A Domain Decomposition Preconditioner for Hermite Collocation Problems

Gabriel Mateescu[†] Calvin J. Ribbens[‡] Layne T. Watson[§]

Abstract

We propose a preconditioning method for linear systems of equations arising from piecewise Hermite bicubic collocation applied to two-dimensional elliptic PDEs with mixed boundary conditions. We construct an efficient, parallel preconditioner for the GMRES method. The main contribution of the paper is a novel interface preconditioner derived in the framework of substructuring and employing a local Hermite collocation discretization for the interface subproblems based on a hybrid fine-coarse mesh. Interface equations based on this mesh depend only weakly on unknowns associated with subdomains. The effectiveness of the proposed method is highlighted by numerical experiments that cover a variety of problems.

1 Introduction

We are interested in parallel iterative solution methods applicable to high order discretization methods for linear second-order two-dimensional elliptic PDEs. Parallelism is essential for solving large problems. We describe a new preconditioning algorithm for a Krylov subspace iterative solution of linear systems arising from a piecewise Hermite bicubic collocation discretization of elliptic PDEs defined on a two-dimensional domain. For convenience, we refer to the collocation method as *Hermite collocation*.

[†]National Research Council Canada, Ottawa, Canada K1A 0R6.

[‡]Department of Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, VA, 24061. Supported in part by DARPA ARO grant DAAH04-94-G-0010.

[§]Departments of Computer Science and Mathematics, Virginia Polytechnic Institute & State University, Blacksburg, VA, 24061. Supported in part by AFOSR grant F49620-96-1-0104 and NSF grant DMS-9625968.

Hermite collocation is an attractive discretization technique for at least two reasons. First, unlike a finite element Galerkin method, collocation does not require the computation of any integrals for generating the coefficient matrix. Second, for a general linear elliptic operator, with sufficiently smooth coefficients and Dirichlet boundary conditions, Hermite collocation can provide $O(h^4)$ accuracy in the L^2 norm, where h is the grid step [2].

We employ the Generalized Minimal Residual (GMRES) method [14], whose strengths include generality and robustness. The discrete operator A generated by Hermite collocation is typically ill-conditioned and unpreconditioned GMRES as well as point-Jacobi preconditioned GMRES exhibit poor convergence. Therefore, finding a good preconditioner is critical for solving Hermite collocation problems with a Krylov-subspace method.

The main contribution of the paper is a preconditioner for solving Hermite collocation problems with Krylov subspace methods. The preconditioner can be applied to a second-order elliptic PDE defined on a rectangular domain Ω and with general boundary conditions. In the framework of substructuring, we introduce a novel interface preconditioner, which we call the *edge preconditioner*. The edge preconditioner is constructed on a hybrid fine-coarse grid defined for each interface. The proposed method has high parallelism and our experimental results indicate good convergence and accuracy.

Bialecki [1] has proposed a domain decomposition solver for Poisson's equation on a rectangle, discretized with Hermite collocation. Bialecki, et al. [3] have described an additive Schwarz method for solving the Dirichlet Poisson's equation on a rectangle. Christara and Smith [7] have designed multilevel methods for two-dimensional elliptic PDEs discretized with quadratic spline collocation. We are not aware of another preconditioner derived by substructuring for Hermite collocation problems arising from the discretization of general two-dimensional elliptic PDEs.

The remainder of the paper is organized as follows. After describing the Hermite collocation discretization of a boundary value problem in Section 2, we present background information and related work in Section 3. The edge preconditioner is described in Section 4. Numerical results are presented in Section 5 followed by concluding remarks in Section 6.

2 The Hermite Collocation Problem

Let $\Omega = (0,1) \times (0,1)$ and $\partial\Omega$ its boundary. We consider the boundary value problem

$$\mathbf{L}u = g \quad \text{in } \Omega \qquad \text{and} \qquad \mathbf{B}u = t \quad \text{on } \partial\Omega, \tag{1}$$

where $\mathbf{L}$ is a linear elliptic operator, $\mathbf{B} = p\,\frac{\partial}{\partial \mathbf{n}} + q$, g, t, p, and q are at least C^0, and $\mathbf{n}$ is the outward unit normal to the boundary. Let n_0 be an integer greater than one, and define the partitions $X = \{x_k\}_{k=0}^{n_0-1}$ and $Y = \{y_m\}_{m=0}^{n_0-1}$ of $[0,1]$, where $x_k = k\,h$, $y_m = m\,h$, and $h = 1/(n_0-1)$. The set of grid points, or *nodes*, in $\bar{\Omega} = \Omega \cup \partial\Omega$ is $G^h = X \times Y$. The method proposed here can also be applied when X and Y are of different sizes.

Collocation methods approximate the unknown function u by a function U in a finite dimensional function space $\mathcal{V}$. In Hermite collocation, $\mathcal{V}$ is the space of continuously differentiable piecewise bicubic polynomials. Formally, $\mathcal{V} = \mathcal{V}_x \otimes \mathcal{V}_y$, where $\mathcal{V}_z = \{v \in C^1[0,1] : v|_{[z_k, z_{k+1}]} \in \mathbb{P}_3, k = 0, \ldots, n_0 - 2\}$, $z \in \{x, y\}$, and $\mathbb{P}_3$ is the set of polynomials of degree at most 3. Several authors [1, 3] seek a solution in the space $\mathcal{V}^0 = \mathcal{V}_x^0 \otimes \mathcal{V}_y^0$, where $\mathcal{V}_z^0 = \{v \in \mathcal{V}_z : v(0) = v(1) = 0\}$, where $z \in \{x, y\}$. Considering $\mathcal{V}^0$ avoids including the boundary equations in the linear system, but $\mathcal{V}^0$ can only be used for solving the homogeneous Dirichlet problem.

Let $n_G = n_0^2$ be the number of nodes in G^h, and $\mathcal{N} : G^h \mapsto \{1, \ldots, n_G\}$ be a node ordering. Following [9], with each node n in G^h, $n = \mathcal{N}(x_k, y_m)$, there are associated four *Hermite bicubic* basis functions, $\Phi_n^{(i)}$, $i = 1, 2, 3, 4$, centered at n and with support $[x_{k-1}, x_{k+1}] \times [y_{m-1}, y_{m+1}]$. The set $\mathcal{B} = \{\Phi_n^{(i)} \mid 1 \le i \le 4, 1 \le n \le n_G\}$ is a *basis* for $\mathcal{V}$. The expression of a function $U \in \mathcal{V}$ in the basis $\mathcal{B}$ is

$$U(x,y) = \sum_{i=1}^{4} \sum_{n=1}^{n_G} U_n^{(i)} \Phi_n^{(i)}(x,y), \tag{2}$$

$U_n^{(1)} = U(x_k, y_m)$, $U_n^{(2)} = \frac{\partial U}{\partial y}(x_k, y_m)$, $U_n^{(3)} = \frac{\partial U}{\partial x}(x_k, y_m)$, $U_n^{(4)} = \frac{\partial^2 U}{\partial x \partial y}(x_k, y_m)$. Notice that we have implicitly defined a *type order* of the unknowns $U_n^{(i)}$ associated with a node; that is, the superscripts 1, 2, 3, 4 are associated with U, $\frac{\partial U}{\partial y}$, $\frac{\partial U}{\partial x}$, $\frac{\partial^2 U}{\partial x \partial y}$, respectively.

