Algebraic Domain Decomposition Preconditioners

L. Giraud[*] R. S. Tuminaro[†]

Technical Report ENSEEIHT-IRIT RT/APO/06/07
October 2006

Abstract

In this chapter, some popular and well-known domain decomposition preconditioners are described from an algebraic perspective. Specific emphasis is given to techniques that are well-suited to the parallel solution of large-scale scientific applications and industrial numerical simulations. Some computational aspects related to their parallel implementation are also addressed. This chapter is not intended for specialists in domain decomposition but rather for scientists who have some knowledge of linear algebra and discretization techniques and who would like an introduction to domain decomposition.
Keywords : algebraic preconditioners, matrix partitioning, mesh partitioning, overlapping techniques, non-overlapping approaches, two-level preconditioning.

1 Introduction

The term *domain decomposition* covers a fairly large range of computing techniques for the numerical solution of partial differential equations (PDEs) in time and space. Generally speaking, it refers to the splitting of the computational domain into sub-domains with or without overlap. The splitting strategy is generally governed by various constraints/objectives. It might be related to

- some PDE features to, for instance, couple different models such as the Euler and Navier-Stokes equations in computational fluid dynamics;
- some mesh generator/CAD constraints to, for instance, merge a set of grids meshed independently (using possible different mesh generators) into one complex mesh covering an entire simulation domain;
- some parallel computing objective where the overall mesh is split into sub-meshes of approximately equal size to comply with load balancing constraints.

In this chapter we consider this latter situation and focus specifically on the associated domain decomposition techniques for the parallel solution of large linear systems, $Ax = b$, arising from PDE discretizations. Some of the presented techniques can be used as stationary iterative schemes that converge to the linear system solution. However, domain decomposition schemes are most effective and require less tuning when they are employed as a preconditioner to accelerate the convergence of a Krylov method [31, 48]. Used as a preconditioner, these schemes effectively define a non-singular matrix M which transforms the original linear system solved by the Krylov technique

[*]ENSEEIHT-IRIT, 2 rue Camichel, 31 071 Toulouse, France.

[†]MS 9159 - Sandia National Laboratory, Livermore, CA, 94551 USA. Supported by the Applied Mathematical Sciences program, U.S. Department of Energy, Office of Energy Research, and was performed at Sandia National Laboratories, operated for the U.S. Department of Energy under contract No. DE-AC04-94AL85000.

to either $MAx = Mb$ (left preconditioning), $AMy = b$ with $x = My$ (right preconditioning), or $M^{\frac{1}{2}}AM^{\frac{1}{2}}y = M^{\frac{1}{2}}b$ with $x = M^{\frac{1}{2}}y$ (symmetric preconditioning). It is important to recognize that Krylov methods do not require the matrices A, M or $M^{\frac{1}{2}}$ to be formed. Instead, procedures for applying A and M to a vector must be provided. The numerical requirements for a good preconditioner is that the spectrum of the preconditioned matrix is clustered. Such a feature ensures fast convergence of the conjugate gradient method (CG) for symmetric positive definite (SPD) problems as illustrated by the CG convergence rate bound given by [30]:

$$||e^{(k)}||_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k ||e^{(0)}||_A,$$

where $e^{(k)} = x^* - x^{(k)}$ denotes the error associated with the iterate at step k and κ is the condition number of the preconditioned linear system $M^{\frac{1}{2}}AM^{\frac{1}{2}}$ (which is simply the ratio of the largest to smallest eigenvalue). From this bound, it can be seen that when the condition number is small (i.e. $\kappa \approx 1$), CG convergences rapidly. Similar results exist for applying Krylov solvers to unsymmetric systems (e.g. GMRES) [48].

In addition to reducing the condition number of the linear system, a preconditioner should be inexpensive to compute, to store and to apply. In a parallel distributed framework the construction and the application of the preconditioner should also be easily parallelizable.

In the next sections an overview of some popular domain decomposition preconditioners is given from an algebraic perspective. Matrices arising from both finite differences and finite elements are considered. For finite differences, the linear system is fully assembled and the domain decomposition techniques correspond to matrix splittings. For finite elements, it is possible to fully assemble the matrix as with finite differences. However, this chapter focuses on the popular finite element practice of only partially assembling matrices on interfaces. That is, each processor is restricted so that it assembles matrix contributions coming only from finite elements owned by the processor. In this case, the domain decomposition techniques correspond to a splitting of the underlying mesh as opposed to splitting the matrix. From a parallel point of view they require different data structures and possibly enable different algorithmic choices. We begin by defining notation in Section 2. Then in Section 3, we present approaches that rely on overlapping sub-domains while the next section is devoted to non-overlapping techniques. In Section 5 we discuss some multi-level schemes that have *optimal* convergence properties for the solution of linear systems arising from elliptic PDEs.

2 Background and notation

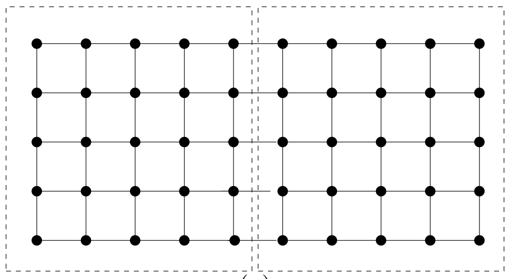
When finite differences are used, it is common to partition the mesh vertices to build sub-domains, while partitioning elements is preferred in a finite element framework. The former can be equivalently interpreted in term of matrix partitioning. The two situations are illustrated in Figure 1 for a five point finite difference stencil in two dimensions using two sub-domains. It is important to notice that when the mesh elements are partitioned, sub-domains share vertices that overlap along the interface.

2.1 Matrix partitioning

Consider a linear system

$$Ax = b \tag{1}$$

where $A = (a_{i,j})$ is a $n \times n$ nonsingular matrix having a symmetric nonzero pattern. An associated graph $G_A = (W_A, E_A)$ can be defined where the set of vertices $W_A = \{1, \ldots, n\}$ represents the



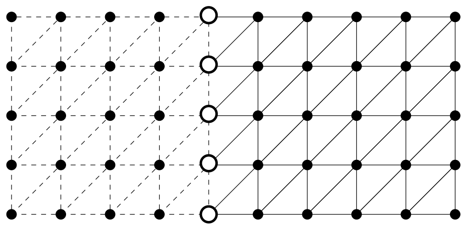


Figure 1: Partition of the domain based on vertex spiting (a) and element splitting (b). Shared vertices are indicated by a circle.

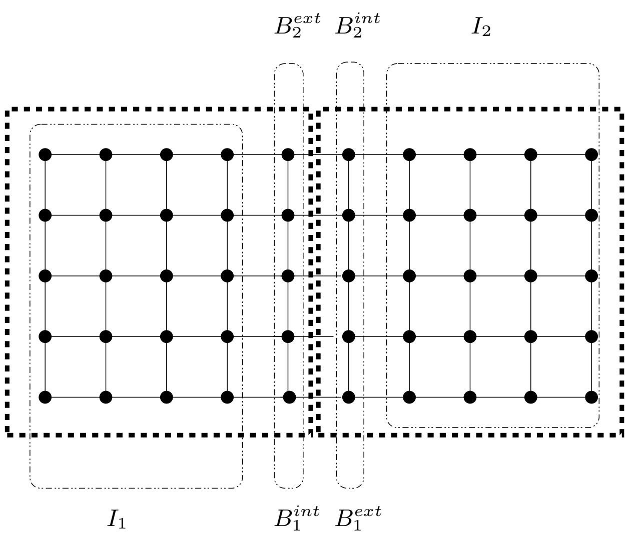


Figure 2: Illustrations of the subsets involved in a matrix partitioning associated with a five-point finite difference stencil in two-dimension.

unknowns and the set of edges $E_A = \{(i,j)\ s.t.\ a_{ij} \neq 0\}$ represents pairs of vertices that are coupled by a nonzero element in A. It is assumed that graph partitioning has been applied resulting in N non-overlapping subsets W_i^0 whose union is W_A. These subsets are referred to as sub-domains. Based on this partition, overlapping subsets are constructed as follows. Define W_i^1 as the one-overlap decomposition of W where $W_i^1 \supset W_i^0$ is obtained by including all the immediate neighboring vertices to those in W_i^0. By recursively applying this idea, the δ-overlap partition of W_A can be defined where the subsets are denoted W_i^{δ}.

In addition to sub-domains, define $B_i^{ext} = W_i^1 \backslash W_i^0$ as the set of immediate neighboring vertices to W_i^0. B_i^{ext} corresponds to the external interface of the i^{th} sub-domain in a finite difference framework. The internal interface (within a sub-domain) is denoted by B_i^{int} and is the subset of W_i^0's vertices connected to B_i^{ext}. Finally, $I_i = W_i^0 \backslash B_i^{int}$ is the set of *internal* vertices of W_i^0. To clarify this notation, Figure 2 illustrates a two-dimensional example with a five-point finite difference stencil (so that the matrix graph corresponds to the vertex connectivity in the mesh).

Corresponding to each subset W_i^0 we define a rectangular $n \times n_i$ extension matrix ${R_i^0}^T$ whose action extends by zero a vector of values associated with nodes in W_i^0 where $n_i = |W_i^0|$ (and $|\ .\ |$

denotes cardinality). The entries of ${R_i^0}^T$ are zeros and ones. For notational simplicity, we will omit the superscript *0*. That is, $R_i = R_i^0$ and $W_i = W_i^0$. Notice that R_i is a canonical restriction matrix whose action restricts a full vector in W to a vector of size n_i by choosing entries corresponding to W_i. By construction the columns of the R_i's are structurally orthogonal. Similarly for the δ-overlap partition we can define restriction operators R_i^δ and extension operators as their transpose. Notice that due to overlap, the columns of the R_i^δ's are no longer orthogonal.

