LOW RANK OFF-DIAGONAL BLOCK PRECONDITIONERS FOR SOLVING SPARSE LINEAR SYSTEMS ON PARALLEL COMPUTERS

RANDALL BRAMLEY AND VLADIMIR MEÑKOV
DEPARTMENT OF COMPUTER SCIENCE
INDIANA UNIVERSITY – BLOOMINGTON*

Abstract. For a sparse linear system $Ax = b$, preconditioners of the form $C = D + L + U$, where D is the block diagonal part of A (or incomplete factorization approximation of its blocks), and L and U are block strictly lower and upper triangular matrices composed of low-ranks approximations of the respective blocks of A, are examined. C is applied directly, by solving $Cz = w$, or partially, by applying one step of BSSOR to $Cz = w$.

Use of low-rank approximations of off-diagonal blocks is common in dense systems, but apparently has not been considered for sparse systems. This paper examines ways of defining the off-diagonal blocks and provides a detailed analysis for systems occuring in solving Laplace equation on a uniform mesh. Methods of applying C as a parallel preconditioner are proposed and analyzed, their cost being compared to that of applying a Jacobi preconditioner D. Testing results are presented, comparing the use of low-rank approximations with Jacobi and block SSOR preconditioning.

1. Introduction.

Solving a large linear system of equations

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

is a common subproblem encountered in the numerical solution of differential and integral equations. Typical methods for integral equations give *dense* matrices A, where most of the entries are nonzero. Differential equation solvers typically give *sparse* matrices A, where most of the entries are zero and not stored. The linear systems are usually solved with preconditioned conjugate gradient (PCG) like methods, which solve a related system

$$C^{-1}Ax = C^{-1}b,$$

where C is preconditioner. This involves solving an auxiliary system

$$Cu = v, \tag{2}$$

on each iteration.

An effective preconditioner is one which reduces the cost of the iterative solution, by reducing the number of iterations while keeping the cost per iteration reasonable. When the matrix A is large and dense, hierarchical methods are often used because it is impractical to compute and store the entire matrix. These methods consist of replacing the off-diagonal blocks of the matrix with low-rank approximations, which are naturally given by the application. For example, in graphics radiosity problems a scene is divided into a large number of patches, and the entry $A(i, j)$ gives the interaction between patches i and j. The contribution of a group of patches distant from a patch i can be

* WORK SUPPORTED BY NSF GRANTS CDA-9303189, ASC-9502292 AND ROME LABS AF 30602-92-C-0135

well-approximated by averaging the individual contributions from the group and then replacing the corresponding entries in the matrix with that average. This and related techniques can reduce the complexity of the matrix-vector product needed by PCG methods from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$.

In this paper we examine using low-rank approximations of off-diagonal blocks (LOB) for *sparse* matrices, to define preconditioners for PCG methods. In Section 3, two such preconditioners are proposed: direct LOB, whose preconditioner $C = D+L+U$ is obtained from A by replacing its ODBs with their low-rank approximations (LRA); and BSSOR LOB, which is one step of BSSOR applied to $Cz = w$.

Domain decomposed matrices from partial differential equations are used, where the off-diagonal blocks correspond to interfaces between subdomains. Construction of LOB preconditioners is discussed Section 4. Block SSOR preconditioners are examined with the diagonal blocks consisting of incomplete factorizations of the corresponding blocks of A, and the off-diagonal blocks replaced low-rank approximations. A difficulty with this approach is the lack of a way to naturally define those low rank approximations.

In Section 5.1, spectral properties of a direct LOB preconditioner for a sample problem are studied.

Advantages of the new preconditioners include more efficient computational kernels and reduced lengths of interprocessor communication messages when the blocks are distributed on a parallel machine. A new well-parallelizable method for solving $Bz = w$, using Sherman – Morrison – Woodbury formula and having some of the advantages of direct inverse approximations in its application, is described in Section 6. Discussion in Section 7 shows how this method is related to commonly used solution methods using Schur complement, and computational results are presented in Section 8.

1.1. Mesh notation. The kind of systems (1) that we are concerned with in this article are block systems, where partitioning of the matrix and vectors into blocks is derived from the partitioning of the geometrical domain Ω on which the differential equation is considered. The domain Ω is partitioned into p subdomains Ω_k $(k = 1, 2, \ldots, p)$, and each mesh node is assigned to one subdomain. Thus it is possible to use the same notation Ω_k both for the k-th subdomain, for the set of mesh nodes that are in Ω_k, and for the k-th group of variables in a vector-column (consisting of values of mesh function in the nodes of the subdomain Ω_k). The matrix A and the preconditioner C are partitioned into blocks conformally with this partitioning of vector columns.

We are considering only matrices A that are *structurally symmetric*, i.e. $a_{ij} \neq 0$ if $a_{ji} \neq 0$. Thus one can talk about the undirected graph of A, whose nodes correspond to variables, and in which nodes i and j are connected with an edge whenever $a_{ij} \neq 0$ [6].

To describe the structure of the matrix A, it is handy to use the predicate $\mathrm{neig}(\pi, \kappa)$, which is defined to be true if and only if the nodes π and κ are "neighbors" in the graph of A, i.e. if the $a_{\pi\kappa} \neq 0$.

Sometimes it is convenient to use the notion of "the set of nodes in Ω_k adjacent to

Ω_l", defined as

$$, _{kl} = \{\pi \mid \pi \in \Omega_k \ \wedge \ ((\exists \kappa \in \Omega_l)\ \mathrm{neig}(\pi, \kappa))\}.$$

The predicate $\mathrm{neig}(\Omega_k, \Omega_l)$ is true if and only if the two node groups Ω_k and Ω_l are adjacent in the graph of matrix A, i.e. $, _{kl} \neq \emptyset$.

2. Existing approaches. Various preconditioners have been proposed for iterative solving of block systems.

Block diagonal (Jacobi) preconditioning (C_2 with $Q = 0$) is typically used in parallel computations. If diagonal blocks are assigned to processors, the preconditioner can be applied without interprocessor synchronization or communication. Unfortunately, block diagonal preconditioned systems often fail to converge, especially for non-symmetric A. Block symmetric Gauss-Seidel (C_1 with Q_L and Q_U the corresponding parts of A) is significantly more robust, but the block lower and upper triangular system solves are inherently sequential. One way to recover parallelism is to increase the number of blocks in the partitioning of A and use a level scheduling [1, 2]. However, this also causes the quality of preconditioning to fall, and in the limiting case becomes pointwise symmetrized Gauss-Seidel, which typically provides poor quality of preconditioning.

In this paper we borrow an idea from the solution of integral equations, and use low-rank approximations of the off-diagonal blocks. By varying the rank of the ODBs the new method can be parametrized to vary in quality between that of block diagonal and block SSOR [12] preconditioner.

3. Proposed Approach. Advantages of a preconditioner may consist in its improved *quality* (as measured by the convergence rate of an iterative process) for some types of A; its *cost* (in terms of the number of operations or the execution time required to solve the system (2)); or in its *parallel properties*, as measured by the degree of utilization of the processors of a parallel computer on which (2) is solved.

Proposed preconditioners. In this article we are studying the properties of some block preconditioners that use low-rank approximations of the off-diagonal blocks of A. Two classes of such preconditioners are considered:

- Direct LOB preconditioning

  $$C = L + D + U, \tag{3}$$

  where D is a diagonal matrix composed of blocks that approximate in some way diagonal blocks of A, and L and U are strictly block lower and upper triangular matrix composed of blocks that are low-rank approximations of the ODBs of A.
- BSSOR LOB preconditioning

  $$C = (\omega L + D)D^{-1}(\omega U + D), \tag{4}$$

  where D is a diagonal matrix composed of blocks that approximate in some way diagonal blocks of A; L and U are strictly block lower and upper triangular matrix composed of blocks that are low-rank approximations of the ODBs of A, and ω is a real parameter.

Rank-related notation. For a given block matrix Q (such as $L+U$ in (3)), let $r_{kl} = \operatorname{rank} Q_{kl}$, let $M = \sum_{k,l} r_{kl}$ be the sum of ranks of all blocks of Q, let $m_l = \sum_k r_{kl}$ be the sum of ranks of the blocks of Q in the l-th block column, and let m be the maximum of the sum of the ranks of the blocks in any one block column or block row of Q:

$$m = \max\{\max_l m_l, \max_k \sum_l r_{kl}\}.$$

If no block of Q has rank higher than one, then M becomes the number of non-zero blocks in Q, and m the maximum number of non-zero blocks in any one row or column of Q.

If A is generated by a finite-difference or a finite-element method for a differential equation, the sum of ranks of the off-diagonal blocks in its k-th block row or block column is roughly equal to the number of mesh nodes adjacent to the borders of the k-th subdomain; the number of non-zero elements in the ODBs of the k-th block row or block column is typically of the same order of magnitude. This number is much smaller than the rank of diagonal blocks: for a subdomain of $n = \mathcal{O}(l^d)$ nodes in d-dimensional space, the number of the border nodes is $\mathcal{O}(l^{d-1}) = \mathcal{O}(n^{(d-1)/d})$. Thus if L and U are composed of the original ODBs of this A, $m = \mathcal{O}(n^{(d-1)/d})$, i.e. $m = \mathcal{O}(n^{1/2})$ in 2D or $m = \mathcal{O}(n^{2/3})$ in 3D.

In Section 4 we introduce low-rank approximation techniques that produce off-diagonal blocks of ranks lower than these estimates. Each parametrized technique introduces a family of approximations, so that the rank of off-diagonal blocks varies from zero (Jacobi method) to one (rank-one off-diagonal blocks) to the rank of the original ODBs of A. The number of iterations required is likely to decrease as the rank of the ODBs increases; on the other hand, with increasing rank the advantages of the method (reduced interprocessor communication and higher parallelism) gradually disappear. In general the BSSOR LOB preconditioner (4) does not promise fewer iterations than the incomplete block SSOR preconditioner

$$C = (\omega L_A + D)D^{-1}(\omega U_A + D), \tag{5}$$

where L_A and U_A are strictly block lower and upper triangular parts of A, respectively; after all, L and U in (4) are merely approximations of these L_A and U_A. Nor is BSSOR LOB (with any rank of the ODBs) significantly cheaper in operations per iteration than incomplete block SSOR, since the cost of computing with off-diagonal blocks is small compared with the cost of solving the system with the matrix D. However, solving the block triangular systems in BSSOR LOB can be efficiently parallelized, using a special case of an approach developed for the direct LOB method.

The direct LOB preconditioner may be of higher quality in incomplete block SSOR or BSSOR LOB with the same D and off-diagonal blocks of similar rank. Although its cost is higher than that of incomplete block SSOR, it is competitive, if the rank of the ODBs is low enough. Direct LOB also parallelizes well as shown in Section 6. For a particular approach to solving (4) and (3), Section 6.8 discuss the restrictions that need to be imposed on m for that approach to be efficient.

4. Types of the low-rank approximation. In both direct LOB and BSSOR LOB, the blocks of D are obtained by an incomplete LU factorization of the respective diagonal blocks of A; complete LU factorization is a special case of the incomplete one.

We have experimented with several methods for generating the off-diagonal blocks. In all cases, each non-zero off-diagonal block L_{kl} or U_{kl} was represented as a sum of rank-one terms $B_{kl} = \sum_{s=1}^{r_{kl}} u_{kl,s} v_{kl,s}^{\mathrm{T}}$, where

$$B_{kl} = \begin{cases} L_{kl}, & \text{if } k > l; \\ U_{kl}, & \text{if } k < l; \end{cases}$$

and the rank of the approximation (r_{kl}) varied.

Here we examine three methods for for defining the sets of vectors $u_{kl,s}$ and $v_{kl,s}$, so that the blocks B_{kl} approximate, in some way, off-diagonal blocks A_{kl} of A: singular value decomposition, lumping, and a projection method.

4.1. Singular Value Decomposition. For any real matrix A of rank r there are two sets of orthonormal real vectors $\{u_s\}_{s=1,\ldots,r}$ and $\{v_s\}_{s=1,\ldots,r}$ of appropriate dimensions, and a set of real positive numbers $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$ such that

$$A = \sum_{s=1}^{r} \sigma_s u_s v_s^{\mathrm{T}},$$

constituting its singular value decomposition [7]. Let us define, for any $M \leq r$, the partial sum X_M, involving the M largest singular values, as

$$A_M = \sum_{s=1}^{M} \sigma_s u_s v_s^{\mathrm{T}}, \tag{6}$$

For any $M \leq r$, A_M is the best (or at least *a best*) rank-M approximation of A in a number of senses, e.g. in any unitarily invariant matrix norm ([8, Example 7.4.52]), such as the matrix norm subordinate to the vector 2-norm or Frobenius norm. So an obvious choice of rank-M approximation of an off-diagonal block A_{kl} is

$$B_{kl} = (A_{kl})_M,$$

with the partial sum $(\cdot)_M$ defined as in (6).

This approximation is not practical, because calculating singular vectors is computationally expensive. Furthermore, the non-zero ODBs of matrices appearing in many typical discretized PDE systems don't have a few large singular values and many small ones. This implies that $(A_{kl})_M$ is not likely to be a very good approximation of A until M is close to $\mathrm{rank}\,(A_{kl})$. However, it provides a useful comparison with other methods.

4.2. "Lumping". "Averaging", or "lumping", is the simplest way to obtain a rank-one approximation of off-diagonal blocks of A. Let $e = [\,1 \quad \cdots \quad 1\,]^{\mathrm{T}}$. Suppose that for an ODB A_{kl} of A there is rank-one matrix B_{kl} such that

$$A_{kl}e \;=\; B_{kl}e \quad \text{and}$$

$$e^{\mathrm{T}} A_{kl} \;=\; e^{\mathrm{T}} B_{kl}. \tag{7}$$

PROPOSITION 4.1. *For a matrix A and vectors d_1 and d_2 of conformal dimensions, the set of solutions B of the system*

$$\begin{cases} d_1^{\mathrm{T}} A = d_1^{\mathrm{T}} B \\ A d_2 = B d_2 \\ \operatorname{rank} B = 1 \end{cases} \tag{8}$$

is characterized by five cases:

1. *If $d_1^{\mathrm{T}} A d_2 \neq 0$, system (8) has a unique solution*

$$B = \operatorname{Lump}_{d_1,d_2}(A) \equiv (A d_2 d_1^{\mathrm{T}} A)/(d_1^{\mathrm{T}} A d_2). \tag{9}$$

2. *If $d_1^{\mathrm{T}} A d_2 = 0$ but $d_1^{\mathrm{T}} A \neq 0$, then the set of solutions of (8) is*

$$\left\{ B = (u d_1^{\mathrm{T}} A)/(d_1^{\mathrm{T}} u),\ (\forall u \mid d_1^{\mathrm{T}} u \neq 0) \right\}.$$

3. *If $d_1^{\mathrm{T}} A d_2 = 0$ but $A d_2 \neq 0$, then the set of solutions of (8) is*

$$\left\{ B = (A d_2 v^{\mathrm{T}})/(v^{\mathrm{T}} d_2),\ (\forall v \mid v^{\mathrm{T}} d_2 \neq 0) \right\}.$$

4. *If $d_1^{\mathrm{T}} A = 0$ and $A d_2 = 0$, then the set of solutions of (8) is*

$$\left\{ B = uv,\ (\forall u \mid d_1^{\mathrm{T}} u = 0)(\forall v \mid v^{\mathrm{T}} d_2 = 0) \right\}.$$

5. *If $d_1^{\mathrm{T}} A d_2 = 0$ but $d_1^{\mathrm{T}} A \neq 0$ and $A d_2 \neq 0$, then (8) has no solution.*

Proof. In each case, substitution of the formula for B into the equations (8) shows that the given values of B satisfy the equations.

