MGM: A MESHFREE GEOMETRIC MULTILEVEL METHOD FOR SYSTEMS ARISING FROM ELLIPTIC EQUATIONS ON POINT CLOUD SURFACES

GRADY B. WRIGHT[*], ANDREW M. JONES[†], AND VARUN SHANKAR[‡]

Abstract. We develop a new meshfree geometric multilevel (MGM) method for solving linear systems that arise from discretizing elliptic PDEs on surfaces represented by point clouds. The method uses a Poisson disk sampling-type technique for coarsening the point clouds and new meshfree restriction/interpolation operators based on polyharmonic splines for transferring information between the coarsened point clouds. These are then combined with standard smoothing and operator coarsening methods in a V-cycle iteration. MGM is applicable to discretizations of elliptic PDEs based on various localized meshfree methods, including RBF finite differences (RBF-FD) and generalized finite differences (GFD). We test MGM both as a standalone solver and preconditioner for Krylov subspace methods on several test problems using RBF-FD and GFD, and numerically analyze convergence rates, efficiency, and scaling with increasing point cloud sizes. We also perform a side-by-side comparison to algebraic multigrid (AMG) methods for solving the same systems. Finally, we further demonstrate the effectiveness of MGM by applying it to three challenging applications on complicated surfaces: pattern formation, surface harmonics, and geodesic distance.

Key words. PDEs on surfaces, RBF-FD, GFD, meshfree, meshless, multilevel, preconditioners

AMS subject classifications. 65F08, 65F10, 65N55, 65M55, 65N22, 65M22

1. Introduction. Partial differential equations (PDEs) defined on surfaces (or manifolds) arise in many areas of science and engineering, where they are used to model, for example, atmospheric flows [55], chemical signaling on cell membranes [31], morphogenesis [47], and textures for computer graphics [51]. Solutions of these models can rarely be achieved by analytical means and must instead be approximated using numerical techniques. While numerical methods for PDEs on the sphere have been developed since the 1960s [55], development of methods for PDEs on more general surfaces only began in the late 1980s [15], with interest growing considerably in the early 2000s [16]. These techniques include surface finite element (SFE) [16], embedded finite element (EFE) [8, 36], and closest point (CP) [32] methods. More recently, various meshfree (or meshless) methods have also been developed for PDEs on general surfaces that use a local stencil approach, including radial basis function-finite differences (RBF-FD) [2, 29, 40, 44, 45, 53], generalized finite differences (GFD) [48], and generalized moving least squares (GMLS) [22, 30, 49]. These methods can be applied for surfaces represented only by point clouds and do not require a surface triangulation like SFE methods or a level-set representation of the surface like EFE methods. Additionally, these meshfree methods approximate the solutions directly on the point cloud and do not extend the PDEs into the embedding space like the EFE and CP methods.

In this paper, we concentrate on local meshfree methods for elliptic PDEs on surfaces, which are challenging to solve with iterative methods because of the poor conditioning of the systems. We specifically focus on the surface Poisson and shifted (or screened) surface Poisson problems, which arise, for example, in surface hydrodynamics [22], computer graphics [41], and time-implicit discretizations of surface reaction diffusion equations [45]. We focus on two methods for these PDEs: polyharmonic spline-based RBF-FD with polynomials and GFD. These meshfree discretizations result in large, sparse, non-symmetric, linear systems of equations that need to be solved. Direct solvers for these systems have most commonly been used, but these do not scale well to large point clouds and high-orders of accuracy, motivating the need for efficient and robust iterative methods.

Multigrid methods are known to be effective solvers and preconditioners for linear systems that arise from discretizing elliptic PDEs (e.g., [50]). These methods can be classified into two types: geometric and algebraic. While algebraic multigrid (AMG) methods are general purpose

---
[*]Boise State University (gradywright@boisestate.edu).

[†]Boise State University (andrewjones237@u.boisestate.edu).

[‡]University of Utah (shankar@cs.utah.edu).

solvers/preconditioners, geometric methods, when they can be developed, generally converge faster and work as better preconditioners for Krylov subspace methods. Geometric multigrid methods have been developed for SFE and CP discretizations (e.g., [26] and [10]) and it is the aim of this paper to also develop these methods for meshfree discretizations.

The basic components of geometric multigrid methods that need to be developed are (1) techniques for coarsening the grid or mesh, (2) constructing interpolation/restriction operators for transferring the information between levels, (3) discretizing the differential operator on the coarser levels, (4) smoothing the approximate solution, and (5) solving the system on the coarsest level. The first component presents a challenge for meshfree surface PDEs as there is only a point cloud available and no grid or mesh to create a hierarchy of coarser levels. To overcome this challenge we use the weighted sample elimination (WSE) method from [57], which is a general purpose method for selecting quasi-uniformly spaced subsets of points from a point cloud and falls into the general category of Poisson disk sampling methods [9]. The lack of a grid or mesh also presents a challenge for component (2) as standard transfer operators cannot be used. To overcome this challenge, we use RBFs to construct the interpolation operators for transferring the defect from coarser to finer levels. For the restriction operators, we simply use the transpose of the interpolation operators, which is a standard choice [50]. With these transfer operators, we generate component (3) using a Galerkin projection, often referred to as the Galerkin coarse grid operator. Finally, for component (4) we use standard Gauss-Seidel smoothing and for (5) we use a direct solver. We combine all of these components in a V-cycle iteration and apply it both as a solver and preconditioner. The resulting method is entirely meshfree and we refer to it as the meshfree geometric multilevel[1] (MGM) method.

The new MGM method has some similarities to the meshfree *multicloud* methods [24, 59], but also some key differences. The first major difference is that multicloud methods have been developed for PDEs posed in planar domains, whereas MGM is for surface PDEs. Another difference is with the choice of transfer operators. The method of [59] uses one-point, piecewise constant operators, while the method of [24] uses two-point, inverse-distance weighted interpolation and restriction operators. It is not clear how these latter transfer operators should be generalized to surfaces. MGM instead uses transfer operators based on RBFs, which are well suited for interpolation on surfaces [19]. A second difference is the strategy for geometric coarsening of the given point cloud. Multicloud methods use a graph coloring-type scheme for finding maximally-independent subsets of vertices to determine the coarser levels. This does not allow the size of the point clouds on the coarser levels to be controlled precisely, and it limits the coarsening factors to approximately four (for 2D problems). MGM instead uses WSE [57], which allows for arbitrary coarsening factors and for the sizes of the points in the coarser levels to be controlled exactly. A third difference is that MGM can handle degenerate PDEs (e.g., the surface Poisson equation), while the multicloud methods have been tailored to non-degenerate PDEs (e.g., planar Poisson equation with mixed Dirichlet-Neumann boundary conditions). Finally, multicloud methods have only been tested on second order accurate discretizations of PDEs; these discretizations typically only use small stencils. We demonstrate that MGM works for discretizations at least up to sixth order accurate with large stencil sizes.

We note also that there are other methods in the RBF literature for numerically solving PDEs that use the term multilevel (e.g., [17,18,27]), but that are very different from MGM. These methods approximate the solution of a PDE starting on a coarse point cloud and then successively update the solution by solving the PDE for the residual on a nested sequence of finer point clouds. However, they do not cycle back through the coarser levels. On each level the solution is computed using collocation with compactly supported RBFs to keep the computational cost under control. These methods are not iterative and are not designed to solve a linear system that arises from a discretization of a PDE like MGM. They do, however, have a convergence theory for certain classes of PDEs posed on certain domains [17, 27], but this theory does not include surfaces beyond the sphere.

---

[1]We use the term multilevel rather than multigrid, since this method does not depend on a grid.

The remainder of the paper is organized as follows. In Section 2, an overview is given of the two meshfree methods for surface PDEs that the MGM algorithm is used to solve. The next section presents the RBF-based transfer operators. Section 4 describes the remaining components of the MGM algorithm, including a discussion of the changes necessary to solve the degenerate surface Poisson problem. Section 5 presents an extensive array of numerical results for the MGM method, including a comparison with algebraic multigrid (AMG) methods. Section 6 then uses the method in three challenging applications to further demonstrate its effectiveness. Finally, the paper concludes with some final remarks on the method and some future directions in Section 7.

1.1. Assumptions and notation. Throughout the manuscript we let $\mathcal{M}$ be a smooth embedded manifold of co-dimension one in $\mathbb{R}^3$ with no boundary and let $\Delta_\mathcal{M}$ denote the Laplace-Beltrami operator (LBO) (or surface Laplacian) on $\mathcal{M}$. When referencing points on $\mathcal{M}$, we assume they are represented as coordinates in $\mathbb{R}^3$, e.g., for $\mathbf{x} \in \mathcal{M}$, and we write $\mathbf{x} = (x, y, z)$. For a point $\mathbf{x} \in \mathcal{M}$, we let $T_\mathbf{x}\mathcal{M}$ denote the tangent plane to $\mathcal{M}$ at $\mathbf{x}$. We denote normal vectors to $\mathcal{M}$ as $\mathbf{n}$ and assume that they are available either analytically or using some approximation technique (see for example [25]).

We use sub/superscripts h and H on variables to indicate whether they are associated with the fine or coarse level point clouds, respectively. For example, X_h and X_H denote the set of points in the fine level and coarse level point clouds, respectively. This is meant to mimic the notation that is used in traditional grid based geometric multigrid methods [50], but these parameters do not relate to anything specific about the spacing of the points and do not need to be computed.

The focus of this study is on the elliptic equation

$$\mathcal{L}u = f, \tag{1.1}$$

where $u : \mathcal{M} \to \mathbb{R}$ is unknown and $f : \mathcal{M} \to \mathbb{R}$ is known. Here $\mathcal{L} = \Delta_\mathcal{M}$ or $\mathcal{L} = \mathcal{I} - \mu\Delta_\mathcal{M}$, where $\mathcal{I}$ is the identity operator and $\mu > 0$, which correspond to the surface Poisson and shifted (or screened) surface Poisson equation (1.1), respectively. For the surface Poisson problem, we assume f satisfies the *compatibility condition* $\int_\mathcal{M} f \, dA = 0$, which is a necessary and sufficient condition for (1.1) to have a solution. In this case, any solution is unique up to the addition of a constant since constants satisfy the homogeneous equation.

2. Localized meshfree discretizations. Several localized meshfree methods have been developed for approximating the solution of (1.1), e.g., [2, 29, 30, 40, 44, 45, 49]. For the sake of brevity, we limit the focus of this study to two localized meshfree methods: polyharmonic spline (PHS)-based RBF-FD with polynomials and GFD. Both of these methods use the so-called tangent plane approach, but differ in the approximation spaces used. They should be sufficient to demonstrate the general applicability of the MGM method.

The RBF-FD and GFD methods are based on approximating the strong form of the equation, and amount to discretizing the LBO $\Delta_\mathcal{M}$ over a local stencil of points on the surface. This stencil based approach can generally be described as follows. Let $\mathbf{X}_h = \{\mathbf{x}_i\}_{i=1}^{N_h}$ denote the global point cloud (node set) discretizing $\mathcal{M}$. For each $\mathbf{x}_i \in \mathbf{X}_h$, $i = 1, \ldots, N_h$, let σ_h^i denote the set of indices of the $n_i > 1$ nearest neighbor nodes in $\mathbf{X}_h$ to $\mathbf{x}_j$. Here we use the Euclidean distance in $\mathbb{R}^3$ to define the nearest neighbor distances. The points $\mathbf{X}_h^i = \{\mathbf{x}_j\}_{j \in \sigma_h^i}$ are the *stencil* for $\mathbf{x}_i$, and $\mathbf{x}_i$ is the *stencil center*. The stencil based approximation to $\Delta_\mathcal{M}u$ at $\mathbf{x}_j$ then takes the form

$$\Delta_\mathcal{M}u\big|_{\mathbf{x}_i} \approx \sum_{j \in \sigma_h^i} c_{ij}u(\mathbf{x}_j), \tag{2.1}$$

where c_{ij} are some set of weights determined by the RBF-FD or GFD methods discussed below. These weights can then be assembled into a global N_h-by-N_h (sparse) differentiation matrix D_h and an approximate solution to (1.1) is given as a solution to the linear system

$$L_h u^h = f^h, \tag{2.2}$$

where $u^h, f^h \in \mathbb{R}^{N_h}$ contain the unknown solution and known right hand side of (1.1), respectively, sampled at $\mathbf{X}_h$. The matrix L_h is given by $L_h = D_h$ or $L_h = I_h - \mu D_h$, where I_h is the N_h-by-N_h identity matrix. Note that the latter L_h arises from time-implicit discretizations of surface diffusion-type problems. The MGM method will be used for solving the system (2.2).