With this notation, the matrices $A_i = R_i^T A R_i$ are principal sub-matrices of A. Expressed in matlab-like notation, the block rows of A associated with W_i have the following block structure (up to a column permutation) :

$$A(W_i^0, :) = \left(\begin{array}{c|c} A(W_i^0, W_i^1) & 0 \end{array} \right)$$

where

$$A(W_i^0, W_i^1) = \left(\begin{array}{cc|c} A_{I_i,I_i} & A_{I_i,B_i^{int}} & 0 \\ A_{B_i^{int},I_i} & A_{B_i^{int},B_i^{int}} & A_{B_i^{int},B_i^{ext}} \end{array} \right)$$

with

$$A_i = A(W_i^0, W_i^0) = \begin{pmatrix} A_{I_i,I_i} & A_{I_i,B_i^{int}} \\ A_{B_i^{int},I_i} & A_{B_i^{int},B_i^{int}} \end{pmatrix}. \tag{2}$$

In a parallel distributed environment each sub-domain is assigned to one processor and processor i stores $A(W_i^0, W_i^1)$. For any vectors involved in a computation processor i computes the entries associated with indices in W_i^0 possibly using entries from other processors depending on the numerical kernel. To perform a matrix-vector product, for instance, processor i receives vector entries associated with B_i^{ext} from its neighbors and sends to its neighbors vector entries associated with B_i^{int}.

2.2 Mesh partitioning

Consider a finite element mesh covering the computational domain Ω. For simplicity assume that piecewise linear elements F_k are used such that solution unknowns are associated with mesh vertices. Further, define an associated connectivity graph $G_\Omega = (W_\Omega, E_\Omega)$. The graph vertices $W_\Omega = \{1, \ldots, n_e\}$ correspond to elements in the finite element mesh. The graph edges correspond to element pairs that share at least one mesh vertex. That is, $E_\Omega = \{(i,j)\ s.t.\ F_i \cap F_j \neq \emptyset\}$. Assume that the connectivity graph has been partitioned resulting in N non-overlapping subsets Ω_i^0 whose union is W_Ω. These subsets are referred to as sub-domains and are also often referred to as substructures. Following the matrix partitioning section, the Ω_i^0 can be generalized to overlapping subsets of graph vertices. In particular, construct Ω_i^1, the one-overlap decomposition of Ω, by taking Ω_i^0 and including all graph vertices corresponding to immediate neighbors of the vertices in Ω_i^0. By recursively applying this definition, the δ-layer overlap of W_Ω is constructed and the sub-domains are denoted Ω_i^δ.

Corresponding to each sub-domain Ω_i^0 we define a rectangular extension matrix ${\mathcal{R}_i^0}^T$ whose action extends by zero a vector of values defined at *mesh* vertices associated with the finite elements contained in Ω_i^0. The entries of ${\mathcal{R}_i^0}^T$ are still zeros and ones. For simplicity, we once again omit the *0* superscripts and define $\mathcal{R}_i = \mathcal{R}_i^0$ and $\Omega_i = \Omega_i^0$. Notice that the columns of a given $\mathcal{R}_k$ are still orthogonal, but that between the different $\mathcal{R}_i$'s some columns are no longer orthogonal. This is due to the fact that some mesh vertices overlap even though the graph vertices defined by Ω_i are non-overlapping (shared mesh vertices see Figure 1 (b)). Let Γ_i be the set of all mesh vertices belonging to the interface of Ω_i; that is mesh vertices lying on $\partial\Omega_i \backslash \partial\Omega$. Similarly, let $\mathcal{I}_i$ be the set of all remaining mesh vertices within the sub-domain Ω_i (i.e. interior vertices). Considering only the discrete matrix contributions arising from finite elements in Ω_i gives rise to the following local discretization matrix :

$$\mathcal{A}_i = \begin{pmatrix} \mathcal{A}_{\mathcal{I}_i\mathcal{I}_i} & \mathcal{A}_{\mathcal{I}_i\Gamma_i} \\ \mathcal{A}_{\Gamma_i\mathcal{I}_i} & \mathcal{A}_{\Gamma_i\Gamma_i} \end{pmatrix} \tag{3}$$

where interior vertices have been ordered first. The matrix $\mathcal{A}_i$ corresponds to the discretization of the PDE on Ω_i with Neumann boundary condition on Γ_i and the one-one block $\mathcal{A}_{\mathcal{I}_i\mathcal{I}_i}$ corresponds to the discretization with homogeneous Dirichlet conditions on Γ_i. The completely assembled discretization matrix is obtained by summing the contributions over the substructures/sub-domains :

$$A = \sum_{i=1}^{N} \mathcal{R}_i^T \mathcal{A}_i \mathcal{R}_i. \tag{4}$$

In a parallel distributed environment each sub-domain is assigned to one processor and typically processor i stores $\mathcal{A}_i$. A matrix-vector product is performed in two steps. First a local matrix-vector product involving $\mathcal{A}_i$ is performed followed by a communication step to assemble the results along the interface Γ_i.

For the δ-overlap partition we can define a corresponding restriction operator $\mathcal{R}_i^\delta$ which maps mesh vertices in Ω to the subset of mesh vertices associated with finite elements contained in Ω_i^δ. Corresponding definitions of Γ_i^δ and $\mathcal{I}_i^\delta$ follow naturally as the boundary and interior mesh vertices associated with finite elements in Ω_i^δ. The discretization matrix on Ω_i^δ has a similar structure to the one given by (3) and is written as

$$\mathcal{A}_i^\delta = \begin{pmatrix} \mathcal{A}_{\mathcal{I}_i^\delta\mathcal{I}_i^\delta} & \mathcal{A}_{\mathcal{I}_i^\delta\Gamma_i^\delta} \\ \mathcal{A}_{\Gamma_i^\delta\mathcal{I}_i^\delta} & \mathcal{A}_{\Gamma_i^\delta\Gamma_i^\delta} \end{pmatrix}. \tag{5}$$

3 Overlapping domain decomposition approaches

The domain decomposition methods based on overlapping sub-domains are most often referred to as Schwarz methods due to the pioneering work of Schwarz in 1870 [51]. This work was not intended as a numerical algorithm but was instead developed to show the existence of the elliptic problem solution on a complex geometry formed by overlapping two simple geometries where solutions are known. With the advent of parallel computing this basic technique, known as the alternating Schwarz method, has motivated considerable research activity. In this section, we do not intend to give an exhaustive presentation of all work devoted to Schwarz methods. Only additive variants that are well-suited to straightforward parallel implementation are considered. Within additive variants, computations on all sub-domains are performed simultaneously while multiplicative variants require some sub-domain calculations to wait for results from other sub-domains. The multiplicative versions often have connections to block Gauss-Seidel methods while the additive variants correspond more closely to block Jacobi methods. We do not further pursue this description but refer the interested reader to [56]. It is, however, important to recognize that some multiplicative version can run efficiently on parallel computers though they are typically complicated to properly implement. To load balance these variants, sub-domains must first be colored and then assigned to processors such that each processor is responsible for several sub-domains of different colors.

3.1 Additive Schwarz preconditioners

Although not formally a domain decomposition preconditioner, the simplest method based on the matrix decomposition introduced in the previous section is the block Jacobi preconditioner given

	# sub-domains			
δ	2	4	8	16
1	7	8	10	12
2	5	7	7	8
4	3	5	6	6

Table 1: Number of iterations using $\mathcal{M}_{AS}^{\delta}$ for solving a Poisson problem on the mesh depicted in Figure 3.

by

$$M_{BJ} = \sum_{i=1}^{N} R_i^T A_i^{-1} R_i.$$

This method is fully parallel as each block of the preconditioner acts only on internal unknowns stored by each processor. In a mesh partitioning framework the overlap among interface unknowns significantly complicates the block Jacobi implementation and so the additive Schwarz preconditioner is preferred.

If a matrix decomposition is applied, the classical additive Schwarz preconditioner with δ-overlap is defined as follows

$$M_{AS}^{\delta} = \sum_{i=1}^{N} \left(R_i^{\delta}\right)^T \left(A_i^{\delta}\right)^{-1} R_i^{\delta}. \tag{6}$$

The corresponding mesh partitioned additive Schwarz preconditioner is given by

$$\mathcal{M}_{AS}^{\delta} = \sum_{i=1}^{N} \left(\mathcal{R}_i^{\delta-1}\right)^T \left(\mathcal{A}_{\mathcal{I}_i\mathcal{I}_i}^{\delta}\right)^{-1} \mathcal{R}_i^{\delta-1}. \tag{7}$$

Here the δ-overlap is defined in terms of finite element decompositions. The preconditioner and the $\mathcal{R}_i^{\delta-1}$ operators, however, act on mesh vertices corresponding to the sub-meshes associated with the finite element decomposition. Notice that M_{AS}^0 reduces to M_{BJ} and that $\mathcal{M}_{AS}^0$ is not defined. In both cases, the preconditioner is symmetric (or symmetric positive definite) if the original system, A, is symmetric (or symmetric positive definite).

Parallel implementation of this preconditioner requires a factorization of a Dirichlet problem on each processor in the setup phase. Each invocation of the preconditioner requires two neighbor-neighbor communications. The first corresponds to obtaining values within overlapping regions associated with the restriction operator. The second corresponds to summing the results of the backward/forward substitution via the extension operator.