Let us prove now that there are no other solutions than indicated above. Since $\operatorname{rank} B = 1$, $B = uv^{\mathrm{T}}$ with some u and v. The first two equations of (8) give then

$$(d_1^{\mathrm{T}} u) v^{\mathrm{T}} \;=\; d_1^{\mathrm{T}} A \quad \text{and} \tag{10}$$

$$u(v^{\mathrm{T}} d_2) \;=\; A d_2. \tag{11}$$

If we define $c_1 = d_1^{\mathrm{T}} u$ and $c_2 = v^{\mathrm{T}} d_2$, then

$$s \equiv d_1^{\mathrm{T}} A d_2 = d_1^{\mathrm{T}} B d_2 = (d_1^{\mathrm{T}} u)(v^{\mathrm{T}} d_2) = c_1 c_2. \tag{12}$$

In case 1, $s \neq 0$, which means, due to (12), that both $c_1 \neq 0$ and $c_2 \neq 0$. It follows then from (10) and (11) that $v^{\mathrm{T}} = c_1^{-1} d_1^{\mathrm{T}} A$ and $u = c_2^{-1} A d_2$. From (12) we have $c_1 c_2 = s$, and thus $B = uv^{\mathrm{T}} = (A d_2 d_1^{\mathrm{T}} A)/(d_1^{\mathrm{T}} A d_2)$.

In case 2, it follows from $d_1^{\mathrm{T}} A \neq 0$ that $d_1^{\mathrm{T}} B = c_1 v^{\mathrm{T}} \neq 0$, and $c_1 = d_1^{\mathrm{T}} u \neq 0$.

In case 3, it follows from $A d_2 \neq 0$ that $B d_2 = u c_2 \neq 0$, and $c_2 = v^{\mathrm{T}} d_2 \neq 0$.

In case 4, the zero matrix $B = 0$ is a solution; any non-zero solution $B = uv^{\mathrm{T}}$ must have $u \neq 0$ and $v \neq 0$. Since $d_1^{\mathrm{T}} B = c_1 v^{\mathrm{T}} = 0$ and $B d_2 = u c_2 = 0$, we have $c_1 = 0$ and $c_2 = 0$, i.e. $d_1^{\mathrm{T}} u = 0$ and $v^{\mathrm{T}} d_2 = 0$.

In case 5, from $d_1^{\mathrm{T}} A = c_1 v^{\mathrm{T}} \neq 0$ we obtain $c_1 \neq 0$, and from $A d_2 = u c_2 \neq 0$ we obtain $c_2 \neq 0$. Therefore, by (12), $d_1^{\mathrm{T}} A d_2 = c_1 c_2 \neq 0$, but this contradicts the assumption that $d_1^{\mathrm{T}} A d_2 = 0$, and thus shows non-existence of a solution. $\square$

Proposition 4.1 shows that for any ODB A_{kl} of A such that $s(A_{kl}) \equiv e^{\mathrm{T}} A e \neq 0$, the unique rank-one matrix with the properties (7) is $B_{kl} = \mathrm{Lump}_{e,e}(A)$, given by (9).

In terms of the graph associated with the matrix A_{kl}, replacing A_{kl} with B_{kl} corresponds to cutting all edges connecting nodes of Ω_k with those of Ω_l, and connecting instead every border node i of Ω_k with every border node j of Ω_l.

An important property of the approximation $\mathrm{Lump}_{e,e}(A_{kl})$ is that blocks B_{kl} precisely approximate the respective blocks A_{kl} on any mesh function ϕ that is constant on subdomains; i.e., for any such function ϕ, for any of its blocks l,

$$B_{kl}\phi_l = A_{kl}\phi_l. \tag{13}$$

This is also true of any mesh function ϕ such that for any pair of contacting subdomains (Ω_k, Ω_l), its value is constant on the set of the "border nodes" , $_{kl}$.

The bounds of the spectrum of AC^{-1} where the preconditioner $C = D + L + U$, composed as in (3), has D composed of the diagonal blocks of A, and L and U composed of rank-one blocks of obtained with $\mathrm{Lump}_{e,e}(\cdot)$, are estimated for a class of matrices A later in this article (Section 5.1).

PROPOSITION 4.2. *If* $\mathrm{rank}\, A = 1$ *and* $d_1^{\mathrm{T}} A d_2 \neq 0$, *then* $\mathrm{Lump}_{d_1,d_2}(A) = A$.

Proof. Any rank-one matrix A can be represented as $A = uv^{\mathrm{T}}$. Substituting $A = uv^{\mathrm{T}}$ into (9) we obtain $\mathrm{Lump}_{d_1,d_2}(A) = uv^{\mathrm{T}} = A$. $\square$

Proposition 4.2 means that the approximation $\mathrm{Lump}_{e,e}(A_{kl})$ possesses the following property: if the block A_{kl} is already of the rank 1, then

$$\mathrm{Lump}_{e,e}(A_{kl}) = A_{kl}, \tag{14}$$

i.e. the lumping processes reconstructs the original rank-one off-diagonal block correctly. However, using $\mathrm{Lump}_{e,e}(\cdot)$ is not the only possible way to reconstruct a rank-one block. For arbitrary vectors d_1 and d_2 of appropriate dimensions, such that $d_1^{\mathrm{T}} A_{kl} d_2 \neq 0$, the lumping process $\mathrm{Lump}_{d_1,d_2}(\cdot)$, given by (9), has the same property (14).

Although the lumping method using $\mathrm{Lump}_{e,e}(\cdot)$ seems "physically natural" when e.g. off-diagonal elements of A are all non-positive or all non-negative, in some cases its meaningfulness is more problematic. When $e^{\mathrm{T}} A_{kl} e = 0$, it either cannot be used at all, or gives an ambiguous result. When $e^{\mathrm{T}} A_{kl} e$ is small (e.g. when $\left| e^{\mathrm{T}} A_{kl} e \right| \ll \|A_{kl}\|_\infty$), preconditioner quality is likely to be very poor (worse than that of an identity matrix)

4.3. Projection method. The lumping method can be generalized to higher ranks. Suppose the off-diagonal block A_{kl} is $n_k \times n_l$. For a given M-dimensional subspace $\mathcal{X}$ of mesh functions on Ω_l, and for a given M-dimensional subspace $\mathcal{Y}$ of mesh functions on Ω_k, consider the following problem: Construct a block B_{kl}, of lowest possible rank so that for every $\phi \in \mathcal{X}$, $B_{kl}\phi = A_{kl}\phi$, and for every $\psi \in \mathcal{Y}$ $B_{kl}^{\mathrm{T}}\psi = A_{kl}^{\mathrm{T}}\psi$.

In matrix terms these conditions are: Let X be an $n_l \times M$ matrix such that its columns constitute a basis in $\mathcal{X}$:

$$\mathcal{X} = \mathrm{span}\,\{\vec{x}_1, \ldots, \vec{x}_M\};$$

Let Y be an $n_k \times M$ matrix such that its columns constitute a basis in $\mathcal{Y}$:

$$\mathcal{Y} = \operatorname{span}\{\vec{y}_1, \ldots, \vec{y}_M\}.$$

Find a matrix B_{kl} of the lowest possible rank such that

$$B_{kl}X = A_{kl}X, \quad \text{and} \tag{15}$$

$$Y^{\mathrm{T}}B_{kl} = Y^{\mathrm{T}}A_{kl}. \tag{16}$$

The lumping method conditions are a special case of conditions (15,16), with $\mathcal{X} = \operatorname{span}\{e\}$ and $\mathcal{Y} = \operatorname{span}\{e\}$.

The following generalization of Proposition 4.1 describes the solution(s) of (15,16).

THEOREM 4.3. *Let $\mathcal{X}$ be a subspace of $\mathbf{R}^{n_x}$, $\mathcal{Y}$ be a subspace of $\mathbf{R}^{n_y}$, and $A \in \mathbf{R}^{n_y \times n_x}$. Let $m_x = \dim \mathcal{X}$, $m_y = \dim \mathcal{Y}$, $r_x = \dim A\mathcal{X}$, $r_y = \dim A^{\mathrm{T}}\mathcal{Y}$. Let $r = \operatorname{rank} Y^{\mathrm{T}}AX$, where X is a matrix whose columns constitute a basis in $\mathcal{X}$, and Y is a matrix whose columns constitute a basis in $\mathcal{Y}$.*

Let[1]

$$\begin{aligned}
\mathcal{X}_4 &= \mathcal{X}^{\perp} \quad \textit{(the orthogonal complement to } \mathcal{X} \textit{ in } \mathbf{R}^{n_x}\textit{)},\\
\mathcal{X}_3 &= \ker(A|_{\mathcal{X}})\\
\mathcal{X}_{1\text{-}2} &= \{x \mid (x \in \mathcal{X}) \wedge (x \perp \mathcal{X}_3)\} \quad \textit{(the orthogonal complement to } \mathcal{X}_3 \textit{ in } \mathcal{X}\textit{)},\\
\mathcal{X}_2 &= \{x \mid (x \in \mathcal{X}_{1\text{-}2}) \wedge (Ax \perp \mathcal{Y})\},\\
\mathcal{X}_1 &= \{x \mid (x \in \mathcal{X}_{1\text{-}2}) \wedge (x \perp \mathcal{X}_2)\} \quad \textit{(the orthogonal complement in } \mathcal{X}_{1\text{-}2} \textit{ to } \mathcal{X}_2\textit{)}.
\end{aligned}$$

In $\mathcal{Y}$, the four subspaces $\mathcal{Y}_1$, $\mathcal{Y}_2$, $\mathcal{Y}_3$, and $\mathcal{Y}_4$ are defined in a similar way.

Let P_X and P_Y be the orthogonal projectors onto $\mathcal{X}$ and $\mathcal{Y}$, and P_{X_i}, P_{Y_i} be the orthogonal projectors onto $\mathcal{X}_i$ and $\mathcal{Y}_i$, $i = 1, 2, 3, 4$.

Under these conditions, for any matrix B satisfying the conditions

$$B|_{\mathcal{X}} = A|_{\mathcal{X}}, \quad \textit{and} \tag{17}$$

$$B^{\mathrm{T}}|_{\mathcal{Y}} = A^{\mathrm{T}}|_{\mathcal{Y}} \tag{18}$$

$\operatorname{rank} B \geq r_B \equiv r_x + r_y - r$, and all rank-$r_B$ matrices B satisfying (17,18) are of the form $B = B_0 + B_1 + B_2$, with

$$B_0 = A(P_Y A P_X)^{+} A, \tag{19}$$

$$B_1 = AP_{X_2}(I + R_X P_{X_4}), \tag{20}$$

$$B_2 = (I + P_{Y_4} R_Y) P_{Y_2} A, \tag{21}$$

for some matrices $R_X \in \mathbf{R}^{n_x \times n_x}$, $R_Y \in \mathbf{R}^{n_y \times n_y}$.

Proof. It follows from the definition of $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$, and $\mathcal{X}_4$ that they are mutually orthogonal subspaces, and their direct sum is the entire $\mathbf{R}^{n_x}$: $\mathbf{R}^{n_x} = \mathcal{X}_1 \oplus \mathcal{X}_2 \oplus \mathcal{X}_3 \oplus \mathcal{X}_4$.

[1] For a vector u in a vector space and a subspace $\mathcal{V}$ of the same space, we use the notation $u \perp \mathcal{V}$ to mean that $u^{\mathrm{T}}v = 0$ for every $v \in \mathcal{V}$; similarly, $\mathcal{U} \perp \mathcal{V}$ means that for every $u \in \mathcal{U}$ and for every $v \in \mathcal{V}$, $u^{\mathrm{T}}v = 0$.

Similarly, $\mathcal{Y}_1$, $\mathcal{Y}_2$, $\mathcal{Y}_3$, and $\mathcal{Y}_4$ are mutually orthogonal and $\mathbf{R}^{n_y} = \mathcal{Y}_1 \oplus \mathcal{Y}_2 \oplus \mathcal{Y}_3 \oplus \mathcal{Y}_4$. We can therefore construct an orthonormal basis in $\mathbf{R}^{n_x}$, consisting of $\dim \mathcal{X}_1$ vectors in $\mathcal{X}_1$, $\dim \mathcal{X}_2$ vectors in $\mathcal{X}_2$, $\dim \mathcal{X}_3$ vectors in $\mathcal{X}_3$, and $\dim \mathcal{X}_4$ vectors in $\mathcal{X}_4$. Let $X_{1\text{-}2\text{-}3\text{-}4} = [\, X_1, \quad X_2, \quad X_3, \quad X_4\,]$ be the orthogonal matrix composed of these vectors. Also construct a similar basis in $\mathbf{R}^{n_y}$, with $Y_{1\text{-}2\text{-}3\text{-}4} = [\, Y_1, \quad Y_2, \quad Y_3, \quad Y_4\,]$ composed of its vectors.

We will also need matrices of the same dimensions as $X_{1\text{-}2\text{-}3\text{-}4}$ and $Y_{1\text{-}2\text{-}3\text{-}4}$, in which some of the block columns are replaced with zeros. The following notation is used: $X_{1\text{-}2\text{-}3\text{-}0} = [\, X_1, \quad X_2, \quad X_3, \quad 0\,]$, $X_{0\text{-}2\text{-}0\text{-}0} = [\, 0, \quad X_2, \quad 0, \quad 0\,]$, $X_{0\text{-}0\text{-}0\text{-}4} = [\, 0, \quad 0, \quad 0, \quad X_4\,]$, etc. Their products form orthogonal projectors onto subspaces, e.g. $X_{0\text{-}2\text{-}0\text{-}0} X_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} = X_{0\text{-}2\text{-}0\text{-}0} X_{0\text{-}2\text{-}0\text{-}0}^{\mathrm{T}} = P_{X_2}$. For compactness, the notation $X_{1\text{-}2\text{-}3} = [\, X_1, \quad X_2, \quad X_3\,]$, $X_{1\text{-}2} = [\, X_1, \quad X_2\,]$, etc. will also be used.

Note that $\dim \mathcal{X}_4 = n_x - \dim \mathcal{X} = n_x - m_x$, and $\dim \mathcal{X}_3 = \dim \ker(A|_{\mathcal{X}}) = \dim \mathcal{X} - \dim(A\mathcal{X}) = m_x - r_x$. Thus $\dim \mathcal{X}_{1\text{-}2} = \dim \mathcal{X} - \dim \mathcal{X}_3 = r_x$. Similarly $\dim \mathcal{Y}_4 = n_y - m_y$, $\dim \mathcal{Y}_3 = m_y - r_y$, and $\dim \mathcal{Y}_{1\text{-}2} = r_y$.