In the remainder of this section, we give specific details on determining the stencil weights in (2.1) for the LBO using RBF-FD and GFD methods. Since both of these methods use the tangent plane technique, we review it first.

2.1. Tangent plane method. The tangent plane idea was introduced by Demanet [13] and recently further refined by Suchde & Kuhnert [48], Shaw [46], and, in the case of the unit two-sphere, by Gunderman et. al. [23]. The central idea of the method is to approximate the LBO at the center of each stencil $\mathbf{X}_h^i$ using an approximation to the standard Laplacian on the plane tangent to the surface at the stencil center, $T_{\mathbf{x}_i}\mathcal{M}$. The approximation is constructed from a projection of the stencil points to $T_{\mathbf{x}_i}\mathcal{M}$. In this work, we use the projection advocated in [48], which is known as an orthographic projection when $\mathcal{M}$ is the unit sphere.
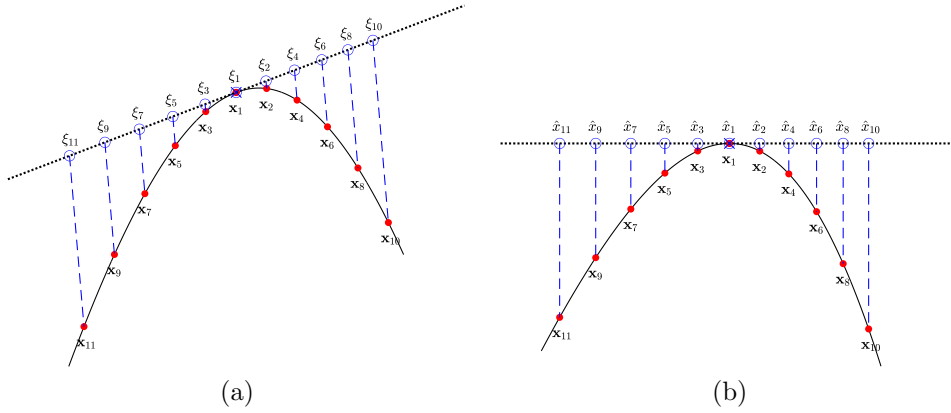

FIG. 1. *Illustration of the tangent plane method for a 1D surface (curve). The solid black lines indicates the surface, the solid red circles mark the $n = 11$ stencil nodes, the open blue circles mark the projected nodes, and the $\times$'s marks the stencil center. (a) Direct projection of the stencil points according to (2.3). (b) Rotation and projection of the stencil points according to (2.5).*

With out loss of generality, we describe the method for the first stencil $\mathbf{X}_h^1$ with index set σ_h^1. To simplify notation, we set the number of stencil points to $n_1 = n$ and assume $\sigma_h^1 = \{1, \ldots, n\}$, so that the stencil points are simply $\mathbf{X}_h^1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The tangent plane method from [48] projects these points onto $T_{\mathbf{x}_1}\mathcal{M}$ along the normal vector $\mathbf{n}_1$ (the normal to the surface at $\mathbf{x}_1$). This projection is illustrated in Figure 1 (a) for a one dimensional surface (curve) in $\mathbb{R}^2$. The projected points can be computed explicitly by

$$(2.3) \qquad \boldsymbol{\xi}_j = (I - \mathbf{n}_1 \mathbf{n}_1^T)(\mathbf{x}_j - \mathbf{x}_1), \; j = 1, \ldots, n,$$

where we have shifted the projected points so that $\boldsymbol{\xi}_1$ is at the origin. For a two dimensional surface, the projected points can be expressed in terms of orthonormal vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ that span $T_{\mathbf{x}_1}\mathcal{M}$ as follows

$$(2.4) \qquad \boldsymbol{\xi}_j = \underbrace{\begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} \hat{x}_j \\ \hat{y}_j \end{bmatrix}}_{\hat{\mathbf{x}}_j}, \; j = 1, \ldots, n.$$

The 2D coordinates $\hat{\mathbf{x}}_j$ in the tangent plane for the projected points are what will be used for constructing the approximations to the Laplacian. These can be computed directly from the relationship (2.3) and (2.4) as

$$\hat{\mathbf{x}}_j = R^T(\mathbf{x}_j - \mathbf{x}_1), \; j = 1, \ldots, n, \tag{2.5}$$

where we have used $R^T(I - \mathbf{n}_1\mathbf{n}_1^T) = R^T$. We denote the the projected stencil as $\hat{\mathbf{X}}_h^1 = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n\}$. The above procedure is repeated for every stencil $\mathbf{X}_h^i$, to obtain the projected stencils $\hat{\mathbf{X}}_h^i$, $i = 1, \ldots, N_h$.

   We note that, geometrically speaking, (2.5) amounts to first shifting the stencil points so the center is at the origin, rotating them so the normal $\mathbf{n}_1$ is orthogonal to the xy-plane, and then dropping the third component. This is illustrated in Figure 1 (b) for the case of a 1D curve.

   2.2. PHS-based RBF-FD with polynomials. The RBF-FD method for determining the weights in (2.1) can be derived by constructing an RBF interpolant over each of the projected stencil points to the tanget plane, applying the standard 2D Laplacian to the interpolants, and then evaluating them at the stencil center. In this study we focus on interpolants constructed from PHS kernels and polynomials [5, 14, 46]. Without loss of generality, we again describe the method for the first stencil, with $\mathbf{X}_h^1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, to simplify notation.

   For the stencil $\mathbf{X}_h^1$, the PHS interpolant to the projected stencil $\hat{\mathbf{X}}_h^1$ takes the form

$$s(\hat{\mathbf{x}}) = \sum_{i=1}^{n} a_i \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2k+1} + \sum_{j=1}^{L} b_j p_j(\hat{\mathbf{x}}), \tag{2.6}$$

where $\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} & \hat{y} \end{bmatrix}^T \in T_{\mathbf{x}_1}\mathcal{M}$, $\| \cdot \|$ denotes the Euclidean norm, k is the order of the PHS kernel, and $\{p_1, \ldots, p_L\}$ is a basis for bivariate polynomials in $T_{\mathbf{x}_1}\mathcal{M}$ of degree ℓ (so that $L = (\ell+1)(\ell+2)/2$). The order k controls the smoothness of the PHS and is chosen such that $0 \le k \le \ell$. We note that the polynomials can be chosen to be the standard bivariate monomials in the components of $\hat{\mathbf{x}}$. For samples $\{u_1, \ldots, u_n\}$ of an arbitrary function at the stencil points $\mathbf{X}_h^1$, the coefficients for the interpolant in the tangent plane are determined by the conditions

$$s(\hat{\mathbf{x}}_i) = u_i, \; i = 1, \ldots, n \quad \text{and} \quad \sum_{i=1}^{n} a_i p_j(\hat{\mathbf{x}}_i) = 0, \; j = 1, \ldots, L. \tag{2.7}$$

These extra L moment conditions are needed to make the solution well-posed, since the PHS kernels are (strictly) positive definite on the subspace of $\mathbb{R}^n$ consisting of vectors that satisfy these conditions [52]. The interpolation coefficients can be determined from the solution of the following linear system

$$\begin{bmatrix} A & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} = \begin{bmatrix} \underline{u} \\ \underline{0} \end{bmatrix}, \tag{2.8}$$

where $A_{ij} = \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^{2k+1}$ $(i, j = 1, \ldots, n)$, $P_{ij} = p_j(\hat{\mathbf{x}}_i)$ $(i = 1, \ldots, n, \, j = 1, \ldots, L)$, and underlined terms denote vectors containing the corresponding terms in (2.7). If the stencil nodes $\hat{\mathbf{X}}_h^1$ are such that the only bivariate polynomial of degree ℓ that vanishes at $\hat{\mathbf{X}}_h^1$ is the trivial polynomial (i.e., rank$(P) = L$), then (2.8) has a unique solution, so that (2.6) is well-posed [52]. This is a mild condition on the stencil nodes, especially for "scattered" nodes on the tangent plane.

The stencil weights c_{1j} in (2.1) are determined from the approximation

$$\Delta_{\mathcal{M}} u\big|_{\mathbf{x}_1} \approx \hat{\Delta} s\big|_{\hat{\mathbf{x}}_1} = \sum_{i=1}^{n} a_i \hat{\Delta}(\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2k+1})\big|_{\hat{\mathbf{x}}_1} + \sum_{j=1}^{L} b_j \hat{\Delta}(p_j(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_1}$$

$$= \begin{bmatrix} \hat{\Delta}\underline{s} & \hat{\Delta}\underline{p} \end{bmatrix}^T \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix},$$

where $\hat{\Delta} = \partial_{\hat{x}\hat{x}} + \partial_{\hat{y}\hat{y}}$, $\hat{\Delta}\underline{s}$ and $\hat{\Delta}\underline{p}$ are vectors containing the entries $\hat{\Delta}(\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2k+1})\big|_{\hat{\mathbf{x}}_1}$, $i = 1, \ldots, n$, and $\hat{\Delta}(p_j(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_1}$, $j = 1, \ldots, L$, respectively. Using (2.8) in the above expression, the stencil weights are given as the solution to the following linear system

$$(2.9) \qquad \begin{bmatrix} A & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{c} \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \hat{\Delta}\underline{s} \\ \hat{\Delta}\underline{p} \end{bmatrix},$$

where $\underline{c}$ contains c_{1j} and $\underline{\lambda}$ are unused.

We note that one can interpret (2.9) as the solution to an equality constrained optimization problem where the weights are determined by enforcing they are exact for $\hat{\Delta}(\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_i\|^{2k+1})\big|_{\hat{\mathbf{x}}_1}$, $j = 1, \ldots, n$, subject to the constraint that they are also exact for $\hat{\Delta}(p_j(\hat{\mathbf{x}}))\big|_{\hat{\mathbf{x}}_1}$, $j = 1, \ldots, L$. Under this interpretation, $\underline{\lambda}$ is the vector of Lagrange multipliers [5].

In this work, we choose the order of the PHS as $k = \ell$ and fix the stencil size $n_j = n$, $j = 1, \ldots, N$ as $n = 2L$, which is a common choice for RBF-FD methods [5]. The degree ℓ of the appended polynomial can then be used to control the approximation order of the method, with larger ℓ leading to higher orders [12].

2.3. GFD. This method is similar to the RBF-FD method, but instead of using an interpolant, the method is based on a (weighted) polynomial least squares approximant. Using the same notation and assumptions as the previous section and again focusing only on the first stencil $\mathbf{X}_h^1$, the approximant for the projected stencil $\hat{\mathbf{X}}_h^1$ takes the form

$$(2.10) \qquad q(\hat{\mathbf{x}}) = \sum_{j=1}^{L} b_j p_j(\hat{\mathbf{x}}).$$

The coefficients of the approximant are determined from the samples $\{u_1, \ldots, u_n\}$ according to the the following weighted least squares problem:

$$(2.11) \qquad \underline{b}^* = \underset{\underline{b} \in \mathbb{R}^L}{\operatorname{argmin}} \sum_{i=1}^{n} w(\hat{\mathbf{x}}_i)(q(\hat{\mathbf{x}}_i) - u_i)^2 = \underset{\underline{b} \in \mathbb{R}^L}{\operatorname{argmin}} \|W^{1/2}(P\underline{b} - \underline{u})\|_2^2,$$

where $W = \operatorname{diag}(w(\hat{\mathbf{x}}_i))$. Here we again assume the stencil nodes $\hat{\mathbf{X}}_h^1$ have the property that P is full rank so that (2.11) has a unique solution. There are many different options for selecting the weight function w in the literature. In this work, we follow [48] and use the following Gaussian function:

$$w(\hat{\mathbf{x}}_i) = \exp\left(-\alpha \frac{\|\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_i\|^2}{\rho_1^2 + \rho_i^2}\right),$$

where ρ_k is the support of the kth projected stencil $\hat{\mathbf{X}}_h^i$, i.e. the radius of the minimum ball centered at $\hat{\mathbf{x}}_i$ that encloses all the points in $\hat{\mathbf{X}}_h^i$. The parameter $\alpha > 0$ is used for controlling the shape of the weight function and is typically chosen in an ad hoc manner [48].

The stencil weights in (2.1) are determined from the approximation

$$\Delta_{\mathcal{M}} u\big|_{\mathbf{x}_1} \approx \hat{\Delta} q\big|_{\hat{\mathbf{x}}_1} = (\hat{\Delta}\underline{p})^T \underline{b}^*.$$

Using the normal equation solution of (2.11) in the above expression, the vector of stencil weights $\underline{c}$ is given as

$$(2.12) \qquad\qquad \underline{c} = WP(P^TWP)^{-1}(\hat{\Delta}\underline{p}).$$

In practice, a QR factorization of WP is used instead of the normal equations to improve the numerical conditioning of (2.12).