In general, larger overlap usually leads to faster convergence up to a certain point where increasing the overlap does not further improve the convergence rate. Unfortunately, larger overlap implies greater communication and computation requirements. For example, expanding a cubic sub-domain consisting of d^3 vertices by just one in each dimension gives rise to approximately $6d^2$ additional vertices. For $d = 20$, this corresponds to a 30% increase in vertices which can lead to significantly more than a 30% growth in the computation when direct factorizations are used on sub-domains. Typical convergence behavior is illustrated in Table 1 where Krylov iterations are given as a function of the number of sub-domains and overlap for a two-dimensional Laplacian. This problem is presented in [56] and is solved on the unstructured airfoil mesh depicted in Figure 3.

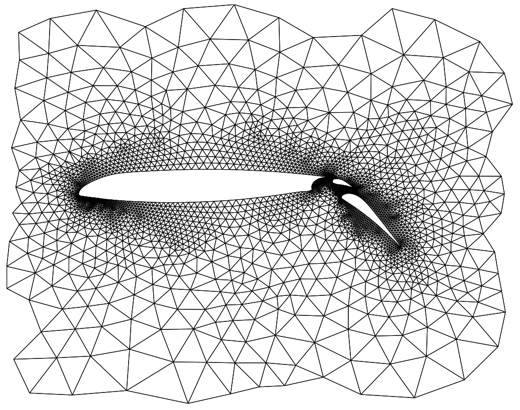


Figure 3: Unstructured mesh used for a piecewise linear finite element discretization.

3.2 Restricted additive Schwarz preconditioner

A variant of the classical additive Schwarz method is introduced in [16] which avoids one communication step when applying the preconditioner. This variant is referred to as Restricted Additive Schwarz (RAS). The corresponding preconditioner is defined in the matrix partitioning context by

$$M^{\delta}_{RAS} = \sum_{i=1}^{N} R_i^T \left(A_i^{\delta}\right)^{-1} R_i^{\delta}. \tag{8}$$

Notice that even within overlapping regions each unknown is updated by only one sub-domain while contributions from all overlapping sub-domains are summed in the classical additive Schwarz method. This variant does not have a natural counterpart in a mesh partitioning framework that by construction has overlapping sets of vertices. Consequently, the closest mesh partitioning counterpart solves a Dirichlet problem on a large sub-domain but considers the solution only within the substructure. That is,

$$\mathcal{M}^{\delta}_{RAS} = \sum_{i=1}^{N} \mathcal{R}_i^T \left(\mathcal{A}^{\delta}_{\mathcal{I}_i\mathcal{I}_i}\right)^{-1} \mathcal{R}_i^{\delta-1}. \tag{9}$$

The so-called additive Schwarz with harmonic extension is obtained by switching the restriction and extension matrices [16]. In the matrix partitioning context, this is defined by :

$$M^{\delta}_{ASH} = \sum_{i=1}^{N} \left(R_i^{\delta}\right)^T \left(A_i^{\delta}\right)^{-1} R_i. \tag{10}$$

The main drawback to $M^{\delta}_{RAS}, \mathcal{M}^{\delta}_{RAS},$ and M^{δ}_{ASH} is that they give rise to unsymmetric preconditioners even when A is symmetric. The restricted additive Schwarz with harmonic extension (RASHO) has been proposed to address this [14]. It gives rise to a symmetric preconditioner when applied to a symmetric problem but requires a more elaborated procedure to define the overlap.

Surprisingly, M_{RAS} often converges faster than M_{AS} and only requires half the communication making it frequently superior on parallel distributed computers. Of course it might not be suitable for symmetric positive definite problems as it requires a non-symmetric Krylov solver. These features are illustrated in Table 2 presented in [16]. The results correspond to the solution of a linear system in an inexact Newton scheme for the steady-state transonic three-dimensional

	M_{AS}			M_{RAS}		
# domains	# iter	CPU	% comm	# iter	CPU	% comm
4	131	24 s	5 %	100	19 s	3 %
8	140	15 s	8 %	105	12 s	6 %
16	145	9 s	16 %	106	7 s	13 %

Table 2: Number of Krylov iterations, CPU time in second and percentage of time spent in communication on a IBM SP platform - $\delta = 1$.

compressible Euler equations discretized on a fully unstructured mesh. We refer to [16] and the references therein for a complete description of this test problem.

All of the above techniques make use of a matrix inverse (i.e. a direct solver or an exact factorization) of a local Dirichlet matrix. In practice, it is common to replace this with an incomplete factorization [48, 47] or an approximate inverse [5, 6, 21, 32, 39]. This is particularly important for three-dimensional problems where exact factorizations are often expensive in terms of both memory and floating-point operations. While this usually slightly deteriorates the convergence rate, it can lead to a faster method due to the fact that each iteration is less expensive. The numerical effect of the inexact local solves is illustrated in Table 3 for the solution of a linear system arising from the discretization of a convection-diffusion problem in two-dimensions. Each sub-domain consists of a 128×128 mesh. Thus, as the number of processors (or sub-domains) increases, the size of the entire problem increases. An incomplete LU factorization with thresholding [47] is used to approximately solve the local Dirichlet problems and MA41 [2, 3] provides the exact factorizations. Convergence is declared when the initial residual is reduced by seven orders of magnitude. For this example, the size of the exact factors are about twice as large as the size of the inexact factors (e.g. $8 \cdot 10^5$ vs. $5 \cdot 10^5$ non-zeros on 256 processors). While exact sub-domain

	Exact		Inexact	
# domains	# iter	Time	# iter	Time
8 $(= 2 \times 4)$	65	1.9	74	2.2
64 $(= 8 \times 8)$	154	9.0	193	12.0
256 $(= 16 \times 16)$	295	30.0	360	39.0

Table 3: Number of full-GMRES iterations and parallel elapsed time in seconds using M_{RAS} with exact and inexact local solve - $\delta = 1$.

factorization is frequently advantageous in two dimensions (and for relatively thin structures in three dimensions), incomplete factorizations are often needed in three dimensions. Table 4 illustrates this on a diffusion-dominated scalar convection-diffusion problem. Each sub-domain consists of a $10 \times 10 \times 10$ mesh. SuperLU [41] provides the direct factorizations and an ILU(0) algorithm is used for the inexact solves. Convergence is declared when the initial residual is reduced by ten orders of magnitude. As in two dimensions, fewer iterations are required with the exact solves. In this case, however, incomplete factorization leads to significantly faster overall solve times. This is due to the relatively large number of nonzeros in the exact factors. It should also be noted that sub-domains far from the boundary (on the sixty-four processor job) have 60% more vertices due to overlapping. We point out that results displayed in Table 3 and Table 4 are run on different hardware using different sparse direct solvers and different thresholds for the stopping criterion. The computing plateforms are based on Power PC G5 2Ghz for the former and on Dual 3.6 GHz Intel EM64T processors for the latter. Consequently the run times between them should not be

	Exact		Inexact	
# domains	# iter	Time	# iter	Time
8 (= $2 \times 2 \times 2$)	20	1.7	39	0.05
64 (= $4 \times 4 \times 4$)	40	6.1	76	0.14

Table 4: Number of GMRES iterations and parallel elapsed time in seconds using M_{RAS} with exact and inexact local solve - $\delta = 1$. GMRES restarted after 30 iterations

compared. Finally, we mention that these techniques based on Schwarz variants are available in several large parallel software libraries see for instance [4, 35, 37, 42, 60].

4 Non-overlapping domain decomposition approaches

In this section, methods based on non-overlapping regions are described. Such domain decomposition algorithms are often referred to as sub-structuring schemes. This terminology comes from the structural mechanics discipline where non-overlapping ideas were first developed. In this early work the primary focus was on direct solvers. Associating one frontal matrix with each sub-domain allows for coarse grain multiple front direct solvers [23]. Motivated by parallel distributed computing and the potential for coarse grain parallelism, considerable research activity developed around iterative domain decomposition schemes. A very large number of methods have been proposed and we cannot cover all of them. Therefore, the main highlights are surveyed.

The governing idea behind sub-structuring or Schur complement methods is to split the unknowns in two subsets. This induces the following block reordered linear system :

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix} \tag{11}$$

where x_Γ contains all unknowns associated with sub-domain interfaces and x_I contains the remaining unknowns associated with sub-domain interiors. The matrix A_{II} is block diagonal where each block corresponds to a sub-domain interior. Eliminating x_I from the second block row of equation (11) leads to the reduced system

$$Sx_\Gamma = f_\Gamma - A_{\Gamma I} A_{II}^{-1} f_I, \text{ where } S = A_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} A_{I\Gamma} \tag{12}$$

and S is referred to as the *Schur complement matrix*. This reformulation leads to a general strategy for solving (11). Specifically, an iterative method can be applied to (12). Once x_Γ is determined, x_I can be computed with one additional solve on the sub-domain interiors. Further, when A is symmetric positive definite (SPD), the matrix S inherits this property and so a conjugate gradient method can be employed.

The iterative solution to (12) requires a procedure to apply a vector to the Schur complement matrix as well as a suitable preconditioner. In most iterative schemes, the explicit formation of the Schur complement matrix is avoided due to the potentially large cost in both time and storage. In the remainder of this section we explore Schur complement representations that are suitable for parallel computers. Starting in Section 4.1.1, the question of preconditioning is addressed.