Consider the form A' the matrix A assumes in the new bases:

$$A' \equiv Y_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} A X_{1\text{-}2\text{-}3\text{-}4} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} = \begin{bmatrix} A_{11} & 0 & 0 & A_{14} \\ 0 & 0 & 0 & A_{24} \\ 0 & 0 & 0 & 0 \\ A_{41} & A_{42} & 0 & A_{44} \end{bmatrix},$$

where $A_{ij} = Y_i^{\mathrm{T}} A X_j$, $i, j = 1, \ldots 4$. (To simplify expressions, we omit prime symbols over the symbols for the blocks of A' and B'). The third block row and the third block column are composed of zeros because, by definition of $\mathcal{X}_3$, $A\mathcal{X}_3 = \{0\}$, and, by definition of $\mathcal{Y}_3$, $A^{\mathrm{T}}\mathcal{X}_4 = \{0\}$. Blocks A_{12} and A_{22} are zeros because, by definition of $\mathcal{X}_2$, $A\mathcal{X}_2 \perp \mathcal{Y}$; block A_{21} is zero because, by definition of $\mathcal{Y}_2$, $A^{\mathrm{T}}\mathcal{Y}_2 \perp \mathcal{X}$.

Consider now the $m_y \times m_x$ top-left-corner submatrix of A',

$$A'_{1\text{-}2\text{-}3,1\text{-}2\text{-}3} = Y_{1\text{-}2\text{-}3}^{\mathrm{T}} A X_{1\text{-}2\text{-}3} = \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Since the columns of $Y_{1\text{-}2\text{-}3}$ form a basis in $\mathcal{Y}$, and those of $X_{1\text{-}2\text{-}3}$ form a basis in $\mathcal{X}$, it follows from the definition of r that

$$\operatorname{rank} A'_{1\text{-}2\text{-}3,1\text{-}2\text{-}3} = \operatorname{rank} A_{11} = r. \tag{22}$$

Moreover, this rank is the number of linearly independent columns in $A'_{1\text{-}2\text{-}3,1\text{-}2\text{-}3}$, and thus

$$r = \dim \operatorname{Ran}\left((Y_{1\text{-}2\text{-}3}^{\mathrm{T}} A)|_{\mathcal{X}}\right), \tag{23}$$

On the other hand, $\mathcal{X}_3 \oplus \mathcal{X}_2 = \ker((Y_{1\text{-}2\text{-}3}^{\mathrm{T}} A)|_{\mathcal{X}})$, which means that $\dim \mathcal{X}_3 + \dim \mathcal{X}_2 = \dim \ker((Y_{1\text{-}2\text{-}3}^{\mathrm{T}} A)|_{\mathcal{X}}) = \dim \mathcal{X} - \dim \operatorname{Ran}\left((Y_{1\text{-}2\text{-}3}^{\mathrm{T}} A)|_{\mathcal{X}}\right)$, i.e. $\dim \mathcal{X}_2 = \dim \mathcal{X} - \dim \mathcal{X}_3 -$

$\dim \mathrm{Ran}\,((Y^{\mathrm{T}}_{1\text{-}2\text{-}3}A)|_{\mathcal{X}}) = r_x - \dim \mathrm{Ran}\,((Y^{\mathrm{T}}_{1\text{-}2\text{-}3}A)|_{\mathcal{X}}) - \dim \mathcal{X}_3$. Due to (23) this gives $\dim \mathcal{X}_2 = r_x - r$ and

$$\dim \mathcal{X}_1 = \dim \mathcal{X}_{1\text{-}2} - \dim \mathcal{X}_2 = r. \tag{24}$$

Similarly, $\dim \mathcal{Y}_2 = r_y - r$ and

$$\dim \mathcal{Y}_1 = r. \tag{25}$$

Equations (22), (24) and (25) indicate that A_{11} is a full-rank, i.e. non-singular, square matrix.

The columns of the $(n_y - m_y) \times (r_x - r)$ block A_{42} are linearly independent; otherwise the r_x columns of

$$Y^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} A X_{1\text{-}2} = \begin{bmatrix} A_{11} & 0 \\ 0 & 0 \\ 0 & 0 \\ A_{41} & A_{42} \end{bmatrix}$$

won't be independent, and they must be, since $\mathrm{rank}\,(Y^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} A X_{1\text{-}2}) = \mathrm{rank}\,(Y^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} A X_{1\text{-}2\text{-}3}) = \mathrm{rank}\,(A X_{1\text{-}2\text{-}3}) = \dim(A\mathcal{X}) = r_x$. Similarly, the rows of the $(r_y - r) \times (n_y - m_y)$ block A_{24} are independent as well.

Condition (17) can be restated in the new basis as "the first 3 block columns of $B' \equiv Y^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} B X_{1\text{-}2\text{-}3\text{-}4}$ are the same as those of A'"; condition (18) means "the first 3 block rows of B' are the same as those of A'". To satisfy them, a B must have, in the new bases, the form

$$B' \equiv Y^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} B X_{1\text{-}2\text{-}3\text{-}4} = \begin{bmatrix} A_{11} & 0 & 0 & A_{14} \\ 0 & 0 & 0 & A_{24} \\ 0 & 0 & 0 & 0 \\ A_{41} & A_{42} & 0 & B_{44} \end{bmatrix}.$$

Conversely, for any B_{44}, $B = Y^{-\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} B' X^{-1}_{1\text{-}2\text{-}3\text{-}4}$ with B' as above will satisfy (17,18). Any B' can thus be given by

$$B' = B'_0 + B'_1 + B'_2 + R',$$

where

$$B'_0 = \begin{bmatrix} A_{11} \\ 0 \\ 0 \\ A_{41} \end{bmatrix} A_{11}^{-1} \begin{bmatrix} A_{11} & 0 & 0 & A_{14} \end{bmatrix} = Y^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} A X_1 (Y_1^{\mathrm{T}} A X_1)^{-1} Y_1^{\mathrm{T}} A X^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4},$$

$$B'_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & A_{42} & 0 & 0 \end{bmatrix} = Y^{\mathrm{T}}_{1\text{-}2\text{-}3\text{-}4} A X_{0\text{-}2\text{-}0\text{-}0}, \quad B'_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{24} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = Y^{\mathrm{T}}_{0\text{-}2\text{-}0\text{-}0} A X_{1\text{-}2\text{-}3\text{-}4},$$

$$R' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_{44} \end{bmatrix},$$

with an arbitrary $R_{44} = B_{44} - A_{41}A_{11}^{-1}A_{14}$.

Analyzing the splitting of B', we see that $\operatorname{rank} B'_0 = \operatorname{rank} A_{11} = r$, $\operatorname{rank} B'_1 = \operatorname{rank} A_{42} = r_x - r$, and $\operatorname{rank} B'_2 = \operatorname{rank} A_{24} = r_y - r$. Because A_{11} is a full-rank block, it is impossible to express any non-zero column of B'_1 as a linear combination of columns of B'_0, nor is it possible to express any non-zero row of B'_2 as a linear combination of rows of B'_0. Thus $\operatorname{rank}(B'_0 + B'_1 + B'_2) = r_B \equiv r + (r_x - r) + (r_y - r) = r_x + r_y - r$.

Since any R_{44} will keep (17,18) satisfied and cannot lower the rank of B', we choose R_{44} to avoid increasing the rank of B'. This can only be done by adding linear combinations of columns of A_{42} to columns of R_{44}, and rows of A_{24} to rows of R_{44}. So choosing

$$R_{44} = A_{42}Q_{24} + Q_{42}A_{24}$$

with arbitrary Q_{24} and Q_{42} gives for R'

$$R' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_{44} \end{bmatrix} = B'_1 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{24} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & Q_{42} & 0 & 0 \end{bmatrix} B'_2 =$$

$$Y_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} A X_{0\text{-}2\text{-}0\text{-}0} X_{0\text{-}2\text{-}0\text{-}0}^{\mathrm{T}} R_X X_{0\text{-}0\text{-}0\text{-}4} + Y_{0\text{-}0\text{-}0\text{-}4}^{\mathrm{T}} R_Y Y_{0\text{-}2\text{-}0\text{-}0} Y_{0\text{-}2\text{-}0\text{-}0}^{\mathrm{T}} A X_{1\text{-}2\text{-}3\text{-}4},$$

with arbitrary $R_X \in \mathbf{R}^{n_x \times n_x}$, $R_Y \in \mathbf{R}^{n_y \times n_y}$.

Return to the original bases in $\mathbf{R}^{n_x}$ and $\mathbf{R}^{n_y}$, using the formula $B = {Y_{1\text{-}2\text{-}3\text{-}4}}^{-\mathrm{T}} B' X_{1\text{-}2\text{-}3\text{-}4}^{-1} = Y_{1\text{-}2\text{-}3\text{-}4} B' X_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}}$. In the sum $B = B_0 + B_1 + B_2$, the first term is

$$B_0 = Y_{1\text{-}2\text{-}3\text{-}4} B'_0 X_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} = A X_1 (Y_1^{\mathrm{T}} A X_1)^{-1} Y_1^{\mathrm{T}} A, \tag{26}$$

Since the last three block rows and the last three block columns of $Y_{1\text{-}2\text{-}3\text{-}0}^{\mathrm{T}} A X_{1\text{-}2\text{-}3\text{-}0}$ (as well as, therefore, of $(Y_{1\text{-}2\text{-}3\text{-}0}^{\mathrm{T}} A X_{1\text{-}2\text{-}3\text{-}0})^{+} Y_{1\text{-}2\text{-}3\text{-}4} A$) are composed of all zeros, equation (26) can be re-written as

$$B_0 = A X_{1\text{-}2\text{-}3\text{-}4} (Y_{1\text{-}2\text{-}3\text{-}0}^{\mathrm{T}} A X_{1\text{-}2\text{-}3\text{-}0})^{+} Y_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} A, \tag{27}$$

Since for an orthogonal matrix S and any matrix Q, $(SQ)^{+}S = Q^{+}$, (27) gives $B_0 = A X_{1\text{-}2\text{-}3\text{-}4} X_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} (Y_{1\text{-}2\text{-}3\text{-}4} Y_{1\text{-}2\text{-}3\text{-}0}^{\mathrm{T}} A X_{1\text{-}2\text{-}3\text{-}0} X_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}})^{+} Y_{1\text{-}2\text{-}3\text{-}4} Y_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} A$. Since $Y_{1\text{-}2\text{-}3\text{-}0} Y_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} = P_Y = P_Y^{\mathrm{T}}$ and $X_{1\text{-}2\text{-}3\text{-}0} X_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} = P_X$, this in its turn gives (19).

Combining the expression for B'_1 and the first term of the expression for R' gives

$$B_1 = A X_{0\text{-}2\text{-}0\text{-}0} (I + X_{0\text{-}2\text{-}0\text{-}0}^{\mathrm{T}} R_X X_{0\text{-}0\text{-}0\text{-}4}) X_{1\text{-}2\text{-}3\text{-}4}^{\mathrm{T}} = A P_{X_2} (I + R_X P_{X_4}),$$

i.e. (20). Similarly,

$$B_2 = Y_{1\text{-}2\text{-}3\text{-}4}(I + Y^{\mathrm{T}}_{0\text{-}0\text{-}0\text{-}4} R_Y Y_{0\text{-}2\text{-}0\text{-}0}) Y^{\mathrm{T}}_{0\text{-}2\text{-}0\text{-}0} A = (I + P_{Y_4} R_Y) P_{Y_2} A,$$

i.e. (21), is obtained from the expression for B_2' and the second term of the expression for R'. □

It follows from the definition of r in Theorem 4.3 that it is also the rank of the restriction of the quadratic form $h(y, x) \equiv y^{\mathrm{T}} A x$ on $Y \times X$. The following corollary shows that if the restriction of $h(y, x) \equiv y^{\mathrm{T}} A x$ on $\mathcal{Y}_{1-2} \times \mathcal{X}_{1-2}$ is a full-rank quadratic form, the solution of (15,16) will be unique.

COROLLARY 4.4. *Under conditions of Theorem 4.3, if $r = r_x = r_y$, then minimum rank of a matrix B satisfying (17,18) is r; such matrix is unique and is given by*

$$B = A(P_Y A P_X)^{+} A. \tag{28}$$

Proof. Since in this case $\mathcal{X}_2 = \{0\}$ and $\mathcal{Y}_2 = \{0\}$, it follows from Theorem 4.3 that $r_B = r$, the minimum-rank B is unique and is given by (28). □

COROLLARY 4.5. *Under conditions of Theorem 4.3, if $r_x = r_y = r$, and if X_b is an $n_x \times r$ matrix composed of columns constituting a basis in X_1 and Y_b is an $n_y \times r$ matrix composed of columns constituting a basis in Y_1, then*

$$B = A X_b (Y_b^{\mathrm{T}} A X_b)^{-1} Y_b^{\mathrm{T}} A. \tag{29}$$

Proof. As in Corollary 4.4, since $r = r_x = r_y$, there are no terms B_1 and B_2 in B.

Since the set of columns of X_1 and the set of columns of X_b each constitute a basis in $\mathcal{X}_1$, there exists a non-singular square matrix Q_x such that $X_1 = X_b Q_x$; similarly, $Y_1 = Y_b Q_y$ with some non-singular square matrix Q_y. Substituting this into (26) we have

$$B = B_0 = A X_b Q_x (Q_y^{\mathrm{T}} Y_b^{\mathrm{T}} A X_b Q_x)^{-1} Q_y^{\mathrm{T}} Y_b^{\mathrm{T}} A = A X_b (Y_b^{\mathrm{T}} A X_b)^{-1} Y_b^{\mathrm{T}} A,$$

i.e. (29). □

The following corollary simplifies the solution of (15,16) when the restriction of $\alpha(y, x) \equiv y^{\mathrm{T}} A x$ on $\mathcal{Y} \times \mathcal{X}$ is a full-rank quadratic form.

COROLLARY 4.6. *Under conditions of Theorem 4.3, if matrices X and Y consist of columns that form bases in $\mathcal{X}$ and $\mathcal{Y}$, and $m_x = m_y = r$ (i.e. $H \equiv Y^{\mathrm{T}} A X$ is square and non-singular), the minimum-rank solution of (17,18) has rank r, is unique and is given by*

$$B = A X^{\mathrm{T}} (Y^{\mathrm{T}} A X)^{-1} Y^{\mathrm{T}} A. \tag{30}$$

Proof. Since in this case $\mathcal{X} = \mathcal{X}_1$ and $\mathcal{Y} = \mathcal{Y}_1$, equation (29) in Corollary 4.5 becomes (30). □

The result given in Corollary 4.6 can be also proven independently of Theorem 4.3; see Appendix A for a simple direct proof.)

4.3.1. A choice of Y. The subspaces $\mathcal{X}$ and $\mathcal{Y}$ can be chosen independently. However, setting

$$Y = A_{kl}X. \tag{31}$$

results in the matrix H being Hermitian and nonnegative definite. The following theorem gives a formula for B_{kl} for such $\mathcal{Y}$.

