Springel, Monthly Notices of the Royal Astronomical Society **453**, 4278 (2015), arXiv:1506.06140 [astro-ph.CO].

[19] O. Zanotti, F. Fambri, and M. Dumbser, Monthly Notices of the Royal Astronomical Society **452**, 3010 (2015).

[20] J. M. Miller and E. Schnetter, Class. Quantum Grav. **34**, 015003 (2017), arXiv:1604.00075 [gr-qc].

[21] L. E. Kidder, S. E. Field, F. Foucart, E. Schnetter, S. A. Teukolsky, A. Bohn, N. Deppe, P. Diener, F. Hébert, J. Lippuner, J. Miller, C. D. Ott, M. A. Scheel, and T. Vincent, J. Comput. Phys. **335**, 84 (2017), arXiv:1609.00098 [astro-ph.HE].

[22] F. Fambri, M. Dumbser, S. Köppel, L. Rezzolla, and O. Zanotti, Mon. Not. Roy. Astr. Soc. (2018), 10.1093/mnras/sty734, arXiv:1801.02839 [physics.comp-ph].

[23] F. HÃľbert, L. E. Kidder, and S. A. Teukolsky, Phys. Rev. **D98**, 044041 (2018), arXiv:1804.02003 [gr-qc].

[24] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, SIAM Journal on Numerical Analysis **39**, 1749 (2002).

[25] J. Stiller, in *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016* (Springer, 2017) pp. 189–201, arXiv:1612.04796 [cs.NA].

[26] M. Kronbichler and W. A. Wall, SIAM Journal on Scientific Computing **40**, A3423 (2018), arXiv:1611.03029 [math.NA].

[27] P. W. Fick, arXiv preprint (2014), arXiv:1401.0339 [math.NA].

[28] J. E. Kozdon and L. C. Wilcox, Journal of Scientific Computing **76**, 1742 (2018), arXiv:1706.00513 [math.NA].

[29] J. E. Kozdon, L. C. Wilcox, T. Hagstrom, and J. W. Banks, Journal of Computational Physics (2019), arXiv:1806.06103 [math.NA].

[30] H. P. Pfeiffer, J. Hyperbol. Differ. Eq. **2**, 497 (2005), gr-qc/0412002.

[31] G. Cook, Living Rev. Rel. **3** (2000), 5.

[32] S. R. Lau, H. P. Pfeiffer, and J. S. Hesthaven, Commun. Comput. Phys. **6**, 1063 (2009), arXiv:0808.2597.

[33] S. R. Lau, G. Lovelace, and H. P. Pfeiffer, Phys. Rev. D **84**, 084023 (2011), arXiv:1105.3922 [gr-qc].

[34] K. Henriksson, F. Foucart, L. E. Kidder, and S. A. Teukolsky, Class. Quantum Grav. **33**, 105009 (2016), arXiv:1409.7159 [gr-qc].

[35] A. Tsokaros, B. C. Mundim, F. Galeazzi, L. Rezzolla, and K. Uryū, Physical Review D **94**, 044049 (2016), arXiv:1605.07205 [gr-qc].

[36] H. P. Pfeiffer, L. E. Kidder, M. A. Scheel, and S. A. Teukolsky, Comput. Phys. Commun. **152**, 253 (2003), gr-qc/0202096.

[37] S. Sherwin, R. Kirby, J. Peiró, R. Taylor, and O. Zienkiewicz, International journal for numerical methods in engineering **65**, 752 (2006).

[38] D. A. Di Pietro and A. Ern, *Mathematical aspects of discontinuous Galerkin methods*, Vol. 69 (Springer Science &amp; Business Media, 2011).

[39] G. Mengaldo, D. De Grazia, D. Moxey, P. Vincent, and S. Sherwin, Journal of Computational Physics **299**, 56 (2015).

[40] Y. Saad, *Iterative methods for sparse linear systems* (Siam, 2003).

[41] Y. Notay, SIAM Journal on Scientific Computing **22**, 1444 (2000).

[42] P. F. Antonietti and P. Houston, Journal of Scientific Computing **46**, 124 (2011).

[43] W. Briggs, V. Henson, and S. McCormick, *A Multigrid Tutorial: Second Edition* (Society for Industrial and Applied Mathematics, 2000).

[44] R. S. Sampath, S. S. Adavani, H. Sundar, I. Lashuk, and G. Biros, in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing* (IEEE Press, 2008) p. 18.

[45] H. Sundar, G. Biros, C. Burstedde, J. Rudi, O. Ghattas, and G. Stadler, in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analy-*

sis (IEEE Computer Society Press, 2012) p. 43.

[46] Y. Shang, Computers &amp; Mathematics with Applications **57**, 1369 (2009).

[47] H.-B. Li, T.-Z. Huang, Y. Zhang, X.-P. Liu, and T.-X. Gu, Applied Mathematics and Computation **218**, 260 (2011).