To achieve the $O(h^4)$ accuracy of the solution predicted in [2], the collocation points are determined in terms of the set of Gauss points in $(0,1)$, $\mathcal{G}_1 = \{\lfloor \frac{k-1}{2} \rfloor h + \frac{h}{6}\left(3 + (-1)^k \sqrt{3}\right), 1 \le k \le 2(n_0-1)\}$. The sets of collocation points in Ω and on $\partial\Omega$ are, respectively, $\mathcal{G} = \{(\xi, \zeta) : \xi, \zeta \in \mathcal{G}_1\}$ and $\partial\mathcal{G} = \{(\xi, \zeta) : \xi, \zeta \in \{0,1\}\} \cup \{(\xi, \zeta) : \xi \in \mathcal{G}_1, \zeta \in \{0,1\}\} \cup \{(\xi, \zeta) : \xi \in \{0,1\}, \zeta \in \mathcal{G}_1\}$. Notice that the set $\mathcal{G}$ has $4(n_0-1)^2$ elements and $\partial\mathcal{G}$ has $8(n_0-1)+4$ elements; thus, $\mathcal{G} \cup \partial\mathcal{G}$ has $4n_G$ elements.

The Hermite collocation problem is to find $U \in \mathcal{V}$ which satisfies (1) at the collocation points, i.e.,

$$\begin{aligned} \mathbf{L}U(\xi,\zeta) &= g(\xi,\zeta), \qquad (\xi,\zeta) \in \mathcal{G}, \\ \mathbf{B}U(\xi,\zeta) &= t(\xi,\zeta), \qquad (\xi,\zeta) \in \partial\mathcal{G}, \end{aligned} \tag{3}$$

where U is given by (2). The purpose of the paper is to solve the linear system $A\,\mathbf{u} = \mathbf{f}$ arising from (3), where $\mathbf{u} \in \mathbb{R}^N$, $A \in \mathbb{R}^{N\times N}$ is the coefficient matrix, and $N = 4n_G$.

3 Background and Related Work

3.1 Substructuring

Substructuring works by partitioning the problem domain Ω into nonoverlapping *subdomains* separated by an *interface*. One-dimensional partitioning creates *parallel strips* (the subdomains) separated by *edges* (the interface). For a two-dimensional partitioning, the interface consists of edges and *vertices*, where the latter are endpoints common to several edges.

Let $\bar{\Omega}$ be partitioned into n_I parallel strips (subdomains) $\Omega_1, \ldots, \Omega_{n_I}$ and n_E edges $\mathcal{E}_1, \ldots, \mathcal{E}_{n_E}$ with $n_E = n_I - 1$, where the partitioning of the domain is performed along grid lines. For the sake of simplicity of notation, assume that the parallel strips are horizontal; for vertical strips, the description is analogous. Let $y_j \in Y$ be the grid lines which define the edges $\mathcal{E}_j = (0,1) \times y_j$, where $1 \le j \le n_E$.

After partitioning, the nodes, unknowns, and equations are reordered in *substructuring order*: the nodes in the subdomains Ω_i, $1 \le i \le n_I$ are numbered first, followed by the nodes in $\mathcal{E}_j$, $1 \le j \le n_E$, in increasing order of i and j. Figure 1 illustrates the substructuring order of nodes for a parallel strip decomposition with three strips.

For Hermite collocation, the order of the grid nodes does not completely specify the unknown and equation order. The unknown order is specified by the type order described in Section 2. On the other hand, the order of the equations is that of the collocation points. We choose an order of the collocation points (equations) which satisfies two criteria: (i) substructuring preserves the contiguity of the indices for the equations corresponding to a subdomain, and (ii) the ith equation contains the ith unknown, for each $i, 1 \le i \le N$. To meet these requirements, we employ the *collorder* numbering proposed by Dyksen and Rice in [8], where it is described as follows: *The grid [...] points are numbered in a natural way from west to east, south to north. The collocation points (equations) are then associated with the nearest grid point and are numbered in groups of four in the order of their corresponding grid point.* Figure 2 illustrates the natural order (column-wise) of the nodes and the collorder of the collocation points.

Since the unknown and equation orderings are bound to the grid node order

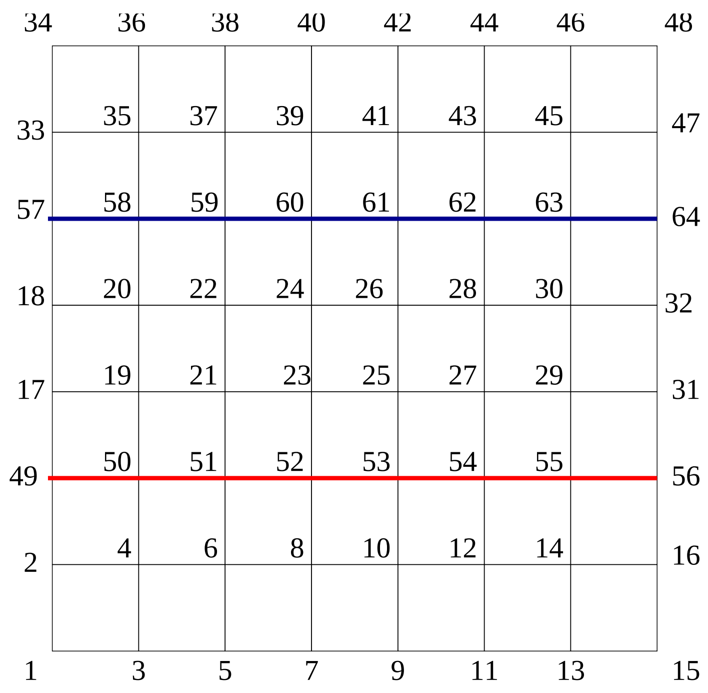

Figure 1: Substructuring order for a three-strip partition.

via the type order and collorder, respectively, performing the substructuring ordering of the nodes induces a permutation of the unknowns and equations, i.e., a symmetric permutation of the coefficient matrix. Hence, after substructuring the coefficient matrix is $A = P\,A_0\,P^T$, where P is a permutation matrix and A_0 is the coefficient matrix before substructuring.

Moreover, by the type order and collorder, each unknown and equation is mapped to a node, so we can associate every unknown and equation with either a subdomain or an edge. Denote the set of indices of the unknowns and equations associated with Ω_i by I_i, and let $I = \cup_{i=1}^{n_I} I_i$. Likewise, denote by E_j the set of indices associated with $\mathcal{E}_j$, and $E = \cup_{j=1}^{n_E} E_j$. The unknown and right hand side vectors $\mathbf{u}$ and $\mathbf{f}$ can be written as $\mathbf{u} = (\mathbf{u}_I, \mathbf{u}_E)$ and $\mathbf{f} = (\mathbf{f}_I, \mathbf{f}_E)$, respectively. We employ superscripts to indicate subcomponents of a vector, e.g., $\mathbf{u}_I^{(i)}$ denotes the part of $\mathbf{u}_I$ associated with Ω_i and $\mathbf{f}_I^{(j)}$ denotes the part of $\mathbf{f}_E$ associated with E_j.

