Domain Decomposition Solvers and Preconditioners for the Implicit Closest Point Method

by

Ian May

B.Sc., Colorado State University, 2016

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in the Department of Mathematics Faculty of Science

© Ian May 2018

SIMON FRASER UNIVERSITY
Summer 2018

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.
Approval

Name: Ian May
Degree: Master of Science (Mathematics)
Title: Domain Decomposition Solvers and Preconditioners for the Implicit Closest Point Method

Examining Committee: Chair: Ralf Wittenberg
Associate Professor

Steven Ruuth
Senior Supervisor
Professor

Ronald Haynes
Supervisor
Professor
Department of Mathematics and Statistics
Memorial University of Newfoundland

Benjamin Adcock
Internal Examiner
Assistant Professor

Date Defended: July 17th, 2018
Abstract

The numerical treatment of surface intrinsic elliptic PDE presents several interesting challenges over those posed on flat space. The *implicit closest point method*, (*iCPM*) is an embedding method well suited to these problems, and allows the treatment of general surfaces, $S$. An extension operator brings surface bound information into the embedding space, to be constant in the surface normal direction, and allows the solution of the problem by standard methods. The positive Helmholtz equation, $(c - \Delta_S) u = f$, is considered with tensor product barycentric Lagrange interpolation defining the extension operator and standard second-order centered differences for the ambient Laplacian. Under this scheme, a non-symmetric system with poor sparsity is obtained, which reduces the performance of iterative solvers and motivates the development of specialized solvers. Optimized restricted additive Schwarz (ORAS) methods are well suited to this task and are formulated for these problems. The interesting geometry of the problem presents challenges for the construction of subdomains as well as the enforcement of Robin boundary conditions. The developed solvers perform well over a range of test problems with the optimized methods providing a distinct advantage. With Krylov acceleration, convergence is obtained rapidly with diminished difference between the optimized and non-optimized methods.

**Keywords:** Closest point method, domain decomposition, parallel numerical linear algebra
Table of Contents

Approval ii
Abstract iii
Table of Contents iv
List of Tables vi
List of Figures vii

1 Introduction 1
   1.1 Numerical approaches to surface PDEs 2
   1.2 Domain decomposition methods 2
   1.3 Outline 3

2 The implicit closest point method 5
   2.1 Direct closest point method and notation 5
      2.1.1 The extension operator 5
      2.1.2 Discretization of the direct CPM 6
   2.2 The implicit formulation 8

3 Domain Decomposition 10
   3.1 Continuous formulation 10
      3.1.1 Contraction factor analysis 13
   3.2 Restricted additive Schwarz and generalizations 16
      3.2.1 Use as a preconditioner 18
   3.3 Example: Poisson equation in the plane 19

4 Subdomain Construction 23
   4.1 Graph partitioning problem 23
      4.1.1 Nearest neighbor coupling 25
      4.1.2 Full coupling 25
      4.1.3 Interface alignment 26
List of Tables

Table 6.1 Introduced parameters .............................................. 38
Table 6.2 Circle parameter sweep ............................................. 39
Table 6.3 Circle parameter sweep, higher resolution ......................... 39
Table 6.4 Sphere parameter sweep ............................................. 43
Table 6.5 Sphere parameter sweep, high resolution .......................... 43
Table 6.6 Torus parameter sweep, low resolution ............................ 46
Table 6.7 Torus parameter sweep .............................................. 46
Table 6.8 Triangulated surface parameter sweep, low resolution ............... 50
Table 6.9 Triangulated surface parameter sweep, high resolution ............. 50
Table 6.10 Strong scaling, circle .............................................. 54
Table 6.11 Strong scaling, torus .............................................. 54
Table 6.12 Weak scaling, circle .............................................. 55
Table 6.13 Weak scaling, torus .............................................. 56
List of Figures

Figure 2.1 Closest point discretization ............................................ 7
Figure 3.1 Schematic for domain decomposition ................................... 11
Figure 3.2 Splitting of $\mathbb{R}^2$ ..................................................... 14
Figure 3.3 Oblique Robin contraction factor ....................................... 17
Figure 3.4 DD in the plane, effect of subdomain count ......................... 20
Figure 3.5 DD in the plane, effect of overlap width ............................. 21
Figure 3.6 DD in the plane, effect of Robin weight ............................... 22
Figure 4.1 Graph connectivity options ............................................. 24
Figure 4.2 Comparison of disjoint partition interfaces .......................... 26
Figure 4.3 Disjoint interface alignment ............................................. 26
Figure 4.4 Subdomain construction .................................................. 28
Figure 4.5 Conormal smoothing ..................................................... 30
Figure 5.1 Robin condition convergence study .................................... 36
Figure 6.1 (O)RAS Relative performance, circle ................................. 40
Figure 6.2 (O)RAS Relative performance, high resolution circle .......... 41
Figure 6.3 Subdomain count influence, circle .................................... 41
Figure 6.4 Overlap influence, circle ................................................. 42
Figure 6.5 (O)RAS Relative performance, sphere ................................. 44
Figure 6.6 (O)RAS Relative performance, high resolution sphere .......... 44
Figure 6.7 Subdomain count influence, sphere .................................... 45
Figure 6.8 Overlap width influence, sphere ....................................... 45
Figure 6.9 (O)RAS Relative performance, low resolution torus ............. 47
Figure 6.10 (O)RAS Relative performance, high resolution torus ........... 47
Figure 6.11 Subdomain count influence, torus ................................... 48
Figure 6.12 Overlap width influence, torus ....................................... 48
Figure 6.13 Stanford bunny triangulation .......................................... 49
Figure 6.14 (O)RAS Relative performance, low resolution Stanford Bunny 51
Figure 6.15 (O)RAS Relative performance, high resolution Bunny .......... 51
Figure 6.16 Subdomain count influence, Stanford Bunny ..................... 52
Figure 6.17 Overlap width influence, Stanford Bunny . . . . . . . . . . . . . . . 52
Chapter 1

Introduction

Elliptic partial differential equations form a fundamental class of partial differential equations. Due to their inherently global nature, discretizations of these problems generally lead to large coupled systems of equations whose effective computational treatment requires care. The canonical example of an elliptic PDEs, and arguably the most prevalent and recognizable PDE of any sort, is Laplace’s equation which arises naturally in a diverse range of applications. The Laplace operator generalizes away from flat space to (smooth enough) surfaces quite readily but the numerical treatment of such problems remains an active pursuit. We take as an indicative model problem the surface intrinsic positive Helmholtz equation

\[(c - \Delta_S) u = f \quad \text{on } S,\]  

where \(\Delta_S\) is the Laplace-Beltrami operator defined with respect to a smooth surface \(S\), taken without boundary for the moment, and \(c \in \mathbb{R}^+\) is some positive constant. The Laplace-Beltrami operator acts within the surface, accounting for its geometric properties intrinsically. The standard Laplacian enjoys a massive amount of computational support with no shortage of relevant and well studied discretizations. Elliptic PDEs posed over surfaces, as in the model equation (1.1), present interesting challenges not found in the flat case and have not seen as much development. In what follows, several numerical schemes for this problem are presented with their relative merits and faults. Though we select the closest point method specifically for the work here, all schemes yield a linear system of equations that must be solved in some reasonable fashion.

Generally, as the discretization becomes finer and the systems grow in size they will also see rising condition numbers and the convergence behavior of standard iterative methods on their own will diminish. Preconditioning can address these faults and return the convergence behavior. Furthermore, the advent of parallel preconditioners and solvers address the issue of system size by allowing a splitting of the system into separate parts that may be treated by distinct computers linked through a network interface (distributed memory.
computing. Herein, we focus on the use of domain decomposition methods for the solution or preconditioning of these problems while opening opportunities for parallel computation.

1.1 Numerical approaches to surface PDEs

Several schemes exist for the discretization of equation (1.1) with a few important and indicative methods described here. A first class of methods seek to parameterize the surface and utilize standard numerical techniques for flat space equations with respect to this parameterization. This is highly effective as it allows one to use their experience with standard numerical methods for flat problems. However it is quite problem specific and inhibits exploration of solution behavior over a collection of surfaces. Many surfaces can only be globally parametrized with the introduction of a coordinate singularity, like the poles of a sphere, requiring special treatment. Multiple charts could be used to avoid coordinate singularities, such as the cubed sphere [17], but this again inhibits exploration of other surfaces.

Another class of methods avoids the parameterization problem by finding a triangulation of the surface and discretizing the equation via finite elements. This approach preserves the dimensionality as before and further gives rise to highly sparse and symmetric matrices. Beyond the truncation error present in the scheme there is an additional source of error coming now from the approximation of the surface by the triangulation. This is sensitive to mesh quality and often requires re-meshing for triangulations coming from computer graphics scenarios. The relative ease of \( h \)-adaptivity is nullified here as any subdivision scheme now needs to produce finer triangles whose vertices still lie on the surface to faithfully capture the underlying geometry.

The level set methods strike a nice middle-ground between the parametric approach and the direct discretization of the surface. Here, the surface is only considered implicitly as the level set of some auxiliary function over a higher dimensional space where the surface may be embedded. These embedding methods avoid discretizing the entire ambient space and instead pose the problem only on a narrow band, as the closest point method does (cf. Section 2.1.2), but have the key drawback of introducing degenerate equations [18] that require special care to solve and do not generalize easily to higher order surface operators.

1.2 Domain decomposition methods

The linear systems eventually arising from this discretization are somewhat sparse, non-symmetric, and most conveniently cast in an unassembled form (cf. equation (2.11) where the individual factors may not be stored). The first and last properties motivate the use of iterative linear solvers while the second reduces the list of appropriate solvers somewhat. The standard Laplacian exhibits the same sparsity under many discretizations and its widespread applicability has resulted in a great number of solution techniques. These equations are well suited to the solution by domain decomposition methods. These methods originate from the
analytic treatment of such equations, but have found new utility in reducing the solution cost of associated numerical schemes. Indeed, the theoretical performance of these solution schemes has mostly been established in the context of Laplacian equations.

Though the discrete form of \( (1.1) \) lacks some of the nice properties of the standard Laplacian, its treatment by these methods seems most promising. Several factors reduce the performance of black-box solvers. Ideally these operators are left unassembled, yet most black box solvers and preconditioners require access to the assembled matrix. For instance, algebraic multigrid and domain decomposition methods typically infer the coupling of the unknowns by examining the non-zero structure of the matrix. Improved transmission conditions within the schemes are most cleanly treated from the continuous perspective. Indeed, as shall be seen, these conditions are imposed by modifying one factor of the global operator with no simple approach being available from the final matrix form. Finally the closest-point operators are weakly non-local, requiring special care.

1.3 Outline

In what follows, the discretization of the model equation is discussed in Chapter 2 where a few schemes are considered. The closest point schemes are recalled in some detail, with some important properties highlighted along the way, in Sections 2.1 and 2.2.

Following this, the Schwarz type domain decomposition schemes are defined in Chapter 3 with their continuous formulations detailed in 3.1. Section 3.2 therein moves these continuous formulations into the discrete, exposing their utility for the numerical treatment of problems such as equation \( (1.1) \) and casts the whole scheme in a way amenable to Krylov acceleration.

With this prerequisite information in hand the application of domain decomposition to closest point problems proceeds in Chapter 4 where the meshes constructed for the closest point problem are partitioned and useful subdomains are constructed. Section 4.2 highlights some challenges in this construction, seemingly unique to these discretizations, but whose resolution may well be of use outside of this application.

Chapter 5 continues by formulating transmission operators well suited to these problems and considers their implementation. Here again lie interesting and unique challenges, though again the solutions presented here may find use in other contexts.

With these thoughts developed, the full scheme is developed and tested in Chapter 6 where all of the introduced parameters are swept over and the convergence results are catalogued. The behavior of the developed methods is compared to what we would expect from experience with similar methods in flat space.

Finally, the thesis is concluded with some thoughts on how and when to select a solver from these definitions as well as some possible directions for future work. An appendix is also
given which comments on some implementation details and points to a public repository of all code developed for this project.
Chapter 2

The implicit closest point method

The closest point schemes [18, 13, 11] share many of the guiding principles of the level set formulation over unfitted bulk meshes mentioned in Section 1.1. The surface is only treated implicitly, thus avoiding the need for parameterizations. The construction of more resolved solutions is simple, as compared to the triangulated schemes requiring a new mesh. In contrast to the level set formulation, this method does not lead to a degenerate form of the governing PDE. To ease the exposition of the implicit closest point method, a direct formulation of the closest point method is reviewed first to establish notation and demonstrate some important properties.

2.1 Direct closest point method and notation

The direct version of the scheme, introduced in [18], is reproduced specifically for equation (1.1) over a smooth surface $S$, taken without boundary initially, and embedded in $\mathbb{R}^d$. The treatment of surfaces with boundaries are addressed below in Chapter 5. Since the closest point method is an embedding scheme, moving between the surface and the embedding space is critical to its definition. The central construct here is the extension operator $E$, which will be defined in the following section along with a some of its useful properties.

2.1.1 The extension operator

The schemes hinge on the definition of the closest point function

$$CP_S : \mathbb{R}^d \to S,$$

which identifies each point in $\mathbb{R}^d$ with the point nearest to it on the surface. The closest point function is defined almost everywhere with the exceptions arising from those few points whose closest points are not unique, for instance the center of a sphere, or the axis of revolution defining a torus. With the closest point function, surface intrinsic quantities and functions can be extended naturally into the embedding space by composition. The action
of this composition induces the extension operator

\[ Ef := f \circ CP_S , \] (2.2)

for functions \( f : S \to \mathbb{R}^n \) defined intrinsic to the surface. The extended function now takes values on the embedding space that are constant along the surface normal directions, as all points in this direction share the same closest point, and recovers the original function when restricted back to the surface. These surface normals will be of use throughout and are denoted \( \hat{n} \), customarily pointing away from the surface. Though the surface normal vectors are only defined on the surface, they can be extended to the embedding space by the same closest point identification and will often be used there. We use no special notation identifying this extension as the use is unambiguous.

For sufficiently smooth surfaces, \( S \), and functions, \( f : S \to \mathbb{R} \), the gradient of their extension lies tangential to the surface, \( \nabla (Ef) \perp \hat{n} \). This arises from the extended function being constant in the surface normal direction. This identification leads to the first important equivalence: the surface intrinsic gradient of a function is the restriction of the gradient of the extended function back to the surface [18, 13]

\[ \nabla_S f = (\nabla (Ef))|_S . \] (2.3)

Consider a smooth vector field, \( v \), over a region of the embedding space containing the surface. If this vector field is tangential to all constant distance displacements of the surface, then the surface intrinsic divergence of this field restricted to the surface is equivalent to the restriction of the ambient divergence. This second important equivalence presents itself as [18, 13]

\[ \nabla_S \cdot (v|_S) = (\nabla \cdot v)|_S . \] (2.4)

Finally, the action of Laplace-Beltrami operator over \( S \) can be now formulated in terms of the standard Laplacian on \( \mathbb{R}^d \)

\[ \Delta_S u = (\Delta (Eu))|_S , \] (2.5)

which arises through the combination of the prior two principles [18]. This connection from the surface to the embedding space through the extension operator and the closest point function forms the foundation of the closest point method.

### 2.1.2 Discretization of the direct CPM

The discussion leading to equation (2.5) indicates that we may apply the Laplace-Beltrami operator to a function defined on a surface through the standard Laplacian on the embedding space. This concept induces a companion equation to (1.1), now posed on the embedding space.
space

$$\begin{cases}
(c - \Delta E)u = Ef & \text{in } \Omega_\gamma(S) \\
\frac{\partial u}{\partial n} = 0 & \text{on } \partial\Omega_\gamma(S)
\end{cases}
$$

(2.6)

where $\Omega_\gamma(S) \subset \mathbb{R}^d$ is delimited by two displacements of the surface with a total separation of width $\gamma$ as illustrated in Figure 2.1. The solution to equation (1.1) is then recovered from (2.6) by restriction to $S$.

