Preconditioners based on the Alternating-Direction-Implicit algorithm for the 2D steady-state diffusion equation with orthotropic heterogeneous coefficients ☆

Longfei Gao[a], Victor M. Calo[b]

[a] *Applied Mathematics & Computational Science, King Abdullah University of Science and Technology*
[b] *Center for Numerical Porous Media, Applied Mathematics & Computational Science, Earth Science & Engineering, King Abdullah University of Science and Technology*

Abstract

In this paper, we combine the Alternating Direction Implicit (ADI) algorithm with the concept of preconditioning and apply it to linear systems discretized from the 2D steady-state diffusion equations with orthotropic heterogeneous coefficients by the finite element method assuming tensor product basis functions. Specifically, we adopt the compound iteration idea and use ADI iterations as the preconditioner for the outside Krylov subspace method that is used to solve the preconditioned linear system. An efficient algorithm to perform each ADI iteration is crucial to the efficiency of the overall iterative scheme. We exploit the Kronecker product structure in the matrices, inherited from the tensor product basis functions, to achieve high efficiency in each ADI iteration. Meanwhile, in order to reduce the number of Krylov subspace iterations, we incorporate partially the coefficient information into the preconditioner by exploiting the local support property of the finite element basis functions. Numerical results demonstrated the efficiency and quality of the proposed preconditioner.

Keywords: alternating direction implicit, Kronecker product, finite element method, tensor product basis functions, preconditioning, compound iteration

1. Introduction

The Alternating Direction Implicit (ADI) algorithm, which belongs to the category of matrix-splitting iterative methods, was first proposed almost six decades ago for solving

☆This work was supported in part by the King Abdullah University of Science and Technology (KAUST) Center for Numerical Porous Media and by an Academic Excellence Alliance program award from KAUST's Global Collaborative Research under the title "Seismic wave focusing for subsurface imaging and enhanced oil recovery."

Email addresses: `longfei.gao@kaust.edu.sa` (Longfei Gao), `victor.calo@kaust.edu.sa` (Victor M. Calo)

parabolic and elliptic partial differential equations, see [1–4]. With a proper set of acceleration parameters, the ADI algorithm can be very powerful, given that each iteration can be performed efficiently. For example, consider the 2D Poisson equation, defined on a rectangle with full Dirichlet boundary conditions:

$$-\left(u_{,xx} + u_{,yy}\right) = f. \tag{1}$$

When discretized by the finite element method, see [5–7], with tensor product basis functions, each of $u_{,xx}$ and $u_{,yy}$ leads to a matrix that possesses the Kronecker product structure, see [8–10]. This Kronecker product structure can be exploited to perform each ADI iteration efficiently, see [11].

However, this elegant algorithm is fragile and its application is limited to certain model problems only, mainly due to the fact that the Kronecker product structure can be easily destroyed by the appearance of complex coefficients. When encountered with complex coefficients, we resort to the compound iteration idea, originally proposed in [12], to extend the practice of the ADI algorithm. Specifically, we first construct an approximate matrix for which the ADI algorithm can be applied efficiently. Then, a fixed number of ADI iterations is applied to invert this approximate matrix, serving as the preconditioner for the original matrix. These iterations are called the inner iterations. Outside these ADI iterations, another iterative scheme, called the outer iterations, is applied to solve the preconditioned linear system.

The convergence speed of this compound iteration scheme depends on the quality of the approximate matrix. In this paper, we focus on linear systems arising from the finite element discretization with tensor product basis functions of the 2D steady-state diffusion equations with orthotropic heterogeneous coefficients. For these linear systems, the ADI algorithm is not applicable directly due to the appearance of these coefficients. By exploiting the local support property of the basis functions used in the finite element discretization, we are able to construct an approximate matrix that can incorporate partially the coefficient information and meanwhile, enable the use of the ADI algorithm. Various numerical tests are performed to verify the efficiency of the compound iteration scheme with the proposed preconditioner.

The rest of this paper is organized as follows. In section 2, we briefly describe the ADI algorithm. In section 3, we present the definition and properties of the Kronecker product, as well as its associated efficient algorithms. In section 4, we examine the linear system arising from the 2D Poisson equation by the finite element discretization with tensor product basis functions and demonstrate how the Kronecker product structure can be combined with the ADI algorithm to solve this linear system efficiently. In section 5, we discuss the 2D steady-state diffusion equations with orthotropic heterogeneous coefficients. We present a preconditioner based on the ADI algorithm and the local support property of the basis functions. Numerical examples are presented in section 6 to demonstrate the performance of this preconditioner. Finally, we conclude with section 7.

2. Alternating direction implicit algorithm

The Alternating Direction Implicit (ADI) algorithm was originally proposed in [1] in 1955 as an iterative method for parabolic and elliptic partial differential equations (PDEs). We consider the linear system:

$$(K^X + K^Y)b = \mathcal{F}, \tag{2}$$

where K^X and K^Y are square matrices while b and $\mathcal{F}$ are vectors of compatible sizes.

The following scheme can be used to solve (2) iteratively:

$$\begin{aligned}
(r^{(k)}\Sigma + K^X)b^{(k+\frac{1}{2})} &= (r^{(k)}\Sigma - K^Y)b^{(k)} + \mathcal{F}; && \text{(3a)}\\
(r^{(k)}\Sigma + K^Y)b^{(k+1)} &= (r^{(k)}\Sigma - K^X)b^{(k+\frac{1}{2})} + \mathcal{F}, && \text{(3b)}
\end{aligned}$$

where $k = 0, 1, 2, \cdots$ stands for the iteration step, Σ is a matrix with the same size of K^X or K^Y, $b^{(k)}$ is the approximation of the solution vector b at iteration step k and $r^{(k)}$ is a scalar number, called the acceleration parameter, to be explained with detail in section 4. If both (3a) and (3b) can be solved efficiently and the total amount of iterations is reasonable, the overall cost of this algorithm can be very low. This is the underlying idea of the ADI algorithm.

At first glance, the assumption that both (3a) and (3b) can be solved efficiently might seem questionable. However, if linear system (2) arises from the finite difference discretization with the five-point stencil of the 2D Poisson equation (1):

$$-\left(u_{,xx} + u_{,yy}\right) = f,$$

where $u_{,xx}$ and $u_{,yy}$ correspond to K^X and K^Y, respectively, then both K^X and K^Y can be reformulated as tridiagonal matrices after suitable permutations.

An efficient algorithm (the tridiagonal matrix algorithm, also known as the Thomas algorithm, see [13, 14]) can be applied to invert non-singular tridiagonal matrices efficiently. Therefore, if adding the extra term $r^{(k)}\Sigma$ does not destroy this tridiagonal structure in K^X and K^Y, for instance, when Σ is the identity matrix, (3a) and (3b) can be solved efficiently. This is the original motivation behind the development of this iterative scheme.

However, there are other cases where the individual equations in (3) can be solved efficiently. One noticeable example appears when the Poisson equation (1) is discretized by the finite element method with tensor product basis functions, see [11]. In this case, both K^X and K^Y, which still correspond to $u_{,xx}$ and $u_{,yy}$ in (1), respectively, possess the Kronecker product property. We briefly explain the Kronecker product in section 3.

The inverse of a non-singular Kronecker product matrix can be applied efficiently, see [15–17]. If in addition, the extra term $r^{(k)}\Sigma$ does not destroy the Kronecker product structure and the invertibility of K^X and K^Y, both (3a) and (3b) can be solved efficiently and therefore, the ADI algorithm is still applicable.

3. Kronecker product

3.1. Definition and properties

The Kronecker product, denoted by $\otimes$, is a special name for the tensor product when restricted on matrices. Symbolically, for matrices A and B of arbitrary sizes,

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix},$$

where a_{ij} is the entry of A at the ith row and jth column with i ranging from 1 to m and j ranging from 1 to n. In other words, $A \otimes B$ means that every entry in the first matrix A is replaced by the second matrix B and then scaled by that replaced entry.

The Kronecker product has many useful properties. We list several of them here:

- Mixed-product:
$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD);$$
- Inverse:
$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1};$$
- Transpose:
$$(A \otimes B)^{T} = A^{T} \otimes B^{T};$$
- Associative:
$$(A \otimes B) \otimes C = A \otimes (B \otimes C);$$
- Bilinear:
$$\begin{aligned} &A \otimes (B + C) = (A \otimes B) + (A \otimes C), \\ &(A + B) \otimes C = (A \otimes C) + (B \otimes C), \\ &(cA) \otimes B = A \otimes (cB) = c(A \otimes B). \end{aligned}$$

In the mixed-product and associative properties, the matrices need to have compatible sizes such that the multiplications make sense; in the inverse property, both A and B need to be invertible; in the bilinear property, c stands for a scaler number. For derivations of the above properties, see [8–10]. For more details about the Kronecker product, including its applications, one can also consult [18–23].

3.2. Efficient algorithms

An efficient algorithm for inverting a Kronecker product matrix can be developed by exploiting its structure, for instance, see [15–17]. We restate this algorithm here for completeness. For simplicity, we first introduce the operators Vec and Mat, which will be useful in the upcoming discussion. These two operators are borrowed from [10] and [24]. In their definitions given below, we also borrow the colon notation ':' from MATLAB [25] to specify index values. For instance, '$1 : m$' means the corresponding index varies from 1 to m, while ':' without the starting and ending value means the corresponding index varies through all its possible values.

Definition 1. *Let $F \in R^{m\times n}$, $Vec(F)$ is the operator that returns a column vector of length mn by stacking all the columns of F together under the natural order:*

$$Vec(F) \equiv \begin{bmatrix} F(:,1) \\ \vdots \\ F(:,n) \end{bmatrix}.$$

Definition 2. *Let f be a column vector of length mn, $Mat(f,m,n)$ is the operator that returns a matrix of size $m \times n$ by chopping f into n pieces of length m and putting these pieces into the matrix under the natural order:*

$$Mat(f,m,n) \equiv \left[f(1:m), \cdots, f\left((n-1)m+1:nm\right)\right].$$

With the above operators, we now present Algorithm 1 to solve the following linear system that involves a Kronecker product matrix:

$$\left(M^y \otimes M^x\right) x = f, \tag{4}$$

where matrices M^y and M^x have sizes $N_y \times N_y$ and $N_x \times N_x$, respectively, while x and f are column vectors with length N_yN_x.

Algorithm 1.

```
1: $F = Mat(f, N_x, N_y)$;
2: $T = zeros(N_x, N_y)$;
3: $X = zeros(N_x, N_y)$;
4: for $j = 1 : N_y$ do
5:     $T(:,j) = M^x \backslash F(:,j)$;
6: end for
7: for $i = 1 : N_x$ do
8:     $X(i,:)^T = M^y \backslash T(i,:)^T$;
9: end for
10: $x = Vec(X)$;
```

In Algorithm 1, we also borrow the commands '*zeros*' and '\' from MATLAB, where $zeros(m,n)$ returns an m-by-n matrix of zeros while $A\backslash b$ returns the solution x of $Ax = b$.

A visualization of Algorithm 1 is presented in Figure 1: We first apply M^x to all the columns of the original data matrix F and then apply M^y to all the rows of the intermediate data matrix T.

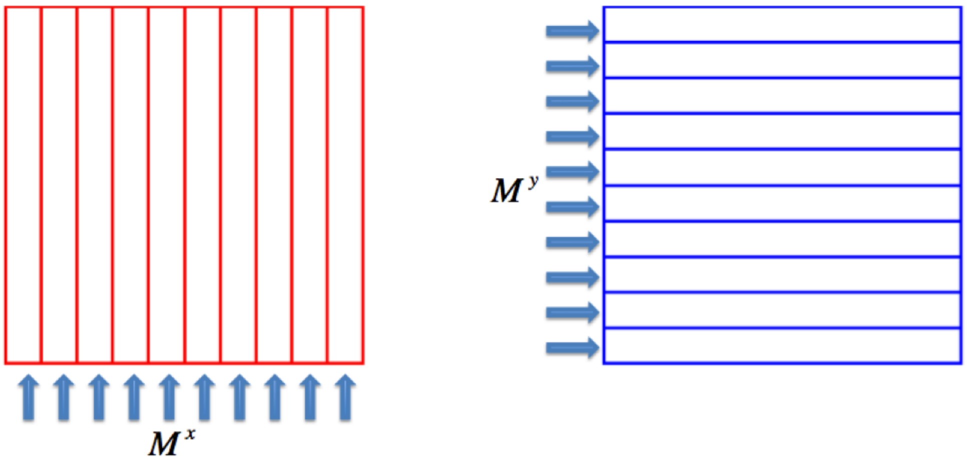


Figure 1: Visualization of Algorithm 1.