In this work, we choose the number of points in the stencils $\mathbf{X}_h^i$ for this method in the same manner as the RBF-FD technique. Note that, similar to RBF-FD, increasing the polynomials degree ℓ also increases the order of accuracy of the GFD method.

3. Multilevel transfer operators using RBFs. Operators for transferring information between coarse and fine levels are one of the key components of multilevel methods. The interpolation transfer operators are used to transfer information from coarse level to a finer level, while the restriction operators are used for the reverse. Let $\mathbf{X}_h = \{\mathbf{x}_i\}_{i=1}^{N_h} \subset \mathcal{M}$ denote the fine set of nodes and $\mathbf{X}_H = \{\mathbf{y}_j\}_{j=1}^{N_H} \subset \mathcal{M}$ the coarse set, where $N_H < N_h$. We denote the interpolation operator by I_H^h and the restriction operator by I_h^H. These can be represented as (sparse) matrices, so that for a vector u_H of data on the coarse nodes $\mathbf{X}_H$, the vector containing the interpolation of u^H to $\mathbf{X}_h$ is given as $u^h = I_H^h u_H$. In this section, we discuss a novel meshfree method for constructing I_H^h based on RBF interpolation. For the restriction operator, we use $I_h^H = (I_H^h)^T$, which is a standard choice, especially in AMG methods [50, Appendix A].

Similar to the discrete LBO, we compute the interpolation operator using a stencil based approach. Letting σ_H^i be the indicies of the m_i nearest neighbors in $\mathbf{X}_H$ to $\mathbf{x}_i$, the interpolation of $\{u_j^H\}_{j\in\sigma_H^i}$ to u_i^h, the entry in u_h corresponding to $\mathbf{x}_i$, is given as

$$(3.1) \qquad\qquad u_i^h = \sum_{j\in\sigma_H^i} d_{ij} u_j^H.$$

We again use the Euclidean distance in $\mathbb{R}^3$ to define the index set σ_H^i. The weights for each stencil can be assembled to form the (sparse) interpolation matrix I_H^h.

We use local RBF interpolants about each stencil $\mathbf{X}_H^i = \{\mathbf{y}_j\}_{j\in\sigma_H^i}$ to determine the weights in (3.1) and form these interpolants in the embedding space $\mathbb{R}^3$. This is considerably simpler than using intrinsic coordinates to $\mathcal{M}$ and the resulting interpolants have good approximation properties [19]. One could alternatively use interpolants in the tangent plane about each stencil center similar to Section 2.2, but these are only accurate when the surface is well discretized by the underlying point cloud. This will not necessarily be the case for the nodes $\mathbf{X}_H$ as we coarsen the finer levels. We again use a PHS kernel to form the interpolants and describe the method for the first stencil $\mathbf{X}_H^1$, which, to simplify the notation, we assume to consist of the nodes $\{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$.

For the stencil $\mathbf{X}_H^1$, the PHS interpolant takes the same form as (2.6), but with $\hat{\mathbf{x}}$ replaced by $\mathbf{x}$ and $\{\hat{\mathbf{x}}_i\}$ is replaced by $\{\mathbf{y}_i\}$. Additionally, we only consider the PHS kernel with $k = 0$ and a constant term appended to the interpolant (i.e. $\ell = 0$). The weights d_{1j} in (3.1) are determined by evaluating this interpolant at $\mathbf{x}_1$ and can be computed in a similar procedure to that used in deriving the system (2.9). The linear system for the interpolation weights takes the form

$$(3.2) \qquad\qquad \begin{bmatrix} A & \underline{1} \\ \underline{1}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \underline{d} \\ \lambda \end{bmatrix} = \begin{bmatrix} \underline{s} \\ 1 \end{bmatrix},$$

Algorithm 4.1 Two-Level Cycle

1: Pre-smooth initial guess: $u^h \leftarrow \text{presmooth}(L_h, u^h, f^h, \nu_1)$
2: Compute residual: $r^h = f^h - L_h u^h$
3: Restrict the residual to $\mathbf{X}_H$: $r^H = I_h^H r^h$
4: Solve for the defect: $L_H e^H = r^H$
5: Interpolate the defect to $\mathbf{X}_h$: $e^h = I_H^h e^H$
6: Correct the approximation: $u^h \leftarrow u^h + e^h$
7: Post-smooth the approximation: $u^h \leftarrow \text{postsmooth}(L_h, u^h, f^h, \nu_2)$

where $A_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$ $(i, j = 1, \ldots, m)$, $\underline{1}$ is the vector of length n with all ones, and $\underline{s}$ has entries $\|\mathbf{x}_1 - \mathbf{y}_i\|$, $i = 1, \ldots, m$. Again, λ is unused.

While higher order PHS kernels ($k > 0$) and higher degree polynomials ($\ell > 0$) could be used in constructing the interpolation weights, we found that the simple formulation above gave good results, while also being efficient, for the range of problems we considered. This formulation also has the added benefit that the system (3.2) has a unique solution, provided the points are distinct [52]. When using larger k and ℓ this may not be the case as the points must be unisolvent with respect to the space of trivariate polynomials of degree ℓ, i.e., $\text{rank}(P) = L$. Since the interpolation is done in the embedding space, this can be an issue for certain algebraic surfaces (e.g., the sphere with $\ell \geq 2$).

4. Meshfree geometric multilevel (MGM) method. In this section we present the MGM method for solving the discrete problem (2.2). We first present the MGM method in terms of a two-level cycle, which is summarized in Algorithm 4.1, describing its primary components: coarsening the point cloud $\mathbf{X}_h \to \mathbf{X}_H$, forming the coarse level operator L_H, smoothing the approximation, and solving for the defect on the coarse level. The interpolation/restriction operators are described in the previous section. We then focus on some modifications to the algorithm that are necessary when (2.2) corresponds to the surface Poisson problem. This is followed by a description of the multilevel extension of the method. Finally, we comment on using the method as a preconditioner for Krylov subspace methods.

4.1. Node coarsening. The technique we propose for generating the coarser point clouds on general surfaces is based on the (WSE) method from [57]. This algorithm falls into the category of Poisson disk sampling methods, which produce quasiuniformly spaced point sets [9]. The WSE method approximates the solution to the following optimization problem: Given a point cloud $\mathbf{X}_h$ with N_h samples, determine a subset $\mathbf{X}_H$ of $\mathbf{X}_h$ with N_H samples that has maximal Poisson disk radius. The Poisson disk radius is defined as one half the minimum distance between neighboring points in the set (which is called the separation radius in the meshfree methods literature [52]). This optimization problem is NP complete, but the WSE algorithm approximates the solution in $N_h - N_H$ steps with a theoretical complexity of $\mathcal{O}(N_h \log N_h)$ operations [57]. The method works for point clouds defined on many different sampling domains, including arbitrary manifolds, where it uses the Euclidean norm in $\mathbb{R}^3$ to define nearest neighbor distances. We use the implementation by the author of the WSE method, called `cySampleElimination`, that is provided in the `cyCodeBase` [58].

In this work, we coarsen the point cloud $\mathbf{X}_h$ by a fixed factor of 4, so that $\mathbf{X}_H$ has $N_H = \lfloor N_h/4 \rfloor$ points. This mimics the standard coarsening of geometric multigrid for two-dimensional domains. We tested other coarsening factors, but found that coarsening by 4 generally gave the best results in terms of iteration count and wall clock time for the multilevel method. Figure 2 illustrates the coarse point clouds $\mathbf{X}_H$ with this coarsening factor computed from the WSE algorithm for two example surfaces.

4.2. Coarse level operator. There are two main approaches to constructing the coarse level operator in multilevel methods. The first is *direct discretization*, where the differential operator is
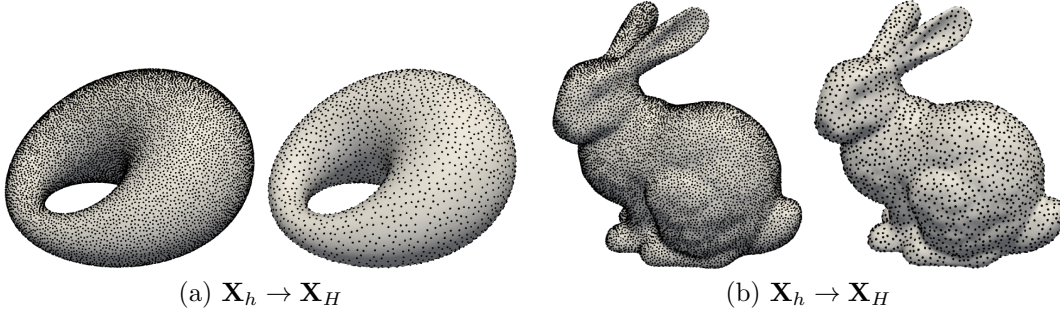
(a) $\mathbf{X}_h \to \mathbf{X}_H$ (b) $\mathbf{X}_h \to \mathbf{X}_H$

FIG. 2. *Illustration of the WSE algorithm for generating a coarse level set $\mathbf{X}_H$ of $N_H = \lfloor N_h/4 \rfloor$ points from a fine level set $\mathbf{X}_h$ of N_h points. Here $N_h = 14561$ & $N_H = 3640$ for the cyclide (a) and $N_h = 14634$ & $N_H = 3658$ for the Stanford Bunny (b) .*

discretized directly on the coarse level points $\mathbf{X}_H$. The second is based on a Galerkin projection involving the interpolation I_H^h and restriction I_h^H operators, and is defined as follows:

$$(4.1) \qquad\qquad\qquad\qquad L_H = I_h^H L_h I_H^h.$$

This latter operator, referred to as the *Galerkin coarse grid operator*, provides a simple means of coarsening L_h and has been shown to be robust for a large class of problems, especially those where a direct discretization on the coarse grid does not adequately represent the approximation on the fine grid [50, §7.7.4]. It also gives rise to a variational principle that is exploited in the analysis of algebraic multigrid (AMG) [50, §A.2.4] methods. While this latter result relies on the matrix being symmetric positive define, modifications to this theory have also been developed for non-symmetric problems, which involve choosing the restriction operator differently than the transpose of the interpolation operator [34]. We use the Galerkin approach for forming L_H, as we have found that it approximates the fine grid operator on the coarser levels better than the direct discretization technique and it makes for a more robust solver/preconditioner. While L_h is not symmetric for our discretizations, we have nonetheless found that simply choosing $I_h^H = (I_H^h)^T$ works well over a large array of test problems.

One disadvantage of the Galerkin approach is that L_H has to be formed explicitly through sparse matrix-matrix multiplication, which is more computationally expensive in terms of time and memory than the direct discretization approach. Several researchers have developed methods to reduce this cost on parallel architectures (e.g., [4, 6]), but we have not used these methods in our implementation. We simply construct L_H as part of a set-up phase using a sparse matrix library. However, we do some minor alterations to improve the computational performance. These include reordering the rows and columns of L_h to decrease its bandwidth using the reverse Cuthill-McKee (RCM) algorithm prior to forming L_H. This essentially leads to a reordering of the nodes $\mathbf{X}_h$, which in turn leads to a reordering of the interpolation operator I_H^h. We also use RCM to reorder the rows and columns of L_H after it is formed, which leads to a re-ordering of the nodes $\mathbf{X}_h$ and of the columns of I_H^h. We have found that these matrix reorderings not only reduce the wall-clock time of MGM, but also the number of iterations to reach convergence (usually by 2-3 iterations).

4.3. Smoother and coarse level solver. For the smoothing operator we use classical Gauss-Seidel (GS) method. One application of the smoother can be written as

$$(4.2) \qquad\qquad\qquad\qquad u^h \leftarrow u^h + B_h^{-1}(f^h - L_h u^h),$$

where B_h is the lower triangular part (called forward GS) or upper triangular part (called backward GS) of L_h. In some cases we vary the version of the smoother for the pre- and post- smoothing

operations (e.g., forward GS for pre-smooth and backward GS for the post-smooth). We denote the number of applications of the smoother for the pre- and post-phases of the cycle as ν_1 and ν_2, respectively.

To solve for the defect on the coarse level, we use a direct solver based on a sparse LU factorization of L_H (e.g., SuiteSparse or SuperLU).

4.4. Modifications to the two-level cycle for the surface Poisson problem. When (2.2) corresponds to the discretization of a surface Poisson problem ($L_h = D_h$), the system is singular and some modifications to the two level cycle in Algorithm 4.1 are necessary. To understand the nature of the singularity, we can look at the continuous problem (1.1). As discussed in Section 1.1, this problem has a solution if and only if the right hand side satisfies the compatibility condition. Furthermore, the solution is only unique up to the addition of a constant. The degeneracy in the continuous problem manifests in the discrete problem as a one dimensional null space of L_h corresponding to constant vectors. The discrete analog of the consistency condition is that (2.2) has a solution if and only if f^h is orthogonal to the left null vector of L_h (i.e., f^h is in the range of L_h). Also, similar to the continuous case, any solution of (2.2) is only unique up to the addition of a constant vector.