Figure 2.1: The stencil for the centered difference Laplacian is visible for one active node, marked as a large filled circle. Two of the neighboring nodes are ghost nodes that only appear indirectly. From here, the stencils for extension of all nodes needed for this centered difference are gathered into the three overlapping blue squares, sized for bi-cubic interpolation. In the direct closest point method, the node marked by the large filled circle depends on all nodes in the blue region.

The closest point method provides the connection to the embedding space, and leaves open the option to use a multitude of discretizations once there. A fairly standard formulation places a structured, uniform, grid over $\mathbb{R}^d$ with all nodes lying within $\Omega_\gamma(S)$ being identified as active nodes and collected into the set $N_A$, with cardinality $N_A$. Ghost nodes layered over the active nodes will be needed momentarily for the completion of differential operator stencils, and are gathered into the set $N_G$, with cardinality $N_G$.

The discrete extension operator, $E \in \mathbb{R}^{(N_A+N_G) \times N_A}$, identifies the active and ghost nodes with their closest points and propagates information from the active nodes out to all nodes by constructing a local interpolant from the active node information. More concretely, consider a node $x_i \in N_A \cup N_G$ with closest point $CP_S(x_i)$ and a vector $\mathbf{u} \in \mathbb{R}^{N_A}$ containing the function values over the active nodes. Tensor product barycentric Lagrange interpolation [4] is used to construct a polynomial interpolant, of degree $p$ in each spatial variable, from the $(p+1)^d$ cube of active nodes surrounding $CP_S(x_i)$. The evaluation of this interpolant at the closest point then extends the sampled function out onto the active and ghost nodes.
The extension operator inherits linearity from the polynomial interpolation problem and is written as a matrix with \((p + 1)^d\) non-zeros per row, as the interpolation weights associated with that node’s stencil. With this notion established the role of the bandwidth \(\gamma\) becomes apparent, the cube of active nodes forming the interpolation stencil must all lie within the band and be characterized as active. For interpolation with degree \(p\) polynomials one finds that a bandwidth of
\[
\gamma = (p + 2) h \sqrt{d},
\] (2.7)
with \(h\) as the grid spacing, suffices (in contrast to the band in [18] which must include ghost nodes as well for the explicit formulation).

The Laplacian on the embedding space is treated herein by the standard second order accurate centered difference formula. With the values over the ghost nodes being populated by the extension operator, and thus constant along surface normals, the homogeneous Neumann conditions in (2.6) arise naturally. The discrete Laplacian, \(\Delta^h \in \mathbb{R}^{N_A \times (N_A + N_G)}\), has its image only over the active nodes. The composition \(\Delta^h_{S,dir} = \Delta^h E\in \mathbb{R}^{N_A \times N_A}\) then acts only over the active nodes with the ghost nodes appearing implicitly. The discrete form of (2.6) then becomes
\[
(cI - \Delta^h_{S,dir}) u = f,
\] (2.8)
with \(u, f \in \mathbb{R}^{N_A}\). The right hand side is evaluated over the active nodes by use of the closest point function. This formulation discretizes equation (1.1) using the direct formulation of the closest point method for the Laplace-Beltrami operator. Finally, Figure 2.1 displays the total effect of this discretization by considering one active node in particular and tracing all dependencies that it has.

### 2.2 The implicit formulation

The direct form of the discrete Laplace-Beltrami operator defined in Section 2.1.2, given by \(\Delta^h_{S,dir} = \Delta^h E\), is unsuitable for use in the discretization of elliptic equations like (1.1) arising from implicit time integrations. The spectrum of this operator has a few small positive eigenvalues [13] that lead to instability when used within implicit time integration schemes. The *implicit closest point method* (iCPM) introduced in [13] resolves these issues with a modified form of the discrete Laplace-Beltrami operator.

Looking at a semi-discrete form of the closest point Laplace-Beltrami operator (embedded in two dimensions for clarity of exposition)

\[
\Delta^h_{S,dir} u = \frac{1}{h^2} \left( -4 u (CP_S(x, y)) + u (CP_S(x - h, y)) + u (CP_S(x, y + h)) + u (CP_S(x + h, y)) + u (CP_S(x, y - h)) \right),
\] (2.9)
one recognizes that the diagonal term \( u(\text{CP}_S(x,y)) \) is redundantly extended since \( u(x,y) = u(\text{CP}_S(x,y)) \) already holds. Returning to the general case in \( d \)-dimensions, the diagonal of the ambient discrete Laplacian may be removed before composition with the extension

\[
\Delta_{S,\text{imp}}^h = -\frac{2d}{h^2} I + \left( \frac{2d}{h^2} I + \Delta^h \right) E ,
\]

which avoids these self-interpolation effects and produces the essential operator within the implicit closest point method [13]. Once again considering Figure 2.1, the implicit form would give a similar picture with the selected node (solid circle) no longer being extended from its interpolation stencil. From this the implicit form of the discrete model equation is given by

\[
\left( c + \frac{2d}{h^2} \right) I - \left( \frac{2d}{h^2} I + \Delta^h \right) E \right) \mathbf{u} = \mathbf{f} .
\]

Interestingly, taking \( c = 0 \) here yields the same discrete operator formed in [11] for treating Laplace-Beltrami eigenvalue problems. Directly discretizing this operator to find the direct form in Section 2.1.2 gives a matrix with a great number of false eigenvalues clustered near 0. Regularization of that operator was achieved by equation (2.10) with \( c = 0 \), removing the issues surrounding the near-null eigenfunctions.

The remainder of this thesis will use the implicit formulation of equation (2.10), and as such the subscript \( \text{imp} \) will be dropped. This maintains the applicability of the developed schemes to transient problems and will keep the behavior reasonable in the limit of small \( c \). Additionally, we may hope that the regularization of the spectrum will improve the performance of the iterative solvers explored herein.
Domain Decomposition

Domain decomposition seeks to solve a single large problem by splitting it into a collection of smaller subproblems. Decoupling these smaller problems from each other exposes parallelism [21] and allows one to deal with much smaller matrices. These schemes are generally formulated in one of two ways. First, the continuous domain is split and appropriate boundary conditions are imposed on the newly introduced interfaces [6]. These local problems are then discretized in a way that respects the global discretization. Or second, the discrete subproblems may be defined directly through restrictions [21] and modifications [22] of the linear system defining the global system. The first approach is more intuitive, and as shall become apparent in Chapter 5, is more natural for the application of these ideas to closest point problems. Here the continuous and algebraic formulations are detailed independent of the closest point method to simplify the presentation. A number of great treatises on domain decomposition exist, including [21, 25, 16] and [6]. The notation and style is similar to that used within [6].

3.1 Continuous formulation

The domain decomposition schemes treated herein will be developed for a generic equation posed on $\Omega \subset \mathbb{R}^n$

$$\begin{cases}
  \mathcal{L}u = f & \text{in } \Omega \\
  \mathcal{B}u = g & \text{on } \partial \Omega
\end{cases} \quad (3.1)$$

with $\mathcal{L}$ a linear elliptic operator and $\mathcal{B}$ a linear boundary operator. This global domain is partitioned into $N_S$ disjoint subdomains, $\bar{\Omega}_j$. These partitions are then contained inside a set of overlapped subdomains, $\bar{\Omega}_j \subset \Omega_j$, possibly chosen to introduce some convenient geometry. Further, denote the portion of the artificial interface introduced by $\partial \Omega_j$ lying in the $k^{th}$ disjoint subdomain by $\Gamma_{jk} = (\partial \Omega_j \cap \Omega_k) \setminus \partial \Omega$ (note that $\Gamma_{jk} \neq \Gamma_{kj}$). A representation of the classic keyhole domain (used in Schwarz’s original work [20]) can be seen in Figure 3.1.
The additive Schwarz method begins with an approximate solution, $u_j^{(0)}$, on each overlapped subdomain (or really just over the interfaces) and solves the subproblems

$$
\begin{align*}
\mathcal{L}u_j^{(n+1)} &= f & \text{in } \Omega_j \\
\mathcal{B}u_j^{(n+1)} &= g & \text{on } \partial \Omega \cap \partial \Omega_j, \\
\mathcal{T}_{jk}u_j^{(n+1)} &= \mathcal{T}_{jk}u_k^{(n)} & \text{on } \Gamma_{jk}, \ \forall \ k,
\end{align*}
$$

(3.2)

over all subdomains $j = 1, 2, \ldots, N_S$ to produce improved approximations to the solutions. An important addition here is the appearance of linear boundary operators $\mathcal{T}_{jk}$ linking the subsolutions together. Due to this role, the operators $\mathcal{T}_{jk}$ are usually called transmission operators. Two forms of transmission operator are of interest here

$$
\begin{align*}
\mathcal{T}_{jk} &= \text{identity} & \text{or} \\
\mathcal{T}_{jk} &= \left( \frac{\partial}{\partial \hat{n}_{jk}} + \alpha \right),
\end{align*}
$$

(3.3) \quad (3.4)

with the first enforcing Dirichlet conditions over the artificial boundaries and the second enforcing Robin conditions with $\hat{n}_{jk}$ as the outward normal direction on $\Gamma_{jk}$. The weight $\alpha$ present in the Robin condition is free and may be optimized to provide the fastest convergence. The boundary locations and normal directions do not coincide, so $\mathcal{T}_{jk} \neq \mathcal{T}_{kj}$.

These subproblems are solved in a decoupled fashion, with the right-hand sides of the transmission conditions being updated only after all subproblems have updated solutions. Importantly, all operators present are linear, and hence from these subproblems we may subtract the exact solution restricted to each overlapping subdomain to form the local error.
\[ e_j^{(n)} = u_j^{(n)} - u|_{\Omega_j}. \] In the interior of \( \Omega_j \) one finds

\[
\mathcal{L}e_j^{(n)} = \mathcal{L}u_j^{(n)} - \mathcal{L}u \tag{3.5}
\]
\[
= f|_{\Omega_j} - f \tag{3.6}
\]
\[
= 0, \quad \text{(in } \Omega_j), \tag{3.7}
\]

where the global boundary conditions also homogenize to yield the equivalent subproblems

\[
\begin{cases}
\mathcal{L}e_j^{(n+1)} = 0 & \text{in } \Omega_j \\
\mathcal{B}e_j^{(n+1)} = 0 & \text{on } \partial\Omega \cap \partial\Omega_j, \\
T_{jk}e_j^{(n+1)} = T_{jk}e_k^{(n)} & \text{on } \Gamma_{jk}, \ \forall \ k,
\end{cases} \tag{3.8}
\]

which show how the error evolves.

Following [6], a second reformulation of this scheme will be useful in the remaining discussion. Take \( \chi_j \in C^\infty(\Omega) \) as a partition of unity, \( \sum_j \chi_j(x) = 1 \) for all \( x \in \Omega \), where the supports respect the overlapping subdomains, \( \text{supp}(\chi_j) \subset \Omega_j \). These induce extension operators, \( \mathcal{E}_j \), bringing functions defined on \( \Omega_j \) to those on \( \Omega \) through extension by zero. The extension operators satisfy the relation

\[ w = \sum_j \mathcal{E}_j \left( \chi_j w|_{\Omega_j} \right), \tag{3.9} \]

for functions \( w \) supported on the global domain. This allows the construction of a global solution from the local solutions of the problems in equation (3.2) through this partition of unity

\[ u^{(n)} = \sum_j \mathcal{E}_j \left( \chi_j u_j^{(n)} \right), \tag{3.10} \]

which allows the formation of the global residual at the \( n^{th} \) iteration as \( r^{(n)} := f - \mathcal{L}u^{(n)}. \)

In the residual formulation (3.2) is equivalent to solving the problems

\[
\begin{cases}
\mathcal{L}v_j^{(n)} = r^{(n)} & \text{in } \Omega_j \\
\mathcal{B}v_j^{(n)} = 0 & \text{on } \partial\Omega \cap \partial\Omega_j, \\
T_{jk}v_j^{(n)} = 0 & \text{on } \Gamma_{jk}, \ \forall \ j
\end{cases} \tag{3.11}
\]

with the global solution update

\[ u^{(n+1)} = u^{(n)} + \sum_j \mathcal{E}_j \left( \chi_j v_j^{(n)} \right). \tag{3.12} \]
To see this equivalence consider the local update given as \( u^{(n+1)} = u^{(n)} \mid_{\Omega_j} + v_j^{(n)} \). Inside the \( j \)th overlapped subdomain

\[
\mathcal{L} \left( u^{(n)} \mid_{\Omega_j} + v_j^{(n)} \right) = \mathcal{L} u^{(n)} \mid_{\Omega_j} + \mathcal{L} v_j^{(n)} = (\mathcal{L} u^{(n)} + r^{(n)}) \mid_{\Omega_j} = f_{\mid_{\Omega_j}},
\]

which is consistent with (3.2) inside \( \Omega_j \). The recognition that on the global domain boundary the subsolutions are never updated, since \( B u_j^{(n+1)} = g = B u_j^{(n)} \), directly yields the homogeneous condition on \( v_j^{(n)} \) over \( \partial \Omega \cap \partial \Omega_j \). Finally the transmission condition homogenizes since \( T_{jk} u_j^{(n+1)} = T_{jk} u_j^{(n)} \) holds over the boundary and \( T_{jk} \) is linear. This arises from (3.12) as a consequence of \( \chi_j \) being zero over all of \( \partial \Omega_j \setminus \partial \Omega \).

Here the same iteration has been cast in three forms. The first in (3.2) is intuitive and lays out the governing idea behind the scheme. The second form in (3.8) shows how the error evolves through the iteration and will be of use for showing convergence of the iteration and estimating the rate. The final iteration defined by (3.11) homogenizes the transmission conditions and casts the update in terms of the residual. The former property greatly simplifies the implementation of the scheme as the transmission operator does not need to be evaluated over the other subsolutions. The latter ultimately allows the whole iteration to be embedded within a Krylov accelerator as in Section 3.2.1.

### 3.1.1 Contraction factor analysis

Results regarding the convergence rates of the presented domain decomposition methods are available only in special cases. An indicative case where specific results can be obtained is to consider \( \mathbb{R}^2 \) split into two semi-infinite half-planes, \( \Omega_1 = (-\infty, \delta) \times \mathbb{R} \) and \( \Omega_2 = (0, \infty) \times \mathbb{R} \). The target problem is still the positive Helmholtz model problem \( (c - \Delta) u = f, c > 0 \) now paired with a decay condition at infinity. Here the convergence analysis of [6] is generalized slightly to treat the typical Robin conditions as well as oblique Robin conditions where some tangential information is also incorporated. This does not improve the convergence rate, and in fact hinders it, but will be useful when considering the approximations made in Section 4.2.2. The additive optimized Schwarz method solves the problems

\[
\begin{cases}
(c - \Delta) u_1^{(n+1)} = f & \text{in } \Omega_1 \\
u_1^{(n+1)} \to 0 & \text{as } x \to -\infty , \\
T_{g}^+ u_1^{(n+1)} = T_{g}^+ u_2^{(n)} & \text{on } x = \delta
\end{cases}
\]

(3.16)
Figure 3.2: The splitting of $R^2$ used in the contraction factor analysis can be seen here. In particular, the direction along which the oblique Robin condition is enforced is shown relative to the outward pointing normal vectors for each subdomain.

and

$$
\begin{align*}
(c - \Delta) u_2^{(n+1)} &= f \quad \text{in } \Omega_2 \\
u_2^{(n+1)} &\to 0 \quad \text{as } x \to \infty , \\
\mathcal{T}_\theta^- u_2^{(n+1)} &= \mathcal{T}_\theta^- u_1^{(n)} \quad \text{on } x = 0
\end{align*}
$$

(3.17)

where the boundary operator $\mathcal{T}_\theta^\pm = (\pm \cos \theta \frac{\partial}{\partial x} + \sin \theta \frac{\partial}{\partial y} + \alpha)$ enforces an oblique Robin condition with $\theta$ as the angle between the normal and the direction that the derivative is enforced. Figure 3.2 shows this splitting and the oblique directions used in the transmission condition.

