

A HIGH-ORDER FAST DIRECT SOLVER FOR SURFACE PDEs*

DANIEL FORTUNATO[†]

Abstract. We introduce a fast direct solver for variable-coefficient elliptic partial differential equations on surfaces based on the hierarchical Poincaré–Steklov method. The method takes as input an unstructured, high-order quadrilateral mesh of a surface and discretizes surface differential operators on each element using a high-order spectral collocation scheme. Elemental solution operators and Dirichlet-to-Neumann maps tangent to the surface are precomputed and merged in a pairwise fashion to yield a hierarchy of solution operators that may be applied in $\mathcal{O}(N \log N)$ operations for a mesh with N elements. The resulting fast direct solver may be used to accelerate high-order implicit time-stepping schemes, as the precomputed operators can be reused for fast elliptic solves on surfaces. On a standard laptop, precomputation for a 12th-order surface mesh with over 1 million degrees of freedom takes 17 seconds, while subsequent solves take only 0.25 seconds. We apply the method to a range of problems on both smooth surfaces and surfaces with sharp corners and edges, including the static Laplace–Beltrami problem, the Hodge decomposition of a tangential vector field, and some time-dependent nonlinear reaction–diffusion systems.

Key words. surface PDE, fast direct solver, Laplace–Beltrami, hierarchical Poincaré–Steklov method, spectral element method, reaction–diffusion equation

AMS subject classifications. 65N35, 65N55, 65M60

1. Introduction. Surface-bound phenomena arise in a wide variety of applications, including electromagnetics [28, 29, 30, 17], plasma physics [6, 36, 55], biological pattern formation [48], bulk–surface diffusion processes [21, 58], biomechanics [70, 59], and fluid dynamics [64, 72]. Modeling such phenomena often leads to surface partial differential equations (PDEs)—that is, partial differential equations whose differential operators are taken with respect to the on-surface metric. Thus, the numerical solution of surface PDEs is an important component in the simulation of surface phenomena.

High-order numerical methods have gained popularity in recent years due to their high levels of accuracy and efficiency per degree of freedom, and a panoply of numerical methods have been developed for discretizing and solving PDEs on surfaces using a variety of geometry representations, with both low-order and high-order accuracy. Broadly speaking, surface PDE solvers may be divided into three main categories: mesh-based methods (e.g., finite element [27] and integral-equation-based [60] methods), meshless methods (e.g., radial basis function methods [34, 75]), and embedded methods (e.g., level set [11] and closest point [74] methods). Briefly, let us highlight each category with a particular focus on the fast and high-order accurate solvers available to each:

- **Mesh-based methods.** Perhaps most popular among mesh-based methods is the surface finite element method (FEM) [23, 24, 25, 26, 20, 27], which weakly enforces the PDE on each element of a surface mesh, along with coupling conditions at element interfaces. Surface discretizations based on the high-order FEM have been used in the setting of isogeometric analysis, but typically employ at most cubic or quartic polynomials [46, 49]. In addition, little work has been done on developing efficient solvers for the resulting linear systems in the high-order context on surfaces, where metric quantities can lead

*Submitted to the editors October 3, 2022.

Funding: This work was funded by the Simons Foundation.

[†]Center for Computational Mathematics, Flatiron Institute, New York, NY 10010 (dfortunato@flatironinstitute.org).

to variable coefficients with strong anisotropies. Geometric multigrid methods have achieved mesh-independent convergence rates when solving surface PDEs discretized by FEM, but have traditionally been applied to discretizations using only linear [51, 12, 1] or quadratic [15] basis functions. Sparse direct solvers such as UMFPACK [19] may be applied to surface FEM discretizations for moderate problem sizes, but often suffer from dense fill-in at higher orders.

For simple surface PDEs such as the Laplace–Beltrami equation, second-kind integral equation formulations exist based on the Green’s function for the three-dimensional Laplacian [60]. These formulations can be discretized to high-order accuracy to yield well-conditioned matrices which are amenable to solution via an iterative method in only a constant number of iterations. Moreover, each iteration may be accelerated to linear time by the fast multipole method [42]. However, the application of specialized near-singular quadrature rules can significantly hinder runtimes, even in modern implementations [2]. Finally, integral-equation-based approaches may not be applicable to the general variable-coefficient surface PDEs considered in this work.

For these mesh-based methods, high-order accuracy can be hampered by a surface mesh that is only low-order smooth. Recent work has shown that a flat surface mesh can be used to define a nearby high-order smooth surface mesh in a way that preserves multiscale features [73].

- **Meshless methods.** Meshless methods use scattered data (typically, a point cloud with neighbor information) to represent a surface via interpolation. The flexibility and robustness of the point cloud representation make such methods well suited to computing with noisy surface data, as might arise in imaging applications. Meshless methods based on radial basis functions (RBFs) can yield high-order accurate solutions of surface PDEs [34, 68], and result in sparse linear systems which may be inverted with a direct solver for moderate problem sizes. However, scaling can break down at high orders due to dense fill-in [67]. Recently, preconditioned iterative methods [52] and meshless multigrid methods [76] have been shown to yield “mesh”-independent convergence rates at higher orders, at least up to order 7. RBF methods have also been applied to solve PDEs on evolving surfaces [75].
- **Embedded methods.** Embedded methods recast a surface PDE as a volumetric PDE posed on all of \mathbb{R}^3 via extension, using a suitable mapping between points on the surface and points in the ambient space. Implicit surface representations based on the level set method have been used with Cartesian finite differences in the volume to yield second-order accurate discretizations for surface PDEs [11, 43, 44], though solving in a narrow band can degrade the order of convergence [43]. In addition, standard iterative solvers can require many iterations to converge [44], and the effect of higher-order finite difference stencils on solver complexity has not been well studied in this context.

Closest point methods [63, 54, 74, 53] take a similar approach, but can handle a range surfaces with open ends, cusps, and corners through the use of a closest point extension. Higher-order finite difference stencils have been studied using narrow-banding with closest point extension, and the resulting linear systems are generally sparse. However, iterative methods can stall without sufficient preconditioning. To that end, narrow-band multigrid methods have been used in conjunction with the closest point method to yield

mesh-independent iteration counts, though only for second-order stencils [16].

Finally, it should be noted that embedded methods necessarily require physical quantities defined on the surface to be extended into the volume—a task that can be challenging to do with high-order accuracy [5].

In this work we propose a fast direct solver for a high-order spectral element discretization of a surface PDE. Given a collection of geometric patches defining the surface, the method directly discretizes the strong form of the surface PDE on each patch using classical Chebyshev spectral collocation [71, 13]. Continuity and continuity of the binormal derivative are imposed between neighboring patches, which are merged pairwise in a tree-like fashion using the hierarchical Poincaré–Steklov (HPS) scheme [56, 40, 39, 7, 32, 57]. The method can be viewed as an operator analogue of classical nested dissection [37], with a factorization complexity of $\mathcal{O}(N^{3/2})$ and subsequent solve complexity of $\mathcal{O}(N \log N)$ for a mesh with N elements.

Direct solvers offer a few key advantages in the high-order setting. First, the complexity constant associated with direct solvers is typically small (at least for two-dimensional problems). Second, the large condition numbers commonly associated with high-order discretizations do not affect the performance of a direct solver [57]. Likewise, physical parameters such as variable coefficients and wavenumbers which could degrade the performance of an iterative method do not affect the runtime of a direct solver. Lastly, as the factorizations from a direct solver can be stored for reuse, repeated solves of the same PDE with different data are very fast. Because of this, direct solvers can be used to accelerate semi-implicit time-stepping schemes [8, 9] or handle localized changes to geometry [77] while maintaining high-order accuracy in space. Though the surfaces we consider in this work are embedded in three dimensions, their surface PDEs are two-dimensional problems, enabling all the benefits of fast direct solvers in two dimensions.

The paper is structured as follows. In [section 2](#), we define surface differential operators and spectral collocation on a single surface patch. [Section 3](#) describes the Schur complement method for domain decomposition on surfaces, starting from the linear system of a single element and moving to two “glued” elements. In [section 4](#), we describe the hierarchical fast direct solver and its computational complexity. [Section 5](#) presents numerical examples demonstrating both the speed and high-order convergence of the method for a variety of static and time-dependent problems.

2. Formulation and discretization.

2.1. Surface differential operators. Let Γ be a smooth surface embedded in \mathbb{R}^3 locally parametrized by the mapping $\mathbf{x}(\xi, \eta): \mathbb{R}^2 \rightarrow \mathbb{R}^3$, where $\mathbf{x}(\xi, \eta) = (x(\xi, \eta), y(\xi, \eta), z(\xi, \eta))$. The metric tensor g on Γ is given by

$$g = \begin{bmatrix} g_{\xi\xi} & g_{\xi\eta} \\ g_{\eta\xi} & g_{\eta\eta} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{bmatrix},$$

where $\mathbf{x}_\xi = \partial\mathbf{x}/\partial\xi$ and $\mathbf{x}_\eta = \partial\mathbf{x}/\partial\eta$. Let the components of the inverse of g be denoted by

$$g^{-1} = \begin{bmatrix} g^{\xi\xi} & g^{\xi\eta} \\ g^{\eta\xi} & g^{\eta\eta} \end{bmatrix} = \frac{1}{|g|} \begin{bmatrix} g_{\eta\eta} & -g_{\xi\eta} \\ -g_{\eta\xi} & g_{\xi\xi} \end{bmatrix},$$

with $|g| = \det g$, and define the inverse metric quantities

$$(2.1) \quad \xi_{\mathbf{x}} = g^{\xi\xi} \mathbf{x}_\xi + g^{\eta\xi} \mathbf{x}_\eta, \quad \eta_{\mathbf{x}} = g^{\xi\eta} \mathbf{x}_\xi + g^{\eta\eta} \mathbf{x}_\eta.$$

The surface gradient $\nabla_\Gamma = (\partial_x^\Gamma, \partial_y^\Gamma, \partial_z^\Gamma)$ of a scalar-valued function $u = u(\xi, \eta) := u(\mathbf{x}(\xi, \eta))$ defined on Γ is a tangential vector field which can be written as [33]

$$(2.2) \quad \nabla_\Gamma u = \begin{bmatrix} \partial_x^\Gamma u \\ \partial_y^\Gamma u \\ \partial_z^\Gamma u \end{bmatrix} = \begin{bmatrix} \xi_x \frac{\partial u}{\partial \xi} + \eta_x \frac{\partial u}{\partial \eta} \\ \xi_y \frac{\partial u}{\partial \xi} + \eta_y \frac{\partial u}{\partial \eta} \\ \xi_z \frac{\partial u}{\partial \xi} + \eta_z \frac{\partial u}{\partial \eta} \end{bmatrix}.$$

The components ∂_x^Γ , ∂_y^Γ , and ∂_z^Γ may be interpreted as tangential derivative operators on Γ in the Cartesian directions \mathbf{e}_x , \mathbf{e}_y , and \mathbf{e}_z , respectively. Similarly, the surface divergence of a vector field $\mathbf{u} = \mathbf{u}(\xi, \eta) = (u_1(\xi, \eta), u_2(\xi, \eta), u_3(\xi, \eta))$ tangent to Γ can be written as

$$\nabla_\Gamma \cdot \mathbf{u} = \partial_x^\Gamma u_1 + \partial_y^\Gamma u_2 + \partial_z^\Gamma u_3.$$

The surface Laplacian Δ_Γ , or Laplace–Beltrami operator, is then given by

$$\Delta_\Gamma u = \nabla_\Gamma \cdot \nabla_\Gamma u = \partial_x^\Gamma \partial_x^\Gamma u + \partial_y^\Gamma \partial_y^\Gamma u + \partial_z^\Gamma \partial_z^\Gamma u.$$

The Laplace–Beltrami operator is the generalization of the standard Laplacian to general manifolds. Just as the Poisson equation is the canonical elliptic PDE in Euclidean space, the Laplace–Beltrami equation is the canonical elliptic PDE on surfaces. Higher-order surface differential operators, such as the surface biharmonic operator, may be defined similarly. For simplicity of presentation, we restrict our attention to second-order surface differential operators in this work.

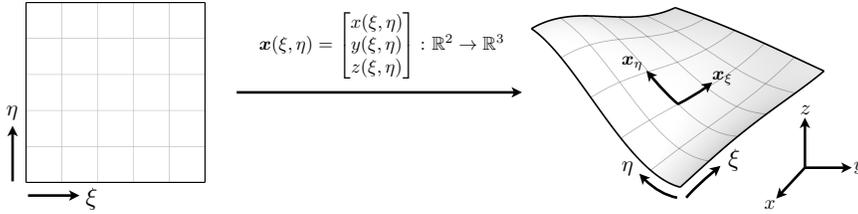


FIG. 2.1. The reference square with coordinates $(\xi, \eta) \in [-1, 1]^2$ is mapped to a quadrilateral element via the mapping $\mathbf{x}(\xi, \eta): [-1, 1]^2 \rightarrow \mathbb{R}^3$. The partial derivatives \mathbf{x}_ξ and \mathbf{x}_η are used to form the metric tensor g , which encodes how lengths and angles change over the surface.

We consider a general elliptic surface PDE on Γ ,

$$(2.3) \quad \mathcal{L}_\Gamma u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma,$$

where $f(\mathbf{x})$ is a smooth function on Γ and \mathcal{L}_Γ is a variable-coefficient, second-order, linear, elliptic partial differential operator of the form

$$(2.4) \quad \mathcal{L}_\Gamma u(\mathbf{x}) = \sum_{i=1}^3 \sum_{j=i}^3 a_{ij}(\mathbf{x}) \partial_i^\Gamma \partial_j^\Gamma u(\mathbf{x}) + \sum_{i=1}^3 b_i(\mathbf{x}) \partial_i^\Gamma u(\mathbf{x}) + c(\mathbf{x})u(\mathbf{x})$$

with smooth coefficients $a_{11}(\mathbf{x})$, $a_{22}(\mathbf{x})$, $a_{33}(\mathbf{x})$, $a_{12}(\mathbf{x})$, $a_{23}(\mathbf{x})$, $a_{13}(\mathbf{x})$, $b_1(\mathbf{x})$, $b_2(\mathbf{x})$, $b_3(\mathbf{x})$, and $c(\mathbf{x})$ for $\mathbf{x} \in \Gamma$. Here, we identify ∂_1^Γ , ∂_2^Γ , ∂_3^Γ with ∂_x^Γ , ∂_y^Γ , ∂_z^Γ , respectively. If Γ is not a closed surface, (2.3) may also be subject to boundary conditions, e.g., $u(\mathbf{x}) = g(\mathbf{x})$ for $\mathbf{x} \in \partial\Gamma$ and some function g .