THEOREM 4.7. *Under conditions of Theorem 4.3, if $\mathcal{Y} = A\mathcal{X}$, the minimum-rank solution of solution of (17,18) has rank r, is unique, and is given by*

$$B = P_{A\mathcal{X}}A = AX((AX)^{\mathrm{T}}AX)^{+}(AX)^{\mathrm{T}}A, \tag{32}$$

where $P_{A\mathcal{X}}$ is the orthogonal projector onto $A\mathcal{X}$, and X is a matrix whose columns are vectors of a basis in $\mathcal{X}$.

Proof. It follows from $\mathcal{Y} = A\mathcal{X}$ that $\mathcal{X}_2 = \{0\}$ (there are no $x \in \mathcal{X}$ such that $Ax \neq 0$ but $Ax \perp \mathcal{Y}$), and $\mathcal{Y}_2 = \mathcal{Y}_3 = \{0\}$ (for each $y \in \mathcal{X} \setminus \{0\}$ there is an $x \in X$ such that $y^{\mathrm{T}}x \neq 0$). We can thus use Corollary 4.6.

Since $\mathcal{X}_2 = \{0\}$, $\mathcal{X} = \mathcal{X}_1 + \mathcal{X}_3 = \mathcal{X}_1 + \ker(A|_{\mathcal{X}})$; therefore if the columns of X_b constitute a basis in $\mathcal{X}_1$, then the columns of AX_b will constitute a basis in Y. Since $\mathcal{Y}_2 = \mathcal{Y}_3 = \{0\}$, $Y = Y_1$, and the columns of AX_b are thus a basis in Y_1. Now by (29) in Corollary 4.6, $B = AX_b((AX_B)^{\mathrm{T}}AX_B)^{-1}(AX_b)^{\mathrm{T}}A = P_{\mathcal{Y}}A$, which gives the first equality in (32). Since $\mathcal{Y} = \mathcal{Y}_1 = \mathrm{Ran}\,(AX_b) = \mathrm{Ran}\,(AX)Y$, the second equality in (32) follows from $P_{\mathrm{Ran}\,(Y)} = Y(Y^{\mathrm{T}}Y)^{+}Y^{\mathrm{T}}$. □.

An advantage of using (31) is that, as long as $\ker(A_{kl}|_{\mathcal{X}}) = \{0\}$ (this can be achieved, for example, by orthogonalizing the columns of $A_{kl}X$, and reducing the dimension of $\mathcal{X}$ if necessary), the matrix $H = (A_{kl}X)^{\mathrm{T}}A_{kl}X$ is symmetric and positive definite, which makes calculating B_{kl} easier.

In a few numerical experiments it was observed that the preconditioner created with blocks B_{kl} obtained in this manner was much better than the one obtained with $\mathcal{X}$ and $\mathcal{Y}$ chosen independently under one of the schemes outlined in Subsection 4.3.2. In the latter case H was likely to be, and B_{kl} to have some large elements.

4.3.2. Strategies for the choice of $\mathcal{X}$. Various techniques can be used in choosing X in (15), i.e. choosing the subspace $\mathcal{X}$ of mesh functions on which B_{kl} is to be an exact approximation to A_{kl}. For a space $\hat{\mathcal{X}}$ of mesh function on Ω, let $\hat{\mathcal{X}}|_{\Omega_l}$ be the space of mesh functions that are restrictions of the elements of $\hat{\mathcal{X}}$ on Ω_l. For a subspace $\hat{\mathcal{X}}$ of the space of mesh functions on the domain, one may want to design a preconditioner such that $(A|_{\hat{\mathcal{X}}}) = (C|_{\hat{\mathcal{X}}})$. A direct LOB preconditioner (3) with this property can be designed as follows:

- For the diagonal block D_{kk}, the diagonal block A_{kk} is used.
- The off-diagonal block B_{kl} is obtained by applying formula (32) to A_{kl}, where $\mathcal{X}$ is $\hat{\mathcal{X}}|_{\Omega_l}$.

Several options for the choice of $\hat{\mathcal{X}}$ are reasonable: for example, the space of all polynomial functions of geometrical mesh coordinates up to a specified maximum degree, or the space of trigonometric functions of coordinates. Calculating X for this

method requires access to the coordinates of the mesh nodes as well as to the matrix A. If a number of eigenvectors of A are known, they can be used as the basis of $\hat{\mathcal{X}}$ as well. For each ODB of the preconditioner constructed using this technique,

$$\operatorname{rank} B_{kl} = \operatorname{rank} X = \dim A_{kl}\mathcal{X} \leq \dim \mathcal{X} \leq \dim \hat{\mathcal{X}}.$$

Instead of choosing a global $\hat{\mathcal{X}}$ and using restrictions of its mesh functions on subdomains, one can directly construct a local space $\mathcal{X}$ for each pair of contacting subdomains using a somewhat similar technique. This approach may be more economical then the previous one, since when X is constructed this way, more functions in $A_{kl}X$ are likely to be linearly independent; thus constructing a preconditioner with ODBs of the same ranks by this method costs less than by the previous one.

In a two-dimensional problem, the border between two subdomains is a one-dimensional curve. Suppose that a coordinate system is introduced with one of the coordinate axis running in the general direction of the border between the subdomains. Then instead of using polynomials of two variables, one can use polynomials of this "along-the-border" coordinate as the basis an $\mathcal{X}$. Moreover, if the variables (and the respective nodes) are numbered in some regular way, one may consider using the node number as the proxy for the node coordinate; this obviates the need to use actual node coordinates.

In a three-dimensional problem, the border between two subdomains is a two-dimensional surface. If one introduces some two-dimensional coordinate system on this surface, the space of polynomial functions (up to a specified degree) of these two coordinates can be used as $\mathcal{X}$.

In the special case when the nodes of a given border surface are in a plane, the two approaches (restricting polynomials of 3 variables onto the 2-D plane vs. using polynomials of the 2 surface coordinates) are equivalent.

4.3.3. SVD method as a special case of the projection method. The SVD method described in Section 4.1 can be thought of as a special case of the method described in this section. If for a given M one substitutes into (15) the matrix X composed of the first M right singular vectors of A_{kl}, and into (16) the matrix Y composed of the first M left singular vectors, the resulting preconditioner block B_{kl} given by (30) will be exactly the block $A_M(A_{kl})$ given by (6).

4.4. Original ODBs. As the rank of the ODBs created by a partial singular-value decomposition or by using the projection method increases, eventually the off-diagonal blocks are the same as those of A. Although the blocks are not "low-rank", it is the limiting case of a family of preconditioners.

5. Preconditioner Quality Analysis.

5.1. A case study: bounds of $\sigma(AC^{-1})$ for a Laplace-operator A and a C from (3). In this section, the bounds of the spectrum $\sigma(AC^{-1})$ are calculated for a simple model problem:

- the matrix A belongs to a certain class (discretized Poisson equation with Dirichlet boundary conditions on a contiguous 2D or 3D domain Ω divided

into p non-overlapping rectangular subdomains Ω_k ($k = 1, 2, \ldots, p$) bordering with one another);

- C is a direct LOB preconditioner given by (3), with D consisting of the diagonal blocks of A, and rank-one off-diagonal blocks.

We show that the spectrum of AC^{-1} is bounded by:

$$\frac{c_1 c_2}{n} < \lambda(AC^{-1}) < 2,$$

where n is the maximum subdomain size, $c_1 > 0$ is a constant independent of the domain geometry, and $c_2 > 0$ is a parameter depending on the way the domain is split into subdomains, but not on n; however, for a class of domains, $c_2 > 1/4$.

Since in this model problem both A and C are Hermitian, the bounds of $\sigma(AC^{-1})$ are also the bounds of the Rayleigh ratio $(x^{\mathrm{T}}Ax)/(x^{\mathrm{T}}Cx)$.

We use the symbol $\partial\Omega$ for the outer border of Ω. A subdomain Ω_k is an *inner subdomain* if it is not adjacent to the outer Dirichlet border $\partial\Omega$; Ω_k is an *outer subdomain* if it is adjacent to $\partial\Omega$. Only one restriction on the partitioning scheme is imposed: if a subdomain Ω_k is an outer subdomain, then at least one of its sides (a line in 2D case, and a plane in 3D case) must be adjacent to $\partial\Omega$ along its entire length (in 2D case) or area (in 3D case).

A regular uniform mesh with some step h is considered on that domain; n is the *discretized subdomain size*, by which we mean that for every subdomain Ω_k its size does not exceed $nh \times nh$ or $nh \times nh \times nh$ in 2- and 3D cases, respectively. The subdomain borders run *between* mesh nodes, so that none of the nodes is exactly on the border between two subdomains. The usual 5-point (in 2D) or 7-point (in 3D) discretization of the Poisson equation is considered. For this kind of mesh, the graph of the matrix is isomorphic to the mesh itself (more exactly, to the graph composed of the inner nodes of Ω and the mesh segments connecting those nodes).

In this section, letters π, κ, ρ will be used to denote nodes of the mesh. Two nodes π and κ are connected in the graph of this matrix A if only if they are connected by the lines of the mesh; thus, for the predicate $\mathrm{neig}(\cdot,\cdot)$, as introduced in Section 1.1,

$$\mathrm{neig}(\pi,\kappa) = ((\pi_x = \kappa_x \wedge |\pi_y - \kappa_y| = h) \vee (\pi_y = \kappa_y \wedge |\pi_x - \kappa_x| = h)) \tag{33}$$

in the 2D case and

$$\begin{aligned} \mathrm{neig}(\pi,\kappa) \;=\; & ((\pi_x = \kappa_x \wedge \pi_y = \kappa_y \wedge |\pi_z - \kappa_z| = h) \vee \\ & (\pi_x = \kappa_x \wedge |\pi_y - \kappa_y| = h \wedge \pi_z = \kappa_z) \vee \\ & (|\pi_x - \kappa_x| = h \wedge \pi_y = \kappa_y \wedge \pi_z = \kappa_z)) \end{aligned} \tag{34}$$

in the 3D case, where for any node ρ, ρ_x, ρ_y, and ρ_z are its Cartesian coordinates. With the specified kind of mesh, the predicate $\mathrm{neig}(\Omega_k, \Omega_l)$ is true if and only if the subdomains Ω_k and Ω_l have a common border. The sets , $_{kl}$ are defined as in Section 1.1.

The number of nodes in , $_{kl}$ (equal to the number of nodes in , $_{lk}$, since the mesh is regular) will be designated by $n_{kl} = n_{lk}$. Since no border between two subdomains is longer than n, for any k and l $n_{kl} \leq n$.

Since a Dirichlet problem is considered, it is convenient for recording some equations to use "phantom nodes" located on the outer border of the domain Ω. The entire set of these outer-border nodes will be designated as $\partial\Omega$. For convenience in denoting summations, Ω and all Ω_k exclude $\partial\Omega$ nodes.

A mesh function u's value at node π will be designated u_π. Since we are analyzing a generalized eigenvalue problem, for each $\rho \in \partial\Omega$, u_ρ will be 0, unless otherwise specified. In this section, for the matrices involved (such as A and C), A_{kl} is the block of the matrix corresponding to the kth block row and the lth block column; and $A(\pi, \kappa)$ is the matrix element (a single number) in row π and column κ.

The partition of the sets of nodes into groups corresponds to the division of Ω into subdomains. The diagonal blocks of A, A_{ii} include elements $A(\pi, \kappa)$ where π and κ belong to the same domain Ω_k; the off-diagonal blocks A_{kl} (which are non-zero only if $\mathrm{neig}(\Omega_k, \Omega_l)$) include elements $A(\pi, \kappa)$ for $\pi \in \, , _{kl} \subset \Omega_k$ and $\kappa \in \, , _{lk} \subset \Omega_j$. Matrix rows and columns correspond to nodes in Ω, and matrix block rows and block columns correspond to subdomains of Ω.

Using the "neighboring-node" notation, the matrix A can be described as

$$A(\pi, \kappa) = \begin{cases} 4, & \text{if } \pi = \kappa; \\ -1, & \text{if } \mathrm{neig}(\pi, \kappa); \\ 0, & \text{otherwise.} \end{cases}$$

The direct LOB preconditioner C is constructed as in (3). Its diagonal blocks are the same as those of A. The rank-one off-diagonal blocks are defined by "lumping": for $\pi \in \Omega_k$ and $\kappa \in \Omega_l$,

$$C(\pi, \kappa) = \begin{cases} -(\eta_{kl})_\pi (\eta_{lk})_\kappa, & \text{if } \pi \in \, , _{kl} \text{ and } \kappa \in \, , _{lk}; \\ 0, & \text{otherwise,} \end{cases}$$

where the components of the vectors η_{kl} are defined as

$$(\eta_{kl})_\pi = \begin{cases} \frac{1}{\sqrt{n_{kl}}}, & \text{if } \pi \in \, , _{kl} \\ 0, & \text{otherwise.} \end{cases}$$

This means that for any two neighboring domains Ω_k and Ω_l, $\eta_{kl}^{\mathrm{T}} \eta_{kl} = 1$.

Since A is symmetric and positively defined, and (as it will be shown) so is C, AC^{-1} exists and is non-singular; is spectrum is the same as that of the Hermitian matrix $C^{-1/2} A C^{-1/2}$, or that of the generalized eigenvalue problem

$$Au = \lambda C u. \tag{35}$$

The lower and the upper bound of the spectrum $\sigma(AC^{-1})$ are the same as those of the Rayleigh ratio, which is defined for the mesh function (vector) u as

$$r(u) = \frac{\alpha(u, v)}{\beta(u, v)},$$

where

$$\alpha(u, v) = u^{\mathrm{T}} A v \quad \text{and} \quad \beta(u, v) = u^{\mathrm{T}} C v.$$

This ratio is defined and positive for every $u \neq 0$.

For brevity, the following notation will also be used:

$$\alpha(u) \equiv \alpha(u,u), \quad \text{and} \quad \beta(u) \equiv \beta(u,u).$$

Due to the structure of the matrices A and C, the quadratic forms $\alpha(u,v)$ and $\beta(u,v)$ can be split:

$$\alpha(u,v) = \alpha_\Omega(u,v) + \alpha_\Gamma(u,v), \qquad \beta(u,v) = \alpha_\Omega(u,v) + \beta_\Gamma(u,v).$$

Here the the term $\alpha_\Omega(u,v)$ is

$$\alpha_\Omega(u,v) = \sum_{k=1}^{p} \alpha_{\Omega_k}(u,v),$$

with

$$\alpha_{\Omega_k}(u,v) = u_k^{\mathrm{T}} A_{kk} v_k = \sum_{\substack{\pi \in \Omega_k \cup \partial\Omega, \kappa \in \Omega_k \cup \partial\Omega \\ \mathrm{neig}(\pi,\kappa) \wedge (\pi < \kappa)}} (u_\pi - u_\kappa)(v_\pi - v_\kappa).$$

As already noted, $u_\pi = 0$ for every $\pi \in \partial\Omega$. The "comparison" $\pi < \kappa$ means only that all nodes are arranged in some way, and used to prevent counting the same terms twice.