Consider the matrix partition situation depicted earlier in Figure 1. The original linear system has the form

$$\left(\begin{array}{cc|cc} A_{I_1,I_1} & A_{I_1,B_1^{int}} & 0 & 0 \\ A_{B_1^{int},I_1} & A_{B_1^{int},B_1^{int}} & A_{B_1^{int},B_2^{ext}} & 0 \\ \hline 0 & A_{B_2^{int},B_2^{ext}} & A_{B_2^{int},B_2^{int}} & A_{B_2^{int},I_2} \\ 0 & 0 & A_{I_2,B_2^{int}} & A_{I_2,I_2} \end{array}\right) \begin{pmatrix} x_{I_1} \\ x_{B_1^{int}} \\ x_{B_2^{int}} \\ x_{I_2} \end{pmatrix} = \begin{pmatrix} b_{I_1} \\ b_{B_1^{int}} \\ b_{B_2^{int}} \\ b_{I_2} \end{pmatrix}. \tag{13}$$

Eliminating internal variables reduces to a linear system defined on the interfaces (this is sometimes called *condensation*) :

$$\left(\begin{array}{c|c} S_1 & A_{B_1^{int},B_1^{ext}} \\ \hline A_{B_2^{int},B_2^{ext}} & S_2 \end{array}\right)\begin{pmatrix} x_{B_1^{int}} \\ x_{B_2^{int}} \end{pmatrix} = \begin{pmatrix} f_{B_1^{int}} \\ f_{B_2^{int}} \end{pmatrix}, \qquad (14)$$

where $S_i = A_{B_i^{int},B_i^{int}} - A_{B_i^{int},I_i} A_{I_i,I_i}^{-1} A_{I_i,B_i^{int}}$ and $f_{B_i^{int}} = b_{B_i^{int}} - A_{B_i^{int},I_i} A_{I_i,I_i}^{-1} b_{I_i}$. This generalizes to a N domain case where the condensed linear system is

$$\begin{pmatrix} S_1 & E_{1,2} & \cdots & E_{1,N} \\ E_{2,1} & S_2 & \cdots & E_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ E_{N,1} & E_{N,1} & \cdots & S_N \end{pmatrix}\begin{pmatrix} x_{B_1^{int}} \\ x_{B_2^{int}} \\ \vdots \\ x_{B_N^{int}} \end{pmatrix} = \begin{pmatrix} f_{B_1^{int}} \\ f_{B_2^{int}} \\ \vdots \\ f_{B_N^{int}} \end{pmatrix}.$$

and the blocks $E_{i,k}$ correspond to rows of A associated with $x_{B_i^{int}}$ and columns associated with $x_{B_j^{int}}$. While the S_i's are generally dense, processor i can perform a matrix-vector product with S_i by solving an A_{I_i,I_i} problem.

The main drawback of this approach is that the interface between two adjacent sub-domain interiors has two layers (see Figure 1). This means that the corresponding Schur complement system is twice as large is it needs to be. To reduce the Schur complement size, the two internal interfaces need to be effectively merged so that there is only one layer between internal sub-domains. Unfortunately, this complicates parallel implementation and load balancing decisions. Which processor should own a particular interface or should the interface be shared? While merging interfaces in a matrix partitioning framework is somewhat involved, it is relatively transparent from a mesh partitioning framework. Not surprisingly, the structural analysis finite element community has been heavily involved with these techniques. Not only is their definition fairly natural in a finite element framework but their implementation can preserve data structures and concepts already present in large engineering software packages.

Let Γ denote the entire interface defined by $\Gamma = \cup\, \Gamma_i$ where $\Gamma_i = \partial\Omega_i \backslash \partial\Omega$ as in Section 2. As interior unknowns are no longer considered, new restriction operators must be defined as follows. Let $\mathcal{R}_{\Gamma_i} : \Gamma \rightarrow \Gamma_i$ be the canonical point-wise restriction which maps full vectors defined on Γ into vectors defined on Γ_i. Analogous to (4), the Schur complement matrix (11) can be written as the sum of elementary matrices

$$S = \sum_{i=1}^{N} \mathcal{R}_{\Gamma_i}^T \mathcal{S}_i \mathcal{R}_{\Gamma_i} \qquad (15)$$

where

$$\mathcal{S}_i = \mathcal{A}_{\Gamma_i\Gamma_i} - \mathcal{A}_{\Gamma_i\mathcal{I}_i} \mathcal{A}_{\mathcal{I}_i\mathcal{I}_i}^{-1} \mathcal{A}_{\mathcal{I}_i\Gamma_i} \qquad (16)$$

is a local Schur complement and is defined in terms of sub-matrices from the local Neumann matrix $\mathcal{A}_i$ given by (3). Notice that this form of the Schur complement has only one layer of interface unknowns between sub-domains and allows for a straight-forward parallel implementation.

4.1 Schur complement Preconditioning

While the Schur complement system is significantly better conditioned than the original matrix A, it is important to consider further preconditioning when employing a Krylov method. It is well-known, for example, that $\kappa(A) = \mathcal{O}(h^{-2})$ when A corresponds to a standard discretization (e.g. piecewise linear finite elements) of the Laplace operator on a mesh with spacing h between the grid points. Using two non-overlapping sub-domains effectively reduces the condition number of the

Schur complement matrix to $\kappa(S) = \mathcal{O}(h^{-1})$. While improved, preconditioning can significantly lower this condition number further.

One of the difficulties associated with preconditioning non-overlapping methods is that the Schur complement matrix entries are not generally available. This even complicates the implementation of block-Jacobi methods. One case where the Jacobi preconditioner is relatively straightforward is the matrix partitioning situation. Consider again the example corresponding to Figure 1 and equation (13) where the inverse of the S_i's are needed in a block-Jacobi scheme. These can be obtained using the following factorization for the local Dirichlet matrix (2) :

$$A_i = \begin{pmatrix} Id_{I_i} & 0 \\ A_{B_i^{int},I_i} A_{I_i,I_i}^{-1} & Id_{B_i^{int}} \end{pmatrix} \begin{pmatrix} A_{I_i,I_i} & 0 \\ 0 & S_i \end{pmatrix} \begin{pmatrix} Id_{I_i} & A_{I_i,I_i}^{-1} A_{I_i,B_i^{int}} \\ 0 & Id_{B_i^{int}} \end{pmatrix} \tag{17}$$

where Id_X is the identity matrix of dimension $|X|$. Inverting both sides gives

$$A_i^{-1} = \begin{pmatrix} Id_{I_i} & -A_{I_i,I_i}^{-1} A_{I_i,B_i^{int}} \\ 0 & Id_{B_i^{int}} \end{pmatrix} \begin{pmatrix} A_{I_i,I_i}^{-1} & 0 \\ 0 & S_i^{-1} \end{pmatrix} \begin{pmatrix} Id_{I_i} & 0 \\ -A_{B_i^{int},I_i} A_{I_i,I_i}^{-1} & Id_{B_i^{int}} \end{pmatrix}, \tag{18}$$

and so

$$S_i^{-1} = \begin{pmatrix} 0 & Id_{B_i^{int}} \end{pmatrix} A_i^{-1} \begin{pmatrix} 0 \\ Id_{B_i^{int}} \end{pmatrix}.$$

The implementation of this scheme requires the factorization of A_{I_i,I_i} to implicitly perform the matrix-vector product involving the Schur complement and the factorization of the matrices A_i to implicitly apply the block-Jacobi preconditioner. We mention that closely related methods are described in [49] in the framework of general linear system solutions.

4.1.1 The tangential preconditioner

The simplest preconditioner that one can consider for Schur complement systems are modified forms of $A_{\Gamma\Gamma}$. The main problem with using $A_{\Gamma\Gamma}$ directly is that it can be much more diagonally dominant than S. This is due to the diagonal entries of $A_{\Gamma\Gamma}$ which also include contributions from domain interiors which are not balanced by off-diagonal entries (i.e. the entries in $A_{\Gamma I}$). To avoid this problem, preconditioning by inverting the so-called tangential component of $A_{\Gamma\Gamma}$ can be used instead. The off-diagonals of the tangential operator are identical to $A_{\Gamma\Gamma}$. The diagonal, however, includes only contributions that lie within the interface. This approach is easy to implement. No special geometric considerations are needed as the treatment of faces, edges, and cross points is identical. For numerical experiments with this type of preconditioner and its variants we refer to [15, 17].

4.1.2 The probing technique

First introduced to construct Jacobians [22], probing approximates interface coupling by a matrix with a specified sparsity pattern using matrix-vector products between S and a few carefully chosen vectors. Motivated by the observation that interface entries in S decay rapidly from the diagonal [29], a band matrix pattern is often used. The basic idea is illustrated by considering the reconstruction of a tridiagonal matrix. Specifically, all coefficients of a tridiagonal can be obtained using $p^{(0)} = (0,0,1,0,0,1,0,0,\cdots)^T$, $p^{(1)} = (1,0,0,1,0,0,\cdots)^T$, and $p^{(2)} = (0,1,0,0,1,0,0,\cdots)^T$:

$$\begin{pmatrix} c_{11} & c_{12} & & & & \\ c_{21} & c_{22} & c_{23} & & & \\ & c_{32} & c_{33} & c_{34} & & \\ & & c_{43} & c_{44} & c_{45} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & 0 \\ c_{21} & c_{22} & c_{23} \\ c_{34} & c_{32} & c_{33} \\ c_{44} & c_{45} & c_{43} \\ \vdots & \vdots & \vdots \end{pmatrix}.$$

While the Schur complement is not tridiagonal, tridiagonal probing often gives a good approximation to a single interface in two dimensions (For SPD systems, [18] discusses probing to preserve SPD properties. Additionally, a larger band matrix is needed in three dimensions).