The primary issue that arises with using multilevel methods (and other iterative methods) for these types of singular systems stems from the fact that, in practice, f_h is rarely in the range of L_h. This can cause the iterations to fail to converge to a suitable approximation. Three standard approaches to bypass this issue include the following. First, one can project f^h into the range of L_h. However, this requires computing the left null vector, which can be computationally expensive[2]. It also requires modifying the coarse level solver to use the pseudoinverse (or some approximate inverse). A second approach is to impose that the solution is zero at one point. This fixes the non-uniqueness issue and transforms the problem into solving a non-singular system of one dimension smaller. However, this can lead to a deterioration of the convergence of the multilevel method since the pointwise condition is not well approximated on coarser levels [54]. Additionally, the solution to this approach can be less accurate and less smooth than the projection approach [56]. The third approach is to enforce a global constraint on the solution, such as the discrete mean of u_h is zero [50, §5.6.4]. This constraint can be enforced using a Lagrange multiplier, which transforms the linear system into the constrained system

$$(4.3) \qquad \underbrace{\begin{bmatrix} L_h & b_h^T \\ b_h & 0 \end{bmatrix}}_{\tilde{L}_h} \underbrace{\begin{bmatrix} u^h \\ \lambda^h \end{bmatrix}}_{\tilde{u}^h} = \underbrace{\begin{bmatrix} f^h \\ 0 \end{bmatrix}}_{\tilde{f}^h},$$

where b_h is a row vector of length N with all of its components set to $1/N_h$ (i.e., the discrete mean operator), and λ^h is the Lagrange multiplier. Provided b_h is not orthogonal to the left null space of L_h (which is likely to be true because of the compatibility condition for the continuous problem), this constrained system will have a unique solution [50, Lemma 5.6.1]. Furthermore, if this condition holds, the solution will be the same (up to a constant) as the projection approach, since $f^h - \lambda^h b_h^T$ is then necessarily in the range of L_h. We use the third approach in the MGM method.

Some modifications to the two-level cycle are required to handle the constrained system (4.3). First, the transfer operators have to also transfer the Lagrange multipler through the fine and coarse levels and the Galerkin coarse grid operator has to include the transferred constraint. We follow the

---

[2]Note that L_h is not symmetric, so the constant vector is not necessarily the left null vector

---

Algorithm 4.2 MGM preprocessing phase

1: **Input:** Fine level nodes $\mathbf{X}_1$ and operator L_1; minimum number of coarse level points $N_{\min}$
2: Re-order rows and columns of L_1 using RCM
3: Re-order $\mathbf{X}_1$ according to the RCM ordering
4: Compute number of levels: $p = \lfloor \log(N_1/N_{\min})/\log(4)\rfloor + 1$
5: **for** $j = 1 \ldots p - 1$ **do**
6:      Generate coarse point cloud $\mathbf{X}_{j+1}$ with $N_{j+1} = \lfloor N_1/4^j \rfloor$ points
7:      Generate interpolation operator I_{j+1}^j from $\mathbf{X}_{j+1}$ to $\mathbf{X}_j$
8:      Set the restriction operator to $I_j^{j+1} = (I_{j+1}^j)^T$
9:      Generate Galerkin coarse level operator $L_{j+1} = I_{j+1}^j L_j I_j^{j+1}$
10:     Re-order rows and columns of L_{j+1} using RCM
11:     Re-order rows of I_{j+1}^j and columns of I_j^{j+1} according to the RCM ordering
12: **end for**
13: Compute sparse LU decomposition of L_p

---

approach from [1] and modify these operators according to the following definitions:

$$(4.4) \qquad \tilde{L}_H = \underbrace{\begin{bmatrix} I_h^H & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{I}_h^H} \underbrace{\begin{bmatrix} L_h & b_h^T \\ b_h & 0 \end{bmatrix}}_{\tilde{L}_h} \underbrace{\begin{bmatrix} I_H^h & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{I}_H^h} = \begin{bmatrix} I_h^H L_h I_H^h & I_h^H b_h^T \\ b_h I_H^h & 0 \end{bmatrix},$$

where $\tilde{I}_H^h$ and $\tilde{I}_h^H$ are the modified interpolation and restriction operators, respectively, and $\tilde{L}_H$ is the modified Galerkin operator. These modified transfer operators simply pass the Lagrange multiplier between levels without alteration.

For the smoother of the constrained system (4.3), we use the approach discussed in [50, §5.6.5], where only the solution u_h is smoothed and the constraint is left alone. We again use GS for smoothing u_h and one application of the modified smoother takes the form

$$u^h \leftarrow u^h + B_h^{-1}(f^h - L_h u^h - b_h \lambda^h),$$

where B_h is the same as (4.2). This smoother is equivalent to one iteration of the undamped inexact Uzawa method with the Schur complement set equal to zero [7].

Finally, we use a direct solve to compute the defect e^h and Lagrange multiplier λ^H on the coarse level. This system takes the form $\tilde{L}_H \tilde{e}^H = \tilde{r}^H$, where $\tilde{e}^H = \begin{bmatrix} e^H & \lambda^H \end{bmatrix}^T$ and $\tilde{r}^H$ is the restricted residual for the modified system: $\tilde{r}^H = \tilde{I}_h^H(\tilde{f}^h - \tilde{L}_h \tilde{u}^h)$. When solving a Poisson problem, we use the modifications described above in Algorithm 4.1.

4.5. Multilevel extension. The multilevel extension of the two-level cycle can be obtained by applying it recursively until a sufficiently coarse level is reached to make a direct solver practical. To simplify the notation in describing the multilevel cycle, we replace the h/H superscript/subscript notation with a number corresponding to the level, with $j = 1$ being the finest level. For example, for the jth level, $\mathbf{X}_j$ denotes the point cloud, N_j denotes its size, L_j denotes the operator, r^j denotes the residual, and I_j^{j+1} is the restriction to level $j + 1$.

Before the multilevel cycle begins, we compute all the coarse point clouds, transfer operators, and Galerkin coarse level operators in a preprocessing step, which is outlined in Algorithm 4.2. The number of levels, p, depends on the number of fine level nodes and minimum number of nodes on the coarsest level, $N_{\min}$, and is determined on line 4 of this algorithm. This guarantees that the number of nodes on the coarsest level satisfies $N_{\min} \leq N_p < 4N_{\min}$. We note that when using WSE

to generate the coarse point cloud $\mathbf{X}_j$ on line 6 of the preprocessing algorithm, we use the finer point cloud $\mathbf{X}_{j+1}$, rather than the finest node $\mathbf{X}_1$. This reduces the cost in performing this step.

The multilevel cycle is outlined in Algorithm 4.3 in non-recursive form. This algorithm is what we call the MGM method and corresponds to a traditional V-cycle in multigrid methods, which is typically denoted $V(\nu_1, \nu_2)$ corresponding to the number of pre-/post smoothing operations. Other cycling methods can also be used (e.g., F- or W-cycle [50, §2.4]), but we limit our focus to the V-cycle. While this algorithm is described for a shifted Poisson problem, it can be easily modified for solving a Poisson problem following the modifications discussed in Section 4.4.

Algorithm 4.3 MGM V(ν_1, ν_2)-cycle

1: **Input:** Right hand side f^1; Initial guess u^1; Number levels p; $\{L_j\}_{j=1}^{p-1}$; $\{I_{j+1}^j\}_{j=1}^{p-1}$; $\{I_j^{j+1}\}_{j=1}^{p-1}$;
2:        RCM re-orderings; Sparse LU factorization of L_p;
3: Re-order f^1 and u^1 according to RCM re-ordering of L_1
4: Presmooth initial guess: $u^1 \leftarrow \text{presmooth}(L_1, u^1, f^1, \nu_1)$
5: Compute/restrict residual: $r^1 = I_1^2(f^1 - L_1 u^h)$
6: **for** $j = 2 \ldots p - 1$ **do**
7:       Presmooth defect: $e^j = \text{presmooth}(L_j, 0, r^j, \nu_1)$
8:       Compute/restrict residual: $r^{j+1} = I_j^{j+1}(r^j - L_j e^j)$
9: **end for**
10: Compute defect: Solve $L_p e^p = r^p$ using sparse LU decomposition of L_p
11: **for** $j = p - 1, \ldots, 2$ **do**
12:       Interpolate/correct defect: $e^j \leftarrow e^j + I_{j+1}^j e^{j+1}$
13:       Post smooth defect: $e^j \leftarrow \text{postsmooth}(L_j, e^j, r^j, \nu_2)$
14: **end for**
15: Interpolate defect/correct approximation: $u^1 \leftarrow u^1 + I_2^1 e^2$
16: Post smooth approximation: $u^1 \leftarrow \text{postsmooth}(L_1, u^1, f^1, \nu_2)$
17: Undo re-ordering of u^1 from RCM of re-ordering of L_1

4.6. Preconditioner for Krylov subspace methods. The MGM method has the benefit of being relatively straightforward to implement. However, as shown in the numerical experiments in the next section, it may converge slowly when using it as a standalone solver, especially for higher order discretizations of the LBO on more irregular point clouds. A common approach to bypassing these issues for standard geometric and algebraic multigrid methods is to combine them with a Krylov subspace method (e.g., [50, §7.8] or [20, 39]). In this case, multigrid is viewed as preconditioner for the Krylov method. We also take this approach with MGM, using it a preconditioner for two Krylov methods: generalized minimum residual (GMRES) and bi-conjugate gradient stabilized (BiCGSTAB) [42]. This combination appears to result in an efficient and robust method for solving the discretized surface Poisson and shifted Poisson equations on quite complicated surface.

5. Numerical results. In this section, we analyze the MGM method as a solver and preconditioner for the Poisson and shifted Poisson problem on two surfaces: the unit sphere and the cyclide. The latter is shown in Figure 2 and the implicit equation describing the surface is given in [29]. We test the method on both RBF-FD and GFD discretizations using the parameters given in the first part of Table 1. In all the tests, we are interested in how the method scales to higher order discretizations, and thus give results for polynomial degrees $\ell = 3$, 5, and 7, which correspond to approximately second, fourth, and sixth order accuracy, respectively [46, 48][3]. For the sphere tests,

---

[3]In some cases, it is possible to achieve these same orders with polynomials of one degree less when the point cloud has a regular spacing [30].

we generate the point clouds from the vertices of icosahedral node sets, which are used extensively in numerical weather prediction [33]. For the cyclide, we use point clouds produced from Poisson disk sampling of the surface. This latter approach results in much more unstructured point clouds than the sphere case (see Figure 2 for an illustration).

Variable	Description	Value(s)
	Parameters for the discretization the LBO	
ℓ	Poly. degree for discretizing the LBO with RBF-FD or GFD	3, 5, or 7
k	Order of the PHS kernel for discretizing the LBO with RBF-FD	ℓ
α	Weighting parameter for the Gaussian kernel in GFD	4 or 5
n	Stencil size for discretizing the LBO with RBF-FD or GFD	$(\ell+1)(\ell+2)$
	Parameters for MGM	
$N_{\min}$	Minimum number of nodes on the coarsest level	250
B_h	Pre- and post-smoother (see (4.2))	Forward GS
ν_1, ν_2	Number of applications of the pre- and post-smoother	1
m	Stencil size of the interpolation/restriction operators	3

TABLE 1
Description of parameters and their values used in the numerical results.

Unless otherwise specified, the parameters of the MGM method are set according to those given in the second part of Table 1. We tested the method with different combinations of these parameters and found that the ones listed in the table generally gave the best results in terms of iteration count and wall-clock time. Additionally, when using MGM with Krylov methods, we use it as a right preconditioner, which is generally recommended [20]. Finally, all the MGM results presented were obtained from a MATLAB implementation of the method, with a MEX interface to the WSE method, which is implemented in C++.

In the first several experiments, we compare MGM to AMG, as implemented in the Python package PyAMG [37]. In addition to being very popular blackbox solvers and preconditioners for a wide range of problems, AMG methods have been used previously for solving linear systems associated with meshfree discretizations of elliptic PDEs in the plane [43] and on surfaces [22]. We use the smoothed aggregation version of AMG, as we found it performed better than classical AMG. Additionally, we use one application of symmetric GS as the pre- and post-smoother, a V-cycle for the multilevel cycle, and sparse LU for the coarse level solver. We experimented with other combinations of parameters and again found these generally gave the best results in terms of iteration count and wall-clock time. Additionally, when using PyAMG with GMRES, we use it as a right preconditioner (with the fgmres option), while for BiCGSTAB we use it as a left preconditioner (as this is the only option). Finally, in the comparisons with AMG, we focus on the shifted Poisson problem (with $\mu = 1$) as PyAMG does not offer a specialized way to deal with the constrained system (4.3). These comparisons include all the results in Sections 5.1–5.3.