Writing these equations in terms of the solution error (cf. equation (3.8)) and taking the Fourier transform in the infinite $y$-direction we find the equations

$$
\begin{align*}
\left( c + k^2 - \frac{\partial^2}{\partial x^2} \right) \hat{e}_1^{(n+1)} &= 0 \quad \text{on } \Omega_1 \\
\hat{e}_1^{(n+1)} &\to 0 \quad \text{as } x \to -\infty , \\
\hat{\mathcal{T}}_\theta^+ \hat{e}_1^{(n+1)} &= \hat{\mathcal{T}}_\theta^+ \hat{e}_2^{(n)} \quad \text{on } x = \delta
\end{align*}
$$

(3.18)

and

$$
\begin{align*}
\left( c + k^2 - \frac{\partial^2}{\partial x^2} \right) \hat{e}_2^{(n+1)} &= 0 \quad \text{on } \Omega_2 \\
\hat{e}_2^{(n+1)} &\to 0 \quad \text{as } x \to \infty , \\
\hat{\mathcal{T}}_\theta^- \hat{e}_2^{(n+1)} &= \hat{\mathcal{T}}_\theta^- \hat{e}_1^{(n)} \quad \text{on } x = 0
\end{align*}
$$

(3.19)

where $k$ is the Fourier conjugate variable to $y$. The wavenumbers $k$ decouple entirely and this is now simply a system of second-order ordinary boundary value problems. Taking
\[ \lambda = \sqrt{c + k^2}, \]  
the solutions to these equations can be written as

\[
\begin{align*}
\hat{e}_1^{(n+1)}(x; k, c, \alpha, \delta, \theta) &= \gamma_{1,+}^{(n+1)} e^{\lambda(x-\delta)} + \gamma_{1,-}^{(n+1)} e^{-\lambda(x-\delta)} \\
\hat{e}_2^{(n+1)}(x; k, c, \alpha, \delta, \theta) &= \gamma_{2,+}^{(n+1)} e^{\lambda x} + \gamma_{2,-}^{(n+1)} e^{-\lambda x},
\end{align*}
\]

(3.20)

(3.21)

with \( \gamma_{1,-}^{(n+1)} = \gamma_{2,+}^{(n+1)} = 0 \) due to the conditions at infinity. Inserting these forms for the error modes into the transmission conditions yields

\[
\begin{align*}
\gamma_{1,+}^{(n+1)} (\alpha - ik \sin \theta + \lambda \cos \theta) &= \gamma_{2,-}^{(n)} (\alpha - ik \sin \theta - \lambda \cos \theta) e^{-\lambda \delta} \\
\gamma_{2,-}^{(n+1)} (\alpha - ik \sin \theta + \lambda \cos \theta) &= \gamma_{1,+}^{(n)} (\alpha - ik \sin \theta - \lambda \cos \theta) e^{-\lambda \delta}.
\end{align*}
\]

(3.22)

(3.23)

These two coefficients depend on each other at different iterations. This dependency can be removed by moving back an additional iteration, so that \( \gamma_{1,+}^{(n+1)} \) depends on \( \gamma_{1,+}^{(n-1)} \) instead of \( \gamma_{2,-}^{(n)} \) (and similarly for \( \gamma_{2,-}^{(n+1)} \)). Upon doing so one finds

\[
\begin{align*}
\gamma_{1,+}^{(n+1)} &= \gamma_{1,+}^{(n-1)} \left( \frac{\alpha - ik \sin \theta - \lambda \cos \theta}{\alpha - ik \sin \theta + \lambda \cos \theta} \right)^2 e^{-2 \lambda \delta} \\
\gamma_{1,+}^{(n+1)} &= \gamma_{1,+}^{(n-1)} \left( \frac{\alpha - ik \sin \theta + \lambda \cos \theta}{\alpha - ik \sin \theta + \lambda \cos \theta} \right)^2 e^{-2 \lambda \delta}.
\end{align*}
\]

(3.24)

(3.25)

The contraction factor (convergence factor in [6]) is defined as the amplitude ratio of each error mode between iterations

\[
\frac{\gamma_{1,+}^{(n+1)}}{\gamma_{1,+}^{(n-1)}} := \rho_{OR}^2 (k; c, \alpha, \delta, \theta),
\]

(3.26)

where the square arises from there being two iterations within the relationship. The evolution of \( \hat{e}_2 \) reveals the same contraction factor, that is \( \frac{\gamma_{1,+}^{(n+1)}}{\gamma_{1,+}^{(n-1)}} = \frac{\gamma_{2,-}^{(n+1)}}{\gamma_{2,-}^{(n-1)}} \) holds. For the method to be convergent the contraction factor must be strictly smaller than 1 for all \( k \), since this assures that every mode in the error decays as the scheme is iterated forward.

Finally, we extract a factor of

\[
\rho_{OR} (k; c, \alpha, \delta, \theta) = \left. \frac{\alpha - ik \sin \theta - \sqrt{c + k^2} \cos \theta}{\alpha - ik \sin \theta + \sqrt{c + k^2} \cos \theta} \right| e^{-\delta \sqrt{c + k^2}},
\]

(3.27)

for the oblique Robin conditions, where the definition of \( \lambda \) has been reinserted. When \( \theta = 0 \) is taken, this form reduces precisely to the one given in [6] for Robin conditions. There are a few interesting observations to be made:

- Increasing the overlap damps the high frequency error modes very effectively
- Increasing the model parameter \( c \) leads to faster decay of the lower modes
• For \( \theta = 0 \) there is always a mode satisfying \( \alpha = \sqrt{c + k^2} \) which contracts immediately

• As \( \theta \to \pi/2 \) Dirichlet conditions are recovered

• For \( \theta \neq 0 \) the contraction factor is never zero for finite \( k \)

Figure 3.3 shows the behavior of the contraction factor through various parameter sweeps and compares it against the Dirichlet and pure Robin cases.

### 3.2 Restricted additive Schwarz and generalizations

Discretizing the prior scheme, equations (3.11) and (3.12), yields the restricted additive Schwarz (RAS) family of solvers [7, 5, 23]. The (sub)domains \( \Omega, \tilde{\Omega}_j, \) and \( \Omega_j \) are replaced by \( \Sigma, \tilde{\Sigma}_j \) and \( \Sigma_j \) upon discretization. As before, the disjoint partitions satisfy \( \Sigma = \bigcup \tilde{\Sigma}_j \) and \( \tilde{\Sigma}_j \cap \tilde{\Sigma}_k = \emptyset \) for \( j \neq k \). Additionally, \( \tilde{\Sigma}_j \subset \Sigma_j \).

The partition of unity and extension operators used previously now correspond to the restriction and extension operators

\[
\begin{align*}
\tilde{R}_j : \Sigma &\rightarrow \tilde{\Sigma}_j \\
\tilde{R}_j^T : \tilde{\Sigma}_j &\rightarrow \Sigma
\end{align*}
\]

(3.28)

\[
\begin{align*}
R_j : \Sigma &\rightarrow \Sigma_j \\
R_j^T : \Sigma_j &\rightarrow \Sigma
\end{align*}
\]

(3.29)

where the restrictions truncate solutions to the domain of interest and extensions pad vectors by zeros to match the full set of degrees of freedom. The relation given by (3.9) has a discrete counterpart given by

\[
w = \sum_j \tilde{R}_j^T \tilde{R}_j w ,
\]

(3.30)

for vectors supported on \( \Sigma \). More compactly, this asserts that \( \sum_j \tilde{R}_j^T \tilde{R}_j = I \) over \( \Sigma \) (cf. [6] for discrete partitions of unity in the non-RAS case).

Now suppose that equation (3.1) is discretized over \( \Sigma \) to give \( Au = f \). The discrete subproblems in (3.11) become

\[
\begin{align*}
\begin{cases}
A_j v_j^{(n)} = R_j \left( f - Au^{(n)} \right) \\
u^{(n+1)} = u^{(n)} + \sum_j \tilde{R}_j^T \tilde{v}_j^{(n)}
\end{cases},
\end{align*}
\]

(3.31)

where \( A_j \) is the discretization of the differential operator on \( \Omega_j \) including the transmission and boundary conditions. The local correction supported on \( \Sigma_j \) is \( v_j^{(n)} \), while \( \tilde{v}_j^{(n)} := \tilde{R}_j \tilde{R}_j^T v_j^{(n)} \) contains its contribution limited to the \( j \)th disjoint partition. The approximation to the global solution at the \( n \)th iteration is denoted \( u^{(n)} \), which is updated through the disjoint corrections extended by \( \tilde{R}_j^T \). These linear problems may be solved in parallel with the only dependency between them now held in the global matrix-vector multiply needed to
Figure 3.3: The contraction factor given by equation (3.27) is plotted with various parameter values. In the top three plots the solid lines correspond to the pure Robin case, dashed lines correspond to the oblique case with $\theta = \pi/12$ and dotted lines give the Dirichlet behavior. In the first plot, the model parameter $c$ is varied by two orders of magnitude showing an appreciable difference in the convergence rate of the lowest modes under the Robin type conditions. The second plot varies the Robin weight $\alpha$, showing how the the performance may be improved by tuning this parameter. The third plot varies the overlap size with no overlap in red and a large overlap in blue. Without overlap, the Robin conditions yield contractive schemes with reduced performance on the higher modes while the Dirichlet conditions fail. With large overlap (blue) the convergence rates are fast for all modes. Finally, in the last panel the angle $\theta$ is swept from 0 to $\pi/2$ transitioning from pure Robin on the bottom to Dirichlet above.
form the residual. Inverting the above systems and inserting them into the update formula yields

\[ v_j^{(n)} = A_j^{-1}R_j \left(f - A\left[u^{(n)}\right] \right) \]  
\[ \sum_j R_j^T v_j^{(n)} = \left( \sum_j \tilde{R}_j^T \tilde{R}_j R_j^T A_j^{-1} R_j \right) \left(f - A\left[u^{(n)}\right] \right), \]  

where the restrictions are all taken with respect to the overlapped degrees of freedom but the subsequent extension to the global solution vector is done with respect to the disjoint degrees of freedom.

### 3.2.1 Use as a preconditioner

In the case where Dirichlet transmission conditions are imposed in (3.11) the local discrete operators can be obtained via restrictions and extensions of the global matrix by \( A_j = R_j A R_j^T \) since these conditions simply impose zeros outside of \( \Sigma_j \) anyway. In this case the solver is termed *Restricted Additive Schwarz* (RAS) [6, 5], and the preconditioner matrix defined in (3.33) is denoted

\[ M_{RAS}^{-1} = \sum_j \tilde{R}_j^T \tilde{R}_j R_j^T R_j A_j^{-1} R_j. \]  

For the Robin transmission conditions, defined in (3.4), the local operators \( A_j \) are not obtained by simple restrictions of the global matrix. The preconditioner matrix can still be formed

\[ M_{ORAS}^{-1} = \sum_j \tilde{R}_j^T \tilde{R}_j R_j^T A_j^{-1} R_j, \]  

and the scheme is referred to as *Optimized Restricted Additive Schwarz* (ORAS) [23]. The weight \( \alpha \) present in the Robin conditions may be chosen to optimize the convergence rate. In either case, the net effect of this iteration is to create a sequence of solution estimates via

\[ u^{(n+1)} = u^{(n)} + M^{-1} \left(f - A u^{(n)} \right), \]  

from some initial guess \( u^{(0)} \). However, by linearity of the problem, the exact solution could be subtracted from both sides of this to obtain

\[ u^{(n+1)} - u = u^{(n)} - u + M^{-1} \left(f - A u^{(n)} \right), \]  
\[ e^{(n+1)} = e^{(n)} + M^{-1} \left(f - A \left(u + e^{(n)}\right) \right) = \left(I - M^{-1} A \right) e^{(n)}, \]
giving the error propagation matrix \((I - M^{-1}A)\). The spectral radius of this matrix being less than 1 is necessary and sufficient for the iteration to converge \([6, 16]\). Additionally, the solution iteration in (3.36) and error propagation in (3.39) can be seen to be the simple Richardson iteration relaxation scheme \([26]\) applied to the system

\[
M^{-1}Au = M^{-1}f .
\]  

(3.40)

For this reason, the matrix \(M^{-1}\), or more accurately the collective action of solves in (3.33), is called a preconditioner for \(A\). Furthermore, since Krylov solvers need only apply matrix-vector products, the modified system in (3.40) can be treated by simply applying the action of \(M^{-1}\) to the result of all matrix-vector products involved.

### 3.3 Example: Poisson equation in the plane

As a simple first example we consider Poisson’s equation on the unit square \(\Omega = (0,1)^2\)

\[
\begin{align*}
\Delta u &= e^{xy} \sin(2\pi x) \sin(2\pi y) & (x,y) \in \Omega \\
u &= 0 & (x,y) \in \partial \Omega,
\end{align*}
\]

(3.41)

with homogeneous Dirichlet conditions posed globally and a simple, but non-trivial forcing in the interior. The global problem is discretized by the standard second-order accurate five-point centered difference Laplacian

\[
\Delta u_{i,j} \approx \frac{1}{\Delta x^2} (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) ,
\]

(3.42)

which gives a discrete equation \(Au = f\). We treat this equation by the domain decomposition methods described above. This problem is posed over two different grids, a coarse one with \(64^2\) unknowns, and a finer one with \(256^2\) unknowns.

The disjoint subdomains, \(\Sigma_j\), are formed by dividing these grids into 6 equal sized strips in the \(y\)-direction and a varied number of strips in the \(x\)-direction to achieve different total subdomain counts \(N_S\). The overlapping subdomains, \(\Sigma_j\), are then formed by adding \(NO\) layers of grid points around each disjoint subdomain making the overall overlap width \(2NO\Delta x\). The residual framework introduced in equation (3.31) is used here so the local problems use restrictions of the global residual as forcing functions. The Robin boundary conditions, \(\frac{\partial v}{\partial n} + \alpha v = 0\), are formed using centered differences along the boundaries. A ghost node lying outside the subdomain is used whose value is determined through the use of the Laplacian difference operator at the boundary \([9]\). At \(x = 1\) for instance

\[
(4 + 2\alpha \Delta x) v_{M,j} - 2v_{M-1,j} - v_{M,j+1} - v_{M,j-1} = -r_{M,j} ,
\]

(3.43)
Figure 3.4: The effect of subdomain count on these methods is shown here with the solvers in the left panel, and preconditioned GMRES in the right panel. Solid lines denote the low resolution case with $64^2$ unknowns while the dashed lines show the higher resolution case with $256^2$ unknowns. For both resolutions, the circles correspond to the RAS methods and the triangles correspond to the ORAS methods with a Robin weight of $\alpha = 10$. As would be expected the iterations required to converge increase as the number of subdomains increases. The high resolution case is more affected than the low resolution case and similarly the RAS methods are more sensitive than the ORAS methods.

is enforced as the equation for $v_{M,j}$. $M$ denotes the final grid point, taking the value $M = 64$ or $M = 256$ herein. Finally, at the corners of the subdomain the normal vectors are taken to be diagonal and two differences with two ghost points are used.