If we use a direct method to apply the inverses of M^x and M^y in Algorithm 1, and if in addition, both M^x and M^y are dense, the factorization cost is $\mathcal{O}\big((N_x)^3 + (N_y)^3\big)$ while the total substitution cost in Algorithm 1 is $\mathcal{O}\big((N_x)^2 N_y + (N_y)^2 N_x\big)$. However, if both M^x and M^y are banded diagonal matrices with bandwidth independent of the matrix size, the factorization cost is $\mathcal{O}\left(N_x + N_y\right)$ while the total substitution cost in Algorithm 1 is $\mathcal{O}\left(N_x N_y\right)$. This banded diagonal structure is common for 1D matrices arising from finite element discretizations due to the local support of basis functions.

A similar algorithm exists for applying the matrix vector multiplication with a Kronecker product matrix. For instance, to multiply $M^y \otimes M^x$ with x and assign the result to f, we can use the following algorithm:

Algorithm 2.

```
 1: $F = zeros(N_x, N_y)$;
 2: $T = zeros(N_x, N_y)$;
 3: $X = Mat(x, N_x, N_y)$;
 4: for $j = 1 : N_y$ do
 5:     $T(:, j) = M^x * X(:, j)$;
 6: end for
 7: for $i = 1 : N_x$ do
 8:     $F(i, :)^T = M^y * T(i, :)^T$;
 9: end for
10: $f = Vec(F)$;
```

In Algorithm 2, we use '$*$' to denote the 1D matrix vector multiplication.

In the next section, we demonstrate how to combine the Kronecker product property with the ADI algorithm for a model problem.

4. Model problem: the 2D Poisson equation

The idea of combining the Kronecker product with the ADI algorithm has been discussed in [11]. Below, we briefly explain it for completeness. Let us consider a model problem, i.e., the 2D Poisson equation, defined on a rectangle, as descreibed in (1): $-(u_{,xx} + u_{,yy}) = f$, to demonstrate the mechanism of combining the Kronecker product properties with the ADI algorithm, as well as its efficiency. For simplicity, we assume zero Dirichlet boundary conditions on the boundary. However, the method presented here can also be applied to other boundary conditions.

4.1. Kronecker product structure in the stiffness matrix

Discretized by the finite element method with a set of basis functions, denoted with B, model problem (1) leads to linear system:

$$(K^X + K^Y)b = \mathcal{F},$$

as shown in (2), where K^X and K^Y correspond to $u_{,xx}$ and $u_{,yy}$, respectively. $\left(K^X + K^Y\right)$ is the stiffness matrix.

We denote the partial derivative of B with respect to x as $B_{,x}$ and its partial derivative with respect to y as $B_{,y}$. If we list B, $B_{,x}$ and $B_{,y}$ as column vectors, K^X and K^Y can be expressed in the following succinct forms:

$$K^X = \int_\Omega (B_{,x})(B_{,x})^T d\Omega \quad \text{and} \quad K^Y = \int_\Omega (B_{,y})(B_{,y})^T d\Omega. \tag{5}$$

If B is built as the tensor product of two 1D basis sets B^y and B^x, i.e., $B = B^y \otimes B^x$, where B^y and B^x are also listed as column vectors, we have:

$$B_{,y} = B^y_{,y} \otimes B^x \quad \text{and} \quad B_{,x} = B^y \otimes B^x_{,x}, \tag{6}$$

where $B^x_{,x}$ and $B^y_{,y}$ stand for the derivatives of B^x and B^y, respectively. After substituting these two relations into (5) and applying the mixed-product property, we arrive at:

$$K^X = \int_\Omega \left(B^y(B^y)^T\right) \otimes \left(B^x_{,x}(B^x_{,x})^T\right) d\Omega \quad \text{and} \quad K^Y = \int_\Omega \left(B^y_{,y}(B^y_{,y})^T\right) \otimes \left(B^x(B^x)^T\right) d\Omega. \tag{7}$$

Since the integration domain Ω is rectangular, the integrals in (7) can be separated into products of 1D integrals, which leads to:

$$\begin{aligned} K^X &= \left(\int_y B^y(B^y)^T dy\right) \otimes \left(\int_x B^x_{,x}(B^x_{,x})^T dx\right); \\ K^Y &= \left(\int_y B^y_{,y}(B^y_{,y})^T dy\right) \otimes \left(\int_x B^x(B^x)^T dx\right). \end{aligned} \tag{8}$$

Noticing that $\left(\int_y B^y(B^y)^T dy\right)$ and $\left(\int_x B^x(B^x)^T dx\right)$ are the 1D mass matrices built with B^y and B^x, respectively, while $\left(\int_y B^y_{,y}(B^y_{,y})^T dy\right)$ and $\left(\int_x B^x_{,x}(B^x_{,x})^T dx\right)$ are the corresponding 1D stiffness matrices, we can express K^X and K^Y in the following concise forms:

$$K^X = M^y \otimes K^x \quad \text{and} \quad K^Y = K^y \otimes M^x, \tag{9}$$

with the help of the following notations:

$$\begin{aligned} M^y &= \int_y B^y(B^y)^T dy, & M^x &= \int_x B^x(B^x)^T dx; \\ K^y &= \int_y B^y_{,y}(B^y_{,y})^T dy, & K^x &= \int_x B^x_{,x}(B^x_{,x})^T dx. \end{aligned}$$

Moreover, we define M, the corresponding 2D mass matrix, as

$$M = M^y \otimes M^x, \tag{10}$$

which plays an important role in the ADI algorithm discussed in the next section.

4.2. Applying the ADI algorithm to the model problem

In this section, we demonstrate how to apply the ADI algorithm to solve linear system $\left(K^X + K^Y\right) b = \mathcal{F}$ that corresponds to the finite element discretization of model problem (1) with tensor product basis functions.

Knowing that both K^X and K^Y possess the Kronecker product structure and that Algorithm 1 can efficiently apply the inverse of a Kronecker product matrix (given that it is invertible), we first want to find a suitable matrix Σ for the ADI algorithm, as shown in (3), such that the Kronecker product structure is preserved. The mass matrix M, as defined in (10), is a natural choice since by the bilinear property of the Kronecker product, we have:

$$\begin{aligned} r^{(k)}M + K^X &= r^{(k)}M^y \otimes M^x + M^y \otimes K^x = M^y \otimes \left(r^{(k)}M^x + K^x\right); \\ r^{(k)}M + K^Y &= r^{(k)}M^y \otimes M^x + K^y \otimes M^x = \left(r^{(k)}M^y + K^y\right) \otimes K^x. \end{aligned}$$

Since the mass matrix M is symmetric positive-definite, the extra term $r^{(k)}M$, with $r^{(k)} > 0$, ensures that the matrices to be inverted in (3a) and (3b) are non-singular, even for the case when Neumann boundary conditions are imposed on two parallel edges of the rectangular domain such that either K^X or K^Y is singular. But more importantly, this extra term can accelerate the convergence of the ADI algorithm, if chosen properly.

To see why it is so, we first subtract the following equivalent forms of (2):

$$\begin{aligned} (r^{(k)}M + K^X)b &= (r^{(k)}M - K^Y)b + \mathcal{F}, \\ (r^{(k)}M + K^Y)b &= (r^{(k)}M - K^X)b + \mathcal{F}, \end{aligned}$$

from (3a) and (3b), respectively, obtaining:

$$(r^{(k)}M + K^X)e^{(k+\frac{1}{2})} = (r^{(k)}M - K^Y)e^{(k)}, \tag{11a}$$

$$(r^{(k)}M + K^Y)e^{(k+1)} = (r^{(k)}M - K^X)e^{(k+\frac{1}{2})}, \tag{11b}$$

where $e^{(i)} = b^{(i)} - b$ for $i = k,\ k + \frac{1}{2}$ and $k + 1$.

Substituting $e^{(k+\frac{1}{2})}$ from (11a) into (11b), we get

$$e^{(k+1)} = \left[(r^{(k)}M + K^Y)^{-1}(r^{(k)}M - K^X)(r^{(k)}M + K^X)^{-1}(r^{(k)}M - K^Y)\right] e^{(k)}. \tag{12}$$

With the following definition:

$$\mathcal{P}^{(k)} = (r^{(k)}M + K^Y)^{-1}(r^{(k)}M - K^X)(r^{(k)}M + K^X)^{-1}(r^{(k)}M - K^Y), \tag{13}$$

we can write the relations between the errors at different iterations as:

$$e^{(k+1)} = \left(\mathcal{P}^{(k)}\right) e^{(k)}; \tag{14}$$

$$e^{(k+1)} = \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right) e^{(0)}. \tag{15}$$

From the definition of $\mathcal{P}^{(k)}$ in (13), as well as the error relations in (14) and (15), it is clear that the convergence speed of the ADI algorithm depends on the choice of $r^{(k)}$. We refer to $r^{(k)}$ as the acceleration parameter in the rest of this paper.

4.3. Selection of the acceleration parameters

To see how to determine a good set of acceleration parameters for linear system (2) that corresponds to the model problem, we first present a lemma regarding the generalized eigenvalue decomposition of matrices K^X and K^Y.

Lemma 1. *Assume the generalized eigenvalue decomposition for 1D matrices K^x and K^y exist:*

$$K^xV^x = M^xV^xD^x; \tag{16a}$$

$$K^yV^y = M^yV^yD^y, \tag{16b}$$

where D^x and D^y are diagonal matrices with their diagonal entries being the generalized eigenvalues, then for the 2D matrices $K^X = M^y \otimes K^x$, $K^Y = K^y \otimes M^x$ and $M = M^y \otimes M^x$, we have the following generalized eigenvalue decomposition:

$$K^XV = MVD^X; \tag{17a}$$

$$K^YV = MVD^Y, \tag{17b}$$

with $V = V^y \otimes V^x$, $D^X = I^y \otimes D^x$ and $D^Y = D^y \otimes I^x$ where I^x and I^y stand for the identity matrices with the same sizes as D^x and D^y, respectively.

If we further have relations:

$$(V^x)^T M^x V^x = I^x; \tag{18a}$$
$$(V^y)^T M^y V^y = I^y, \tag{18b}$$

then the following relation also holds:

$$V^T M V = I^y \otimes I^x. \tag{19}$$

Proof. By the properties of the Kronecker product, particularly the mixed-product property and the associative property, we can easily verify (17a), (17b) and (19) as follows:

$$\begin{array}{rcl}
K^X V &=& (M^y \otimes K^x)(V^y \otimes V^x) \\
&=& (M^y V^y) \otimes (K^x V^x) \\
&=& (M^y V^y I^y) \otimes (M^x V^x D^x) \\
&=& (M^y \otimes M^x)(V^y \otimes V^x)(I^y \otimes D^x) \\
&=& M V D^X; \\
\\
K^Y V &=& (K^y \otimes M^x)(V^y \otimes V^x) \\
&=& (K^y V^y) \otimes (M^x V^x) \\
&=& (M^y V^y D^y) \otimes (M^x V^x I^x) \\
&=& (M^y \otimes M^x)(V^y \otimes V^x)(D^y \otimes I^x) \\
&=& M V D^Y; \\
V^T M V &=& (V^y \otimes V^x)^T (M^y \otimes M^x)(V^y \otimes V^x) \\
&=& \big((V^y)^T M^y V^y\big) \otimes \big((V^x)^T M^x V^x\big) \\
&=& I^y \otimes I^x.
\end{array}$$

□

Relations (17a) and (17b) tell us that K^X and K^Y share the same set of generalized eigenvectors. Denoting the columns of V as v_i, for $i = 1, \ldots, N$, where N is the dimension of matrix V, we can write:

$$K^X v_i = \lambda_i^X M v_i; \tag{20a}$$
$$K^Y v_i = \lambda_i^Y M v_i, \tag{20b}$$

where λ_i^X and λ_i^Y are the diagonal entries in D^X and D^Y that correspond to v_i, respectively. Moreover, we call v_i and v_j M-orthonormal since according to (19), we have:

$$(v_i)^T M v_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

To clarify, in the following content, when using the terminology 'generalized eigenvalue/eigenvector' of K^X or K^Y, we refer to the generalized eigenvalue/eigenvector defined by (20a) or (20b), respectively. Similarly, the generalized eigenvalue/eigenvector of K^x or K^y corresponds to (16a) or (16b), respectively.