For simplicity, assume a partition in which every strip contains $n_0 \times K$ grid nodes, where $K \geq 2$ is the number of grid lines in a strip, along the direction of partitioning and $n_0 = K\,n_I + n_E$. Then, the size of I_i is $N_I = 4\,K\,n_0$, the size of E_j is $N_E = 4\,n_0$ and the size of $I \cup E$ is equal to the number of unknowns

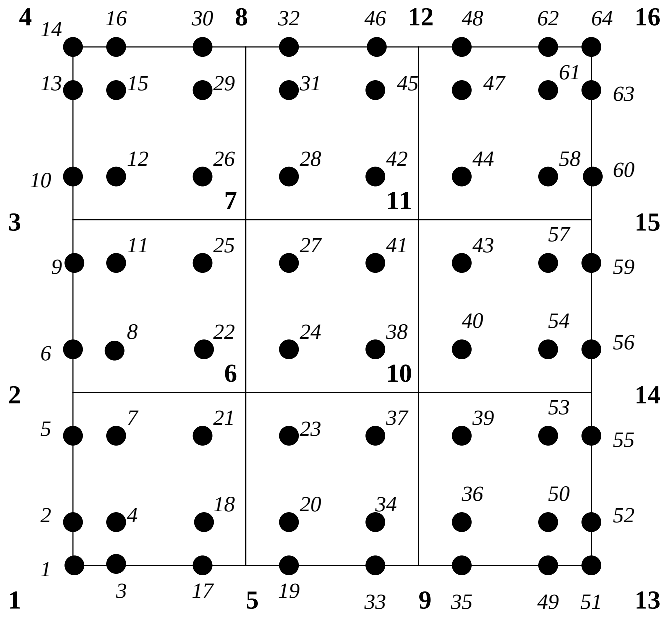
Figure 2: Natural order of nodes (bold numbers) and collorder of collocation points, where the collocation points are represented as disks, for a domain Ω with a 4×4 grid.

$N = n_I\, N_I + n_E\, N_E$.

The coefficient matrix A in the substructuring order,

$$A = \begin{pmatrix} A_{II} & A_{IE} \\ A_{EI} & A_{EE} \end{pmatrix}, \qquad \text{where } A_{II} = \text{blockdiag}(A_{II}^{(1)}, A_{II}^{(2)}, \ldots, A_{II}^{(n_I)}), \quad (4)$$

has an *arrowhead* structure, i.e., it can be viewed as a 2×2 block matrix in which the upper left block A_{II} has a block diagonal structure. The first subscript of a matrix block denotes whether the coefficients in that block belong to subdomain (I) or edge (E) equations. The second subscript denotes whether the coefficients multiply subdomain (I) or edge (E) unknowns. For example, A_{IE} denotes the coefficients of the interface unknowns in the subdomain equations. For a Hermite collocation discretization, $A_{EI} \neq A_{IE}^T$ even when $\mathbf{L}$ is self-adjoint [4]. The resulting linear system is

$$\begin{pmatrix} A_{II} & A_{IE} \\ A_{EI} & A_{EE} \end{pmatrix} \begin{pmatrix} \mathbf{u}_I \\ \mathbf{u}_E \end{pmatrix} = \begin{pmatrix} \mathbf{f}_I \\ \mathbf{f}_E \end{pmatrix}.$$

The *Schur complement* of A_{II} in A is

$$S = A_{EE} - A_{EI} A_{II}^{-1} A_{IE}, \qquad (5)$$

allowing us to write a block-UDL factorization of A^{-1} as

$$A^{-1} = \begin{pmatrix} I & -A_{II}^{-1}A_{IE} \\ 0 & I \end{pmatrix} \begin{pmatrix} A_{II}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -A_{EI}A_{II}^{-1} & I \end{pmatrix}. \qquad (6)$$

3.2 Preconditioners Derived using Substructuring

We need to replace S^{-1} in (6) by a preconditioner for two reasons. First, S is a dense matrix and explicitly constructing it is expensive. Second, inverting or factoring a dense matrix is expensive as well.

Preconditioners are derived in the substructuring framework following a *two-scale procedure* [6]: there is a *fine grid*, with step-size h, and a *coarse* grid, with step-size H, associated with Ω. The coarse grid corresponds to a two-dimensional partitioning and provides the global transfer of information which is needed because the solution $\mathbf{u}(\mathbf{x}_i)$ of an elliptic PDE at any point $\mathbf{x}_i$ depends on all the components of $\mathbf{f}$. The coarse grid creates an interface between subdomains consisting of edges and vertices, and the unknown vector is $\mathbf{u} = (\mathbf{u}_I, \mathbf{u}_E, \mathbf{u}_V)$, where V is the set of indices of the unknowns associated with the vertices.

The seminal paper of Bramble, Pasciak, and Schatz [5] has proposed a preconditioning strategy in this framework. The Bramble-Pasciak-Schatz (BPS) preconditioner can described as follows. Consider the approximation of S^{-1},

$$\hat{S}^{-1} = \begin{pmatrix} S_{1,1}^{-1} & 0 & \dots & 0 & 0 \\ 0 & S_{2,2}^{-1} & & 0 & 0 \\ \vdots & & \ddots & 0 & \vdots \\ 0 & \dots & \dots & S_{n_E,n_E}^{-1} & 0 \\ 0 & \dots & \dots & 0 & S_{VV}^{-1} \end{pmatrix}, \qquad (7)$$

where $S_{i,i}$, $1 \leq i \leq n_E$, is the block of S that contains the coefficients of $\mathbf{u}_E^{(i)}$ which belong to equations associated with $\mathcal{E}_i$, and S_{VV} is the block of S that contains the coefficients $\mathbf{u}_V$ which belong to equations associated with the vertex nodes of the coarse grid. Relation (7) is based on a block diagonal approximation of S. The BPS preconditioner uses a *coarse grid operator* A_H to provide global coupling between subdomains, and an *interpolation map* $R_H^T : V \mapsto E \cup V$ is used to approximate the solution on $E \cup V$. The map $R_H : E \cup V \mapsto V$ is the *weighted restriction map* from $E \cup V$ to V. With these definitions the BPS preconditioner is

$$\hat{S}_{BPS}^{-1} = \begin{pmatrix} S_{1,1}^{-1} & 0 & \dots & 0 & 0 \\ 0 & S_{2,2}^{-1} & & 0 & 0 \\ \vdots & & \ddots & 0 & \vdots \\ 0 & \dots & \dots & S_{n_E,n_E}^{-1} & 0 \\ 0 & \dots & \dots & 0 & 0 \end{pmatrix} + R_H^T A_H^{-1} R_H. \tag{8}$$

In a sense, the BPS method only provides a strategy for constructing a preconditioner rather than specifying an actual preconditioner. That is, it does not specify an efficient scheme for approximating $S_{i,i}$; recall that such an approximation is desired because computing $S_{i,i}$ explicitly is expensive. Furthermore, the applicability of this method to non self-adjoint Hermite collocation problems has not been thoroughly investigated. The preconditioner that we propose provides a method for approximating $S_{i,i}$, and accounts for the global coupling through the hybrid coarse/fine grids associated with the edges, rather than using the operator A_H.

4 The Preconditioner

In this section we describe the edge preconditioner which is derived by partitioning the domain in parallel strips and approximating the Schur complement using a discretization on a special grid called the *edge grid*. The edge grid discretization induces subproblems which have weak coupling with the subdomains. This sets the stage for deriving a block diagonal approximation to S, from which an effective parallel preconditioner for A is obtained.

The proposed preconditioner provides an approximate solution of the boundary value problems induced by substructuring. These problems are defined in the next subsection.

