SIAM J. SCI. COMPUT.
Vol. 42, No. 6, pp. A3584–A3609


# SCHWARZ SOLVERS AND PRECONDITIONERS FOR THE CLOSEST POINT METHOD*

IAN C. T. MAY†, RONALD D. HAYNES‡, AND STEVEN J. RUUTH†

**Abstract.** The discretization of surface intrinsic elliptic partial differential equations (PDEs) poses interesting challenges not seen in flat spaces. The discretization of these PDEs typically proceeds by either parametrizing the surface, triangulating the surface, or embedding the surface in a higher dimensional flat space. The closest point method (CPM) is an embedding method that represents surfaces using a function that maps points in the embedding space to their closest points on the surface. In the CPM, this mapping also serves as an extension operator that brings surface intrinsic data onto the embedding space, allowing PDEs to be numerically approximated by standard methods in a narrow tubular neighborhood of the surface. We focus on numerically approximating the positive Helmholtz equation, $(c - \Delta_{\mathcal{S}})\, u = f,\ c \in \mathbb{R}^+$, by the CPM paired with finite differences. This yields a large, sparse, and nonsymmetric system to be solved. Herein, we develop restricted additive Schwarz (RAS) and optimized restricted additive Schwarz (ORAS) solvers and preconditioners for this discrete system. In particular, we develop a general strategy for computing overlapping partitions of the computational domain and defining the corresponding Dirichlet and Robin transmission conditions. We demonstrate that the convergence of the ORAS solvers and preconditioners can be improved by using a modified transmission condition where more than two overlapping subdomains meet. Numerical experiments are provided for a variety of analytical and triangulated surfaces. We find that ORAS solvers and preconditioners outperform their RAS counterparts and that, as expected using domain decomposition (DD) as a preconditioner rather than as a solver gives faster convergence. The methods exhibit good parallel scalability over the range of process counts tested.

**Key words.** domain decomposition, optimized Schwarz method, closest point method, Laplace–Beltrami

**AMS subject classifications.** 65F10, 65N22, 65N55

**DOI.** 10.1137/19M1288279

## 1. Introduction.

Elliptic partial differential equations (PDEs) can lead to large coupled systems of equations upon discretization. Care is required in solving such systems to maintain reasonable memory requirements and to yield solutions in a timely manner. The focus here is on the efficient numerical solution of elliptic PDEs intrinsic to surfaces. As a representative model problem, we consider the surface intrinsic positive Helmholtz equation,

$$(c - \Delta_{\mathcal{S}})\, u = f, \tag{1.1}$$

where $\Delta_{\mathcal{S}}$ denotes the Laplace–Beltrami operator associated with the surface $\mathcal{S} \subset \mathbb{R}^d$, and $c \in \mathbb{R}^+$ is a positive constant. Such systems also appear for certain time-discretized diffusion and reaction-diffusion phenomena intrinsic to surfaces [22]. This equation also arises when finding the eigenvalues of the Laplace–Beltrami operator [21], a problem that occurs in shape classification [29] and other applications.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 19, 2019; accepted for publication (in revised form) August 26, 2020; published electronically November 9, 2020.

https://doi.org/10.1137/19M1288279

**Funding:** This work was supported by NSERC Canada through grants RGPIN 2016-04361 and RGPIN 2018-04881, by ACEnet (ace-net.ca), and by Compute Canada.

†Department of Mathematics, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada (mayianm@sfu.ca, sruuth@sfu.ca).

‡Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, Newfoundland A1C 5S7, Canada (rhaynes@mun.ca).

Several methods have been proposed for the discretization of surface intrinsic PDEs. Parametrizing the surface leads to efficient methods [14] when the parametrization is available or can be approximated. Finite element methods operating on a triangulation of the surface [13] lead to sparse and symmetric systems of equations, but the approach is sensitive to the quality of the triangulation. Embedding methods seek to place the surface and the discretization in a higher dimensional flat space, $\mathbb{R}^d$, where more familiar numerical methods are available. The surface can be posed as a level set of a function in the embedding space where finite differences in $\mathbb{R}^d$ may be used [5], though the representation of open surfaces and the imposition of artificial boundary conditions at the boundary of the computational domain are nontrivial.

The closest point method (CPM) [30, 23] is an embedding method which extends values on the surface into the embedding space via a mapping from points in the embedding space to their closest points on the surface. The discretization of the surface intrinsic PDE can then be achieved by a standard flat-space method. The efficient solution of the system of equations produced by this method as applied to (1.1) is the focus of this paper. The CPM allows for the discretization over a wide range of surfaces with the same implementation. Triangulated surfaces [20], surfaces of varying codimension [30], and moving surfaces [27] all can be treated by the CPM. Section 2.1 recalls the CPM and establishes the notation and conventions used herein.

Domain decomposition (DD) methods have proven to be a powerful approach for the efficient solution of systems of linear equations, especially those arising from elliptic PDEs [28, 34]. In these methods, the large problem of interest is decomposed into many smaller and computationally cheaper subproblems. These subproblems can be rooted in purely algebraic formulations or obtained from continuous subproblems that can then be discretized. The solutions of these subproblems are assembled together to approximate the solution of the targeted large system, the accuracy of which is then iteratively improved to achieve a given tolerance [32, 11]. Furthermore, this procedure makes an excellent preconditioner for Krylov methods, such as GMRES [32, 11].

In many DD schemes these smaller subproblems can be decoupled and solved in parallel. A prior paper from the authors of this article showed some preliminary results on the application of classical Schwarz type DD solvers and preconditioners to the CPM [26]. The present work focuses on additive Schwarz type solvers and preconditioners that allow for parallelization, and, in particular, on optimized Schwarz methods which use enhanced boundary, or transmission, conditions on the subproblems to accelerate convergence of the iteration. This work introduces a boundary alignment procedure (section 3.2), presents the use of multiple Robin weights in the optimized restricted additive Schwarz (ORAS) method (section 5.2.2), and evaluates the utility of the developed methods as preconditioners for GMRES. These extensions over the previous work are accompanied by substantially more numerical tests. The result is a more robust and general method.

The CPM and DD schemes are reviewed briefly in section 2. Section 3 considers the desirable properties of the disjoint subdomains and how they may be created. Section 4 builds an overlapping decomposition from these disjoint subdomains and addresses some concerns that arise from the atypical geometry of the problem. Then, section 5 completes the formulation of the solution scheme by placing appropriate boundary conditions on the now known subdomains. Section 6 evaluates the performance of the method by first sweeping over the introduced parameters and then testing the parallel scalability of the method for a fixed set of parameters.

**2. Background.** The CPM and additive Schwarz type DD methods are presented here to highlight important considerations and establish notation.

**2.1. The closest point method.** The CPM allows for the discretization of surface intrinsic differential operators using standard Cartesian methods by first extending the solution into a narrow region surrounding the surface. By keeping the solution constant in the surface normal direction, the gradient, divergence, and Laplacian over the embedding space all recover their surface intrinsic values when restricted to the surface [30]. To make this discussion concrete, we formulate a discretization of (1.1) by the CPM paired with finite differences over $\mathbb{R}^d$.

Consider a surface, $\mathcal{S}$, embeddable in $\mathbb{R}^d$. Define the closest point function for $\mathcal{S}$ as

$$CP_{\mathcal{S}} : \mathbb{R}^d \to \mathcal{S}, \qquad x \mapsto \arg\min_{y \in \mathcal{S}} ||x - y||_2, \tag{2.1}$$

which identifies the closest point on the surface, $y$, for any point, $x$, in the embedding space. To avoid introducing discontinuities into $CP_{\mathcal{S}}$, we assume that the computational domain, $\Omega_D$, has been chosen to consist only of points within a distance $\kappa_\infty^{-1}$ of the surface $\mathcal{S}$, where $\kappa_\infty$ is an upper bound on the curvatures of $\mathcal{S}$ [8].

A finite difference discretization of a surface PDE by the CPM is obtained by specifying a finite difference stencil to approximate the flat-space differential operator and a degree, $p$, for the interpolating polynomials forming the discrete extension operator. We place a uniform grid with spacing $h$ in a suitable neighborhood of the surface (this will be made precise later). Assuming sufficient smoothness of $f$ and $CP_{\mathcal{S}}$ [24], values of $f$ on the surface may be extended off the surface in the normal direction to $\mathcal{O}\left(h^{p+1}\right)$ accuracy by evaluating a tensor product barycentric Lagrangian interpolant [4].

To illustrate the procedure, Figure 1 shows an extension stencil in a $d = 2$ dimensional embedding space using bicubic interpolation ($p = 3$). In the figure, the $i$th point in the grid, $x_i$, is identified, along with its closest point, $CP_{\mathcal{S}}(x_i)$, and the $(3+1)^2$ points surrounding it in the embedding space. The $i$th row of $\mathbf{E}$ will thus have 16 nonzero entries given by the tensor product of the weights in the one dimensional stencils used to approximate $f$ at $CP_{\mathcal{S}}(x_i)$.

We are now in position to define the grid. The discrete extension operator relies only on points in a neighborhood of the surface, and as a consequence the CPM may be posed over a neighborhood of the surface. This dramatically reduces the number of points in the mesh relative to a global computation (e.g., over a computational cube encompassing the shape) and ensures that during refinement the growth of the number of points will scale with the dimension of the surface rather than the dimension of the embedding space. In this work, we collect all nodes used for interpolation (the *active nodes*) into a set $\Sigma_A$ of size $N_A$. At the edge of this set there will be active nodes lacking neighboring nodes, and thus the active nodes have incomplete finite difference stencils. We denote by $\Sigma_G$ the set of all *ghost nodes* (with $N_G = \#\Sigma_G$) needed to complete these stencils and note that the grid consists of the nodes in $\Sigma_A \cup \Sigma_G \subset \Omega_D$.

The sets $\Sigma_A$ and $\Sigma_G$ may be formed by assigning to them all nodes within specified radii of the surface [30]. We refer to this as the *tube-radius construction*. Alternatively, the sets $\Sigma_A$ and $\Sigma_G$ may be formed algorithmically, as in [23], where the active and ghost nodes are identified by the lists $L$ and $G$, respectively. This *algorithmic approach* leads to a smaller number of active nodes and thus smaller linear systems.

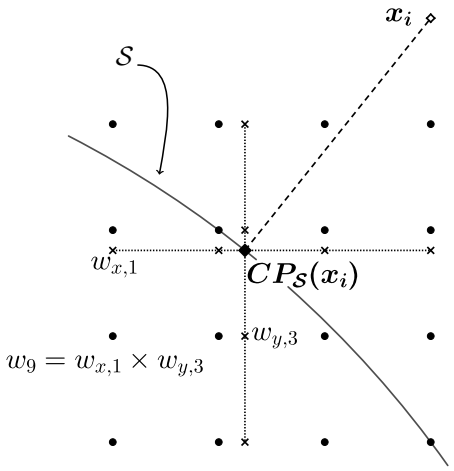


FIG. 1. *A bicubic extension stencil around a circle $\mathcal{S}$ embedded in $d = 2$ dimensional space. Grid point $x_i$ (empty diamond) has closest point $CP_{\mathcal{S}}(x_i)$ (filled diamond). The associated 16 grid point interpolation stencil surrounding $CP_{\mathcal{S}}(x_i)$ is displayed using filled circles. The weights in the one dimensional stencils, shown by crosses and dotted lines, are given by barycentric Lagrangian interpolation. The tensor product of the one dimensional interpolation weights gives the weights in the bicubic interpolation. For instance, the ninth point in the extension stencil aligns with the first point in the x-direction and the third point in the y-direction and thus has the weight $w_9 = w_{x,1} \times w_{y,3}$.*

We use the former tube-radius construction for simple surfaces with analytically defined closest point functions and use the algorithmic approach for more complicated surfaces specified by an input triangulation. It is beneficial to demonstrate that both approaches yield domains appropriate for our proposed DD methods.