The term $\alpha_\Gamma(u,v)$ is

$$\alpha_\Gamma(u,v) = \sum_{\substack{k,l \\ \mathrm{neig}(\Omega_k,\Omega_l)}} \alpha_{\Gamma_{kl}}(u,v)$$

with

$$\alpha_{\Gamma_{kl}}(u,v) = u_k^{\mathrm{T}} A_{kl} v_l = \sum_{\substack{\pi \in \Gamma_{kl}, \kappa \in \Gamma_{lk} \\ \mathrm{neig}(\pi,\kappa)}} (u_\pi - u_\kappa)(v_\pi - v_\kappa).$$

Notice that

$$\alpha(u,v) = \alpha_\Omega(u,v) + \alpha_\Gamma(u,v) = \sum_{\substack{\pi \in \Omega \cup \partial\Omega \\ \kappa \in \Omega \cup \partial\Omega \\ \mathrm{neig}(\pi,\kappa) \wedge (\pi < \kappa)}} (u_\pi - u_\kappa)(v_\pi - v_\kappa),$$

as could be expected.

In the preconditioner C, the term $\beta_\Gamma(u,v)$ is

$$\beta_\Gamma(u,v) = \sum_{\substack{k,l \\ \mathrm{neig}(\Omega_k,\Omega_l)}} \beta_{\Gamma_{kl}}(u,v)$$

with

$$\beta_{\Gamma_{kl}}(u,v) = u_k^{\mathrm{T}} C_{kl} v_l = \frac{1}{n_{kl}} \sum_{\pi \in \Gamma_{kl}} \sum_{\kappa \in \Gamma_{lk}} (u_\pi - u_\kappa)(v_\pi - v_\kappa).$$

This split of $\beta(u)$ into the sum of non-negative terms can be used to prove positive definiteness of C: Suppose that $\beta(u) = 0$ for some non-zero mesh function u. This means that for every subdomain Ω_k, $\alpha_{\Omega_k}(u) = 0$, which is possible only if u is a constant function on every subdomain (i.e. $(\forall \Omega_k)(\exists c_k)(\forall \pi \in \Omega_k)(u(\pi) = c_k)$). But if u is a constant function on every subdomain, then there must be a subdomain on which it is a non-zero; on the other hand, on every subdomain Ω_k adjacent to the outer Dirichlet border $\partial\Omega$, $u_{\Omega_k} = 0$. As a result (due to contiguity of Ω), there must exist two neighboring subdomains Ω_k and Ω_l such that $c_k = 0$ and $c_l \neq 0$; therefore, $\beta_{\Gamma_{kl}} > 0$, which contradicts the assumption $\beta(u) = 0$.

5.1.1. Subspaces. Consider the generalized eigenvalue problem (35) with u in the space of the mesh functions on Ω. The value 1 is an eigenvalue of this problem; the corresponding eigenspace $\mathcal{M}$ is

$$\mathcal{M} = \{u \mid (\forall k, l \mid \mathrm{neig}(\Omega_k, \Omega_l))\ A_{kl} u_l = C_{kl} u_l\}$$

A subspace of $\mathcal{M}$, $\mathcal{M}_1$, can be defined as the space of all mesh functions that are zero at all nodes near the borders between subdomains, i.e.

$$\mathcal{M}_1 = \{u \mid (\forall k, l \mid \mathrm{neig}(\Omega_k, \Omega_l))\ u_{\Gamma_{kl}} = 0\}.$$

Since $\mathcal{M}$ is the eigenspace for the eigenvalue 1, eigenvectors for all other eigenvalues are A-orthogonal (and C-orthogonal) to it. Since $\mathcal{M}$ is not a null space, 1 is an eigenvalue of the generalized eigenvalue problem. This means that if we are interested in other eigenvalues, it is sufficient to study the eigenvalue problem only on vectors from $\mathcal{Q} = \mathcal{M}^{\perp_A}$. Similarly,

$$\min_{u \neq 0} \frac{\alpha(u)}{\beta(u)} = \min_{u \in \mathcal{Q} \setminus \{0\}} \frac{\alpha(u)}{\beta(u)},$$

and

$$\max_{u \neq 0} \frac{\alpha(u)}{\beta(u)} = \max_{u \in \mathcal{Q} \setminus \{0\}} \frac{\alpha(u)}{\beta(u)},$$

provided these min and max are less than and greater than 1, respectively. (And in fact they are).

The following lemma is important:

LEMMA 5.1. *If $u \in \mathcal{Q}$ and $v \in \mathcal{M}$, then $\alpha(u) - \beta(u) = \alpha(u - v) - \beta(u - v)$.*

Proof.. Obviously, $\alpha(u - v) = \alpha(u) - 2\alpha(u, v) + \alpha(v)$. Since u and v are A-orthogonal, $\alpha(u, v) = 0$, and $\alpha(u - v) = \alpha(u) + \alpha(v)$. Likewise, since u and v are also C-orthogonal, $\beta(u, v) = 0$, and $\beta(u - v) = \beta(u) + \beta(v)$. This gives $\alpha(u) - \beta(u) = \alpha(u - v) - \beta(u - v) + \beta(v) - \alpha(v)$. But since $v \in \mathcal{M}$, $Av = Cv$ and $\alpha(v) = \beta(v)$. Therefore,

$$\alpha(u) - \beta(u) = \alpha(u - v) - \beta(u - v).$$

□

5.1.2. A lower bound on the spectrum of AC^{-1}. Lemma 5.1, and the two following lemmas can be used to estimate the lower bound of the spectrum of $C^{-1}A$, which is also a lower bound on the Rayleigh ratio $r(u)$.

LEMMA 5.2. *There is a constant $c_0 > 0$ such that for every mesh function u, for every outer subdomain Ω_k of Ω, and for every inner border Γ_{kl} of this subdomain, $\|u_{\Gamma_{kl}}\|^2 \le c_0 n \alpha_{\Omega_k}(u)$.*

This lemma can be proven by expressing the value of u in each node κ of Γ_{kl} as the sum of some finite differences (taken along some path from some phantom node $\pi \in \partial\Omega$ (where $u(\pi) = 0$) to κ), using the Cauchy inequality, and then summing up.

LEMMA 5.3. *There is a constant $c_1 > 0$ such that for every subdomain Ω_i of Ω, for every inner border Γ_{il} of this subdomain, and for every mesh function u such that the average value of u on Ω_i is zero, $\|u_{\Gamma_{il}}\|^2 \le n \alpha_{\Omega_i}(u)$.*

We outline the proof for the 2D case; it can be easily extended to the 3D case. Represent u_{Ω_i}, the restriction of u on Ω_i, as a linear combination of Fourier components. If the subdomain consists of $(n_x \times n_y)$ nodes, numbered with the pairs of integers (x, y) $(0 \le x < n_x,\ 0 \le y < n_x)$, then u on Ω_i can be represented as

$$\sum_{\substack{0 \le k < n_x \\ 0 \le m < n_y \\ k+m>0}} f_{km}\phi_{km}, \tag{36}$$

with the basis functions

$$\phi_{km}(x,y) = \frac{1}{(k/n_x + m/n_y)} \cos\left(\frac{\pi k(x+1/2)}{n_x}\right) \cos\left(\frac{\pi m(y+1/2)}{n_y}\right).$$

Since the average value of u_{Ω_i} is zero, there is no ϕ_{00} term in (36).

The basis functions ϕ_{km} are α_{Ω_i}-orthogonal. Moreover, one can show that there are some constants $d_1 > 0$, $d_2 > 0$, independent of n_x and n_y, that for all k and m

$$d_1 \le \alpha_{\Omega_i}(\phi_{km}) \le d_2.$$

This leads to

$$\alpha_{\Omega_i}(u) \ge c n_x n_y \sum_{k,m} f_{km}^2 \tag{37}$$

for some $c > 0$ (independent of n_x and n_y). For one of the borders, e.g. $x = 0$, the Cauchy inequality gives

$$\|u_{\Gamma_{x=0}}\|^2 \le c' n_y \sum_m \left(\sum_k \frac{|f_{km}|}{(k/n_x + m/n_y)}\right)^2 \le c' n_y \sum_m \left(\left(\sum_k f_{km}^2\right)\left(\sum_k \frac{1}{(k/n_x + m/n_y)^2}\right)\right) \tag{38}$$

for some $c' > 0$.

The sum $S_m = \sum_k \frac{1}{(k/n_x + m/n_y)^2}$ in (38) involves summation $\sum_{k=1}^{n_x - 1}$ when $m = 0$, and $\sum_{k=0}^{n_x - 1}$ when $m > 0$. In the former case $S_0 \le d_3 n_x^2$, and in the latter case $S_m \le d_4 n_x n_y$, with some $d_3, d_4 > 0$. Thus for every m

$$\sum_k \frac{1}{(k/n_x + m/n_y)^2} \le c'' n_x \max\{n_x, n_y\}$$

for some $c'' > 0$. With this inequality, (38) gives

$$\|u_{\Gamma_{x=0}}\|^2 \leq c'c''n_xn_y\sum_{k,m} f_{km}^2 \max\{n_x, n_y\},$$

which, combined with (37), gives

$$\|u_{\Gamma_{x=0}}\|^2 \leq c''' \max\{n_x, n_y\}\alpha_{\Omega_i}(u)$$

for some $c''' > 0$. Similar calculations can be performed for the other borders of the subdomain with analogous results. The inequality of the lemma holds also in the case when a subdomain borders with more than four other subdomains. In this case, a border , $_{il}$ may include fewer nodes, and $\|u_{\Gamma_{il}}\|$ will be smaller than, or equal to, the one in the foregoing proof.

THEOREM 5.4. *There is a constant $c_1 > 0$ independent of the domain geometry, and a parameter $c_2 > 0$ depending only on the topological properties of the partitioning of the domain into subdomains, such that for every $u \in \mathcal{Q}$, $|\alpha(u) - \beta(u)| \leq c_1c_2n\alpha_\Omega(u)$.*

Proof. Consider an arbitrary mesh function $u \in \mathcal{Q}$. Now construct a function $v \in \mathcal{M}$ with the following properties:

- The average value of $(u - v)$ in every *inner* subdomain Ω_k is 0.
- In every *inner* subdomain Ω_k, $\alpha_{\Omega_k}(u - v) = \alpha_{\Omega_k}(u)$.
- The restriction of v on every *outer* subdomain is 0.

Such a function v can be constructed as the piecewise-constant function whose value in all outer subdomains is 0, and whose value in every node of each inner subdomain Ω_k is the average value of u in the subdomain. By Lemma 5.1,

$$|\alpha(u) - \beta(u)| = |\alpha(u - v) - \beta(u - v)| = |\alpha_\Gamma(u - v) - \beta_\Gamma(u - v)| =$$

$$= \left|\sum_{\substack{k,l \\ \text{neig}(\Omega_k,\Omega_l)}} \alpha_{\Gamma_{kl}}(u - v) - \beta_{\Gamma_{kl}}(u - v)\right| = \left|\sum_{\substack{k,l \\ \text{neig}(\Omega_k,\Omega_l)}} (u - v)^{\mathrm{T}}(A_{\Omega_k,\Omega_l} - C_{\Omega_k,\Omega_l})(u - v)\right|.$$

Due to the structure of the matrix blocks $(A_{\Omega_k,\Omega_l} - C_{\Omega_k,\Omega_l})$, for any two neighboring domains Ω_k and Ω_l,

$$(u - v)^{\mathrm{T}}(A_{\Omega_k,\Omega_l} - C_{\Omega_k,\Omega_l})(u - v) \leq \|(u - v)_{\Gamma_{kl}}\| \, \|(u - v)_{\Gamma_{lk}}\|$$

Using Lemmas 5.2 and/or 5.3 (depending on whether the subdomains are inner or outer) gives

$$(u - v)^{\mathrm{T}}(A_{\Omega_k,\Omega_l} - C_{\Omega_k,\Omega_l})(u - v) \leq c_1n\sqrt{\alpha_{\Omega_k}(u - v)}\sqrt{\alpha_{\Omega_l}(u - v)},$$

with some c_1 independent of n or on the way Ω is partitioned into subdomains. For the entire sum

$$|\alpha(u) - \beta(u)| \leq c_1nS(u - v),$$

where the functional $S(w)$ is defined as

$$S(w) = \sum_{\substack{k,l \\ \mathrm{neig}(\Omega_k,\Omega_l)}} \sqrt{\alpha_{\Omega_k}(w)}\sqrt{\alpha_{\Omega_l}(w)}.$$

To find bound $S(w)$ from above, notice that

$$S(w) = \sigma^{\mathrm{T}} M \sigma,$$

with the p-dimensional vector σ

$$\sigma = \left[\sqrt{\alpha_{\Omega_1}(w)}, \quad \cdots, \quad \sqrt{\alpha_{\Omega_p}(w)} \right]^{\mathrm{T}}$$

and the symmetric $(p \times p)$ connectivity matrix M, which is determined only by the domain partitioning scheme:

$$M_{ij} = \begin{cases} 1, & \text{if } \mathrm{neig}(\Omega_k, \Omega_l); \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$S(w) \leq \rho(M) \left\|\sigma\right\|^2 = \rho(M)\alpha_\Omega(w).$$

The spectral radius $\rho(M)$ depends only on the domain partitioning scheme, and not on n. Since $\rho(M) \leq \|M\|_\infty$, for a given partitioning scheme $\rho(M)$ does not exceed the maximum number of neighbors a subdomain can have. For a large class of regular partitionings — where the domain is divided into subdomains by means of a regular rectangular grid — this number is 4 (in 2D case) or 6 (in 3D case), and therefore $\rho(M) \leq 4$ or $\rho(M) \leq 6$, respectively.