4.1 Problem decomposition

For each edge $\mathcal{E}_j = (0,1) \times y_i$, we define $\bar{\mathcal{E}}_j = (0,1) \times (y_j - h_E, y_j + h_E)$, where $1 \leq j \leq n_E$ and $h_E \geq h$. Let $\mathcal{E} = \cup_{j=1}^{n_E} \mathcal{E}_j$. We approximate problem (1) by n_I subdomain problems

$$\begin{aligned} &\mathbf{L}\hat{u}_i = g - \mathbf{L}\hat{u}_E \text{ in } \Omega_i, \text{ for } 1 \leq i \leq n_I, \\ &\mathbf{B}\hat{u}_i = t \text{ on } \partial\Omega_i \cap \partial\Omega \text{ and } \hat{u}_i = \partial\hat{u}_i/\partial y = 0 \text{ on } \partial\Omega_i \cap \mathcal{E}, \\ &\text{with } \hat{u}_i \in C^1(\Omega), \hat{u}_i \equiv 0 \text{ on } \bar{\Omega} - (\Omega_i \cup \partial\Omega_i)\,, \end{aligned} \tag{9}$$

and n_E interface problems

$$\begin{array}{l}\mathbf{L}\hat{u}_{E_j} = g - \mathbf{L}\hat{u}_I \text{ in } \bar{\mathcal{E}}_j, \text{ for } 1 \le j \le n_E, \\ \mathbf{B}\hat{u}_{E_j} = t \text{ on } \partial\bar{\mathcal{E}}_j \cap \partial\Omega \text{ and } \hat{u}_{E_j} = \partial\hat{u}_{E_j}/\partial y = 0 \text{ on } \partial\bar{\mathcal{E}}_j \cap \Omega, \\ \text{with } \hat{u}_{E_j} \in C^1(\Omega), \hat{u}_{E_j} \equiv 0 \text{ on } \bar{\Omega} - \bar{\mathcal{E}}_j,\end{array} \tag{10}$$

where $\hat{u}_I = \sum_{i=1}^{n_I} \hat{u}_i$ and $\hat{u}_E = \sum_{j=1}^{n_E} \hat{u}_{E_j}$. The solution of (1) is approximated by $\hat{u} = \hat{u}_I + \hat{u}_E$. Our preconditioner, which is derived in Section 4.3, is an approximate solution of the Hermite bicubic collocation problems induced by (9) and (10), and specifies a value of h_E.

4.2 Schur Complement Block Structure

For strip substructuring, A_{EE} is a block diagonal matrix,

$$A_{EE} = \begin{pmatrix} A_{EE}^{(1)} & 0 & \dots & 0 \\ 0 & A_{EE}^{(2)} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & A_{EE}^{(n_E)} \end{pmatrix}., \tag{11}$$

where the block $A_{EE}^{(i)}$ represents the coefficients of the unknowns associated with edge i in the equations imposed at the collocation points associated with edge i.

Define $A_{EI}^{(i,j)}$ to be the matrix block which contains the coefficients of the unknowns from subdomain j in the equations imposed at the collocation points associated with edge i. The matrix A_{EI} has the structure

$$A_{EI} = \begin{pmatrix} A_{EI}^{(1,1)} & A_{EI}^{(1,2)} & 0 & 0 & \dots & 0 & 0 \\ 0 & A_{EI}^{(2,2)} & A_{EI}^{(2,3)} & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & A_{EI}^{(n_E,n_E)} & A_{EI}^{(n_E,n_E+1)} \end{pmatrix}. \tag{12}$$

Similarly, define $A_{IE}^{(i,j)}$ to be the matrix block which contains the coefficients of the unknowns from edge j in the equations imposed at the collocation points associated with subdomain i. The matrix A_{IE} has the structure

$$A_{IE} = \begin{pmatrix} A_{IE}^{(1,1)} & 0 & 0 & \dots & 0 & 0 \\ A_{IE}^{(2,1)} & A_{IE}^{(2,2)} & 0 & \dots & 0 & 0 \\ & \ddots & \ddots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & A_{IE}^{(n_E,n_E-1)} & A_{IE}^{(n_E,n_E)} \\ 0 & 0 & 0 & \dots & 0 & A_{IE}^{(n_E+1,n_E)} \end{pmatrix}. \tag{13}$$

The Schur complement matrix S associated with a strip decomposition has a block tridiagonal structure, with a diagonal block for each edge (see [11] for a proof). Specifically,

$$S = \begin{pmatrix} S_{1,1} & S_{1,2} & 0 & 0 & 0 & \dots & 0 & 0 \\ S_{2,1} & S_{2,2} & S_{2,3} & 0 & 0 & \dots & 0 & 0 \\ 0 & S_{3,2} & S_{3,3} & S_{3,4} & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & S_{n_E,n_E-1} & S_{n_E,n_E} \end{pmatrix}, \qquad (14)$$

where the blocks have the shape $N_E \times N_E$ and are given by

$$S_{i,i} = \ A_{EE}^{(i)} \ - \ \begin{pmatrix} A_{EI}^{(i,i)} & A_{EI}^{(i,i+1)} \end{pmatrix} \begin{pmatrix} A_{II}^{(i,i)^{-1}} & 0 \\ 0 & A_{II}^{(i+1,i+1)^{-1}} \end{pmatrix} \begin{pmatrix} A_{IE}^{(i,i)} \\ A_{IE}^{(i+1,i)} \end{pmatrix},$$

where $1 \le i \le n_E$, and

$$S_{i,i+1} = -A_{EI}^{(i,i+1)} \ A_{II}^{(i+1)^{-1}} \ A_{IE}^{(i+1,i+1)}, \quad 1 \le i < n_E,$$

$$S_{i,i-1} = -A_{EI}^{(i,i)} \ A_{II}^{(i)^{-1}} \ A_{IE}^{(i,i-1)}, \quad 1 < i \le n_E.$$

Our approach to approximate S^{-1}, i.e., to derive a preconditioner for S, is to construct a new Schur complement problem such that the effect of the coupling between the edge and subdomain problems, due to the terms $A_{EI}^{(i,i)}$ and $A_{EI}^{(i,i+1)}$, is reduced. This reduction of mutual coupling allows us to select a block diagonal matrix which provides an accurate preconditioner. The next subsection describes and justifies the technique we propose.

4.3 The Edge Preconditioner

The edge preconditioner is constructed in two main stages. In the first stage, we define a new Schur complement problem $\tilde{S}$, in terms of a hybrid coarse/fine grid. In the second stage, we obtain an efficient approximation of $\tilde{S}^{-1}$ which constitutes the edge preconditioner $\hat{S}^{-1}$.

The edge subproblem derived on the fine grid

$$A_{EI}\mathbf{u}_I + A_{EE}\mathbf{u}_E = \mathbf{f}_E \qquad (15)$$

is approximated with the subproblem

$$\hat{A}_{EI}\mathbf{u}_I + \hat{A}_{EE}\mathbf{u}_E = \mathbf{f}_E, \qquad (16)$$

which is derived using the edge grid, as described next. We define an *edge neighborhood* $\hat{\mathcal{E}}_i = [0,1] \times [\hat{y}_i - H, \hat{y}_i + H]$, for each edge $1 \leq i \leq n_E$, where $H = K\,h$, and $\hat{y}_i = iH + (i-1)h$. The *edge grid* associated with the ith edge is $X \times \{\hat{y}_i - H, \hat{y}_i, \hat{y}_i + H\}$. Figure 3 shows the edge grid for the bottom edge of a three-strip decomposition. For the ith edge, we approximate the restriction of u to $\hat{\mathcal{E}}_i$ by an element in the function space of piecewise bicubic polynomials on $\hat{\mathcal{E}}_i$, where the Hermite bicubic basis functions have support on translates of $[-h,h] \times [-H,H]$.