A possible discretization for the Laplace–Beltrami operator over $\mathcal{S}$ may now be formed by composing the finite difference operator, $\Delta^h \in \mathbb{R}^{N_A \times (N_A+N_G)}$, with the discrete extension operator, $\mathbf{E} \in \mathbb{R}^{(N_A+N_G)\times N_A}$, containing the interpolation weights for all nodes in $\Sigma_A$ and $\Sigma_G$. This *direct discretization* of the Laplace–Beltrami operator, $\Delta^h\mathbf{E}$, is a suitable choice for explicit time-stepping of parabolic PDEs. Unfortunately, the direct discretization has several small, spurious positive eigenvalues as well as complex eigenvalues with nonnegligible imaginary parts. As a consequence, this discretization is not suitable for the numerical solution of eigenvalue problems [21]. Furthermore, the spectrum of the direct discretization severely limits the allowable time step-size of implicit time-stepping methods, rendering them inefficient compared to the simpler explicit time-stepping methods [23]. Of final importance is that use of the direct discretization tends to hinder the effectiveness of iterative solvers. One can resolve these issues by using a stabilized discretization,

$$\Delta_{\mathcal{S}}^h = -\frac{2d}{h^2} I + \left(\Delta^h + \frac{2d}{h^2} I\right) \mathbf{E}, \tag{2.2}$$

which removes redundant extensions while remaining consistent [23, 21].

The effective iterative solution of linear systems defined by this discretization of the model problem (1.1) sees two main obstacles: the matrix (2.2) is nonsymmetric even though the continuous operator is self-adjoint, and further its spectrum has complex eigenvalues even though the imaginary parts are small.

**2.2. Domain decomposition solvers and preconditioners.** There are many techniques for solving the linear systems that arise from the discretization of elliptic PDEs. If the system must be solved multiple times and the operators involved are fixed (as occurs when using an implicit time-stepper with a fixed step size on a linear problem with nonvarying coefficients), then prefactoring the matrix may be best (within time and memory constraints). Each time-step then requires only a forward and backward solve, with the initial cost of factorization amortized by each step of time evolution. However, the solution of the same systems by adaptive time-stepping methods, or the discretization of nonlinear elliptic/parabolic systems solved by a Newton iteration, yields systems that are not amenable to prefactoring. Parallelization of iterative methods is much simpler than for direct solvers, and even the repeated solution of fixed systems may be better treated by iterative solvers with the use of appropriate preconditioners [3, 31]. As such, the design of effective iterative solvers is crucial and will be our focus herein.

A powerful option for solving such systems is through the use of DD methods as solvers, or as preconditioners for Krylov methods. Here the solution of the global problem is sought by repeatedly solving many smaller problems obtained by partitioning the domain and imposing appropriate conditions on the artificial boundaries. Thorough accounts of these methods can be found in [34, 32, 11, 28] among many others. Here the focus will be on the restricted additive Schwarz method (RAS) and related methods as developed in [6, 33].

The general formulation of these methods is best illustrated for the continuous problem (1.1), and we leave the discretization for later. The global domain $\mathcal{S}$ is divided into $N_S$ disjoint subdomains, $\widetilde{\mathcal{S}}_j$, with a corresponding partitioning into $N_S$ overlapping subdomains $\mathcal{S}_j$. Define $\Gamma_{jk} := \partial\mathcal{S}_j \cap \widetilde{\mathcal{S}}_k$ as the portion of the boundary of the $j$th overlapping subdomain lying in the $k$th disjoint subdomain. Given an initial guess for the solution of the global problem $u^{(0)}$, defined at least over the boundaries of each subdomain $\mathcal{S}_j$, we can find a sequence of new solutions by solving the subproblems

$$\begin{cases} (c - \Delta_{\mathcal{S}})\, u_j = f & \text{in } \mathcal{S}_j, \\ \mathcal{T}_{jk} u_j = \mathcal{T}_{jk} u^{(n)} & \text{on } \Gamma_{jk} \quad \forall\ k \end{cases} \tag{2.3}$$

for the local solutions $u_j$. A new global solution can be constructed as follows from the restriction of the local solutions to their disjoint subdomains:

$$u^{(n+1)} = \sum_j u_j^{(n+1)}\Big|_{\widetilde{\mathcal{S}}_j}. \tag{2.4}$$

The boundary operators, $\mathcal{T}_{jk}$, transmit information between the subdomains and thus are usually called transmission operators. The operators of interest herein are

$$\mathcal{T}_{jk} = \mathit{identity} \qquad \text{or} \tag{2.5}$$

$$\mathcal{T}_{jk} = \left(\frac{\partial}{\partial \hat{q}_{jk}} + \alpha\right), \tag{2.6}$$

where $\hat{q}_{jk}$ is the outward pointing unit *conormal* vector on $\Gamma_{jk}$. The conormal is defined as the vector that is simultaneously orthogonal to the surface normal, $\hat{n}$, and the subdomain boundary, $\Gamma_{jk}$. The first option (2.5) enforces Dirichlet conditions to produce the classic RAS solver [6] when paired with the solution reconstruction in

(2.4). The second option, (2.6), enforces a Robin condition along each interface and gives the ORAS family of solvers [33].

Following the notation for the continuous problems, we take $\Sigma_A$, $\widetilde{\Sigma}_j$, and $\Sigma_j$ as the global set, the disjoint subsets, and the overlapping subsets, respectively, of the active nodes in the discretization. Equation (2.2) is gathered into the matrix $\mathbf{A}$ so that $\mathbf{Au} = \mathbf{f}$ corresponds to a discretization of (1.1) with $\mathbf{u}$ and $\mathbf{f}$ as vectors supported on $\Sigma_A$. The algebraic RAS iteration then proceeds by defining restriction operators $\widetilde{\mathbf{R}}_j : \Sigma_A \to \widetilde{\Sigma}_j$ and $\mathbf{R}_j : \Sigma_A \to \Sigma_j$, which simply truncate functions on $\Sigma_A$ to functions on $\widetilde{\Sigma}_j$ and $\Sigma_j$, respectively. Their transposes extend functions on these subdomains by zeros onto the entirety of $\Sigma_A$. Linearity of the problem may be exploited to instead solve for additive corrections to the current solution estimate [11], which naturally induces Dirichlet transmission conditions. For the RAS solver, the right-hand side of each local problem becomes the restriction of the residual at the prior iteration,

$$\mathbf{R}_j \mathbf{A} \mathbf{R}_j^T \mathbf{e}_j = \mathbf{R}_j \left( \mathbf{f} - \mathbf{A}\mathbf{u}^{(n)} \right), \tag{2.7}$$

where the local corrections, $\mathbf{e}_j$, are subsequently limited to the disjoint partition $\widetilde{\Sigma}_j$, extended by zeros onto $\Sigma_A$, and added to the existing solution. Defining the local operator $\mathbf{A}_j = \mathbf{R}_j \mathbf{A} \mathbf{R}_j^T$ and gathering all steps together, one finds the update

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \left[ \sum_{j=1}^{N_S} \widetilde{\mathbf{R}}_j^T \mathbf{A}_j^{-1} \mathbf{R}_j \right] \left( \mathbf{f} - \mathbf{A}\mathbf{u}^{(n)} \right), \tag{2.8}$$

where the summation in brackets forms the preconditioning operator $\mathbf{M}_{RAS}^{-1}$. This may be embedded within a Krylov scheme for additional acceleration, which effectively solves $\mathbf{M}_{RAS}^{-1}\mathbf{Au} = \mathbf{M}_{RAS}^{-1}\mathbf{f}$ instead of the original system. Notably, the local solves may be done in parallel, requiring limited communication from the neighboring subdomains to populate the overlapping portion of the residuals. With the introduction of more advanced transmission conditions, the local operators $\mathbf{A}_j$ cannot be obtained from simple restrictions of the global matrix [33, 11] as in (2.7), but the overall form of (2.8) as a solver or preconditioner remains valid.

## 3. Disjoint subdomain construction.

To apply DD methods to the linear system arising from the CPM, an appropriate decomposition of the active nodes, $\Sigma_A$, must be found. The ability to obtain an arbitrary number of partitions for any given surface, relying only on information that the CPM can provide, is critical to obtaining a general method.

### 3.1. Graph-based partitioning.

The problem of disjoint subdomain construction is closely related to the graph partitioning problem, and it is quite popular within the DD community to use graph partitioning tools such, as METIS [18], to partition the mesh. The discrete domain, a point cloud in this case, is reinterpreted as a graph with connectivity describing the coupling between different nodes. For typical problems posed on flat spaces, this graph construction is fairly simple. For example, finite difference schemes have nodes connected to their immediate neighbors and possibly further connected through the stencil of the discrete operator. Finite element discretizations use elements as nodes and connect them to those elements sharing a face or edge (the dual-graph of the mesh [18]).

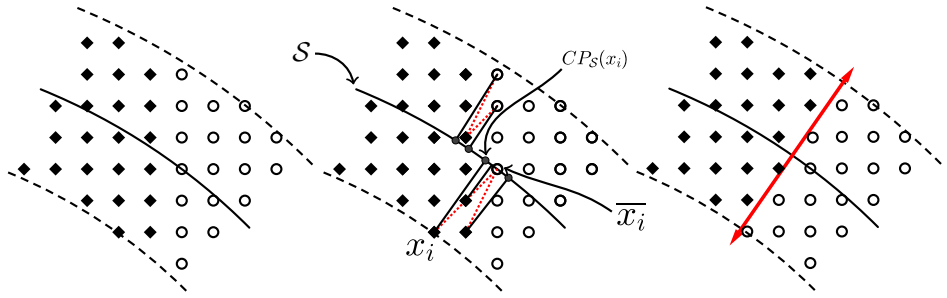


FIG. 2. *The left panel displays a typical disjoint partitioning from the nearest neighbor coupling scheme. Diamonds denote one subdomain and circles the other. The middle panel displays select interface nodes and their closest points, and highlights the relationship with solid black lines. The corresponding nearest grid points are related to the interface nodes by red dashed lines. In the plot, we label the surface $\mathcal{S}$, an interface node $x_i$, its closest point $CP_{\mathcal{S}}(x_i)$, and the corresponding nearest grid point $\overline{x}_i$. Since $\overline{x}_i$ belongs to a different subdomain than $x_i$, the interface node $x_i$ is marked for migration. The right panel illustrates the quality of the reassignment using a surface normal (red).*

The CPM offers several possible connectivity schemes due to the presence of extension, ambient discrete Laplacian, and assembled discrete Laplace–Beltrami operators. Connecting nodes through their extension stencils (Figure 1) or through the full stencil of the discrete Laplace–Beltrami operator, however, yields graphs with a large number of edges. The presence of these edges can create partitions that are contiguous with respect to the graph but not the underlying grid.

Our partitioning strategy uses the stencil of the ambient Laplacian. This creates a graph which connects each node to its immediate neighbors (and no others). We have found that this creates a much simpler graph than those created by other possible connectivity schemes. Furthermore, it ensures that the generated partitions are contiguous within the graph as well as the mesh.

**3.2. Alignment of the partition interfaces.** In the CPM, surface values are extended to the embedding space in a manner constant along the direction normal to the surface. As a consequence, it is natural to seek partitions such that the interfaces between partitions are aligned with the surface normals. While the transmission conditions defined in section 5 can be applied directly to the subdomains generated from the METIS splitting, ignorance of the underlying geometry means that the partitions will not respect the surface normals.

To improve interface alignment, we propose a simple procedure; see Figure 2 for an illustration. First, nodes along the interface between partitions are identified as those having neighbors in two or more partitions. For each interface node $x_i$, we determine the nearest grid point $\overline{x}_i$ to $CP_{\mathcal{S}}(x_i)$. (In the infrequent case where $CP_{\mathcal{S}}(x_i)$ is a grid point, we assign $\overline{x}_i = CP_{\mathcal{S}}(x_i)$.) If the interface node $x_i$ belongs to a different partition than $\overline{x}_i$, then $x_i$ is flagged for migration into the partition containing $\overline{x}_i$. After all interface nodes are queried, those flagged are moved to their target partitions. This process may need to be repeated to fully align interfaces that are far from normal. We have found that applying $p+1$ passes ($p$ being the interpolation degree) works well in practice, and we use this approach in all our numerical results. See Figure 3 for a comparison of results using the METIS splitting and the proposed alignment procedure.