Lemma 1 also tells us that the generalized eigenvalues of K^X are repeated: K^X has the same set of distinct generalized eigenvalues as K^x. Analogous result holds for K^Y and K^y.

Moreover, coming from the finite element discretization of the 2D Poisson equation (1), K^x and K^y are symmetric positive semi-definite while M^x and M^y are symmetric positive-definite. This guarantees that all the diagonal entries of D^x and D^y are non-negative real numbers and there exist full rank matrices V^x and V^y such that (16) and (18) hold. Under these conditions, the columns of V form a basis for vector space R^N.

With all these properties established above, we can now proceed to analyze the relations between errors at different iterations and the selection of acceleration parameters. We start by introducing the M-norm:

Definition 3. *Given a symmetric positive-definite matrix M, the M-norm of a vector v with compatible size is defined as:*

$$\|v\|_M = \left(v^T M v\right)^{\frac{1}{2}}.$$

For matrix A with compatible size, the induced matrix M-norm is defined as:

$$\|A\|_M = \sup_{v \neq 0} \frac{\|Av\|_M}{\|v\|_M}.$$

With the above definitions, we have the following theorem that states the relations between errors at different iterations:

Theorem 1. *If K^X, K^Y and M are $N \times N$ symmetric positive-definite matrices and relations* (17a) *and* (17b) *hold, then*

$$\left\|e^{(k+1)}\right\|_M \leq \left\|\prod_{j=0}^{k} \mathcal{P}^{(j)}\right\|_M \cdot \left\|e^{(0)}\right\|_M, \tag{21}$$

where

$$\left\|\prod_{j=0}^{k} \mathcal{P}^{(j)}\right\|_M = \max_{1\leq i\leq N} \left\{\prod_{j=0}^{k} \left|\frac{r^{(j)} - \lambda_i^X}{r^{(j)} + \lambda_i^X}\right| \cdot \left|\frac{r^{(j)} - \lambda_i^Y}{r^{(j)} + \lambda_i^Y}\right|\right\} < 1, \tag{22}$$

and

$$\left\|\prod_{j=0}^{k+1} \mathcal{P}^{(j+1)}\right\|_M < \left\|\prod_{j=0}^{k} \mathcal{P}^{(j)}\right\|_M. \tag{23}$$

Proof. Definition 3 immediately leads us to inequality (21), according to relation (15).

Now let us focus on the term $\left\|\prod_{j=0}^{k} \mathcal{P}^{(j)}\right\|_M$, which bounds the error reduction rate after k iterations. For any generalized eigenvector v_i and a positive number $r^{(k)}$, we have the following relations that can be easily verified based on (20a) and (20b):

$$(r^{(k)}M - K^X)v_i = (r^{(k)} - \lambda_i^X)Mv_i;$$
$$(r^{(k)}M - K^Y)v_i = (r^{(k)} - \lambda_i^Y)Mv_i,$$

and

$$\frac{1}{(r^{(k)}+\lambda_i^X)}v_i = (r^{(k)}M+K^X)^{-1}Mv_i;$$
$$\frac{1}{(r^{(k)}+\lambda_i^Y)}v_i = (r^{(k)}M+K^Y)^{-1}Mv_i.$$

These relations enable us to look through the effect of applying $\mathcal{P}^{(k)}$ on a single generalized eigenvector v_i:

$$\mathcal{P}^{(k)}v_i = \left(\frac{r^{(k)}-\lambda_i^X}{r^{(k)}+\lambda_i^X}\right)\cdot\left(\frac{r^{(k)}-\lambda_i^Y}{r^{(k)}+\lambda_i^Y}\right)v_i, \tag{24}$$

and similarly for $\left(\prod_{j=0}^{k}\mathcal{P}^{(j)}\right)$:

$$\left(\prod_{j=0}^{k}\mathcal{P}^{(j)}\right)v_i = \left(\prod_{j=0}^{k}\left(\frac{r^{(j)}-\lambda_i^X}{r^{(j)}+\lambda_i^X}\right)\cdot\left(\frac{r^{(j)}-\lambda_i^Y}{r^{(j)}+\lambda_i^Y}\right)\right)v_i. \tag{25}$$

Since $\{v_i\}_{i=1}^N$ are M-orthonormal with each other, while $r^{(k)}$, λ_i^X and λ_i^Y are all positive numbers, we readily have the inequalities (22) and (23). □

Remark 1. *The symmetric positive-definite requirement on K^X and K^Y in Theorem 1 can be relaxed so that only one of K^X and K^Y need to be symmetric positive-definite while the other can be symmetric positive semi-definite. Being able to allow this relaxation is important for the case when Neumann boundary conditions are imposed on two parallel edges of the rectangular domain.*

Relations (21) and (22) motivate the investigation on the following min-max problem in order to select a good set of acceleration parameters:

$$\{s^{(j)}\}_{j=0}^k = \underset{\{r^{(j)}\}_{j=0}^k}{\arg\min}\left\{\max_{1\le i\le N}\left\{\prod_{j=0}^{k}\left|\frac{r^{(j)}-\lambda_i^X}{r^{(j)}+\lambda_i^X}\right|\cdot\left|\frac{r^{(j)}-\lambda_i^Y}{r^{(j)}+\lambda_i^Y}\right|\right\}\right\}. \tag{26}$$

However, (26) is not practical since it involves all the eigenvalues $\{\lambda_i^X\}_{i=1}^N$ and $\{\lambda_i^X\}_{i=1}^N$. Instead, the following relaxed min-max problem can serve as a good surrogate:

$$\{s^{(j)}\}_{j=0}^k = \underset{\{r^{(j)}\}_{j=0}^k}{\arg\min}\left\{\max_{0<\alpha\le x\le\beta}\left\{\left|\prod_{j=0}^{k}\frac{r^{(j)}-x}{r^{(j)}+x}\right|\right\}\right\}, \tag{27}$$

where $0<\alpha\le\lambda_i^X,\lambda_i^Y\le\beta$ for $1\le i\le N$.

The relaxed min-max problem (27) has been thoroughly investigated in [26]. It has been shown that there exists a unique set of distinct parameters $\{s^{(j)}\}_{j=0}^k$ as the solution of (27). More importantly, [26] provides a theoretically sound and numerically elegant algorithm to find these parameters for the special cases when $k=2^i$ with i a nonnegative integer, see also [27, 28]. We call these parameters the optimal acceleration parameters. Moreover, [26] also provides a tight upper bound on the error reduction rate associated with these optimal acceleration parameters.

4.4. Efficiency of the ADI algorithm

In Table 1, we demonstrate the performance of the ADI algorithm with optimal acceleration parameters of the model problem $(K^X + K^Y)b = \mathcal{F}$, discretized from (1) with different mesh sizes.

The number of optimal acceleration parameters is denoted with k, while 32×32, 128×128 and 512×512 elements uniform meshes are used. The actual relative errors (in l_2 norm) are recorded in columns tagged as 'observed' while their upper bounds provided from [26] are recorded in columns tagged as 'predicted'. For all the simulations involved, zero initial points and random right-hand sides are used.

	32×32		128×128		512×512	
k	observed	predicted	observed	predicted	observed	predicted
1	8.90E-01	8.92E-01	9.71E-01	9.72E-01	9.93E-01	9.93E-01
2	3.77E-01	3.78E-01	6.17E-01	6.20E-01	7.85E-01	7.88E-01
4	3.84E-02	3.86E-02	1.21E-01	1.21E-01	2.37E-01	2.38E-01
8	3.71E-04	3.72E-04	3.66E-03	3.66E-03	1.45E-02	1.46E-02
16	3.44E-08	3.46E-08	3.35E-06	3.35E-06	5.28E-05	5.29E-05
32	1.08E-14	2.99E-16	2.70E-12	2.81E-12	6.98E-10	7.01E-10
64	1.04E-14	2.24E-32	1.02E-13	1.97E-24	5.03E-13	1.23E-19

Table 1: Errors and their upper bounds.

Table 1 demonstrates the fast convergence speed of the ADI algorithm. It becomes more apparent when we compare its error reduction rates against the conjugate gradient method without preconditioner for 64 iterations: which is 1.70E-09, 5.06E-03 and 4.88E-01 for meshes with 32×32, 128×128 and 512×512 elements, respectively.

Moreover, the upper bound is extremely tight and therefore, can serve as a good indicator for deciding how many optimal acceleration parameters shall be used, given a desired relative error. For a few cases shown in Table 1, the observed relative error are actually bigger than their predicted upper bounds. This is due to the effect of round-off error.

4.5. Limitation and generalization

For realistic problems that are more complicated than the presented model problem, the assumption that K^X and K^Y share the same set of generalized eigenvectors is rarely satisfied, see [27]. There exist various attempts on extending the theory and practice of the ADI algorithm to more general cases, among which we find the 'compound iteration' presented in [12] a particularly appealing idea.

In [12], the ADI algorithm is used as the 'inner iteration' to approximately invert an approximate matrix of $\left(K^X + K^Y\right)$ that satisfies this assumption. This iterative scheme serves as the preconditioner while the conjugate gradient method is used as the 'outer iteration' to solve the preconditioned linear system.

In the next section, we propose preconditioners based on this compound iteration idea for more complicated problems.

5. Preconditioning

Earlier attempts on using the ADI iterations as a preconditioner for the Krylov subspace methods can be found in [12, 29–33]. A parallel implementation is discussed in [34]. Here, we apply this idea to the 2D steady-state diffusion equation with orthotropic heterogeneous coefficients. To achieve high efficiency, we exploit the Kronecker product structure in the preconditioner to perform the ADI iterations. Other attempts on exploiting the Kronecker product structure in solving linear systems can be found in [10, 20, 22, 35].

5.1. Steady-state diffusion equation with orthotropic heterogeneous coefficients

We consider the following 2D steady-state diffusion equation defined on a rectangle:

$$-\nabla \cdot \big(\boldsymbol{\kappa}(x,y)\nabla u(x,y)\big) = f(x,y), \tag{28}$$

with orthotropic coefficients:

$$\boldsymbol{\kappa}(x,y) = \begin{bmatrix} \kappa_{11}(x,y) & 0 \\ 0 & \kappa_{22}(x,y) \end{bmatrix}$$

and full Dirichlet boundary conditions.

Similar as in section 4.1, when discretized with the finite element method using the tensor product basis functions $B = B^y \otimes B^x$, (28) leads to the following linear system:

$$\left(\mathbb{K}^X + \mathbb{K}^Y\right) b = \mathcal{F}, \tag{29}$$

where

$$\mathbb{K}^X = \int_\Omega \left(B^y (B^y)^T\right) \otimes \left(B^x_{,x} (B^x_{,x})^T\right) \kappa_{11} d\Omega, \tag{30a}$$

$$\mathbb{K}^Y = \int_\Omega \left(B^y_{,y} (B^y_{,y})^T\right) \otimes \left(B^x (B^x)^T\right) \kappa_{22} d\Omega. \tag{30b}$$

Although the integration domain is still rectangular and tensor product basis functions are used, in general, the integrals in (30a) and (30b) cannot be written as products of 1D integrals due to the appearance of κ_{11} or κ_{22}, which couples the two directions. Therefore, $\mathbb{K}^X$ and $\mathbb{K}^Y$ do not possess the Kronecker product structure and the ADI algorithm shown in section 4 is no longer applicable. For this more complicated problem, we resort to preconditioning and the compound iteration idea.