For each edge i, $1 \leq i \leq n_E$, we collocate at the points $\{(\xi,\zeta) : (\xi,\zeta) \in \mathcal{G} \cup \partial\mathcal{G}$ and $\hat{y}_i - h/2 < \zeta < \hat{y}_i + h/2\}$ in the new function space, obtaining the approximation to the ith edge subproblem:

$$\hat{A}_{EI}^{(i,i)}\mathbf{u}_I^{(i)} + \hat{A}_{EI}^{(i,i+1)}\mathbf{u}_I^{(i+1)} + \hat{A}_{EE}^{(i)}\mathbf{u}_E^{(i)} = \mathbf{f}_E^{(i)}.$$

Collecting all n_E edge subproblems yields (16), and a possible approximation

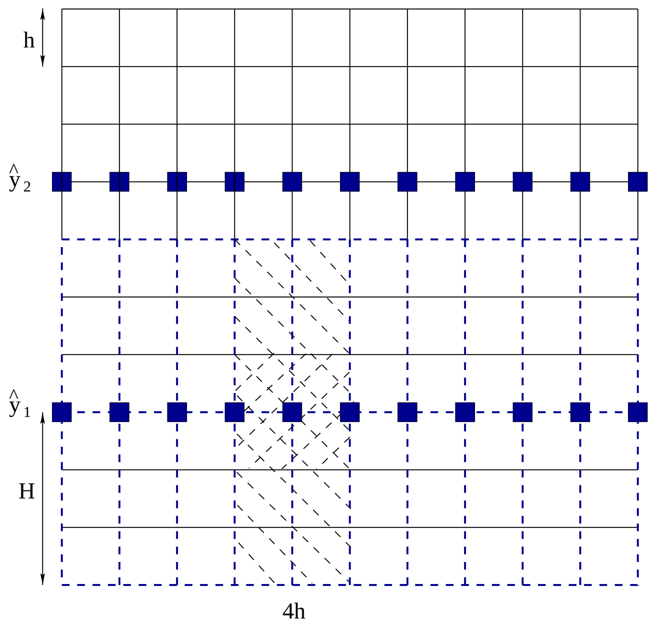

Figure 3: Support of the piecewise bicubics basis functions centered at the point $(4\,h, H)$. The support for edge grid discretization is the area hashed with south-east to north-west lines; the support for the original fine grid discretization is hashed with south-west to north-east lines. Edge nodes are shown as square boxes.

to the Schur complement S in (5):

$$\tilde{S} = \hat{A}_{EE} - \hat{A}_{EI} A_{II}^{-1} A_{IE}. \tag{17}$$

The block structure of $\tilde{S}$ is the same as that of S (see relation (14)). Note that there is still coupling between the edge subproblems and the subdomain interior subproblems, something we would like to avoid in a parallel algorithm. However, the important effect of using the edge grid is to reduce the magnitude of the coefficients in the matrix $\hat{A}_{EI}$ as compared to the coefficients in A_{EI}. This effect is quantized by the following lemma (see [11] for a proof):

Lemma 1 *There exists a positive number μ, independent of h and H, such that*

$$\hat{A}_{EE} - \tilde{S} = \frac{h}{H} \left(B_{EI} + \frac{h}{H} C_{EI} + D \right) P \, A_{II}^{-1} \, A_{IE},$$

where $A_{EI} = B_{EI} + C_{EI}$ is a splitting of A_{EI}, P is a permutation matrix, and D is a sparse matrix with at most eight nonzeros per row and $\|D\|_\infty \leq \mu$.

This means $\hat{A}_{EI} A_{II}^{-1} A_{IE}$ in (17) is smaller than the corresponding term $A_{EI} A_{II}^{-1} A_{IE}$ in (5). This fact motivates choosing $\hat{A}_{EE}$ as the edge preconditioner. In other words, $\hat{S}$ is derived from $\tilde{S}$ by setting $\hat{S}_{i,i} = \hat{A}_{EE}^{(i)}$, $\hat{S}_{i,i+1} = 0$, and $\hat{S}_{i,i-1} = 0$. Thus, our edge preconditioner is $\hat{S} = \hat{A}_{EE}$, i.e.,

$$\hat{S}^{-1} = \begin{pmatrix} \left(\hat{A}_{EE}^{(1)}\right)^{-1} & 0 & 0 & \dots & 0 \\ 0 & \left(\hat{A}_{EE}^{(2)}\right)^{-1} & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \left(\hat{A}_{EE}^{(n_E)}\right)^{-1} \end{pmatrix}. \tag{18}$$

4.4 The System Preconditioner

The approximation of A^{-1} is obtained from (6) and (18) as

$$M^{-1} = \begin{pmatrix} I & -A_{II}^{-1} A_{IE} \\ 0 & I \end{pmatrix} \begin{pmatrix} A_{II}^{-1} & 0 \\ 0 & \hat{S}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -\hat{A}_{EI} A_{II}^{-1} & I \end{pmatrix}.$$

We determine the LU factorization of $A_{II}^{(i)}$, where $1 \leq i \leq n_I$, and of $A_{EE}^{(i)}$, where $1 \leq i \leq n_E$. The LU factorization is computed with scaled partial pivoting, based on the `BAND GE` module in the `ELLPACK` package [12]. The factored blocks are stored and used to perform triangular solves.

An application of the preconditioner (see Algorithm 1) is the computation of $\mathbf{v} = M^{-1}\mathbf{u}$ for a given $\mathbf{u} = (\mathbf{u}_I, \mathbf{u}_E)^T$, and consists of a series of matrix-vector multiplications, triangular solves, and vector subtractions. Note that every step of the algorithm is fully parallel due to the block diagonal structure of A_{II} and $\hat{A}_{EE}$. Moreover, if the number of subdomains is $n_I = \Theta(\sqrt{N})$ ($f = \Theta(g)$ means $f = O(g)$ and $g = O(f)$), then the cost of the algorithm is $O(N)$ arithmetic operations. This can be seen as follows.

Algorithm 1 Preconditioner application: $\mathbf{v} = M^{-1}\mathbf{u}$.

```
▷ Purpose:   Compute $\mathbf{v} = M^{-1}\mathbf{u}$
▷ Inputs:    $\hat{S}$ and $A_{II}$ (factored), $\hat{A}_{EI}$, $A_{IE}$, and $\mathbf{u} = (\mathbf{u}_I, \mathbf{u}_E)^T$
▷            descriptors: $n_E, N_E$, and $n_I, N_I$
▷ Output:    $\mathbf{v} = M^{-1}\mathbf{u}$, where $\mathbf{v} = (\mathbf{v}_I, \mathbf{v}_E)^T$

1  $\mathbf{x}_I = A_{II}^{-1}\,\mathbf{u}_I$;                        ▷ Solve the harmonic problem
2  $\mathbf{x}_E = \hat{A}_{EI}\,\mathbf{x}_I$;                       ▷ Correction for right hand side of the interface
3  $\mathbf{y}_E = \mathbf{u}_E - \mathbf{x}_E$;                      ▷ Update right hand side on the interface
4  $\mathbf{v}_E = \hat{A}_{EE}^{-1}\,\mathbf{y}_E$;                  ▷ Solve on the interface
5  $\mathbf{y}_I = A_{IE}\,\mathbf{v}_E$;                             ▷ Harmonic extension of $\mathbf{v}_E$
6  $\mathbf{v}_I = A_{II}^{-1}(\mathbf{u}_I - \mathbf{y}_I)$;         ▷ Solve on all the subdomains
```

Let C be an $m \times m$ matrix with $l = u$, where l and u are the lower and upper bandwidth of C, respectively. The cost of a triangular solve for C is $3\,l\,m$. We have $l = \frac{2\sqrt{N}+4}{n_I}$ for $A_{II}^{(i)}, 1 \leq i \leq n_I$, and $l = 7$ for $\hat{A}_{EE}^{(j)}, 1 \leq j \leq n_E$. The cost of the lower and upper triangular solves is $6\,N_I \left(1 + 4\frac{\sqrt{N}+2}{n_I}\right)$ for $A_{II}^{(i)}$ and $42N_E$ for $\hat{A}_{EE}^{(j)}$. Because $\hat{A}_{EI}$ and A_{IE} have at most 16 nonzero entries per row, the cost of the matrix-vector multiplies $\hat{A}_{EI}\,\mathbf{x}_I$ and $A_{IE}\,\mathbf{v}_E$ for Algorithm 1 is $O(32\,n_E\,N_E)$ and $O(32\,n_I\,N_I)$, respectively. The combined cost of steps 2 and 5 of the algorithm is $O(n_E\,N_E + n_I\,N_I) = O(N)$. For $n_I = \Theta(\sqrt{N})$, the cost of an application of the preconditioner has the upper bound

$$O\left(N + 12\,N_I \left(n_I + 4(\sqrt{N} + 2)\right) + 42\,N_E\,n_E\right) = O(N),$$

The preconditioner is applied for each GMRES iteration, but is constructed only once. With the notation of the previous paragraph, the cost of factoring

C is $O(l^2 m)$. When the number of subdomains is $n_I = \Theta(\sqrt{N})$, the cost of factoring $A_{II}^{(i)}$ is $O(N_I)$ and the cost of factoring A_{EE} is $O(N_E)$. Since the cost of the edge grid discretization is $O(N)$, constructing the preconditioner has the cost $O(N + n_I\, N_I + n_E\, N_E) = O(N)$.