[48] P. Ghysels, P. Kłosiewicz, and W. Vanroose, Numerical Linear Algebra with Applications **19**, 253 (2012).

[49] J. A. Scales, Geophysical Journal International **97**, 179 (1989).

[50] H. E. Bell, The American Mathematical Monthly **72**, 292 (1965).

[51] H. C. Elman, O. G. Ernst, and D. P. O'leary, SIAM Journal on scientific computing **23**, 1291 (2001).

[52] P. Houston, D. Schötzau, and T. P. Wihler, Mathematical Models and Methods in Applied Sciences **17**, 33 (2007).

[53] P. Houston, E. Süli, and T. P. Wihler, IMA journal of numerical analysis **28**, 245 (2008).

[54] P. Houston, J. Robson, and E. Süli, IMA journal of numerical analysis **25**, 726 (2005).

[55] C. Bi, C. Wang, and Y. Lin, Computer Methods in Applied Mechanics and Engineering **297**, 140 (2015).

[56] D. Schötzau, C. Schwab, T. Wihler, and M. Wirz, *Exponential convergence of hp-DGFEM for elliptic problems in polyhedral domains* (Springer, 2014).

[57] P. Houston, D. Schötzau, and T. Wihler, J. Sci. Comput. **22**, 347 (2005).

[58] P. Hansbo and M. G. Larson, Computer Methods in Applied Mechanics and Engineering **200**, 3026 (2011).

[59] L. Zhu, S. Giani, P. Houston, and D. Schötzau, Mathematical Models and Methods in Applied Sciences **21**, 267 (2011).

[60] D. Schötzau and L. Zhu, Applied Numerical Mathematics **59**, 2236 (2009).

[61] P. Houston, I. Perugia, and D. Schötzau, Comput. Methods Appl. Mech. Eng. **194**, 499 (2005).

[62] C. Lovadina and L. Marini, J. Sci. Comput. **40**, 340 (2009).

[63] W. F. Mitchell and M. A. McClain, in *Recent advances in computational and applied mathematics* (Springer, 2011) pp. 227–258.

[64] J. M. Melenk and B. I. Wohlmuth, Advances in Computational Mathematics **15**, 311 (2001).

[65] Y. Feng, M. Straka, T. Di Matteo, and R. Croft, (2015).

[66] P. E. Buis and W. R. Dyksen, ACM Transactions on Mathematical Software (TOMS) **22**, 18 (1996).

[67] C. Burstedde, L. C. Wilcox, and O. Ghattas, SIAM Journal on Scientific Computing **33**, 1103 (2011).

[68] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, L. C. McInnes, and B. F. Smith, "PETSc home page," (2001), `http://www.mcs.anl.gov/petsc`.

[69] B. Stamm and T. Wihler, Mathematics of Computation **79**, 2117 (2010).

[70] T. W. Baumgarte, N. O'Murchadha, and H. P. Pfeiffer, Phys. Rev. D **75**, 044009 (2007), gr-qc/0610120.

[71] G. B. Cook, Phys. Rev. D **50**, 5025 (1994), gr-qc/9404043.

[72] G. B. Cook and H. P. Pfeiffer, Physical Review D **70**, 104016 (2004), gr-qc/0407078.

[73] M. Caudill, G. B. Cook, J. D. Grigsby, and H. P. Pfeiffer, Physical Review D **74**, 064011 (2006), arXiv:1410.7431 [physics.comp-ph].

[74] H. P. Pfeiffer, L. E. Kidder, M. A. Scheel, and S. A. Teukolsky, Computer physics communications **152**, 253 (2003), gr-qc/0202096.

[75] S. Brandt and B. Brügmann, Physical Review Letters **78**, 3606 (1997), gr-qc/9703066.

[76] M. Ansorg, B. Brügmann, and W. Tichy, Physical Review D **70**, 064011 (2004), gr-qc/0404056.

[77] M. Ansorg, Class. Quantum Grav. **24**, S1 (2007), gr-qc/0612081.

[78] K. A. Dennison, T. W. Baumgarte, and H. P. Pfeiffer, Phys. Rev. D **74**, 064016 (2006), gr-qc/0606037.

[79] G. Lovelace, R. Owen, H. P. Pfeiffer, and T. Chu, Phys. Rev. **D78**, 084017 (2008), arXiv:0805.4192 [gr-qc].

[80] M. S. Fabien, M. G. Knepley, R. T. Mills, and B. M. Riviere, SIAM Journal on Scientific Computing **41**, C73 (2019).

[81] S. Muralikrishnan, T. Bui-Thanh, and J. N. Shadid, arXiv preprint (2019), arXiv:1903.11045 [math.NA].

[82] C. Loken, D. Gruner, L. Groer, R. Peltier, N. Bunn, M. Craig, T. Henriques, J. Dempsey, C.-H. Yu, J. Chen, L. J. Dursi, J. Chong, S. Northrup, J. Pinto, N. Knecht, and R. V. Zon, J. Phys.: Conf. Ser. **256**, 012026 (2010).