Kronecker Product Approximation Preconditioners for Convection-diffusion Model Problems

Laura Grigori, Hua Xiang

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Kronecker Product Approximation Preconditioners for Convection-diffusion Model Problems

Laura GRIGORI — Hua XIANG

N° 6536

Mai 2008

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6536--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Kronecker Product Approximation Preconditioners for Convection-diffusion Model Problems

Laura GRIGORI* , Hua XIANG †

Thème NUM — Systèmes numériques
Équipe-Projet Grand-large

Rapport de recherche n° 6536 — Mai 2008 — 25 pages

Abstract: We consider the preconditioning of linear systems arising from four convection-diffusion model problems: scalar convection-diffusion problem, Stokes problem, Oseen problem, and Navier-Stokes problem. For these problems we identify an explicit Kronecker product structure of the coefficient matrices, in particular for the convection term. For the latter three model cases, the coefficient matrices have a 2×2 block structure, where each block is a Kronecker product or a summation of several Kronecker products. We use these structures to design a diagonal block preconditioner, a tridiagonal block preconditioner and a constraint preconditioner. The constraint preconditioner can be regarded as the modification of the tridiagonal block preconditioner and of the diagonal block preconditioner based on the cell Reynolds number. For this reason, the constraint preconditioner is usually more efficient. We also give numerical examples to show the efficiency of these preconditioners.

Key-words: preconditioning, Kronecker product approximation, convection-diffusion problem

* INRIA Saclay-Ile de France, Bat 490, Universite Paris-Sud 11, 91405 Orsay France (laura.grigori@inria.fr).
† INRIA Saclay-Ile de France, Bat 490, Universite Paris-Sud 11, 91405 Orsay France (hua.xiang@inria.fr).

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Préconditionnements basés sur une approximation du produit de Kronecker pour les problèmes modèles de convection-diffusion

Résumé : Nous considérons le préconditionnement des systèmes linéaires provenant de quatre problèmes modèle de convection-diffusion: le problème scalaire de convection-diffusion, le problème de Stokes, le problème de Oseen, et le problème de Navier-Stokes. Nous identifions une structure de produit de Kronecker pour la matrice de coefficients, en particulier pour le terme de convection. Pour les trois derniers cas modèle, les matrices ont une structure de blocs 2×2, où chaque bloc est un produit de Kronecker ou une addition de quelques produits de Kronecker. Nous utilisons cette structure pour construire un préconditionneur bloc diagonal, un préconditionneur bloc tridiagonal et un préconditionneur à contraintes. Le préconditionneur à contraintes peut être considéré comme une modification des préconditionneurs bloc tridiagonal et bloc diagonal basé sur le nombre de Reynolds d'une cellule. Pour cette raison le préconditionneur à contraintes est plus efficace. Nous présentons des exemples numeriques qui montrent l'efficacité de ces préconditionneurs.

Mots-clés : préconditionnement, approximation du produit de Kronecker, problème de convection-diffusion

1 Introduction

In many applications, we need to solve a linear system $Ax = b$. When iterative methods are used, a preconditioner P is sought, such that $P^{-1}A$ has better spectral properties and the linear system $P^{-1}Ax = P^{-1}b$ is faster to converge. Even though there exist efficient preconditioners, such as ILU, SPAI, etc [3, 4, 16, 19, 28, 29], it is difficult to design a preconditioner that is efficient for any linear system. Usually an efficient preconditioner for one class of problem may do not work for another class. Hence a vrey used approach is to seek problem specific methods. For example, in stochastic automata networks (SANs) [20], the authors choose a Kronecker product preconditioner as $P = G \otimes F$.

The Kronecker product has the important property: $(G \otimes F)^{-1} = G^{-1} \otimes F^{-1}$, if the inversion exists [12]. When $G \otimes F$ is of order n^2, G and F can be just of order n. It means that the inversion of a big matrix can be obtained by the inversion of much smaller matrices. Obviously it has great advantage to only work on small matrices. Then the solution of $Px = b$ will be quite easy. For a matrix $Z = [z_1, \cdots, z_n] \in \mathbb{R}^{m\times n}$ with $z_i \in \mathbb{R}^{m\times 1}(i = 1, \cdots, n)$, we define $\mathsf{vec}(Z) \in \mathbb{R}^{mn\times 1}$ by $\mathsf{vec}(Z) = [z_1^T, \cdots, z_n^T]^T$, that is, the vec operator stacks the columns of a matrix one underneath the other. Then the matrix-vector product $P^{-1}z$ can be obtained by $P^{-1}z = \mathsf{vec}(F^{-1}ZG^{-T})$, where $\mathsf{vec}(Z) = z$.