5 Numerical Experiments

In this section we illustrate the effectiveness of the proposed preconditioner, compare it to a block-Jacobi preconditioner and to threshold incomplete LU factorization (ILUT) [13], and investigate its performance as the problem size and the number of subdomains scale. We employ right-preconditioned GMRES(50), i.e., restarted GMRES using a Krylov subspace of dimension 50. The stopping criterion is relative residual reduction of $\epsilon = 10^{-5}$. The software is written in C and Fortran, and uses ELLPACK's band Gaussian elimination for the subdomain solves. The CPU time is given in seconds and is measured using the library function `times`. The tests are run on a Digital Personal Workstation 500au, with a 500-MHz Alpha processor, 2-MByte of cache, 512-MByte of main memory, and running Digital Unix V4.0D.

We use the following problems to evaluate the performance of the edge preconditioner (EP) (problems drawn from the examples in Appendix A of [12]):

Problem 1. Problem 1 in [12] has a self-adjoint operator

$$\frac{\partial}{\partial x}\left(e^{xy}\,\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(e^{-xy}\,\frac{\partial u}{\partial y}\right) - \frac{1}{1+x+y}u = g_1, \tag{19}$$

defined on the unit square, where the right-hand side g_1 is chosen so that the true solution is $u_1 = \frac{3}{4}e^{xy}\sin(\pi x)\sin(\pi y)$. The boundary conditions are Dirichlet on one side and Neumann on the other three sides: $u = 0$ on $x = 1$ and $\frac{\partial u}{\partial \mathbf{n}} = \frac{\partial u_1}{\partial \mathbf{n}}$ on $y = 0$, $y = 1$, and $x = 0$.

Problem 2. Problem 2 in [12] has a general operator

$$\frac{\partial^2 u}{\partial x^2} + (1+y^2)\,\frac{\partial^2 u}{\partial y^2} - \frac{\partial u}{\partial x} - (1+y^2)\,\frac{\partial u}{\partial y} = g_2, \tag{20}$$

defined on the unit square, where the right-hand side g_2 is determined by the solution $u_2 = e^{x+y} + (x^2 - x)^2 \log(1+y^2)$. The boundary conditions are again Dirichlet on one side and Neumann on the other three: $u = u_2$ on $x = 1$ and $\frac{\partial u}{\partial \mathbf{n}} = \frac{\partial u_2}{\partial \mathbf{n}}$ on $y = 0$, $y = 1$, and $x = 0$.

Problem 3. Problem 12 in [12] has oscillatory coefficients of $\frac{\partial u}{\partial x}$ and u:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + (1 + \sin(\alpha\, x))\frac{\partial u}{\partial x} - \cos(\alpha\, y)\, u = g_3, \tag{21}$$

and is also defined on the unit square, with g_3 determined by the solution $u_3 = \cos(\beta\, y) + \sin\beta\,(x - y)$. The boundary conditions are Dirichlet on all four sides: $u = u_3$ on $\partial\Omega$. For our experiments we set $\alpha = \beta = \pi$.

Recall that n_0 is the number of grid lines in each direction, n_I is the number of strips, $n_E = n_I - 1$ is the number of edges, K is the number of grid lines per strip and $n_0 = n_E + Kn_I$. Thus, for the edge-preconditioned method, $n_0 = (K+1)n_I - 1$. The grid for block-Jacobi (BJ) preconditioner is chosen so that each diagonal block has the same size. The largest problem solved has 56644 unknowns (119×119 grid).

First, we compare EP preconditioning to an incomplete LU preconditioner (ILUT from SPARSKIT [15]). Incomplete factorization preconditioners are a common choice for PDE solving. We use two criteria to compare the two preconditioners: the spectrum of the preconditioned matrix $A\,M^{-1}$, and the number of GMRES iterations. A spectral analysis is a commonly used quality measure for preconditioned iterative methods, e.g., see [10]. The spectrum analysis is made in terms of the number of *outlier eigenvalues*, i.e., eigenvalues of $A\,M^{-1}$ lying in the complex plane outside a disk of radius 0.1 and center (1,0). The parameters of ILUT are a fill level of 32 and a drop tolerance of 0.001. Figure 5 shows that, as the problem size increases, the gap between the number of outliers for EP and ILUT increases in favor of EP. In other words, the eigenvalues of EP are clustered around the unit eigenvalue closer than those of ILUT. Figure 5 compares the number of GMRES iterations for EP and ILUT. Due to tighter clustering of eigenvalues, EP significantly outperforms ILUT in terms of iteration count.

For the examples shown, it must be pointed out that despite the spectral comparison, the total time to solution for ILUT is actually better than for EP. For example, on Problem 1, with $n_0 = 39$ grid lines in each dimension (6084 unknowns), EP-preconditioned GMRES takes 1.62 seconds while ILUT-preconditioned GMRES takes 1.05 seconds. There are several points which must be kept in mind when interpreting these results, however. First, the cost per iteration of EP is generally higher than for ILUT since EP requires two subdomain solves per iteration (steps 1 and 6 of Algorithm 1) while ILUT requires only one solve.

Secondly, for our examples, the preconditioner setup time for EP is considerably higher than for ILUT, accounting for almost all of the difference between

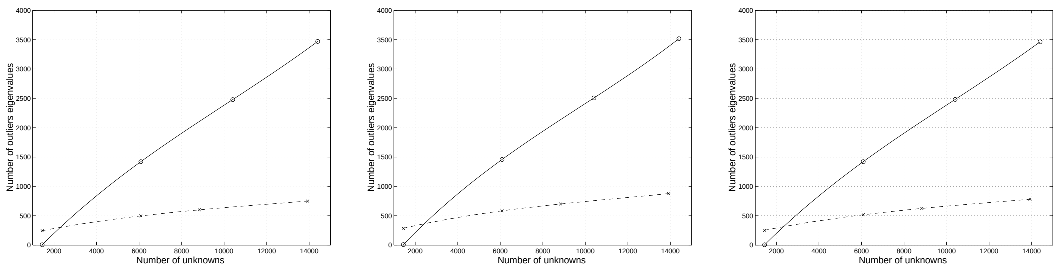

Figure 4: Spectrum comparison between ILUT with fill level of 32, drop tolerance 0.001 (solid lines), and EP with 4 strips (dashed lines), for Problems 1 (left), 2 (center) and 3 (right). Outliers are eigenvalues lying in the complex plane outside a disk of radius 0.1 and center $(1, 0)$.