5.2. The simplest choice: $(K^X + K^Y)^{-1}$

The simplest choice of preconditioner would be $(K^X + K^Y)^{-1}$, with K^X and K^Y as defined in (7). The ADI algorithm can be used to apply $(K^X + K^Y)^{-1}$ efficiently, serving as the inner iteration. Then, the preconditioned linear system can be solved by Krylov subspace methods, serving as the outer iteration. This is precisely the compound iteration idea proposed in [12].

A clarification of terminology is necessary here: $(K^X + K^Y)^{-1}$ is actually not the preconditioner in the strict sense. Instead, the approximate inverse of $(K^X + K^Y)$ induced by the ADI iterations is the preconditioner. However, for simplicity, we still call $(K^X + K^Y)^{-1}$ the preconditioner, assuming there is no ambiguity. Besides, we may also refer to the ADI iterations, or the inner iterations, as the preconditioner.

There are still several questions left regarding this compound iteration idea:

1. What are we really using as the preconditioner and does it have a matrix form?

2. If so, is this matrix symmetric positive-definite such that the conjugate gradient method can be applied as the outer iteration?

3. How many inner iterations are needed?

We answer these questions one by one in the following.

5.2.1. Matrix form of the preconditioner

Recall the relation between the errors at different ADI iterations, as shown in (15):

$$e^{(k+1)} = \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right) e^{(0)},$$

where $e^{(0)} = b^{(0)} - b$ and $e^{(k+1)} = b^{(k+1)} - b$ with b denoting the true solution of linear system $(K^X + K^Y)b = \mathcal{F}$. Substituting $e^{(0)}$ and $e^{(k+1)}$ to the error relation (15), we have:

$$b^{(k+1)} = \left(I - \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)\right) b + \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right) b^{(0)}. \tag{31}$$

If we use zero initial guess for the ADI iterations, i.e., $b^{(0)} = 0$, (31) can be simplified to:

$$b^{(k+1)} = \left(I - \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)\right) b. \tag{32}$$

Further substituting the true solution $b = (K^X + K^Y)^{-1}\mathcal{F}$ into (32) leads to:

$$b^{(k+1)} = \left(I - \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)\right) (K^X + K^Y)^{-1}\mathcal{F}. \tag{33}$$

Now it is clear that the matrix form of the preconditioner can be written as:

$$\left(I - \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)\right) (K^X + K^Y)^{-1}. \tag{34}$$

Therefore, if we fix the number of inner iterations k for each outer iteration, this matrix is fixed as well, thus suitable for serving as the preconditioner of Krylov subspace methods.

5.2.2. Symmetry and positive definiteness of the preconditioner

Since matrix $(\mathbb{K}^X + \mathbb{K}^Y)$ is symmetric positive-definite (SPD), the conjugate gradient (CG) method is the most desirable method for the outer iteration. However, CG requires the preconditioner to be SPD as well, which is yet unclear for this inner iteration preconditioner.

First, let us recall the effect of applying $\left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)$ on a generalized eigenvector v_i, as shown in (24):

$$\left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right) v_i = \left(\prod_{j=0}^{k} \left(\frac{r^{(j)} - \lambda_i^X}{r^{(j)} + \lambda_i^X}\right) \cdot \left(\frac{r^{(j)} - \lambda_i^Y}{r^{(j)} + \lambda_i^Y}\right)\right) v_i.$$

In other words, v_i is an eigenvector of $\left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)$. Using $D_{\mathcal{P}}$ to denote the diagonal matrix with its ith diagonal entry being the eigenvalue of $\left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)$ that corresponds to v_i, we have:

$$\left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right) V = V D_{\mathcal{P}}. \tag{35}$$

On the other hand, according to (17a) and (17b), we can write:

$$\left(K^X + K^Y\right) V = MVD \tag{36}$$

where $D = (I^y \otimes D^x + D^y \otimes I^x)$ is also diagonal.

We can rewrite (35) and (36), respectively, as

$$\left(I - \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)\right) = V\left(I - D_{\mathcal{P}}\right) V^{-1}$$

and

$$\left(K^X + K^Y\right)^{-1} = VD^{-1}V^{-1}M^{-1},$$

which lead us to the following equation:

$$\left(I - \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)\right) \left(K^X + K^Y\right)^{-1} = V\left(\left(I - D_{\mathcal{P}}\right) D^{-1}\right) (MV)^{-1}.$$

Recalling the relation $V^T M V = I$ from Lemma 1, we have:

$$\left(I - \left(\prod_{j=0}^{k} \mathcal{P}^{(j)}\right)\right) \left(K^X + K^Y\right)^{-1} = V\left(\left(I - D_{\mathcal{P}}\right) D^{-1}\right) V^T. \tag{37}$$

From (37), it is clear that if all the diagonal entries of $(I - D_{\mathcal{P}})$ and D are positive, the preconditioner, as shown in (34), is SPD. Since all the acceleration parameters are positive, the above assumption is actually satisfied and (34) is indeed SPD.

To sum up, the preconditioner induced from $\left(K^X + K^Y\right)^{-1}$ by a fixed number of ADI iterations with zero initial guess is symmetric positive-definite, and therefore, it can be combined with the conjugate gradient method.

5.2.3. *Number of the inner iteration steps*

Since $(K^X + K^Y)$ is just an approximation of $(\mathbb{K}^X + \mathbb{K}^Y)$, applying its inverse super-accurately does not benefit us much on reducing the number of outer iterations. Rather, only an approximate inverse is needed. Our experience is that a number of ADI iterations that can achieve a relative error around $1e$-3 is enough for most cases.

Given matrix $(K^X + K^Y)$, we go through the following procedure to decide the number of inner iterations: First, we estimate its smallest and largest generalized eigenvalues and set them as the lower bound α and upper bound β for the min-max problem (27), respectively; Then, we choose the number of inner iteration steps such that the upper bound of error reduction rate associated with the solution of (27) is smaller than $1e$-3.

5.3. *Partially incorporate the coefficients*

The preconditioner $\left(K^X + K^Y\right)^{-1}$ does not contain any information from the coefficients κ_{11} and κ_{22}. In this section, we propose a more sophisticated preconditioner that can partially incorporate these coefficients and meanwhile, can still be applied efficiently. This preconditioner possesses a similar structure to the generalized Kronecker product structure discussed in [19]. A similar idea has been applied to the mass matrix case, see [36].

5.3.1. *Approximations of $\mathbb{K}^X$ and $\mathbb{K}^Y$ that partially incorporate the coefficient information*

First, we consider a representative entry of $\mathbb{K}^X$:

$$\mathbb{K}^X_{\mu\nu} = \int_\Omega (B^y_i B^y_k)(B^x_{j,x} B^x_{\ell,x})\kappa_{11} d\Omega,$$

where $\mu = (i-1)N_x + j$, $\nu = (k-1)N_x + \ell$. If the basis function B^y_i only has local support, the integration region for $\mathbb{K}^X_{\mu\nu}$ on the y direction is restricted to the support region of B^y_i, which is a thin strip of the whole rectangle, as illustrated on the left of Figure 2.

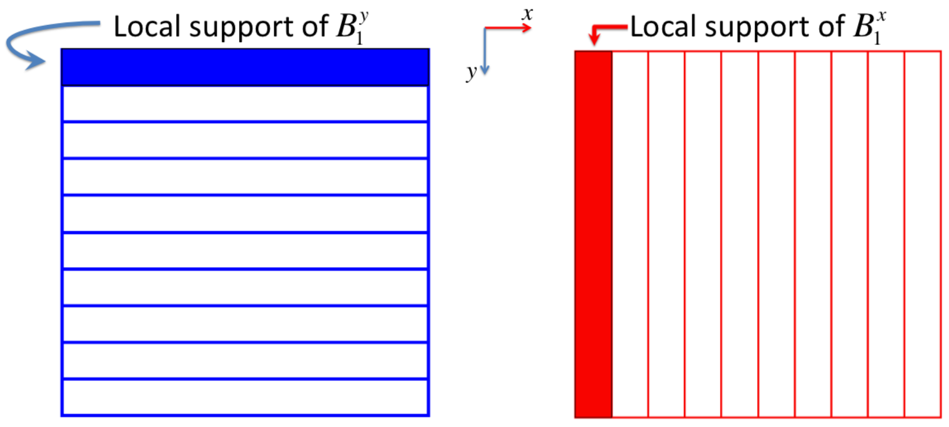


Figure 2: Local support property of the basis functions.

Therefore, this representative entry can be written as:

$$\mathbb{K}^X_{\mu\nu} = \int_\Omega (B^y_i B^y_k)(B^x_{j,x} B^x_{\ell,x})\kappa_{11}^{y|i} d\Omega,$$

where $\kappa_{11}^{y|i}$ stands for the restriction of the coefficient κ_{11} to the ith horizontal thin strip.

It is plausible to assume that $\kappa_{11}^{y|i}$ does not have much variation on the y direction, simply due to the short width of the support region. Under this assumption, we approximate $\mathbb{K}^X_{\mu\nu}$ as:

$$\mathbb{K}^X_{\mu\nu} \approx \left(\int_y B_i^y B_k^y dy\right)\left(\frac{1}{L_i^y}\int_\Omega B^x_{j,x}B^x_{\ell,x}\kappa_{11}^{y|i} d\Omega\right),$$

where L_i^y stands for the length of the support region for basis function B_i^y. By introducing the following notation:

$$K_i^x = \frac{1}{L_i^y}\int_\Omega B^x_{,x}(B^x_{,x})^T \kappa_{11}^{y|i} d\Omega, \tag{38}$$

for $i = 1, \cdots, N_y$, we can write down the corresponding approximation of $\mathbb{K}^X$ as:

$$\mathbb{K}^X \approx \mathcal{K}^X = \left(I_1^y M^y\right) \otimes K_1^x + \cdots + \left(I_{N_y}^y M^y\right) \otimes K_{N_y}^x, \tag{39}$$

where I_i^y denotes the $N_y \times N_y$ matrix with value 1 at its ith diagonal entry and value 0 at every other entry. Each K_i^x contains partial information of κ_{11}.

Moreover, the following derivation holds:

$$\begin{aligned}
\mathcal{K}^X &= \left(I_1^y M^y\right) \otimes \left(K_1^x I^x\right) + \cdots + \left(I_{N_y}^y M^y\right) \otimes \left(K_{N_y}^x I^x\right) \\
&= \left(I_1^y \otimes K_1^x\right)\left(M^y \otimes I^x\right) + \cdots + \left(I_{N_y}^y \otimes K_{N_y}^x\right)\left(M^y \otimes I^x\right) \\
&= \left(I_1^y \otimes K_1^x + \cdots + I_{N_y}^y \otimes K_{N_y}^x\right)\left(M^y \otimes I^x\right) \\
&= \begin{bmatrix} K_1^x & & \\ & \ddots & \\ & & K_{N_y}^x \end{bmatrix}\begin{bmatrix} & & \\ & M^y \otimes I^x & \\ & & \end{bmatrix}.
\end{aligned} \tag{40}$$

According to (40), $\mathcal{K}^X$ can be written as the product of a block diagonal matrix and a Kronecker product matrix. Therefore, an efficient algorithm can be developed to apply its inverse, as presented in Algorithm 3 and illustrated in Figure 3.