In this paper we will consider four convection-diffusion model problems: scalar convection-diffusion problem, Stokes problem, Oseen problem and Navier-Stokes problem in a rectangle domain $\Omega = (0, 1) \times (0, 1)$. After finite element method (FEM) or finite difference method (FDM) discretization, we will arrive at solving a linear system, and this will involve Kronecker product structures. For the latter three cases, their coefficient matrices also have a well-known 2-by-2 block structure, i.e., the saddle point system in the following form.

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{1}$$

We can make full use of its block structure, and design diagonal block preconditioner, tridiagonal block preconditioner, or constraint preconditioner [6, 13, 14, 17, 18, 30, 26]. In our model problems, we will find more detailed structure in each block. In fact, both the convection term and the diffusion term are a summation of a series of Kronecker products, and each block in the coefficient of (1) can be expressed by Kronecker products. To our knowledge, it is the first time that the whole matrix of Stokes problem, Oseen problem, or Navier-Stokes problem is expressed by Kronecker products, though the Kronecker product structure for the diffusion term has appeared in [10]. In this paper, we study, in addition to the block structure in (1), the Kronecker product structure in each block, and use it to construct different preconditioners. We first use one Kronecker product to approximate the convection term and the diffusion term, and then approximate the Schur complement $S = -BA^{-1}B^T$. Ususally the Schur complement is not easy to approximate. In this paper we use one Kronecker product to approximate it.

The rest of this paper is arranged as follows. In section 2, we investigate the coefficient matrices arising from FEM or FDM discretization of the convection-diffusion problems, and give the explicit Kronecker product structures in the coefficient matrices. In section 3, we design the Kronecker product approximation preconditioners according to the structure of the coefficient matrices. It mainly based on using one Kronecker product approximation to approximate the summation of several Kronecker products. The numerical examples in section 4 show the efficiency of this kind of preconditioner. Conclusion remarks are given in section 5.

2 Coefficient Matrices

In this section, we use FEM or FDM to discrete the model problems in the domain $\Omega = (0,1)\times(0,1)$ with uniform grids, and obtain the coefficient matrices of these convection-diffusion model problems. We will find Kronecker product structure in the coefficient matrices [32].

2.1 Scalar convection-diffusion problem

We first consider the simple scalar convection-diffusion problem:

$$Lu := -\nu\Delta u + \mathbf{w}\cdot\nabla u = f \quad \text{in } \Omega, \tag{2}$$

with Dirichlet boundary condition on $\partial\Omega$. Here we consider the special case of the horizontal wind, i.e., $\mathbf{w} = [1,0]^T$.

We specifically consider the Streamline Upwind Petrov-Galerkin (SUPG) method [23, 25] with bilinear finite elements on a regular grid with square elements of size $h \times h$, where $h = 1/(N+1)$, and N represents the number of inner nodes along each side. We arrange the nodes from left to right and from bottom to top.

Define $M := \text{tridiag}(1,4,1), K := \text{tridiag}(-1,2,-1), C := \text{tridiag}(-1,0,1)$. After discretization we get the discretization form of each term as follows, where u is the discretization of the corresponding variable in (2). Here we also list the corresponding difference stencils.