The treatment of the global problem by the (O)RAS solver is summarized in Algorithm 1. Therein the local operators are built and factored, after which equation (3.36) successively applied. Convergence is declared when the relative norm of the residual, 

$$\frac{\|r^{(n)}\|_2}{\|r^{(0)}\|_2},$$

decays to some predetermined tolerance. Here, a tolerance of $10^{-6}$ is used, though the results are not significantly impacted by tightening it.

Figure 3.4 shows that the required number of iterations increases as the number of subdomains is increased. This is expected from the global nature of the Poisson equation and the delay in iterations for separated subproblems to influence each other [16, 6]. Interestingly, the ORAS methods using the Robin transmission conditions are only weakly affected. Additionally, the preconditioned GMRES results are not quite monotone with the $N_S = 36$ case behaving unusually well. As the overlap width is increased in Figure 3.5 the RAS methods see diminished iterations to convergence at the cost of more expensive local solves. The ORAS methods operate nearly independently of the overlap width. Finally, the effect of the Robin weight is measured not in terms of iteration counts but rather the ratio of the RAS iterations to the ORAS iterations, as shown in Figure 3.6. This allows the effect of the Robin weight to be viewed as the advantage that ORAS provides over RAS as the solution cost is the same when $N_S$ and $N_O$ are kept fixed.
Figure 3.5: The effect of the overlap width on these methods is shown here with the same pattern as in Figure 3.4. The solvers (left panel) and the preconditioners (right panel) are similarly effected in all cases. As before, the high resolution cases (dashed lines) are effected more strongly than the low resolution cases. The ORAS cases seem nearly independent of the overlap width as may be expected from the convergence behavior from Section 3.1.1.

**Algorithm 1 (O)RAS Solver and preconditioner**

```plaintext
for all $\Sigma_j$ do
    Pre-factor $A_j = L_j U_j$
end for

$u^{(0)} = 0$

$r^{(0)} = f$

$n \leftarrow 0$

while $||r^{(n)}||/||r^{(0)}|| > tol$ do
    $v^{(n)} \leftarrow 0$
    for all $\Sigma_j$ do
        Solve $A_j v_j = R_j r^{(n)}$
        $v^{(n)} \leftarrow v^{(n)} + R_j^T v_j$
    end for
    $u^{(n+1)} \leftarrow u^{(n)} + v^{(n)}$
    $r^{(n+1)} \leftarrow f - Au^{(n+1)}$
end while
```

21
The ORAS solvers provide the greatest speed up when the weight is kept small, but destabilize as they tend towards the singular Neumann case. Indeed, with $\alpha = 5$ and $256^2$ unknowns the ORAS solvers diverge after a few iterations, seen as missing data points in the left panel of Figure 3.6. The improvement available to the preconditioned GMRES methods is smaller but still significant, particularly for the high resolution case. This same style of presentation is used in Chapter 6.

Figure 3.6: The Robin weight $\alpha$ is varied and the effects on the convergence behavior for the solvers (left panel) and preconditioners (right panel) are plotted. Instead of raw iteration counts the ratio of RAS iterations to ORAS iterations is plotted to capture the relative effect of introducing the improved transmission conditions. The low resolution cases, $64^2$ unknowns, are shown as solid lines while the high resolution cases, $256^2$ unknowns, are shown as dot-dashed lines. The large overlap cases show diminished factors of improvement since the RAS methods behave better at these overlaps while the ORAS methods are nearly independent of overlap width (c.f. Figure 3.5). There is no significant difference between the 24 and 36 subdomain cases.
Chapter 4

Subdomain Construction

To treat the relevant discrete form for the model problem given in (2.11) by domain decomposition methods, we need to first find a partitioning of the space. To keep the scheme general, the strategy for partitioning must be able to split arbitrary surfaces into any number of subdomains. To achieve this, the method cannot rely on information specific to the surface beyond what the closest point method already needs. Ideally, the generated subdomains would respect the surface normals to simplify the description of the local problems. Two heuristic approaches are given for this, though the transmission conditions that we propose in Chapter 5 and use throughout this work are robust to misalignment with the surface normals.

To obtain this desired generality we do not partition the surface but rather the set of active nodes. Within the domain decomposition community, it is popular to find partitions by building a graph over the mesh and using a graph partitioner such as METIS [8] or SCOTCH [15]. Here, we follow this idea and extend it to our case where the preservation of the underlying geometry is of interest. Several possible connectivity schemes are commented on, and ties are made back to some other preconditioning strategies.

4.1 Graph partitioning problem

The graph partitioning problem is closely related to the subdomain generation problem for domain decomposition schemes. Many discretization schemes introduce some mesh over the domain, over which a graph can be constructed with edges connecting the constituents. Finite difference schemes would generally take the connections of this graph as those induced by the stencil of the discrete operator. Finite element and finite volume schemes are partitioned element by element (or cell by cell) with the graph connections running between those elements sharing a face or edge.

The ideal partitioning would create the desired number of local meshes with the minimal number of edge cuts. This minimizes the number of nodes in the overlap region, to be constructed in Section 4.2, for a given overlap width. This keeps size of the local problems
The two graph connectivities discussed in Sections 4.1.1 and 4.1.2 can be seen here with the simple centered difference scheme on top and the full stencil approach below. In each, all of the connections for the node indicated by the solid diamond are drawn, though for the construction of the entire graph all connections for each node would need to be added.

Additionally, for an implementation built with distributed memory parallelism this will reduce the message sizes for overlap regions that need to be populated from another process. The partitioning software introduced in [8] is used to generate all partitionings, with an objective function satisfying precisely these desires. The METIS documentation defines additional partitioning objectives and discusses their use in more detail. This choice mirrors much of the domain decomposition community at large [6].

The discretization of the iCPM relies on two stencils, one for the centered difference Laplacian on the embedding space, and another for the interpolation within the extension operator (2.10). These stencils can be used to produce different graphs that capture different features of the problem. Two contrasting possibilities are highlighted here. The simplest considers only the finite difference operator and requires minimal input while the second mimics the behavior of some algebraic multigrid black box preconditioners [28]. Figure 4.1 illustrates these two schemes with the simple nearest neighbor coupling of Section 4.1.1 shown above and the full stencil coupling of Section 4.1.2 below.

To guide the discussion in the remainder of this chapter it will be useful to define several node sets and types of nodes:

- **Disjoint partition** $\tilde{\Sigma}_j$: The same node set as in Section 3.2, obtained here from the graph partitioner or by exploiting some underlying symmetry.

- **Overlapping partition** $\Sigma_j$: The disjoint partition is grown outwards to build overlapping subdomains.
• **Boundary nodes** $\Sigma_{BC}^j$: These nodes are necessary for the imposition of the boundary conditions on the local problem. They are active nodes but are not contained in $\Sigma_j$.

• **Local ghost nodes** $\Sigma^G_j$: The nodes layered on $\Sigma_j$ to complete the stencil of the ambient Laplacian and the boundary operator, layered on $\Sigma_{BC}^j$, if needed. Those around $\Sigma_j$ are also ghosts in the global problem but those surrounding $\Sigma_{BC}^j$ may not be.

• **Neighboring nodes**: The nodes adjacent to a given node along the principal axes of the mesh that lie within the computational tube or local mesh.

• **Interfacial node**: A node in $\tilde{\Sigma}_j$ or $\Sigma_j$ whose local list of neighbors does not correspond with its global list of neighbors.

The node sets $\tilde{\Sigma}_j$, $\Sigma_j$, and $\Sigma_{BC}^j$ are all active and share the following properties, $\tilde{\Sigma}_j \subset \Sigma_j \subset \mathcal{N}_A$, $\Sigma_{BC}^j \subset \mathcal{N}_A$, and importantly $\Sigma_j \cap \Sigma_{BC}^j = \emptyset$. The creation of these sets is treated each in turn in the following subsections.

### 4.1.1 Nearest neighbor coupling

Using the stencil of the centered difference Laplacian yields a sparse graph of maximal degree $2d$, twice the dimension of the embedding space, which is simple to create and partition, but in turn ignores nearly all of the underlying geometry. The generated partitions generally will not follow the surface normals, with the utility of the partitions coming only from the narrowness of the mesh and the relatively small number of edge cuts induced by partitions aligned to the surface normals. This graph connectivity can be seen in the upper portion of Figure 4.1 with an indicative interface between partitions visible in the second panel of Figure 4.2.

### 4.1.2 Full coupling

Another option is to augment the graph obtained from the nearest neighbor scheme by now including the entire interpolation stencil of each node in its connected vertices. This gives a much denser graph, whose maximal degree now depends on the interpolation order $2d + (p + 1)d$, but includes a great deal of additional information. An exemplary set of connections can be seen in the lower portion of Figure 4.1 with the interface between partitions visible in the third panel of Figure 4.2. In fact, from Figure 4.2 we see that the non-locality present in the problem has produced a partition that is disconnected in the embedding space which does not readily serve our purposes. This can be mitigated by the addition of overlaps as in Section 4.2 but shall not be pursued any further. Though this fails to be a useful strategy for partitioning it does hint at the poor performance of some algebraic multigrid black box preconditioners for this problem. The methods generally rely on building a graph based on the nonzero pattern of the matrix and subsequently pruning the induced adjacency matrix [28].
Figure 4.2: Several interfaces between disjoint partitions are gathered here for a closest point mesh over a circular surface. The first panel shows the ideal situation as a manually obtained partition exploiting the circular symmetry. The second panel shows the result of the nearest neighbor graph partitioning, which fails to follow the surface normals but remains connected. The third panel uses the full coupling scheme to define its graph which gives rise to a disconnected partition, though it does otherwise follow the surface normal very well. Finally, the last panel gives the result of applying the interface corrector defined in Section 4.1.3 to the second panel and now respects the surface normal.

Figure 4.3: A typical disjoint partitioning from the nearest neighbor coupling scheme is shown in the first panel with diamonds denoting one subdomain and circles the other. From here all interface nodes have the closest points generated, shown as solid black lines in the middle panel, and if the lattice point closest to this point on the surface belongs to a different subdomain they are marked for migration. The third panel shows a surface normal near the interface to illustrate the quality of the reassignment.

4.1.3 Interface alignment

The partitions visible in the second panel of Figure 4.2 were generated by the nearest neighbor coupling scheme and are used as the foundation for the rest of this thesis. Notably, these do not follow the surface normals. The formulation of the overlaps and construction of boundary information as described in the remainder of this chapter, as well as the transmission conditions posed in Chapter 5, are robust enough to deal with this misalignment. Regardless, we now describe a simple scheme for the alignment of these interfaces.

The METIS \cite{Karypis} partitioning obtained from the nearest neighbor scheme defined above can be aligned to the surface normals by visiting each node and querying its closest point. Projecting this point on the surface to the nearest lattice point identifies this node with the one most closely aligned to it in the surface normal direction. If this node’s assigned partition differs from this projected closest point then it is marked for reassignment. After
marking all nodes, they may be moved from their current subdomain to the subdomain of their projected point, then all subdomains are re-indexed to keep them associated with contiguous sets of unknowns. Figure 4.3 illustrates this process for an indicative interface from the nearest neighbor scheme defined in Section 4.1.1. In the last panel of Figure 4.2 the action of this process on a real partitioning can be seen.

4.2 Construction of overlaps

After the disjoint partitions $\tilde{\Sigma}_j$ are found from METIS, a set of overlapped partitions and all associated interface information needs to be created. A simple and effective method for the overlap construction visits all interfacial nodes in each partition and adds their neighbors into the new overlapped version, $\Sigma_j$. This process is then repeated a suitable number of times over the new interface of $\Sigma_j$ to achieve larger overlap widths. If $n$ passes of this process are applied to all subdomains, then an overlap width on the order of $2n\Delta x$ will be achieved. From these overlapping partitions two pieces of subdomain information need to be extracted before the transmission operators proposed in Chapter 5 can be constructed. The boundary of the subsurface induced by this collection of nodes must be found, and because of the indirect route to obtain this partitioning, will need to be approximated. Further, if Robin conditions are to be used then the boundary conormals will be needed. These are unit vectors anchored at each boundary location that simultaneously lie tangential to the surface and normal to the constructed subsurface boundary. The nodes in each of these sets are illustrated Figure 4.4.

4.2.1 Construction of effective boundaries

After the overlapping partitions $\Sigma_j$ are found, boundary points on the surface must be constructed to identify this collection of nodes with a patch of the surface. Additionally, there needs to be enough nodes present to actually impose the boundary conditions. First, all of the interfacial nodes in each $\Sigma_j$ are visited and their closest points collected into the set $X_j^{int}$ as a tentative effective boundary. The interpolation stencil of each $x_i \in X_j^{int}$ is found and any nodes missing from $\Sigma_j$ are placed in the set $\Sigma_j^{BC}$. Any incomplete stencils for the centered difference Laplacian are now completed by adding those points to $\Sigma_j^{BC}$ as well. The active nodes for the definition of the local problem is the union $\Sigma_j \cup \Sigma_j^{BC}$ but the boundary nodes are kept separate to aid in the imposition of transmission conditions.

To complete the construction of the mesh for the local problem, ghost nodes must be added as in the global case. For RAS solvers with Dirichlet transmission conditions, ghost nodes are only needed around $\Sigma_j$, to complete the stencil of the ambient Laplacian and agree with the ghost nodes defined on the global problem. The ORAS solvers using Robin transmission conditions require additional ghost nodes layered over the boundary nodes $\Sigma_j^{BC}$ which are not ghost nodes in the global problem. Either way, the set of ghost nodes
Figure 4.4: Here, the node sets used in the construction of one subproblem are built up successively. First the disjoint nodes $\tilde{\Sigma}_j$ that arise from the graph partitioning stage are identified in blue. Four layers of overlap nodes are added to create the grown partition $\Sigma_j$, which includes all of the blue nodes as well as those shown in green. From the interfacial nodes of $\Sigma_j$ the effective boundary locations are generated, and visible as the black diamonds. The nodes from incomplete stencils are gathered to form $\Sigma^{BC}_j$ seen in yellow. Finally, the ghost node set $\Sigma^G_j$ is formed as a layer around all nodes, shown in red where those bordering $\Sigma_j$ also lie in the set of global ghost nodes while those bordering $\Sigma^{BC}_j$ are active nodes in the global classification. The boundary conormal vectors as constructed in Section 4.2.2 are shown in grey.

is identified as $\Sigma^G_j$. Finally, these three node sets, the overlapped active nodes in $\Sigma_j$, the boundary (transmission) nodes in $\Sigma^{BC}_j$, and the ghost nodes in $\Sigma^G_j$, specify all members of the local discrete problem.

The boundary construction is finalized by associating all nodes in $\Sigma^{BC}_j$ and members of $\Sigma^G_j$ with points on the effective boundary $X^{\text{int}}_j$. This association is made by a method similar to that proposed in [12] for handling point clouds. Each effective boundary point $x_i \in X^{\text{int}}_j$ is considered and all nodes within a distance of $\frac{p+2}{p+1}\gamma$ are visited. If the visited node is unassociated, or closer to this boundary point, it is associated with $x_i$. This reduces the cost of constructing the boundary considerably over the naive approach of visiting each node in $\Sigma^{BC}_j$ and searching for the nearest boundary point in $X^{\text{int}}_j$ without introducing much additional complication. With this identification made, the local problem now has its own closest point function

$$ CP_{S_j}(y_i) = \begin{cases} CP_S(y_i), & y_i \in \Sigma_j \\ CP_S(y_i), & y_i \in \Sigma^G_j \text{ and globally a ghost} \\ \arg \min_{x \in X^{\text{int}}_j} |y_i - x|, & \text{otherwise} \end{cases} $$

(4.1)
This agrees with the global closest point function over $\Sigma_j$, but points to the appropriate member of $X_j^{int}$ for nodes in $\Sigma_j^{BC}$.