By construction of v, $\alpha_\Omega(u - v) = \alpha_\Omega(u)$, and so

$$|\alpha(u) - \beta(u)| \leq c_1 \rho(M) n \alpha_\Omega(u).$$

This proves Theorem 5.4. □

THEOREM 5.5. *There is a constant $c_1' > 0$ independent of the domain geometry, and a parameter $c_2 > 0$ depending only on the topology of the domain partitioning, such that for every $u \in \mathcal{Q}$, and for large enough n,*

$$\frac{\alpha(u)}{\beta(u)} \geq \frac{c_1' c_2}{n}.$$

Proof.. By Theorem 5.4, $|\beta(u) - \alpha(u)| \leq c_1 \rho(M) n \alpha_\Omega(u)$; therefore,

$$\beta(u) \leq \alpha(u) + c_1 \rho(M) n \alpha_\Omega(u) \leq (1 + c_1 \rho(M) n)\alpha(u).$$

Choosing the constant $c_1' > 0$ smaller than $1/c_1$, and $c_2 = 1/\rho(M)$, for large enough n

$$\alpha(u) \geq \frac{c_1' c_2}{n} \beta(u).$$

□

5.1.3. An upper bound on the spectrum of AC^{-1}. LEMMA 5.6. *For any mesh function u, for any two bordering subdomains Ω_k and Ω_l,*

$$\alpha_{\Gamma_{kl}}(u) \leq 2\beta_{\Gamma_{kl}}(u).$$

Proof. Since the values of $\alpha_{\Gamma_{kl}}(u)$ and $\beta_{\Gamma_{kl}}(u)$ depend only on the values of u at the nodes that are in $, _{kl}$ and $, _{lk}$, it is sufficient to consider these quadratic forms on the $2n_{kl}$-dimensional space $\mathcal{L}$ of mesh functions defined on $, _{kl} \cup , _{ji}$, so that

$$\alpha_{\Gamma_{kl}}(u) = \begin{bmatrix} u_{kl}^{\mathrm{T}} & u_{lk}^{\mathrm{T}} \end{bmatrix} A_{\Gamma_{kl}} \begin{bmatrix} u_{kl} \\ u_{lk} \end{bmatrix}, \qquad \beta_{\Gamma_{kl}}(u) = \begin{bmatrix} u_{kl}^{\mathrm{T}} & u_{lk}^{\mathrm{T}} \end{bmatrix} C_{\Gamma_{kl}} \begin{bmatrix} u_{kl} \\ u_{lk} \end{bmatrix},$$

The nodes being numbered appropriately, the matrices can be written as

$$A_{\Gamma_{kl}} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes I \qquad \text{and} \qquad C_{\Gamma_{kl}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes I + \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \otimes \eta_{kl}\eta_{lk}^{\mathrm{T}}.$$

The matrices $A_{\Gamma_{kl}}$ and $C_{\Gamma_{kl}}$ have the same set of eigenvectors, but different eigenvalues. Namely,

1. On vector $x = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}} \otimes \eta$,
$$A_{\Gamma_{kl}} x = 0, \qquad C_{\Gamma_{kl}} x = 0.$$
2. On vectors $x = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}} \otimes \xi$, with ξ such that $\xi^{\mathrm{T}}\eta = 0$,
$$A_{\Gamma_{kl}} x = 0, \qquad C_{\Gamma_{kl}} x = x.$$
3. On vector $x = \begin{bmatrix} 1 & -1 \end{bmatrix}^{\mathrm{T}} \otimes \eta$,
$$A_{\Gamma_{kl}} x = 2x, \qquad C_{\Gamma_{kl}} x = 2x.$$
4. On vectors $x = \begin{bmatrix} 1 & -1 \end{bmatrix}^{\mathrm{T}} \otimes \xi$, with ξ such that $\xi^{\mathrm{T}}\eta = 0$,
$$A_{\Gamma_{kl}} x = 2x, \qquad C_{\Gamma_{kl}} x = x.$$

So for any $x \in \mathcal{L}$,

$$x^{\mathrm{T}} A_{\Gamma_{kl}} x \leq 2x^{\mathrm{T}} C_{\Gamma_{kl}} x,$$

proving Lemma 5.6. □

LEMMA 5.7. *For any mesh function u,*

$$\alpha_{\Gamma}(u) \leq 2\beta_{\Gamma}(u).$$

Proof. Results from the summing up the Lemma 5.6 inequalities for all borders between subdomains. □

THEOREM 5.8. *For any mesh function u,*

$$\alpha(u) \le 2\beta(u).$$

Proof. By the previous lemma,

$$\alpha(u) = \alpha_\Omega(u) + \alpha_\Gamma(u) \le \alpha_\Omega(u) + 2\beta_\Gamma(u) \le 2(\alpha_\Omega(u) + \beta_\Gamma(u)) = 2\beta(u).$$

□

This result can be somewhat refined using Theorem 5.4, to give

$$\alpha(u) \le (2 - c/n)\beta(u).$$

5.1.4. Summary. Remembering that for all $u \in \mathcal{M}$ $\alpha(u) = \beta(u)$, and using Theorems 5.5 and 5.8, for any mesh function $u \neq 0$

$$\frac{c_1' c_2}{n} \le \frac{\alpha(u)}{\beta(u)} \le 2,$$

and thus obtain the following range for the spectrum of AC^{-1}:

$$\frac{c_1' c_2}{n} \le \lambda(AC^{-1}) \le 2.$$

6. Parallel application of the preconditioner. For a typical domain-decomposed problem, using the BSSOR LOB preconditioner instead of a BSSOR preconditioner with original off-diagonal blocks of A does not reduce the operation count. If any usual method of applying BSSOR preconditioner is used, no parallelism improvement is achieved, as the number and the order of messages sent between processors stay the same. But, if $\operatorname{rank} B_{kl} \ll \operatorname{rank} A_{kl}$, the *message size*, i.e. the amount of data in a message, becomes less. With $B_{kl} = \sum_{s=1}^{M_{kl}} u_{kl,s} v_{kl,s}^{\mathrm{T}}$, one distributes data among the processors so that during L-solve, instead of sending $A_{kl}x_l$ (which takes, in the best possible format, $\operatorname{rank} A_{kl}$ floating-point values), one will need to send only M_{kl} numbers: dot product results $\left\{v_{kl,s}^{\mathrm{T}} x_l\right\}_{s=1,\dots,M_{kl}}$.

However, using preconditioners with low-rank ODBs offers the following potential advantage: when D is a block-diagonal matrix and Q_L and Q_U are strictly block triangular matrices composed of low-rank blocks, it is possible to design well parallelizable methods for solving systems $Bx = y$, where B is either $D + Q_L + Q_U$ (the entire direct LOB preconditioner (3)), or one of the two factors of the BSSOR LOB preconditioner (4): $B_2 = D + Q_U$ and $B_1 = (D + Q_L)D^{-1}$.

Applying direct LOB preconditioner (3) is the same as solving the system

$$Bx = y, \tag{39}$$

where the non-singular $n \times n$ matrix B has the splitting

$$B = D + Q \tag{40}$$

with a non-singular block-diagonal matrix $D = \mathrm{diag}\{D_{11}, D_{22}, \ldots, D_{pp}\}$ and the block matrix $Q = Q_L + Q_U$.

Applying the BSSOR LOB preconditioner (4) involves solving two systems of the same, or similar, form. This preconditioner can be dealt with as a product of two matrices: $C = B_1 B_2$, one of which ($B_2 = D + Q_U$) has the same splitting (40) with $Q = Q_U$, and the other can be represented as

$$B_1 = (D + Q)D^{-1}, \tag{41}$$

with the same D as in (40) and $Q = Q_L$.

This section introduces an easily and efficiently parallelizable direct method for solving block linear system (39) with the matrix of the form (40) or (41), taking advantage of the low ranks of the blocks of Q. The method's operation count is close to the cost of calculating a matrix-vector product Qw for some w, plus at most twice the cost of calculating $D^{-1}w$ for some w. When implemented on a parallel machine the processor utilization can be as good as that of those operations. From the design of the LOB preconditioners (4) and (3) one can notice that the two forms of Q that we are actually dealing with are:

- Strictly block lower or upper triangular Q. This is the case when our B is one of the factors $D + \omega U$ and $D + \omega L$ that appear in the preconditioner (4).
- A block matrix Q with zero diagonal blocks, such as $L + U$ in (3).

However, the method discussed in this section does not use the particular structure of Q, except that one of the steps of the algorithms will be shown to be faster when Q is block triangular

We assume that the n variables are divided into p blocks (so that B and other matrices of the same dimension consist of $p \times p$ blocks). A typical source of the blocking would be from domain decomposition, with $p \ll n$.

Define r_{kl}, m_l, m, and M as in Section 3. Let $\mathcal{P}$ be the set of all the pairs (k, l) such that $Q_{kl} \neq 0$. Each block of Q can be represented as

$$Q_{kl} = \sum_{i=1}^{r_{kl}} u_{kl}^i {v_{kl}^i}^{\mathrm{T}} = U_{kl} V_{kl}^T, \tag{42}$$

with $U_{kl} = [\, u_{kl}^1 \quad u_{kl}^2 \quad \cdots \quad u_{kl}^{r_{kl}} \,]$ and $V_{kl} = [\, v_{kl}^1 \quad v_{kl}^2 \quad \cdots \quad v_{kl}^{r_{kl}} \,]$.

Since (39) is used in iterative solving of (1), it needs to be solved for many right-hand sides y, which are not available at the same time. Blocks of D and Q are constructed and represented to allow: (1) solving systems with the diagonal blocks D_{kk}, and (2) performing matrix-vectors multiplications with the blocks Q_{kl}. E.g. each block of D is represented as a product of a lower and an upper triangular matrices obtained by incomplete LU factorization of the diagonal blocks of A.

The goal is a direct preconditioner whose operation count is comparable to that of any method that uses the two above-mentioned operations (such as, e.g., the usual block elimination method, in the case when B is block-triangular), but which can be efficiently parallelized.

6.1. Data representation and distribution. The data are distributed among the processors with

- D_{kk} stored on processor k;
- U_{kl} (composed of the vectors $\{u_{kl}^s\}_{s=1,\ldots,r_{kl}}$) stored on processor k;
- V_{kl} (composed of the vectors $\{v_{kl}^s\}_{s=1,\ldots,r_{kl}}$) is stored on processor l.

Here processor k may be a virtual processor, and if fewer than p processors are available, several virtual processors are mapped to one physical processor.

6.2. Representation of B^{-1}. From (42), $Q = UV^{\mathrm{T}}$, with M-column matrices U and V:

$$U = \begin{bmatrix} U_{11} & 0 & \cdots & 0 & U_{12} & 0 & \cdots & 0 & \cdots & U_{1p} & 0 & \cdots & 0 \\ 0 & U_{21} & \cdots & 0 & 0 & U_{22} & \cdots & 0 & \cdots & 0 & U_{2p} & \cdots & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & \cdots & \vdots & & \ddots & \vdots \\ 0 & & 0 & U_{p1} & 0 & & 0 & U_{p2} & \cdots & 0 & & 0 & U_{pp} \end{bmatrix} \tag{43}$$

$$V = \begin{bmatrix} V_{11} & V_{21} & \cdots & V_{p1} & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & V_{12} & V_{22} & \cdots & V_{p2} & \cdots & \vdots & 0 & & \vdots \\ \vdots & & & & \vdots & & & & \cdots & 0 & & & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & V_{1p} & V_{2p} & \cdots & V_{pp} \end{bmatrix} \tag{44}$$

The block column ordering shown above is chosen so that G will have the same block structure as Q, see Section 6.3. Block columns in (43) and (44) corresponding to zero blocks Q_{kl} are absent (have no columns).

Since $B = D + UV^{\mathrm{T}}$, the Sherman–Morrison–Woodbury formula [11] gives

$$B^{-1} = (D + UV^{\mathrm{T}})^{-1} = D^{-1} - D^{-1}U(I + G)^{-1}V^{\mathrm{T}}D^{-1}, \tag{45}$$

with $G = V^{\mathrm{T}}D^{-1}U$ of order M. For B_1 from (41) we have

$$B_1^{-1} = D(D + UV^{\mathrm{T}})^{-1} = I - U(I + G)^{-1}V^{\mathrm{T}}D^{-1}, \tag{46}$$

Since the non-zero eigenvalues of G are the same as those of $D^{-1}UV^{\mathrm{T}}$,

$$\sigma(I + G) \setminus \{1\} = \sigma(D^{-1}B) \setminus \{1\},$$

where $\sigma(H)$ is the set of eigenvalues of the square matrix H. Since B is non-singular, so is $I + G$. Furthermore, if $M < n$ (i.e. if the ranks of the blocks of Q are low enough), then $\operatorname{rank}(UV^{\mathrm{T}}) \leq M < n$, and $1 \in \sigma(D^{-1}B)$. Therefore $\sigma(I + G) \subset \sigma(D^{-1}B)$, and $\operatorname{cond}(I + G) \leq \operatorname{cond}(D^{-1}B)$.

6.3. Structure of G. The following proposition shows that the block sparsity pattern of G is contained in that of Q:

PROPOSITION 6.1. *If $Q_{kl} = 0$, then $G_{kl} = 0$.*

Proof. Let G be partitioned as

$$G = \begin{bmatrix} G_{11} & \cdots & G_{1p} \\ G_{21} & \cdots & G_{2p} \\ \vdots & & \vdots \\ G_{p1} & \cdots & G_{pp} \end{bmatrix},$$

where an $m_k \times m_l$ block G_{kl} has the structure

$$\begin{aligned} G_{kl} &= \begin{bmatrix} 0 & \cdots & 0 & V_{1k}^{\mathrm{T}} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & V_{2k}^{\mathrm{T}} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & V_{pk}^{\mathrm{T}} & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} D_{11}^{-1}U_{11} & 0 & \cdots & 0 \\ 0 & D_{22}^{-1}U_{21} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & D_{pp}^{-1}U_{p1} \end{bmatrix} \\ &= \begin{bmatrix} 0 & \cdots & 0 & H_{1kl} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & H_{2kl} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & H_{pkl} & 0 & \cdots & 0 \end{bmatrix}, \end{aligned} \tag{47}$$

with zeros in the first $k-1$ block columns, and the last $p-k$ block columns, and $H_{jkl} = V_{jk}^{\mathrm{T}} D_{kk}^{-1} U_{kl}$ of size $r_{jk} \times r_{kl}$. The equation for H_{jkl} shows that the block G_{kl} can be non-zero only if U_{kl} is non-zero, which, in its turn, can be non-zero only if Q_{kl} is non-zero. □

Since we use *no-cancellation assumption* [6, Section 2.2.2], i.e. disregard exact cancellation in floating-point computation, it follows from Proposition 6.1 that G has the same non-zero block structure as Q.

COROLLARY 6.2. *If Q is strictly block upper or lower triangular, then so is G.*

Since H_{jkl} is $r_{jk} \times r_{kl}$, the total number of non-zero elements in G is $\mathrm{nnz}\,(G) = \sum_k \left((\sum_j r_{jk})(\sum_l r_{kl}) \right) = \sum_{kl} (r_{kl} m_k) \le mM$.

If no block of Q has rank higher than one, then $mM \le p^3$. If Q is block-tridiagonal, with all non-zero blocks of rank one, then $mM < 3p$.

6.4. Outline of the solution method. The parallel solution method for solving $Bx = y$ is based on (45), and will be referred to later as the SMW-based method.