For multiple domains, it is natural to construct separate approximations for each interface between adjacent sub-domains. This gives rise to a block diagonal preconditioner. The main issue is how to probe different interfaces. If *all* interfaces are probed simultaneously, interference between interfaces can cloud the approximation. Instead, it is better to use coloring ideas. An example from [20] is given in Figure 4. Composite probe vectors are defined using individual edge probes. When constructing a tridiagonal, for instance, three composite vectors are defined using $p^{(0)}, p^{(1)}$, and $p^{(2)}$ on all vertical edges with 0's on all horizontal edges. Similarly, another three composite vectors are constructed for the horizontal edges. While this works well for isotropic

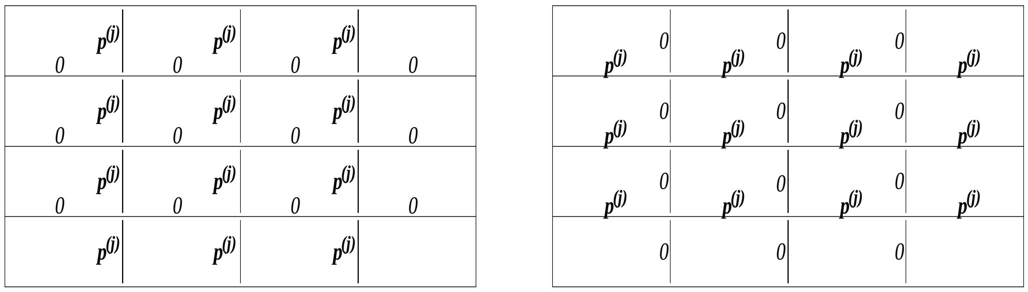


Figure 4: Vertical/horizontal probe vectors giving rise to preconditioner M^{vh}.

problems, interference can still occur for anisotropic problems. Figure 5 given in [27] illustrates a multi-color approach where vertical probes are split into red/black colors; the same is done for horizontal probes. The approximation on one edge is not distorted by other edges as there is no

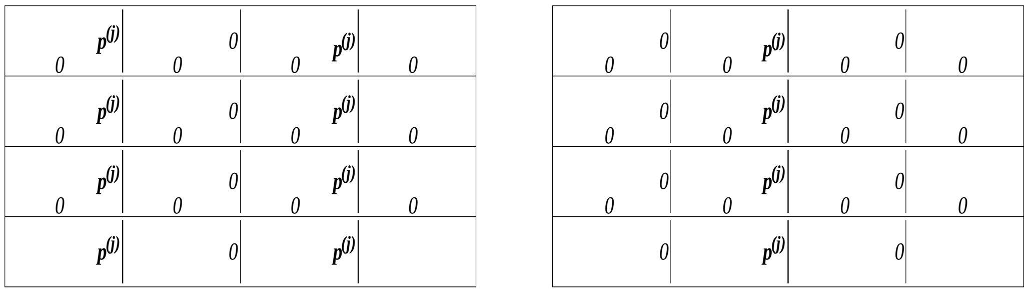


Figure 5: Red/black vertical probe vectors giving rise to preconditioner M^{rb}.

Schur complement coupling between different edges within the same color. Of course adding colors

probing	# sub-domains		
variant	4×4	8×8	16×16
M^{rb}	19	42	69
M^{vh}	36	115	310

Table 5: Krylov iterations to solve $10^{-3}u_{xx} + u_{yy} = f$ on the unit square.

increases the setup costs. For the examples illustrated in Figure 4 and Figure 5 the cost is six versus twelve matrix-vector products. Table 5 gives the number of Krylov iterations required for convergence on an anisotropic problem discretized with a five point finite difference stencil. The size of the sub-domain is kept constant and the number of domains is varied from sixteen to two hundred and fifty-six. It can be seen that the M^{rb} convergence rate is faster and that this rate deteriorates more slowly than M^{vh} as the number of domains increases. In this case, the additional setup cost avoids interference that ruins the simpler probing approximation. The resulting scheme is much more effective. We refer to [27] for a detailed description of this probing variant.

While regular grids are easiest, nonuniform grids can use similar coloring ideas though the implementation is more complex. More about coloring/probing can be found in [52]. On unstructured three dimensional meshes, probing is not straightforward to implement. For example, more complex graph ideas are needed to generate a target sparsity pattern for the probe approximation.

4.1.3 The Neumann-Dirichlet preconditioner

When a symmetric constant coefficient problem is sub-divided into two non-overlapping domains such that the sub-domains are exact mirror images, it follows that the Schur complement contribution from both the left and right domains is identical. That is, $\mathcal{S}_1 = \mathcal{S}_2$. Consequently, the inverse of either $\mathcal{S}_1$ or $\mathcal{S}_2$ are ideal preconditioners as the preconditioned linear system is well-conditioned, e.g. $S\mathcal{S}_1^{-1} = 2I$. A factorization similar to (17) can be applied to the local Neumann problem (3) on Ω_1 :

$$\mathcal{A}_1 = \begin{pmatrix} Id_{\mathcal{I}_1} & 0 \\ \mathcal{A}_{\mathcal{I}_1\Gamma_1} A_{\mathcal{I}_1\mathcal{I}_1}^{-1} & Id_{\Gamma_1} \end{pmatrix} \begin{pmatrix} A_{\mathcal{I}_1\mathcal{I}_1} & 0 \\ 0 & \mathcal{S}_1 \end{pmatrix} \begin{pmatrix} Id_{\mathcal{I}_1} & A_{\mathcal{I}_1\mathcal{I}_1} A_{\Gamma_1\mathcal{I}_1} \\ 0 & Id_{\Gamma_1} \end{pmatrix}$$

to obtain

$$\mathcal{S}_1^{-1} = \begin{pmatrix} 0 & Id_{\Gamma_1} \end{pmatrix} (\mathcal{A}_1)^{-1} \begin{pmatrix} 0 \\ Id_{\Gamma_1} \end{pmatrix}.$$

In general, most problems will not have mirror image sub-domains and so $\mathcal{S}_1 \neq \mathcal{S}_2$. However, if the underlying system within the two sub-domains is similar, the inverse of $\mathcal{S}_1$ should make an excellent preconditioner. The corresponding linear system is

$$\left(I + \mathcal{S}_1^{-1}\mathcal{S}_2\right) x_{\Gamma_1} = (\mathcal{S}_1)^{-1} b_{\Gamma_1}$$

so that each Krylov iteration solves a Dirichlet problem on Ω_2 (to apply $\mathcal{S}_2$) followed by a Neumann problem on Ω_1 to invert $\mathcal{S}_1$. The Neumann-Dirichlet preconditioner was introduced in [7].

Generalization of the Neumann-Dirichlet preconditioner to multiple domains can be done easily when a red-black coloring of sub-domains is possible such that sub-domains of the same color do not share an interface. In this case, the preconditioner is just the sum of the inverses corresponding to the black sub-domains:

$$S = \sum_{i \in B} \mathcal{R}_{\Gamma_i}^T (\mathcal{S}_i)^{-1} \mathcal{R}_{\Gamma_i} \qquad (19)$$

where B corresponds to the set of all black sub-domains.

4.1.4 The Neumann-Neumann preconditioner

Similar to the Neumann-Dirichlet method, the Neumann-Neumann preconditioner implicitly relies on the similarity of the Schur complement contribution from different sub-domains. In the Neumann-Neumann approach the preconditioner is simply the weighted sum of the inverse of the $\mathcal{S}_i$. In the two mirror image sub-domains case,

$$S^{-1} = \frac{1}{2}\left(\mathcal{S}_1^{-1} + \mathcal{S}_2^{-1}\right).$$

This motivates using the following preconditioner with multiple sub-domains :

$$M_{NN} = \sum_{i=1}^{N} \mathcal{R}_{\Gamma_i}^T D_i \mathcal{S}_i^{-1} D_i \mathcal{R}_{\Gamma_i} \qquad (20)$$

where the D_i are diagonal weighting matrices corresponding to a partition of unity. That is,

$$\sum_{i=1}^{N} \mathcal{R}_{\Gamma_i}^T D_i \mathcal{R}_{\Gamma_i} = Id_\Gamma.$$

The simplest choice for D_i is the diagonal matrix with entries equal to the inverse of the number of sub-domains to which an unknown belongs. The Neumann-Neumann preconditioner was first discussed in [8] and further studied in [57] where different choices for weight matrices are discussed. It should be noted that the matrices $\mathcal{S}_i$ can be singular for internal sub-domains because they correspond to pure Neumann problems. The Moore-Penrose pseudo-inverse is often used for the inverse local Schur complements in (20) but other choices are possible such as inverting $\mathcal{A}_i + \epsilon I$ where ϵ is a small shift.

The Neumann-Neumann preconditioner is very attractive from a a parallel implementation point of view. In particular, all interface unknowns are treated similarly and no distinction is required to differentiate between unknowns on faces, edges, or cross points.

5 Multi-level techniques

The solution of an elliptic problem at any point depends on the solution of the forcing function from all other points within the domain. That is, the associated Green's functions are global. The main difficulty with all of the methods presented so far is that they are basically local. One application of the preconditioner basically mixes information between neighboring sub-domains. The implication is that the corresponding convergence rate of these methods must degrade as the number of sub-domains increases. By way of example, the condition number of the preconditioned linear system using the Neumann-Neumann method can be bounded by

$$\frac{C(1+\log(H/h))^2}{H^2}$$

where h is the mesh spacing, H is the sub-domain spacing (i.e. approximate diameter of a sub-domain), and C is a constant independent of mesh spacing and sub-domain spacing (see [46]). As the number of processors/sub-domains increases, H^{-2} grows. This increase in the the condition number bound is consistent with an observed rise in the number of Krylov iterations.

To rectify this situation, a number of algorithms have been proposed. The basic idea is to use methods of the previous section in conjunction with some type of coarse mesh system. The goal is to construct preconditioners with optimal convergence rates for given classes of elliptic problems.

In the domain decomposition context, optimal typically means that the condition number of the associated preconditioned systems can be bounded independent of the number of sub-domains and either independent or logarithmically dependent on the size of the sub-domains. The coarse problem usually corresponds to a projection of the original operator onto a space that captures the lower resolution behavior of the linear system. The term *coarse* system is used because its dimension is generally significantly lower than the original. When coarse level ideas are incorporated into domain decomposition, there are strong ties to multigrid methods (see for example [64]). In fact, many of the schemes from the previous section can be interpreted as smoothers within a multigrid method. The coarse spaces used in multigrid methods and domain decomposition methods also have many connections. In fact, some ideas presented in the next sub-section have their roots from the algebraic multigrid community.