### 4.2.2 Boundary conormal directions

The imposition of Robin conditions in Section 5.3 will need information about the boundary conormal direction, the direction tangential to the surface and normal to the boundary. Much like the boundary locations in the prior section these will need to be approximated. Next, tentative conormal vectors are constructed in the simplest way possible and subsequently smoothed to yield useful and consistent vectors for the construction of the Robin boundary operators in Section 5.3.

**Tentative conormal vectors**

Take $x_i \in \Sigma_j^{BC}$ (or the appropriate part of $\Sigma_j^G$) as a boundary node and define $d_i = x_i - CP_{S_j}(x_i)$ as the vector pointing from the nearest boundary location to this particular boundary node. A naive approach to define the conormal directions would simply project the surface normal component out of $d_i$ to obtain

$$q_i = \frac{d_i - (d_i \cdot n_i)n_i}{|d_i - (d_i \cdot n_i)n_i|},$$

(4.2)

as an approximation to the conormal anchored at $CP_{S_j}(x_i)$. These vectors are quite sensitive to the layout of the boundary nodes and the shape of the local subdomain, giving wildly varying conormal directions. The left panel of Figure 4.5 shows these tentative conormals for a typical subdomain obtained on a sphere. To be useful for the transmission conditions in the following chapter they will need to be smoothed to capture the shape of the subsurface boundary more appropriately.

**Least squares smoothing**

A fairly robust means of smoothing these conormal directions for two dimensional surfaces embedded in three dimensions is by representing the component directions of the conormal vectors by a polynomial over the embedding space. Low degree multivariate polynomials are capable of resolving the subsurface boundary and can be found by solving a least squares problem over the effective boundary locations. The tensor product monomials $x^\eta y^\mu z^\nu$, $\eta, \mu, \nu \leq p$ for degrees $p = 2, 3$ are considered and their evaluations over all of the effective boundary points are collected into a Vandermonde matrix $V$. For three dimensional problems there are many more boundary points than the dimension of this polynomial space and thus the matrix $V$ has many more rows than columns. The coefficients of this polynomial expansion are found by solving, in the least squares sense, the overdetermined system

$$V a^{(n)} = c^{(n)},$$

(4.3)
Figure 4.5: Here one partition from a mesh over a unit sphere is shown. The left panel shows the conormals constructed through the naive scheme, which cross each other a great deal. In the right panel, 5 passes of the smoother have been applied yielding noticeably cleaner conormal vectors.

where $c^{(n)}$ holds the $n^{th}$ component of the tentative conormal vectors respecting the ordering of the Vandermonde matrix. The total number of boundary points is considerably smaller than the number of active points defining the local problem and the least squares solution needed here adds negligible time and memory requirements to the initialization of the solver.

From the solutions of these systems the $n^{th}$ component of the smoothed conormal vectors are found by evaluating

$$\tilde{q}_n(x, y, z) = \sum_{\eta, \mu, \nu \leq p} a_{\eta \mu \nu}^{(n)} x^\eta y^\mu z^\nu,$$

at $CP_{S_j}(x_i)$ for each boundary node and component $n$. The new conormal vectors are generally not tangential to the surface or of unit magnitude so they must be re-projected according to equation (4.2). These vectors are considerably more regular in their orientation and capture the boundary behavior much better than the tentative ones do. This procedure can be repeated several times to continue relaxing the conormal vectors for improved regularity.

**Neighborhood smoothing**

The least squares approach does not work well for one dimensional surfaces embedded in two dimensions. The number of boundary points is quite small and the subsurface boundaries are (nearly) zero dimensional making the rows of the Vandermonde matrix nearly identical. Instead, a simpler smoothing operation can be done by locally averaging the tentative conormal vectors with each other.

The neighborhoods considered here are collections of boundary nodes whose effective surface boundary locations are near to each other. Each boundary node $x_i \in \Sigma_j^G$ can be associated not only with its effective closest point but also the lattice point closest to that point. All indices of boundary nodes sharing their closest lattice point with $CP_{S_j}(x_i)$ can
be collected in the set $\mathcal{L}_i \subset \Sigma_j^{BC} \cup \Sigma_j^G$ so that the weighted average can be written as

$$\tilde{q}_i = q_k + \frac{1}{2} \sum_{k \in \mathcal{L}_i} q_k,$$

(4.5)

to form a smoothed representative vector. The factor of $1/2$ weights the original vector over its neighbors, but is arbitrary in value. This vector will in general not lie tangential to the surface, as in the least squares case above, and will need to be re-projected and normalized as in equation (4.2). It is for this reason that the average in equation (4.5) is not properly normalized to begin with.
Chapter 5

Transmission conditions

With the global mesh successfully partitioned into disjoint and overlapping subdomains, \( \Sigma_j \) and \( \Sigma_j \) respectively, the local operators consistent with equation (2.11) can be defined. The interfaces introduced by this splitting are completely artificial and their treatment will affect the performance of the schemes greatly. Critically, the boundary conditions posed on these interfaces to complete the local problems need to be posed in a way that is consistent with the global problem. First the conditions for this algebraic consistency are discussed and related back to conditions on the stencil of the boundary operator. Following this, Dirichlet and Robin conditions are considered along with some observations on their required accuracy.

In the closest point method the application of boundary conditions is done through a modification of the extension operator \[18, 11\]

\[
A_j = \left( c + \frac{2d}{h^2} \right) I - \left( \frac{2d}{h^2} + \Delta^h \right) \begin{pmatrix} E \\ T_j \end{pmatrix},
\]

over the boundary nodes \( \Sigma_j^{BC} \). These extensions are found by discretizing the boundary operator and isolating each individual node’s role within it, which is simplified by only needing to consider homogeneous boundary conditions. With this modified extension constructed and placed into \( T_j \), the ambient Laplacian may be applied to the whole of the local problem. The second index in the continuous transmission operators, \( T_{jk} \) in equation (3.2), is suppressed as this block contains all discrete operators belonging to the \( j^{th} \) subproblem. This process can be seen as a generalization of the standard method for imposing boundary conditions with derivative information on finite difference problems, as was seen in equation (3.43). This modification will be made concrete for Dirichlet conditions in Section 5.2, and Robin conditions in Section 5.3, but first the critical conditions for algebraic consistency are considered.
5.1 Overlap requirements

In the continuous setting, there are few restrictions on the splitting and the boundary conditions posed on the subdomains; in fact the homogenization of the boundary conditions in (3.11) relied only on an overlap being present. Upon discretization however, the consistency of the preconditioner defined in (3.35) with the global system (2.11) is not trivial. In [23] these consistency requirements were derived in a fairly general setting for both multiplicative and additive optimized Schwarz methods. Consider first the generic local system of (3.2) and its direct discretization

\[
\tilde{A}_j u_j^{(n+1)} = f_j + \sum_{k \neq j} T_{jk} u_k^{(n)},
\]

(5.2)

where \(u_j\) is the local solution, \(f_j = R_j f\) is the local forcing, and the effect of the transmission operators \(T_{ij}\) has been moved to the right hand side. The splitting with respect to the restrictions \(R_j\) and modified local operators \(A_j\) is defined to be consistent [23] if the global exact solution satisfies (5.2). For any pair \(u\) and \(f\) solving the global system the expression

\[
\tilde{A}_j R_j u = R_j f + \sum_{k \neq j} T_{jk} R_k u,
\]

(5.3)

must also hold. Applying a restriction operator to the global system and inserting the consistency definition (5.3) connects the global system to the modified local operators

\[
R_j A u = R_j f
= \tilde{A}_j R_j u - \sum_{k \neq j} T_{jk} R_k u.
\]

(5.4)

(5.5)

After rearranging, one finds

\[
0 = \left( \tilde{A}_j R_j - \sum_{k \neq j} T_{jk} R_k - R_j A \right) u
\]

(5.6)

\[
R_j A = \tilde{A}_j R_j - \sum_{k \neq j} T_{jk} R_k
\]

(5.7)

where the final line follows as a consequence of the solution \(u\) being arbitrary.

5.2 Dirichlet Transmission

Dirichlet boundary conditions make for the simplest transmission conditions possible. In the initial formulation of the closest point method, first-order accurate conditions were considered by simply propagating the desired value out to all of the boundary nodes. In our problem, the conditions are homogeneous and only zeros need to be pushed out. As
such, the transmission block of the extension operator is simply a zero matrix adjoined to an identity matrix

\[
T_j = \begin{pmatrix} 0_{[\Sigma j^{BC} \times |\Sigma j|}} & I_{[\Sigma j^{BC} \times [\Sigma j^{BC}]} \end{pmatrix}.
\]  

(5.8)

The local right hand side, \( f_j \), has its entries over the active boundary nodes overwritten by zeros. Notably, this formulation is identical to what one would obtain by constructing the preconditioner in a purely algebraic fashion, since zeros are written out and their contributions vanish. The benefit in building the local operators this way lies in paving the way towards the later transmission operators.

Using the effective boundary locations defined in Section 4.2.1, one could also impose second-order accurate Dirichlet conditions. Take \( x_i \in \Sigma_j^{BC} \) as a boundary node and \( CP_{\Sigma_j} (x_i) \) as its effective boundary location. A homogeneous Dirichlet boundary condition can be imposed by identifying this boundary node with its mirror point through the boundary instead of its closest point. Extending this against the mirror point \( y_i = (2CP_{\Sigma_j} (x_i) - x_i) \) we have that

\[
u(x_i) + u(CP_{\Sigma_j} (y_i)) = 0,
\]

(5.9)

enforces that the solution over the boundary nodes is equal and opposite that of their mirror points. Passing from the active nodes to the ghost nodes, the solution will pass through zero to satisfy the condition (to second-order accuracy) [11]. Each row of the transmission block within the extension operator is then populated by this condition, where the second term is expanded as the interpolation stencil of the mirror point \( y_i \).

In the discretization of the second-order method one sees that solution values inside the subdomain, those participating in the interpolation stencil of the mirror point, are effected in a way that is inconsistent with the global system. Indeed, if the interpolation stencil of \( y_i \) contains any of the points from the disjoint partition \( \tilde{\Sigma}_j \) underlying this local problem on \( \Sigma_j \) then their updates will violate the overlap condition. The first-order Dirichlet conditions defined prior do not suffer this deficiency as they do not rely on any interior information, and will satisfy the requirement on the overlap automatically. It is for this reason that primarily the first-order Dirichlet conditions are considered herein.

### 5.3 First-order accurate Robin

Robin transmission conditions can dramatically increase the convergence rate of Schwarz solvers [23, 22, 24, ?] by passing derivative information between the subdomains. For the surface problems considered here the Robin boundary condition over the \( j^{th} \) subdomain becomes

\[
\left( \frac{\partial}{\partial q} + \alpha \right) u = 0 \quad \text{on } \partial S_j.
\]

(5.10)
As in Section 4.2.2, \( q \) is the boundary conormal lying normal to the subsurface boundary \( \partial S_j \) but tangential to the surface. To cast this into a form more easily discretized within the closest point method it is beneficial to consider a boundary node \( x_i \in \Sigma^j_{BC} \). Taking this as simply a point of interest in the embedding space the discussion can remain in the continuous setting. Take \( q_i \) as the conormal direction at the closest boundary point \( CP_{S_j}(x_i) \) and recall the vector \( d_i = x_i - CP_{S_j}(x_i) \) from above. The derivative in the conormal direction may be recast to one in the direction \( d_i \)

\[
\frac{\partial u}{\partial d_i} = \frac{\partial u}{\partial q_i} \cos \theta_i + \frac{\partial u}{\partial q_i} \cos \theta_i \cos \theta_i = \frac{\partial u}{\partial q_i} \cos \theta_i ,
\]

where \( \theta_i \) is the angle between the conormal \( q_i \) and the node’s displacement \( d_i \). The derivative in the normal direction vanishes since the solution is constant in that direction. From this identification, the discretization of the condition in (5.10) becomes quite accessible. As we saw in the previous section the order of accuracy of the boundary condition is not so critical, and instead a condition with a minimal stencil size should be sought. Replacing the derivative with a forward difference and the Dirichlet component with an identity extension yields

\[
\left( \frac{\partial}{\partial q} + \alpha \right) u \approx \frac{u(x_i) - u(CP_{S_j}(x_i))}{d_i \cdot q_i} + \alpha u(x_i) ,
\]

at each boundary point. This discretization induces the modified extension

\[
u(x_i) - \left( \frac{1}{1 + \alpha d_i \cdot q_i} \right) u(CP_{S_j}(x_i)) = 0 ,
\]

such that each row of the transmission block, \( T_j \), contains a diagonal term and the interpolation stencil for the boundary node. We discretize condition (5.10) to first-order accuracy as will be shown momentarily.

Before verifying the truncation error in this approximation to the boundary operator, it is worth making a few observations. Due to the location of the boundary nodes the included angle between the displacement and the conormal will always be acute, and thus the dot product \( d_i \cdot q_i \geq 0 \) will always be positive. The construction of the approximate boundaries and conormals from above preserves this property. Furthermore, in the case of orthogonality between these, the extension recovers the typical active node extension.

### 5.3.1 Convergence study

To the knowledge of the author, the specific discretization of the Robin condition given in equation (5.14) has not appeared before. To validate the expectation that this enforces the
Robin boundary condition to first-order we solve

\[
\begin{cases}
    (1 - \Delta_S) u = 1 - \theta^2 & \text{on } S \\
    u(0) = 0 \\
    \left. \left( \frac{\partial}{\partial \theta} + 1 \right) u \right|_{\theta=2} = 0
\end{cases},
\]

where $S$ is an arc of the unit circle centered on the origin in $\mathbb{R}^2$ delimited by $0 < \theta < 2$ measured in radians. This has the exact solution $u_{ex}(\theta) = 2 \left( e^{-\theta-1} + e^{\theta-1} - e^{-\theta-3} \right) - \theta^2 - 1$. This choice of domain ensures that the place where the Robin condition occurs is not aligned with the grid, while keeping the comparison to an exact solution simple. The Dirichlet condition at $\theta = 0$ is enforced to second-order through equation (5.9) and the interior is treated by bi-cubic interpolation and second-order centered differences so that the experiment can focus on the Robin condition.

The grid spacing is successively halved in size from $\Delta x = 1/32$ to $\Delta x = 1/1024$ and the obtained solution is compared against the true solution extended exactly out to the active nodes in the embedding space. The errors measured in the infinity norm over this range of grids are shown in Figure 5.1. We observe that the slope of fitted line is 1.00638 which is consistent with first-order accuracy.

![Figure 5.1:](image-url)
Chapter 6

Results

The development of the (O)RAS methods throughout this thesis has relied on a generic partitioning scheme as given in Chapter 4, the development of appropriate transmission conditions in Chapter 5, and in the case of Robin conditions some key approximations to the subsurface boundary locations and conormal directions in Sections 4.2.1 and 4.2.2 respectively. In everything that follows, the choice of method for Krylov subspace acceleration is GMRES [19], since the systems arising from the closest point method are not symmetric and GMRES has been found to work well with Schwarz domain decomposition preconditioners [16, 6]. First, a parameter study will show experimentally the convergence behavior of these methods, and build some intuition towards the selection of parameters for larger or more complicated problems. Additionally, several key approximations have been made for the imposition of the Robin conditions and this will serve to validate the consistency of these approximations. Following this, a sequence of large problems will be solved with just a few realizations of the developed solver over different numbers of processors. The profiling information from this will then give a sense of the scalability of the method. All tests have been run on the TornGat compute cluster hosted at Memorial University of Newfoundland.