Algorithm 3.

```
1:  $B = Mat(b, N_x, N_y)$;
2:  $T = zeros(N_x, N_y)$;
3:  $X = zeros(N_x, N_y)$;
4:  for $j = 1 : N_y$ do
5:      $T(:, j) = K_j^x \backslash B(:, j)$;
6:  end for
7:  for $i = 1 : N_x$ do
8:      $X(i, :)^T = M^y \backslash T(i, :)^T$;
9:  end for
10: $x = Vec(X)$;
```

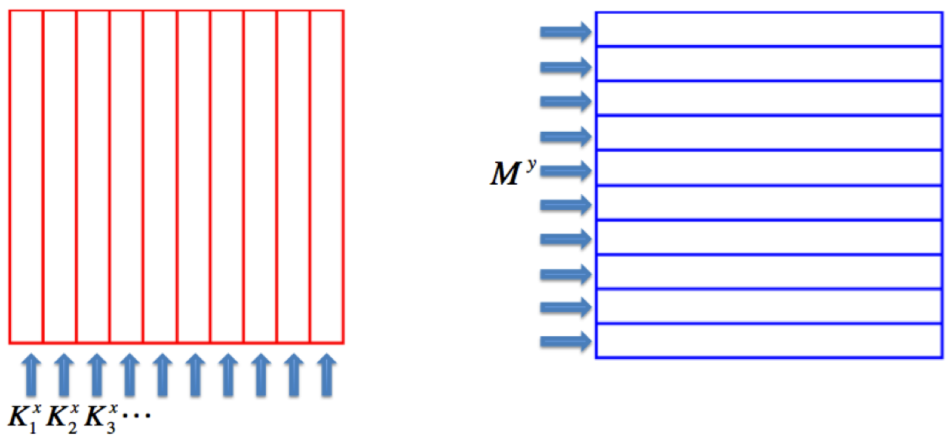


Figure 3: Visualization for Algorithm 3.

Similarly, given a representative entry of $\mathbb{K}^Y$:

$$\mathbb{K}^Y_{\mu\nu} = \int_\Omega (B^y_{i,y} B^y_{k,y})(B^x_j B^x_\ell)\kappa_{22} d\Omega,$$

where $\mu = (i-1)N_x + j$, $\nu = (k-1)N_x + \ell$, it can also be written as

$$\mathbb{K}^Y_{\mu\nu} = \int_\Omega (B^y_{i,y} B^y_{k,y})(B^x_j B^x_\ell)\kappa^{x|\ell}_{22} d\Omega$$

by restricting the integration domain to the local support of basis function B^x_ℓ, as illustrated on the right of Figure 2. We further approximate $\mathbb{K}^Y_{\mu\nu}$ as:

$$\mathbb{K}^Y_{\mu\nu} \approx \left(\frac{1}{L^x_\ell} \int_\Omega B^y_{i,y} B^y_{k,y} \kappa^{x|\ell}_{22} d\Omega \right) \left(\int_x B^x_j B^x_\ell dx \right),$$

where L^x_ℓ is the length of the support region for basis function B^x_ℓ. With the following notation:

$$K^y_\ell = \frac{1}{L^x_\ell} \int_\Omega B^y_{,y} (B^y_{,y})^T \kappa^{x|\ell}_{22} d\Omega, \tag{41}$$

for $\ell = 1, \cdots, N_x$, we have the corresponding approximation for $\mathbb{K}^Y$:

$$\mathbb{K}^Y \approx \mathcal{K}^Y = K^y_1 \otimes \left(M^x I^x_1\right) + \cdots + K^y_{N_x} \otimes \left(M^x I^x_{N_x}\right), \tag{42}$$

where I^x_ℓ denotes the $N_x \times N_x$ matrix with value 1 at its ℓth diagonal entry and value 0 at any other entry. Each K^y_ℓ contains partial information of κ_{22}.

Moreover, the following derivations hold:

$$\begin{aligned}
\mathcal{K}^Y &= \left(I^y K^y_1\right) \otimes \left(M^x I^x_1\right) + \cdots + \left(I^y K^y_{N_x}\right) \otimes \left(M^x I^x_{N_x}\right) \\
&= \left(I^y \otimes M^x\right)\left(K^y_1 \otimes I^x_1\right) + \cdots + \left(I^y \otimes M^x\right)\left(K^y_{N_x} \otimes I^x_{N_x}\right) \\
&= \left(I^y \otimes M^x\right)\left(K^y_1 \otimes I^x_1 + \cdots + K^y_{N_x} \otimes I^x_{N_x}\right).
\end{aligned}$$

An efficient algorithm for applying the inverse of $\mathcal{K}^Y$ is presented in Algorithm 4 and illustrated in Figure 4. Specifically, how to apply the inverse of $\left(I^y \otimes M^x\right)$ is simple due to its Kronecker product structure, as demonstrated in lines 4-6 of Algorithm 4. However, how to apply the inverse of $\left(K_1^y \otimes I_1^x + \cdots + K_{N_x}^y \otimes I_{N_x}^x\right)$ is less straightforward. First, we notice that

$$\begin{aligned}
&\left(K_1^y \otimes I_1^x + \cdots + K_{N_x}^y \otimes I_{N_x}^x\right)\left((K_1^y)^{-1} \otimes I_1^x + \cdots + (K_{N_x}^y)^{-1} \otimes I_{N_x}^x\right)\\
=&\sum_{j=1}^{N_x}\sum_{\ell=1}^{N_x}\left(K_j^y \otimes I_j^x\right)\left((K_\ell^y)^{-1} \otimes I_\ell^x\right)\\
=&\sum_{j=1}^{N_x}\sum_{\ell=1}^{N_x}\left(K_j^y (K_\ell^y)^{-1}\right) \otimes \left(I_j^x I_\ell^x\right)\\
=&\sum_{\ell=1}^{N_x} I^y \otimes I_\ell^x\\
=&I^y \otimes I^x.
\end{aligned}$$

Therefore, applying the inverse of

$$\left(K_1^y \otimes I_1^x + \cdots + K_{N_x}^y \otimes I_{N_x}^x\right)$$

is equivalent to applying

$$\left((K_1^y)^{-1} \otimes I_1^x + \cdots + (K_{N_x}^y)^{-1} \otimes I_{N_x}^x\right). \tag{43}$$

We can use Algorithm 2 to apply each term in (43) efficiently. However, for their sum, we do not have to go through Algorithm 2 to apply each one of these terms individually. Instead, due to the special forms of matrices I_ℓ^x, we can apply (43) in a collective manner, as shown in lines 7-9 of Algorithm 4.

Algorithm 4.

```
1:  $B = Mat(b, N_x, N_y)$;
2:  $T = zeros(N_x, N_y)$;
3:  $X = zeros(N_x, N_y)$;
4:  for $j = 1 : N_y$ do
5:      $T(:, j) = M^x \backslash B(:, j)$;
6:  end for
7:  for $i = 1 : N_x$ do
8:      $X(i, :)^T = K_i^y \backslash T(i, :)^T$;
9:  end for
10: $x = Vec(X)$;
```

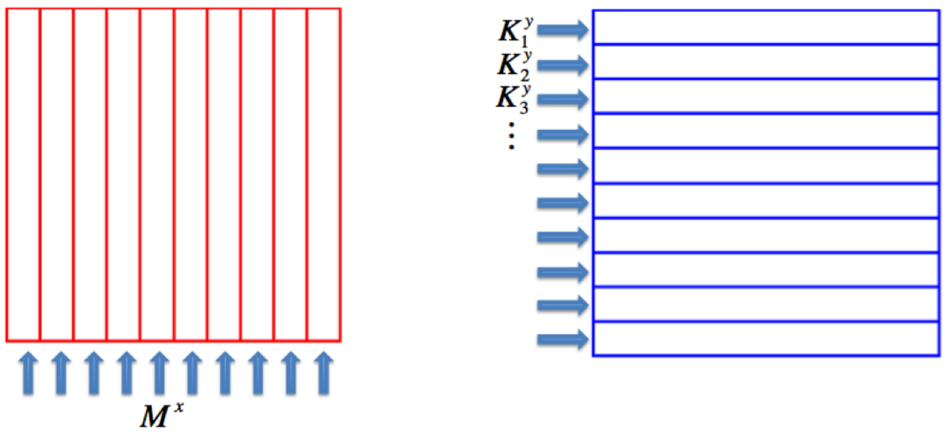


Figure 4: Visualization for Algorithm 4.

5.3.2. Symmetrization

We have constructed approximations for $\mathbb{K}^X$ and $\mathbb{K}^Y$, i.e., $\mathcal{K}^X$ and $\mathcal{K}^Y$, respectively, and presented efficient algorithms to apply their inverses. It is natural to combine them with the ADI algorithm to precondition linear system (29):

$$\left(\mathbb{K}^X + \mathbb{K}^Y\right) b = \mathcal{F}.$$

An inner iteration preconditioner for $\left(\mathbb{K}^X + \mathbb{K}^Y\right)$ can be constructed by applying the ADI algorithm on its approximation: $\left(\mathcal{K}^X + \mathcal{K}^Y\right)$. However, due to the asymmetry of $\left(\mathcal{K}^X + \mathcal{K}^Y\right)$, the induced preconditioner is also asymmetric and therefore, cannot be combined with the conjugate gradient method (CG). We need to use some other Krylov subspace method, for instance, the generalized minimal residual method (GMRES), or to symmetrize the preconditioner. We choose the second path due to the efficiency of CG on symmetric positive-definite matrices.

It takes two steps to symmetrize the preconditioner. First, we need to obtain a symmetric approximation of $\left(\mathcal{K}^X + \mathcal{K}^Y\right)$. To achieve this goal, we resort to the Cholesky factorization. We use L^y to denote the lower Cholesky factor of M^y, i.e., $M^y = L^y\left(L^y\right)^T$, and L^x to denote the lower Cholesky factor of M^x, i.e., $M^x = L^x\left(L^x\right)^T$. According to the Kronecker product properties, we have:

$$\begin{aligned} M^y \otimes I^x &= \left(L^y (L^y)^T\right) \otimes \left(I^x I^x\right) = \left(L^y \otimes I^x\right)\left((L^y)^T \otimes I^x\right); \\ I^y \otimes M^x &= \left(I^y I^y\right) \otimes \left(L^x (L^x)^T\right) = \left(I^y \otimes L^x\right)\left(I^y \otimes (L^x)^T\right). \end{aligned}$$

We define $\mathcal{S}^X$ and $\mathcal{S}^Y$ as follows:

$$\begin{aligned} \mathcal{S}^X &= \left(L^y \otimes I^x\right)\left(I_1^y \otimes K_1^x + \cdots + I_{N_y}^y \otimes K_{N_y}^x\right)\left((L^y)^T \otimes I^x\right); \\ \mathcal{S}^Y &= \left(I^y \otimes L^x\right)\left(K_1^y \otimes I_1^x + \cdots + K_{N_x}^y \otimes I_{N_x}^x\right)\left(I^y \otimes (L^x)^T\right). \end{aligned}$$

It is easy to verify that $\mathcal{S}^X$ and $\mathcal{S}^Y$ are both symmetric. We simply use $\mathcal{S}^X$ and $\mathcal{S}^Y$ to approximate $\mathcal{K}^X$ and $\mathcal{K}^Y$, respectively.

Second, we need to ensure that the preconditioner induced from the symmetrized approximation $\left(\mathcal{S}^X + \mathcal{S}^Y\right)$ by the ADI iterations maintains the symmetry. For this purpose, after selecting a set of acceleration parameters, we apply the cycle of ADI iterations twice, with a reversed order of parameters in the second time. The proof of symmetry for the aforementioned preconditioner can be found in Appendix A, along with the proof of its positive definiteness.

Besides preserving the symmetry, the selection of acceleration parameters also becomes a question since the condition in Theorem 1 that $\mathcal{S}^X$ and $\mathcal{S}^Y$ share the same set of generalized eigenvectors is no longer satisfied. In addition, the estimation of the smallest and largest generalized eigenvalues also becomes unclear.

However, we only need to apply $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$ approximately. In this case, we estimate the smallest and largest generalized eigenvalues of K^X and K^Y, scaled by the magnitude of κ_{11} and κ_{22}, respectively, and then choose the lower bound α and upper bound β for min-max problem (27) accordingly. The corresponding solution of (27) is selected as the set of acceleration parameters.

6. Numerical examples

In this section, we show some numerical results to demonstrate the performance of the preconditioners described in Sections 5.2 and 5.3. Specifically, we apply the compound iterations to linear system (29):

$$(\mathbb{K}^X + \mathbb{K}^Y)b = \mathcal{F},$$

discretized from partial differential equation (28), defined on a unit square with full Dirichlet boundary conditions, for different orthotropic coefficients κ_{11} and κ_{22}, as illustrated in Figures 5-7. Exact formulae for these coefficients can be found in Appendix B.