2.2. Discrete surfaces and spectral collocation. To numerically compute with the surface Γ , we must discretize the geometry. While many choices of discrete surface representation exist, such as level sets [66] and point clouds [61, 34], we choose to use surface meshes as they provide a flexible level of compatibility with computer-aided design (CAD) software. Specifically, we assume that Γ has been provided as a high-order quadrilateral¹ surface mesh consisting of a set of tensor-product elements $\{\mathcal{E}_k\}_{k=1}^N$. Such a mesh may be constructed through the use of high-order meshing software such as Gmsh [38] or Rhinoceros [62].

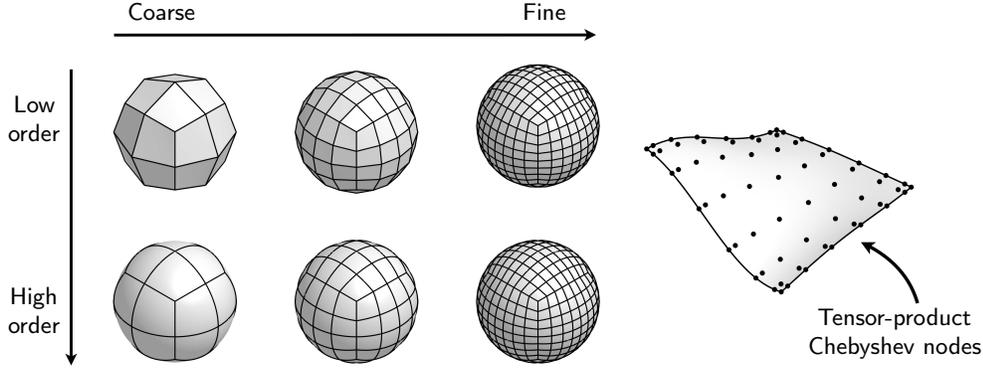


FIG. 2.2. (Left) Surface meshes may be constructed and refined along two axes: a coarse mesh may be subdivided into a fine mesh with many elements and the geometry may be represented using low-order or high-order polynomials. (Right) We represent each element using tensor product Chebyshev nodes of the second kind.

Our high-order format is based on tensor-product Chebyshev nodes, though other node sets could suffice. Let $\{\boldsymbol{\xi}_{ij} = (\xi_{ij}, \eta_{ij})\}_{i,j=1}^{p+1}$ be the set of tensor-product second-kind Chebyshev nodes of order p over the reference square $[-1, 1]^2$. We say that the mesh is order p if each element has been provided as a set of nodes $\{\mathbf{x}_{ij} = (x_{ij}, y_{ij}, z_{ij})\}_{i,j=1}^{p+1}$ sampled at $\{\boldsymbol{\xi}_{ij}\}_{i,j=1}^{p+1}$ on Γ . That is, the coordinate mapping for each element is a function $\mathbf{x}(\xi, \eta) : [-1, 1]^2 \mapsto \mathbb{R}^3$ such that $\mathbf{x}(\xi_{ij}, \eta_{ij}) = (x_{ij}, y_{ij}, z_{ij})$ for $i, j = 1, \dots, p+1$ (see Figure 2.1). Thus, on each element we can numerically approximate the coordinate mapping via interpolation through the nodes $\{\mathbf{x}_{ij}\}_{i,j=1}^{p+1}$:

$$(2.5) \quad \mathbf{x}(\xi, \eta) \approx \sum_{i=1}^{p+1} \sum_{j=1}^{p+1} \mathbf{x}_{ij} \ell_j(\xi) \ell_i(\eta), \quad (\xi, \eta) \in [-1, 1]^2,$$

where ℓ_j is the j th Lagrange polynomial associated with the second-kind Chebyshev nodes. Partial derivatives of the elemental coordinate maps, \mathbf{x}_ξ and \mathbf{x}_η , may then be computed through numerical spectral differentiation [71] and used to form an approximation to the metric tensor, g , on each element.

We now describe a spectral collocation method to discretize (2.3) on a single element \mathcal{E}_k . To approximate functions defined on each element we use an isoparametric representation, wherein the function is approximated on the same order- p nodes as the element geometry. For a function $u(\mathbf{x})$ defined for $\mathbf{x} \in \mathcal{E}_k$, denote by $u_{ij} \approx u(\mathbf{x}_{ij})$; that is, u_{ij} is simply the function u sampled on the grid. Then, just like (2.5), we have

¹We use quadrilateral elements throughout this work for simplicity and ease of implementation, but the algorithms described below can be modified to use triangular elements instead.

that

$$u(\xi, \eta) \approx \sum_{i=1}^{p+1} \sum_{j=1}^{p+1} u_{ij} \ell_j(\xi) \ell_i(\eta), \quad (\xi, \eta) \in [-1, 1]^2,$$

where we have introduced the slight abuse of notation $u(\xi, \eta) := u(\mathbf{x}(\xi, \eta))$.

To discretize \mathcal{L}_Γ on the element \mathcal{E}_k , we will compute discrete operators on the reference square $[-1, 1]^2$ and then map them to \mathcal{E}_k using the numerical coordinate mapping. Let $D \in \mathbb{C}^{(p+1) \times (p+1)}$ be the one-dimensional spectral differentiation matrix [71] associated with second-kind Chebyshev points on the interval $[-1, 1]$ and let $I \in \mathbb{C}^{(p+1) \times (p+1)}$ be the identity matrix. Two-dimensional differentiation matrices on the reference square $[-1, 1]^2$ in the ξ - and η -directions can be constructed through the Kronecker products

$$D_\xi = D \otimes I, \quad D_\eta = I \otimes D,$$

respectively, where $D_\xi, D_\eta \in \mathbb{C}^{(p+1)^2 \times (p+1)^2}$. Let $M[u] \in \mathbb{C}^{(p+1)^2 \times (p+1)^2}$ denote the diagonal multiplication matrix formed by placing the entries of u_{ij} along the diagonal; we refer to $M[u]$ as a multiplication matrix because $M[u]v_{ij}$ computes the pointwise product of the functions u and v on the grid. Using (2.2), differentiation matrices corresponding to the components of the surface gradient on \mathcal{E}_k are given by

$$(2.6) \quad \begin{aligned} D_x^\Gamma &= M[\xi_x] D_\xi + M[\eta_x] D_\eta, \\ D_y^\Gamma &= M[\xi_y] D_\xi + M[\eta_y] D_\eta, \\ D_z^\Gamma &= M[\xi_z] D_\xi + M[\eta_z] D_\eta, \end{aligned}$$

respectively, where the values $\{(\xi_{\mathbf{x}})_{ij}\}$ and $\{(\eta_{\mathbf{x}})_{ij}\}$ are tabulated using the numerical approximations to \mathbf{x}_ξ , \mathbf{x}_η , and g in (2.1). The operator \mathcal{L}_Γ given in (2.4) is then discretized on the element \mathcal{E}_k as the $(p+1)^2 \times (p+1)^2$ matrix

$$(2.7) \quad L_{\mathcal{E}_k} = \sum_{i=1}^3 \sum_{j=i}^3 M[a_{ij}] D_i^\Gamma D_j^\Gamma + \sum_{i=1}^3 M[b_i] D_i^\Gamma + M[c],$$

with all variable coefficients sampled on the grid. Again, we identify D_1^Γ , D_2^Γ , D_3^Γ with D_x^Γ , D_y^Γ , D_z^Γ , respectively. Discrete surface gradient, surface divergence, and Laplace–Beltrami operators may be similarly defined. For instance, the discrete Laplace–Beltrami operator may be discretized as $\Delta_\Gamma \approx (D_x^\Gamma)^2 + (D_y^\Gamma)^2 + (D_z^\Gamma)^2$.

3. Domain decomposition on surfaces.

3.1. A single element. Consider the single-element boundary value problem on the element \mathcal{E}_k ,

$$(3.1a) \quad \mathcal{L}_\Gamma u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \mathcal{E}_k,$$

$$(3.1b) \quad u(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\mathcal{E}_k.$$

Using (2.7), we discretize (3.1a) as the $(p+1)^2 \times (p+1)^2$ linear system

$$(3.2) \quad L_{\mathcal{E}_k} \mathbf{u} = \mathbf{f},$$

where $\mathbf{u}, \mathbf{f} \in \mathbb{C}^{(p+1)^2 \times 1}$ are the vectors formed by stacking the values $\{u_{ij}\}$ and $\{f_{ij}\}$ column-wise. To impose the boundary condition (3.1b) we use the approach

of “boundary bordering,” whereby rows of the differential operator corresponding to boundary nodes are replaced by rows enforcing the boundary conditions. While the technique of rectangular spectral collocation [22] provides a robust way to impose general boundary conditions in spectral collocation methods without row deletion, a simple boundary bordering approach suffices for the Dirichlet problems considered here.

To illustrate which rows of (3.2) correspond to boundary nodes, it is helpful to reorder the degrees of freedom. To that end, let \mathcal{J}_i and \mathcal{J}_b be the sets of (linear) indices for which the Chebyshev nodes $\{\xi_{ij}\}$ lie in the interior of the reference square or on its boundary, respectively, and denote their sizes by $n_i = |\mathcal{J}_i| = (p-1)^2$ and $n_b = |\mathcal{J}_b| = 4p$. For a matrix or vector, we will use superscripts to denote indices for row and column slicing; e.g., for a matrix A , A^{ib} is the submatrix $A(\mathcal{J}_i, \mathcal{J}_b)$. Reordering the degrees of freedom in (3.2) in the order $\{\mathcal{J}_i, \mathcal{J}_b\}$ gives a block linear system,

$$(3.3) \quad \begin{bmatrix} L_{\mathcal{E}_k}^{\text{ii}} & L_{\mathcal{E}_k}^{\text{ib}} \\ L_{\mathcal{E}_k}^{\text{bi}} & L_{\mathcal{E}_k}^{\text{bb}} \end{bmatrix} \begin{bmatrix} \mathbf{u}^i \\ \mathbf{u}^b \end{bmatrix} = \begin{bmatrix} \mathbf{f}^i \\ \mathbf{f}^b \end{bmatrix}.$$

We now proceed by replacing the last n_b rows of (3.3) with the boundary conditions (3.1b), yielding the modified linear system

$$(3.4) \quad \begin{bmatrix} L_{\mathcal{E}_k}^{\text{ii}} & L_{\mathcal{E}_k}^{\text{ib}} \\ 0 & I_{n_b} \end{bmatrix} \begin{bmatrix} \mathbf{u}^i \\ \mathbf{u}^b \end{bmatrix} = \begin{bmatrix} \mathbf{f}^i \\ \mathbf{g} \end{bmatrix},$$

where $\mathbf{g} \in \mathbb{C}^{n_b \times 1}$ is the vector of boundary values of g and I_{n_b} is the $n_b \times n_b$ identity matrix. Next, we eliminate \mathbf{u}^b from this system by performing a Schur complement. In this case, as (3.1b) imposes Dirichlet boundary conditions, we immediately obtain that $\mathbf{u}^b = \mathbf{g}$. Substituting this into (3.4) yields a reduced $n_i \times n_i$ linear system for the interior unknowns \mathbf{u}^i ,

$$(3.5) \quad L_{\mathcal{E}_k}^{\text{ii}} \mathbf{u}^i = \mathbf{f}^i - L_{\mathcal{E}_k}^{\text{ib}} \mathbf{g}.$$

Upon solving (3.5), the solution values at all Chebyshev nodes on \mathcal{E}_k have now been computed. It is useful to note a few things:

- First, we may write the solution u as the sum of a homogeneous solution, w , and a particular solution, v ,

$$u = w + v,$$

where w satisfies

$$(3.6) \quad \begin{aligned} \mathcal{L}_\Gamma w(\mathbf{x}) &= 0, & \mathbf{x} \in \mathcal{E}_k, \\ w(\mathbf{x}) &= g(\mathbf{x}), & \mathbf{x} \in \partial\mathcal{E}_k, \end{aligned}$$

and v satisfies

$$(3.7) \quad \begin{aligned} \mathcal{L}_\Gamma v(\mathbf{x}) &= f(\mathbf{x}), & \mathbf{x} \in \mathcal{E}_k, \\ v(\mathbf{x}) &= 0, & \mathbf{x} \in \partial\mathcal{E}_k. \end{aligned}$$

Equation (3.5) then becomes

$$(3.8) \quad \begin{aligned} \mathbf{w}^i &= -(L_{\mathcal{E}_k}^{\text{ii}})^{-1} L_{\mathcal{E}_k}^{\text{ib}} \mathbf{g}, & \mathbf{v}^i &= (L_{\mathcal{E}_k}^{\text{ii}})^{-1} \mathbf{f}^i, \\ \mathbf{w}^b &= \mathbf{g}, & \mathbf{v}^b &= 0, \end{aligned}$$

where $\mathbf{u} = \mathbf{w} + \mathbf{v}$.

- Second, we may encode as a linear operator the action of solving (3.6) given any boundary data g . This operator, often called the “solution operator,” is discretized as the $(p+1)^2 \times n_b$ matrix

$$(3.9) \quad S_{\mathcal{E}_k}(\mathcal{J}_i, \cdot) = - (L_{\mathcal{E}_k}^{ii})^{-1} L_{\mathcal{E}_k}^{ib}, \quad S_{\mathcal{E}_k}(\mathcal{J}_b, \cdot) = I_{n_b}.$$

Given *any* vector of boundary data \mathbf{g} , the product $\mathbf{w} = S_{\mathcal{E}_k} \mathbf{g}$ approximates w satisfying (3.6).