1) Discretization of the diffusion term: $\frac{\nu}{6}(M\otimes K + K\otimes M)u$.

$$\frac{\nu}{6}\times \begin{array}{ccc} -2 & -2 & -2 \\ -2 & 16 & -2 \\ -2 & -2 & -2 \end{array}$$

2) Discretization of the convection term: $\frac{h}{12}(M\otimes C)u$.

$$\frac{h}{12}\times \begin{array}{ccc} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{array}$$

3) Discretization of the stabilization term: $\frac{\delta}{6}(M\otimes K)u$, where $\delta = \frac{h}{2\|\mathbf{w}\|}\left(1-\frac{1}{P_e}\right)$, and P_e is the Peclet number: $P_e = \frac{h\|\mathbf{w}\|}{2\nu}$.

$$\frac{\delta}{6}\times \begin{array}{ccc} -1 & 2 & -1 \\ -4 & 8 & -4 \\ -1 & 2 & -1 \end{array}$$

Similar expressions for these three terms can be found in [21]. For the boundary condition, we take $u(x,1) = 1(k/(N+1) \le x < 1)$, and $u = 0$ elsewhere on $\partial\Omega$. We choose $k = 7$ in

the computation. Let $B = [0_{N\times(N-1)}, \overline{b}]$, where $\overline{b} = [\overline{b}_1, \cdots, \overline{b}_N]^T$, and $\overline{b}_j = 0$ $(j = 1, \cdots, 5)$, $\overline{b}_6 = \frac{\nu}{3} - \frac{h}{12} + \frac{\delta}{6}$, $\overline{b}_7 = \frac{2\nu}{3} - \frac{h}{12} - \frac{\delta}{6}$, $\overline{b}_j = \nu$ $(j = 8, \cdots, N-1)$, $\overline{b}_N = \frac{2\nu}{3} + \frac{h}{12} - \frac{\delta}{6}$. Then the boundary conditions can be expressed by B. At last we need to solve the following linear systems,

$$\left[M \otimes \left(\frac{\nu+\delta}{6}K + \frac{h}{12}C\right) + \frac{\nu}{6}K \otimes M\right] u = \mathsf{vec}(B). \tag{3}$$

2.2 Stokes problem

Consider the Stokes flow in a rectangle domain Ω:

$$\begin{aligned} -\nu\Delta\mathbf{u} + \nabla p &= 0, \\ -\nabla\cdot\mathbf{u} &= 0, \end{aligned} \tag{4}$$

where $\mathbf{u} = (u, v)^T$ denotes the velocity field, and p the pressure. Here we use Dirichlet boundary condition: $u = v = 0$ on $x = 0, x = 1, y = 0$; $u = 1$, $v = 0$ on the top boundary $y = 1$.

We discrete the computation domain with $Q_1 - P_0$ element, where the velocity is located on the node, the pressure is constant in the center of each element, and the cell width is $h = 1/n$. After discretization of (4), we obtain

$$\begin{bmatrix} A_1 & 0 & B_1^T \\ 0 & A_1 & B_2^T \\ B_1 & B_2 & C \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} f_1 \\ 0 \\ 0 \end{bmatrix}, \tag{5}$$

where u, v and p are numbered from left to right and from bottom to top. The coefficient matrix can be given in detail as follows,

$$\begin{bmatrix} \nu/6\,(M\otimes K + K\otimes M) & 0 & h/2\,(H_n^T \otimes H_o^T) \\ 0 & \nu/6\,(M\otimes K + K\otimes M) & h/2\,(H_o^T \otimes H_n^T) \\ h/2\,(H_n \otimes H_o) & h/2\,(H_o \otimes H_n) & -\beta h^2(I\otimes T_N + T_N \otimes I) \end{bmatrix}.$$