For all the results shown in this section, uniform meshes are used. Moreover, the conjugate gradient (CG) method is applied as the outer iteration and forced to stop whenever the relative residual (in l_2 norm) is smaller than $1e$-7 or 200 iterations have been performed. When the iterative process exceeds 200 iterations without converging, we use '—' to denote the corresponding result.

6.1. Highly orthotropic coefficients

In this example, we test the numerical performances corresponding to the proposed preconditioners with highly orthotropic coefficients, where κ_{11} is of order 1 while κ_{22} is of order 10^5, as shown in Figure 5.

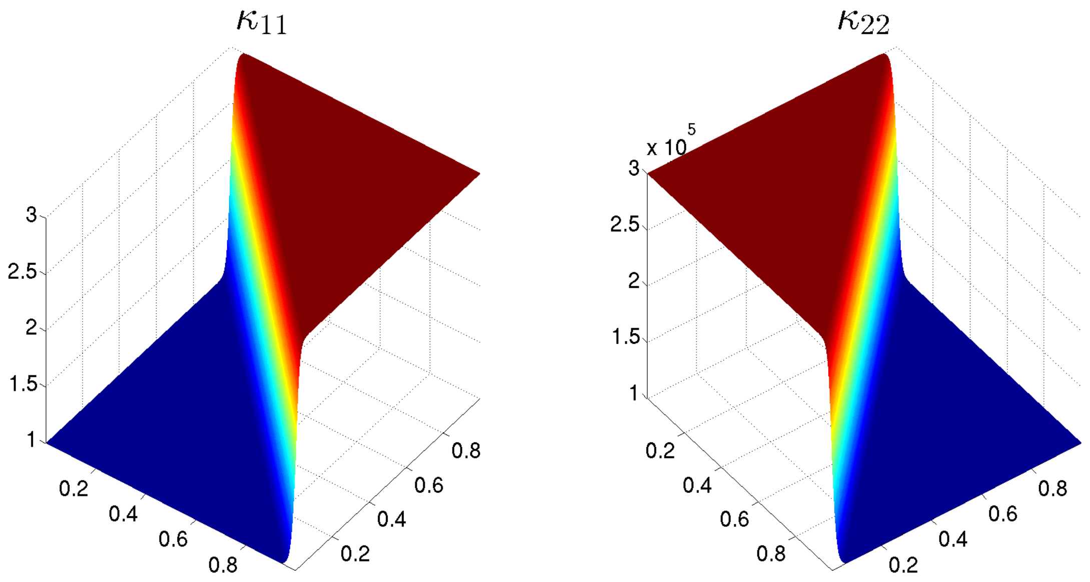


Figure 5: Highly orthotropic coefficients.

Table 2 records the number of CG iterations required for convergence with respect to different mesh sizes, while the number of inner iterations is fixed at 64. The two inner iteration preconditioners, induced from $\left(K^X + K^Y\right)$ and $\left(\mathcal{S}^X + \mathcal{S}^Y\right)$, respectively, are tested. The incomplete Cholesky (IC) preconditioner with zero fill-in is also presented for comparison.

N_{1D}	32	64	128	256	512	1024
$\left(K^X + K^Y\right)^{-1}$	14	15	15	16	16	16
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	6	6	5	5	4	3
IC	4	4	4	4	5	9

Table 2: Number of CG iterations vs. mesh size; $N_{inner} = 64$.

From Table 2, we can see that all three preconditioners can handle these highly orthotropic coefficients very well, in the sense that the number of CG iterations is small and not growing drastically as the mesh gets refined. This is confirmed by the solving time corresponding to these preconditioners, recorded in Table 3. These data of elapsed time, as well as those appeared later, are recorded with the MATLAB routines '*tic*' and '*toc*'.

N_{1D}	32	64	128	256	512	1024
$\left(K^X + K^Y\right)^{-1}$	1.8E-01	5.3E-01	1.5E+00	6.2E+00	3.4E+01	1.6E+02
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	1.3E+00	2.9E+00	6.1E+00	1.9E+01	5.3E+01	1.6E+02
IC	1.7E-03	3.3E-03	1.0E-02	4.5E-02	2.0E-01	1.3E+00

Table 3: Solving time vs. mesh size; $N_{inner} = 64$.

From Table 3, we observe that the solving time corresponding to the IC preconditioner is much smaller than those corresponding to $\left(K^X + K^Y\right)^{-1}$ and $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$. This is also the case for the setup time, as recorded in Table 4.

N_{1D}	32	64	128	256	512	1024
$\left(K^X + K^Y\right)^{-1}$	2.8E-03	5.2E-03	1.0E-02	2.1E-02	4.6E-02	8.2E-02
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	1.0E-02	2.5E-02	8.6E-02	3.2E-01	1.4E+00	5.6E+00
IC	1.4E-04	5.5E-04	3.8E-03	2.3E-02	8.1E-02	2.7E-01

Table 4: Setup time vs. mesh size; $N_{inner} = 64$.

For this example, where the coefficients are only mildly heterogeneous, the IC preconditioner is favorable against $\left(K^X + K^Y\right)^{-1}$ and $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$ in terms of the time consumed, at least for mesh size up to $h = 1/1024$.

In Table 5, we also record the numbers of CG iterations required for convergence with respect to different numbers of inner iterations, denoted as N_{inner}, while the mesh size is fixed at $1/256$. The last row, tagged as 'bound', record the upper bounds of the relative error associated with the selected acceleration parameters for linear system $(K^X + K^Y)b = \mathcal{F}$.

We observe that the number of CG iterations stops decreasing when the number of inner iterations is sufficiently large, which validates the conjecture we made before that the preconditioners $\left(K^X + K^Y\right)^{-1}$ and $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$ only need to be applied approximately.

N_{inner}	2	4	8	16	32	64
$\left(K^X + K^Y\right)^{-1}$	146	35	19	16	16	16
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	25	11	7	5	5	5
Upper bound	8.45E-01	4.20E-01	8.87E-02	3.94E-03	7.74E-06	3.00E-11

Table 5: Number of CG iterations vs. number of inner iterations; $N_{1D} = 256$.

6.2. Sinusoidal coefficients

In this example, we test the numerical performance corresponding to the proposed preconditioners with coefficients that are sinusoidal functions, as shown in Figure 6. The incomplete Cholesky (IC) preconditioner with zero fill-in is also presented for comparison.

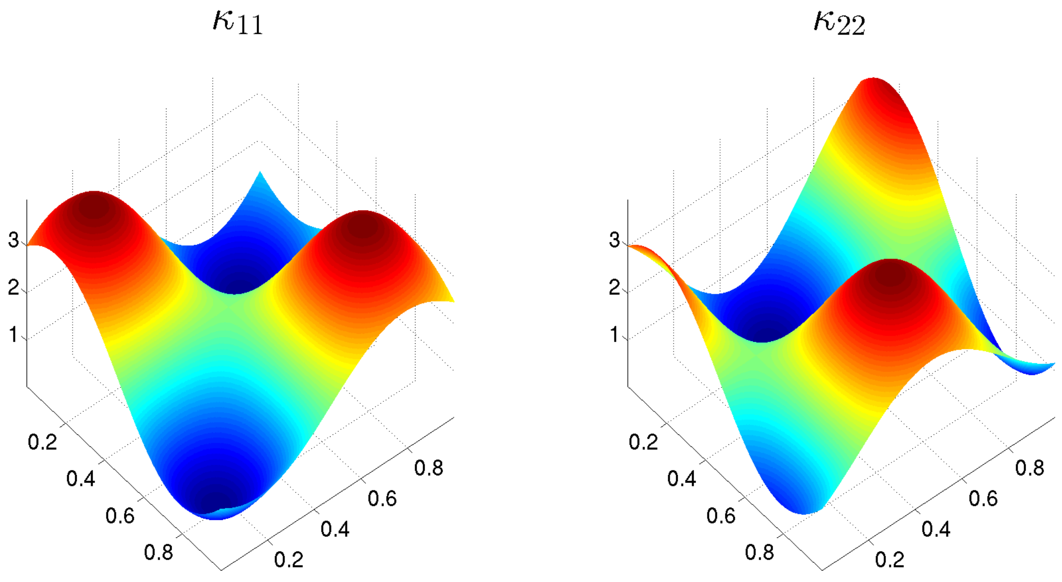


Figure 6: Sinusoidal coefficients.

Table 6 records the numbers of CG iterations required for convergence with respect to mesh size, while the number of inner iterations is fixed at 64. From Table 6, we observe that preconditioner $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$ works well in terms of the number of CG iterations. On the other hand, the preconditioner $\left(K^X + K^Y\right)^{-1}$ and the IC preconditioner are not satisfactory due to the increase in the numbers of CG iterations as the mesh gets refined. For mesh size $h = 1/512$, the IC preconditioner already exceeds the prescribed maximum 200 iterations without converging.

N_{1D}	32	64	128	256	512	1024
$\left(K^X + K^Y\right)^{-1}$	40	58	76	88	94	96
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	5	5	4	4	3	3
IC	22	45	90	186	—	—

Table 6: Number of CG iterations vs. mesh size; $N_{inner} = 64$.

Moreover, for preconditioner $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$, we even see a slight decrease of the number of CG iterations as we refine the mesh. This is due to an improved approximability of the preconditioner as it is constructed based on the local support property of the basis functions. In other words, the finer the mesh size is, the better the preconditioner $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$ incorporates the coefficient information.

The advantage of preconditioner $\left(\mathcal{S}^X+\mathcal{S}^Y\right)^{-1}$ against $\left(K^X+K^Y\right)^{-1}$ can also be observed in Table 7, where the solving time is recorded. Meanwhile, the setup time is recorded in Table 8. Although the setup time of $\left(K^X + K^Y\right)^{-1}$ is much smaller than of $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$, it is almost negligible when compared with the solving time. Therefore, $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$ is still favorable against $\left(K^X + K^Y\right)^{-1}$. Although the time consumed by the IC preconditioner is still the least among the three preconditioners when the iteration converges, we expect it to increase dramatically as we refine the mesh, due to the increase of iteration steps needed for convergence.

N_{1D}	32	64	128	256	512	1024
$\left(K^X + K^Y\right)^{-1}$	1.1E+00	3.6E+00	1.1E+01	3.6E+01	2.0E+02	9.3E+02
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	1.9E+00	3.7E+00	4.9E+00	1.5E+01	4.0E+01	1.6E+02
IC	9.4E-03	3.7E-02	1.7E-01	1.6E+00	—	—

Table 7: Solving time vs. mesh size; $N_{inner} = 64$.

N_{1D}	32	64	128	256	512	1024
$\left(K^X + K^Y\right)^{-1}$	4.5E-03	1.2E-02	2.0E-02	2.1E-02	4.6E-02	8.2E-02
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	1.4E-02	5.3E-02	1.6E-01	3.3E-01	1.3E+00	5.4E+00
IC	2.2E-04	8.2E-04	3.3E-03	2.3E-02	—	—

Table 8: Setup time vs. mesh size; $N_{inner} = 64$.

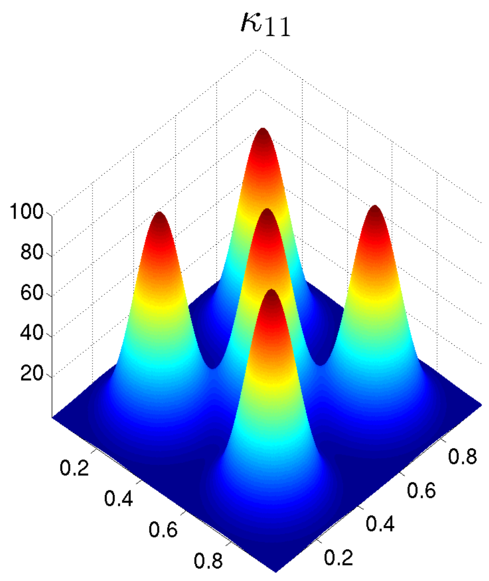
In Table 9, where the numbers of CG iterations with respect to different numbers of inner iterations are recorded for $h = 1/256$, we observe again that the number of CG iterations is not further reduced after the number of inner iterations is large enough.