Remark 3.1. From a domain decomposition point of view it is convenient to work on a boundary grid which does not include the four corner nodes, so operators that act on the boundary degrees of freedom (such as $S_{\mathcal{E}_k}$) may be unambiguously separated into pieces that act on each side of an element [40]. To achieve this effect, we choose the boundary grid on each edge to be the first-kind Chebyshev nodes and apply mappings to all boundary operators to re-interpolate to this node set. For stability, we use a boundary grid of degree $p-2$ on each side [39]; thus, the number of boundary nodes is modified to be $n_b = 4(p-1)$.

We now have all the ingredients necessary to locally solve a surface PDE on each element of the mesh. To couple elements together, we take a non-overlapping domain decomposition approach. We begin with the simplest possible domain decomposition problem, consisting of just two surface elements.

3.2. Two “glued” elements. Consider the model problem of two “glued” surface elements, \mathcal{E}_α and \mathcal{E}_β , separated by an interface $\mathcal{I}_{\alpha\beta} = \mathcal{E}_\alpha \cap \mathcal{E}_\beta$, depicted in Figure 3.1 (left):

$$(3.10) \quad \begin{aligned} \mathcal{L}_\Gamma u_\alpha(\mathbf{x}) &= f_\alpha(\mathbf{x}), & \mathbf{x} \in \mathcal{E}_\alpha, \\ \mathcal{L}_\Gamma u_\beta(\mathbf{x}) &= f_\beta(\mathbf{x}), & \mathbf{x} \in \mathcal{E}_\beta, \\ u_\alpha(\mathbf{x}) &= g_\alpha(\mathbf{x}), & \mathbf{x} \in \partial\mathcal{E}_\alpha \setminus \mathcal{I}_{\alpha\beta}, \\ u_\beta(\mathbf{x}) &= g_\beta(\mathbf{x}), & \mathbf{x} \in \partial\mathcal{E}_\beta \setminus \mathcal{I}_{\alpha\beta}, \\ u_\alpha(\mathbf{x}) &= u_\beta(\mathbf{x}), & \mathbf{x} \in \mathcal{I}_{\alpha\beta}, \\ \frac{\partial u_\alpha}{\partial \mathbf{n}_b}(\mathbf{x}) &= \frac{\partial u_\beta}{\partial \mathbf{n}_b}(\mathbf{x}), & \mathbf{x} \in \mathcal{I}_{\alpha\beta}, \end{aligned}$$

where \mathbf{n}_b is the binormal vector along the interface. As before, we write u_α and u_β as the sum of homogeneous and particular solutions, $u_\alpha = w_\alpha + v_\alpha$ and $u_\beta = w_\beta + v_\beta$, with discrete versions $\mathbf{u}_\alpha = \mathbf{w}_\alpha + \mathbf{v}_\alpha$ and $\mathbf{u}_\beta = \mathbf{w}_\beta + \mathbf{v}_\beta$, where w_α and w_β satisfy (3.10) with $f_\alpha \equiv f_\beta \equiv 0$ and v_α and v_β satisfy (3.10) with $g_\alpha \equiv g_\beta \equiv 0$.

Let $\mathcal{J}_{s_\alpha}, \mathcal{J}_{s_\beta} \subset \{1, \dots, n_b\}$ be index sets corresponding to the shared interface nodes (i.e., the nodes which are interior to the merged domain $\mathcal{E}_\alpha \cup \mathcal{E}_\beta$) with respect to elements \mathcal{E}_α and \mathcal{E}_β , and \mathcal{J}_{b_α} and \mathcal{J}_{b_β} the remaining indices corresponding to the boundary nodes. See Figure 3.1 (right) for a diagram. As a reminder, we use superscripts to denote index slicing. As the nodes of \mathcal{E}_α and \mathcal{E}_β are identical along the shared interface $\mathcal{I}_{\alpha\beta}$, continuity of the solution across $\mathcal{I}_{\alpha\beta}$ simply means that $\mathbf{u}_\alpha^{s_\alpha} = \mathbf{u}_\beta^{s_\beta}$, so let us denote these solution values by $\mathbf{u}_{\mathcal{I}_{\alpha\beta}}$. To enforce continuity of the binormal derivative, let us evaluate the outward pointing binormal vectors of each element \mathcal{E}_k at the n_b boundary nodes and collect the Cartesian components of each into the discrete vectors $\mathbf{n}_{k,x}$, $\mathbf{n}_{k,y}$, and $\mathbf{n}_{k,z}$, for $k = \alpha, \beta$. Then, we may define binormal derivative operators $D_{\mathcal{E}_\alpha}$ and $D_{\mathcal{E}_\beta}$ that map function values in each element

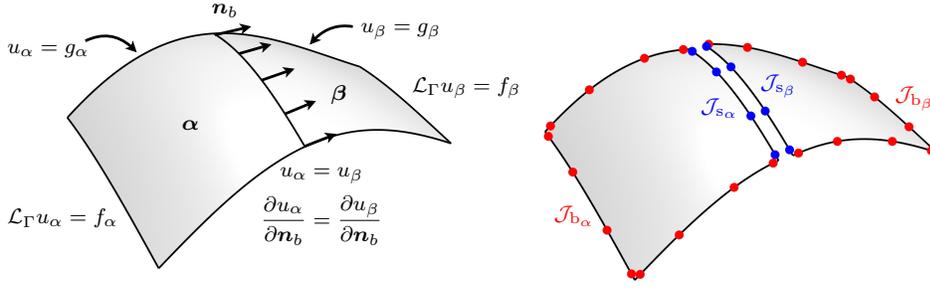


FIG. 3.1. (Left) Two patches are “glued” together by imposing continuity and continuity of the binormal derivative across the interface between them. (Right) Indices corresponding to the shared and unshared boundary degrees of freedom are recorded for each patch. Values at the shared interface can be eliminated using the Schur complement method.

to the values of the outgoing flux on the boundary via

$$(3.11) \quad D_{\mathcal{E}_k} = \mathbf{n}_{k,x} \circ D_x^\Gamma(\mathcal{J}_b, \cdot) + \mathbf{n}_{k,y} \circ D_y^\Gamma(\mathcal{J}_b, \cdot) + \mathbf{n}_{k,z} \circ D_z^\Gamma(\mathcal{J}_b, \cdot),$$

for $k = \alpha, \beta$, where \circ denotes the Hadamard product. The derivatives of u_α and u_β in the direction \mathbf{n}_b along the interface can then be written as $(D_{\mathcal{E}_\alpha} \mathbf{u}_\alpha)^{s_\alpha}$ and $-(D_{\mathcal{E}_\beta} \mathbf{u}_\beta)^{s_\beta}$, respectively. Therefore, continuity of the binormal derivative across the interface is equivalent to

$$(3.12) \quad (D_{\mathcal{E}_\alpha} \mathbf{u}_\alpha)^{s_\alpha} + (D_{\mathcal{E}_\beta} \mathbf{u}_\beta)^{s_\beta} = 0.$$

Now, let $S_{\mathcal{E}_\alpha}$ and $S_{\mathcal{E}_\beta}$ be the solution operators for elements \mathcal{E}_α and \mathcal{E}_β , respectively. If we knew the value of $\mathbf{u}_{\mathcal{I}_{\alpha\beta}}$, then boundary data on all sides of each element would be specified and we could directly apply the solution operators to each element in an uncoupled fashion. So let $\tilde{\mathbf{g}}_\alpha, \tilde{\mathbf{g}}_\beta \in \mathbb{C}^{n_b \times 1}$ be the vectors of boundary data on all sides of each element, with entries given by $\tilde{\mathbf{g}}_\alpha^{s_\alpha} = \tilde{\mathbf{g}}_\beta^{s_\beta} = \mathbf{u}_{\mathcal{I}_{\alpha\beta}}$, $\tilde{\mathbf{g}}_\alpha^{b_\alpha} = \mathbf{g}_\alpha$, and $\tilde{\mathbf{g}}_\beta^{b_\beta} = \mathbf{g}_\beta$. With this definition, the homogeneous solutions on each element can be written as $\mathbf{w}_\alpha = S_{\mathcal{E}_\alpha} \tilde{\mathbf{g}}_\alpha$ and $\mathbf{w}_\beta = S_{\mathcal{E}_\beta} \tilde{\mathbf{g}}_\beta$. Expanding $\mathbf{u}_\alpha = \mathbf{w}_\alpha + \mathbf{v}_\alpha$ and $\mathbf{u}_\beta = \mathbf{w}_\beta + \mathbf{v}_\beta$ in (3.12), we have

$$(D_{\mathcal{E}_\alpha} S_{\mathcal{E}_\alpha} \tilde{\mathbf{g}}_\alpha + D_{\mathcal{E}_\alpha} \mathbf{v}_\alpha)^{s_\alpha} + (D_{\mathcal{E}_\beta} S_{\mathcal{E}_\beta} \tilde{\mathbf{g}}_\beta + D_{\mathcal{E}_\beta} \mathbf{v}_\beta)^{s_\beta} = 0.$$

The matrices $\Sigma_{\mathcal{E}_\alpha} := D_{\mathcal{E}_\alpha} S_{\mathcal{E}_\alpha}$ and $\Sigma_{\mathcal{E}_\beta} := D_{\mathcal{E}_\beta} S_{\mathcal{E}_\beta}$ are discrete *Dirichlet-to-Neumann* maps, so termed because they map Dirichlet data on the boundary of \mathcal{E}_k to the outward flux of the local solution to the PDE on the boundary of \mathcal{E}_k . The vectors $\mathbf{v}'_\alpha := D_{\mathcal{E}_\alpha} \mathbf{v}_\alpha$ and $\mathbf{v}'_\beta := D_{\mathcal{E}_\beta} \mathbf{v}_\beta$ are the “particular fluxes”—the outgoing fluxes of the particular solution evaluated on the boundary of each element. Using this notation and recalling the definitions of $\tilde{\mathbf{g}}_\alpha$ and $\tilde{\mathbf{g}}_\beta$, continuity of the binormal derivative across the interface yields an equation for the unknown vector of interface values $\mathbf{u}_{\mathcal{I}_{\alpha\beta}}$,

$$\left(\Sigma_{\mathcal{E}_\alpha}^{s_\alpha s_\alpha} \mathbf{u}_{\mathcal{I}_{\alpha\beta}} + \Sigma_{\mathcal{E}_\alpha}^{s_\alpha b_\alpha} \mathbf{g}_\alpha + \mathbf{v}'_\alpha^{s_\alpha} \right) + \left(\Sigma_{\mathcal{E}_\beta}^{s_\beta s_\beta} \mathbf{u}_{\mathcal{I}_{\alpha\beta}} + \Sigma_{\mathcal{E}_\beta}^{s_\beta b_\beta} \mathbf{g}_\beta + \mathbf{v}'_\beta^{s_\beta} \right) = 0.$$

Writing $\mathbf{u}_{\mathcal{I}_{\alpha\beta}}$ as the sum of homogeneous and particular solutions, $\mathbf{u}_{\mathcal{I}_{\alpha\beta}} = \mathbf{v}_{\mathcal{I}_{\alpha\beta}} + \mathbf{w}_{\mathcal{I}_{\alpha\beta}}$, and substituting into the above, yields linear systems for the homogeneous and particular interface unknowns,

$$(3.13a) \quad - \left(\Sigma_{\mathcal{E}_\alpha}^{s_\alpha s_\alpha} + \Sigma_{\mathcal{E}_\beta}^{s_\beta s_\beta} \right) \mathbf{w}_{\mathcal{I}_{\alpha\beta}} = \Sigma_{\mathcal{E}_\alpha}^{s_\alpha b_\alpha} \mathbf{g}_\alpha + \Sigma_{\mathcal{E}_\beta}^{s_\beta b_\beta} \mathbf{g}_\beta,$$

$$(3.13b) \quad - \left(\Sigma_{\mathcal{E}_\alpha}^{s_\alpha s_\alpha} + \Sigma_{\mathcal{E}_\beta}^{s_\beta s_\beta} \right) \mathbf{v}_{\mathcal{I}_{\alpha\beta}} = \mathbf{v}'_\alpha^{s_\alpha} + \mathbf{v}'_\beta^{s_\beta}.$$

Upon solution of (3.13b) and (3.13a), boundary data is known on all sides of both elements \mathcal{E}_α and \mathcal{E}_β , and the local solution operators $S_{\mathcal{E}_\alpha}$ and $S_{\mathcal{E}_\beta}$ may be applied to recover the solutions \mathbf{u}_α and \mathbf{u}_β to (3.10) in the interiors of each element. As before, we may encode as a linear operator the action of solving (3.13a) given any boundary data \mathbf{g}_α and \mathbf{g}_β . This is the solution operator for the interface, $S_{\mathcal{I}_{\alpha\beta}}$, and is given by the solution to

$$(3.14) \quad - \left(\Sigma_{\mathcal{E}_\alpha}^{s_\alpha s_\alpha} + \Sigma_{\mathcal{E}_\beta}^{s_\beta s_\beta} \right) S_{\mathcal{I}_{\alpha\beta}} = \begin{bmatrix} \Sigma_{\mathcal{E}_\alpha}^{s_\alpha b_\alpha} & \Sigma_{\mathcal{E}_\beta}^{s_\beta b_\beta} \end{bmatrix}.$$

The operator $S_{\mathcal{I}_{\alpha\beta}}$ takes in Dirichlet data on the boundary nodes of the merged domain $\mathcal{E}_{\alpha\beta} := \mathcal{E}_\alpha \cup \mathcal{E}_\beta$ and returns the values of the homogeneous solution along the interface, $\mathbf{w}_{\mathcal{I}_{\alpha\beta}}$.