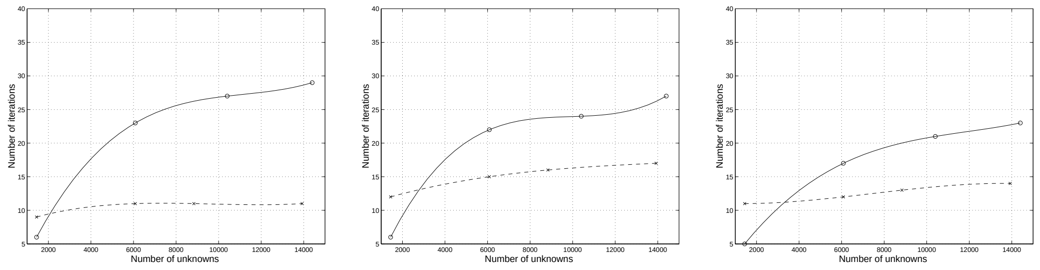

Figure 5: Convergence comparison between ILUT with fill level of 32, drop tolerance 0.001 (solid lines), and EP with 4 strips (dashed lines), for Problems 1 (left), 2 (center) and 3 (right).

the two methods. This is not surprising since, with small n_I, complete factorizations of the relatively large subdomain matrices $A_{II}^{(i)}$ are considerably more expensive than an incomplete factorization of A, especially with a low ILU fill-in parameter. However, using more subdomains reduces the size of the subdomain matrices and can reduce the overall cost of the EP setup phase. For our experiments, the number of subdomains was held constant at $n_I = 4$. If the number of subdomains scales with n_0, then the results for EP improve (see the complexity analysis in Section 4). For example, with $n_0 = 39$ and $n_I = 5$ subdomains, the time improves to 1.36 seconds. Furthermore, note that EP's advantages in iteration count will yield greater benefits for problems where the preconditioner can be re-used, so that the setup cost can be amortized over multiple solves, e.g., problems with multiple right hand sides, and nonlinear or time-dependent problems with an outer iteration.

A further advantage of EP over ILU is that the latter requires the user to select parameters. The performance of ILU can be quite sensitive to the choice of these parameters. For our experiments, we determined good choices experimentally, and these choices were very important to the success of the method. For example, with $n_0 = 39$, using a fill level of 16 instead of 32 causes the number of GMRES iterations to increase from 24 to 42; and a drop tolerance of 0.01 rather than 0.001 causes the number of iterations to increase from 24 to 45.

Finally, we note that EP is designed to be a highly parallelizable preconditioner, while ILU is not. Although the experiments reported here are all run on a sequential machine, our ultimate goal is to derive a highly parallel algorithm.

Since ILUT does not parallelize well we turn now to a comparison of EP with a preconditioner that does parallelize easily, namely block-Jacobi (BJ). Figures 6 and 7 show the EP versus BJ performance comparison results for Problem 1. Each curve in Figures 6 and 7 corresponds to a fixed number of subdomains or Jacobi blocks, with n_0 increasing. Notice that for the EP preconditioner the number of iterations is virtually independent of the problem size and grows very slowly with the number of subdomains. Clearly there is a big advantage for EP over BJ, and this advantage grows with problem size. Performance comparisons of BJ and EP for the other two problems yield results similar to those for Problem 1.

In Table 1 we compare further the block-Jacobi strategy versus the proposed edge preconditioner for Problems 1, 2, and 3. The focus here is on the influence of mixed (vs. Dirichlet) boundary conditions. We modify our test problems so that there is both a "Dirichlet" and a "mixed" boundary condition version for each. Specifically, the Dirichlet versions of Problems 1 and 2 are obtained by imposing $u = u_1$ and $u = u_2$, respectively, on $\partial\Omega$; the mixed boundary conditions version of Problem 3 is obtained by imposing $u = u_3$ on $x = 1$ and $\frac{\partial u}{\partial \mathbf{n}} = \frac{\partial u_3}{\partial \mathbf{n}}$ on $y = 0$, $y = 1$, and $x = 0$. Five subdomains were used for all the cases in Table 1; results with a different number of subdomains are similar. Note also that different n_0 values are used for BJ and EP because of the different constraints required by our implementation of the two preconditioners, i.e., BJ requires that the number of blocks evenly divide n_0, while EP requires that $n_0 = (K+1)n_I - 1$ for some constant K. Note that the EP data actually corresponds to a slightly larger problem than the one solved by BJ.

The results in Table 1 are evidence that EP can be more robust than BJ in the presence of mixed boundary conditions. For all the model problems, switching from Dirichlet to mixed boundary conditions causes a smaller relative rise in the number of EP iterations than in the number of BJ iterations. For example, for Problem 3, the number of EP iterations increases with 14%, while the number of BJ iterations almost doubles.

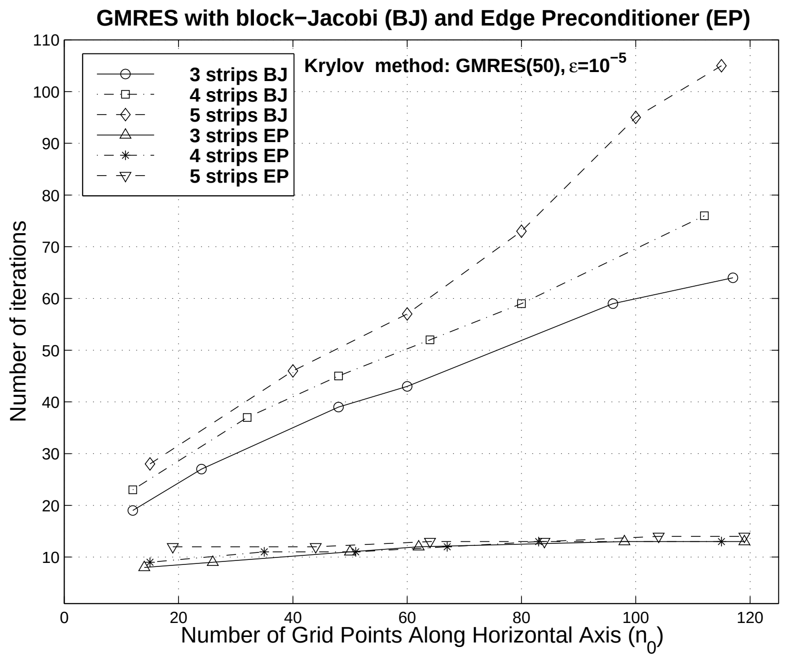

Figure 6: Number of GMRES iterations for block-Jacobi and edge preconditioning.

The scalability of the method with respect to the problem size and number of subdomains is investigated next. First, we keep the number of subdomains constant and increase the total problem size. In Figure 5 we saw that the number of iterations for the EP method was virtually independent of problem size. Table 2 shows the performance of the method for a larger data set, involving more subdomains and all three problems. Again we see that the number of iterations depends very weakly on the problem size.