That is, $A_1 = \nu/6\,(M\otimes K + K\otimes M)$, $B_1 = h/2\,(H_n \otimes H_o)$, $B_2 = h/2\,(H_o \otimes H_n)$, $C = -\beta h^2(I \otimes T_N + T_N \otimes I)$, where $M = \mathrm{tridiag}(1, 4, 1) \in \mathbb{R}^{(n-1)\times(n-1)}$, $K = \mathrm{tridiag}(-1, 2, -1) \in \mathbb{R}^{(n-1)\times(n-1)}$, $T_N = \mathrm{tridiag}(-1, 2, -1) - e_1e_1^T - e_ne_n^T \in \mathbb{R}^{n\times n}$, and H_o, H_n are bidiagonal matrices with $H_o = \mathsf{sparse}(1:n-1, 1:n-1, -\mathsf{ones}(1, n-1), n, n-1) + \mathsf{sparse}(2:n, 1:n-1, \mathsf{ones}(1, n-1), n, n-1) \in \mathbb{R}^{n\times(n-1)}$, $H_n = \mathsf{sparse}(1:n-1, 1:n-1, \mathsf{ones}(1, n-1), n, n-1) + \mathsf{sparse}(2:n, 1:n-1, \mathsf{ones}(1, n-1), n, n-1) \in \mathbb{R}^{n\times(n-1)}$. Here sparse and ones are MATLAB notations. In the right hand side, $f_1 = \left(6 \times \frac{\nu}{6}\right)(\epsilon_{n-1} \otimes \epsilon) \in \mathbb{R}^{(n-1)^2\times 1}$, where e_1 and e_n are the first and n-th column vector of unit matrix I_n, ϵ_{n-1} is the $(n-1)$th column vector of unit matrix I_{n-1}, $\epsilon = [1, \cdots, 1]^T \in \mathbb{R}^{(n-1)\times 1}$. In local stabilization, we take $\beta = 0.25$, and take $\beta = 1$ for global stabilization.

2.3 Oseen problem

We consider the Oseen flow in the rectangular region Ω, with the same Dirichlet boundary conditions on $\partial\Omega$ as Stokes problem. Let $\mathbf{w} = (w_1, w_2)^T$ denote the wind. The governing equations are

$$\begin{aligned} -\nu\Delta\mathbf{u} + (\mathbf{w}\cdot\nabla)\mathbf{u} + \nabla p &= \mathbf{f}, \\ -\nabla\cdot\mathbf{u} &= 0. \end{aligned} \tag{6}$$

We divide the flow region into a uniform $n \times n$ grid of cells with width $h = 1/n$. We use Marker and Cell (MAC) discretization, and the discrete velocities and pressures are defined on a staggered grid [10]. The MAC finite difference discretization of (6) yields

$$\begin{bmatrix} \nu A_1 + \frac{h}{2} N_1 & 0 & h(I_n \otimes H_o^T) \\ 0 & \nu A_2 + \frac{h}{2} N_2 & h(H_o^T \otimes I_n) \\ h(I_n \otimes H_o) & h(H_o \otimes I_n) & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} f_1 \\ 0 \\ 0 \end{bmatrix}, \tag{7}$$

where A_i $(i = 1, 2)$ is the discrete Laplace operator, corresponding to diffusion, and N_i $(i = 1, 2)$ relates to the discrete convection operator. In the right hand side, $f_1 = 2\nu\,(e_n \otimes u_b) \in \mathbb{R}^{n(n-1)\times 1}$, where $u_b \in \mathbb{R}^{(n-1)\times 1}$ is the velocity on the top boundary.

We know that the diffusion terms A_1 and A_2 can be expressed by Kronecker products [10]. For convection term, we can also express it explicitly by the summation of Kronecker products. Thus we can further give the detail structures of A_1, A_2, N_1 and N_2 as follows.

$$A_1 = I_n \otimes T_D + T_E \otimes I_{n-1}, \qquad N_1 = \sum_{i=1}^{n} \left(e_i e_i^T \otimes W_i^X\right) - \sum_{j=1}^{n-1} \left(W_j^Y \otimes \epsilon_j \epsilon_j^T\right), \tag{8}$$

$$A_2 = I_{n-1} \otimes T_E + T_D \otimes I_n, \qquad N_2 = \sum_{\alpha=1}^{n-1} \left(\epsilon_\alpha \epsilon_\alpha^T \otimes \widehat{W}_\alpha^X\right) - \sum_{\beta=1}^{n} \left(\widehat{W}_\beta^Y \otimes e_\beta e_\beta^T\right). \tag{9}$$

where $T_D = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{(n-1)\times(n-1)}$, $T_E = \text{tridiag}(-1, 2, -1) + e_1 e_1^T + e_n e_n^T \in \mathbb{R}^{n\times n}$ [10], and