Note that when ‘‘gluing’’ two elements, computing $S_{\mathcal{I}_{\alpha\beta}}$ and $\mathbf{v}_{\mathcal{I}_{\alpha\beta}}$ only requires knowledge of the Dirichlet-to-Neumann map and particular flux for each element. Thus, obtaining the Dirichlet-to-Neumann map and particular flux of the merged domain will allow us to recursively apply this ‘‘gluing’’ procedure. The matrix $\Sigma_{\mathcal{E}_\alpha}^{s_\alpha s_\alpha} + \Sigma_{\mathcal{E}_\beta}^{s_\beta s_\beta}$ may be viewed as a Schur complement of the block linear system corresponding to direct discretization of (3.10), taken with respect to the interface degrees of freedom. The Schur complement also allows us to construct the Dirichlet-to-Neumann map for the merged domain $\mathcal{E}_{\alpha\beta}$, using the Dirichlet-to-Neumann maps for the elements \mathcal{E}_α and \mathcal{E}_β and the solution operator for the interface, $S_{\mathcal{I}_{\alpha\beta}}$, as

$$(3.15) \quad \Sigma_{\mathcal{E}_{\alpha\beta}} = \begin{bmatrix} \Sigma_{\mathcal{E}_\alpha}^{b_\alpha b_\alpha} & 0 \\ 0 & \Sigma_{\mathcal{E}_\beta}^{b_\beta b_\beta} \end{bmatrix} + \begin{bmatrix} \Sigma_{\mathcal{E}_\alpha}^{b_\alpha s_\alpha} \\ \Sigma_{\mathcal{E}_\beta}^{b_\beta s_\beta} \end{bmatrix} S_{\mathcal{I}_{\alpha\beta}}.$$

Similarly, the particular flux for the merged domain is given by

$$(3.16) \quad \mathbf{v}'_{\alpha\beta} = \begin{bmatrix} \mathbf{v}'_\alpha{}^{b_\alpha} \\ \mathbf{v}'_\beta{}^{b_\beta} \end{bmatrix} + \begin{bmatrix} \Sigma_{\mathcal{E}_\alpha}^{b_\alpha s_\alpha} \\ \Sigma_{\mathcal{E}_\beta}^{b_\beta s_\beta} \end{bmatrix} \mathbf{w}_{\mathcal{I}_{\alpha\beta}}.$$

4. A fast direct solver. We now formulate the general hierarchical fast direct solver for a mesh of N elements, $\{\mathcal{E}_k\}_{k=1}^N$. The solver is based on the the HPS scheme [56, 40, 39, 7, 32, 57] and proceeds pairwise by recursively merging neighboring elements or groups of elements until a factorization of the surface PDE over entire surface mesh has been computed. We refer to this method as the surface HPS scheme.

4.1. Local factorization stage. The method begins with the local factorization stage, in which a solution operator $S_{\mathcal{E}_k}$, Dirichlet-to-Neumann map $\Sigma_{\mathcal{E}_k}$, particular solution \mathbf{v}_k , and particular flux \mathbf{v}'_k are computed for every element \mathcal{E}_k in the mesh, $k = 1, \dots, N$. As the construction of these quantities is entirely local to each element, this stage can be trivially parallelized across the mesh. The local factorization stage is outlined in [Algorithm 4.1](#).

4.2. Global factorization stage. Once local operators have been computed for each element, the solver proceeds to the global factorization stage. Elements (or groups of elements) are merged pairwise in an upward pass according to a set of merge indices \mathcal{T}_ℓ , where $(\alpha, \beta) \in \mathcal{T}_\ell$ indicates that elements \mathcal{E}_α and \mathcal{E}_β should be merged at level ℓ in the hierarchy. Upon merging the two elements \mathcal{E}_α and \mathcal{E}_β , a new parent element $\mathcal{E}_{\alpha\beta} := \mathcal{E}_\alpha \cup \mathcal{E}_\beta$ is created which may be subsequently merged at higher levels in the tree. Thus, a complete specification of the tree is given by the level-ordered set

Algorithm 4.1 Local factorization stage**Input:** Mesh $\{\mathcal{E}_k\}_{k=1}^N$, surface differential operator \mathcal{L}_Γ , righthand side f **Output:** Solution operators $\{S_{\mathcal{E}_k}\}$, Dirichlet-to-Neumann maps $\{\Sigma_{\mathcal{E}_k}\}$, particular solutions $\{\mathbf{v}_k\}$, and particular fluxes $\{\mathbf{v}'_k\}$ for every element $k = 1, \dots, N$.

- 1: Construct interior and boundary index sets $\mathcal{J}_i, \mathcal{J}_b$.
- 2: **for** $k = 1, \dots, N$ **do**
- 3: Evaluate f at the nodes of \mathcal{E}_k to yield \mathbf{f}_k .
- 4: Compute discrete differentiation matrices, $D_x^\Gamma, D_y^\Gamma, D_z^\Gamma$, on \mathcal{E}_k using (2.6).
- 5: Compute discrete operator $L_{\mathcal{E}_k}$ using (2.7).
- 6: Compute the homogeneous solution operator $S_{\mathcal{E}_k}$ such that

$$S_{\mathcal{E}_k}(\mathcal{J}_i, \cdot) = - (L_{\mathcal{E}_k}^{\text{ii}})^{-1} L_{\mathcal{E}_k}^{\text{ib}}, \quad S_{\mathcal{E}_k}(\mathcal{J}_b, \cdot) = I_{n_b}.$$

- 7: Compute the discrete particular solution such that $\mathbf{v}_k^i = (L_{\mathcal{E}_k}^{\text{ii}})^{-1} \mathbf{f}_k^i, \mathbf{v}_k^b = 0$.
- 8: Compute the Dirichlet-to-Neumann map $\Sigma_{\mathcal{E}_k} := D_{\mathcal{E}_k} S_{\mathcal{E}_k}$.
- 9: Compute the flux of the particular solution, $\mathbf{v}'_k := D_{\mathcal{E}_k} \mathbf{v}_k$.
- 10: Reinterpolate boundary grids of $S_{\mathcal{E}_k}, \Sigma_{\mathcal{E}_k}$ according to Remark 3.1.
- 11: **return** $\{S_{\mathcal{E}_k}\}, \{\Sigma_{\mathcal{E}_k}\}, \{\mathbf{v}_k\}, \{\mathbf{v}'_k\}$ for $k = 1, \dots, N$

of merge indices $\{\mathcal{T}_\ell\}_{\ell=0}^L$, with $\ell = 0$ corresponding to the leaf level (containing all the individual elements of the mesh) and $\ell = L$ corresponding to the top level (containing just two elements). For each merge, the interface linear systems are constructed as in subsection 3.2 and inverted directly, yielding interfacial solution operators, interfacial particular solutions, merged Dirichlet-to-Neumann maps, and merged particular fluxes for the parent element. The global factorization stage is outlined in Algorithm 4.2.

Remark 4.1. The Laplace–Beltrami problem $\Delta_\Gamma u = f$ on a closed surface Γ is rank-one deficient, but is uniquely solvable under the mean-zero conditions $\int_\Gamma u = \int_\Gamma f = 0$. In the present scheme this rank deficiency is only seen in the final merge at the top level in the hierarchy, i.e., $\dim(\text{null } A) = 1$ where $A = \Sigma_{\mathcal{E}_\alpha}^{s_\alpha} + \Sigma_{\mathcal{E}_\beta}^{s_\beta}$ is the interface matrix at the top level. We fix this rank deficiency by adding the mean-zero constraint to the top-level linear system, yielding a modified matrix $A + \mathbf{q}\mathbf{q}^T$, where \mathbf{q} is the vector of scaled quadrature weights on the interface. Then $\dim(\text{null}(A + \mathbf{q}\mathbf{q}^T)) = 0$ [69, 47].

4.3. Solve stage. Once the global factorization stage is complete, the algorithm enters the solve stage. For a surface with a boundary, Dirichlet data g is evaluated at the boundary nodes and passed to the top-level solution operator, where it is used to recover the solution on the top-level interface. (For a closed surface, no boundary conditions are needed, and the solution on the top-level interface is simply given by the interfacial particular solution.) This data is passed to the child elements, where this procedure continues recursively in a downward pass until the values of the solution have been tabulated on all element interfaces. The local solution operators computed in the local factorization stage may then be applied to recover the solution in the interior of each element in a parallel fashion. The solve stage is outlined in Algorithm 4.3.

The solve stage may be repeatedly executed with different boundary data g with no need to revisit the local and global factorization stages. Moreover, the particular solutions and particular fluxes throughout the hierarchy may be efficiently updated for different righthand sides f , without the need to recompute solution operators and Dirichlet-to-Neumann maps. If factorizations of the interface linear systems are stored

Algorithm 4.2 Global factorization stage (upward pass)

Input: Mesh $\{\mathcal{E}_k\}_{k=1}^N$, Dirichlet-to-Neumann maps $\{\Sigma_{\mathcal{E}_k}\}_{k=1}^N$, merge indices $\{\mathcal{T}_\ell\}_{\ell=0}^L$

Output: Solution operators $\{S_{\mathcal{I}_{\alpha\beta}}\}$, Dirichlet-to-Neumann maps $\{\Sigma_{\mathcal{E}_{\alpha\beta}}\}$, particular solutions $\{\mathbf{v}_{\mathcal{I}_{\alpha\beta}}\}$, and particular fluxes $\{\mathbf{v}'_{\alpha\beta}\}$ for every merge $(\alpha, \beta) \in \mathcal{T}_\ell$ for every level ℓ

- 1: **for** each level $\ell = 0, 1, \dots, L$ **do**
 - 2: **for** each pair $(\alpha, \beta) \in \mathcal{T}_\ell$ **do**
 - 3: Define the merged domain $\mathcal{E}_{\alpha\beta} := \mathcal{E}_\alpha \cup \mathcal{E}_\beta$.
 - 4: Define the shared interface $\mathcal{I}_{\alpha\beta} := \mathcal{E}_\alpha \cap \mathcal{E}_\beta$.
 - 5: Define indices $\mathcal{J}_{s_\alpha}, \mathcal{J}_{s_\beta}$ for the shared interface nodes on $\mathcal{E}_\alpha, \mathcal{E}_\beta$.
 - 6: Define indices $\mathcal{J}_{b_\alpha}, \mathcal{J}_{b_\beta}$ for the unshared boundary nodes on $\mathcal{E}_\alpha, \mathcal{E}_\beta$.
 - 7: Solve the linear systems (3.14) and (3.13b) for the solution operator $S_{\mathcal{I}_{\alpha\beta}}$ and particular solution $\mathbf{v}_{\mathcal{I}_{\alpha\beta}}$ on the interface.
 - 8: Compute the merged Dirichlet-to-Neumann map $\Sigma_{\mathcal{E}_{\alpha\beta}}$ via (3.15).
 - 9: Compute the merged particular flux $\mathbf{v}'_{\alpha\beta}$ via (3.16).
 - 10: **return** $\{S_{\mathcal{I}_{\alpha\beta}}\}, \{\Sigma_{\mathcal{E}_{\alpha\beta}}\}, \{\mathbf{v}_{\mathcal{I}_{\alpha\beta}}\}, \{\mathbf{v}'_{\alpha\beta}\}$ for all $(\alpha, \beta) \in \mathcal{T}_\ell$ for every level ℓ
-

for every merge, then one need only apply those factorizations (e.g., using forward- and back-substitution) in an upwards pass through the hierarchy in order to update the righthand side f .

Algorithm 4.3 Solve stage (downward pass)

Input: Merged element $\mathcal{E}_{\alpha\beta}$ (or leaf element \mathcal{E}_k), Dirichlet data $\mathbf{g}_{\alpha\beta}$

Output: Solutions $\{\mathbf{u}_k\}_{k=1}^N$ for every element

- 1: **if** \mathcal{E}_k is a leaf **then**
 - 2: Compute the local solution $\mathbf{u}_k := S_{\mathcal{E}_k} \mathbf{g}_k + \mathbf{v}_k$.
 - 3: **return** \mathbf{u}_k
 - 4: **else**
 - 5: Look up the elements $\mathcal{E}_\alpha, \mathcal{E}_\beta$ that were merged to make $\mathcal{E}_{\alpha\beta}$.
 - 6: Look up the shared interface $\mathcal{I}_{\alpha\beta}$.
 - 7: Compute the solution on the shared interface, $\mathbf{u}_{\mathcal{I}_{\alpha\beta}} := S_{\mathcal{I}_{\alpha\beta}} \mathbf{g}_{\alpha\beta} + \mathbf{v}_{\mathcal{I}_{\alpha\beta}}$.
 - 8: Define boundary data on the children, $\mathbf{g}_\alpha, \mathbf{g}_\beta$, with $\mathbf{g}_\alpha^{b_\alpha} := \mathbf{g}_{\alpha\beta}^{b_\alpha}, \mathbf{g}_\beta^{b_\beta} := \mathbf{g}_{\alpha\beta}^{b_\beta}$, and $\mathbf{g}_\alpha^{s_\alpha} = \mathbf{g}_\beta^{s_\beta} := \mathbf{u}_{\mathcal{I}_{\alpha\beta}}$.
 - 9: Recursively compute $\{\mathbf{u}_\alpha\}$ using Algorithm 4.3 with Dirichlet data \mathbf{g}_α .
 - 10: Recursively compute $\{\mathbf{u}_\beta\}$ using Algorithm 4.3 with Dirichlet data \mathbf{g}_β .
 - 11: **return** $\{\mathbf{u}_\alpha\} \cup \{\mathbf{u}_\beta\}$
-

4.4. Computational complexity. The computational complexity of the surface HPS method is identical to that of the two-dimensional HPS scheme on unstructured meshes [32]. We briefly summarize it now.

The total number of degrees of freedom in the order- p mesh $\{\mathcal{E}_k\}_{k=1}^N$ scales as Np^2 . On each element \mathcal{E}_k , construction of the solution operator $S_{\mathcal{E}_k}$ requires the solution of a $\mathcal{O}(p^2) \times \mathcal{O}(p^2)$ linear system with $\mathcal{O}(p)$ righthand sides, which can be computed in $\mathcal{O}(p^6)$ operations. The Dirichlet-to-Neumann map $\Sigma_{\mathcal{E}_k}$ can be computed as a matrix-matrix product in $\mathcal{O}(p^5)$ operations. Therefore, the local factorization

stage (Algorithm 4.1) requires $\mathcal{O}(Np^6)$ operations.