**4. Overlapping subdomain construction.** The procedure of section 3 partitions the global set of active nodes, $\Sigma_A$, into disjoint subsets, $\widetilde{\Sigma}_j, j = 1\ldots, N_S$.

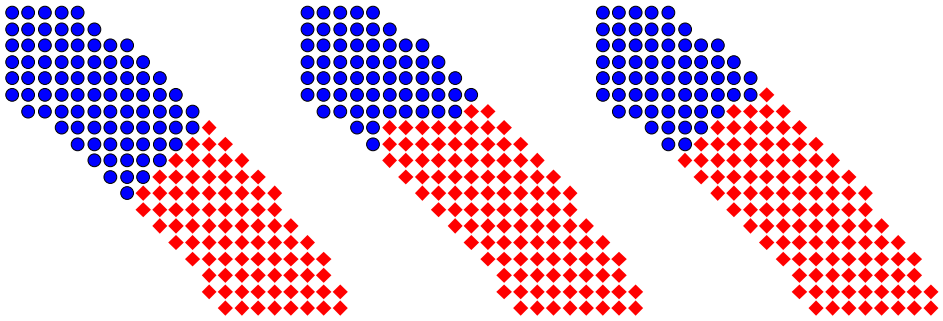

FIG. 3. *Partitions of the mesh for a circular problem. The left panel shows idealized partitions obtained by manually splitting the nodes into wedges around the circle. The middle panel shows a pair of partitions obtained from the METIS splitting using the nearest neighbor coupling. Notice that the interface does not respect the surface normal direction. Applying the process discussed in section* 3.2 *yields the result displayed in the right panel. Improved alignment with the normal is obtained.*

We have already seen in section 2.2 that overlapping partitions $\Sigma_j$ must be constructed from $\widetilde{\Sigma}_j$, just as the subdomains $\mathcal{S}_j$ were constructed from $\widetilde{\mathcal{S}}_j$ in the continuous formulation.

In section 5 we will provide full details on the construction of the discrete local problems. Several sets of nodes will be required. These will be defined as needed in sections 4.1 and 4.2. Throughout, we shall refer to Figure 4, a partition for a circular arc, to help illustrate the main ideas. Briefly, the sets to be constructed are as follows:

- $\widetilde{\Sigma}_j, j = 1, \ldots, N_S$: disjoint sets of active nodes, which together form the global set $\Sigma_A$ (circles in Figure 4);
- $\Sigma_j$: the overlapping set of active nodes in the local subproblem, $\widetilde{\Sigma}_j \subset \Sigma_j$ (circles and triangles in Figure 4);
- $\Sigma_j^G$: ghost nodes for the local subproblem that are also ghost nodes in the global problem (cross marks in Figure 4);
- $\Sigma_j^{BC}$: nodes needed to enforce the transmission condition (diamonds and plus signs in Figure 4);
- $\Lambda_j$: effective boundary locations, generated as closest points of the interface nodes in $\Sigma_j$ (approximately overlapping stars in Figure 4).

**4.1. Formation of overlaps and boundary nodes.** The set $\Sigma_j$ is grown from the disjoint partition $\widetilde{\Sigma}_j$ by layering additional, globally active nodes. The first layer is built by visiting all interface nodes in the local mesh $\Sigma_j$ and adding any missing neighboring nodes that are in $\Sigma_A$. The added nodes become the new interface nodes, and the process is repeated. A total of $N_O$ passes through the interface nodes creates the desired overlap between the local partitions. An illustration of these overlap nodes (triangles) with $N_O = 4$ is shown in Figure 4.

The set of local ghost nodes, $\Sigma_j^G$, is formed from the nodes needed to complete the finite difference stencils over the active nodes $\Sigma_j$. The set $\Sigma_j^G$ (cross marks in Figure 4) consists only of those nodes that are ghost nodes in the global problem as well. Together, $\Sigma_j$ and $\Sigma_j^G$ define the interior of the $j$th subdomain. Over these nodes, the discretization of the PDE is identical to the global problem, with the right-hand side populated by the residual from the previous iteration as in (2.7).

After forming the local active and ghost nodes, there will still be incomplete interpolation and finite difference stencils near the boundary of the subdomain. The

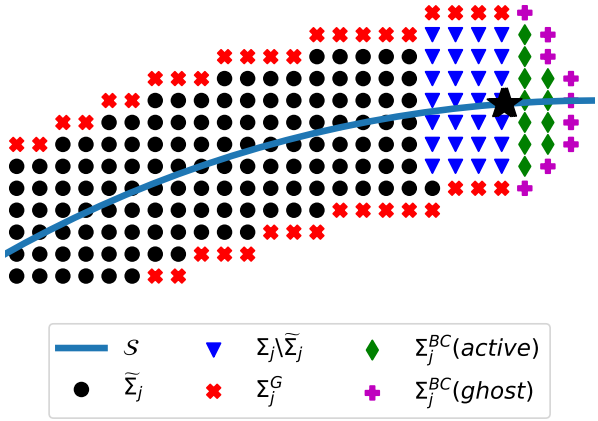


FIG. 4. *The node sets used in the construction of one subproblem. These are built up successively. We start with the disjoint nodes $\widetilde{\Sigma}_j$ coming from the graph partitioning (black circles). Four layers of overlapping nodes are added to create the grown partition $\Sigma_j$, which includes all of the black nodes as well as the blue triangles. The ghost node set $\Sigma_j^G$ is formed as a layer around these active nodes and is shown by red cross marks. The nodes from incomplete stencils are shown as green diamonds, with their ghost nodes shown as purple plus signs, forming $\Sigma_j^{BC}$ collectively. Effective boundary locations, defined in section* 4.2*, are displayed as (approximately overlapping) black stars.*

nodes needed to complete these stencils are gathered into $\Sigma_j^{BC}$ as active nodes. These are kept separate from $\Sigma_j$ since the extension operator over $\Sigma_j^{BC}$ will be modified in section 5 to enforce the transmission conditions.

For Robin transmission conditions, additional ghost nodes need to be layered around the active nodes in $\Sigma_j^{BC}$. These ghost nodes are also added to $\Sigma_j^{BC}$. The nodes in $\Sigma_j^{BC}$ are shown in Figure 4 as diamonds for the active boundary nodes and as plus signs for the ghost boundary nodes.

**4.2. Subsurface representation.** The active nodes $\Sigma_j$ can be used to partition the surface $\mathcal{S}$ into a set of overlapping subsurfaces $\mathcal{S}_j$. In our discretization of the Robin conditions in section 5.2, a closest point representation of the subsurface, $\mathcal{S}_j$, will be needed. Conormal vectors, those that are simultaneously orthogonal to the surface normal direction and $\partial\mathcal{S}_j$, must also be generated to specify the Neumann component of the Robin transmission condition. Note that these constructions are not needed for the simpler Dirichlet boundary conditions presented in section 5.1.

The closest points of the final layer of overlap nodes approximate the subsurface boundary, $\partial\mathcal{S}_j$. We collect these boundary approximations into a set $\Lambda_j \subset \mathbb{R}^d$. To obtain a closest point representation for the subsurface, $\mathcal{S}_j$, we select the closest points for boundary nodes $\Sigma_j^{BC}$ from $\Lambda_j$. This yields the desired subsurface representation,

$$CP_{\mathcal{S}_j}(x_i) = \begin{cases} CP_{\mathcal{S}}(x_i), & x_i \in \Sigma_j \cup \Sigma_j^G, \\ \arg\min_{y \in \Lambda_j} ||x_i - y||_2, & x_i \in \Sigma_j^{BC}. \end{cases} \tag{4.1}$$

The closest point computation in the final line is done efficiently by looping over points in $\Lambda_j$: For a given $y_i \in \Lambda_j$, all boundary nodes within the radius of the interpolation stencil are visited. If a node is unaffiliated, then its closest point is assigned to be $y_i$. Otherwise, if $y_i$ is closer to the node than the previous candidate, then it is updated.

This approach to building a closest point function is reminiscent of the method in [20] for constructing closest point functions from triangulated surfaces.

Boundary conormal vectors, which are simultaneously orthogonal to $\partial\mathcal{S}_j$ and the surface normal $\hat{n}_i$, are needed at each of the approximate boundary locations (see Figure 5 for an illustration). We approximate the conormal by computing the displacement vector connecting a node $x_i \in \Sigma_j^{BC}$ to its effective boundary location $d_i := x_i - CP_{\mathcal{S}_j}(x_i)$ and keep only the component that is orthogonal to the surface normal. Upon normalization, a usable approximation to the conormal vector is obtained,

$$\hat{q}_i = \frac{d_i - (d_i \cdot \hat{n}_i)\,\hat{n}_i}{||d_i - (d_i \cdot \hat{n}_i)\,\hat{n}_i||_2}. \tag{4.2}$$

Should $d_i$ lie completely in the surface normal direction, $\hat{q}_i$ is set to the zero vector. As is explained in section 5.2, replacing $\hat{q}_i$ in this special case is robust and corresponds to applying a standard closest point extension from the surface, i.e., it treats the node as if it were an active node in $\Sigma_j$.

**5. Local problems and transmission conditions.** The convergence rates of Schwarz type DD solvers and preconditioners depend strongly on the type of transmission conditions posed along the artificial interfaces. In order for the solvers to recover the global discrete solution, the discretization of the transmission conditions must be compatible with the global discretization of the problem [11]. We demonstrate that the forthcoming transmission conditions are compatible with the global discretization through a numerical example in section 6.1.

This discretization requires special attention for the CPM, as the imposition of boundary conditions occurs through modification of the extension operator prior to composition with the differential operator on the embedding space. As such, generation of transmission operators from splittings of the assembled matrix is not straightforward. For this reason, the subproblems are first considered from the continuous point of view and then discretized, rather than found directly from the algebraic perspective. The local operators will take the form

$$\mathbf{A}_j = \left(c + \frac{2d}{h^2}\right)\mathbf{I} - \left(\frac{2d}{h^2} + \Delta_j^h\right)\begin{bmatrix}\mathbf{E}_j \\ \mathbf{T}_j\end{bmatrix}, \tag{5.1}$$

where $\mathbf{E}_j$ is the portion of the global extension operator acting on $\Sigma_j$, and $\mathbf{T}_j$ is the extension operator, acting over the active nodes in $\Sigma_j^{BC}$, that is modified to enforce the transmission conditions. As shown in section 2.2, the linearity of the problem may be used to solve for additive corrections to the solution. This modification also serves to homogenize the transmission conditions. Accordingly, the final rows of the restricted residual on the right-hand side of the local problem, $\mathbf{A}_j e_j = r_j$, are set to zero to match. Throughout this paper, the local problems are solved directly via LU factorization.