5.1. Standalone solver vs. preconditioner. In the first set of tests, we compare MGM and PyAMG both as standalone solvers and preconditioners. For the latter approaches we refer to these solvers as MGM GMRES, MGM BiCGSTAB, PyAMG GMRES, and PyAMG BiCGSTAB, to indicate the type of Krylov method employed. We use these solvers on the shifted Poisson problem on the unit sphere and cyclide with N_h=2,621,422 and N_h=2,097,152 nodes, respectively. For the BiCGSTAB results, we count the number of applications of the preconditioner as the iterations since each step of this method applies the preconditioner twice, whereas GMRES applies it once.

Figure 3 displays the results in terms of relative residual vs. iteration count for RBF-FD, while Figure 4 displays the results for GFD. For the RBF-FD results, we see that the methods using MGM
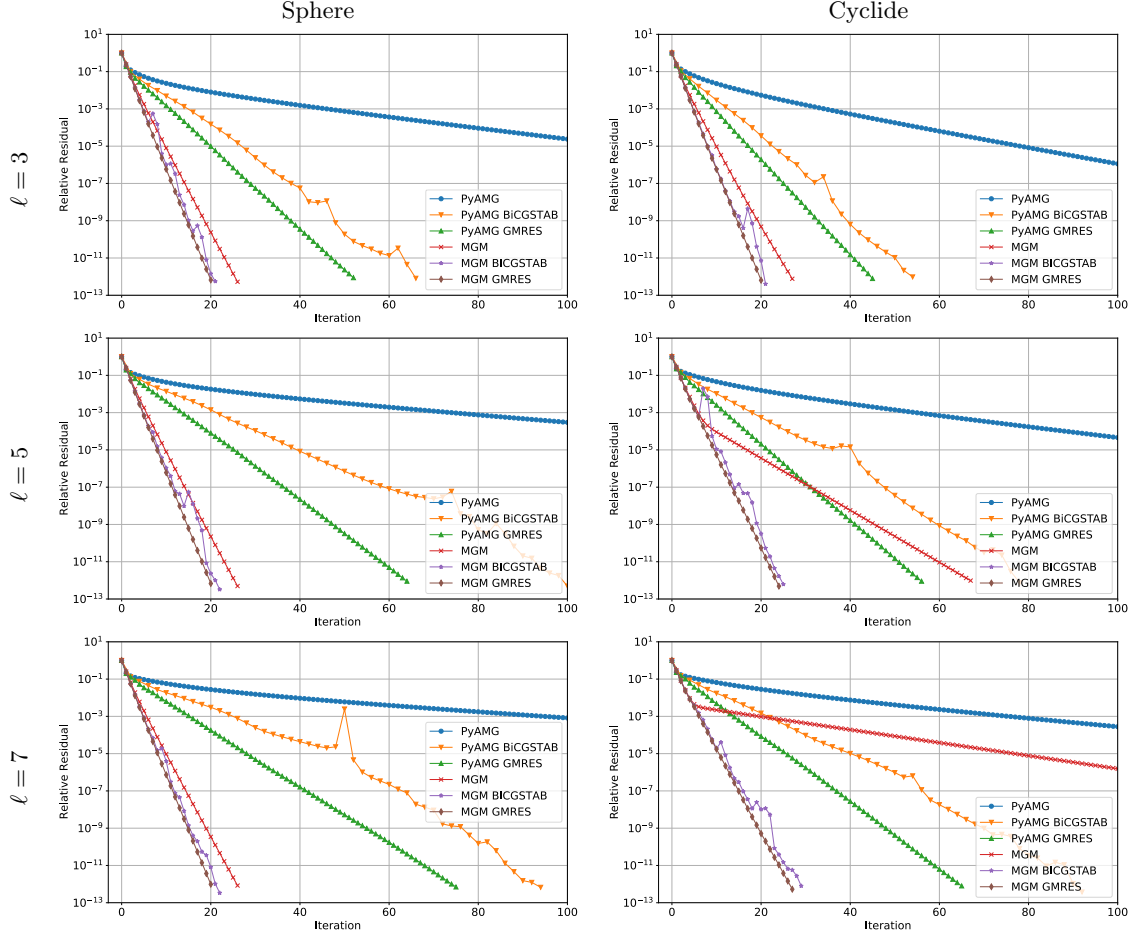
FIG. 3. *Convergence results for MGM and PyAMG based solvers for RBF-FD discretizations of a shifted Poisson problem with random right hand side. The sphere results are for $N_h = 2621442$, while for the cyclide $N_h = 2097152$.*

converge more rapidly than the methods based on PyAMG for both surfaces. For the sphere, MGM works very well as a standalone solver and preconditioner even as ℓ increases, but for the cyclide the convergence rates of MGM as a standalone solver decrease considerably. This may be due to the more irregular nature of the cyclide point cloud. We note, however, that the preconditioned versions of MGM only have a very mild decrease in convergence rates for the cyclide. The figures also show that the methods using PyAMG do not converge as rapidly as the corresponding MGM methods, with the fastest converging PyAMG method taking more than double the number of iterations as the fastest MGM method when $\ell = 3$ and triple when $\ell = 5$ and 7. We see similar patterns in the GFD results, but the methods based on both MGM and PyAMG generally converge faster in this case and the gap between the fastest converging MGM and PyAMG methods is not as wide. Finally, we note that MGM BiCGSTAB seems to converge at a very similar rate to MGM GMRES, whereas this does not hold for PyAMG. This is a promising result for large systems since the storage requirements of BiCGSTAB are fixed, whereas they grow with the size of the Krylov subspace for GMRES [20].

These experiments also indicate that, while MGM can be an effective standalone solver for small ℓ (lower order discretizations), it is more robust for larger ℓ (higher order discretizations) and when
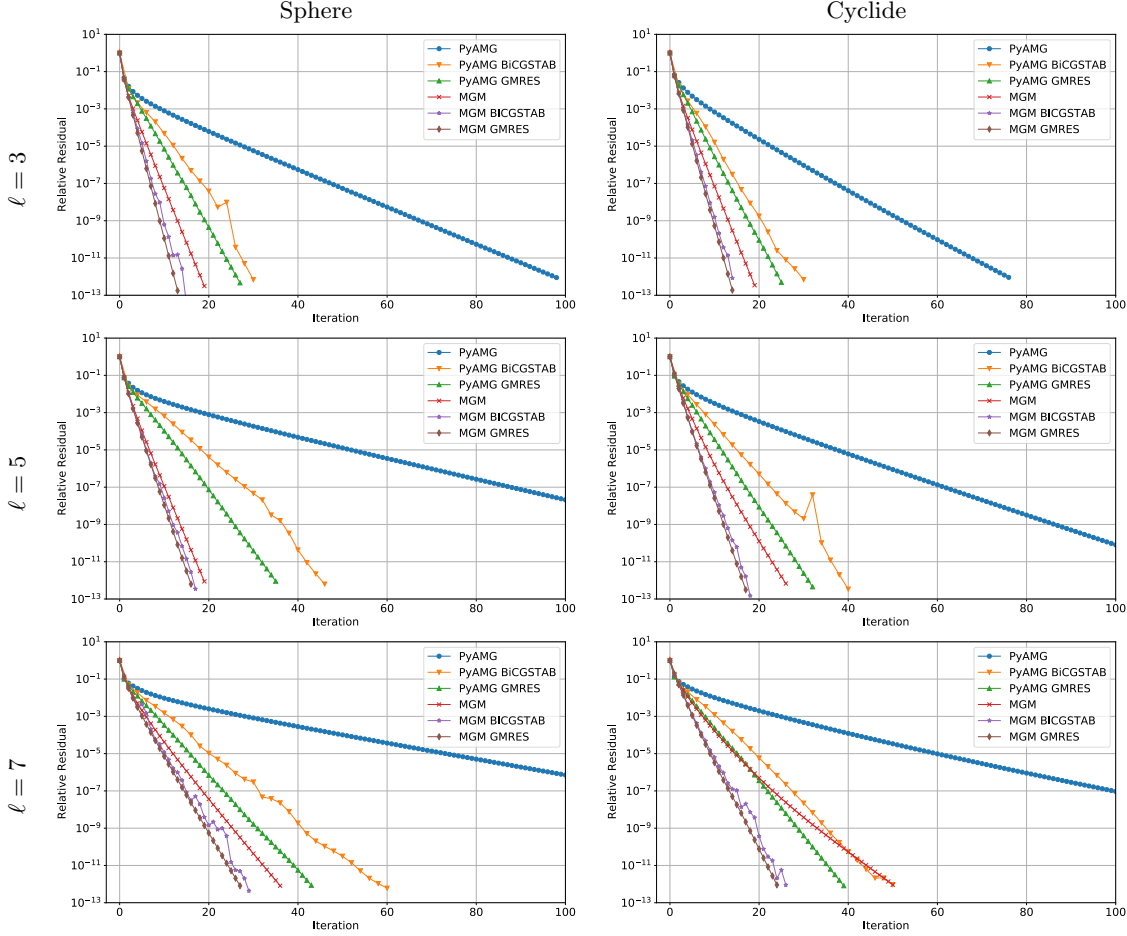
FIG. 4. *Same as Figure 3, but for GFD discretizations of the shifted surface Poisson problem.*

used as a preconditioner. This also seems to be the case when applying it to different surfaces and point clouds based on regular nodes (like the sphere) and irregular nodes (like the cyclide). From the PyAMG results, it is clear that it should be used as a preconditioner to get the most robust results, which is generally the case for AMG methods applied to nonsymmetric systems [20].

5.2. Scaling with problem size. In the next set of tests, we examine how both the MGM and PyAMG methods scale as the size of the point clouds N_h increases. We focus on the preconditioned versions of these methods and test them again on the sphere and cyclide. Figure 5 displays the results for the RBF-FD and GFD methods in terms of number of iterations required to reach a relative residual of 10^{-12}. We see from these plots that the preconditioned MGM methods appear to scale much better than the PyAMG methods, both in terms of N_h and ℓ. For RBF-FD discretizations, the increase in the iteration count for MGM is more mild with increasing ℓ than for GFD. However, the iteration count is lower for the GFD discretizations; we examine this further in Section 5.4.

In Figure 6 we display the wall-clock times for the results in Figure 5 for GMRES PyAMG and MGM. These results were run on a Linux Workstation with Intel i9-9900X 3.5 GHz processor (with no explicit parallelization) and do not include the preprocessing times. While it is not a good idea to compare execution times between methods implemented in two different scripting languages,
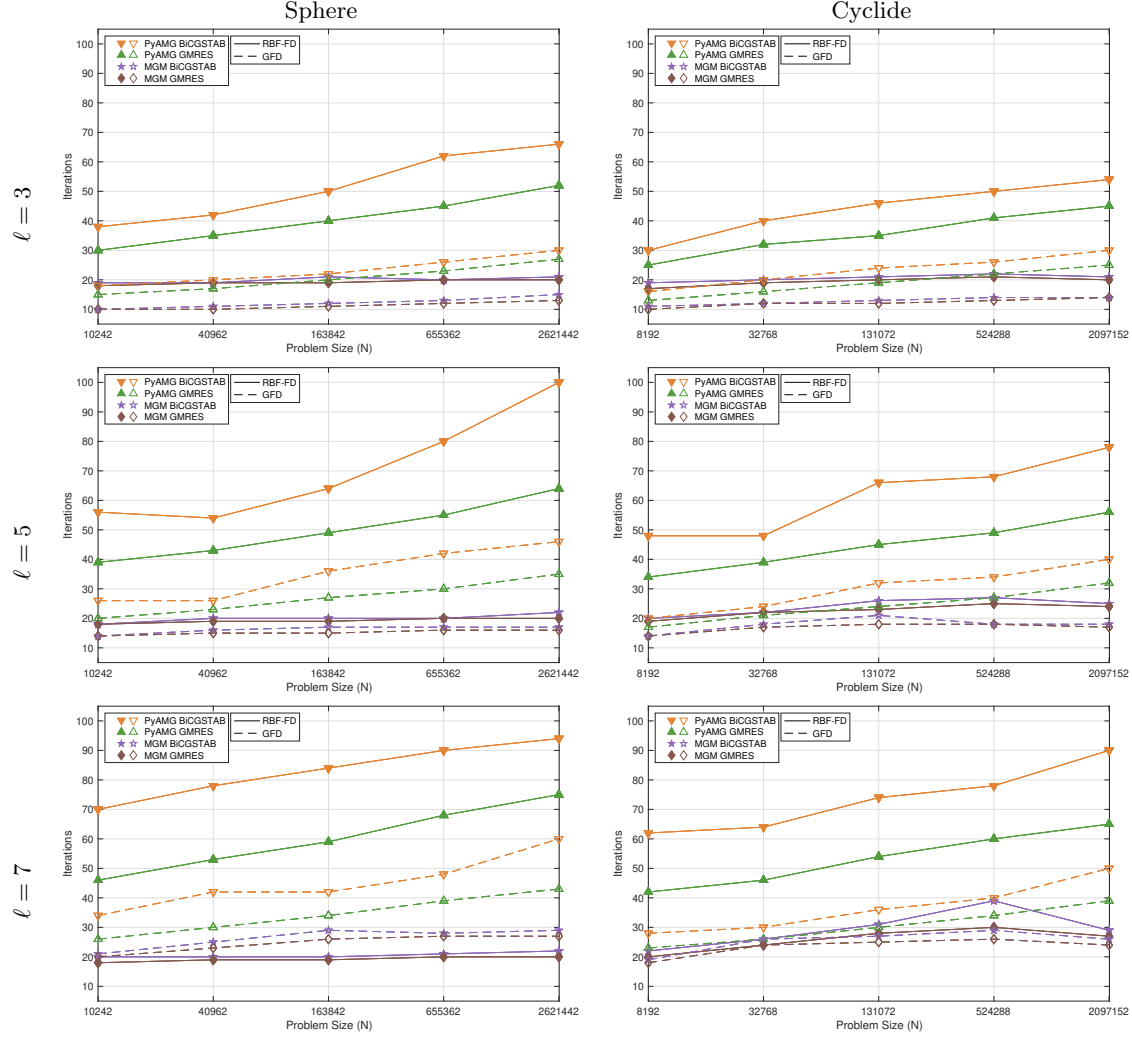
FIG. 5. *Comparison of the number of PyAMG and MGM preconditioned GMRES/BiCGSTAB iterations required to reach a relative residual tolerance of 10^{-12} for solving the shifted Poisson problem ($\mu = 1$) as a function of the point cloud size N_h (marked as N in the plots). Filled markers and solid lines mark the results for RBF-FD and open markers and dashed lines mark the results for GFD.*