The cost of the global factorization stage depends on the merge order defined by the hierarchical indices $\{\mathcal{T}_\ell\}_{\ell=0}^L$. As in classical nested dissection, the hierarchy should be as balanced as possible so that the indices $\{\mathcal{T}_\ell\}_{\ell=0}^L$ define an approximately balanced binary tree with $L \sim \log N$. A balanced partitioning of the mesh may be automatically computed by conversion to a graph partitioning problem [50]. We assume that the merge indices $\{\mathcal{T}_\ell\}_{\ell=0}^L$ define a hierarchy of $\mathcal{O}(\log N)$ levels, with level ℓ containing $\mathcal{O}(2^{-\ell}N)$ elements. Computing the merged solution operator for a merge in level ℓ requires solving an interface linear system of size $\mathcal{O}(2^{\ell/2}p) \times \mathcal{O}(2^{\ell/2}p)$. Hence, the cost of performing all merges on level ℓ scales as $(2^{-\ell}N) \cdot (2^{\ell/2}p)^3 = 2^{\ell/2}Np^3$. Summing over all $\mathcal{O}(\log N)$ levels yields an overall complexity for the global factorization stage (Algorithm 4.2) of $\mathcal{O}(N^{3/2}p^3)$.

At level ℓ of the solve stage, the solution on the shared interface is recovered via a matrix-vector product with an $\mathcal{O}(2^{\ell/2}p) \times \mathcal{O}(2^{\ell/2}p)$ matrix in $\mathcal{O}((2^{\ell/2}p)^2)$ operations. The cost of computing this product for all interfaces on level ℓ scales as $(2^{-\ell}N) \cdot (2^{\ell/2}p)^2 = Np^2$. Hence, the solution at all interfaces on all levels of the hierarchy can be obtained in $\mathcal{O}(p^2N \log N)$ operations. At the bottom level, the solution is computed in the interior of each element via multiplication with the local solution operator, requiring a total of $\mathcal{O}(Np^3)$ operations. Therefore, the overall cost of the solve stage (Algorithm 4.3) is $\mathcal{O}(p^2N \log N + Np^3)$.

We take a fixed- p perspective throughout this paper, choosing to refine the solution by increasing the number of surface elements N (i.e., h -refinement). While p -refinement and hp -adaptivity are powerful tools for solving PDEs in complex geometries where smoothness criteria can be considered in a localized fashion [32], on discrete surfaces the neighborhood of every point is only approximately smooth, making it difficult to use hp -adaptivity in a localized way. From this perspective then, the overall complexity of the scheme is $\mathcal{O}(N)$ for the local factorization stage, $\mathcal{O}(N^{3/2})$ for the global factorization stage, and $\mathcal{O}(N \log N)$ for the solve stage. Updating the particular solution for a new righthand f takes $\mathcal{O}(N \log N)$ time.

The memory complexity mirrors that of the solve stage, as every level of the hierarchy stores dense solution operators and Dirichlet-to-Neumann maps. Therefore, the memory cost scales as $\mathcal{O}(N \log N)$.

5. Numerical examples. We now demonstrate the convergence and performance of the surface HPS method. The method is implemented in an open-source MATLAB package that provides abstractions for computing with functions on surfaces [31]. Codes for the numerical examples presented below are publicly available [31]. All simulations were run in MATLAB R2021a on a MacBook Pro with a 2.4 GHz 8-core Intel Core i9-9980HK CPU and 64 GB of memory.

5.1. Convergence.

5.1.1. Laplace–Beltrami problem on a smooth surface. We begin with a pure Laplace–Beltrami problem on a smooth surface,

$$\Delta_\Gamma u = f,$$

where u and f are mean-zero functions over Γ , i.e., $\int_\Gamma f(\mathbf{x}) d\mathbf{x} = 0$. To test the convergence of the method, we take Γ to be the unit sphere; an exact solution to the Laplace–Beltrami problem is then given by the spherical harmonic $u(\mathbf{x}) = Y_\ell^m(\mathbf{x})$ when $f(\mathbf{x}) = -\ell(\ell+1)Y_\ell^m(\mathbf{x})$. Figure 5.1 (left) shows the $(\ell, m) = (20, 10)$ spherical harmonic evaluated on a cubed sphere mesh. In Figure 5.1 (right), we measure spatial

convergence to the exact solution under mesh refinement for polynomial orders $p = 4, 8, 12, 16,$ and 20 . As the mean mesh size $h \rightarrow 0$, the L^∞ relative error decreases at a rate of $\mathcal{O}(h^{p-1})$, indicating high-order convergence.

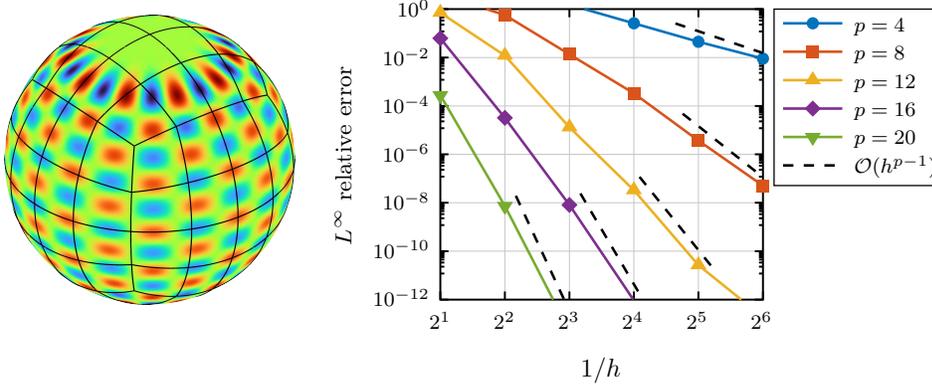


FIG. 5.1. (Left) A high-order cubed sphere mesh is used to approximate a spherical harmonic. (Right) The surface HPS scheme converges at a rate of $\mathcal{O}(h^{p-1})$.

5.1.2. Laplace–Beltrami problem on surfaces with edges and corners.

In the present domain decomposition context, the interface conditions enforced between elements may still be enforced on a surface with sharp edges and corners. However, the binormal vector \mathbf{n}_b along the interface between two elements may no longer be uniquely defined, as the elements on either side of a sharp edge will possess locally different binormal vectors. To remedy this, a natural choice is to enforce that the local fluxes on each element—defined via their respective local binormal vectors—must balance [41].

Just as the solution to Laplace’s equation in a domain with corners may possess weak corner singularities, solutions to the Laplace–Beltrami problem on a non-smooth surface may possess weak vertex singularities. Let Γ be a non-smooth surface with a countable number of sharp vertices—that is, vertices whose conic angle does not equal 2π . Then in a polar neighborhood of vertex i , the solution u to the Laplace–Beltrami problem $\Delta_\Gamma u = f$ scales as $u \sim r^{2\pi/\gamma_i}$, where γ_i is the conic angle of the vertex [41, Lemma 1]. Figure 5.2 shows a self-convergence study of a Laplace–Beltrami problem on the surface of a cube, where $\gamma_i = 3\pi/2$ and $u \sim r^{4/3}$. Hence, $u \in H^2(\Gamma)$ [41, Theorem 4] and second-order convergence is expected [14]. Using a uniform surface mesh, high-order convergence is lost in the presence of the vertex singularities regardless of the polynomial order employed. In this case, a refinement strategy based on hp -adaptivity may be used to recover super-algebraic convergence [45, 32].

Note that in the case of a non-smooth surface Γ with sharp edges but no sharp vertices, the solution to the Laplace–Beltrami problem no longer possesses geometrically-induced corner singularities and high-order convergence may still be obtained. Figure 5.3 shows self-convergence of a Laplace–Beltrami problem on a twisted toroidal surface with sharp edges, where high-order convergence is recovered.

5.2. Performance. To illustrate the computational complexity of the surface HPS method, we measure the runtime and memory consumption of the factorization and solve stages as the number of elements in the mesh N is increased. Figure 5.4 (left) plots the runtime of each stage versus N for the simple Laplace–Beltrami problem

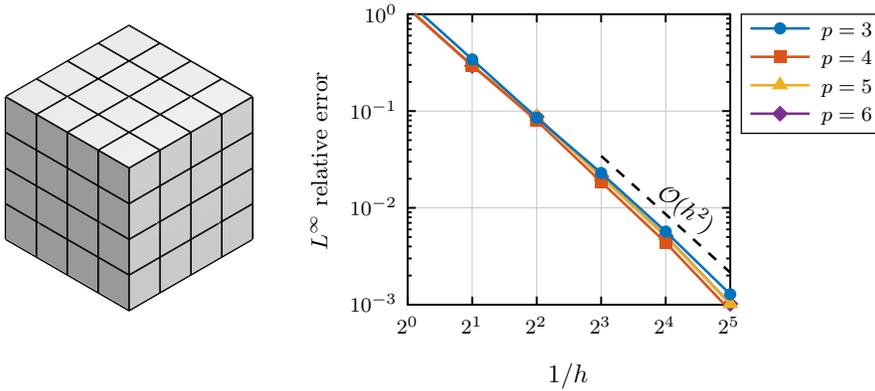


FIG. 5.2. Self-convergence study of the Laplace–Beltrami problem on the surface of a cube. High-order convergence is lost due to the weak vertex singularities induced by the corners of the cube. Polynomial orders $p = 3, 4, 5,$ and 6 all exhibit second-order convergence, as the solution u is only in $H^2(\Gamma)$.

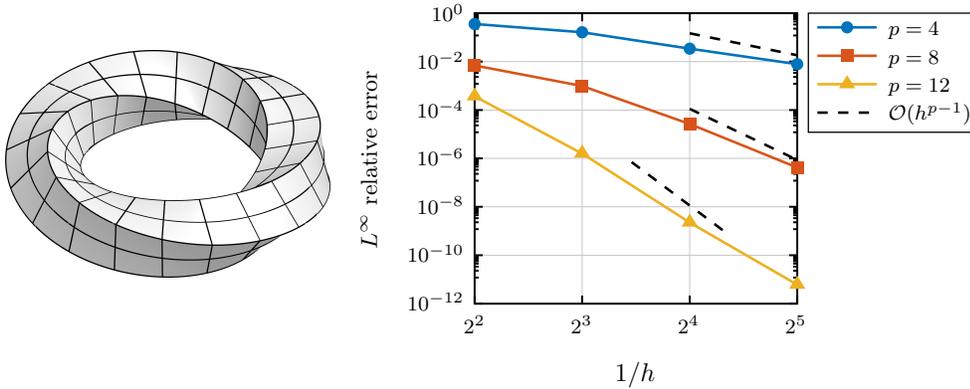


FIG. 5.3. Self-convergence study of the Laplace–Beltrami problem on the surface of a twisted torus. High-order convergence can be obtained in the presence of sharp edges as long as no sharp corners are present.

considered in [subsection 5.1.1](#), with $p = 16$. Although the theoretical complexity of the factorization stage is $\mathcal{O}(N^{3/2})$, here we observe $\mathcal{O}(N)$ scaling even for a problem with over 10^6 degrees of freedom. We expect that for a sufficiently fine mesh, the asymptotic complexity of $\mathcal{O}(N^{3/2})$ will eventually dominate. Also plotted is the runtime complexity of updating the particular solutions given a new righthand side f . [Figure 5.4](#) (right) depicts the memory complexity (i.e., total storage cost) of the method as $N \rightarrow \infty$. Again, while $\mathcal{O}(N \log N)$ memory complexity is expected, in practice we observe $\mathcal{O}(N)$ complexity for moderately-sized problems.

To better place the surface HPS method in the broader context of solvers for surface PDEs, we benchmarked our method against three alternative approaches for solving a reference Laplace–Beltrami problem $\Delta_\Gamma u = f$ on a stellarator geometry [\[35, 55\]](#). We manufacture a righthand side f so that the solution u is the Coulombic potential induced by a unit charge located at the off-surface point $(0, 0, 3)$. The solution is visualized in [Figure 5.5](#) (left). We compare the surface HPS approach to: a high-order finite element method discretized using the MFEM [\[3\]](#) library and solved used

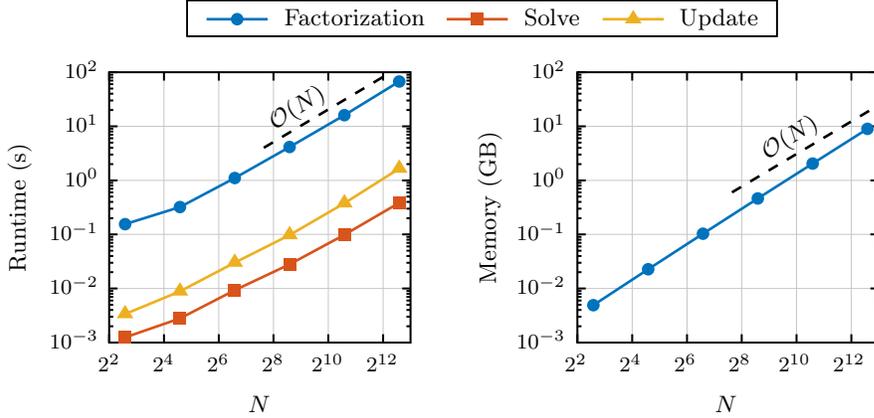


FIG. 5.4. Computational complexity of the surface HPS method. (Left) The runtime of the factorization stage is observed to scale as $\mathcal{O}(N)$, even for a problem with 10^6 degrees of freedom. For large enough problem sizes, the asymptotic scaling $\mathcal{O}(N^{3/2})$ will eventually dominate. (Right) The total memory required to store all operators in the surface HPS method is also observed to scale linearly in this regime.

UMFPACK [19]; a pseudospectral method preconditioned by the flat Laplacian, which can be applied fast using the fast Fourier Transform (FFT) and is specifically designed for use on genus-one surfaces [47]; and a layer-potential preconditioner [60] discretized using spectrally-accurate quadratures [55]. For each method, we perform a sequence of mesh refinements and record both the runtime and the L^∞ relative error between the computed and reference solutions. As the surface HPS method and the high-order finite element method are both based on unstructured meshes, we use the same sequence of mesh refinements for each, with $p = 20$; for the spectral methods, we refine the number of grid points on the surface globally. Figure 5.5 plots the relative error versus the runtime for all the methods. While the FFT-based method is dominant for lower accuracies, the surface HPS method starts to narrow the gap around a relative accuracy of 10^{-5} . However, the FFT-based approach is only applicable on toroidal surfaces of genus one, whereas the other approaches can be used on general surfaces. The high-order finite element method is competitive for lower orders, but when run at 20th order (shown here), it performs about 10 times slower than the surface HPS scheme for a given relative accuracy.