**5.1. Dirichlet transmission conditions.** Homogeneous Dirichlet conditions take the form

$$u\left(CP_{\mathcal{S}_j}(x_i)\right) = 0 \quad \forall x_i \in \Sigma_j^{BC} \tag{5.2}$$

and can be enforced to first order accuracy by holding the solution at zero over all nodes in $\Sigma_j^{BC}$, i.e., by $u(x_i) = 0\ \forall x_i \in \Sigma_j^{BC}$, and completely ignoring any underlying

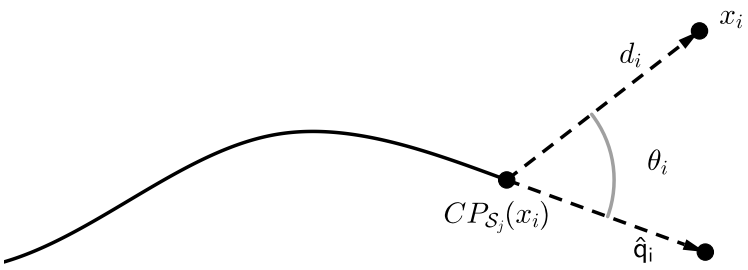


FIG. 5. *Here the relevant components for the first order accurate Robin condition are shown. The boundary node $x_i \in \Sigma_j^{BC}$ is associated with its closest point on the surface $CP_{\mathcal{S}_j}(x_i)$, from which the displacement vector $d_i$ and conormal vector $\hat{q}_i$ are constructed.*

geometry. The modified extension is thus the identity over all nodes in $\Sigma_j^{BC}$ and takes the form $\mathbf{T}_j = \begin{bmatrix} 0 & \mathbf{I} \end{bmatrix}$, with the zero matrix padding the columns corresponding to interior nodes $\Sigma_j$. Since $r_j$ is set to zero in the final entries, corresponding to the boundary nodes $\Sigma_j^{BC}$, these transmission conditions yield the same solver as algebraic RAS, with a splitting over the same overlapping and disjoint node sets $\Sigma_j$ and $\widetilde{\Sigma}_j$. This would not be the case for Dirichlet conditions enforced to second order accuracy.

**5.2. Robin transmission conditions.** Optimized restricted additive Schwarz (ORAS) methods enforce Robin transmission conditions. The incorporation of derivative information can dramatically increase the convergence rate of the scheme [11, 15, 33]. Posing these transmission conditions over the same disjoint and overlapping partitions as before leads again to homogeneous boundary conditions when solving for solution corrections.

The discretization of Robin boundary conditions has the side effect of changing the equations that boundary nodes must satisfy, and as such the local operators in (2.8) cannot be obtained by restrictions of the global operator. This modification of the equations induces a condition on the overlap of the subdomains, requiring the stencil of the Robin transmission operator to lie completely within the overlap region [33]. This, paired with the relative lack of importance of the order of accuracy of the imposed transmission conditions, motivates the construction of a Robin boundary operator for the CPM using a minimal stencil size. We propose a first order accurate discretization here that combines a forward difference for the Neumann part with a point evaluation for the Dirichlet part.

The Robin condition

$$\left.\frac{\partial u}{\partial \hat{q}_i}\right|_{CP_{\mathcal{S}_j}(x_i)} + \alpha u\left(CP_{\mathcal{S}_j}(x_i)\right) = 0 \tag{5.3}$$

will be enforced at each boundary location $CP_{\mathcal{S}_j}(x_i) \in \Lambda_j$; see Figure 5 for an illustration. Notice that the partial derivative in the boundary conormal direction, $\hat{q}_i$, can be written in terms of the node's displacement from its effective boundary position through a simple change of variables. This yields

$$\left.\frac{\partial u}{\partial d_i}\right|_{CP_{\mathcal{S}_j}(x_i)} = \cos\theta_i \left.\frac{\partial u}{\partial \hat{q}_i}\right|_{CP_{\mathcal{S}_j}(x_i)} + \sin\theta_i \left.\frac{\partial u}{\partial \hat{n}_i}\right|_{CP_{\mathcal{S}_j}(x_i)}, \tag{5.4}$$

where $\theta_i$ is the angle between $\hat{q}_i$ and $d_i$. The partial derivative in the surface normal direction vanishes since the solution is constant along the surface normals. Assuming

for a moment that $\hat{q}_i$ and $d_i$ are not perpendicular, we may combine (5.3) and (5.4) to find

$$\frac{1}{\cos\theta_i}\left.\frac{\partial u}{\partial d_i}\right|_{CP_{\mathcal{S}_j}(x_i)} + \alpha u\left(CP_{\mathcal{S}_j}(x_i)\right) = 0, \tag{5.5}$$

which is easier to discretize. Replacing the partial derivative with a forward difference and discretizing the Dirichlet term as in section 5.1 yields

$$\frac{u(x_i) - u\left(CP_{\mathcal{S}_j}(x_i)\right)}{d_i \cdot \hat{q}_i} + \alpha u(x_i) = 0, \tag{5.6}$$

where the angle $\theta_i$ has been absorbed into the dot product $d_i \cdot \hat{q}_i$. Finally, isolating $u(x_i)$ in this expression gives the relationship that the extension operator must satisfy over the boundary nodes, $\Sigma_j^{BC}$, as

$$u(x_i) = \frac{u\left(CP_{\mathcal{S}_j}(x_i)\right)}{1 + \alpha d_i \cdot \hat{q}_i}. \tag{5.7}$$

As $d_i \cdot \hat{q}_i \to 0$, (5.7) reduces to $u(x_i) = u\left(CP_{\mathcal{S}_j}(x_i)\right)$ which is identical to the extension step applied to the active nodes $\Sigma_j$. This makes sense, as this case only arises when a boundary node is adjacent to an active node. Thus the transmission condition (5.7) remains robust against the apparent possibility of division by zero in (5.6).

**5.2.1. Order of accuracy.** We test this discretization of Robin conditions on a global problem with a known exact solution. Consider $\mathcal{S}$ to be a 2 radian arc of the unit circle. The boundary value problem

$$\begin{cases} (1 - \Delta_{\mathcal{S}})\, u = 1 - \theta^2 & \text{on } \mathcal{S}, \\ u(0) = 0, \\ \frac{\partial u}{\partial \theta}\big|_{\theta=2} + u(2) = 0 \end{cases} \tag{5.8}$$

has the solution $u(\theta) = \cosh(\theta) + (9e^{-2} - 1)\sinh(\theta) - \theta^2 - 1$. The Laplace–Beltrami operator is discretized via the CPM described in section 2.1, with bicubic interpolation for the extension operator and a second order centered difference Laplacian on the embedding space. To isolate the effect of the Robin condition, the Dirichlet condition at $\theta = 0$ is enforced to second order accuracy by the mirror point discretization described in [21]. Finally, the Robin condition at $\theta = 2$ is enforced by the discretization given in (5.7). Notice that placing the Robin condition at $\theta = 2$ yields a conormal direction that is not aligned with the grid on the embedding space. The error in the solution is measured in the infinity norm over a sequence of grid resolutions, as seen in Table 1. We observe that the global error is first order accurate.

TABLE 1

*The global $\infty$-norm error for a CPM discretization of an elliptic problem on a circular arc* (5.8). *The problem was solved on five grids of increasing refinement using the proposed Robin boundary condition discretization* (5.7). *We observe first order accuracy.*

| $h$ | 1/64 | 1/128 | 1/256 | 1/512 | 1/1024 |
|---|---|---|---|---|---|
| $\|\|u - u_{ex}\|\|_\infty$ | $5.15 \times 10^{-2}$ | $2.56 \times 10^{-2}$ | $1.27 \times 10^{-2}$ | $6.36 \times 10^{-3}$ | $3.18 \times 10^{-3}$ |

**5.2.2. Modified scheme for cross points.** Points where more than two subdomains meet are anticipated when a surface embedded in $\mathbb{R}^d$, $d \geq 3$, is partitioned. Nonoverlapping optimized Schwarz methods are destabilized by these points when otherwise optimal values of the Robin parameter are used [16, 19]. This behavior typically does not appear in ORAS methods, used as solvers or as preconditioners, since each unknown is only updated by a single subproblem. However, we do see this behavior for the CPM and attribute it to the presence of the extension operator enlarging the stencil size.

Stability and convergence can be recovered by increasing $\alpha$, thereby weighting the Dirichlet component of the Robin transmission condition more heavily. However, raising the value of $\alpha$ globally is nonoptimal; fewer iterations are required if we use the larger value only in the vicinity of the cross points. This approach weights the Dirichlet term more heavily in regions where it is needed for stability while selecting weights elsewhere for rapid convergence. We let $\alpha^\times$ denote the larger weight used near the cross points and leave $\alpha$ to denote the Robin weight elsewhere. For Poisson and positive Helmholtz problems on the plane, it is known that the optimal values of these weights scale as $\alpha \sim \mathcal{O}\left(h^{-1/2}\right)$ away from the cross points and as $\alpha^\times \sim \mathcal{O}\left(h^{-1}\right)$ near the cross points [16].

Identifying regions for applying each weight proceeds in a fashion similar to the construction of the boundary locations in section 4.2. In each subdomain $\widetilde{\Sigma}_j$, cross point nodes are identified as those having neighbors belonging to more than one other subdomain. A sphere of radius $2N_O h$ is centered on each of these cross point nodes, and all boundary nodes in $\Sigma_j^{BC}$ lying within this sphere are marked to use a Robin weight of $\alpha^\times$ in place of $\alpha$.

**6. Results.** Our development of (O)RAS methods for the iterative solution of the stabilized CPM has seen the introduction of the standard DD parameters, the number of subdomains $N_S$, the overlap width $N_O$, and Robin weight $\alpha$. A subdomain boundary alignment method and two transmission conditions specific to this problem have also been introduced. In the following subsections we sweep over these parameters and evaluate the effect of these problem-dependent choices. The software written for solving these problems is available through a public repository [25]. These results were obtained specifically with commit *fc3612f9*. Later commits will be focused on user friendliness and general applicability of the code base.

The methods are tested over three surfaces of increasing complexity, as seen in Figure 6. The unit sphere is simple, yet extremely important, and makes an ideal first test. The torus, with major radius $R = 2/3$ and minor radius $r = 1/3$, introduces a hole and regions of negative curvature. Finally, the Stanford Bunny [35] is a triangulated surface with regions of high curvature and holes. We use the original Stanford Bunny triangulation with five holes. Our only modification is to scale the bunny to be two units tall. In all cases, the extension operator is constructed with triquadratic tensor product interpolation, as introduced in section 2.1. In all tests, convergence was declared when the 2-norm of the residual vector was reduced by a factor of $10^6$.

**6.1. Consistency of the transmission conditions.** Before evaluating the performance of the proposed solvers and the effects of the various DD parameters, the transmission conditions defined in section 5 are validated for compatibility with the global discretization. This is done numerically by comparing the DD solution to the direct solution of the global system. We consider the unit sphere with $h = 1/25$ and $h = 1/50$, divided into $N_S = 16$ subdomains with an overlap of $N_O = 4$. The Robin transmission conditions use the weights $\alpha = 4$ and $\alpha^\times = 40$. The reference solution

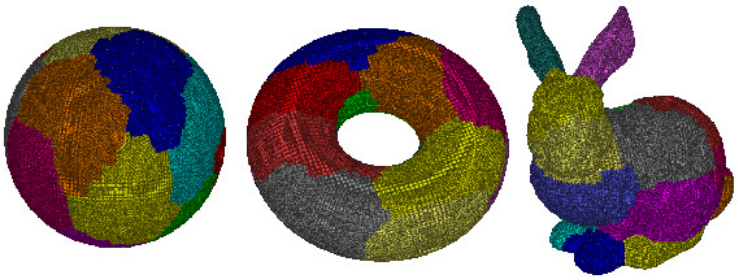

FIG. 6. *The three surfaces used to evaluate our solvers with colors showing individual subdomains obtained from METIS. All surfaces use* $h = 1/30$ *and have* 16 *subdomains. The active nodes are projected onto the surfaces for the sake of visualization.*

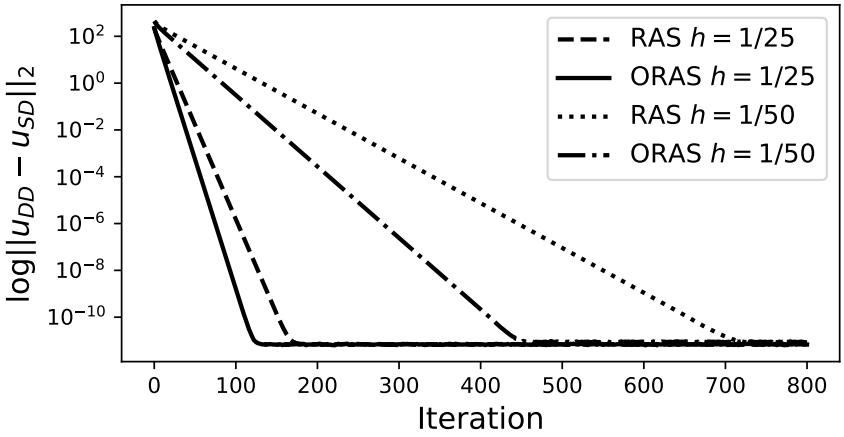


FIG. 7. *The norm of the difference between the DD solution,* $U_{DD}$*, and the MUMPS reference solution,* $U_{SD}$*, is shown here as a function of the DD iteration number for RAS and ORAS solvers on the unit sphere. Grid resolutions of* $h = 1/25$ *and* $h = 1/50$ *were tested. The error compared to the exact continuous solution is* $4.486 \cdot 10^{-3}$ *for* $h = 1/25$*, and* $1.264 \cdot 10^{-3}$ *for* $h = 1/50$*, measured in the* 2*-norm.*

of the global system is obtained with the sparse direct solver MUMPS [1, 2] for each resolution. The right-hand side is chosen such that the exact continuous solution is $\sin^2(\phi)e^{\cos\theta}$, to verify that the error between the DD solution and global discrete solution is below the truncation error present in the discretization.