N_{inner}	2	4	8	16	32	64
$\left(K^X + K^Y\right)^{-1}$	112	93	87	87	87	87
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	23	9	5	4	4	4
Upper bound	8.45E-01	4.20E-01	8.87E-02	3.94E-03	7.74E-06	3.00E-11

Table 9: Number of CG iterations vs. number of inner iterations; $N_{1D} = 256$.

6.3. Gaussian spikes

In this example, we test the numerical performances corresponding to the proposed preconditioners with coefficients that are superpositions of Gaussian functions, as shown in Figure 7. The incomplete Cholesky (IC) preconditioner is also presented for comparison.



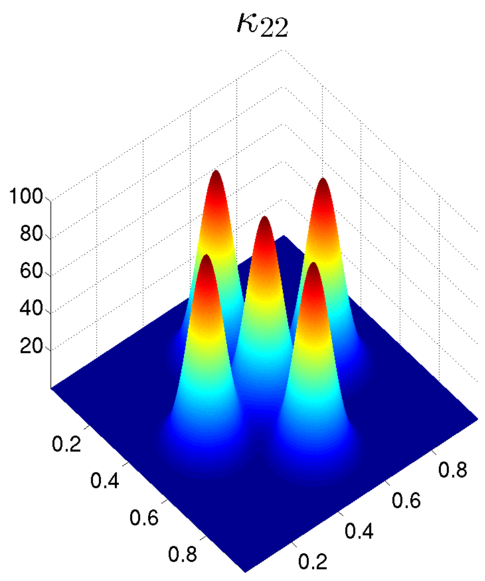


Figure 7: Gaussian spikes.

Table 10 records the numbers of CG iterations required for convergence with respect to mesh size, while the number of inner iterations is fixed at 64. Table 10 records the numbers of CG iterations with respect to different numbers of inner iterations, while the mesh size is fixed at $1/256$.

For these coefficients that are much more complicated than those tested in sections 6.1 and 6.2, the preconditioner $\left(K^X + K^Y\right)^{-1}$ simply does not work and the IC preconditioner only manages to converge within 200 iterations for mesh size up to $h = 1/128$. On the other hand, the preconditioner $\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$ still manages to deliver satisfactory results, given that the number of inner iterations is sufficiently large.

N_{1D}	32	64	128	256	512	1024
$\left(K^X + K^Y\right)^{-1}$	—	—	—	—	—	—
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	16	10	9	9	10	10
IC	33	60	119	—	—	—

Table 10: Number of CG iterations vs. mesh size; $N_{inner} = 64$.

N_{inner}	2	4	8	16	32	64
$\left(K^X + K^Y\right)^{-1}$	—	—	—	—	—	—
$\left(\mathcal{S}^X + \mathcal{S}^Y\right)^{-1}$	—	—	—	28	13	10
Upper bound	8.45E-01	4.20E-01	8.87E-02	3.94E-03	7.74E-06	3.00E-11

Table 11: Number of CG iterations vs. number of inner iterations; $N_{1D} = 256$.

7. Conclusions

The alternating direction implicit algorithm can be combined with the Kronecker product properties to efficiently solve the linear system discretized from the Poisson equation, i.e., the model problem, by the finite element method assuming tensor product basis functions. However, its application to more general problems is limited by its stringent requirements. In this paper, we adopt the compound iteration idea and use the alternating direction implicit algorithm as the inner iterations, serving as the preconditioner for the outer iterations, i.e., the conjugate gradient iterations. By exploiting the local support property of the basis functions, this inner iteration preconditioner can partially incorporate the coefficient information and therefore, obtain good approximability. These compound iterations are applied to steady-state diffusion equations with orthotropic heterogeneous coefficients, achieving satisfactory numerical performances with low setup cost.

Appendix A.

In this appendix, we show that the preconditioner induced from $(\mathcal{S}^X+\mathcal{S}^Y)^{-1}$ by the ADI iterations with a zero initial guess is symmetric positive-definite (SPD).

With a set of acceleration parameters $\{r^{(j)}\}_{j=0}^{k}$, $r^{(j)}>0$, we use the following iterative scheme to solve $(\mathcal{S}^X+\mathcal{S}^Y)b=\mathcal{F}$ approximately.

First, a forward cycle is applied:

$$\begin{aligned}
(r^{(0)}M+\mathcal{S}^X)b^{(0+\frac{1}{2})} &= (r^{(0)}M-\mathcal{S}^Y)b^{(0)}+\mathcal{F},\\
(r^{(0)}M+\mathcal{S}^Y)b^{(0+1)} &= (r^{(0)}M-\mathcal{S}^X)b^{(0+\frac{1}{2})}+\mathcal{F},\\
&\vdots\\
(r^{(k)}M+\mathcal{S}^X)b^{(k+\frac{1}{2})} &= (r^{(k)}M-\mathcal{S}^Y)b^{(k)}+\mathcal{F},\\
(r^{(k)}M+\mathcal{S}^Y)b^{(k+1)} &= (r^{(k)}M-\mathcal{S}^X)b^{(k+\frac{1}{2})}+\mathcal{F}.
\end{aligned}\tag{A.1}$$

Then, a backward cycle is applied:

$$\begin{aligned}
(r^{(k)}M+\mathcal{S}^Y)\tilde{b}^{(0+\frac{1}{2})} &= (r^{(k)}M-\mathcal{S}^X)b^{(k+1)}+\mathcal{F},\\
(r^{(k)}M+\mathcal{S}^X)\tilde{b}^{(0+1)} &= (r^{(k)}M-\mathcal{S}^Y)\tilde{b}^{(0+\frac{1}{2})}+\mathcal{F},\\
&\vdots\\
(r^{(0)}M+\mathcal{S}^Y)\tilde{b}^{(k+\frac{1}{2})} &= (r^{(0)}M-\mathcal{S}^X)\tilde{b}^{(k)}+\mathcal{F},\\
(r^{(0)}M+\mathcal{S}^X)\tilde{b}^{(k+1)} &= (r^{(0)}M-\mathcal{S}^Y)\tilde{b}^{(k+\frac{1}{2})}+\mathcal{F}.
\end{aligned}\tag{A.2}$$

With the following definitions:

$$\mathcal{P}^{(j)}=(r^{(j)}M+\mathcal{S}^Y)^{-1}(r^{(j)}M-\mathcal{S}^X)(r^{(j)}M+\mathcal{S}^X)^{-1}(r^{(j)}M-\mathcal{S}^Y);\tag{A.3a}$$

$$\tilde{\mathcal{P}}^{(j)}=(r^{(j)}M+\mathcal{S}^X)^{-1}(r^{(j)}M-\mathcal{S}^Y)(r^{(j)}M+\mathcal{S}^Y)^{-1}(r^{(j)}M-\mathcal{S}^X),\tag{A.3b}$$

we can write:

$$\tilde{e}^{(k+1)}=\tilde{\mathcal{P}}^{(0)}\dots\tilde{\mathcal{P}}^{(k)}\mathcal{P}^{(k)}\dots\mathcal{P}^{(0)}e^{(0)},\tag{A.4}$$

where $e^{(0)} = b^{(0)} - b$ and $\tilde{e}^{(k+1)} = \tilde{b}^{(k+1)} - b$.

Substituting the true solution $b = (\mathcal{S}^X + \mathcal{S}^Y)^{-1}\mathcal{F}$ into (A.4), we have:

$$\tilde{b}^{(k+1)} = (I - \tilde{\mathcal{P}}^{(0)} \cdots \tilde{\mathcal{P}}^{(k)}\mathcal{P}^{(k)} \cdots \mathcal{P}^{(0)})(\mathcal{S}^X + \mathcal{S}^Y)^{-1}\mathcal{F}, \tag{A.5}$$

given that zero initial guess is used, i.e., $b^{(0)} = 0$. From (A.5), it is clear that the induced preconditioner has the following matrix form:

$$(I - \tilde{\mathcal{P}}^{(0)} \cdots \tilde{\mathcal{P}}^{(k)}\mathcal{P}^{(k)} \cdots \mathcal{P}^{(0)})(\mathcal{S}^X + \mathcal{S}^Y)^{-1}. \tag{A.6}$$

Symmetry

To demonstrate the symmetry of the induced preconditioner, we only need to show that

$$\tilde{\mathcal{P}}^{(0)} \cdots \tilde{\mathcal{P}}^{(k)}\mathcal{P}^{(k)} \cdots \mathcal{P}^{(0)}(\mathcal{S}^X + \mathcal{S}^Y)^{-1} \tag{A.7}$$

is symmetric, since $(\mathcal{S}^X + \mathcal{S}^Y)^{-1}$ itself is symmetric. For this purpose, we first present the following lemma:

Lemma 2. *For $\mathcal{P}^{(j)}$ and $\tilde{\mathcal{P}}^{(j)}$ as defined in* (A.3)*, we have:*

$$\mathcal{P}^{(j)}(\mathcal{S}^X + \mathcal{S}^Y)^{-1} = (\mathcal{S}^X + \mathcal{S}^Y)^{-1}(\tilde{\mathcal{P}}^{(j)})^T. \tag{A.8}$$

Proof. On one hand, we have:

$$\begin{array}{rl} & (r^{(j)}M + \mathcal{S}^X)^{-1}(r^{(j)}M - \mathcal{S}^Y)(\mathcal{S}^X + \mathcal{S}^Y)^{-1} \\ = & (r^{(j)}M + \mathcal{S}^X)^{-1}(r^{(j)}M + \mathcal{S}^X - \mathcal{S}^X - \mathcal{S}^Y)(\mathcal{S}^X + \mathcal{S}^Y)^{-1} \\ = & (\mathcal{S}^X + \mathcal{S}^Y)^{-1} - (r^{(j)}M + \mathcal{S}^X)^{-1}; \end{array} \tag{A.9}$$

On the other hand, we have:

$$\begin{array}{rl} & (\mathcal{S}^X + \mathcal{S}^Y)^{-1}(r^{(j)}M - \mathcal{S}^Y)(r^{(j)}M + \mathcal{S}^X)^{-1} \\ = & (\mathcal{S}^X + \mathcal{S}^Y)^{-1}(r^{(j)}M + \mathcal{S}^X - \mathcal{S}^X - \mathcal{S}^Y)(r^{(j)}M + \mathcal{S}^X)^{-1} \\ = & (\mathcal{S}^X + \mathcal{S}^Y)^{-1} - (r^{(j)}M + \mathcal{S}^X)^{-1}. \end{array} \tag{A.10}$$

Therefore, the following relation holds:

$$(r^{(j)}M+\mathcal{S}^X)^{-1}(r^{(j)}M-\mathcal{S}^Y)(\mathcal{S}^X+\mathcal{S}^Y)^{-1} = (\mathcal{S}^X+\mathcal{S}^Y)^{-1}(r^{(j)}M-\mathcal{S}^Y)(r^{(j)}M+\mathcal{S}^X)^{-1}. \tag{A.11}$$

Similarly, we also have:

$$(r^{(j)}M+\mathcal{S}^Y)^{-1}(r^{(j)}M-\mathcal{S}^X)(\mathcal{S}^X+\mathcal{S}^Y)^{-1} = (\mathcal{S}^X+\mathcal{S}^Y)^{-1}(r^{(j)}M-\mathcal{S}^X)(r^{(j)}M+\mathcal{S}^Y)^{-1}. \tag{A.12}$$

Combining (A.11) and (A.12) with the definitions of $\mathcal{P}^{(j)}$ and $\tilde{\mathcal{P}}^{(j)}$, we readily have relation (A.8). □

By repeatedly applying Lemma 2, it is obvious that (A.7) is equivalent to:

$$\tilde{\mathcal{P}}^{(0)} \cdots \tilde{\mathcal{P}}^{(k)}(\mathcal{S}^X + \mathcal{S}^Y)^{-1}(\tilde{\mathcal{P}}^{(k)})^T \cdots (\tilde{\mathcal{P}}^{(0)})^T$$

and is indeed symmetric. Therefore, the induced preconditioner, with its matrix form shown in (A.6), is also symmetric.