There are many trade-offs between multigrid and domain decomposition algorithms. Multigrid methods usually employ relatively inexpensive smoothers in conjunction with a full hierarchy of coarse resolution systems. Domain decomposition frequently uses more expensive smoothers (e.g. approximate sub-domain solves), but often with only one coarse matrix system. This coarse system is normally much smaller than the original (e.g. the number of nodes is approximately equal to the number of sub-domains). In this case, there is no need to address complexities associated with grid hierarchies nor is it necessary to consider recursive application of the coarsening idea. This is convenient for approaches based on partially assembled finite element matrices which are not easily defined on coarse meshes. Obviously when the coarse system is still large, it is necessary to consider some approximate solution or recursive technique. More information on multigrid methods can be found in [13, 33, 59, 63]. Public domain algebraic multigrid software can be found in [26, 34, 36, 44].

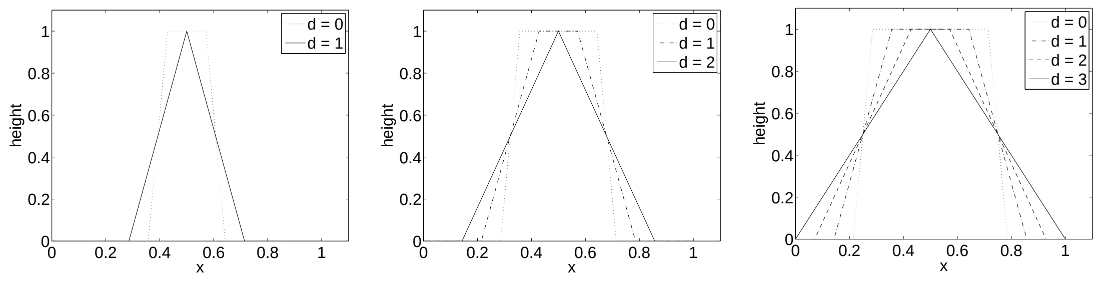
5.1 Two-level overlapping methods

A simple two-level Schwarz method is obtained by adding an additional term to M_{AS}. Namely,

$$M_{AS2} = M_{AS} + R_0^T A_0^{-1} R_0 \tag{21}$$

where R_0 is a projection on to a coarse space V_0 and $A_0 = R_0 A R_0^T$ or A_0 is obtained by rediscretizing the original problem on a coarse mesh. It should be noted that (21) is additive in that the coarse grid correction can occur simultaneously with the M_{AS} contributions. It is also possible to perform the coarse grid correction in a multiplicative fashion by first applying M_{AS}, computing a residual, applying $R_0^T A_0^{-1} R_0$ to the residual and adding the result to the solution obtained by applying M_{AS}. This is not further discussed in this chapter.

Many definitions for the coarse space and the corresponding R_0 operator have been considered (and are still being developed). These can be based on geometric ideas (e.g. linear interpolation for R_0^T), finite element ideas (e.g. finite element basis functions corresponding to a coarse mesh), or algebraic ideas (e.g. using the matrix coefficients to define basis functions with minimal A-norm in the SPD case). Again, there are trade-offs in these different approaches. Geometric schemes are somewhat complicated to implement and are often tied to the resulting application code. Applications with complex geometric features can be particularly challenging to develop. Additionally, they may have robustness issues for problems with highly heterogeneous behavior as the interpolation and restriction do not use material, PDE, or discretization properties. While finite element approaches are more closely tied to the discrete system, they require a more explicit notion of a coarse mesh which makes sense in a finite element context (e.g. all coarse elements are convex). This can be particularly difficult when irregular boundaries are present. Algebraic methods have an advantage in that they do not require an explicit mesh and by using a matrix they have *indirect* access to material, PDE, and discretization properties. Unfortunately, it is



sub-domain width = 3 *sub-domain width = 5* *sub-domain width = 7*

Figure 6: Smoothed Aggregation basis function support for different sub-domain widths and polynomial degrees, d.

not always computationally easy to deduce basic properties of an operator based only on matrix coefficients. For example, bilinear quadrilateral finite elements tend to smear anisotropic behavior among matrix coefficients making this difficult to detect.

The literature on generating coarse spaces is quite extensive. Here, we simply mention one possible algebraic technique that has proved successful in several problem domains and is relatively straight-forward to implement in parallel. This technique is called smoothed aggregation and was initially developed in a multigrid context [12, 62]. To simplify the exposition, we describe smoothed aggregation as it is applied to symmetric scalar elliptic PDE operators. The basic idea is to first generate a tentative interpolation (or prolongation) operator that preserves constants. In a more general setting, the tentative prolongator should preserve the so-called near null space of an operator. This corresponds to preserving rigid body motion in elasticity. Specifically in the constant preserving case, an $n_0 \times n$ boolean matrix is constructed

$$\tilde{R}_0(i,j) = \begin{cases} 1 & \text{if } j \in s_i \\ 0 & \text{if } j \notin s_i \end{cases}$$

where n is the fine grid dimension, n_0 is the number of sub-domains, and s_i is the set of indices contained in the i^{th} sub-domain. It should be noted that this is a slight simplification in that normally the rows of $\tilde{R}_0$ are normalized. The matrix $\tilde{R}_0^T$ can be viewed as a simple transfer operator corresponding to piecewise constant interpolation.

While $\tilde{R}_0$ can be used, a better transfer operator is usually obtained by *smoothing* the basis functions associated with $\tilde{R}_0$. The goal of this smoothing is to reduce the energy of the basis functions, i.e. minimize the $R_0(i,:)AR_0(i,:)^T$ while maintaining the ability to accurately transfer constants. This is done by applying a matrix polynomial

$$R_0^T = p(D^{-1}A)\ \tilde{R}_0^T$$

where D is the diagonal of A, $p(x)$ is a polynomial with $p(0) = 1$, and the polynomial coefficients are related to Chebyshev polynomials [11]. In the simplest case a damped Jacobi method is employed. Figure 6 illustrates the smoothing effect on a single basis function in one dimension. In general, the degree of the polynomial must match the size of the sub-domains so that the basis functions are sufficiently smooth without having too much overlap which would lead to many nonzeros in A_0. Table 6 illustrates some typical iteration counts for an additive Schwarz two-level domain decomposition preconditioner. The results correspond to a standard five-point Laplace discretization using a Schwarz method with one level of overlap in conjunction with a smoothed aggregation coarse correction. The 3×3 sub-domain case uses a first order polynomial. The 5×5 sub-domains use a second order polynomial while the 7×7 case uses a third order

sub-domain	# sub-domains			
size	4×4	8×8	16×16	32×32
3×3	14	17	18	20
5×5	15	17	18	19
7×7	17	18	19	20

Table 6: Number of iterations using smoothed aggregation coarse problem.

polynomial. Convergence is declared when the initial residual is reduced by six orders of magnitude. The results illustrate almost no growth in the required iterations as the number of sub-domains increases. More detail about smoothed aggregation and smoothed aggregation applied within domain decomposition can be found in [11, 12, 38, 40, 50, 61, 62].

5.2 Two-level non-overlapping methods

The first version of a two-level non-overlapping preconditioner was introduced in [9] for the two-dimensional case. To describe the method consider a partitioning of the interface, Γ, into cross points and edges:

$$\Gamma = E \cup C.$$

An edge consists of vertices which border *only* two adjacent sub-domains

$$E_{jl} = (\partial\Omega_j \cap \partial\Omega_l) \setminus \partial\Omega_k \quad \forall k \neq j, l$$

and a cross point is a vertex which is adjacent to more than two sub-domains. We will denote the set of index pairs which give rise to non-empty E_{jl} by T. For each edge in T define the $|E_{jl}| \times |\Gamma|$ projection matrix $R_{E_{jl}}$ which restricts nodal values via point-wise injection to E_{jl}. Its transpose extends functions in E_{jl} by zero on the rest of the interface, Γ. The projection of the Schur complement along each edge is then given by

$$S_{E_{jl}E_{jl}} = R_{E_{jl}} S R_{E_{jl}}^T.$$

To complete the two-level preconditioner, a coarse grid operator must be defined. Assume that Ω_1, ..., Ω_N form the elements of a coarse grid mesh, τ^H, with mesh size H. That is, one coarse mesh point is associated with each fine cross point in C. Two coarse mesh vertices are adjacent if and only if there is an edge E_{jl} that connects the corresponding fine mesh vertices. An interpolation operator is defined by injecting coarse grid values to the corresponding fine grid cross points and performing linear interpolation for each adjacent pair of coarse mesh vertices along the edge connecting them. This linear interpolation operator is denoted R_{BPS}^T. Finally, A_{BPS} is the Galerkin coarse grid operator

$$A_{BPS} = R_{BPS} A R_{BPS}^T \tag{22}$$

defined on τ^H.

With this notation the BPS preconditioner is defined by

$$M_{BPS} = \sum_{(j,l)\in T} R_{E_{jl}}^T S_{E_{jl}E_{jl}}^{-1} R_{E_{jl}} + R_{BPS}^T A_{BPS}^{-1} R_{BPS}, \tag{23}$$

as described in [19]. This preconditioner (23) can be interpreted as a generalized block Jacobi scheme for the Schur complement system (11) where the block diagonal preconditioning for the Schur complement projected onto the cross points is omitted and a coarse grid correction is added.

# sub-domains	4×4	8×8	16×16	32×32
M_{BPS}	18	35	37	37
$\tilde{M}_{BPS,od}$	17	24	22	19

Table 7: # iterations to solve PDEs with discontinuities.