we have included this plot to show how the timing of each method scales with N_h. The dotted line in these plots marks perfect linear scaling and we see that both methods have a very similar slope to this line. We also note that the timing results for BiCGSTAB follow a similar trend to GMRES, so we omitted displaying the results. For reference, the preprocessing times (in seconds) for MGM for $\ell = 3$ on the cyclide were 0.30, 0.62, 1.6, 6.1, and 31 for $N_h = 2^{13}$, 2^{15}, 2^{17}, 2^{19}, and 2^{21}, respectively; comparable times were noted for other ℓ and for the sphere with roughly the same values of N_h. We note that the preprocessing step is dominated by the construction of the restriction and interpolation operators and that this is a pleasingly parallel process that we did not fully exploit.
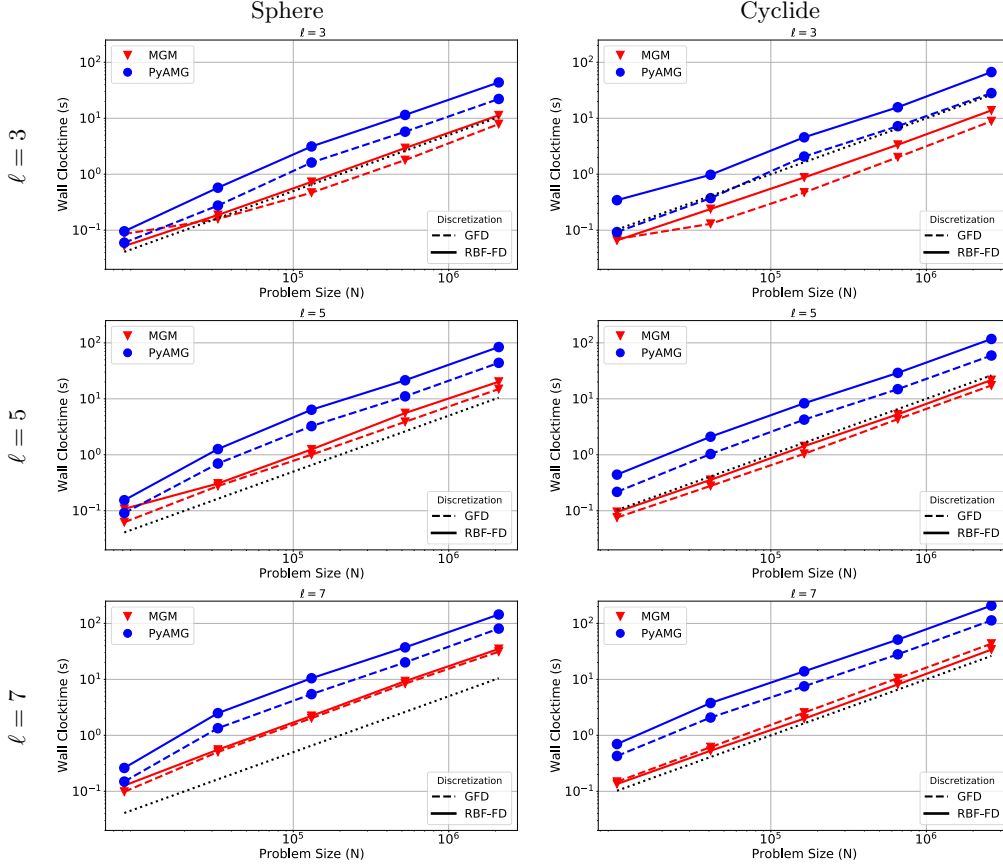
Fig. 6. *Wall-clock time (in seconds) for PyAMG GMRES and MGM GMRES to converge to a relative residual of 10^{-12} as the problem size N_h (marked as N in the plots) increases for solving the shifted Poisson problem ($\mu = 1$). The black dotted line marks linear scaling, $\mathcal{O}(N_h)$, for reference.*

5.3. Spectrum analysis. The previous two sections showed the preconditioned MGM methods outperforming the PyAMG methods. To better understand these results, we investigate the spectrum (eigenvalues) of the preconditioned matrices from both methods. Letting M_h denote the matrix representation for applying one V-cycle of either MGM or PyAMG, we can write the (right) preconditioned system as $L_h M_h z^h = f^h$, where $z^h = (M_h)^{-1} u^h$. The convergence behavior of Krylov methods can be understood by analyzing the spectrum $L_h M_h$. As discussed in, for example [38], the more clustered this spectrum is to one, the faster the Krylov methods will converge. In Figure 7 we display the complete spectrum of the preconditioned matrix $L_h M_h$ of both MGM and PyAMG for the RBF-FD discretizations on the sphere and cyclide. Due to the cost of this eigenvalue computation, we were only able to compute the results for with $N_h = 10242$ and 8192, respectively. We see from the figure that spectra for MGM are more clustered around one than PyAMG for both surfaces and increasing ℓ, which explains the better iteration counts in Table **??**. We omit the results for GFD, but note that the spectra were similar to RBF-FD, but were even more clustered near one.

5.4. Iteration vs. accuracy. In the final set of tests we focus on solving the (discretized) Poisson problem with MGM GMRES and examine how the accuracy of the RBF-FD and GFD discretizations depend on the iteration count for increasing N_h and ℓ. We restrict our attention to
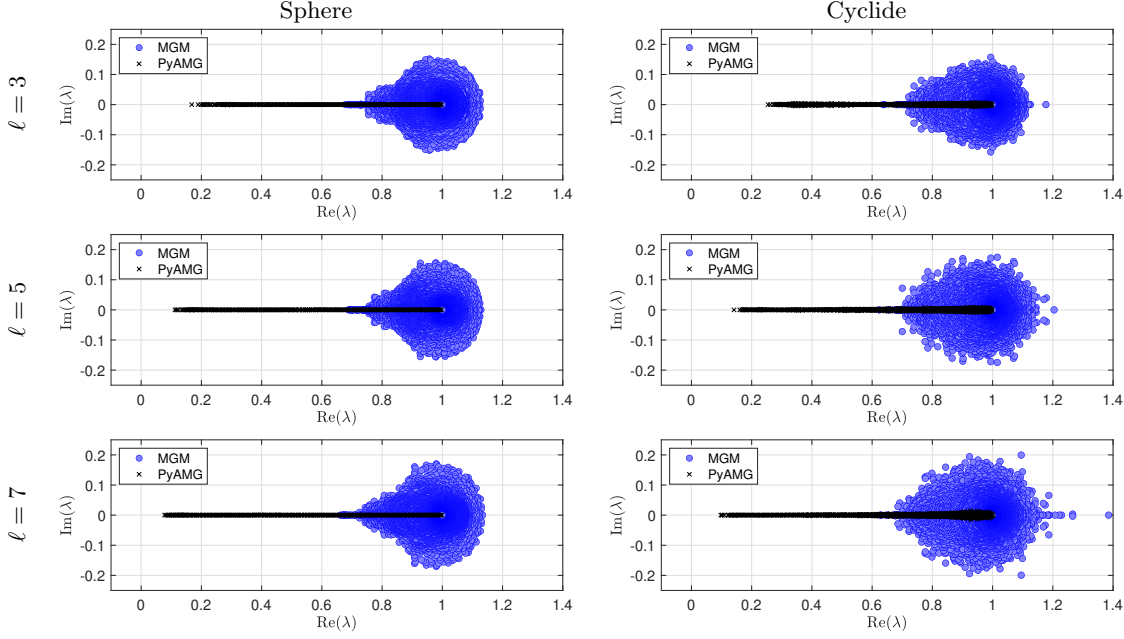
FIG. 7. *The spectra of the preconditioned matrix for shifted Poisson problem ($\mu = 1$) discretized with RBF-FD.*

the sphere, for which it is easy to construct test problems with exact solutions based on spherical harmonics. For the test problem in the experiments, we use the Y_5^4 spherical harmonic, which can be written in Cartesian coordinates as $Y_5^4(x, y, z) = z(x^4 - 6x^2y^2 + y^4)$. We fix the number of iterations of MGM GMRES for solving the discretized systems to $1, 5, 10, 15, 20, 25, 30$, and compute both the relative residual and relative errors (in the 2-norm) in the approximate solutions. Figure 8 displays the results from these experiments. We see from the figure that in almost all cases the minimum error for either the RBF-FD and GFD is reached before the minimal residual is reached. Additionally, the results indicate that while the residuals for GFD converge faster than RBF-FD, the errors for a given N_h and ℓ are smaller for RBF-FD. So the cost per error for both methods is much more comparable than the previous experiments indicated and favor RBF-FD.

6. Applications. In this section we demonstrate the performance of MGM on three different applications involving complicated surfaces represented by relatively large point clouds; see Figure 9. All these applications involve solving discrete (shifted) surface Poisson problems, for which we use the RBF-FD method to approximate the LBO and MGM GMRES to solve the resulting linear systems.

6.1. Surface harmonics. We first consider approximating the first several eigenvalues and eigenfunctions of the LBO on the Chinese Guardian Lion model. The eigenfunctions of the LBO or the "surface harmonics" have been used in various applications in data analysis. For example, Reuter et. al. [41] used the low frequency surface harmonics for shape segmentation and registration.