As both the surface HPS method and the high-order finite element method can use the same underlying high-order mesh, we further compare their runtimes using different polynomial degrees. Figure 5.6 shows the runtime of both methods plotted under mesh refinement (left) using a fixed polynomial order of $p = 20$, and p -refinement (right) using a fixed mesh with $N = 2048$ elements. While the asymptotic complexity of UMFPACK matches that of the surface HPS scheme for a fixed p , the gap between the methods widens as $p \rightarrow \infty$. Finally, we note that while MFEM and UMFPACK are high-performance compiled codes that have been heavily optimized, our MATLAB implementation of the surface HPS scheme is relatively unoptimized.

5.3. Hodge decomposition of a vector field. The Laplace–Beltrami problem arises when computing the Hodge decomposition of tangential vector fields. For a vector field \mathbf{f} tangent to the surface Γ , the Hodge decomposition [65] writes \mathbf{f} as the

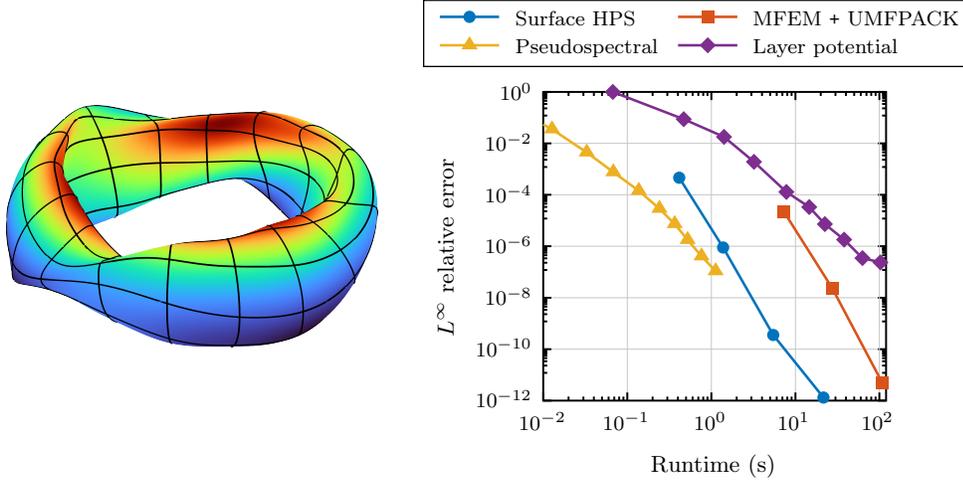


FIG. 5.5. Relative accuracy versus runtime of the surface HPS compared to a high-order finite element method using MFEM [3] and UMFPACK [19], an FFT-based spectral method [47], and a layer potential approach [60]. We use $p = 20$ for both the surface HPS method and the finite element method. The solution to the benchmark problem is visualized on the left.

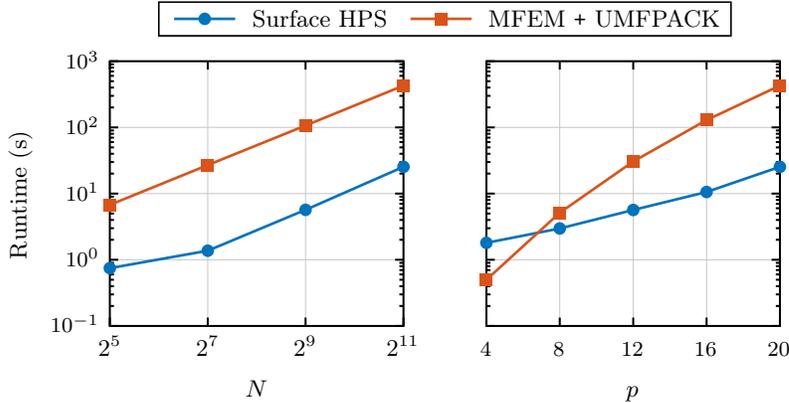


FIG. 5.6. Runtime comparison of the surface HPS solver and the UMFPACK [19] solver applied to a high-order finite element discretization from the finite element package MFEM [3]. The runtime of each method is plotted versus (left) number of elements N under mesh refinement with a fixed polynomial order of $p = 20$, and (right) polynomial order p under p -refinement with a fixed number of elements $N = 2048$.

sum of curl-free, divergence-free, and harmonic components,

$$\mathbf{f} = \underbrace{\nabla_{\Gamma} u}_{\text{curl-free}} + \underbrace{\mathbf{n} \times \nabla_{\Gamma} v}_{\text{divergence-free}} + \underbrace{\mathbf{w}}_{\text{harmonic}},$$

where u and v are scalar functions on Γ and \mathbf{w} is a harmonic vector field, i.e.,

$$\nabla_{\Gamma} \cdot \mathbf{w} = 0, \quad \nabla_{\Gamma} \cdot (\mathbf{n} \times \mathbf{w}) = 0.$$

Such vector fields play an important role in integral-equation-based methods for computational electromagnetics [30, 55]. To numerically compute such a decomposition,

one may solve two Laplace–Beltrami problems for u and v ,

$$\Delta_{\Gamma}u = \nabla_{\Gamma} \cdot \mathbf{f} \quad \text{and} \quad \Delta_{\Gamma}v = -\nabla_{\Gamma} \cdot (\mathbf{n} \times \mathbf{f}),$$

and then set $\mathbf{w} = \mathbf{f} - \nabla_{\Gamma}u - \mathbf{n} \times \nabla_{\Gamma}v$.

Figure 5.7 shows the numerical Hodge decomposition of a random smooth vector field tangent to a deformed toroidal surface, computed on the 16th-order mesh depicted at the top, possessing roughly 200,000 degrees of freedom. The computation took 3.7 seconds. The residuals of the harmonic component are $\|\nabla_{\Gamma} \cdot \mathbf{w}\|_2 \approx 4 \times 10^{-7}$ and $\|\nabla_{\Gamma} \cdot (\mathbf{n} \times \mathbf{w})\|_2 \approx 2 \times 10^{-6}$.

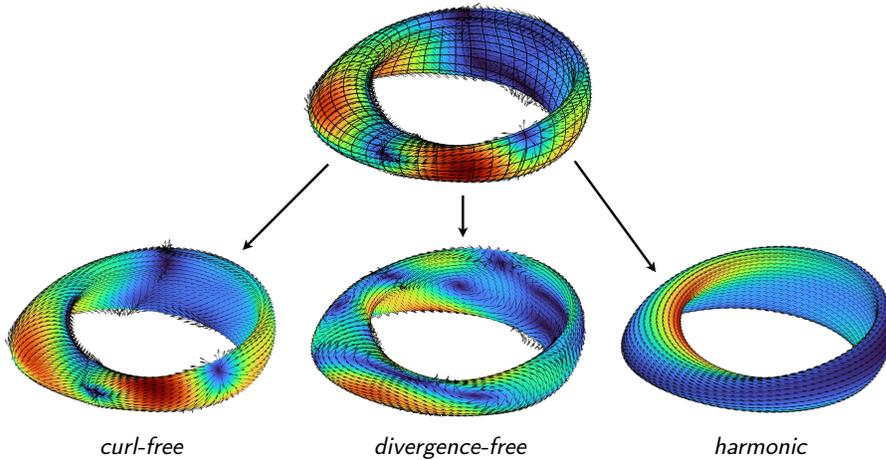


FIG. 5.7. The Hodge decomposition of a random smooth vector field tangent to a deformed toroidal surface is numerically computed using a 16th-order mesh possessing roughly 200,000 degrees of freedom. The computation of the decomposition took 3.7 seconds. The resulting harmonic component \mathbf{w} has residuals $\|\nabla_{\Gamma} \cdot \mathbf{w}\|_2 \approx 4 \times 10^{-7}$ and $\|\nabla_{\Gamma} \cdot (\mathbf{n} \times \mathbf{w})\|_2 \approx 2 \times 10^{-6}$. Colors indicate magnitude.

5.4. Timestepping. Consider a nonlinear time-dependent PDE of the form

$$\frac{\partial u}{\partial t} = \mathcal{L}_{\Gamma}u + \mathcal{N}(u),$$

with \mathcal{L}_{Γ} a linear surface differential operator and \mathcal{N} a nonlinear, non-differential operator. Such PDEs appear as models for reaction–diffusion systems, where \mathcal{L}_{Γ} contains the diffusion terms and \mathcal{N} the reaction terms. As the timescales of diffusion and reaction are often orders of magnitude different, explicit timestepping schemes can suffer from a severe time step restriction. Implicit or semi-implicit timestepping schemes can alleviate stability issues, e.g., by treating the diffusion term \mathcal{L}_{Γ} implicitly.

We use the implicit–explicit backward differentiation formula (IMEX-BDF) family of schemes [4], which are based on the backward differentiation formulae for the implicit term and the Adams–Bashforth for the explicit term. Fix a time step $\Delta t > 0$ and let $u^k(\mathbf{x}) \approx u(\mathbf{x}, k\Delta t)$ denote the approximate solution at time step k . Discretizing in time with the K th-order IMEX-BDF scheme results in a steady-state problem at each time step of the form

$$(5.1) \quad (I - \omega \Delta t \mathcal{L}_{\Gamma}) u^{k+1} = \sum_{i=0}^{K-1} \mu_i u^{k-i} + \Delta t \sum_{i=0}^{K-1} \nu_i \mathcal{N}(u^{k-i}),$$

where ω , μ_i , and ν_i are constants given by

$$\begin{aligned} \text{IMEX-BDF1: } & \omega = 1, & \mu = 1, & \nu = 1, \\ \text{IMEX-BDF2: } & \omega = \frac{2}{3}, & \mu = \left(\frac{4}{3}, -\frac{1}{3}\right), & \nu = \left(\frac{4}{3}, -\frac{2}{3}\right), \\ \text{IMEX-BDF3: } & \omega = \frac{6}{11}, & \mu = \left(\frac{18}{11}, -\frac{9}{11}, \frac{2}{11}\right), & \nu = \left(\frac{18}{11}, -\frac{18}{11}, \frac{6}{11}\right), \\ \text{IMEX-BDF4: } & \omega = \frac{12}{25}, & \mu = \left(\frac{48}{25}, -\frac{36}{25}, \frac{16}{25}, -\frac{3}{25}\right), & \nu = \left(\frac{48}{25}, -\frac{72}{25}, \frac{48}{25}, -\frac{12}{25}\right). \end{aligned}$$

For a fixed surface Γ , fixed time step Δt , and fixed linear operator \mathcal{L}_Γ , the inverse $(I - \omega\Delta t\mathcal{L}_\Gamma)^{-1}$ can be precomputed once for all time steps and applied fast using the HPS scheme, yielding a cost per time of $\mathcal{O}(N \log N)$.

We now apply the IMEX-BDF methods to simulate time-dependent reaction–diffusion on surfaces. We choose three surface meshes of varying complexity: a blob shape defined through deformation of a sphere, a stellarator shape [35] with localized regions of high curvature, and a cow shape defined via a subdivision surface [18] and meshed to high order using Rhinoceros [62]. Figure 5.9 depicts these meshes in the left column. On all three meshes, we use 16th-order elements to represent the solution. We perform a temporal self-convergence study on the cow mesh for the reaction–diffusion problem described in subsection 5.4.2. Using a reference solution computed with a time step of $\Delta t = 2^{-10}$, we simulate to a final time of $t = 1$ over a range of time steps. Figure 5.8 depicts the results, demonstrating that k th-order accuracy is achieved for the IMEX-BDF k method.

In all cases, the overall cost per time step is 0.05 seconds for the blob mesh with 24,576 degrees of freedom (0.04 s for updating the particular solution and 0.01 s for the solve), 0.1 seconds for the stellarator mesh with 49,152 degrees of freedom (0.07 s for updating the particular solution and 0.03 s for the solve), and 0.18 seconds for the cow mesh with 86,784 degrees of freedom (0.13 s for updating the particular solution and 0.05 s for the solve).

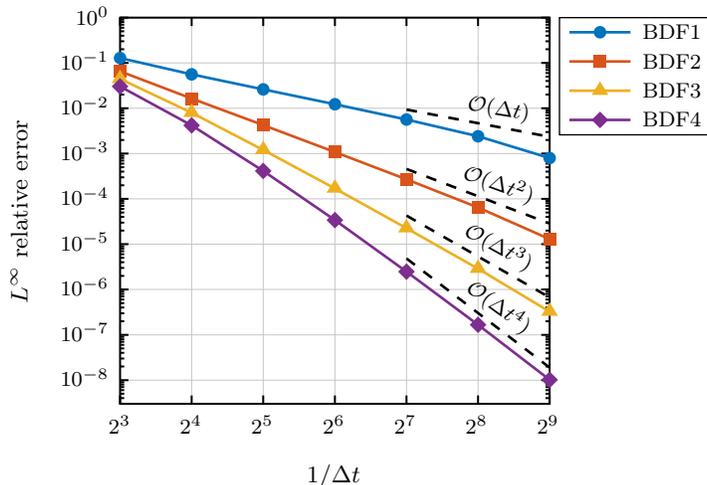


FIG. 5.8. Self-convergence study of the reaction–diffusion system (5.2) on the cow model over a range of time steps. A reference solution is computed using a time step of $\Delta t = 2^{-10}$ to a final time of $t = 1$.