6.1 Introduced parameters

The full definition of the developed RAS and ORAS methods has seen the introduction of several parameters. In the first series of tests, the effects of these parameters are considered and compared to expectation. Table 6.1 recalls all of these parameters as well as their anticipated influence over the solver. Most of these parameters are fairly intuitive and selecting values for them is reasonably simple given the context of a target problem. As such, most will only be swept over limited ranges in the parameter studies of Section 6.2 with the Robin weight $\alpha$ from equation (5.10) being a notable exception. A great deal of effort has gone into determining these weights analytically with an emphasis on positive Helmholtz problems [7]. A surface PDE case has even been considered in [10]. However, the non-locality present in the closest point method, the unpredictable shape of the subdomains,
and the approximations made in constructing the transmission operators all indicate the need for an exhaustive search. After this, one weight that performed well is selected and sweeps are performed over the subdomain counts and overlap widths in isolation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expectation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model weight, $c$</td>
<td>Convergence will slow as $c$ decreases.</td>
</tr>
<tr>
<td>Overlap width, $N_O$</td>
<td>Larger overlaps require fewer iterations to converge. Preconditioner is more costly to apply.</td>
</tr>
<tr>
<td>Subdomain count, $N_S$</td>
<td>More subdomains will require more iterations but increase parallelism.</td>
</tr>
<tr>
<td>Transmission Condition</td>
<td>Robin conditions will converge faster when the overlap condition is met. Dirichlet conditions will be more robust (cf. Section 3.3).</td>
</tr>
<tr>
<td>Robin Weight, $\alpha$</td>
<td>An optimal value will exist but is unknown a priori.</td>
</tr>
</tbody>
</table>

Table 6.1: All relevant parameters are recalled here along with a brief description of their expected effects.

### 6.2 Domain decomposition parameter studies

The parameter studies compiled in this section will all follow the same format. For each surface, two grid resolutions will be considered with two overlap widths and three subdomain counts. Both Dirichlet and Robin conditions are considered with the Robin weight being swept over a wide range of values. Additionally, the subdomain counts and overlap widths are swept independently of all other parameter choices to verify their expected behaviors. Every parameter combination yields both an iterative solver as in equation (3.36) and a preconditioner for GMRES as in equation (3.40) with both formulations considered.

In all cases, the forcing is set to zero with the initial solution set as a random vector. In this way the solvers and preconditioners are directly evaluated on their ability to handle arbitrary error modes. Convergence is declared when the relative residual norm $\|r^n\|/\|r^0\|$ reaches $10^{-6}$ for both the domain decomposition solvers and preconditioners. When possible, the problems are also solved with GMRES with no restarts and no preconditioning to provide a reference. As the number of unknowns increases, the memory costs of GMRES becomes prohibitively expensive making this comparison unavailable for the larger problems.

#### 6.2.1 Parameter study: Circular surfaces

The (unit) circle is the simplest curved surface possible and forms an important test case for these methods. It is one dimensional, smooth, and has no boundary. First, a resolution of $\Delta x = 1/200$ is considered with bi-cubic interpolation used for the extension operator,
yielding 8864 active nodes globally and requiring 3347 iterations of non-preconditioned GMRES. The two dimensionality of the embedding space makes the linear system sparse, particularly if the constituent matrices in equation (2.11) are kept separate. The extension operator has 16 non-zeros per row and the ambient Laplacian (with the diagonal removed) only has 4. Table 6.2 gathers the relevant iteration counts together and Figure 6.1 displays the relative performance of the RAS and ORAS methods by plotting the ratio of their iteration counts. The (O)RAS methods diminish the solution cost greatly in terms of the iteration count and the time to solution.

<table>
<thead>
<tr>
<th>$N_S$</th>
<th>$N_O$</th>
<th>Solver</th>
<th>Preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>4</td>
<td>RAS $\alpha = 4$</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td></td>
<td>1508</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td></td>
<td>2068</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td></td>
<td>446</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td></td>
<td>914</td>
</tr>
<tr>
<td>36</td>
<td>8</td>
<td></td>
<td>1309</td>
</tr>
</tbody>
</table>

Table 6.2: The iteration counts for a unit circle surface discretized with bi-cubic interpolation and grid spacing $\Delta x = 1/200$ (a total of 8864 unknowns). GMRES with no restarts and no preconditioning requires 3347 iterations to meet the tolerance.

The resolution of the grid is doubled to $\Delta x = 1/400$ for a total of 17728 active nodes. This discretization requires 1359 iterations of GMRES without preconditioning to converge. Table 6.3 gathers the iteration counts for the whole of the parameter study with Figure 6.2 showing the ratio of the RAS iteration counts to the ORAS iteration counts.

<table>
<thead>
<tr>
<th>$N_S$</th>
<th>$N_O$</th>
<th>Solver</th>
<th>Preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>4</td>
<td>RAS $\alpha = 4$</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td></td>
<td>1671</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td></td>
<td>2615</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td></td>
<td>4405</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td></td>
<td>781</td>
</tr>
<tr>
<td>36</td>
<td>8</td>
<td></td>
<td>2045</td>
</tr>
</tbody>
</table>

Table 6.3: The same sweep as in Table 6.2 is considered but now with $\Delta x = 1/400$ with a total of 17728 unknowns requiring 1359 iterations of non-preconditioned GMRES. Again the new solvers are seen to perform quite well with the ORAS formulation providing a significant advantage.

The overall behavior of the methods are similar for these two resolutions, as one would hope. In both cases the reduction in iterations to convergence over the non-preconditioned case is quite large and makes these iterative methods quite competitive. The subdomain
Figure 6.1: The ratio of the ORAS to RAS iteration counts from Table 6.2 for the lower resolution case, $\Delta x = 1/200$, are shown here and give a good indication of their relative performance. The left panel shows the ratio of the iteration counts when the methods are used as solvers and the right panel shows the same when the methods are embedded within GMRES as preconditioners. Importantly, the cost of initializing these methods are all very nearly the same making these curves quantitatively meaningful.

count has the expected effect on the convergence behavior: the iteration count increases with the number of subdomains, as is visible in Figure 6.3. The overlap width also has the expected effect with the RAS methods speeding up as the width is increased and the ORAS methods improving somewhat before plateauing, which can be seen in Figure 6.4. Furthermore, we observe that the use of the Robin transmission conditions defined in equation (5.14) can provide a significant speed up over the RAS methods when the Robin weight is chosen appropriately. For all cases the required iterations increase when the grid is refined. However, the ORAS solvers and all of the preconditioners require only slightly more iterations while the RAS solvers require around twice as many iterations.
Figure 6.2: The ratio of the ORAS to the RAS iteration counts from Table 6.3 for the high resolution circle problem, using $\Delta x = 1/400$ for 17728 unknowns, are shown here. The relative performance of the ORAS methods against the RAS methods is more dramatic here for both the solvers (left) and preconditioners (right) than in the low resolution case.

Figure 6.3: The iterations to convergence for the circle problems at both resolutions and for both the (O)RAS solvers (left) and preconditioners (right) are compared against the number of subdomains. The low resolution runs are shown as solid lines while the high resolution tests are dashed. The RAS methods with Dirichlet transmission conditions are marked by circles and the ORAS methods with Robin conditions are denoted by triangles. All cases use an overlap of $N_O = 4$ and the ORAS methods use $\alpha = 16$. The overall trend is for the iterations to convergence to increase as the subdomain count increases, as predicted.
Figure 6.4: Here the overlap width is swept for the circle problems with the solvers shown on the left and preconditioners on the right. The lines are denoted in the same way as Figure 6.3. All cases use $N_S = 24$ subdomains with the ORAS methods employing a weight of $\alpha = 16$. The RAS solvers decrease in required iteration counts as the overlap is increased while the ORAS solvers appear totally independent of this amount in accord with Section 3.1.1.

6.2.2 Parameter study: Spherical surfaces

A natural follow-up to the circle is the sphere, embedded in three dimensions. The higher dimensionality complicates the subsurface boundary construction and tests the approximations made in Chapter 4. As before, we consider two mesh resolutions, both utilizing tri-quadratic interpolation: a coarse mesh with $\Delta x = 1/40$ with 139886 total unknowns and a finer mesh with $\Delta x = 1/80$ with 558806 total unknowns. The system sizes now make the generation of non-preconditioned GMRES results prohibitively expensive. As for the circle problems, we benefit from keeping the extension and ambient Laplacian as separate matrices with the former having 27 non-zeros per row and the latter having just 6. Here, and in all subsequent parameter studies, solves that failed to converge are denoted by – as an entry in the table.

The speedup offered by the ORAS methods over the RAS methods is less than for the circular problems. Figures 6.5 and 6.6 show speedups of the ORAS solvers and preconditioners over the RAS solvers and preconditioners. The higher resolution problem behaves more consistently than the low resolution problem. These marginal improvements match what was observed for the plane in Section 3.3 suggesting that the resolution of the problem is too low for the relative performance between the RAS and ORAS methods to be significant.
Table 6.4: The iterations to convergence for the unit sphere with tri-quadratic interpolation and $\Delta x = 1/40$ (a total of 139886 unknowns) are gathered here. The optimized solvers provide a significant advantage over the non-optimized solvers for good choices of the Robin weight. The optimized methods can offer only a slight speed up, as compared to RAS, when used as a preconditioner. This is expected for a low resolution problem like this, matching what was observed in Section 3.3.

<table>
<thead>
<tr>
<th>$N_S$</th>
<th>$N_O$</th>
<th>Solver</th>
<th>Preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>4</td>
<td>RAS $\alpha = 4$</td>
<td>RAS $\alpha = 4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>109</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td></td>
<td>36</td>
<td>87</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>134</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>51</td>
<td>68</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>177</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>150</td>
<td>203</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>91</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>63</td>
<td>78</td>
</tr>
<tr>
<td>36</td>
<td>8</td>
<td>114</td>
<td>78</td>
</tr>
</tbody>
</table>

Table 6.5: The iterations to convergence for the higher resolution sphere, with $\Delta x = 1/80$ and 558806 unknowns, can be seen here. For $\alpha = 8$ the ORAS solvers show stability problems but become reliable for $\alpha > 8$, where they have a stronger Dirichlet component. As in the low resolution case, the ORAS solvers demonstrate a greater improvement over the RAS solvers than the ORAS preconditioners do over the the RAS preconditioners.

<table>
<thead>
<tr>
<th>$N_S$</th>
<th>$N_O$</th>
<th>Solver</th>
<th>Preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>3</td>
<td>RAS $\alpha = 8$</td>
<td>RAS $\alpha = 8$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>214</td>
<td>16</td>
</tr>
<tr>
<td>48</td>
<td>3</td>
<td>320</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>539</td>
<td>64</td>
</tr>
<tr>
<td>24</td>
<td>6</td>
<td>196</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td></td>
<td>367</td>
<td>231</td>
</tr>
<tr>
<td>48</td>
<td>6</td>
<td>267</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>367</td>
<td>231</td>
</tr>
</tbody>
</table>

43
Figure 6.5: The ratio of the iteration counts from the parameter study from Table 6.4 for the unit sphere with grid spacing $\Delta x = 1/40$ and 139886 unknowns. The ORAS methods provide an advantage as solvers (left) for small values of the Robin weight. The ORAS preconditioners (right) provide an advantage only when the subproblems are sufficiently large and the overlap is small.

Figure 6.6: The iteration ratios for the higher resolution sphere problem, corresponding to Table 6.5. The solvers (left) have a more predictable behavior than the low resolution case. Additionally, all of the ORAS preconditioners (right) improve upon the RAS preconditioners, but to a lesser extent.
Figure 6.7: As the subdomain count increases for the sphere problems the iteration count increases for both the solvers (left) and preconditioners (right). The low resolution cases are shown as solid lines while the high resolution cases are given by dashed lines with the RAS methods marked by circles and the ORAS methods denoted by triangles. The overlap is fixed at $N_O = 6$ for all runs with the ORAS cases using a Robin weight of $\alpha = 16$.

Figure 6.8: The overlap width is varied on the sphere problems here with the lines denoting the same problems as in Figure 6.7. In most cases, the iterations required decreases with larger overlaps, following expectation, except for the high resolution problem treated by the ORAS solver. Throughout, $N_S = 48$ subdomains are used with the ORAS methods again using $\alpha = 16$. 
6.2.3 Parameter study: Toroidal surfaces

We select a torus to test the solvers and preconditioners on a surface that does not solely have positive curvature. The selected torus has a major radius of 0.75 and a minor radius of 0.25 so that its total size is comparable to unit sphere previously considered. The two considered resolutions use $\Delta x = 1/50$ for a total of 150544 unknowns, and $\Delta x = 1/100$ for a total of 605208 unknowns. Both utilize tri-quadratic interpolation. Tables 6.6 and 6.7 give the low and high resolution iteration counts respectively with Figures 6.9 and 6.10 showing this information as the ratio of the RAS iteration counts to the ORAS iteration counts.

<table>
<thead>
<tr>
<th>$N_S$</th>
<th>$N_O$</th>
<th>Solver</th>
<th>Preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>4</td>
<td>RAS $\alpha = 8$ 16 32 64 128</td>
<td>RAS $\alpha = 8$ 16 32 64 128</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>104 70 94 107 262</td>
<td>23 24 25 26 31 34</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>218 112 153 197</td>
<td>29 27 29 36 35</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>101 72 87 99 108 106</td>
<td>19 18 19 20 22</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>119 82 104 115 129 137</td>
<td>21 20 21 23 24</td>
</tr>
<tr>
<td>36</td>
<td>8</td>
<td>118 85 99 144 139</td>
<td>22 22 21 24 23 31</td>
</tr>
</tbody>
</table>

Table 6.6: The low resolution torus problem uses $\Delta x = 1/50$ with tri-quadratic interpolation yielding 150544 active nodes. The same subdomain counts, overlap widths, and Robin weights as the sphere problems are used here.

<table>
<thead>
<tr>
<th>$N_S$</th>
<th>$N_O$</th>
<th>Solver</th>
<th>Preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>3</td>
<td>566 – 403 545</td>
<td>42 39 40 39 42 48</td>
</tr>
<tr>
<td>48</td>
<td>3</td>
<td>684 – 512 687 1035</td>
<td>49 50 48 44 48 52</td>
</tr>
<tr>
<td>72</td>
<td>3</td>
<td>774 – 453 621</td>
<td>54 93 50 51 54 61</td>
</tr>
<tr>
<td>24</td>
<td>6</td>
<td>365 124 210 292 358 404</td>
<td>33 26 28 31 33 35</td>
</tr>
<tr>
<td>48</td>
<td>6</td>
<td>459 – 284 376 453 511</td>
<td>38 53 34 35 38 40</td>
</tr>
<tr>
<td>72</td>
<td>6</td>
<td>397 – 270 341 421 815</td>
<td>43 45 37 39 42 47</td>
</tr>
</tbody>
</table>

Table 6.7: The high resolution torus, using $\Delta x = 1/100$ for 605208 active nodes, behaves similarly to the high resolution sphere. The ORAS solvers face stability issues for small values of $\alpha$, but become reliable and provide speed up after increasing the weight. All ORAS preconditioners have some value of $\alpha$ providing speed up over their RAS counterparts, but not to a great extent.

The region of negative curvature on the torus presents more difficulty in the construction of the effective boundary and conormals for the subsurfaces compared to the sphere. The ORAS methods are more sensitive to the Robin weight than before, particularly in the low resolution case. The benefit provided by the ORAS methods over the RAS methods remains small at the resolutions accessible for testing.
Figure 6.9: The low resolution torus test case uses $\Delta x = 1/50$ for a total of 150544 unknowns. The ORAS solvers provide an advantage over the RAS solvers for small values of $\alpha$ in all tested cases but become unreliable after that. Similar to our prior observations the ORAS preconditioners perform best with a small number of subdomains and limited overlap widths.