Computing the preconditioner.
begin
1. **foreach** $k = 1 : p$ *(in parallel)*
Preprocess D_{kk} for solving $D_{kk} x_k = y_k$ (if necessary)
endfor
2. **foreach** $k = 1 : p$ *(in parallel)* */* Compute entries of G */*
for $l = 1 : p$ *such that* $(k, l) \in \mathcal{P}$
2a. *Solve $D_{kk} \tilde{U}_{kl} = U_{kl}$*
for $j = 1 : p$ *such that* $(j, k) \in \mathcal{P}$
2b. *Set $H_{jkl} = V_{jk}^{\mathrm{T}} \tilde{U}_{kl}$*
endfor

endfor

endfor

2c. If desired, gather all block rows of G on a single processor

3. LU-factor $I + G$

end

Computing G consists of two steps. On step 2a, processor k computes $\tilde{U}_{kl}$ for all l's by solving the linear system

$$D_{kk} \, [\, \tilde{U}_{k1} \quad \tilde{U}_{k2} \quad \cdots \quad \tilde{U}_{kp} \,] = [\, U_{k1} \quad U_{k2} \quad \cdots \quad U_{kp} \,]$$

with $\sum_l r_{kl}$ right-hand sides. On step 2b, processor k computes the blocks G_{kl} for all values l, i.e. the blocks H_{jkl} in the matrix

$$\begin{bmatrix} H_{1k1} & \cdots & H_{1kp} \\ H_{2k1} & \cdots & H_{2kp} \\ \vdots & & \vdots \\ H_{pk1} & \cdots & H_{pkp} \end{bmatrix} = \begin{bmatrix} V_{1k}^{\mathrm{T}} \\ V_{2k}^{\mathrm{T}} \\ \vdots \\ V_{pk}^{\mathrm{T}} \end{bmatrix} \cdot [\, \tilde{U}_{k1} \quad \tilde{U}_{k2} \quad \cdots \quad \tilde{U}_{kp} \,]$$

As a result of this step, G is distributed among the processors by block row. A gather step is needed if one wants G to be stored on a single processor.

Unless G is triangular, some LU-factorization of $I + G$ is needed (step 3). If G is stored on a single processor, factoring is done by that processor; otherwise, it is done in parallel.

Applying the preconditioner. Applying the direct LOB preconditioner (3) means solving a system of the form (39), using formula (45):

begin Solving $Bx = y$

1. **foreach** $k = 1 : p$ *(in parallel)* /* $z = D^{-1}y$ */

Solve $D_{kk} z_k = y_k$

endfor

2. **foreach** $l = 1 : p$ *(in parallel)* /* $t = V^{\mathrm{T}} z$ */

for $k = 1 : p$

Set $t_{kl} = V_{kl}^{\mathrm{T}} z_l$

endfor

endfor

3. Solve $(I + G)s = t$ /* $s = (I + G)^{-1} t$ */

4. **foreach** $k = 1 : p$ *(in parallel)* /* $y' = y - Us$ */

Set $y'_k = y_k - \sum_l U_{kl} s_{kl}$

endfor

5. **foreach** $k = 1 : p$ *(in parallel)* /* $x = D^{-1}y'$ */

Solve $D_{kk} x_k = y'_k$

endfor

end

Applying the BSSOR LOB preconditioner (4) involves solving one system of the form (39) with B of the form (40) using the algorithm given above, and one system

with B of the form (41), which, due to (46), can be done using steps 1–4 of the same algorithm.

Step 3 solves

$$(I+G)s = t, \tag{48}$$

where

$$t = V^{\mathrm{T}} z = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_p \end{bmatrix} \quad \text{with} \quad t_l = \begin{bmatrix} t_{1l} \\ t_{2l} \\ \vdots \\ t_{pl} \end{bmatrix},$$

and s is partitioned conformally with t.

Only step 3 involves interprocessor communications. On step 2, the j-th processor produces all components of t in its block column (t_{kj} for all k); on step 4, the j-th processor needs all components of s in its block row (s_{jl} for all l). Depending on how (48) is solved, step 3 can be implemented in different ways:

1. Store the entire factors of $I+G$ on a single processor, which solves the system (48). First, t is gathered on this processor; after solving, components of s are scattered to the processors that will need them at the next step. Both steps 2 and 4 send M floating-point values between the processors.
   What processor should be used for solving (48)? Normally, it can be one of the processors $1,\dots,p$ that perform all other operations. However, if minimizing the memory allocation per processor is a priority, and there are extra processors available, it may be expedient to use $p+1$ processors, the extra $(p+1)$-st processor being used to store the factors of $I+G$ and solve (48).
2. Have each processor, redundantly, solve (48).
3. Use a parallel method for solving (48), distributing the factors of $I+G$ among all processors. At most pM floating-point numbers are sent between processors. Factors of $I+G$ can be distributed among the p processors by block row. If this scheme results in poor data balance (as in the case of a block-triangular or block-tridiagonal Q), linear speed-up can be achieved by distributing the factors of $I+G$ cyclically by row.

The preconditioner can be applied with one D-solve per iteration instead of two, by replacing steps 4 and 5 with

4′. **foreach** $k = 1:p$ *(in parallel)* /* $x = z - \tilde{U}\ s$ */
   Set $x_k = z_k - \sum_l \tilde{U}_{kl} s_{kl}$
   endfor

where $\tilde{U} = D^{-1}U$ (see step 2a of the preconditioner-preparing algorithm). The disadvantage is that the DAXPY operation in 4′ is done with dense vectors, while that in 5 uses sparse ones. In the remainder of the article, we use 4 and 5 instead of 4′.

6.5. Communication costs. While discussing the communication pattern, one needs to consider three options:

- Sequential handling of $I + G$: after $I + G$ is computed, it is is replicated on each processor[2]; it is factored sequentially, and the LU-solve of (48) is performed sequentially on each iteration.
- Parallel handling of $I + G$, block row distribution: after G is generated, it is distributed by block row, so that elements of G are initially stored at the processors where they are computed. $I + G$ is factored in parallel, and its factors are stored distributed in the same fashion. LU-solve of (48) is done in parallel. This approach is suitable for a block-dense Q, or in any situation where the block-triangular factors obtained by LU-factorization of $I + G$ are block-dense; however, it may not provide for good load balancing in special cases such as a block-triangular or block-tridiagonal Q.
- Parallel handling of $I + G$, cyclic row distribution: first G, and then the LU-factors of $I + G$ are distributed by row in a cyclic fashion. This approach provides for better load balancing even in the case of a block-triangular or block-tridiagonal Q.

Creating G. If parallel factoring of $I + G$ is used with $I + G$ distributed by block row (so that blocks of G are stored on the processors where they are first computed), creating G involves no communications; otherwise (sequential solve, or parallel solve with cyclic row distribution), each processor broadcasts all $\leq m^2$ elements of G it computes.

Factoring $I + G$. Factoring of $I + G$ involves no communications if it is done in sequential mode. In either of the parallel modes, each processor broadcasts all elements of the U-factor of $I + G$ it generates, which is $O(mM)$ floating-point numbers.

Applying the preconditioner. When the preconditioner is applied sequentially, each processor broadcasts the $\leq m$ elements of t it computes. Since solving (48) is replicated on each processor, no components of s or any other data need to be sent between processors.

When (48) is solved in parallel, each processor broadcast all elements it computes, both during L-solve and during U-solve. This is $O(m)$ values, regardless of whether the factors of $I + G$ are distributed by block row, each processor sends

6.6. Parallelism analysis. Among operations needed for solving the system (39) by the above method, all operations are highly parallel (in the sense that, for each of them, calculations pertaining to a particular block of x or y can be performed by the processor that stores that block without any dependence on other processors), with the exception of solving the system (48) with the $M \times M$ matrix $(I + G)$. The right-hand side t of this system is gathered from data from all processors (so a barrier and a gather operation is required); solving the system is done on one processor, and followed by scattering the appropriate parts of the solution s to the other processors.

[2] If $I + G$ is gathered on a single processor instead, the communication will be lower than estimated in this section, since instead of broadcasting data one needs to send them to that one processor only.

6.7. Operation count analysis. Solving (39) with B of the form (40) requires the following number of operations:

$$C_B = C_D + C_V + C_G + C_U + C_D = 2C_D + C_m + C_G.$$

The five terms of the first sum correspond to the five steps of the algorithm. C_D is for solving $Dz = w$, C_V is for one local dot product for each column of the blocks V_{kl}, C_U is for one DAXPY with each column of the blocks U_{kl}'s, and C_G is for solving (48). The sum $C_m = C_V + C_U$ is the number of operations needed to perform a matrix-vector multiplication $Qw = UV^{\mathrm{T}}w$.

For solving (39) with B of the form (41) we have, similarly,

$$C_{B_1} = C_D + C_V + C_G + C_U = C_D + C_m + C_G.$$

6.8. Timing models. During solving $Bx = y$, only solving system (48) requires interprocessor communications; all other operations are parallel. From the operation count data in Section 6.7, the time t_B the method needs to solve (39) is approximately the following:

$$t_B = 2t_D + t_V + t_U + t_G = 2t_D + t_m + t_G. \tag{49}$$

The three main terms here, t_D, t_V, and t_U, are times this parallel computer will take, respectively

t_D: to solve $Dz = w$,

t_V: to calculate a dot product with each column of each block V_{kl},

t_U: to perform a DAXPY operation with each column of each block U_{kl},

with the matrices distributed as described in Section 6.1. The sum $t_m = t_V + t_U$ is the time needed to calculate a matrix-vector product $Qw = UV^{\mathrm{T}}w$.

Similarly, $t_{B_1} = t_D + t_m + t_G$.

Cost of solving (48). The term t_G is the time needed to solve (48). When Q is strictly block triangular, so is G, and $t_G = O(Mm)$.

When the matrix G is not triangular, it still does not depend on the right-hand side y, and so $I + G$ needs to be LU-factored once for all future B-solves. The sequential solve time t_G^{seq} is $O(M^2)$; parallel solve time $t_G^{\mathrm{par}} = O(mM)$. This t_G can be reduced if the block structure of Q is such such that fill-in is limited in some way. E.g., for block-tridiagonal Q, t_G^{seq} is $O(mM)$ and $t_G^{\mathrm{par}} = O(M \max\{1, m/p\})$.

Costs of preparing G. Computing G involves solving M linear systems of the form $D_{jj}\tilde{u}_{ji} = u_{ji}$, and then calculating no more than Mm dot products of the form $v_{kj}^{\mathrm{T}}\tilde{u}_{ji}$. The total number of operations involved is under $m(C_D + C_U) \approx \frac{m}{2}C_B$. Since both D-solves and dot products can be done in parallel,

$$t_{\mathrm{prepG}} \approx \frac{m}{2}t_B. \tag{50}$$

The cost of LU-factoring the matrix $I + G$ depends greatly on the structure of this matrix (and thus on the structure of Q). For a general Q, the operation count for this

factoring is, is the worst case, $C_{\text{fact(I+G)}} = \mathcal{O}(M^3)$, which implies $t^{\text{seq}}_{\text{fact(I+G)}} = \mathcal{O}(M^3)$ for sequential factoring, and

$$t^{\text{par}}_{\text{fact(I+G)}} = \mathcal{O}(mM^2) \tag{51}$$

for factoring on p processors in parallel (the p-th processor calculates the p-th block row of the factors). This result may be better if Q has some special structure making LU-factoring G less expensive than $\mathcal{O}(M^3)$ operations. For example, if Q is block-tridiagonal, $t^{\text{seq}}_{\text{fact(I+G)}} = \mathcal{O}(m^2M)$ and

$$t^{\text{par}}_{\text{fact(I+G)}} = \mathcal{O}(mM \max 1, m/p). \tag{52}$$

The timing model provides a practical definition of "low-rank" for blocks of Q:

(a) The time spent preparing and factoring $I + G$ must be small compared to the iterative solve time for (1), and

(b) t_G must be small relative to $t_D + t_U$, so that

$$t_B \approx 2t_D + t_m. \tag{53}$$

Below we will show what restrictions this conditions impose on m, for a general matrix Q, as well as for the special cases of block-triangular and block-tridiagonal Q. We suppose that:

1. p processors are used.
2. The maximum dimension of a block D_{kk} is $\mathcal{O}(n/p)$.
3. $m = \mathcal{O}(M/p)$, i.e. the maximum sum of block ranks in a block row of Q is of the same order as the average sum.
4. Factoring $I+G$ and solving (48) is done in parallel on p processors. (Estimates for sequential operations can be obtained in a similar way.)
5. t_D is the dominant term in $t_D + t_m$, i.e. $t_D + t_m = \mathcal{O}(t_D)$. This is usually the case with many matrices; and if off-diagonal block computations are expensive relative to the diagonal block computations, i.e. if $t_m > pt_D$, then parallelizing the solution of the system (39) is easy.

The time t_D spent solving the system $Dz = w$ depends greatly on the structure of D. The two obvious limiting cases are

- $t_D = \Theta(n/p)$ (D_{kk} is a diagonal matrix, or a product of LU-factors with few non-zeros per row);
- $t_D = \Theta(n^2/p^2)$ (D_{kk} is a product of dense L- and U-factors).

Estimates below will be made for a Q of arbitrary block structure, as well as for block-triangular and block-tridiagonal Q.

Equation (50) indicates that t_{prepG} is on the order of the time taken by m iterations. Thus the criterion (a) implies that m needs to be small in comparison to the total number of iteration.

In order for the term $t_{\text{fact(I+G)}}$ not to make this ratio much worse, it needs to be of the same order as t_{prepG}, i.e. the condition

$$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D) \tag{54}$$

should hold. For a block-triangular Q, this is not an issue (no factoring is needed); for a general Q, the parallel LU-factoring time estimate (51) is used to rewrite the condition (54) as

$$m = \begin{cases} \mathcal{O}\left(\frac{n^{1/2}}{p^{3/2}}\right), & \text{if } t_D = \Theta\left(n/p\right), \\ \mathcal{O}\left(\frac{n}{p^2}\right), & \text{if } t_D = \Theta\left(n^2/p^2\right) \end{cases} \tag{55}$$

For a block-tridiagonal Q, the parallel LU-factoring time estimate (52) gives

$$m = \begin{cases} \mathcal{O}\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right), & \text{if } t_D = \Theta\left(n/p\right), \\ \mathcal{O}\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right), & \text{if } t_D = \Theta\left(n^2/p^2\right). \end{cases} \tag{56}$$

For an arbitrary Q, the time t_G needed to solve $I + G$ by a parallel dense method is $\mathcal{O}(mM)$. Thus the criterion (b) requires that

$$m = \begin{cases} o\left(\frac{n^{1/2}}{p}\right), & \text{if } t_D = \Theta\left(n/p\right), \\ o\left(\frac{n}{p^{3/2}}\right), & \text{if } t_D = \Theta\left(n^2/p^2\right). \end{cases} \tag{57}$$

For a block-triangular or a block-tridiagonal Q we similarly obtain

$$m = \begin{cases} o\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right), & \text{if } t_D = \Theta\left(n/p\right), \\ o\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right), & \text{if } t_D = \Theta\left(n^2/p^2\right). \end{cases} \tag{58}$$

Obviously, condition (55) is stronger than condition (57), but for a block-tridiagonal Q (58) is stronger than (56).

If Q is ...	To ensure that...	If $t_D = \Theta(n/p)$	If $t_D = \Theta(n^2/p^2)$
arbitrary	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\frac{n^{1/2}}{p^{3/2}}\right)$	$m = \mathcal{O}\left(\frac{n}{p^2}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p}\right)$	$m = o\left(\frac{n}{p^{3/2}}\right)$
blk-tridiag.	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right)$	$m = \mathcal{O}\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right)$	$m = o\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right)$
blk-triang.	$t_G = o(t_D + t_m)$	$m = o\left(\max\{\frac{n}{p^2}, \frac{n^{1/2}}{p^{1/2}}\}\right)$	$m = o\left(\max\{\frac{n^2}{p^3}, \frac{n}{p}\}\right)$

TABLE 1

Low-rank criteria (in the case of parallel solving of $(I + G)s = t$).