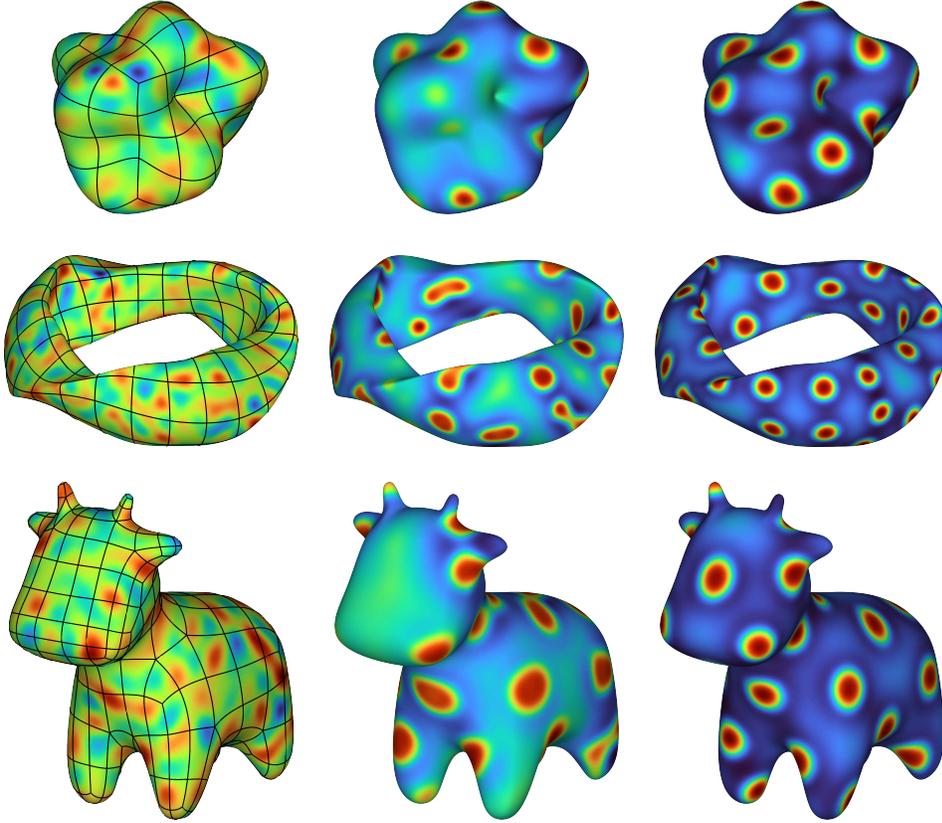


FIG. 5.9. Snapshots of a two-species Turing system simulated on three surfaces at times $t = 0$, $t = 20$, and $t = 200$. We discretize in time with a fourth-order IMEX-BDF4 scheme and precompute a fast direct solver for the implicit problem, which computes the solution at each time step in $\mathcal{O}(N \log N)$ operations.

5.4.1. Turing system. We first consider a two-species reaction–diffusion system on a surface Γ given by [10],

$$\begin{aligned} \frac{\partial u}{\partial t} &= \delta_u \Delta_{\Gamma} u + \alpha u (1 - \tau_1 v^2) + v (1 - \tau_2 u), \\ \frac{\partial v}{\partial t} &= \delta_v \Delta_{\Gamma} v + \beta v \left(1 + \frac{\alpha \tau_1}{\beta} uv \right) + u (\gamma + \tau_2 v). \end{aligned}$$

Solutions u and v to this system can exhibit Turing patterns—namely, spots and stripes—depending on the choice of parameters δ_u , δ_v , α , β , γ , τ_1 , and τ_2 . Here, we take $\alpha = 0.899$, $\beta = -0.91$, $\gamma = -0.899$, $\tau_1 = 0.02$, and $\delta_u = 0.516\delta_v$. On the blob, we take $\delta_v = 0.005$ and $\tau_2 = 0.15$. On the stellarator, we take $\delta_v = 0.02$ and $\tau_2 = 0.2$. On the cow, we take $\delta_v = 0.001$ and $\tau_2 = 0.2$. We simulate this system on all three surface meshes for 2000 time steps until a final time of $t = 200$ using a time step size of $\Delta t = 0.1$. Snapshots of the solutions at times $t = 0$, $t = 20$, and $t = 200$ are shown in Figure 5.9.

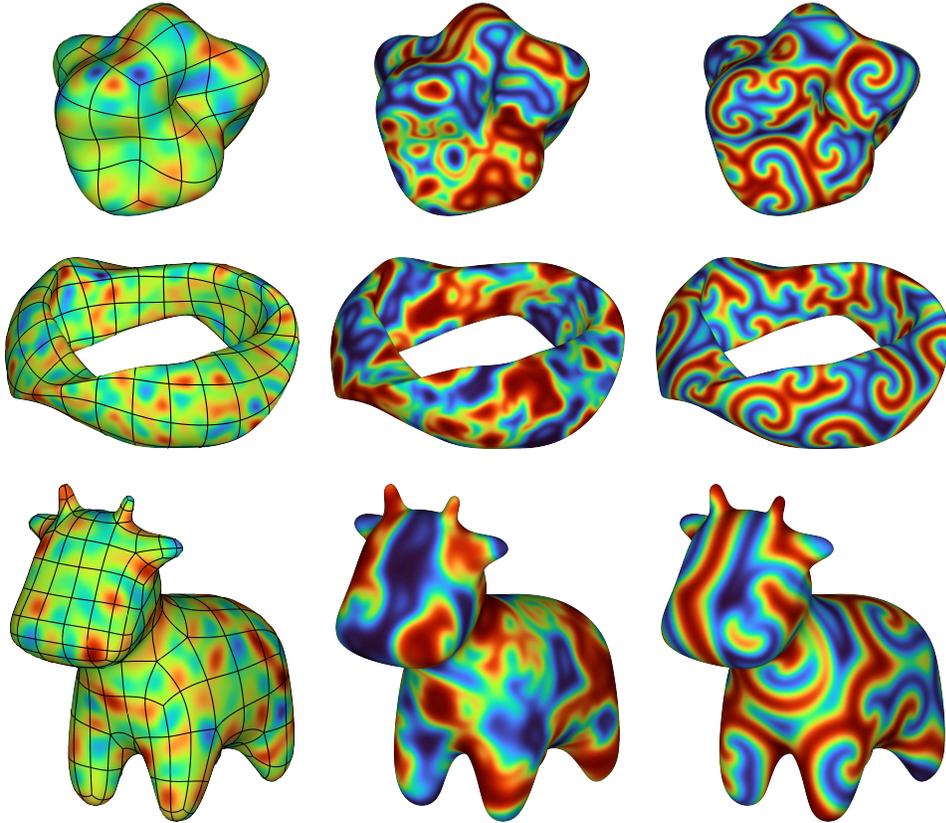


FIG. 5.10. Snapshots of the complex Ginzburg–Landau equation simulated on three surfaces at times $t = 0$, $t = 6$, and $t = 60$. We discretize in time with a fourth-order IMEX-BDF₄ scheme and precompute a fast direct solver for the implicit problem, which computes the solution at each time step in $\mathcal{O}(N \log N)$ operations.

5.4.2. Complex Ginzburg–Landau. The complex Ginzburg–Landau equation on a surface Γ is given by

$$(5.2) \quad \frac{\partial u}{\partial t} = \delta(1 + \alpha i)\Delta_{\Gamma}u + u - (1 + \beta i)u|u|^2,$$

where $u = u(\mathbf{x}, t) \in \mathbb{C}$ and α , β , and δ are parameters. On the blob, we take $\alpha = 0$, $\beta = -1.5$, and $\delta = 10^{-3}$. On the stellarator, we take $\alpha = 0$, $\beta = 1.5$, and $\delta = 10^{-2}$. On the cow, we take $\alpha = 0$, $\beta = 1.5$, and $\delta = 5 \cdot 10^{-4}$. We simulate this system on all three surface meshes for 2000 time steps until a final time of $t = 60$ using a time step size of $\Delta t = 0.03$. Snapshots of the solutions at times $t = 0$, $t = 6$, and $t = 60$ are shown in [Figure 5.10](#).

6. Conclusions and future work. We have presented a high-order accurate, fast direct solver for surface PDEs. The method is competitive with state-of-the-art solvers for elliptic PDEs on surfaces and can provide significant speedups when high polynomial degrees are employed. The precomputed factorizations stored by the solver can be used to accelerate implicit time-stepping schemes, allowing for long-time simulations with a low cost per time step.

Many avenues exist for extending the solver. Although we have restricted our presentation to second-order elliptic surface PDEs, the generalization to high-order differential operators (e.g., the surface biharmonic operator) is straightforward and simply requires the use of a different Poincaré–Steklov operator to impose higher-order continuity between patches. We also hope to extend the method to vector-valued PDEs, with the goal of solving the surface Stokes equations. Evolving and deformable surfaces pose an interesting challenge for the solver, as one factorization can no longer be used to accelerate all time steps. Instead, we plan to investigate the use of the fast direct solver as a preconditioner for nearby surfaces. Finally, we hope to couple this on-surface solver to a full bulk solver to accurately simulate coupled bulk–diffusion processes.

Acknowledgments. We have greatly benefited from conversations with Leslie Greengard, Alex Barnett, Manas Rachh, Dhairya Malhotra, Mike O’Neil, Charlie Epstein, and Mengjian Hua. We thank Pearson Miller and Stas Shvartsman for sparking our interest in the development of fast methods for surface PDEs. Keenan Crane created the cow model used in [Figures 5.9](#) and [5.10](#).

REFERENCES

- [1] J. H. ADLER, I. LASHUK, AND S. P. MACLACHLAN, *Composite-grid multigrid for diffusion on the sphere*, Numer. Linear Algebra Appl., 25 (2018), e2115 (16 pages), <https://doi.org/10.1002/nla.2115>.
- [2] D. AGARWAL, M. O’NEIL, AND M. RACHH, *FMM-accelerated solvers for the Laplace–Beltrami problem on complex surfaces in three dimensions*, Nov. 2021, <https://arxiv.org/abs/2111.10743>.
- [3] R. ANDERSON, J. ANDREJ, A. BARKER, J. BRAMWELL, J.-S. CAMIER, J. CERVENY, V. DOBREV, Y. DUDOUIT, A. FISHER, T. KOLEV, W. PAZNER, M. STOWELL, V. TOMOV, I. AKKERMAN, J. DAHM, D. MEDINA, AND S. ZAMPINI, *MFEM: A modular finite element methods library*, Comput. Math. Appl., 81 (2021), pp. 42–74, <https://doi.org/10.1016/j.camwa.2020.06.009>.
- [4] U. M. ASCHER, S. J. RUUTH, AND B. T. R. WETTON, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 797–823.
- [5] T. ASKHAM AND A. J. CERFON, *An adaptive fast multipole accelerated Poisson solver for complex geometries*, J. Comput. Phys., 344 (2017), pp. 1–22, <https://doi.org/10.1016/j.jcp.2017.04.063>.
- [6] C. V. ATANASIU, A. H. BOOZER, L. E. ZAKHAROV, A. A. SUBBOTIN, AND G. I. MIRON, *Determination of the vacuum field resulting from the perturbation of a toroidally symmetric plasma*, Phys. Plasmas, 6 (1999), pp. 2781–2790, <https://doi.org/10.1063/1.873235>.
- [7] T. BABB, A. GILLMAN, S. HAO, AND P.-G. MARTINSSON, *An accelerated Poisson solver based on multidomain spectral discretization*, BIT Numer. Math., 58 (2018), pp. 851–879, <https://doi.org/10.1007/s10543-018-0714-0>.
- [8] T. BABB, P.-G. MARTINSSON, AND D. APPELÖ, *HPS accelerated spectral solvers for time dependent problems: Part I, Algorithms*, in Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2018, S. J. Sherwin, D. Moxey, J. Peiró, P. E. Vincent, and C. Schwab, eds., vol. 134, Springer International Publishing, Cham, 2020, pp. 131–141, https://doi.org/10.1007/978-3-030-39647-3_9.
- [9] T. BABB, P.-G. MARTINSSON, AND D. APPELÖ, *HPS accelerated spectral solvers for time dependent problems: Part II, Numerical experiments*, in Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2018, S. J. Sherwin, D. Moxey, J. Peiró, P. E. Vincent, and C. Schwab, eds., vol. 134, Springer International Publishing, Cham, 2020, pp. 155–166, https://doi.org/10.1007/978-3-030-39647-3_11.
- [10] R. A. BARRIO, C. VAREA, J. L. ARAGÓN, AND P. K. MAINI, *A two-dimensional numerical study of spatial pattern formation in interacting Turing systems*, Bull Math. Biol., 61 (1999), pp. 483–505, <https://doi.org/10.1006/bulm.1998.0093>.
- [11] M. BERTALMÍO, L.-T. CHENG, S. OSHER, AND G. SAPIRO, *Variational Problems and Partial Differential Equations on Implicit Surfaces*, J. Comput. Phys., 174 (2001), pp. 759–780, <https://doi.org/10.1006/jcph.2001.6937>.