Figure 7 shows the 2-norm of the difference between the reconstructed DD solution and the MUMPS reference solution for each combination of resolution and (O)RAS solver. Notably, the error in the DD solution plateaus at a significantly smaller value than the truncation error present in the discretization itself. Indeed, the error plateau depends only on the condition number of the global matrix and thus on the reliability of the reference MUMPS solution. No dependence on the choice of solver, right-hand side, or DD parameters has been observed.

**6.2. Effect of the Robin weight.** Next, we investigate how the optimal Robin parameter, $\alpha$, depends on the grid spacing for the three surfaces by splitting each into two subdomains with a fixed overlap of $N_O = 4$ and varying $\alpha$ and $h$. There are no cross points when the surface is split into two subdomains.

The unit sphere produces meshes with $N_A = 163542$, $N_A = 654630$, and $N_A =$

TABLE 2

*Iterations to convergence for the unit sphere as the Robin weight $\alpha$ and grid spacing $h$ were varied. The sphere was divided into two subdomains with an overlap of $N_O = 4$. The final column gives the corresponding RAS iteration count.*

| $h$ \ $\alpha$ | 1/4 | 1/2 | 3/4 | 1 | 3/2 | 2 | 4 | 8 | 16 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Solver | | | | | |
| 1/50 | 19 | 14 | 11 | 11 | 15 | 20 | 33 | 49 | 67 | 83 |
| 1/100 | 25 | 17 | 14 | 13 | 16 | 21 | 37 | 61 | 93 | 159 |
| 1/200 | 32 | 22 | 18 | 18 | 18 | 23 | 43 | 75 | 122 | 303 |
| | | | | | Preconditioner | | | | | |
| 1/50 | 11 | 9 | 9 | 8 | 8 | 9 | 11 | 13 | 15 | 17 |
| 1/100 | 12 | 11 | 10 | 9 | 9 | 9 | 12 | 15 | 18 | 20 |
| 1/200 | 14 | 12 | 12 | 11 | 10 | 11 | 13 | 17 | 21 | 31 |

2618046 for grid spacings of $h = 1/50$, $h = 1/100$, and $h = 1/200$, respectively. The torus produces meshes with $N_A = 114692$, $N_A = 454300$, and $N_A = 1818052$ for the same grid spacings. Finally, the Stanford Bunny produces meshes with $N_A = 98714$, $N_A = 396232$, and $N_A = 1585218$, respectively.

Table 2 shows the number of iterations to convergence for the sphere with grid resolutions of $h = 1/50, 1/100, 1/200$ as $\alpha$ is varied. Tables 3 and 4 show the same information for the torus and the Stanford Bunny, respectively. The ORAS solvers on the sphere and torus show flat iteration counts around the optimal $\alpha$ value. Though $\alpha = 1$ appears optimal throughout all resolutions for these two surfaces, we note that the iteration counts for $\alpha = 1/2$ are growing, and the region of flat iteration counts is shifting to larger values of $\alpha$ as the resolution is refined. The ORAS solver on the Stanford Bunny shows a clear dependence of the optimal value for $\alpha$ on the resolution, varying from $\alpha = 1$ when $h = 1/50$ to $\alpha = 2$ when $h = 1/200$. For all three surfaces, the ORAS preconditioner shows the same trend, with the optimal $\alpha$ value increasing with grid refinement, but the effect is weaker.

In each of these cases, the iteration count of the ORAS method is seen to approach the iteration count of the corresponding RAS method as $\alpha$ is increased beyond the optimal value. This agrees with our expectation, since the weight of the Dirichlet term increases as $\alpha$ increases.

**6.3. Necessity of the cross point modification.** In this subsection, we will examine two resolutions of the grid, $h = 1/100$ and $h = 1/200$, with the same number of global unknowns as the corresponding problems in the previous subsection. In the preceding subsection, only two subdomains were used, and thus there were no cross points present in the decomposition. Here we decompose the problems into $N_S = 32$ subdomains, which introduces cross points into the splitting. The Robin weights $\alpha$ and $\alpha^\times$ are varied and the required iterations to convergence for the ORAS solvers and preconditioners gathered. With cross points present, the values of $\alpha$ permitting convergence are shifted upwards, and in the following, $\alpha = 2$ is the minimum tested.

We consider the sphere first. In Figures 8 and 9 we see that the required iterations of the solver and preconditioner depend more strongly on the value of $\alpha$ than $\alpha^\times$, provided $\alpha^\times$ is large enough to yield a convergent method. In particular, when $h = 1/100$, $\alpha = 2$, and $\alpha^\times$ is set to 12, 20, and 40, the solver (preconditioner) requires 141(30), 170(30), and 214(32) iterations, respectively. Similarly, when $h = 1/200$, $\alpha = 2$, and $\alpha^\times$ is set to 20, 40, and 80, the solvers (preconditioners) require 131(32), 170(32), and 211(35) iterations to converge.

TABLE 3

*Iterations to convergence for the torus as the Robin weight $\alpha$ and grid spacing $h$ are varied. The torus was divided into two subdomains with an overlap of $N_O = 4$. The final column gives the corresponding RAS iteration count.*

| $h$ \ $\alpha$ | 1/4 | 1/2 | 3/4 | 1 | 3/2 | 2 | 4 | 8 | 16 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Solver | | | | | |
| 1/50 | 20 | 13 | 11 | 10 | 14 | 18 | 30 | 44 | 59 | 74 |
| 1/100 | 24 | 17 | 13 | 12 | 16 | 20 | 36 | 58 | 86 | 142 |
| 1/200 | 29 | 20 | 17 | 15 | 15 | 20 | 37 | 66 | 107 | 272 |
| | | | | | Preconditioner | | | | | |
| 1/50 | 10 | 9 | 8 | 7 | 7 | 8 | 9 | 10 | 12 | 13 |
| 1/100 | 11 | 10 | 9 | 8 | 8 | 8 | 10 | 12 | 14 | 17 |
| 1/200 | 12 | 11 | 10 | 9 | 9 | 9 | 10 | 12 | 15 | 22 |

TABLE 4

*Iterations to convergence for the Stanford Bunny as the Robin weight $\alpha$ and grid spacing $h$ were varied. The bunny was divided into two subdomains with an overlap of $N_O = 4$. The final column gives the corresponding RAS iteration count. The optimal value of $\alpha$ grows as the grid resolution is refined.*

| $h$ \ $\alpha$ | 1/4 | 1/2 | 3/4 | 1 | 3/2 | 2 | 4 | 8 | 16 | $\infty$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Solver | | | | | |
| 1/50 | 18 | 14 | 11 | 11 | 15 | 19 | 30 | 42 | 55 | 65 |
| 1/100 | 69 | 30 | 21 | 17 | 16 | 20 | 35 | 57 | 83 | 133 |
| 1/200 | 46 | 33 | 27 | 24 | 20 | 18 | 32 | 55 | 89 | 215 |
| | | | | | Preconditioner | | | | | |
| 1/50 | 11 | 9 | 8 | 8 | 9 | 9 | 10 | 12 | 13 | 14 |
| 1/100 | 15 | 12 | 11 | 10 | 10 | 10 | 10 | 14 | 16 | 20 |
| 1/200 | 16 | 15 | 14 | 13 | 12 | 11 | 12 | 15 | 18 | 28 |

Figures 10 and 11 show the same sweeps over $\alpha$ and $\alpha^\times$ but for the torus. For $h = 1/100$, there is an optimum parameter combination for the ORAS solver with $\alpha = 10$ and $\alpha^\times = 10$, requiring only 146 iterations for the solver to converge. Although this implies that the cross point modification is unnecessary, it is the only time this was observed, and it seems to coincide with the splitting produced by METIS for this surface at this resolution and subdomain count. At $h = 1/200$ the optimal solver uses $\alpha = 4$ and $\alpha^\times = 20$. This decrease in the optimal value for $\alpha$ opposes the expected behavior, although the large number of subdomains and complicated geometry make comparisons to the flat case difficult. At this resolution, $\alpha^\times = 20$ is optimal for almost all values of $\alpha$. Consistent with the results for the sphere, the dependence of the preconditioners on $\alpha^\times$ is weak as long as the value is large enough to yield a convergent solver.

Finally, Figures 12 and 13 show the effects of $\alpha$ and $\alpha^\times$ on the iterations to convergence for the ORAS solvers and preconditioners on the Stanford Bunny. Overall the results are quite similar to those of the sphere and torus problems.

Notably, the use of larger values of $\alpha^\times$ allows smaller values of $\alpha$ to be used successfully. For instance, Figure 13 shows that choosing $\alpha^\times = 40$ allows $\alpha = 2$ to be used. This combination of $\alpha^\times$ and $\alpha$ provides the smallest number of iterations to convergence. Conversely, taking $\alpha^\times = 12$ or smaller leads to slow or failed convergence (for example, the ORAS solvers did not converge at all for these values). We further observe that the use of the cross point modification with the smaller allowable values

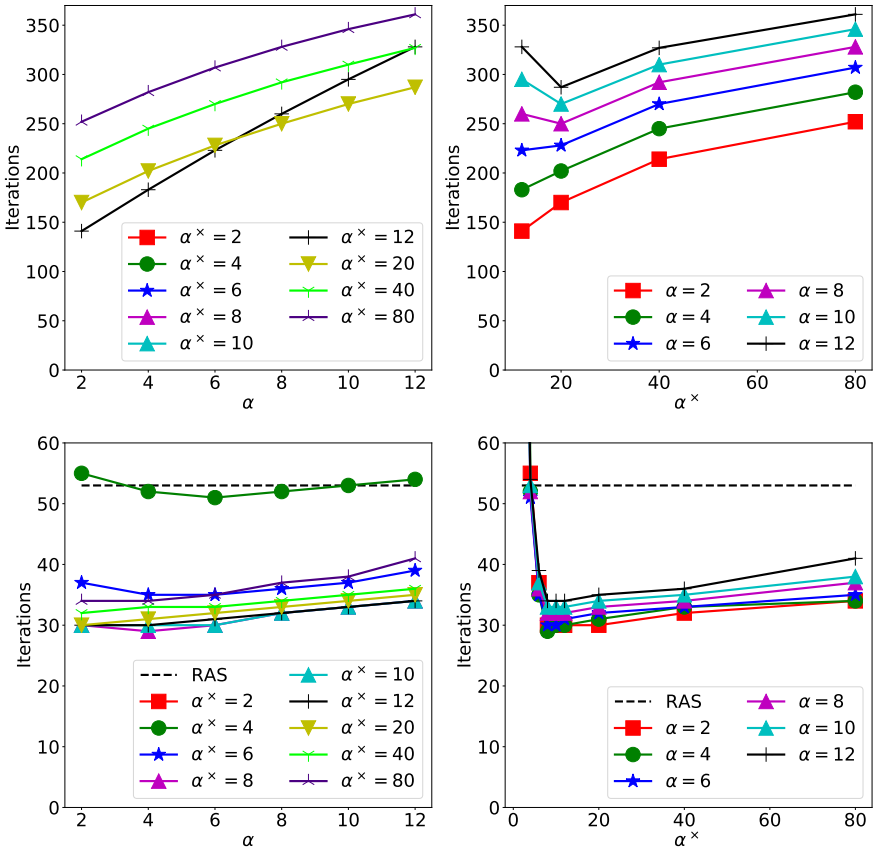


FIG. 8. *Dependence of iterations to convergence on $\alpha$ and $\alpha^{\times}$ for the sphere with $h = 1/100$. The upper panels show the iterations to convergence when ORAS is run as a solver, while the lower panels use ORAS as a preconditioner. The left panels sweep over different values of $\alpha$, holding $\alpha^{\times}$ fixed. The right panels show the same results as a function of $\alpha^{\times}$ for fixed values of $\alpha$. As shown in the upper left panel, $\alpha^{\times} < 12$ gives solvers that never converge regardless of the choice of $\alpha$. For comparison, RAS requires* 500 *iterations as a solver and* 53 *iterations as a preconditioner.*

of $\alpha$ leads to ORAS solvers and preconditioners that enjoy greatly reduced iteration counts relative to their RAS counterparts. The preconditioners again depend very weakly on the value of $\alpha^{\times}$ for values that are sufficiently large.