$$W_i^X = \begin{bmatrix} 0 & w_{1,i}^X & & \\ -w_{1,i}^X & \ddots & \ddots & \\ & \ddots & \ddots & w_{n-2,i}^X \\ & & -w_{n-2,i}^X & 0 \end{bmatrix}, \quad \widehat{W}_\alpha^X = \begin{bmatrix} 0 & \widehat{w}_{1,\alpha}^X & & \\ -\widehat{w}_{1,\alpha}^X & \ddots & \ddots & \\ & \ddots & \ddots & \widehat{w}_{n-1,\alpha}^X \\ & & -\widehat{w}_{n-1,\alpha}^X & 0 \end{bmatrix},$$

$$W_j^Y = \begin{bmatrix} 0 & -w_{j,1}^Y & & \\ w_{j,1}^Y & \ddots & \ddots & \\ & \ddots & \ddots & -w_{j,n-1}^Y \\ & & w_{j,n-1}^Y & 0 \end{bmatrix}, \quad \widehat{W}_\beta^Y = \begin{bmatrix} 0 & -\widehat{w}_{\beta,1}^Y & & \\ \widehat{w}_{\beta,1}^Y & \ddots & \ddots & \\ & \ddots & \ddots & -\widehat{w}_{\beta,n-2}^Y \\ & & \widehat{w}_{\beta,n-2}^Y & 0 \end{bmatrix},$$

$$w_{k,i}^X = \omega_1((k + \frac{1}{2})h, (i - \frac{1}{2})h), \qquad \widehat{w}_{k,\alpha}^X = \omega_1(kh, \alpha h),$$

$$w_{j,k}^Y = \omega_2(jh, kh), \qquad \widehat{w}_{\beta,k}^Y = \omega_2((\beta - \frac{1}{2})h, (k + \frac{1}{2})h).$$

Note that T_D has the same form as K defined before, here we use T_D to follow the notation used in [10]. Obviously, the diffusion terms A_1, A_2 are symmetric, while the convection terms N_1, N_2 are antisymmetric.

For the case where the wind is constant, for example, $\mathbf{w} = (a, b)$, we can further simplify the convection terms as follows.

$$N_1 = aI_n \otimes C_{n-1} + bC_n \otimes I_{n-1}, \tag{10}$$

$$N_2 = aI_{n-1} \otimes C_n + bC_{n-1} \otimes I_n, \tag{11}$$

where $C_n = \text{tridiag}(-1, 0, 1)$, an antisymmetric matrix of order n.

For the constant wind, the convection-diffusion term can be expressed as follows.

$$\nu A_1 + \frac{h}{2}N_1 = I_n \otimes \left(\nu T_D + \frac{ah}{2}C_{n-1}\right) + \left(\nu T_E + \frac{ah}{2}C_n\right) \otimes I_{n-1},$$
$$\nu A_2 + \frac{h}{2}N_2 = I_{n-1} \otimes \left(\nu T_E + \frac{ah}{2}C_n\right) + \left(\nu T_D + \frac{ah}{2}C_{n-1}\right) \otimes I_n.$$

2.4 Navier-Stokes problem

Consider the incompressible flow in Ω, with Dirichlet boundary conditions on $\partial\Omega$. On the boundaries the velocities are zeroes except the horizontal velocity $u = 1$ on the upper boundary. The governing equations are Navier-Stokes equations:

$$\begin{aligned} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p &= \mathbf{f}, \\ -\nabla\cdot\mathbf{u} &= 0. \end{aligned}$$

Here only the Navier-Stokes equation is nonlinear. But we will apply linearization to it. If we use Picard iteration, we need to solve the following Oseen problem at each step.

$$\begin{aligned} -\nu\Delta\mathbf{u}^{(k)} + (\mathbf{u}^{(k-1)}\cdot\nabla)\mathbf{u}^{(k)} + \nabla p^{(k)} &= \mathbf{f}^{(k-1)}, \\ -\nabla\cdot\mathbf{u}^{(k)} &= 0, \end{aligned}$$

where the superscript $(k-1)$ denotes the results obtained by the former Picard iteration step.