The LBO eigenvalue problem is given as $\Delta_{\mathcal{M}} u = \lambda u$. To approximate the solutions of this problem we use the RBF-FD method with $\ell = 5$ to approximate the LBO and ARPACK [28] (accessed through the `eigs` function in MATLAB) to solve the discrete system for the first several eigenpairs that are smallest in magnitude. ARPACK uses the Arnoldi method on the shifted inverse of a matrix to find the eigenpairs closest to the shift σ, which, for the surface problem, requires a routine for repeatedly solving systems of the form $(L_h - \sigma I_h)v^h = f^h$, for different f^h. We use
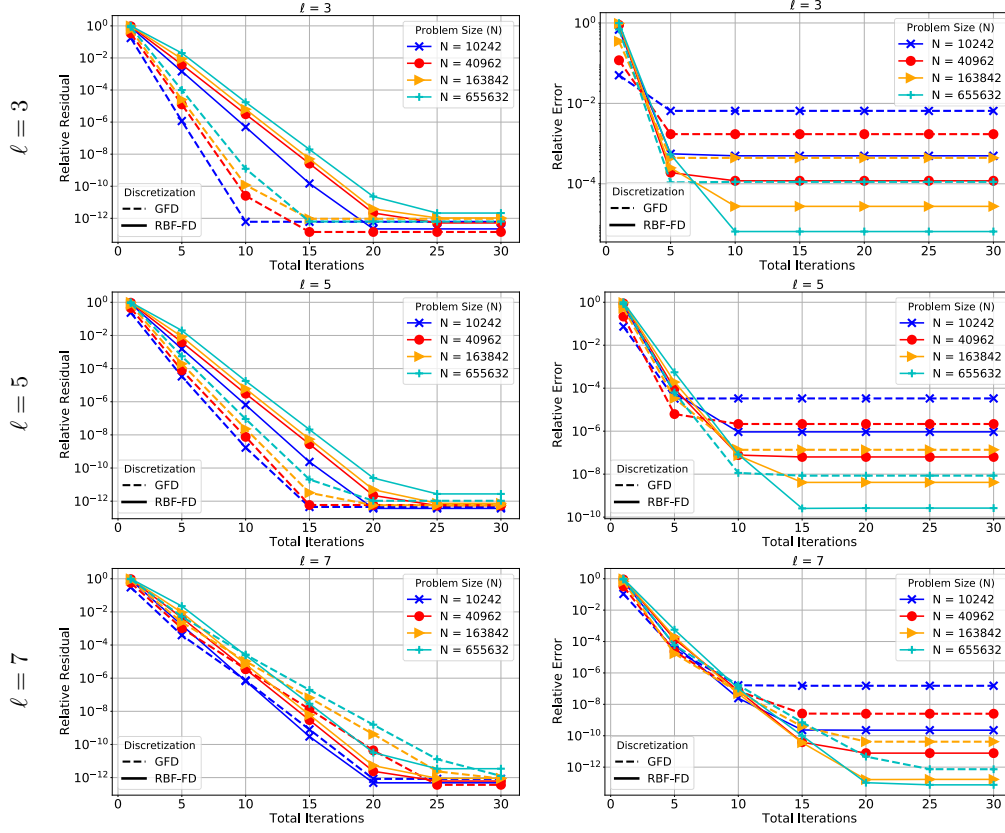
FIG. 8. *Relative residuals (left) and relative 2-norm errors (right) for solving a Poisson problem on the sphere with MGM GMRES. Solid lines correspond to RBF-FD discretizations, while dashed lines correspond to GFD.*
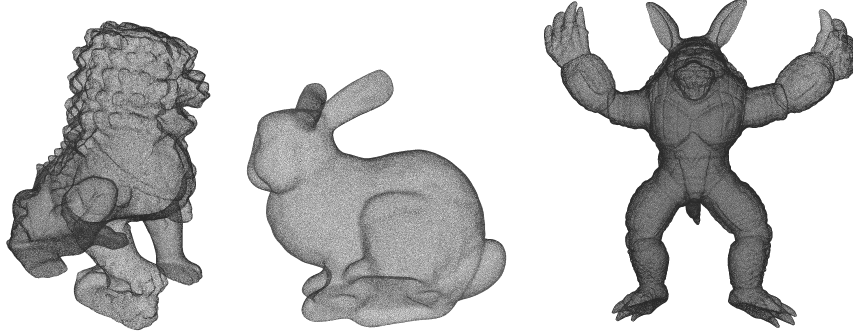

FIG. 9. *Point clouds for the surfaces considered in the applications: Chinese Guardian Lion ($N_h = 436605$), Stanford Bunny ($N_h = 291804$), and Armadillo ($N_h = 872773$).*

MGM GMRES to solve these linear systems with $\sigma = -1$ and set the tolerance to 10^{-10}. Figure 10 displays the first 10 non-zero harmonics computed with this technique. The ARPACK routine used 49 linear system solves to determine the eigenpairs; the median number of MGM GMRES iterations required to solve these systems was only 15 and the max was 16.
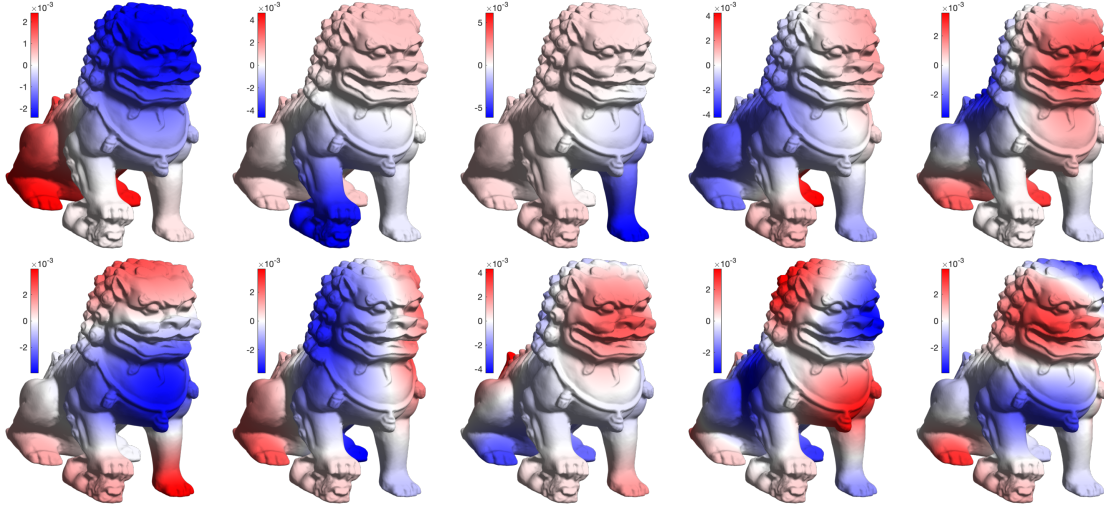
FIG. 10. *Left: pseudocolor map of the first 10 non-zero surface harmonics of the Chinese Guardian Lion model.*

6.2. Pattern formation. We next consider solving two coupled reaction-diffusion (RD) equations on the Stanford Bunny model. These types of equations arise, for example, in phenomenological models of color patterns in animal coats [35]. We consider the Gierer-Meinhardt two-species RD system [21] given as follows:

$$\frac{\partial u}{\partial t} = D_u \Delta_{\mathcal{M}} u + A - Bu + \frac{u^2}{v(1 + Cu^2)}, \tag{6.1a}$$

$$\frac{\partial v}{\partial t} = D_v \Delta_{\mathcal{M}} v + u^2 - v. \tag{6.1b}$$

By altering the parameters A, B, C, D_u, and D_v appropriately, this system can produce solutions that converge to spot or labyrinth patterns at "steady-state" [35]. For the bunny model, we set $A = 0.08$, $B = 1.5$, $C = 0.45$, $D_u = 5 \times 10^{-5}$, and $D_v = 10^{-3}$ to produce the labyrinth pattern. We use a random initial condition, where at each point in X_h the values of u and v are selected from a uniformly random distribution in the interval $[0, 1]$.

To approximate the solution of (6.1) we use the RBF-FD method with $\ell = 3$ to approximate the LBO and apply the second-order accurate semi-implicit backward difference scheme (SBDF2) [3] as the time-stepping method that treats the diffusion implicitly and reactions explicitly. We set the time-step to $\Delta t = 0.05$. The temporal discretization results in two decoupled (discrete) screened Poisson problems that need to be solved at each time-step for which we use GMRES preconditioned with MGM. For the GMRES method we set the tolerance on the relative residual to 10^{-8} and use the previous time-step as the initial guess. We set the final integration to 300 time units, which resulted in a near steady-state pattern. Figure 11 displays the results of the simulations. Included in the figure are the iterations required by the preconditioned GMRES method as a function of time. We see from the figure that the maximum iteration count is 6 for the u variable and 11 for the v variable, and decreases to 3 and 4, respectively as the solutions approach steady-state. The larger iteration count for the v variable is expected since the diffusion coefficient is larger in (6.1b).

6.3. Geodesic distance. Lastly, we consider the classic problem of approximating the geodesic distance from a given point on a surface to all other points. We use the *heat method* introduced by Crane et. al. [11] to solve this problem. This method transforms the non-linear geodesic distance problem, typically formulated in terms of the eikonal equation, into solving a pair of linear parabolic
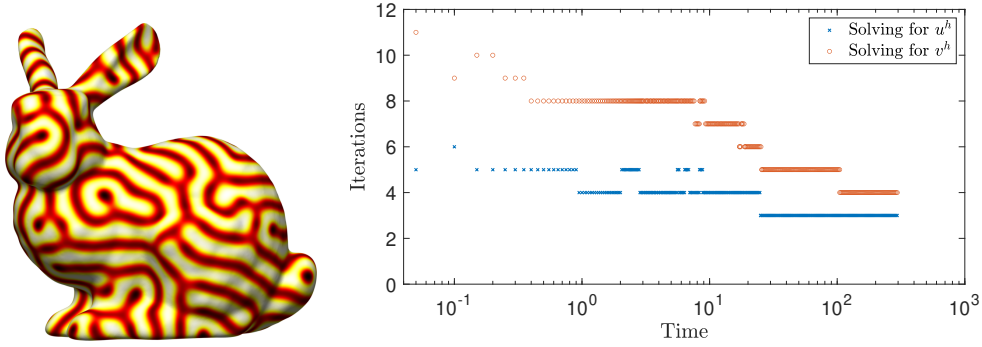
FIG. 11. *Left: pseudocolor map of the u variable in the numerical solution of* (6.1) *on the Stanford Bunny model; the colors transition from white to yellow to red to black, with white corresponding to $u = 0$ and black to $u = 1$. Right: iteration count of MGM GMRES for solving the linear systems associated with u and v variables at each time-step in the semi-implicit scheme for* (6.1).

and elliptic problems. The heat method is comprised of the three steps:

1. Solve $u_t = \Delta_{\mathcal{M}} u$, with $u_0 = \delta(\mathbf{x}^*)$, to some time $t_{\text{final}} > 0$
2. Compute the vector field $\boldsymbol{\eta} = -\nabla_{\mathcal{M}} u / |\nabla_{\mathcal{M}} u|$
3. Solve the Poisson problem $\Delta_{\mathcal{M}} \varphi = \nabla_{\mathcal{M}} \cdot \boldsymbol{\eta}$

Here $\mathbf{x}^*$ denotes the target point on the surface $\mathcal{M}$ to compute the distance from, $\nabla_{\mathcal{M}}$ denotes the surface gradient, $\nabla_{\mathcal{M}} \cdot$ is the surface divergence, and δ denotes the Dirac delta function. As discussed in [11], the function φ approximates the geodesic distance and converges to the exact distance as $t_{\text{final}} \to 0$.
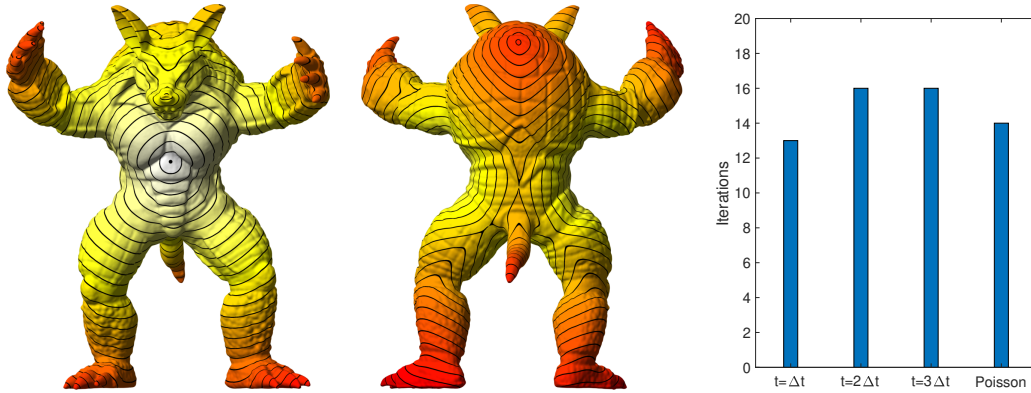

FIG. 12. *Left: pseudocolor map of the approximate geodesic distance from the solid circle on the chest of the Armadillo model (viewed from the front and backside) computed with the heat method. Solid black lines mark the contours of the distance field and the colors transition from white to yellow to red with increasing distance from the solid circle. Right: iteration count of GMRES preconditioned with MGM for solving the linear systems associated with the heat method.*

We apply the heat method on the Armadillo model. We again use the RBF-FD method with $\ell = 3$ to approximate the LBO in steps 1 and 3 above. To approximate the surface gradient and divergence, we also use the RBF-FD method formulated in the tangent plane similar to the method described in [48] for GFD. For these approximations, we use $\ell = 2$, which result in a second-order approximation. We discretize the heat equation in the first step with backward Euler in time with a time-step of $\Delta t = 10^{-3}$ and set $t_{\text{final}} = 3\Delta t$. To solve the linear systems associated with

this implicit discretization and the system from the discretized Poisson equation in step 3, we use GMRES preconditioned with MGM, setting the tolerance to 10^{-8}. The results for a point $\mathbf{x}^*$ on the chest of the Armadillo are displayed in the first to images of Figure 12. The last image in this figure displays the iterations of the preconditioned GMRES method for solving the systems from the heat equation discretization for three time-steps and the Poisson system to determine φ. We see that the iteration count remains low for all these systems.