Figure 6.10: For the high resolution torus we see greater regularity in the performance of the (O)RAS methods with respect to the value of the Robin parameter. The extent to which the ORAS methods improve over the RAS methods is better than in the low resolution case presented in Figure 6.9, but remains small at this resolution.
Figure 6.11: As the subdomain count is increased on the torus the required iterations to converge generally increase, as would be expected. Both of the resolutions have too few unknowns for the large subdomain count to form reasonable subproblems and the final ORAS solver fails. Similarly, the final case for the ORAS preconditioners pass the RAS preconditioners, requiring more iterations to converge.

Figure 6.12: As the overlap width is increased on the torus the (O)RAS solvers and preconditioners require fewer iterations to converge, with the exception of the largest overlap ORAS solver on the high resolution problem.
6.2.4 Parameter study: Triangulated surfaces

To test the applicability of these (O)RAS methods to more general surfaces we use the famous Stanford Bunny [27], scaled to fit its longest dimension into the interval $(-1, 1)$, with the five holes in the bottom filled, and the triangulation slightly coarsened. Figure 6.13 shows the used triangulation and these modifications. We emphasize that these modifications present no difficulty for the defined methods, but rather keep the run time reasonable for the large number of tests needed for the parameter study. This keeps the meaning of the grid spacings consistent with the prior results and continues to avoid the existence of global boundaries. This constitutes the most complicated application of the (O)RAS solvers and preconditioners. The closest point function is no longer given in an exact form and we now need to see how the approximations made in constructing it compound with those made during the subdomain formation.

Figure 6.13: The left shows the used triangulation for the Stanford bunny surface with the noted coarsening. The right shows the filled holes from below, visible as the highly skewed triangles.

Following the parameter studies before, two mesh resolutions are considered with the gathered iteration counts presented in Tables 6.8 and 6.9 for spacings of $\Delta x = 1/75$ and $\Delta x = 1/100$ respectively. The resulting systems have 371659 and 661958 unknowns respectively. Furthermore, Figures 6.14 and 6.15 visualize the relative performance of the optimized and non-optimized methods. Finally, Figures 6.16 and 6.17 vary the subdomain count and overlap width independent from any other parameters.

Throughout these parameter sweeps one can see that the (O)RAS methods perform well on triangulated surfaces, despite the approximations present. The speed up of the optimized methods over the non-optimized methods is particularly impressive considering the difficulty in resolving the subdomain boundary geometry. Similar to all of the prior
| \(N_S\) | \(N_O\) | \(\text{Solver} \quad \alpha = 8\) | 16 | 32 | 64 | 128 | \(\text{Preconditioner} \quad \alpha = 8\) | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|---|---|---|
| 24 | 3 | 489 | 249 | 353 | 453 | 536 | 596 | 41 | 34 | 38 | 41 | 44 |
| 48 | 3 | 758 | 372 | 528 | 682 | 813 | 907 | 51 | 42 | 46 | 50 | 53 | 56 |
| 72 | 3 | 941 | 446 | 634 | 821 | 974 | 1084 | 57 | 49 | 51 | 56 | 59 | 62 |
| 24 | 6 | 309 | 174 | 226 | 273 | 308 | 330 | 32 | 28 | 29 | 30 | 32 | 33 |
| 48 | 6 | 472 | 266 | 350 | 422 | 474 | 506 | 40 | 34 | 36 | 38 | 40 | 41 |
| 72 | 6 | 592 | 320 | 420 | 507 | 572 | 617 | 44 | 40 | 41 | 43 | 45 | 46 |

Table 6.8: The scaled and patched Stanford Bunny is discretized with \(\Delta x = 1/75\) and tri-quadratic interpolation, giving 371659 total unknowns. The ORAS methods remain robust in all cases and provide a significant advantage when run as a solver over the RAS methods. The ORAS preconditioners continue to show little improvement over the RAS preconditioners at these low resolutions.

<table>
<thead>
<tr>
<th>(N_S)</th>
<th>(N_O)</th>
<th>(\text{Solver} \quad \alpha = 8)</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>(\text{Preconditioner} \quad \alpha = 8)</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>3</td>
<td>657</td>
<td>262</td>
<td>385</td>
<td>516</td>
<td>640</td>
<td>734</td>
<td>48</td>
<td>37</td>
<td>44</td>
<td>47</td>
</tr>
<tr>
<td>48</td>
<td>3</td>
<td>929</td>
<td>–</td>
<td>564</td>
<td>743</td>
<td>917</td>
<td>1051</td>
<td>58</td>
<td>68</td>
<td>50</td>
<td>54</td>
</tr>
<tr>
<td>72</td>
<td>3</td>
<td>1142</td>
<td>–</td>
<td>922</td>
<td>1138</td>
<td>1303</td>
<td>56</td>
<td>55</td>
<td>57</td>
<td>63</td>
<td>69</td>
</tr>
<tr>
<td>24</td>
<td>6</td>
<td>406</td>
<td>185</td>
<td>257</td>
<td>325</td>
<td>379</td>
<td>416</td>
<td>38</td>
<td>29</td>
<td>32</td>
<td>35</td>
</tr>
<tr>
<td>48</td>
<td>6</td>
<td>577</td>
<td>288</td>
<td>395</td>
<td>491</td>
<td>566</td>
<td>616</td>
<td>45</td>
<td>37</td>
<td>39</td>
<td>41</td>
</tr>
<tr>
<td>72</td>
<td>6</td>
<td>718</td>
<td>–</td>
<td>460</td>
<td>577</td>
<td>668</td>
<td>732</td>
<td>52</td>
<td>53</td>
<td>46</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 6.9: The Stanford Bunny surface with \(\Delta x = 1/100\) yields 661958 unknowns. Compared to the low resolution bunny problem before, this test shows stability issues for the small values of \(\alpha\), similar to the high resolution torus and sphere. All ORAS cases provide a speed up over their RAS counterparts with the gap remaining wider for the solvers.
parameter sweeps, the ORAS solvers converge the fastest for small values of the Robin weight $\alpha$ but face stability issues when it is chosen too small. The ORAS preconditioners see their best performance for the same or slightly larger values of $\alpha$.

Figure 6.14: The low resolution grid over the scaled Stanford Bunny surface, with $\Delta x = 1/75$, paired with tri-quadratic interpolation yields 371659 active nodes. The ORAS solvers (left) provide a significant advantage for small $\alpha$ for all subdomain choices. The ORAS preconditioners (right) also perform well, although the improvement is less pronounced than for the solver case.

Figure 6.15: The high resolution Stanford Bunny problem takes $\Delta x = 1/100$ with tri-quadratic interpolation for a total of 661958 unknowns. The speed up attained by the ORAS over RAS is greater for this high resolution test compared to the low resolution problem considered in Figure 6.14.
Figure 6.16: As the number of subdomains used for the Stanford Bunny problems is increased the number of iterations to converge increases, regardless of method and resolution. The iteration count for the solvers is shown on the left and the count for preconditioned GMRES on the right. Low resolution problems are given by solid lines and high resolution problems given by dashed lines, with circles denoting RAS methods and triangles marking the ORAS methods. Finally, for all ORAS results the Robin parameter has been fixed at $\alpha = 16$ with an overlap of $N_O = 3$.

Figure 6.17: The overlap for the Stanford Bunny is varied here with the lines marked in the same way as in Figure 6.16. All methods see diminished iteration counts as the overlap width is increased, with the RAS methods relying more heavily on the width than the ORAS methods. The Robin parameter was fixed at $\alpha = 16$ with subdomain counts of $N_S = 24$ and $N_S = 48$ for the low and high resolution problems respectively.
6.3 Computational scaling

Parallel scalability measures how well a parallel algorithm is able to take advantage of increasingly larger machines. Herein we are concerned with two types of parallel scaling, strong scaling which measures the speedup attained by using more processes on a fixed problem size, and weak scaling which increases the problem size with the process count in an effort to keep the number of unknowns per process fixed, and ideally displays constant run time. These metrics can build intuition about how to partition future problems, but face two mitigating factors in their present usefulness. First, as the problem size increases the conditioning of the global system degrades and the overall convergence behavior of the method changes independent of the parallelism. Second, there is no clear way to choose the number of subdomains per process. The larger processor counts allow greater parallelism and an ability to take advantage of more subdomains, but of course increasing the number of subdomains also increases the iterations to convergence. Two level methods [6] are designed to avoid these issues with scalability but lie beyond the scope of this thesis.

The strong scaling results are generated not only with a fixed problem size but also a fixed total number of subdomains, which effectively removes these problems. The weak scaling results are found with a fixed number of subdomains per process so that the total number of subdomains scale with the problem size. Unfortunately, this means that in the weak scaling study the strength of the preconditioner will degrade as conditioning of the system worsens. All results in the following have been averaged over 5 runs to mitigate the effect fluctuations in the cluster and differing node hardware.

6.3.1 Strong scaling

Strong scaling results are gathered in Tables 6.10 and 6.11 for the circle and torus respectively. In each case the resolution of the problem and the number of subdomains used to partition the mesh and generate the local problems is fixed independent of the number of processes. In both cases, we keep approximately 20000 nodes in each disjoint partition. Finally, the cluster we have access to has demonstrated poor communication latency so only the preconditioner performance will be tested here.

Following the parameter study above, we begin with the circle as a simple problem. The lower dimensionality reduces the number of nodes in the overlap regions significantly compared to two dimensional surfaces embedded in three dimensions, and as such, should demonstrate less communication overhead. The circle is discretized with $\Delta x = 1/20000$ and tri-cubic interpolation yielding 888344 total unknowns. The subdomain count is $N_S = 48$ so that each contains approximately 20000 nodes. Table 6.10 gathers the scaling results for this problem over 1, 2, and 4 nodes, using 12, 24, and 48 processes respectively. The setup time consisting of the local mesh and operator construction as well as its factorization, and
the time to convergence of GMRES are pulled out from the total time so that one may see how each phase scales as well as the total time.

<table>
<thead>
<tr>
<th>Nodes (Processes)</th>
<th>1 (12)</th>
<th>2 (24)</th>
<th>4 (48)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup Time [s]</td>
<td>193</td>
<td>100</td>
<td>53</td>
</tr>
<tr>
<td>GMRES Time [s]</td>
<td>2.5</td>
<td>2.1</td>
<td>3.3</td>
</tr>
<tr>
<td>GMRES Iterations</td>
<td>42</td>
<td>43</td>
<td>41</td>
</tr>
<tr>
<td>Time to solution [s]</td>
<td>223</td>
<td>122</td>
<td>71</td>
</tr>
</tbody>
</table>

Table 6.10: The unit circle is discretized with bi-cubic interpolation and $\Delta x = 1/20000$ for a total of 888344 unknowns. The number of subdomains used is fixed at $N_S = 48$ to place approximately 20000 nodes in each disjoint subdomain. We can see here that the setup cost scales almost perfectly since the communication is minimal in this phase. The time for GMRES to converge is negligible in this simple example and doesn’t show any meaningful scaling.

As a follow up to the unit circle, the torus, as described in Section 6.2.3, is again discretized with $\Delta x = 1/100$ and tri-quadratic interpolation giving a system with 605208 unknowns. Once again the total subdomain count is fixed at $N_S = 48$ with each disjoint subdomain now containing approximately 12000 nodes. Table 6.11 holds the scaling results on 1, 2, and 4 nodes, totalling 12, 24, and 48 processes respectively. Compared to the circle problem shown in Table 6.10, the setup cost here is higher even though the number of nodes in each subdomain is larger. The only significant difference between this problem and the former is the reduced sparsity of the extension and Laplacian operators. The cost of the matrix-matrix multiplication and the factoring of the resulting operator is significant and hints at the benefit in keeping the global operator in its unassembled form. Additionally, the cost of each global matrix-vector multiply is significantly higher as can be seen by the large amount of time spent in GMRES even though the total iteration counts are nearly the same.

<table>
<thead>
<tr>
<th>Nodes (Processes)</th>
<th>1 (12)</th>
<th>2 (24)</th>
<th>4 (48)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup Time [s]</td>
<td>887</td>
<td>449</td>
<td>300</td>
</tr>
<tr>
<td>GMRES Time [s]</td>
<td>32</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>GMRES Iterations</td>
<td>48</td>
<td>46</td>
<td>49</td>
</tr>
<tr>
<td>Time to solution [s]</td>
<td>911</td>
<td>475</td>
<td>318</td>
</tr>
</tbody>
</table>

Table 6.11: The torus discretized by a grid with spacing $\Delta x = 1/100$ and tri-quadratic interpolation yields a problem with 605208 unknowns. Again 48 subdomains are used in total with the number per process decreasing as more processes are introduced. The results moving from 1 to 2 nodes show the expected speed up. The 4 node results do not show the expected improvement over the 2 node results in the setup phase where communication should provide little impediment. The GMRES solve scales well throughout.
There are two main contributors to the time to solution: the time to build and factor all of the local systems, and the time for GMRES to converge. The first scales well since the total number of local problems and their size is fixed. With more processes each is responsible for handling fewer local problems and throughout this phase the communication is minimal. The effect of processor count on the time to convergence of GMRES is more complicated. In each iteration there is a global matrix vector multiply, the application of the preconditioner, and calculation of the residual norm. The global matrix vector multiply requires some communication since the matrix is split by rows across processes and not all components of the vector, required to compute a process’ contribution, are stored locally. The application of the preconditioner is made cheaper with more processes since each is responsible for fewer forward and back solves, however the residual in the overlap regions has to be populated through communication resulting in some overhead. The (O)RAS family of methods require no communication for the solution update. Finally, the computation of the residual norm requires a small amount of communication to share the local sums with each other, but this mostly impedes progress by forcing all processes to synchronize after every iteration.

6.3.2 Weak scaling

Achieving good weak scalability with a PDE solver is difficult since as the problem grows the conditioning of the arising system generally degrades, and simultaneously, the strength of the preconditioner diminishes as a sacrifice for increased parallelism. Here we consider the circle and torus as in the prior section. Now, the number of subdomains will be set equal to the number of processes and with the resolution varied to keep the number of unknowns per process close to $10000$. First the circle is considered in Table 6.12 with grid spacings of $\{1/2750, 1/5500, \text{ and } 1/11000\}$ for $\{122152, 244288, \text{ and } 488800\}$ total unknowns to be solved on $\{12, 24, \text{ and } 48\}$ processes respectively. Following this, the weak scaling results for the torus are gathered in Table 6.13 for grid spacings of $\{1/45, 1/65, \text{ and } 1/90\}$ yielding $\{122544, 255760, \text{ and } 489704\}$ total unknowns.

<table>
<thead>
<tr>
<th>Nodes (Processes)</th>
<th>1 (12)</th>
<th>2 (24)</th>
<th>4 (48)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_A (\Delta x)$</td>
<td>122152 (1/2750)</td>
<td>244288 (1/5500)</td>
<td>488800 (1/11000)</td>
</tr>
<tr>
<td>Setup Time [s]</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>GMRES Time [s]</td>
<td>0.19</td>
<td>0.55</td>
<td>1.2</td>
</tr>
<tr>
<td>GMRES Iterations</td>
<td>22</td>
<td>30</td>
<td>43</td>
</tr>
<tr>
<td>Time to solution [s]</td>
<td>18</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 6.12: The circle is considered again, but here with the number of subdomains set equal to the number of processes and with the resolution varied to keep approximately $10000$ unknowns per process. The setup phase scales perfectly, and again the GMRES solve is too small to make any meaningful claims. The problems are small enough that the serial portions of code show in the total time results.
Both surfaces exhibit excellent weak scalability in the setup phase. The communication here is minimal and the parallelism is nearly trivial making the achievement of scaling here quite simple. The GMRES solve for the circle is too brief to provide any meaningful scaling information, while the torus shows the expected behavior. The iteration count climbs as the problems grow and the preconditioner weakens. The total time to solution scales well since the setup cost is dominant for both surfaces.