These results for parallel factoring of $I+G$ and parallel $(I+G)$-solve are summarized in Table 1. Table 2 presents analogously obtained results for sequential LU-factoring and $(I + G)$-solve.

If the symbol o in the restrictions on m (57,58) is replaced with $\mathcal{O}$, then instead of (53) we will have

$$t_B = \mathcal{O}(t_D + t_m). \tag{59}$$

If Q is ...	To ensure that...	If $t_D = \Theta(n/p)$	If $t_D = \Theta(n^2/p^2)$
arbitrary	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\frac{n^{1/2}}{p^2}\right)$	$m = \mathcal{O}\left(\frac{n}{p^{5/2}}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p^{3/2}}\right)$	$m = o\left(\frac{n}{p^2}\right)$
block-tridiagonal	$t_{\text{fact(I+G)}} = \mathcal{O}(mt_D)$	$m = \mathcal{O}\left(\frac{n^{1/2}}{p}\right)$	$m = \mathcal{O}\left(\frac{n}{p^{3/2}}\right)$
	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p}\right)$	$m = o\left(\frac{n}{p^{3/2}}\right)$
block-triangular	$t_G = o(t_D + t_m)$	$m = o\left(\frac{n^{1/2}}{p}\right)$	$m = o\left(\frac{n}{p^{3/2}}\right)$

TABLE 2
Low-rank criteria (in the case of sequential solving of $(I+G)s = t$).

7. Relation between the Sherman–Morrison–Woodbury based method and Schur complement-based methods. The SMW-formula based algorithm for solving (39) described in Section 6 is similar to Schur complement methods [3, Chapter 9], where the solution of the system

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

is calculated by a process that involves matrix-vector multiplications with A_{12} and A_{21} and solving linear systems with A_{11} and $S = A_{22} - A_{12}A_{11}^{-1}A_{21}$. This similarity is not accidental: the Sherman–Morrison–Woodbury formula in (45) can be derived with the help of Schur complement. Consider the system (39,40)

$$(D + Q)x = y$$

with $Q = UV^{\mathrm{T}}$. With an auxiliary vector $s = V^{\mathrm{T}}x$, this system can be re-written as

$$\begin{bmatrix} D & U \\ -V^{\mathrm{T}} & I \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{60}$$

The popular formula

$$\begin{bmatrix} D & U \\ -V^{\mathrm{T}} & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -D^{-1}U \\ 0 & I \end{bmatrix} \begin{bmatrix} D^{-1} & 0 \\ 0 & (I+G)^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ V^{\mathrm{T}}D^{-1} & I \end{bmatrix} =$$

$$\begin{bmatrix} I \\ 0 \end{bmatrix} D^{-1} [\, I \quad 0 \,] - \begin{bmatrix} D^{-1}U \\ I \end{bmatrix} (I+G)^{-1} [\, V^{\mathrm{T}}D^{-1} \quad I \,],$$

with the Schur complement $I + G = I + V^{\mathrm{T}}D^{-1}U$, immediately gives

$$s = (I+G)^{-1}V^{\mathrm{T}}D^{-1}y$$

and

$$x = D^{-1}(y - Us)$$

as the solutions of (60), exactly the result obtained by using (45).

8. Preliminary computational results. The direct LOB preconditioner, applied using the SMW-based method in conjugate gradients stabilized [4] iterations was implemented in pC++ [5] on a 12-processor SGI Power Challenge.

The report [10] describes the results for the implementation where $I+G$ was factored sequentially, and $(I+G)s = t$ was solved sequentially. In that report, the direct LOB preconditioner and the SMW-based application method are referred to as the "LRA SMW preconditioner". Results presented here are obtained using parallel processing of $I+G$ and its factors.

The test problem whose solution results are presented here include two problems with $p = 8$, coming from a 2D finite element problem in CFD (problems `Str389`, with $n = 9275$ nodes, and `20284.mlp`, with $n = 20284$ nodes[3]), as well as the equation

$$u_{xx} + u_{yy} + u_{zz} - 1000x^2 u_x + 1000u = 1$$

discretized on a mesh with $n = 24^3$ nodes in a cubic subdomain divided into $p = 27$ cubic subdomains (problem `eq8`). They were solved using a number of preconditioners:

1. Jacobi ($C = D$).
2. BSSOR LOB ($C = (Q_L + D)D^{-1}(Q_U + D)$; blocks of the strict lower and upper triangular Q_L and Q_U are obtained by one of the low-rank approximation methods (the projection method, Section 4.3) from the respective blocks of A. For the first two problems, the subspace $\mathcal{X}$ for each non-zero off-diagonal block was composed of 3 polynomial function of the node number; for the third problem, each $\mathcal{X}$ was a space of polynomials of the subdomain surface coordinates of the 7-th degree. The preconditioner is applied using a conventional LU-solve method.
3. Direct LOB preconditioner with off-diagonal blocks obtained as above, applied using the SMW-based method (Section 6).
4. BSSOR LOB preconditioner with original ODBs ($C = (L + D)D^{-1}(U + D)$; strict lower and upper triangular L and U are composed from the ODBs of A); applied using a conventional LU-solve method.
5. Direct LOB preconditioner with original ODBs ($C = D + (L + U)$ with the same L and U as above), applied using the SMW-based method.

In all preconditioners, blocks of D were obtained by an incomplete LU-factorization [9] of the diagonal blocks of A. Three levels of fill were allowed for the first two problems, and 14 for the third problem (the hardest).

In Tables 3 and 4, the first number in each cell is the total time spent to solve the problem on the specified number of processors with the specified preconditioner; it is followed by the preconditioner preparation time (LU-factoring D, generating Q, generating and factoring $I + G$) plus the number of iterations times the time per iteration. All times are in seconds.

Data in Tables 3 and 4 show that in the low rank case ($M = 66$ or $M = 84$), the SMW-applied direct LOB preconditioner has a parallel efficiency of 0.85 or better,

[3] Timing results for solving these two problems by the same solution methods, but with $I + G$ handled sequentially, are presented in [10]

Num. proc.	Jacobi	BSSOR LOB (M=84)	direct LOB (SMW) (M=84)	BSSOR orig. ODB	direct LOB, orig. ODB (SMW) (M=1234)
1	$183.2 = 7.9+$ $+212 \cdot 0.83$	$95.3 = 8.6+$ $+67 \cdot 1.29$	$80.5 = 10.8+$ $+51 \cdot 1.37$	$40.9 = 7.9+$ $+23 \cdot 1.44$	$84.1 = 73.5+$ $+6 \cdot 1.77$
2	$36.4 = 4.1+$ $+82 \cdot 0.39$	$67.5 = 4.7+$ $+60 \cdot 1.05$	$37.8 = 5.8+$ $+52 \cdot 0.62$	$28.7 = 4.1+$ $+23 \cdot 1.07$	$43.2 = 37.7+$ $+6 \cdot 0.91$
4	$19.7 = 2.8+$ $+84 \cdot 0.20$	$64.2 = 3.6+$ $+66 \cdot 0.92$	$24.6 = 3.7+$ $+52 \cdot 0.40$	$24.1 = 3.0+$ $+22 \cdot 0.96$	$28.4 = 25.1+$ $+6 \cdot 0.55$
8	$12.6 = 3.6+$ $+84 \cdot 0.11$	$60.8 = 2.7+$ $+66 \cdot 0.88$	$11.4 = 2.4+$ $+52 \cdot 0.17$	$21.9 = 2.0+$ $+22 \cdot 0.90$	$24.9 = 23.2+$ $+6 \cdot 0.29$

TABLE 3

Timing results for Problem `Str389`. $n = 9275$. *Parallel* $(I + G)$*-solve.*

Num. proc.	Jacobi	BSSOR LOB (M=66)	direct LOB (SMW) (M=66)	BSSOR orig. ODB	direct LOB, orig. ODB (SMW) (M=2214)
1	$158.1 = 16.0+$ $+76 \cdot 1.87$	$230.2 = 18.9+$ $+72 \cdot 2.93$	$231.1 = 22.5+$ $+68 \cdot 3.07$	$141.4 = 16.1+$ $+39 \cdot 3.21$	$459.5 = 384.2+$ $+19 \cdot 3.96$
2	$81.0 = 8.4+$ $+78 \cdot 0.93$	$181.9 = 10.3+$ $+70 \cdot 2.45$	$112.1 = 12.3+$ $+68 \cdot 1.47$	$105.4 = 8.2+$ $+39 \cdot 2.49$	$300.2 = 259.7+$ $+19 \cdot 2.13$
4	$38.2 = 4.1+$ $+77 \cdot 0.44$	$167.7 = 6.2+$ $+77 \cdot 2.10$	$55.4 = 7.6+$ $+68 \cdot 0.70$	$95.4 = 4.2+$ $+43 \cdot 2.12$	$160.1 = 133.8+$ $+19 \cdot 1.38$
8	$19.7 = 2.2+$ $+77 \cdot 0.23$	$156.1 = 4.2+$ $+77 \cdot 1.97$	$28.7 = 5.2+$ $+68 \cdot 0.35$	$85.9 = 2.2+$ $+43 \cdot 1.95$	$84.8 = 72.5+$ $+19 \cdot 0.65$

TABLE 4

Timing results for Problem `20284.mlp`, $n = 20284$. *Parallel* $(I + G)$*-solve.*

Num. proc.	Jacobi	BSSOR LOB (M=1728)	direct LOB (SMW) (M=1728)	BSSOR orig. ODB
1	$1806.9 = 60.2+$ $+920 \cdot 1.90$	$3688.0 = 97.9+$ $+1000 \cdot 3.59$	$1334.7 = 369.3+$ $+216 \cdot 4.47$	$3608.2 = 60.3+$ $+1000 \cdot 3.55$
3		$3292.6 = 45.4+$ $+1000 \cdot 3.25$	$384.2 = 110.5+$ $+174 \cdot 1.57$	
9		$2921.2 = 20.1+$ $+1000 \cdot 2.90$	$124.3 = 39.8+$ $+164 \cdot 0.52$	

TABLE 5

Timing results for Problem `eq8`, $n = 13824$. *Parallel* $(I + G)$*-solve. For Jacobi method, the iteration process is aborted because* β *(a CGSTAB parameter) becomes zero. When the number of iterations shown is 1000, it means that convergence has not been reached.*

compared to 0.19 of block SSOR. Sometimes, even superlinear speedup is observed (since more cache memory is used.) Even for M as high as 1000-2000, parallel efficiency of the SMW-applied direct LOB is at least 0.7–0.8.

Further note that often fewer iterations are required for direct LOB than for BSSOR LOB with the same off-diagonal blocks, and especially than for Jacobi method. This indicates that direct LOB provides a better quality preconditioner. For the problem `eq8`, which is harder than the two others, we see that direct LOB achieves the desired degree of convergence after some 100 or 200 iterations, while other methods fail to converge. These results confirm that LRA SMW can provide parallel efficiency while maintaining and even improving the quality of preconditioning that block SSOR provides.

A. Appendix to 4.3. Proof of (30). THEOREM A.1. *Suppose A is a $n_1 \times n_2$ matrix A, X is an $n_2 \times M$ matrix of the rank M, Y is an $n_1 \times M$ matrix of the rank M, and $H = Y^{\mathrm{T}}AX$ is non-singular. Then for any matrix B that simultaneously satisfies the two conditions*

$$BX = AX, \tag{61}$$

$$Y^{\mathrm{T}}B = Y^{\mathrm{T}}A \tag{62}$$

$\operatorname{rank} B \geq M$; only one rank-$M$ matrix B satisfies these conditions, and it is given by the formula

$$B = AXH^{-1}Y^{\mathrm{T}}A \tag{63}$$

Proof. Simple substitution shows that B given by (63) indeed satisfies the conditions (61,62). On the other hand, suppose that B satisfies the conditions (61,62), and $\operatorname{rank} B = r \leq M$. Then B can be represented as $B = UV^{\mathrm{T}}$ with r-column matrices U and V. It follows from (61) that

$$H = Y^{\mathrm{T}}AX = Y^{\mathrm{T}}BX = (Y^{\mathrm{T}}U)(V^{\mathrm{T}}X). \tag{64}$$

By the assumption the $M \times M$ matrix H is non-singular; therefore, $\operatorname{rank} H = M$. This implies, for the factors in (64), that

$$\operatorname{rank}(Y^{\mathrm{T}}U) \geq M, \tag{65}$$

$$\operatorname{rank}(XV^{\mathrm{T}}) \geq M. \tag{66}$$

Since $(Y^{\mathrm{T}}U)$ has only M rows and $r \leq M$ columns, (65) means that $(Y^{\mathrm{T}}U)$ is a non-singular square $M \times M$ matrix. This makes possible to derive, from $AX = U(V^{\mathrm{T}}X)$, the fact that

$$U = AX(V^{\mathrm{T}}X)^{-1} \tag{67}$$

In a similar fashion one can show that $(Y^{\mathrm{T}}U)$ is a square non-singular matrix and

$$V^{\mathrm{T}} = (Y^{\mathrm{T}}U)^{-1}Y^{\mathrm{T}}A \tag{68}$$

From (67,68) it follows that indeed $B = UV^{\mathrm{T}} = AX(V^{\mathrm{T}}X)^{-1}(Y^{\mathrm{T}}U)^{-1}Y^{\mathrm{T}}A = AXH^{-1}Y^{\mathrm{T}}A$, i.e. (63) holds. □

REFERENCES

[1] F. Alvarado, *Ordering schemes for partitioned sparse inverses*, 1989. SIAM Symposium on Sparse Matrices, Salishan Lodge, Gleneden Beach, OR, May 22–24 1989.

[2] E. Anderson, *Parallel implementation of preconditioned conjugate gradient methods for solving sparse systems of linear equations*, Master's thesis, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., 1988.

[3] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1 ed., 1994.

[4] H. V. der Vorst, *Bi–CGSTAB: A fast and smoothly converging variant of Bi–CG for the solution of nonsymmetric linear systems*, SIAM Journal of Scientific and Statistical Computing, 13 (1992), pp. 631–644.

[5] D. Gannon, S. X. Yang, and P. Beckman, *User Guide for a Portable Parallel C++ Programming System, pC++*, Department of Computer Science and CICA, Indiana University, Bloomington, IN, 1994. Available via World Wide Web at http://www.extreme.indiana.edu/sage/pcxx_ug/pcxx_ug.html.

[6] A. George and W.-H. Liu, *The Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Engelwood Cliffs, NJ, 1981.

[7] G. Golub and C. V. Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, 2 ed., 1989.

[8] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1 ed., 1985.

[9] J. Meijerink and H. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[10] V. Meñkov, *Solving block linear systems with low-rank off-diagonal blocks is easily parallelizable*, 1995. Submitted to 1996 Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, April 9–13 1996. Available via World Wide Web at http://ftp.cs.indiana.edu/pub/vmenkov/lowrank/cm96.dvi.

[11] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[12] A. Yeremin and L. Kolotilina, *On a family of two–level preconditionings of the incomplete block factorization type*, Sov. J. Numer. Anal. Math. Modeling, 1 (1986), pp. 293–320.