**6.4. Effect of the overlap width.** The effect of the overlap width was investigated with $N_S = 32$ and $h = 1/200$ fixed for all surfaces. The ORAS methods use Robin parameters of $\alpha = 2$ and $\alpha^{\times} = 40$ for the sphere, $\alpha = 4$ and $\alpha^{\times} = 20$ for the torus, and $\alpha = 2$ and $\alpha^{\times} = 40$ for the Stanford Bunny.

As seen in Table 5, the RAS solvers and preconditioners consistently require fewer iterations as the overlap is widened. The ORAS solvers, contrary to expectation, show increasing iteration counts with widened overlaps, while the preconditioners show little variation. As described in section 5.2.2, as $N_O$ is increased the number of affected nodes grows, and $\alpha^{\times}$ is used more often. This shifts the method away from optimal behavior and yields larger iteration counts. To affirm this, the same surfaces and splittings were run without the cross point modification. The results of this, shown in

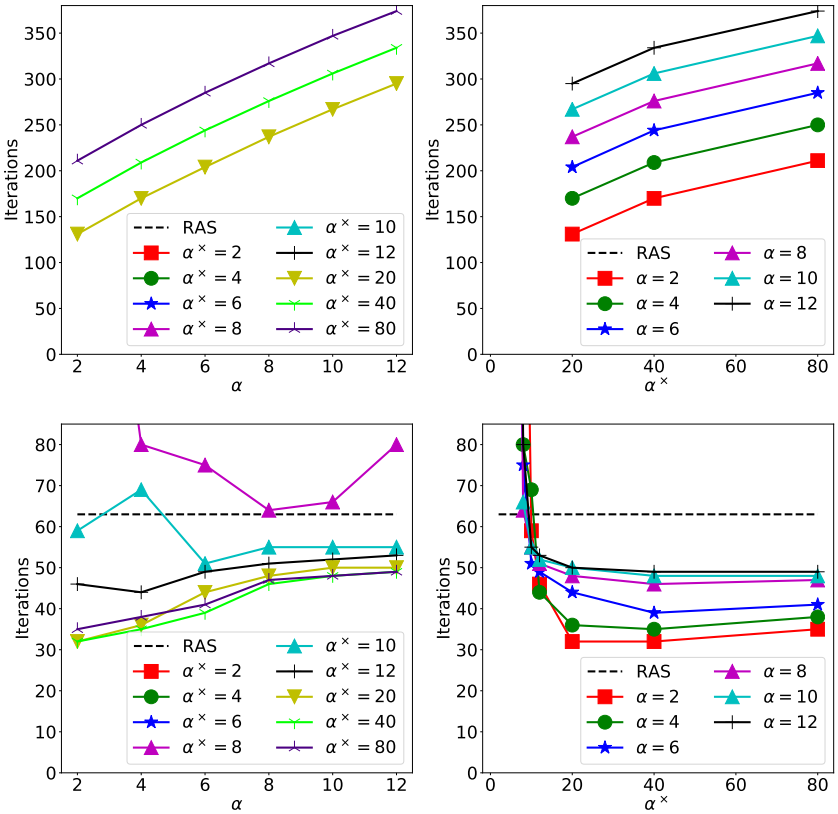


FIG. 9. *Dependence of iterations to convergence on $\alpha$ and $\alpha^\times$ for the sphere with $h = 1/200$. The upper panels show the iterations to convergence when ORAS is run as a solver, while the lower panels use ORAS as a preconditioner. The left panels sweep over different values of $\alpha$, holding $\alpha^\times$ fixed. The right panels show the same results as a function of $\alpha^\times$ for fixed values of $\alpha$. As shown in the upper left panel, $\alpha^\times < 20$ gives solvers that never converge regardless of the choice of $\alpha$. For comparison, RAS requires* 916 *iterations as a solver and* 63 *iterations as a preconditioner.*

Table 6, show decreasing iteration counts as the overlap widens. However, the larger value of $\alpha$, now needed everywhere, slows all of the solvers and preconditioners. For ORAS methods, it is wise to use a small overlap with appropriately chosen parameters $\alpha$ and $\alpha^\times$.

**6.5. Effect of the subdomain count.** As the number of subdomains is increased, the subproblems become cheaper to assemble and factor, and the method exhibits greater parallelism. On the other hand, the local problems see less of the global problem and naturally require more iterations for convergence.

We now examine this in greater detail on our three shapes using a fixed overlap of $N_O = 4$, and subdomain counts $N_S = 32, 64, 96$, with grid spacings of $h = 1/100$ and $1/200$. With these large subdomain counts, there are many cross points scattered throughout the domain. The cross point modified scheme, introduced in section 5.2.2 and examined in section 6.3, is thus used for all ORAS solvers and preconditioners. We set $\alpha = 4$ and $\alpha^\times = 40$ throughout.

Examining the results in Table 7, we immediately see that the ORAS solvers and

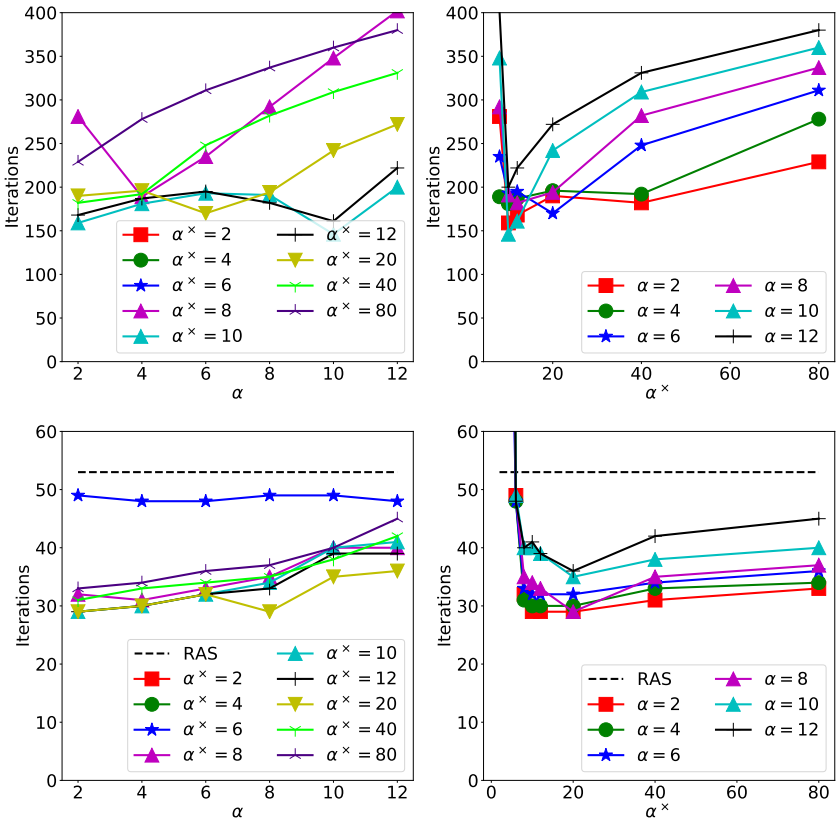


FIG. 10. *Dependence of iterations to convergence on $\alpha$ and $\alpha^\times$ for the torus with $h = 1/100$. The upper panels show the iterations to convergence when ORAS is run as a solver, while the lower panels use ORAS as a preconditioner. The left panels sweep over different values of $\alpha$, holding $\alpha^\times$ fixed. The right panels show the same results as a function of $\alpha^\times$ for fixed values of $\alpha$. As shown in the upper left panel, $\alpha^\times < 8$ gives solvers that never converge regardless of the choice of $\alpha$. For comparison, RAS requires 597 iterations as a solver and 53 iterations as a preconditioner.*

preconditioners outperform their RAS counterparts over the same splittings in all cases, often by a wide margin. However, we see diminishing gains in performance for the ORAS preconditioners over their RAS counterparts as the number of subdomains is increased. For example, for the sphere with $h = 1/100$ and $N_S = 32$ subdomains, the iteration count is reduced from 53 to 33 by switching from RAS to ORAS. The same comparison with $N_S = 64$ subdomains gives a smaller relative reduction from 57 iterations to 46 iterations. Using a large number of subdomains increases the number of cross points, and the modification then affects a larger fraction of the boundary nodes. Raising the resolution of these problems to $h = 1/200$ mitigates the issues with cross point density, and the performance gap between the RAS and ORAS solvers widens for all surfaces and subdomain counts.

All three surfaces show that the larger subdomain counts require more iterations. The rise in iterations to convergence as the number of subdomains is increased is to be expected for any single-level method, such as those presented here. The use of a suitable coarse correction or multigrid method (cf. [7]) would likely mitigate this effect and would pose an interesting follow up to this work.