<table>
<thead>
<tr>
<th>Nodes (Processes)</th>
<th>1 (12)</th>
<th>2 (24)</th>
<th>4 (48)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_A (\Delta x))</td>
<td>255760 (1/65)</td>
<td>489704 (1/90)</td>
<td>945900 (1/125)</td>
</tr>
<tr>
<td>Setup Time [s]</td>
<td>336</td>
<td>312</td>
<td>276</td>
</tr>
<tr>
<td>GMRES Time [s]</td>
<td>10</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>GMRES Iterations</td>
<td>35</td>
<td>42</td>
<td>61</td>
</tr>
<tr>
<td>Time to solution  [s]</td>
<td>351</td>
<td>330</td>
<td>305</td>
</tr>
</tbody>
</table>

Table 6.13: Weak scaling for the torus. The grid size is varied to place approximately 10000 unknowns in each subdomain while the number of subdomains is taken to be double the number of processes. The setup phase and total time exhibit excellent weak scaling, even diminishing in time as the process count is increased. The time spent in GMRES climbs as the required number of iterations increases due to the simultaneously harder problem and weaker preconditioner.
Chapter 7

Concluding remarks

We have sought to solve linear systems of equations from the discretization of elliptic PDEs posed intrinsic to surfaces, Poisson equations in particular, in a parallel and performant manner. We started by selecting the closest point method for the discretization of these problems. In Chapter 2 the closest point function and the associated extension operator were defined and their role in moving the problem off the surface and into an embedding space was shown. Once in the embedding space, standard discretizations can be utilized, with centered second-order differences being selected for simplicity. Additionally, we saw that the direct form of the discrete operator is ill suited for eigenvalue problems and implicit time stepping of diffusive phenomena. As such, the implicit closest point method was introduced so that the remaining work could be applied in these contexts.

With a discretization for the target problem in hand we moved to parallel solvers. Domain decomposition was chosen for this goal, with Chapter 3 being dedicated to its formulation. First we considered continuous problems and Schwarz’s original intentions. To bring this from a technique in analysis to a parallel numerical method we considered an additive variant that allows all subproblems to decouple and be solved simultaneously, exposing parallelism. Further we considered additional transmission conditions on the artificially imposed boundaries to accelerate the convergence. Importantly, we pulled this iterative scheme from the continuous into the discrete to obtain a linear system solver. After some manipulation this solver was written to provide solution corrections instead of entire new solutions and finally placed into a form which could be embedded within a Krylov subspace solver like GMRES. As a preliminary example we considered these methods applied to a standard Poisson problem on the square discretized by centered finite differences, and found importantly that the improved convergence rate afforded by the optimized methods depended on the mesh resolution.

To apply these domain decomposition schemes to the closest point method we first need a way to decompose the domain. Chapter 4 was dedicated to the formation of subdomains from the global meshes used in the closest point method. For generality, a graph partitioning tool, already widely in use in the domain decomposition community, was discussed as
a means for splitting the global mesh into smaller regions. We saw that even though there are many ways to form a graph over the closest point grid, the naive approach yielded the best results. The disjoint partitions generated by METIS could be further tweaked to give subdomains aligned with the surface normals. An overlapping partitioning was easily generated over this disjoint partitioning, giving enough to apply the RAS methods. However, to utilize the ORAS methods with Robin conditions imposed on the subproblems we needed to also generate the boundary geometry. Boundary nodes were applied around the overlapping partitions and identified with their closest point on the subdomain boundary and the conormal direction along which they should enforce the derivative condition. Within this boundary geometry construction the arising conormal vectors could vary significantly between neighbors and some smoothing strategies were discussed.

The final component in the definition of the domain decomposition solvers and preconditioners is to extend the earlier discussion on the closest point method to include surfaces with boundaries. In Chapter 5 we considered first and second order accurate discretizations of Dirichlet boundary conditions and saw that incorporating them into the closest point method required modifying the extension operator. The first order accurate Dirichlet conditions were found to correspond precisely to the algebraic form of the RAS method and selected for further use. First order accurate Robin boundary operators were formed similarly by combining the first order accurate Dirichlet condition with a forward difference along the conormal direction generated in the prior chapter. Finally, the new Robin conditions were verified as being first order accurate with a simple convergence study.

From the parameter sweeps from the first half of Chapter 6 we saw that the ORAS methods always provide a speed up relative to the RAS methods. The performance gap is substantial when they are used as solvers but somewhat diminished when used as preconditioners for GMRES. The ORAS methods are also sensitive to the subdomain geometry for small values of $\alpha$, which unfortunately is also the most performant regime. Both RAS and ORAS make the use of iterative solvers for these systems competitive, particularly when used as a preconditioner for GMRES. The iteration count has been reduced in every case to a number that can be treated without needing restarts, which is a huge improvement over non-preconditioned GMRES. Similarly, when used as solvers these methods require no history of solutions making memory bound problems accessible at the cost of extra run time.

The parallel scalability of these methods was demonstrated in the latter half of Chapter 6. Throughout, we found that the methods exhibited good strong and weak parallel scaling over the problem sizes available. The setup cost was dominant in all examined problems indicating that the developed solvers would provide the greatest benefit in a setting where the same system needs to be solved many times over.
7.1 Picking a method

Through this thesis we have seen a large number of options and picking a particular solver from this whole family to treat a specific problem is not trivial. ORAS is generally a better choice than RAS and should be used unless the implementation proves difficult in a given setting or no robust splitting can be found. The Robin weight will need some experimentation but should be chosen as small as possible without introducing instability. Regardless of choosing RAS or ORAS, the subdomain count should be chosen to place $10000 - 20000$ active nodes in each disjoint partition and the overlap width should be kept minimal. An overlap equal to the size of the interpolation stencil, $N_O = p + 1$, has been found to work well.

7.2 Future directions

The methods developed and tested herein motivate many further investigations. Several interesting options are laid out here, roughly ordered from building directly on this work to more substantial departures. The framework introduced here could be used for (optimized) multiplicative Schwarz schemes. Paired with a suitable subdomain coloring scheme one could retain parallelism and hopefully find a more robust method over all. The main foreseeable obstacle would be coloring in a way that respects not just the disjoint partitioning but also the overlapping one without requiring a substantial and costly amount of communication. Another option would be to generalize the Robin transmission conditions developed here to the treatment of non-overlapping grids. Maintaining consistency of the conormal directions used by neighboring subdomains will require some effort but the reward would be smaller local problems and the elimination of the overlap condition. Any cost reduction in the construction of the local operators and application of the preconditioner each iteration would provide significant time savings. One approach to achieve this would be to increase the sparsity of the local operators, perhaps by mismatching the polynomial degree used for the extension operators. The use of two level Schwarz schemes could provide better parallel scaling and improve the performance of the methods for large subdomain counts. The construction of consistent and stable coarse spaces is in general difficult and may require substantial work. A final possibility would be the application of these solvers to an existing problem currently limited to direct solvers.
Bibliography


Appendix A

Implementation errata

Through the progression of this work, a code written in C++ was written to implement and test all ideas. Distributed memory parallelism is supported through the Message Passing Interface (MPI). PETSc [3] lies at the core of the implementation, providing the iterative linear solvers and distributed memory data structures for matrices and vectors. METIS [8] is accessed through PETSc and provides the graph partitioning discussed in Section 4.1. The classes created are quickly summarized here:

- **Struct: ProblemDefinition**: This struct holds all needed parameters to define the surface, closest point discretization, and domain splitting. Consult the sample input file and command line options in Sections A.3.2 and A.3.1.

- **Class ProblemSetup**: This class populates the entries of the ProblemDefinition structure by reading an input file and command line arguments.

- **Class TVec**: This class defines a simple templated vector class and is used for storing the integer coordinates of grid points as well as defining the surface normal and conormal directions. Additionally, these vectors can hold the extension stencil indices and weights for each point. These vectors are not used in the definition of any linear systems.

- **Class QOTree**: This implements a quadtree/octree structure as a hash map with key values given by the Morton encoding [2]. This allows the storage of the closest point meshes with minimal overhead while allowing fast queries. The meshes are stored with respect to a bounding box under a fixed grid spacing. As such each active (or ghost) node can be identified with an integer tuple giving its position within the box. The hash map stores only the computational tube by identifying the useful tuples with the index of that point within the set of unknowns. This allows the indices of neighboring nodes and nodes within extension stencils to be found easily when no simple offset can be used like in a globally rectilinear grid.

- **Class CPMesh**: This class provides the meshes and all geometric information for the global and local problems. The extension stencils and weights are all generated here, along with the boundary information for the Robin transmission conditions on
the local problems. The QOTree class is used internally. See Section A.1 for greater detail.

- **Class DiffOps:** This acts as a translation from the CPMesh objects into the linear systems to be solved. Routines are provided to construct the Ambient Laplacian and extension matrices with respect to a given mesh. This also provides the initial guesses and right hand sides for the defined problems, as well as the exact solutions when they are available.

- **Class Problem:** This class contains all of the PETSc objects defining the linear systems and the discussed preconditioners. Both global and local problems are created as instantiations of this class, mimicking the behavior of CPMesh and DiffOps. More detail is provided in Section A.2.

- **Class CPPostProc:** Finally, this class writes out all data in the SILO [1] data format for later visualization. The global solution, residual, forcing, and surface normals are written out as both point data over the computational band and surface limited values. Geometric quantities for the first subdomain can also be written out to see the effective boundary and conormal vectors.

The CPMesh and Problem classes are the most substantial and will be discussed in greater detail in the following sections. Finally, a simple driver code and sample run are supplied. All code written to support this thesis, as well as more extensive documentation, can be found in a public repository [14].

### A.1 CPMesh

The global mesh is instantiated as an object of the class CPMesh using the options defined in the ProblemDefinition structure. Serial meshes, which store a copy of the entire global mesh on each process, and distributed meshes, which store the locally relevant portion of the mesh on each process, are both supported. For either mesh type the construction follows roughly the same steps.

A bounding box surrounding the surface and containing the computational band is traversed, respecting the grid spacing, and all nodes within the bandwidth are added to a QOTree object. The index for each one is incremented leaving the unknowns in the lexicographic ordering. For distributed meshes this is done in slices, and after each process counts their local number of found nodes the next process increments the stored indices to produce a complete global set of indices. This QOTree representation of the grid is used to generate a METIS compatible graph. In the distributed case, this is done on a per-process basis and the global graph is stitched together on the root process. From the partitioning the indices are permuted, such that the subdomains each have a continuous set of indices. For distributed meshes the needed coordinates must also be communicated across processes to complete the permutation. The first index of each partition is stored and later used to construct the subdomain meshes. Distributed meshes need to store more than just their subdomain(s) nodes. To simplify the later construction of the individual subdomain meshes the overlap nodes and their indexing also need to be copied between the processes. Finally,
the ghost nodes are layered over the mesh and their indexing resolved between processes if needed. The global CP\texttt{Mesh} object also provides the extension stencils and weights but these are formed as the global Problem object requests them.

The local meshes are created using the global mesh, the global problem, and the indices in the global numbering delimiting the target disjoint partition. These do not need to form \texttt{QOTree} objects to store the grid points as instead they form maps associating the global indices to the local indices. Requests for neighboring grid points are fulfilled by querying the \texttt{QOTree} object in the parent global mesh and translating the indices to match the local numbering. After reading in the disjoint partition from the global mesh, the overlaps are added and the boundary geometry is generated as needed.

### A.2 Problem

The Problem object defining the global system is created following the generation of the global CP\texttt{Mesh} and Diff\texttt{Ops} objects. The extension matrix and ambient Laplacian with the diagonal removed are created as PETSc Mat objects. The matrices are sparse and distributed according to which disjoint subdomains are associated to each process. From these, the shifted surface Laplacian from equation (2.11) is formed as a PETSc shell matrix whose matrix-vector multiply operation is given by applying each operation in succession. The global problem can be used as is for non-preconditioned solves, but otherwise needs the local problems to be constructed. To keep the PETSc objects limited to one class, the least squares solvers for the conormal smoothing within the local mesh creation also lie in this object.

Once the local meshes have been created the local problems may be formed. The extension operators within these problems are formed with respect to the transmission conditions while reusing the global stencils and weights when possible. The local operators must be factored to provide a direct solver and thus the constituent matrices must be multiplied out into the final operator in contrast to the global case.

The local problems are given to the global problem to construct the preconditioner. The maps between global and local indices are formed into PETSc IS (index set) objects to transfer globally supported vectors into locally supported vectors, and vice versa. These together are placed into a shell preconditioner, PETSc PC object, whose action applies the preconditioner to any input vector. The Diff\texttt{Ops} object is queried to create the right hand side and the initial guess. This preconditioner can then be associated to a GMRES solver or used on its own as a solver.

The problem class also has limited support for transient problems which will likely be enlarged later. Currently, only the BDF2 [9] scheme is supported for the heat equation and a few reaction diffusion equations. Limited support is also available for two level ORAS methods using Petrov-Galerkin coarse operators and smoothed aggregation bases.
A.3 Sample run

An annotated sample input file is supplied here followed by some exemplary calls to the executable DDCPSolver, found within the repository upon building it. The full code, including this input file, can be found in a publically accessible repository [14].

A.3.1 Sample input file

The large number of settings required to specify the problem and the behavior of the code motivate the use of input files. Shown here is an annotated input file defining a shifted Poisson equation on a circle.

# Solve shifted Poisson equation on a circle
# Verbose input file to create others from
# Hashes denote comments and are ignored

# All capitals inform a block of common purpose values to set
# They can be called in any order
MESH
Surface Circle
Resolution 200 # The desired resolution, dx=1/res
Dimension 2 # Embedding dimension
GlobalOrder 3 # Interpolation order, global problem
LocalOrder 3 # Interpolation order, local problems
BoundaryOrder 3 # Interpolation order, boundary operators
BoundingBox 410 410 # Upper corner of bounding box
Overlap 4 # Overlap size, N_O
Partitions 24 # Number of partitions, N_S
AlignBounds True # Align subdomain boundaries
ConormSmooth True # Enable conormal smoothing
MeshType Serial # Storage scheme for mesh
PartitionScheme MetisAmb # Partitioning scheme
Poll 512 # Number of polling points

DOMAINDECOMPOSITION
Coarse False # Use coarse space
RandomInitial False # Random initial solution guess
MaxIterations 2000 # Maximum DD solver iterations
Tolerance 1.e-6 # Tolerance on relative residual
Transmission RobFO # Transmission conditions
OSMAlpha 16.0 # Alpha value for Robin conditions

EQUATION
Equation Poisson # Set equation to solve
Parameters 1.0 # c value
A.3.2 Sample calls to the executable

Some sample calls to the executable DDCPSolver, found within the repository [14], are shown.

Run in serial just using the settings in the input file:

```bash
./DDCPSolver -infile inputFiles/circlePoisson.icpm \
-ksp_converged_reason
```

The PETSc flag `-ksp_converged_reason` ensures that GMRES outputs how many iterations it needed and why it converged (or failed).

Run in parallel with Dirichlet transmission conditions and 12 subdomains instead of 24:

```bash
mpiexec -n 4 ./DDCPSolver -infile inputFiles/circlePoisson.icpm \
-ksp_converged_reason -parts 12 -dd_trans_dirfo
```

Note that the number of processes (4) must divide the number of subdomains (12).

Finally, run with written data files suppressed and show the logging information upon completion:

```bash
mpiexec -n 4 ./DDCPSolver -infile inputFiles/circlePoisson.icpm \
-ksp_converged_reason -parts 12 -dd_trans_dirfo -out_none -log_view
```

Notably, the behavior of the code can be changed substantially from the command line, with no modifications to the driver code or input file required. Run with the flag `-help` to see all of the possible flags with short descriptions. See the documentation in [14] for more information.