Finally, we keep the global problem size fixed and increase the number of subdomains. Table 3 shows that the number of iterations increases slowly with the number of subdomains, up to a threshold value of n_I (which grows with n_0), when the dependence becomes significant. This is not surprising since as n_I increases, $K = H/h$ decreases, and hence the accuracy of the preconditioner decreases. However, from Table 3 it is evident that for a modest number of subdomains (e.g., 3–8) the number of iterations and especially the total time stay relatively flat. Since the EP algorithm is fully parallel, this suggests excellent performance is possible on typical shared memory parallel machines. Efficiently using 12 or more processors on problems of this size

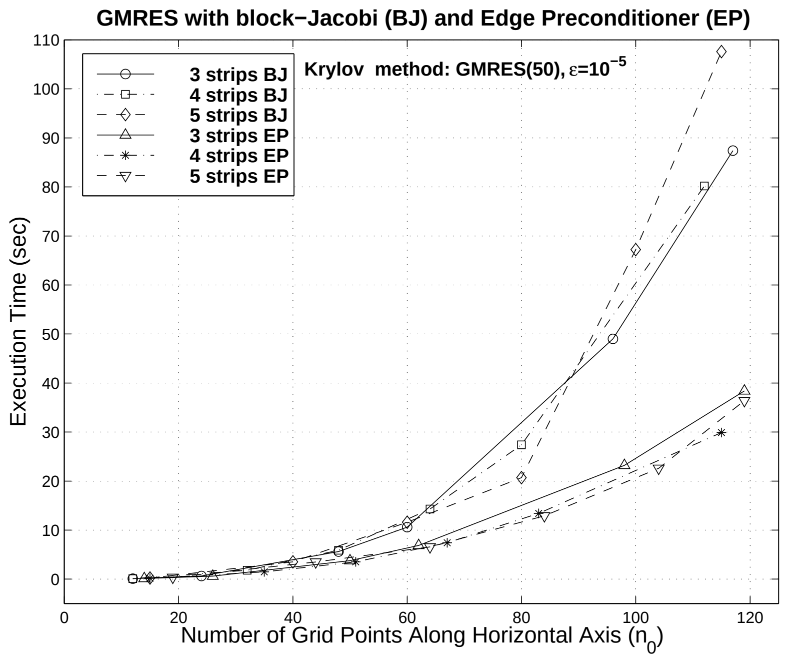

Figure 7: Solver times for block-Jacobi and edge preconditioning.

will likely require a more scalable two-dimensional decomposition strategy. A comparison of serial times between EP and ILUT shows ILUT is better, with the difference attributed to the setup time for EP. How the times for these two will compare for larger problems, in higher dimensions, and especially in a parallel context, is an important future research issue that needs to be resolved.

6 Concluding Remarks

We have introduced an interface preconditioner (EP) that is both accurate and efficient: accurate in that it is a good approximation to the original Schur complement, and efficient in that it is block diagonal, with one block corresponding to each edge subproblem. The preconditioner may provide an $O(N)$-cost solution, and the construction of the preconditioner requires $O(N)$ storage, where $N = 4n_0^2$ is the problem size. Furthermore, all subdomains and edge subproblems can be computed fully in parallel.

Our tests show that our preconditioner is significantly superior to block-Jacobi in terms of both number of iterations and execution time. The number

Table 1: Performance as a function of boundary condition type. Number of iterations (Its) and time in seconds.

Problem		Block Jacobi $n_0 = 95$		Edge $n_0 = 99$		Time Ratio
No.	Boundary cond.	Its	Time	Its	Time	
1	Dirichlet	55	35.0	10	14.8	2.37
	Mixed	83	51.6	14	20.0	2.58
2	Dirichlet	51	32.7	15	21.5	1.52
	Mixed	105	66.1	20	28.2	2.34
3	Dirichlet	49	30.8	14	20.1	1.53
	Mixed	90	56.1	16	22.7	2.47

Table 2: Performance of edge preconditioner for increasing problem size, with fixed number of subdomains. Number of iterations (Its) and time in seconds.

n_I	K	n_0	Problem 1		Problem 2		Problem 3	
			Its	Time	Its	Time	Its	Time
4	8	35	11	1.4	15	1.9	10	1.3
	11	47	11	2.8	16	4.0	12	3.0
	14	59	11	4.8	17	7.4	12	5.4
	17	71	12	8.4	18	12.3	13	9.1
	20	83	13	13.2	18	17.9	14	14.2
	23	95	13	18.6	18	25.9	14	20.0
6	5	35	14	1.6	17	2.0	11	1.3
	7	47	14	3.1	18	4.0	11	2.5
	9	59	14	5.3	18	6.7	12	4.6
	11	71	14	8.3	19	11.1	12	7.2
	13	83	15	12.8	20	16.9	13	11.3
	15	95	15	17.9	21	24.8	14	16.9

of iterations is virtually independent of the problem size for a fixed number of subdomains; for a fixed global problem size, the number of iterations and time depend only weakly on the number of subdomains as long as the granularity (number of grid lines per subdomain) is not too small. Furthermore, the number of iterations is small enough that the GMRES iteration does not have to be restarted, i.e., full GMRES is practical for the problems tested (with N

Table 3: Performance of edge preconditioner for increasing number of subdomains, with fixed total problem size. Number of iterations (Its) and time in seconds.

n_0	K	n_I	Problem 1		Problem 2		Problem 3	
			Its	Time	Its	Time	Its	Time
47	15	3	11	3.2	14	4.0	12	3.5
	11	4	11	2.8	16	4.0	12	3.1
	7	6	14	3.1	18	4.0	11	2.5
	5	8	20	4.2	23	4.8	14	3.0
	3	12	31	6.0	35	6.8	20	3.9
71	23	3	12	9.6	16	12.6	14	11.2
	17	4	12	8.3	18	11.3	13	9.1
	11	6	14	8.3	19	11.1	12	7.2
	8	8	19	9.9	23	11.9	14	7.5
	5	12	31	15.1	34	16.6	19	9.3
95	31	3	13	21.5	17	27.7	15	24.6
	23	4	13	18.6	18	25.3	14	20.1
	15	6	15	18.0	21	25.0	14	17.0
	11	8	19	20.2	23	24.4	14	15.2
	7	12	30	27.5	34	31.3	19	17.5

up to 56000 +) with a moderate residual reduction of 10^{-5}.

It is shown in [11] that the edge preconditioner introduced in this paper can be used to define a preconditioner for two-dimensional substructuring.

References

[1] B. Bialecki. A fast domain decomposition Poisson solver on a rectangle for Hermite bicubic orthogonal spline collocation. *SIAM J. Numer. Anal.*, 30(2):425–434, April 1993.

[2] B. Bialecki. Convergence analysis of orthogonal spline collocation for elliptic boundary value problems. *SIAM J. Numer. Anal.*, 35(2):617–631, April 1998.

[3] B. Bialecki, X.-C. Cai, M. Dryja, and G. Fairweather. An additive Schwarz algorithm for piecewise Hermite bicubic orthogonal spline collocation. In *Domain Decomposition Methods in Science and Engineering:*

The Sixth International Conference on Domain Decomposition, volume 157 of *Contemporary Mathematics*, pages 237–244. American Mathematical Society, 1994.

[4] B. Bialecki and M. Dryja. Multilevel additive and multiplicative methods for orthogonal spline collocation problems. *Numerische Mathematik*, 77(1):35–58, 1997.

[5] J. H. Bramble, J.E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring, I. *Mathematics of Computation*, 47(175):103–134, July 1986.

[6] T. F. Chan and T.P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[7] C. C. Christara and B. Smith. Multigrid and multilevel methods for quadratic spline collocation. *BIT*, 37(4):781–803, December 1997.

[8] W.R. Dyksen and J.R. Rice. A new ordering scheme for the hermite bicubic collocation equations. In *In Elliptic Problem Solvers III*, pages 467–480. Academic Press, 1984.

[9] E. N. Houstis, W. F. Mitchell, and J. R. Rice. Collocation software for second-order elliptic partial differential equations. *ACM Trans. Math. Software*, 11(4):379–412, December 1985.

[10] K. M. Irani, M. P. Kamat, C. J. Ribbens, H. F. Walker, and L. T. Watson. Experiments with conjugate gradient algorithms for homotopy curve tracking. *SIAM J. Optim.*, 1:222–251, 1991.

[11] Gabriel Mateescu. *Domain Decomposition Preconditioners for Hermite Collocation Problems*. PhD thesis, Virginia Tech, Blacksburg, VA, 1998.

[12] J. R. Rice and R. F. Boisvert. *Solving Elliptic Problems Using ELLPACK*. Springer-Verlag, New York, 1985.

[13] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994.

[14] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[15] Youcef Saad. SPARSKIT: A basic tool kit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990.