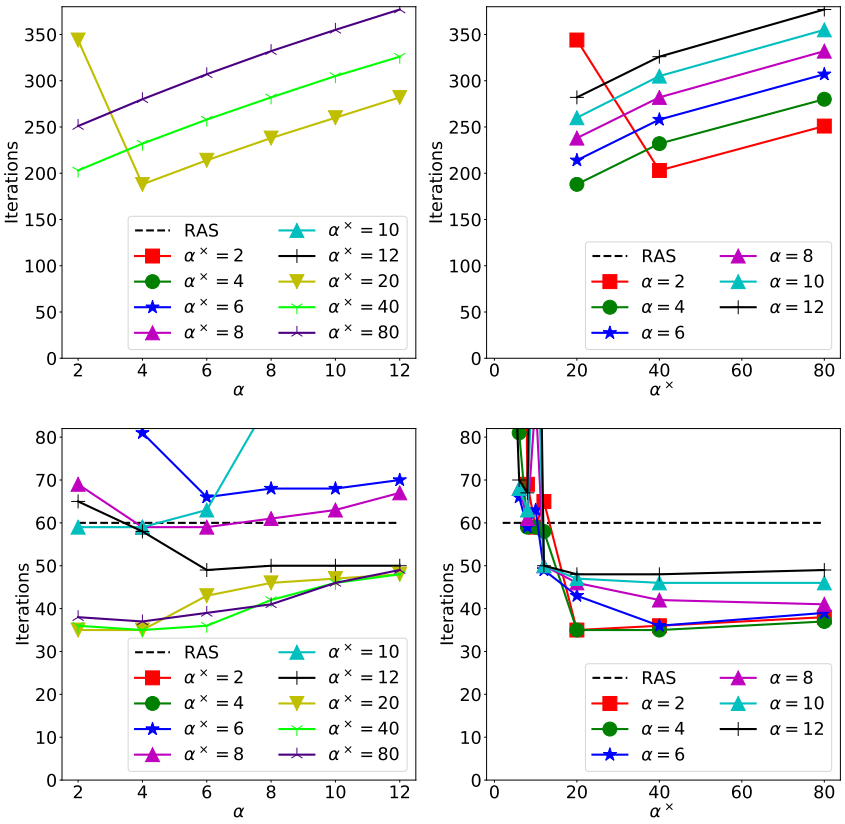


FIG. 11. *Dependence of iterations to convergence on $\alpha$ and $\alpha^\times$ for the torus with $h = 1/200$. The upper panels show the iterations to convergence when ORAS is run as a solver, while the lower panels use ORAS as a preconditioner. The left panels sweep over different values of $\alpha$, holding $\alpha^\times$ fixed. The right panels show the same results as a function of $\alpha^\times$ for fixed values of $\alpha$. As shown in the upper left panel, $\alpha^\times < 20$ gives solvers that never converge regardless of the choice of $\alpha$. For comparison, RAS requires* 1148 *iterations as a solver and* 60 *iterations as a preconditioner.*

**6.6. Parallel performance.** We conclude our evaluation of the (O)RAS methods with a study of the time to solution as the number of processors is increased. All timings recorded here were run using whole nodes of the Graham compute cluster managed by Compute Canada. This system has two 16-core Intel E5-2683 v4 Broadwell 2.1 GHz processors and 125G of memory on each compute node. Throughout this subsection, computations are carried out on the unit sphere with an overlap width of $N_O = 4$; ORAS methods again use Robin parameters of $\alpha = 4$ and $\alpha^\times = 40$.

The timings presented are split into several phases as follows:

- global meshing, which includes building and partitioning the graph as well as scattering the permuted nodes across the processes;
- construction of the global matrix, which includes computation of the extension weights;
- construction and factorization of the local operators;
- solution of the system by the RAS and ORAS solvers; and
- solution of the system by GMRES preconditioned by (O)RAS.

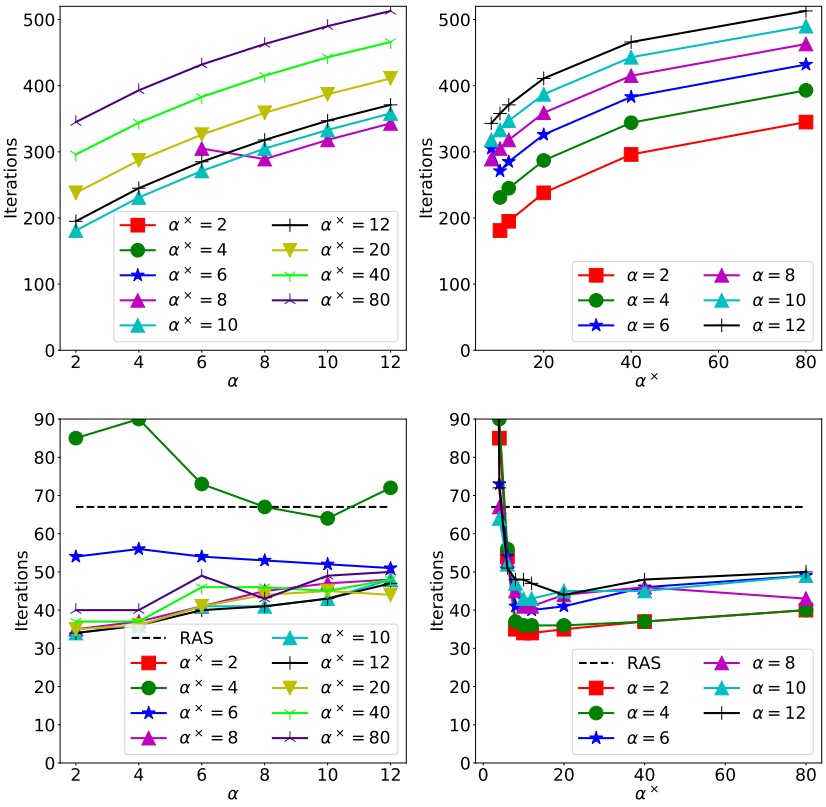


FIG. 12. *Dependence of iterations to convergence on $\alpha$ and $\alpha^{\times}$ for the Stanford Bunny with $h = 1/100$. The upper panels show the iterations to convergence when ORAS is run as a solver, while the lower panels use ORAS as a preconditioner. The left panels sweep over different values of $\alpha$, holding $\alpha^{\times}$ fixed. The right panels show the same results as a function of $\alpha^{\times}$ for fixed values of $\alpha$. As shown in the upper left panel, $\alpha^{\times} < 8$ gives solvers that never converge regardless of the choice of $\alpha$. For comparison, RAS requires* 717 *iterations as a solver and* 67 *iterations as a preconditioner.*

The first three phases are performed before the solution is computed and are thus independent of the number of iterations required by the solver or preconditioner. The final two phases produce the same answer, so in practice only one would be needed. Each case was run five times, with the average time of each phase reported below.

The global meshing procedures have not been fully parallelized in the current implementation of this software. The disjoint partitioning relies on METIS rather than ParMETIS, the parallel version of the software, which introduces an early serial bottleneck. As a consequence, our global meshing times do not show any parallel scalability. Note, however, that the serial components in the global meshing do not affect the subsequent steps which have been fully parallelized.

Strong scalability is quantified by measuring the change in run-time as the number of available processes is increased. Ideally this run-time would decrease in proportion to the number of processes used. In our experiments, the subdomain count is set equal to the number of processes, and the grid resolution is $h = 1/250$, yielding a problem with $N_A = 4098174$ active nodes. Aside from the global meshing phase, the method is easily parallelized, and, as seen in Table 8, all set-up phases of the method

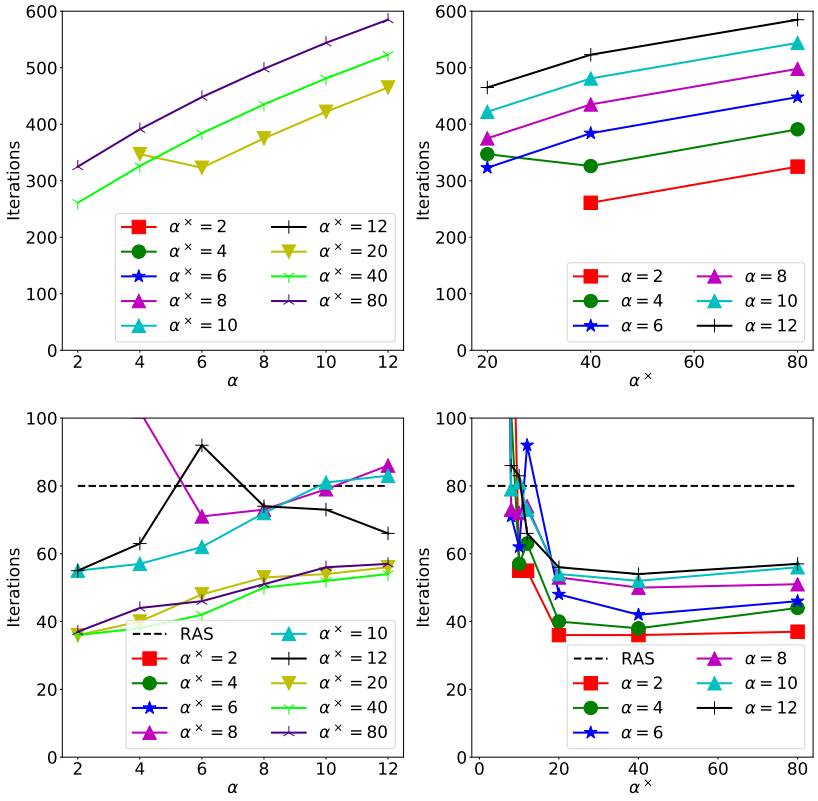


FIG. 13. *Dependence of iterations to convergence on $\alpha$ and $\alpha^\times$ for the Stanford Bunny with $h = 1/200$. The upper panels show the iterations to convergence when ORAS is run as a solver, while the lower panels use ORAS as a preconditioner. The left panels sweep over different values of $\alpha$, holding $\alpha^\times$ fixed. The right panels show the same results as a function of $\alpha^\times$ for fixed values of $\alpha$. For comparison, RAS requires* 1301 *iterations as a solver and* 80 *iterations as a preconditioner.*

except for the meshing procedure show good strong scaling, with the time spent in each phase roughly halving as the number of processors is doubled. The solver phase shows more complicated behavior. As the subdomain count is increased, the solver performance becomes weaker, as noted in section 6.5, and requires more iterations to converge. Simultaneously, the jump from $N_{proc} = 32$ to $N_{proc} = 64$ requires the use of two compute nodes and introduces communication latency. The preconditioned GMRES times do decrease with greater parallelism. The results for the solver, in particular, highlight the need for a compatible coarse space and two-level method. The global meshing procedure requires a nearly constant time due to its serial nature.

The MUMPS [1, 2] direct solver was tested on the same problem used for the strong scalability test. When confined to a single compute node, where $N_{proc} = 32$, the direct solver requires more memory than available, giving a run-time error. Increasing to two compute nodes, with $N_{proc} = 64$, allows the direct solver to successfully run but required 384 seconds to find the factorization and solution. From Table 8 the time required for ORAS preconditioned GMRES on the same problem is only 35 seconds for the assembly and factorization of the local operators, and 12 seconds for GMRES to converge, for a total of 47 seconds. The other phases of the run must be executed

TABLE 5

*The iterations to convergence as the overlap width $N_O$ was varied with cross point modification. The RAS solvers and preconditioners consistently require fewer iterations to converge as $N_O$ grows, as expected. The ORAS solvers defy expectation and show growing iterations to convergence as $N_O$ grows. This is due to cross point modification affecting more nodes at larger overlap widths. This is confirmed by Table 6, where the same sweep is done without cross point modification.*

| $N_O$ | 4 | | 8 | | 12 | |
|---|---|---|---|---|---|---|
| Method | RAS | ORAS | RAS | ORAS | RAS | ORAS |
| Sphere, $h = 1/200$, $N_A = 2618046$ | | | | | | |
| Solver | 916 | 170 | 598 | 182 | 449 | 207 |
| Preconditioner | 63 | 36 | 55 | 32 | 51 | 31 |
| Torus, $h = 1/200$, $N_A = 1818052$ | | | | | | |
| Solver | 738 | 188 | 536 | 217 | 412 | 240 |
| Preconditioner | 60 | 35 | 49 | 33 | 49 | 32 |
| Stanford Bunny, $h = 1/200$, $N_A = 1585218$ | | | | | | |
| Solver | 1301 | 347 | 820 | 307 | 609 | 317 |
| Preconditioner | 80 | 40 | 64 | 36 | 58 | 35 |

TABLE 6

*The iterations to convergence as the overlap width $N_O$ was varied without cross point modification. As $N_O$ is increased, the ORAS solvers and preconditioners show consistently decreasing iterations to convergence. Although this matches the expectation for iteration counts against $N_O$, Table 5 shows that usage of the cross point modification yields faster convergence regardless of overlap width.*

| | Sphere | | | Torus | | | Stanford Bunny | | |
|---|---|---|---|---|---|---|---|---|---|
| $N_O$ | 4 | 8 | 12 | 4 | 8 | 12 | 4 | 8 | 12 |
| Solver | 393 | 319 | 299 | 358 | 312 | 286 | 604 | 503 | 426 |
| Preconditioner | 53 | 49 | 38 | 51 | 47 | 43 | 64 | 53 | 44 |

regardless of the choice of solution method. Similarly, with $N_{proc} = 128$ the direct solver requires 325 seconds for the relevant operations, while ORAS preconditioned GMRES requires just 23 seconds. We conclude that iterative methods are preferable for problems of this scale.