The coarse grid term $R_{BPS}^T A_{BPS}^{-1} R_{BPS}$ incorporates global coupling between distant interfaces. The coarse matrix A_{BPS} can be replaced by

$$S_{BPS} = R_{BPS} S R_{BPS}^T. \tag{24}$$

It has been shown [9, 56] that in general for elliptic problems the operator A_{BPS} is spectrally equivalent to S_{BPS} (in fact they are equal in some situations) and so it is possible to use either A_{BPS} or S_{BPS}. In our experience, S_{BPS} is usually more robust when highly discontinuous coefficients are present though within many computations A_{BPS} may be more attractive as it avoids the Schur complement. Of course some practical way of estimating the Schur complement and Schur complement inverses is needed. This can be done using the techniques of the previous section such as probing or Neumann-Neumann preconditioning.

Grid transfers more elaborate than linear interpolation can be considered for problems with large coefficient jumps. Multigrid transfers for PDEs with large coefficient jumps have been studied since the early 1980's [1]. One basic theme that has emerged centers on approximating the discrete PDE equations when interpolating between two vertices. For example, linear interpolation between two points satisfies the one dimensional Laplace operator (i.e. the second derivative of a linear function is identically zero). Generalizing this, it is possible to consider discretization of the original PDE on each edge, E_{jl}, and to solve this lower dimensional linear system for the basis functions. In two-dimensions this corresponds to solving a one-dimensional linear system with two different forcing functions for each edge. One forcing function is non-zero only at one edge end point (cross point) and the other is non-zero only at the other end point. Table 7 displays results from [28] that illustrate the benefit of this interpolation at a negligible extra computational cost on a heterogeneous diffusion problem $\tilde{M}_{BPS,od}$ denotes the variant built using the operator dependent interpolation. In these experiments the sub-domain size is kept constant and the number of domains is varied. This preconditioner extends to three dimensions by introducing separate terms to act on faces and edges.

Many other preconditioners can be defined by splitting interfaces such as faces, edges and cross points. Among them, we mention for two-dimensional problems the vertex-space preconditioner [53, 55] that adds a contribution from each cross point or vertex (Here the term *vertex* has been historically used as the *cross points* introduced earlier. These points correspond to sub-domain corners). In particular, define R_{RV_j} as the point-wise restriction which maps vectors defined on Γ into vectors defined within a small interface region centered on vertex j. The corresponding vertex-space preconditioner reads

$$M_{Vertex} = M_{BPS} + \sum_{j\in C} R_{RV_j}^T S_{RV_j}^{-1} R_{RV_j}$$

where $S_{RV_j} = R_{RV_j} S R_{RV_j}^T$. The condition number associated with the vertex-space preconditioner is smaller than that associated with BPS, but it is more complicated to implement especially on unstructured meshes. The three dimensional counterpart of the BPS preconditioner is the so-called wire-basket method [10, 54]. Roughly speaking, the faces play the role of the edges in the summation within (23). The coarse problem is defined on the edges and vertices (i.e. the wire-basket of the decomposition) and not only at the vertices. The interpolation operator is based on piecewise constants. Interpolated values within a face are defined by taking the average value along a face boundary (corresponding to adjacent edges and vertices) while interpolated values on edges or vertices correspond to injection.

In both the wire-basket and the smoothed aggregation methods, the grid transfer operators are partially based on piecewise constant interpolation. This essentially guarantees that the null space of the coarse problem and the original problem *match* when the original problem corresponds to a scalar elliptic PDE with Neumann boundary conditions. The balancing Neumann-Neumann preconditioner [43] is another method that in essence uses this idea. It also has an advantage in that it does not require geometric information. Let $Z_i = [z_i^1 ... z_i^{\ell_i}]$ be a set of vectors defined on Γ_i such that the null space of $S^{(i)}$ is included in $span(Z_i)$. For the Laplacian, this null space is just the constant. The balancing Neumann-Neumann method uses a combination of a Neumann-Neumann method with a coarse grid projection defined by the interpolation operator

$$R_{BNN}^T = [R_{\Gamma_1}^T D_1 Z_1 \;\; ... \;\; R_{\Gamma_N}^T D_N Z_N].$$

The original method was presented in a multiplicative context and reads

$$\begin{aligned} M_{BNN} \;\; &= \;\; \left(Id - R_{BNN}^T S_{BNN}^{-1} R_{BNN} S\right) M_{NN} \left(Id - S R_{BNN}^T S_{BNN}^{-1} R_{BNN}\right) \\ &\quad + R_{BNN}^T S_{BNN}^{-1} R_{BNN} \end{aligned} \tag{25}$$

where $S_{BNN} = R_{BNN} S R_{BNN}^T$. The computational cost per sub-domain to apply the preconditioner is two Dirichlet problems solutions plus one Neumann problem solution in addition to the coarse problem solution. This computational cost can be reduced if this preconditioner is used within a projected conjugate gradient method [58]. For the Poisson problem, the size of the coarse problem is equal to the number of sub-domains that do not touch the Dirichlet boundary of Ω and row i of R_{BNN} corresponds to the weighted sum of all unknowns on the interface of the i^{th} sub-domain.

The last set of algorithms that are considered in this chapter are FETI methods [25] (Finite Element Tearing and Interconnecting). These include a fairly large family of schemes that are quite popular in the structural mechanics community. FETI methods differ from all of the algorithms presented in that they solve first for the Langrange multipliers (or dual variables). In particular, the original problem can be recast into the following equivalent linear system

$$\begin{pmatrix} \mathcal{A}_1 & & & & \mathcal{B}_1^T \\ & \cdot & & & \cdot \\ & & \cdot & & \cdot \\ & & & \cdot & \cdot \\ & & & \mathcal{A}_N & \mathcal{B}_N^T \\ \mathcal{B}_1 & \cdot & \cdot & \cdot \;\; \mathcal{B}_N & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_N \\ \lambda \end{pmatrix} = \begin{pmatrix} b_1 \\ \cdot \\ \cdot \\ \cdot \\ b_N \\ 0 \end{pmatrix} \tag{26}$$

where u_i is the solution restricted to sub-domain i, b_i is the forcing function associated with finite element functions corresponding to Ω_i, $\mathcal{B}_i$ is a signed boolean matrix which maps sub-domains to the interface. The last block row of (26) corresponds to the constraint that a shared unknown which appears in u_i and u_j must be equal. As previously noted, some of the $\mathcal{A}_i$'s are singular corresponding to Neumann problems. Let us assume that there are p singular systems and that they are ordered first in (26). In the original FETI method the u_i corresponding to singular systems are effectively split into two components, $\bar{u}_i$ and α_i such that

$$u_i = \bar{u}_i + Z_i \alpha_i \quad \text{and} \quad Z_i^T \bar{u}_i = 0$$

where Z_i is the null space of $\mathcal{A}_i$. The basic idea in FETI is to algebraically eliminate the $\bar{u}_i$'s corresponding to singular systems and to elminate the u_i's corresponding to nonsingular systems.

This leads to the following equivalent set of equations involving α's and λ's:

$$\begin{pmatrix} F & -G \\ -G^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \alpha \end{pmatrix} = \begin{pmatrix} d \\ -e \end{pmatrix}$$

where

$$\begin{array}{rclrcl} F & = & \sum_{i=1}^{N} \mathcal{B}_i A_i^+ \mathcal{B}_i^T, & d & = & \sum_{i=1}^{N} \mathcal{B}_i A_i^+ b_i, \\ G & = & [\ \mathcal{B}_1 Z_1 \ \ \ldots \ \ \mathcal{B}_N Z_p\ ], & e & = & [\ b_1^T Z_1 \ \ \ldots \ \ b_N^T Z_p\ ]^T, \\ \alpha & = & [\ \alpha_1 \ \ \ldots \ \ \alpha_p\ ]^T, & & & \end{array} \tag{27}$$

and $\mathcal{A}_i^+$ denotes the pseudo-inverse of $\mathcal{A}_i$. The α's take on the role of coarse grid variables. The resulting system is solved via a reduced conjugate gradient algorithm requiring at each iteration the solution of a system involving $G^T G$. The FETI algorithm is actually quite similar to the balancing Neumann-Neumann technique and can be loosely considered its dual method focusing on Lagrange multipliers as opposed to primal variables.

Many different FETI variants have been (and continue to be) considered. These employ different local approximations along interfaces or different coarse mesh formulations. One particular interesting variation, FETI-DP [24], essentially splits unknowns into corners (or cross points) and non-corners. An expanded form of (26) is formed where the unknowns include non-corners, corners, and Lagrange multipliers. Conceptually, the non-corners are first eliminated algebraically involving sub-domain solves. The corners are then eliminated algebraically involving a coarse grid solve. The Lagrange multipliers are then determined via a conjugate gradient method where each iteration requires sub-domain solves and a coarse mesh solve. We do not further describe the FETI methods, but refer to [58] and the references therein for a recent and exhaustive presentation of this class of approaches.

6 Final comments

There exists a rich literature on domain decomposition methods. The series of conference proceedings devoted to this topic is a first source of information (see `www.ddm.org`). A concise and nice overview of domain decomposition techniques can be found in [19]. A series of useful books are also available [45, 56, 58] where the interested reader can find detailed presentations of these numerical techniques and a complete bibliography.

References

[1] R. Alcouffe, A. Brandt, J. Dendy, and J. Painter, *The multigrid method for diffusion equations with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 430–454.

[2] P. R. Amestoy and I. S. Duff, *Memory management issues in sparse multifrontal methods on multiprocessors*, Int. J. of Supercomputer Applics., 7 (1993), pp. 64–82.

[3] P. R. Amestoy and C. Puglisi, *An unsymmetrized multifrontal LU factorization*, SIAM J. Matrix Analysis and Applications, 24 (2002), pp. 553–569.