Positive definiteness

To demonstrate the positive definiteness of the induced preconditioner, we need to show that

$$x^T(I - \tilde{\mathcal{P}}^{(0)} \cdots \tilde{\mathcal{P}}^{(k)}\mathcal{P}^{(k)} \cdots \mathcal{P}^{(0)})(\mathcal{S}^X + \mathcal{S}^Y)^{-1}x > 0, \quad \forall x \neq 0. \tag{A.13}$$

Defining $y = (\tilde{\mathcal{P}}^{(k)})^T \cdots (\tilde{\mathcal{P}}^{(0)})^T x$ and $\mathcal{S}^{-1} = (\mathcal{S}^X + \mathcal{S}^Y)^{-1}$, (A.13) is equivalent to:

$$\|x\|_{\mathcal{S}^{-1}} > \|y\|_{\mathcal{S}^{-1}}, \tag{A.14}$$

where $\|x\|_{\mathcal{S}^{-1}}$ is defined as $\|x\|_{\mathcal{S}^{-1}} = x^T\mathcal{S}^{-1}x$.

First, let us examine the matrix

$$(\tilde{\mathcal{P}}^{(j)})^T = (r^{(j)}M - \mathcal{S}^X)(r^{(j)}M + \mathcal{S}^Y)^{-1}(r^{(j)}M - \mathcal{S}^Y)(r^{(j)}M + \mathcal{S}^X)^{-1}.$$

Due to the matrix similarity, we have:

$$\rho\big((\tilde{\mathcal{P}}^{(j)})^T\big) = \rho\big((r^{(j)}M + \mathcal{S}^X)^{-1}(r^{(j)}M - \mathcal{S}^X)(r^{(j)}M + \mathcal{S}^Y)^{-1}(r^{(j)}M - \mathcal{S}^Y)\big),$$

where $\rho(A)$ stands for the spectral radius of the square matrix A. Moreover, we have

$$\begin{aligned} & \rho\big((r^{(j)}M + \mathcal{S}^X)^{-1}(r^{(j)}M - \mathcal{S}^X)(r^{(j)}M + \mathcal{S}^Y)^{-1}(r^{(j)}M - \mathcal{S}^Y)\big) \\ = \; & \|(r^{(j)}M + \mathcal{S}^X)^{-1}(r^{(j)}M - \mathcal{S}^X)(r^{(j)}M + \mathcal{S}^Y)^{-1}(r^{(j)}M - \mathcal{S}^Y)\|_2 \\ \leq \; & \|(r^{(j)}M + \mathcal{S}^X)^{-1}(r^{(j)}M - \mathcal{S}^X)\|_2 \cdot \|(r^{(j)}M + \mathcal{S}^Y)^{-1}(r^{(j)}M - \mathcal{S}^Y)\|_2. \end{aligned}$$

Suppose $\{\lambda_i^X\}_{i=1}^N$ are all the generalized eigenvalues of $\mathcal{S}^X$ and M, satisfying:

$$\mathcal{S}^X v_i^X = \lambda_i^X M v_i^X.$$

We have the following relations:

$$\begin{aligned} (r^{(j)}M - \mathcal{S}^X)v_i^X &= (r^{(j)} - \lambda_i^X)Mv_i^X; \\ (r^{(j)}M + \mathcal{S}^X)^{-1}Mv_i^X &= \frac{1}{r^{(j)} + \lambda_i^X} v_i^X. \end{aligned}$$

Therefore,

$$\|(r^{(j)}M + \mathcal{S}^X)^{-1}(r^{(j)}M - \mathcal{S}^X)\|_2 = \max_{i=1,\ldots,N} \left| \frac{r^{(j)} - \lambda_i^X}{r^{(j)} + \lambda_i^X} \right| \leq 1. \tag{A.15}$$

Similarly, we have:

$$\|(r^{(j)}M + \mathcal{S}^Y)^{-1}(r^{(j)}M - \mathcal{S}^Y)\|_2 = \max_{i=1,\ldots,N} \left| \frac{r^{(j)} - \lambda_i^Y}{r^{(j)} + \lambda_i^Y} \right| \leq 1. \tag{A.16}$$

At least one of $\mathcal{S}^X$ and $\mathcal{S}^Y$ should be positive-definite if the Poisson equation they correspond to is accompanied with Dirichlet boundary condition on part of the boundary. Thus, at least one of the symbols '$\leq$' in (A.15) and (A.16) can be replaced with '$<$', which leads us to:

$$\rho\big((\tilde{\mathcal{P}}^{(j)})^T\big) < 1.$$

Therefore, the absolute value of all the eigenvalues of $(\tilde{\mathcal{P}}^{(j)})^T$ is less than 1.

In other words, if we define $z = (\tilde{\mathcal{P}}^{(0)})^T x$, z is 'shorter' than x from all 'angles'. Since $\|\cdot\|_{\mathcal{S}^{-1}}$ is just a particular angle to measure the length of z and x, we have $\|x\|_{\mathcal{S}^{-1}} > \|z\|_{\mathcal{S}^{-1}}$. By repeatedly applying the same argument, we arrive at (A.14) and therefore, the induced preconditioner is indeed positive-definite.

Appendix B.

In this appendix, formulae for coefficients of the numerical examples shown in section 6 are presented.

Figure 5 (Highly orthotropic coefficients):

$$\begin{aligned}\kappa_{11} &= 10^0\left(2+\frac{e^{100(x+y-1)-1}}{e^{100(x+y-1)+1}}\right);\\ \kappa_{22} &= 10^5\left(2+\frac{e^{100(1-x-y)-1}}{e^{100(1-x-y)+1}}\right).\end{aligned}$$

Figure 6 (Sinusoidal coefficients):

$$\begin{aligned}\kappa_{11} &= \Big(1+0.99\cos\big(5(x-y)\big)\Big)+\Big(1+0.99\sin\big(5(x+y)\big)\Big);\\ \kappa_{22} &= \Big(1+0.99\sin\big(5(x-y)\big)\Big)+\Big(1+0.99\cos\big(5(x+y)\big)\Big).\end{aligned}$$

Figure 7 (Gaussian spikes):

$$\kappa_{11}=\sum_{i=1}^{5}Ae^{-\left(a(x-x_i)^2+c(y-y_i)^2\right)}$$

with $A=100, a=75, c=75$ and $(x_1,y_1)=(0.25,0.25)$; $(x_2,y_2)=(0.25,0.75)$; $(x_3,y_3)=(0.5,0.5)$; $(x_4,y_4)=(0.75,0.25)$; $(x_5,y_5)=(0.75,0.75)$.

$$\kappa_{22}=\sum_{i=1}^{5}Ae^{-\left(a(x-x_i)^2+c(y-y_i)^2\right)}$$

with $A=100, a=150, c=150$ and $(x_1,y_1)=(0.5,0.25)$; $(x_2,y_2)=(0.5,0.75)$; $(x_3,y_3)=(0.5,0.5)$; $(x_4,y_4)=(0.75,0.5)$; $(x_5,y_5)=(0.25,0.5)$.

References

[1] D. W. Peaceman, H. H. Rachford, Jr, The numerical solution of parabolic and elliptic differential equations, Journal of the Society for Industrial & Applied Mathematics 3 (1955) 28–41.

[2] J. Douglas, H. Rachford, On the numerical solution of heat conduction problems in two and three space variables, Transactions of the American mathematical Society 82 (1956) 421–439.

[3] E. L. Wachspress, G. Habetler, An alternating-direction-implicit iteration technique, Journal of the Society for Industrial & Applied Mathematics 8 (1960) 403–423.

[4] G. Birkhoff, R. S. Varga, D. Young, Alternating direction implicit methods, Advances in Computers 3 (1962) 189–273.

[5] P. Ciarlet, The finite element method for elliptic problems, volume 4, North Holland, 1978.

[6] D. Braess, Finite elements: Theory, fast solvers, and applications in solid mechanics, Cambridge University Press, 2001.

[7] T. J. R. Hughes, The Finite Element Method: linear static and dynamic finite element analysis, Dover Publications, 2000.

[8] W. Steeb, T. Shi, Matrix Calculus and Kronecker Product with Applications and C++ Programs, World Scientific, 1997.

[9] D. Turkington, Generalized Vectorization, Cross-Products, and Matrix Calculus, Generalized Vectorization, Cross-products, and Matrix Calculus, Cambridge University Press, 2013.

[10] C. F. Van Loan, The ubiquitous Kronecker product, Journal of Computational and Applied Mathematics 123 (2000) 85–100.

[11] W. R. Dyksen, Tensor product generalized ADI methods for separable elliptic problems, SIAM Journal on Numerical Analysis 24 (1987) 59–76.

[12] E. L. Wachspress, Extended application of alternating direction implicit iteration model problem theory, Journal of the Society for Industrial & Applied Mathematics 11 (1963) 994–1016.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes 3rd Edition: the art of scientific computing, Cambridge University Press, 2007.

[14] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, volume 37, Springer, 2006.

[15] C. de Boor, Efficient computer manipulation of tensor products, ACM Transactions on Mathematical Software (TOMS) 5 (1979) 173–182.

[16] V. Pereyra, G. Scherer, Efficient computer manipulation of tensor products with applications to multidimensional approximation, Mathematics of Computation 27 (1973) 595–605.

[17] P. E. Buis, W. R. Dyksen, Efficient vector and parallel manipulation of tensor products, ACM Transactions on Mathematical Software (TOMS) 22 (1996) 18–23.

[18] J. Brewer, Kronecker products and matrix calculus in system theory, Circuits and Systems, IEEE Transactions on 25 (1978) 772–781.

[19] P. A. Regalia, M. K. Sanjit, Kronecker products, unitary matrices and signal processing applications, SIAM review 31 (1989) 586–613.

[20] J. G. Nagy, M. E. Kilmer, Kronecker product approximation for preconditioning in three-dimensional imaging applications, Image Processing, IEEE Transactions on 15 (2006) 604–613.

[21] D. W. Fausett, C. T. Fulton, Large least squares problems involving Kronecker products, SIAM Journal on Matrix Analysis and Applications 15 (1994) 219–227.

[22] J. Kamm, J. G. Nagy, Kronecker product and SVD approximations in image restoration, Linear algebra and its applications 284 (1998) 177–192.

[23] M. Davio, Kronecker products and shuffle algebra, Computers, IEEE Transactions on 100 (1981) 116–125.

[24] C. D. Martin, Higher-order Kronecker Products and Tensor Decompositions, Ph.D. thesis, Cornell University, 2005.

[25] MATLAB, Version 7.10.0 (R2010a), The MathWorks Inc. Natick, Massachusetts (2010).

[26] E. L. Wachspress, Optimum alternating-direction-implicit iteration parameters for a model problem, Journal of the Society for Industrial & Applied Mathematics 10 (1962) 339–350.

[27] R. S. Varga, Matrix Iterative Analysis, volume 27, Springer, 2009.

[28] E. L. Wachspress, Iterative solution of elliptic systems and applications to the neutron diffusion equations of reactor physics, Prentice-Hall, 1966.

[29] D. Evans, Alternating direction implicit preconditioning methods for self-adjoint elliptic differential equations, Computers & Mathematics with Applications 7 (1981) 151–158.

[30] R. C. Chin, T. A. Manteuffel, J. de Pillis, ADI as a preconditioning for solving the convection-diffusion equation, SIAM journal on scientific and statistical computing 5 (1984) 281–299.

[31] M. Hochbruck, G. Starke, Preconditioned Krylov subspace methods for Lyapunov matrix equations, SIAM Journal on Matrix Analysis and Applications 16 (1995) 156–171.

[32] J. C. Miellou, P. Spitéri, Optimization of the relaxation parameter for SSOR and ADI preconditioning, Numerical Algorithms 29 (2002) 153–195.

[33] K. Jbilou, ADI preconditioned Krylov methods for large Lyapunov matrix equations, Linear Algebra and its Applications 432 (2010) 2473–2485.

[34] H. Jiang, Y. S. Wong, A parallel alternating direction implicit preconditioning method, Journal of computational and applied mathematics 36 (1991) 209–226.

[35] C. F. Van Loan, N. Pitsianis, Approximation with Kronecker products, Technical Report, Cornell University, 1992.

[36] —–, Fast isogeometric solvers for explicit dynamics, Computer Methods in Applied Mechanics and Engineering (submitted).