7. Concluding remarks. We have presented a new geometric multilevel method, MGM, for solving linear systems associated with discretizations of elliptic PDEs on point clouds. The method is entirely meshfree and uses the WSE algorithm for coarsening the point clouds, interpolation/restrictions operators based on polyharmonic spline RBFs, Galerkin coarsening of the operator, and standard smoothers. All of these choices make MGM particularly straightforward to implement. We numerically analyzed the method as a standalone solver and preconditioner on test problems for the sphere and cyclide discretized using RBF-FD and GFD methods, and found that it compares favorably to AMG methods in terms of convergence rates and wall-clock time. When using MGM as a preconditioner, we also found that it scaled well as both the problem size and accuracy of the discretizations increased. Finally, we demonstrated that the method can be used in three challenging applications involving large systems of equations. A full open source implementation of the method is planned with code to reproduce the MGM results.

There are several extensions of MGM that we plan to pursue in the future. One is to test the method on other discretizations. MGM is agnostic to the underlying discretization and could be used even for (nodal) mesh-based discretizations. Here the nodal points of the mesh could be treated as a point cloud and WSE could be applied, or if there is a natural way to coarsen the mesh, then this could be used instead. A second idea we plan to pursue is extending MGM to domains with boundaries, which in principle should be straightforward. Finally, we plan to look into parallel implementations of the method to further improve the performance.

REFERENCES

[1] M. F. ADAMS, *Algebraic multigrid methods for constrained linear systems with applications to contact problems in solid mechanics*, Numerical Linear Algebra with Applications, 11 (2004), pp. 141–153.

[2] D. ÁLVAREZ, P. GONZÁLEZ-RODRÍGUEZ, AND M. KINDELAN, *A local radial basis function method for the Laplace–Beltrami operator*, J. Sci. Comput., 86 (2021), p. 28, https://doi.org/10.1007/s10915-020-01399-3.

[3] U. M. ASCHER, S. J. RUUTH, AND B. T. R. WETTON, *Implicit-explicit methods for time-dependent pdes*, SIAM J. Num. Anal, 32 (1997), pp. 797–823.

[4] G. BALLARD, C. SIEFERT, AND J. HU, *Reducing communication costs for sparse matrix multiplication within algebraic multigrid*, SIAM J. Sci. Comput., 38 (2016), pp. C203–C231, https://doi.org/10.1137/15M1028807.

[5] V. BAYONA, N. FLYER, B. FORNBERG, AND G. A. BARNETT, *On the role of polynomials in RBF-FD approximations: II. numerical solution of elliptic PDEs*, J. Comput. Phys., 332 (2017), pp. 257–273.

[6] N. BELL, S. DALTON, AND L. N. OLSON, *Exposing fine-grained parallelism in algebraic multigrid methods*, SIAM J. Sci. Comput., 34 (2012), pp. C123–C152, https://doi.org/10.1137/110838844.

[7] M. BENZI, G. H. GOLUB, AND J. LIESEN, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137, https://doi.org/10.1017/S0962492904000212.

[8] M. BERTALMÍO, L. CHENG, S. OSHER, AND G. SAPIRO, *Variational problems and partial differential equations on implicit surfaces*, J. Comput. Phys., 174 (2001), pp. 759–780.

[9] R. BRIDSON, *Fast poisson disk sampling in arbitrary dimensions*, in ACM SIGGRAPH 2007 Sketches, SIGGRAPH '07, New York, NY, USA, 2007, ACM, https://doi.org/10.1145/1278780.1278807.

[10] Y. Chen and C. B. Macdonald, *The closest point method and multigrid solvers for elliptic equations on surfaces*, SIAM Journal on Scientific Computing, 37 (2015), pp. A134–A155.

[11] K. Crane, C. Weischedel, and M. Wardetzky, *Geodesics in heat: A new approach to computing distance based on heat flow*, ACM Transactions on Graphics (TOG), 32 (2013), pp. 1–11.

[12] O. Davydov and R. Schaback, *Optimal stencils in sobolev spaces*, IMA J. Num. Analy., 39 (2019), pp. 398–422.

[13] L. Demanet, *Painless, highly accurate discretizations of the Laplacian on a smooth manifold*, Technical report, Stanford University, (2006).

[14] J. Duchon, *Splines minimizing rotation-invariant semi-norms in Sobolev space*, in Constructive Theory of Functions of Several Variables. Lecture Notes in Mathematics (vol 571), W. Schempp and Z. K., eds., Springer, Berlin, 1977, pp. 85–100, https://doi.org/10.1007/BFb0086566.

[15] 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., Lect. Notes Math., v. 1357, Springer, Berlin, 1988.

[16] G. Dziuk and C. M. Elliott, *Finite element methods for surface pdes*, Acta Numerica, 22 (2013), pp. 289–396.

[17] P. Farrell and H. Wendland, *RBF multiscale collocation for second order elliptic boundary value problems*, SIAM J. Numer. Anal., 51 (2013), pp. 2403–2425.

[18] G. E. Fasshauer, *Solving differential equations with radial basis functions: multilevel methods and smoothing*, Adv. Comp. Math., 11 (1999), pp. 139–159.

[19] E. Fuselier and G. B. Wright, *Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates*, SIAM J. Numer. Anal., 50 (2012), pp. 1753–1776.

[20] A. Ghai, C. Lu, and X. Jiao, *A comparison of preconditioned Krylov subspace methods for large-scale non-symmetric linear systems*, Numerical Linear Algebra with Applications, 26 (2018).

[21] A. Gierer and H. Meinhardt, *A theory of biological pattern formation*, Kybernetik, 12 (1972), pp. 30–39.

[22] B. J. Gross, N. Trask, P. Kuberry, and P. J. Atzberger, *Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: A generalized moving least-squares (GMLS) approach*, J. Comput. Phys., 409 (2020), p. 109340, https://doi.org/https://doi.org/10.1016/j.jcp.2020.109340.

[23] D. Gunderman, N. Flyer, and B. Fornberg, *Transport schemes in spherical geometries using spline-based RBF-FD with polynomials*, J. Comput. Phys., 408 (2020), p. 109256.

[24] A. Katz and A. Jameson, *Multicloud: Multigrid convergence with a meshless operator*, J. Comput. Phys., 228 (2009), pp. 5237–5250.

[25] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, *Comparison of surface normal estimation methods for range sensing applications*, in IEEE Int. Conference on Robotics and Automation, 2009, pp. 3206–3211.

[26] C. Landsberg and A. Voigt, *A multigrid finite element method for reaction-diffusion systems on surfaces*, Computing and visualization in science, 13 (2010), pp. 177–185.

[27] Q. T. Le Gia, I. H. Sloan, and H. Wendland, *Multiscale RBF collocation for solving PDEs on spheres*, Numer. Math., 121 (2012), pp. 99–125.

[28] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, SIAM, Philadelphia, PA, USA, 1998.

[29] 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. A219–A2151.

[30] J. Liang and H. Zhao, *Solving partial differential equations on point clouds*, SIAM J. Sci. Comput., 35 (2013), pp. A1461–A1486, https://doi.org/10.1137/120869730.

[31] J. Liu, J. F. Totz, P. W. Miller, A. D. Hastewell, Y.-C. Chao, J. Dunkel, and N. Fakhri, *Topological braiding and virtual particles on the cell membrane*, Proc. Natl. Acad. Sci. U.S.A., 118 (2021).

[32] 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 (2009), pp. 4330–4350.

[33] D. Majewski, D. Liermann, P. Prohl, B. Ritter, M. Buchhold, T. Hanisch, G. Paul, W. Wergen, and J. Baumgardner, *The operational global icosahedral–hexagonal gridpoint model GME: description and high-resolution tests*, Mon. Weather Rev., 130 (2002), pp. 319–338.

[34] T. Manteuffel and B. S. Southworth, *Convergence in norm of nonsymmetric algebraic multigrid*, SIAM Journal on Scientific Computing, 41 (2019), pp. S269–S296, https://doi.org/10.1137/18M1193773.

[35] S. Miyazawa, M. Okamoto, and S. Kondo, *Blending of animal colour patterns by hybridization*, Nature communications, 1 (2010), pp. 1–6.

[36] M. A. Olshanskii and A. Reusken, *Trace finite element methods for pdes on surfaces*, in Geometrically unfitted finite element methods and applications, Springer, 2017, pp. 211–258.

[37] L. N. Olson and J. B. Schroder, *PyAMG: Algebraic multigrid solvers in Python v4.0*, 2018, https://github.com/pyamg/pyamg. Release 4.0.

[38] C. W. Oosterlee and T. Washio, *An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems*, SIAM Journal on Scientific Computing, 19 (1998), pp. 87–110.

[39] C. W. Oosterlee, R. Wienands, T. Washio, and F. J. Gaspar, *The acceleration of multigrid convergence by recombination techniques*, in Multigrid methods, VI (Gent, 1999), vol. 14 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2000, pp. 34–43.

[40] C. Piret and J. Dunn, *Fast RBF OGr for solving PDEs on arbitrary surfaces*, AIP Conference Proceedings,

1776 (2016), p. 070005, https://doi.org/10.1063/1.4965351.

[41] M. REUTER, S. BIASOTTI, D. GIORGI, G. PATANÈ, AND M. SPAGNUOLO, *Discrete Laplace–Beltrami operators for shape analysis and segmentation*, Comput. Graph., 33 (2009), pp. 381–390.

[42] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.

[43] B. SEIBOLD, *Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods*, Numerical Linear Algebra with Applications, 17 (2010), pp. 433–451.

[44] V. SHANKAR, A. NARAYAN, AND R. M. KIRBY, *RBF-LOI: Augmenting radial basis functions (RBFs) with least orthogonal interpolation (LOI) for solving PDEs on surfaces*, J. Comput. Phys., 373 (2018), pp. 722–735.

[45] 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 (2014), pp. 745–768, https://doi.org/10.1007/s10915-014-9914-1.

[46] S. SHAW, *Radial basis function finite difference approximations of the Laplace-Beltrami operator*, master's thesis, Boise State University, USA, 2019, https://doi.org/10.18122/td/1587/boisestate. 1587.

[47] N. STOOP, R. LAGRANGE, D. TERWAGNE, P. M. REIS, AND J. DUNKEL, *Curvature-induced symmetry breaking determines elastic surface patterns*, Nature materials, 14 (2015), pp. 337–342.

[48] P. SUCHDE AND J. KUHNERT, *A meshfree generalized finite difference method for surface PDEs*, Comp. Math. Appl., 78 (2019), pp. 2789–2805, https://doi.org/https://doi.org/10.1016/j.camwa.2019.04.030.

[49] N. TRASK AND P. KUBERRY, *Compatible meshfree discretization of surface PDEs*, Computational Particle Mechanics, 7 (2020), pp. 271–277, https://doi.org/10.1007/s40571-019-00251-2.

[50] U. TROTTENBERG, C. W. OOSTERLEE, AND A. SCHULLER, *Multigrid*, Elsevier, 2000.

[51] G. TURK, *Generating textures on arbitrary surfaces using reaction-diffusion*, Comput. Graph., 25 (1991), pp. 289–298.

[52] H. WENDLAND, *Scattered data approximation*, Cambridge University Press, 2004.

[53] H. WENDLAND AND J. KÜNEMUND, *Solving partial differential equations on (evolving) surfaces with radial basis functions*, Advances in Computational Mathematics, 46 (2020), p. 64.

[54] P. WESSELING, *An Introduction to Multigrid Methods*, John Wiley & Sons, New York, 1992.

[55] D. L. WILLIAMSON, *The evolution of dynamical cores for global atmospheric models*, J. Meteorol. Soc. Jpn., 85B (2007), pp. 241–269.

[56] M. YOON, G. YOON, AND C. MIN, *On solving the singular system arisen from Poisson equation with Neumann boundary condition*, J. Sci. Comput., 69 (2016), pp. 391–405.

[57] C. YUKSEL, *Sample elimination for generating Poisson disk sample sets*, Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015), 34 (2015), pp. 25–32, https://doi.org/10.1111/cgf.12538.

[58] C. YUKSEL, *cyCodeBase*, 2021, https://github.com/cemyuksel/cyCodeBase/.

[59] R. ZAMOLO, E. NOBILE, AND B. ŠARLER, *Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from RBF-FD meshless discretizations*, J. Comput. Phys., 392 (2019), pp. 311–334.