Weak scalability seeks to keep the load per processor nearly constant as $N_{proc}$ is increased, and ideally the time required would stay constant. The weak scalability of the method is evaluated by using one subdomain per process, $N_S = N_{proc}$, and using problem resolutions of $h = 1/125, 1/175, 1/250$ for $N_{proc} = 32, 64, 128$, respectively. These resolutions have $N_A = 1025622, 2005758, 4098174$ active nodes yielding approximately 32000 nodes per subdomain. Table 9 records the time required by each phase of the method for this progression of problems. The construction of the global matrix and the local problems requires nearly constant time, indicating excellent weak scaling. However, as $h$ is refined and $N_S$ is increased, the global system shows worse conditioning, and the (O)RAS solver slows. These effects drive up the number of iterations required for convergence and disrupt the scalability of the solution phases. This is a characteristic of single-level solvers.

In summary, the methods exhibit good parallel scalability over the range of process counts tested. The shortcomings present in the weak scalability of the solution phases of the methods are to be expected for a single-level method such as this. The meshing procedures are obvious candidates for improvement, though this cost is fixed and would be amortized in an application requiring many solutions of the linear system.

TABLE 7

*The iterations to convergence as the subdomain count varies for our three surfaces. Resolutions of $h = 1/100$ and $h = 1/200$ are presented. The ORAS solvers and preconditioners use $\alpha = 4$ and $\alpha^{\times} = 40$ throughout. The iterations to convergence consistently grow as $N_S$ is increased, as expected.*

| $N_S$ | 32 | | 64 | | 96 | |
|---|---|---|---|---|---|---|
| Method | RAS | ORAS | RAS | ORAS | RAS | ORAS |
| Sphere, $h = 1/100$, $N_A = 654630$ | | | | | | |
| Solver | 500 | 245 | 704 | 278 | 818 | 405 |
| Preconditioner | 53 | 33 | 57 | 46 | 59 | 49 |
| Sphere, $h = 1/200$, $N_A = 2618046$ | | | | | | |
| Solver | 916 | 209 | 1228 | 352 | 1318 | 434 |
| Preconditioner | 63 | 35 | 78 | 51 | 99 | 53 |
| Torus, $h = 1/100$, $N_A = 454300$ | | | | | | |
| Solver | 597 | 192 | 780 | 549 | 1099 | 701 |
| Preconditioner | 53 | 33 | 57 | 49 | 63 | 55 |
| Torus, $h = 1/200$, $N_A = 1818052$ | | | | | | |
| Solver | 1148 | 234 | 1511 | 343 | 2147 | 457 |
| Preconditioner | 67 | 38 | 83 | 48 | 104 | 57 |
| Stanford Bunny, $h = 1/100$, $N_A = 396292$ | | | | | | |
| Solver | 717 | 344 | 1015 | 550 | 1306 | 771 |
| Preconditioner | 67 | 37 | 76 | 59 | 82 | 70 |
| Stanford Bunny, $h = 1/200$, $N_A = 1585218$ | | | | | | |
| Solver | 1301 | 326 | 1906 | 528 | 2346 | 687 |
| Preconditioner | 80 | 38 | 140 | 65 | 143 | 73 |

TABLE 8

*The time in seconds for each phase of the method for the sphere with grid spacing $h = 1/250$ and $N_S = N_{proc}$ subdomains. The first three phases are independent of the number of iterations used in the solution phase. Note that only one of the last two phases needs to be used in practice.*

| | RAS | | | ORAS | | |
|---|---|---|---|---|---|---|
| $N_{proc}$ | 32 | 64 | 128 | 32 | 64 | 128 |
| Global mesh | 55 | 53 | 55 | 54 | 53 | 56 |
| Global matrix | 41 | 22 | 12 | 42 | 22 | 11 |
| Local operators | 74 | 34 | 17 | 75 | 35 | 17 |
| Solver | 348 | 276 | 196 | 49 | 68 | 37 |
| Preconditioner | 36 | 22 | 11 | 17 | 12 | 5 |

TABLE 9

*The time in seconds for each phase of the method for the sphere with varying grid spacings. The weak scalability of these methods is evaluated by choosing $N_S = N_{proc}$ and using $h = 1/125, h = 1/175, h = 1/250$ for $N_{proc} = 32, 64, 128$, respectively. The first three phases are independent of the number of iterations used in the solution phase. Note that only one of the last two phases needs to be used in practice. The global meshing time grows in proportion to the problem size due to the sequential parts of that procedure.*

| | RAS | | | ORAS | | |
|---|---|---|---|---|---|---|
| $N_{proc}$ | 32 | 64 | 128 | 32 | 64 | 128 |
| Global mesh | 12 | 25 | 55 | 13 | 25 | 56 |
| Global matrix | 11 | 11 | 12 | 10 | 11 | 11 |
| Local operators | 17 | 17 | 17 | 16 | 17 | 17 |
| Solver | 29 | 89 | 196 | 21 | 30 | 37 |
| Preconditioner | 5 | 7 | 11 | 4 | 5 | 5 |

**7. Conclusion.** This paper develops and evaluates RAS and ORAS DD solvers for the solution of surface intrinsic elliptic PDEs discretized by the CPM. The splitting of the global problem and the construction of the local problems has been kept general through the use of a graph partitioner and the construction of effective boundary geometry that is independent of the irregular subdomains produced by the partitioning. To facilitate the implementation of ORAS methods, a new discretization of the Robin boundary operator for the CPM was given and verified to be first order accurate for a test case. The presence of cross points in the splitting was identified as the source of instability of the ORAS iteration with small values of the Robin parameter. A modified scheme, using larger parameters in the vicinity of cross points, was presented and dramatically improves the performance of the ORAS methods. The solvers were shown to be a viable and effective approach to the parallelized solution of equations arising from a CPM discretization of surface intrinsic elliptic PDEs. As noted in section 6.6, RAS and ORAS are favored over sparse direct solvers, echoing the observations in [7].

Only single-level methods were developed and evaluated herein, and the results of section 6.6 show the need for a two-level solver to obtain good parallel scalability. The goal is to obtain convergence rates with reduced dependence on the number of subdomains. The addition of a coarse space compatible with the CPM is a particularly interesting direction for future work. For scalar elliptic and elasticity problems, much progress has been made on the introduction of coarse spaces; see [10] for overlapping Schwarz methods and [9] in the context of BDDC (balancing domain decomposition by constraints). For the nonoverlapping optimized Schwarz case, see [12, 17]. Exploring these approaches in the embedding space is beyond the scope of this paper but is part of ongoing work.

**Acknowledgment.** The preliminary work of Nathan King helped inspire this project.

REFERENCES

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41, https://doi.org/10.1137/S0895479899358194.

[2] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet, *Hybrid scheduling for the parallel solution of linear systems*, Parallel Comput., 32 (2006), pp. 136–156.

[3] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, 1997, pp. 163–202.

[4] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517, https://doi.org/10.1137/S0036144502417715.

[5] M. Bertalmio, 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.

[6] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797, https://doi.org/10.1137/S106482759732678X.

[7] 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.

[8] J. Chu and R. Tsai, *Volumetric variational principles for a class of partial differential equations defined on surfaces and curves*, Res. Math. Sci., 5 (2018), 19.

[9] C. R. Dohrmann, K. H. Pierson, and O. B. Widlund, *Vertex-based preconditioners for the coarse problems of BDDC*, SIAM J. Sci. Comput., 41 (2019), pp. A3021–A3044, https://doi.org/10.1137/19M1237557.

[10] C. R. Dohrmann and O. B. Widlund, *On the design of small coarse spaces for domain*

*decomposition algorithms*, SIAM J. Sci. Comput., 39 (2017), pp. A1466–A1488, https://doi.org/10.1137/17M1114272.

[11] V. Dolean, P. Jolivet, and F. Nataf, *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*, SIAM, Philadelphia, 2015, https://doi.org/10.1137/1.9781611974065.

[12] O. Dubois, M. J. Gander, S. Loisel, A. St-Cyr, and D. B. Szyld, *The optimized Schwarz method with a coarse grid correction*, SIAM J. Sci. Comput., 34 (2012), pp. A421–A458, https://doi.org/10.1137/090774434.

[13] G. Dziuk and C. Elliott, *Surface finite elements for parabolic equations*, J. Comput. Math., 25 (2007), pp. 385–407.

[14] M. S. Floater and K. Hormann, *Surface parameterization: A tutorial and survey*, in Advances in Multiresolution for Geometric Modelling, Math. Vis., Springer, 2005, pp. 157–186.

[15] M. J. Gander, *Optimized Schwarz methods*, SIAM J. Numer. Anal., 44 (2006), pp. 699–731, https://doi.org/10.1137/S0036142903425409.

[16] M. J. Gander and F. Kwok, *Best Robin parameters for optimized Schwarz methods at cross points*, SIAM J. Sci. Comput., 34 (2012), pp. A1849–A1879, https://doi.org/10.1137/110837218.

[17] R. Haferssas, P. Jolivet, and F. Nataf, *An adaptive coarse space for P. L. Lions algorithm and optimized Schwarz methods*, in Domain Decomposition Methods in Science and Engineering XXIII, Lect. Notes Comput. Sci. Eng. 116, Springer, Cham, 2017, pp. 43–53, https://doi.org/10.1007/978-3-319-52389-7_4.

[18] 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.

[19] S. Loisel, *Condition number estimates for the nonoverlapping optimized Schwarz method and the 2-Lagrange multiplier method for general domains and cross points*, SIAM J. Numer. Anal., 51 (2013), pp. 3062–3083, https://doi.org/10.1137/100803316.

[20] C. Macdonald and S. Ruuth, *Level set equations on surfaces via the closest point method*, J. Sci. Comput., 35 (2008), pp. 219–240.

[21] C. B. Macdonald, J. Brandman, and S. J. Ruuth, *Solving eigenvalue problems on curved surfaces using the closest point method*, J. Comput. Phys., 230 (2011), pp. 7944–7956.

[22] C. B. Macdonald, B. Merriman, and S. J. Ruuth, *Simple computation of reaction-diffusion processes on point clouds*, Proc. Natl. Acad. Sci. USA, 110 (2013), pp. 9209–9214.

[23] 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, https://doi.org/10.1137/080740003.

[24] T. März and C. B. Macdonald, *Calculus on surfaces with general closest point functions*, SIAM J. Numer. Anal., 50 (2012), pp. 3303–3328, https://doi.org/10.1137/120865537.

[25] I. May, *DD-CPM*, https://bitbucket.org/mayianm/dd-cpm/, 2020.

[26] I. May, R. D. Haynes, and S. J. Ruuth, *Domain Decomposition for the Closest Point Method*, preprint, https://arxiv.org/abs/1907.13606, 2019; to appear in Domain Decomposition Methods in Science and Engineering XXV, Lect. Notes Comput. Sci. Eng. 138, Springer.

[27] A. Petras and S. Ruuth, *PDEs on moving surfaces via the closest point method and a modified grid based particle method*, J. Comput. Phys., 312 (2016), pp. 139–156.

[28] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Numer. Math. Sci. Comput., The Clarendon Press, Oxford University Press, 1999.

[29] M. Reuter, F.-E. Wolter, and N. Peinecke, *Laplace–Beltrami spectra as 'shape-DNA' of surfaces and solids*, Comput.-Aided Des., 38 (2006), pp. 342–366.

[30] 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.

[31] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003, https://doi.org/10.1137/1.9780898718003.

[32] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.

[33] A. St-Cyr, M. J. Gander, and S. J. Thomas, *Optimized multiplicative, additive, and restricted additive Schwarz preconditioning*, SIAM J. Sci. Comput., 29 (2007), pp. 2402–2425, https://doi.org/10.1137/060652610.

[34] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer, 2005.

[35] G. Turk and M. Levoy, *Zippered polygon meshes from range images*, in Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94), ACM, 1994, pp. 311–318.