[4] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[5] M. Benzi, C. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method.*, SIAM J. Scientific Computing, 17 (1996), pp. 1135–1149.

[6] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems.*, SIAM J. Scientific Computing, 19 (1998), pp. 968–994.

[7] P. Bjørstad and O. Widlund, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numerical Analysis, 23 (1986), pp. 1097–1120.

[8] J.-F. Bourgat, R. Glowinski, P. L. Tallec, and M. Vidrascu, *Variational formulation and algorithm for trace operator in domain decomposition calculations*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, PA, 1989, SIAM, pp. 3–16.

[9] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *The construction of preconditioners for elliptic problems by substructuring I.*, Math. Comp., 47 (1986), pp. 103–134.

[10] ——, *The construction of preconditioners for elliptic problems by substructuring, IV*, Math. Comp., 53 (1989), pp. 1–24.

[11] M. Brezina, *Robust Iterative Solvers on Unstructured Meshes*, PhD thesis, University of Colorado at Denver, Denver, CO 80217-3364, 1997.

[12] M. Brezina and P. Vaněk, *A black-box iterative solver based on a two-level Schwarz method*, Computing, 63 (1999), pp. 233–263. Dec 07, 1999.

[13] W. L. Briggs, V. E. Henson, and S. McCormick, *A multigrid tutorial, Second Edition*, SIAM, Philadelphia, 2000.

[14] X.-C. Cai, M. Dryja, and M. Sarkis, *Restricted additive Schwarz preconditioners with harmonic overlap for symmetric positive definite linear systems*, SIAM J. Numerical Analysis, 41 (2003), pp. 1209–1231.

[15] X.-C. Cai, W. D. Gropp, and D. E. Keyes, *A comparison of some domain decomposition algorithms for nonsymmetric elliptic problems*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, D. Keyes, T. Chan, G. Meurant, J. Scroggs, and R. Voigt, eds., Philadelphia, PA, 1992, SIAM, pp. 224–235.

[16] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Scientific Computing, 21 (1999), pp. 792–797.

[17] T. F. Chan and D. F. Keyes, *Interface preconditioning for domain-decomposed convection-diffusion operators*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, PA, 1990, SIAM, pp. 245–262.

[18] T. F. Chan and T. P. Mathew, *The interface probing technique in domain decomposition*, SIAM J. Matrix Analysis and Applications, 13 (1992), pp. 212–238.

[19] ——, *Domain Decomposition Algorithms*, vol. 3 of Acta Numerica, Cambridge University Press, Cambridge, 1994, pp. 61–143.

[20] T. F. Chan, T. P. Mathew, and J.-P. Shao, *Fourier and probe variants of the vertex space domain decomposition algorithm*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, D. Keyes, T. Chan, G. Meurant, J. Scroggs, and R. Voigt, eds., Phil., 1992, SIAM, pp. 236–249.

[21] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners.*, SIAM J. Scientific Computing, 21 (2000), pp. 1804–1822.

[22] A. R. Curtis, M. J. D. Powell, and J. K. Reid, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–120.

[23] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, London, 1986.

[24] C. Farhat, M. Lesoinne, P. Le Tallec, K. Pierson, and D. Rixen, *FETI-DP: A dual-primal unified FETI method – part I: A faster alternative to the two-level FETI method.*, Int. J. Numer. Meth. Engrg., 50 (2001), pp. 1523–1544.

[25] C. Farhat and F.-X. Roux, *Implictly parallel processing in strcutural mechanics*, in Computational Mechanics Advances, volume 2, J. T. Oden, ed., North-Holland, Amsterdam, 1994, pp. 1–124.

[26] M. Gee, J. Hu, M. Sala, C. Siefert, and R. S. Tuminaro, *ML 5.0 smoothed aggregation user's guide*, Tech. Rep. SAND2006-2649, Sandia National Laboratories, July 2006.

[27] L. Giraud and R. S. Tuminaro, *Schur complement preconditioners for anisotropic problems*, IMA J. Numerical Analysis, 19 (1999), pp. 1–17.

[28] L. Giraud, F. G. Vasquez, and R. S. Tuminaro, *Grid transfer operators for highly variable coefficient problems in two-level non-overlapping domain decomposition methods*, Numerical Linear Algebra with Applications, 10 (2003), pp. 467–484.

[29] G. Golub and D. Mayers, *The use of preconditioning over irregular regions*, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J. L. Lions, eds., 1984, pp. 3–14. Proceedings of a conference held in Versailles, France, December 12-16,1983.

[30] G. H. Golub and C. F. V. Loan, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, The Johns Hopkins University Press, Baltimore, MD, USA, third ed., 1996.

[31] A. Greenbaum, *Iterative methods for solving linear systems*, SIAM, Philadelphia, PA, USA, 1997.

[32] M. Grote and T. Huckle, *Parallel preconditionings with sparse approximate inverses.*, SIAM J. Scientific Computing, 18 (1997), pp. 838–853.

[33] W. Hackbusch, *Multigrid methods and applications*, Springer-Verlag, 1985.

[34] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Applied Numerical Mathematics: Transactions of IMACS, 41 (2002), pp. 155–177.

[35] M. Heroux, *AztecOO user guide*, Tech. Rep. SAND2004-3796, Sandia National Laboratories, Albuquerque, NM, 87185, 2004.

[36] M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. S. Tuminaro, J. Willenbring, and A. Williams, *An Overview of Trilinos*, Tech. Rep. SAND2003-2927, Sandia National Laboratories, 2003.

[37] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the trilinos project*, ACM Transactions on Mathematical Software, 31 (2005), pp. 397–423.

[38] L. Jenkins, T. Kelley, C. Miller, and C. Kees, *An aggregation-based domain decomposition preconditioner for groundwater flow*, Tech. Rep. TR00–13, Department of Mathematics, North Carolina State University, 2000.

[39] L. Y. Kolotilina, A. Y. Yeremin, and A. A. Nikishin, *Factorized sparse approximate inverse preconditionings. III: Iterative construction of preconditioners*, Journal of Mathematical Sciences, 101 (2000), pp. 3237–3254. Originally published in Russian in *Zap. Nauchn. Semin. POMI*, 248:17-48, 1998.

[40] C. Lasser and A. Toselli, *Convergence of some two-level overlapping domain decomposition preconditioners with smoothed aggregation coarse spaces*, in Recent Developments in Domain Decomposition Methods, L. Pavarino and A. Toselli, eds., vol. 23 of LNCSE, Springer Verlag, 2002.

[41] X. Li, *An overview of SuperLU: Algorithms, implementation, and user interface*, Technical Report LBNL-53848, Lawrence Berkeley National Labs, 2003.

[42] Z. Li, Y. Saad, and M. Sosonkina, *pARMS: A parallel version of the algebraic recursive multilevel solver*, Tech. Rep. Technical Report UMSI-2001-100, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, 2001.

[43] J. Mandel, *Balancing domain decomposition*, Communications in Numerical Methods in Engineering, 9 (1993), pp. 233–241.

[44] *Prometheus*. http://www.columbia.edu/ ma2325/prometheus/.

[45] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, 1999.

[46] Y.-H. D. Roeck and P. L. Tallec, *Analysis and test of a local domain decomposition preconditioner*, in Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, Y. Kuznetsov, G. Meurant, J. Périaux, and O. Widlund, eds., Philadelphia, PA, USA, 1991, SIAM.

[47] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numerical Linear Algebra with Applications, 1 (1994), pp. 387–402.

[48] ———, *Iterative Methods for Sparse Linear Systems.*, SIAM, Philadelphia, 2003. Second edition.

[49] Y. Saad and M. Sosonkina, *distributed Schur complement techniques for general sparse linear systems*, SIAM J. Scientific Computing, 21 (1999), pp. 1337–1356.

[50] M. Sala, J. N. Shadid, and R. S. Tuminaro, *An improved convergence bound for aggregation-based domain decomposition preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 744–756.

[51] H. A. Schwarz, *Über eine grenzübergang durch alternirendes verfahren*, Gesammelete Mathematische Abhandlungen, Springer-Verlag, 2 (1890), pp. 133–143. First published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, vol.15, pp. 272-286, 1870.

[52] C. Siefert and E. de Sturler, *Probing methods for saddle-point problems*, Elec. Trans. Numer. Anal., 22 (2006), pp. 163–183.

[53] B. F. Smith, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, PhD thesis, Courant Institute of Mathematical Sciences, September 1990. Tech. Rep. 517, Department of Computer Science, Courant Institute.

[54] ——, *A domain decomposition algorithm for elliptic problems in three dimensions*, Numer. Math., 60 (1991), pp. 219–234.

[55] ——, *An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems*, SIAM J. Scientific and Statistical Computing, 13 (1992), pp. 364–378.

[56] B. F. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, New York, 1st ed., 1996.

[57] P. L. Tallec, Y.-H. D. Roeck, and M. Vidrascu, *Domain-decomposition methods for large linearly elliptic three dimensional problems*, J. of Computational and Applied Mathematics, 34 (1991), pp. 93–117.

[58] A. Toselli and O. Widlund, *Domain Decomposition methods-Algorithms and Theory*, Springer, 2005.

[59] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[60] R. S. Tuminaro, M. Heroux, S. Hutchinson, and J. Shadid, *Official Aztec user's guide: Version 2.1*, Tech. Rep. Sand99-8801J, Sandia National Laboratories, Albuquerque, NM, 87185, Nov 1999.

[61] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88 (2001), pp. 559–579.

[62] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[63] P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, West Sussex, 1992.

[64] J. Xu, *Iterative methods by space decomposition and subspace correction: A unifying approach*, SIAM Review, 34 (1992), pp. 581–613.