- [12] A. BONITO AND J. E. PASCIAK, *Convergence analysis of variational and non-variational multigrid algorithms for the Laplace-Beltrami operator*, Math. Comput., 81 (2012), pp. 1263–1288, <https://doi.org/10.1090/S0025-5718-2011-02551-2>.
- [13] J. P. BOYD, *Chebyshev and Fourier Spectral Methods*, Dover Publications, Mineola, New York, 2nd ed., 2001.
- [14] S. C. BRENNER AND L. R. SCOTT, *The Mathematical Theory of Finite Element Methods*, vol. 15 of Texts in Applied Mathematics, Springer New York, New York, NY, 2008, <https://doi.org/10.1007/978-0-387-75934-0>.
- [15] M. BURGER, *Finite element approximation of elliptic partial differential equations on implicit surfaces*, Comput. Visual Sci., 12 (2009), pp. 87–100, <https://doi.org/10.1007/s00791-007-0081-x>.
- [16] Y. CHEN AND C. B. MACDONALD, *The closest point method and multigrid solvers for elliptic equations on surfaces*, SIAM J. Sci. Comput., 37 (2015), pp. A134–A155, <https://doi.org/10.1137/130929497>.
- [17] D. COLTON AND R. KRESS, *Inverse Acoustic and Electromagnetic Scattering Theory*, vol. 93 of Applied Mathematical Sciences, Springer, New York, 2013, <https://doi.org/10.1007/978-1-4614-4942-3>.
- [18] K. CRANE, *Keenan's 3D model repository*, <http://www.cs.cmu.edu/~kmc Crane/Projects/ModelRepository/>.
- [19] T. A. DAVIS, *Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method*, ACM Trans. Math. Softw., 30 (2004), pp. 196–199, <https://doi.org/10.1145/992200.992206>.
- [20] A. DEMLOW, *Higher-order finite element methods and pointwise error estimates for elliptic problems on surfaces*, SIAM J. Numer. Anal., 47 (2009), pp. 805–827, <https://doi.org/10.1137/070708135>.
- [21] R. DIEGMILLER, H. MONTANELLI, C. B. MURATOV, AND S. Y. SHVARTSMAN, *Spherical caps in cell polarization*, Biophys. J., 115 (2018), pp. 26–30, <https://doi.org/10.1016/j.bpj.2018.05.033>.
- [22] T. A. DRISCOLL AND N. HALE, *Rectangular spectral collocation*, IMA J. Numer. Anal., (2015), p. dru062, <https://doi.org/10.1093/imanum/dru062>.
- [23] G. DZIUK, *Finite elements for the Beltrami operator on arbitrary surfaces*, in Partial Differential Equations and Calculus of Variations, S. Hildebrandt and R. Leis, eds., Lecture Notes in Mathematics, Springer, Berlin, Heidelberg, 1988, pp. 142–155, <https://doi.org/10.1007/BFb0082865>.
- [24] G. DZIUK AND C. M. ELLIOTT, *Finite elements on evolving surfaces*, IMA J. Numer. Anal., 27 (2007), pp. 262–292, <https://doi.org/10.1093/imanum/drl023>.
- [25] G. DZIUK AND C. M. ELLIOTT, *Surface finite elements for parabolic equations*, J. Comput. Math., 25 (2007), pp. 385–407.
- [26] G. DZIUK AND C. M. ELLIOTT, *Eulerian finite element method for parabolic PDEs on implicit surfaces*, Interfaces Free Boundaries, 10 (2008), pp. 119–138, <https://doi.org/10.4171/ifb/182>.
- [27] G. DZIUK AND C. M. ELLIOTT, *Finite element methods for surface PDEs*, Acta Numer., 22 (2013), pp. 289–396, <https://doi.org/10.1017/S0962492913000056>.
- [28] C. L. EPSTEIN AND L. GREENGARD, *Debye sources and the numerical solution of the time harmonic Maxwell equations*, Commun. Pure Appl. Math., 63 (2010), pp. 413–463, <https://doi.org/10.1002/cpa.20313>.
- [29] C. L. EPSTEIN, L. GREENGARD, AND M. O'NEIL, *Debye sources and the numerical solution of the time harmonic Maxwell equations II*, Commun. Pure Appl. Math., 66 (2013), pp. 753–789, <https://doi.org/10.1002/cpa.21420>.
- [30] C. L. EPSTEIN, L. GREENGARD, AND M. O'NEIL, *Debye sources, Beltrami fields, and a complex structure on Maxwell fields*, Commun. Pure Appl. Math., 68 (2015), pp. 2237–2280, <https://doi.org/10.1002/cpa.21560>.
- [31] D. FORTUNATO, *GitHub repository*, 2022. <https://github.com/danfortunato/surface-hps>.
- [32] D. FORTUNATO, N. HALE, AND A. TOWNSEND, *The ultraspherical spectral element method*, J. Comput. Phys., 436 (2021), p. 110087, <https://doi.org/10.1016/j.jcp.2020.110087>.
- [33] T. FRANKEL, *The Geometry of Physics: An Introduction*, Cambridge University Press, New York, 3rd ed ed., 2012.
- [34] E. J. FUSELIER AND G. B. WRIGHT, *A high-order kernel method for diffusion and reaction-diffusion equations on surfaces*, J. Sci. Comput., 56 (2013), pp. 535–565, <https://doi.org/10.1007/s10915-013-9688-x>.
- [35] P. R. GARABEDIAN, *Three-dimensional codes to design stellarators*, Phys. Plasmas, 9 (2002), pp. 137–149, <https://doi.org/10.1063/1.1419252>.
- [36] P. R. GARABEDIAN, *Three-dimensional analysis of tokamaks and stellarators*, Proc. Natl. Acad.

- Sci. U.S.A., 105 (2008), pp. 13716–13719, <https://doi.org/10.1073/pnas.0806354105>.
- [37] A. GEORGE, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363, <https://doi.org/10.1137/0710032>.
- [38] C. GEUZAIN AND J.-F. REMACLE, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, Int. J. Numer. Methods Eng., 79 (2009), pp. 1309–1331, <https://doi.org/10.1002/nme.2579>.
- [39] A. GILLMAN, A. H. BARNETT, AND P.-G. MARTINSSON, *A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media*, BIT Numer. Math., 55 (2015), pp. 141–170, <https://doi.org/10.1007/s10543-014-0499-8>.
- [40] A. GILLMAN AND P.-G. MARTINSSON, *A direct solver with $O(N)$ complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method*, SIAM J. Sci. Comput., 36 (2014), pp. A2023–A2046, <https://doi.org/10.1137/130918988>.
- [41] T. GOODWILL AND M. O’NEIL, *On the numerical solution of the Laplace-Beltrami problem on piecewise-smooth surfaces*, Aug. 2021, <https://arxiv.org/abs/2108.08959>.
- [42] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348, [https://doi.org/10.1016/0021-9991\(87\)90140-9](https://doi.org/10.1016/0021-9991(87)90140-9).
- [43] J. B. GREER, *An improvement of a recent Eulerian method for solving PDEs on general geometries*, J. Sci. Comput., 29 (2006), pp. 321–352, <https://doi.org/10.1007/s10915-005-9012-5>.
- [44] J. B. GREER, A. L. BERTOZZI, AND G. SAPIRO, *Fourth order partial differential equations on general geometries*, Journal of Computational Physics, 216 (2006), pp. 216–246, <https://doi.org/10.1016/j.jcp.2005.11.031>.
- [45] B. GUO AND I. BABUŠKA, *The h-p version of the finite element method*, Comput. Mech., 1 (1986), pp. 21–41, <https://doi.org/10.1007/BF00298636>.
- [46] T. J. R. HUGHES, J. A. COTTRELL, AND Y. BAZILEVS, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Comput. Methods Appl. Mech. Eng., 194 (2005), pp. 4135–4195, <https://doi.org/10.1016/j.cma.2004.10.008>.
- [47] L.-M. IMBERT-GÉRARD AND L. GREENGARD, *Pseudo-spectral methods for the Laplace–Beltrami equation and the Hodge decomposition on surfaces of genus one*, Numer. Methods Partial Differential Equations, 33 (2017), pp. 941–955, <https://doi.org/10.1002/num.22131>.
- [48] D. JEONG, Y. LI, Y. CHOI, M. YOO, D. KANG, J. PARK, J. CHOI, AND J. KIM, *Numerical simulation of the zebra pattern formation on a three-dimensional model*, Phys. A, 475 (2017), pp. 106–116, <https://doi.org/10.1016/j.physa.2017.02.014>.
- [49] T. JONSSON, M. G. LARSON, AND K. LARSSON, *Cut finite element methods for elliptic problems on multipatch parametric surfaces*, Comput. Methods Appl. Mech. Eng., 324 (2017), pp. 366–394, <https://doi.org/10.1016/j.cma.2017.06.018>.
- [50] G. KARYPIS AND V. KUMAR, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392, <https://doi.org/10.1137/S1064827595287997>.
- [51] C. LANDSBERG AND A. VOIGT, *A multigrid finite element method for reaction-diffusion systems on surfaces*, Comput. Visual Sci., 13 (2010), pp. 177–185, <https://doi.org/10.1007/s00791-010-0136-2>.
- [52] E. LEHTO, V. SHANKAR, AND G. B. WRIGHT, *A radial basis function (RBF) compact finite difference (FD) scheme for reaction-diffusion equations on surfaces*, SIAM J. Sci. Comput., 39 (2017), pp. A2129–A2151, <https://doi.org/10.1137/16M1095457>.
- [53] C. B. MACDONALD, B. MERRIMAN, AND S. J. RUUTH, *Simple computation of reaction-diffusion processes on point clouds*, Proc. Natl. Acad. Sci. U.S.A., 110 (2013), pp. 9209–9214, <https://doi.org/10.1073/pnas.1221408110>.
- [54] C. B. MACDONALD AND S. J. RUUTH, *The implicit closest point method for the numerical solution of partial differential equations on surfaces*, SIAM J. Sci. Comput., 31 (2010), pp. 4330–4350, <https://doi.org/10.1137/080740003>.
- [55] D. MALHOTRA, A. CERFON, L.-M. IMBERT-GÉRARD, AND M. O’NEIL, *Taylor states in stellarators: A fast high-order boundary integral solver*, J. Comput. Phys., 397 (2019), 108791 (24 pages), <https://doi.org/10.1016/j.jcp.2019.06.067>.
- [56] P. MARTINSSON, *A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method*, J. Comput. Phys., 242 (2013), pp. 460–479, <https://doi.org/10.1016/j.jcp.2013.02.019>.
- [57] P.-G. MARTINSSON, *Fast Direct Solvers for Elliptic PDEs*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, 2019, <https://doi.org/10.1137/1.9781611976045>.
- [58] P. W. MILLER, D. FORTUNATO, C. MURATOV, L. GREENGARD, AND S. SHVARTSMAN, *Forced and spontaneous symmetry breaking in cell polarization*, Nat. Comput. Sci., 2 (2022), pp. 504–511, <https://doi.org/10.1038/s43588-022-00295-0>.
- [59] P. W. MILLER, N. STOOP, AND J. DUNKEL, *Geometry of wave propagation on active de-*

- formable surfaces*, Phys. Rev. Lett., 120 (2018), 268001 (6 pages), <https://doi.org/10.1103/PhysRevLett.120.268001>.
- [60] M. O'NEIL, *Second-kind integral equations for the Laplace–Beltrami problem on surfaces in three dimensions*, Adv. Comput. Math., 44 (2018), pp. 1385–1409, <https://doi.org/10.1007/s10444-018-9587-7>.
- [61] C. PIRET, *The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces*, J. Comput. Phys., 231 (2012), pp. 4662–4675, <https://doi.org/10.1016/j.jcp.2012.03.007>.
- [62] ROBERT MCNEEL & ASSOCIATES, *Rhinoceros 3D*, 2022, <https://www.rhino3d.com/>. Version 7.
- [63] S. J. RUUTH AND B. MERRIMAN, *A simple embedding method for solving partial differential equations on surfaces*, J. Comput. Phys., 227 (2008), pp. 1943–1961, <https://doi.org/10.1016/j.jcp.2007.10.009>.
- [64] R. I. SAYE AND J. A. SETHIAN, *Multiscale modelling of evolving foams*, J. Comput. Phys., 315 (2016), pp. 273–301, <https://doi.org/10.1016/j.jcp.2016.02.077>.
- [65] G. SCHWARZ, *Hodge Decomposition - A Method for Solving Boundary Value Problems*, no. 1607 in Lecture Notes in Mathematics, Springer, Berlin, 1995.
- [66] J. A. SETHIAN, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 2nd ed., 1999.
- [67] V. SHANKAR, G. B. WRIGHT, R. M. KIRBY, AND A. L. FOGELSON, *A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction–diffusion equations on surfaces*, J. Sci. Comput., 63 (2015), pp. 745–768, <https://doi.org/10.1007/s10915-014-9914-1>.
- [68] V. SHANKAR, G. B. WRIGHT, AND A. NARAYAN, *A robust hyperviscosity formulation for stable RBF-FD discretizations of advection-diffusion-reaction equations on manifolds*, SIAM J. Sci. Comput., 42 (2020), pp. A2371–A2401, <https://doi.org/10.1137/19M1288747>.
- [69] J. SIFUENTES, Z. GIMBUTAS, AND L. GREENGARD, *Randomized methods for rank-deficient linear systems*, Electron. Trans. Numer. Anal., 44 (2015), pp. 177–188.
- [70] N. STOOP, R. LAGRANGE, D. TERWAGNE, P. M. REIS, AND J. DUNKEL, *Curvature-induced symmetry breaking determines elastic surface patterns*, Nat. Mater., 14 (2015), pp. 337–342, <https://doi.org/10.1038/nmat4202>.
- [71] L. N. TREFETHEN, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000, <https://doi.org/10.1137/1.9780898719598>.
- [72] S. K. VEERAPANENI, A. RAHIMIAN, G. BIROS, AND D. ZORIN, *A fast algorithm for simulating vesicle flows in three dimensions*, J. Comput. Phys., 230 (2011), pp. 5610–5634, <https://doi.org/10.1016/j.jcp.2011.03.045>.
- [73] F. VICO, L. GREENGARD, M. O'NEIL, AND M. RACHH, *A fast boundary integral method for high-order multiscale mesh generation*, SIAM J. Sci. Comput., 42 (2020), pp. A1380–A1401, <https://doi.org/10.1137/19M1290450>.
- [74] I. VON GLEHN, T. MÄRZ, AND C. B. MACDONALD, *An embedded method-of-lines approach to solving partial differential equations on surfaces*, July 2013, <https://arxiv.org/abs/1307.5657>.
- [75] H. WENDLAND AND J. KÜNEMUND, *Solving partial differential equations on (evolving) surfaces with radial basis functions*, Adv. Comput. Math., 46 (2020), p. 64, <https://doi.org/10.1007/s10444-020-09803-0>.
- [76] G. B. WRIGHT, A. M. JONES, AND V. SHANKAR, *MGM: A meshfree geometric multilevel method for systems arising from elliptic equations on point cloud surfaces*, Apr. 2022, <https://arxiv.org/abs/2204.06154>.
- [77] Y. ZHANG, A. GILLMAN, AND S. VEERAPANENI, *A fast direct solver for integral equations on locally refined boundary discretizations and its application to multiphase flow simulations*, Aug. 2021, <https://arxiv.org/abs/2108.07205>.