If we use Picard iteration to solve Navier-Stokes equations, we need to solve an Oseen problem at each time step with the coefficient matrix as in (7). But the convection terms are different, in which the terms W_i^X, W_j^Y, $\widehat{W}_\alpha^X$, $\widehat{W}_\beta^Y$ are defined as follows,

$$W_i^X = \text{diag}(w_i^X, 1) + \text{diag}(-w_i^X, -1), \qquad w_i^X = \frac{1}{2}H_{n-1}^T U e_i,$$
$$W_j^Y = \text{diag}(-w_j^Y, 1) + \text{diag}(w_j^Y, -1), \qquad w_j^Y = \frac{1}{2}V^T H_n e_j,$$
$$\widehat{W}_\alpha^X = \text{diag}(\widehat{w}_\alpha^X, 1) + \text{diag}(-\widehat{w}_\alpha^X, -1), \qquad \widehat{w}_\alpha^X = \frac{1}{2}U H_n e_\alpha,$$
$$\widehat{W}_\beta^Y = \text{diag}(-\widehat{w}_\beta^Y, 1) + \text{diag}(\widehat{w}_\beta^Y, -1), \qquad \widehat{w}_\beta^Y = \frac{1}{2}H_{n-1}^T V^T e_\beta,$$

where $H_n = \mathsf{sparse}(1:n-1, 1:n-1, \mathsf{ones}(1, n-1), n, n-1) + \mathsf{sparse}(2:n, 1:n-1, \mathsf{ones}(1, n-1), n, n-1) \in \mathbb{R}^{n\times(n-1)}$ as defined before, e_i, e_j, e_α, e_β are unit vectors with conformal dimensions, $U \in \mathbb{R}^{(n-1)\times n}$, $V \in \mathbb{R}^{n\times(n-1)}$, which satisfy $u^{(k-1)} = \mathsf{vec}(U)$, $v^{(k-1)} = \mathsf{vec}(V)$. Here we omit the superscript $(k-1)$ in U and V, which denotes the results of the former time step.

The coefficient matrices of (5) and (7) are rank-one deficient. The null-vector is $[0^T, e^T]^T$, where $0 \in \mathbb{R}^{2n(n-1)\times 1}$, $e = [1, \cdots, 1]^T \in \mathbb{R}^{n^2\times 1}$. So p is defined only up to an additive constant. We can, for example, delete the last row and column to make the coefficient matrix nonsingular. Otherwise, since (7) is range symmetric, if we apply GMRES [27] to solve it, we can get $\mathcal{A}^\dagger b$ [5], where $\mathcal{A}$ is the coefficient of (7), b is the right hand side, and $\mathcal{A}^\dagger$ is the Moore-Penrose inverse of $\mathcal{A}$ [1, 12, 31]. We observe that the nonzero eigenvalue of the singular case is bounded further away from the origin than the nonsingular case. Sloving the singular case is usually faster than the nonsingular case[8]. The saddle point system (1) with (1,1) singular block is considered in [9, 15].

3 Kronecker Product Approximation Preconditioner

In the coefficient matrices we just obtained, we can find Kronecker product structure. For the first scalar case, the coefficient matrix is a summation of two Kronecker products. For the latter three cases, there exists a 2×2 block structure in the coefficient matrices, each block is expressed as a Kronecker product or a summation of several Kronecker products. For the summation of several Kronecker products, we employ an approach that uses one Kronecker product to approximate the sum of several Kronecker products. The problem can be expressed as follows. For $G_i, F_i \in \mathbb{R}^{p\times q}(i = 1, \cdots, s)$, we seek $X, Y \in \mathbb{R}^{p\times q}$, such that [24]

$$\min \left\| \sum_{i=1}^{s} (G_i \otimes F_i) - X \otimes Y \right\|_F .$$

Lemma [20] As is stated above, X, Y are the linear combination of G_i and F_i respectively. Assuming $X = \sum_{i=1}^{s} \alpha_i G_i, Y = \sum_{i=1}^{s} \beta_i F_i$, we have [20]

$$\begin{aligned} f(\alpha,\beta) &:= \left\| \sum_{i=1}^{s} (G_i \otimes F_i) - X \otimes Y \right\|_F^2 \\ &= \sum_{i=1}^{s}\sum_{j=1}^{s} \mathrm{tr}(G_i^T G_j)\mathrm{tr}(F_i^T F_j) - 2\sum_{i=1}^{s}\left(\sum_{j=1}^{s} \alpha_j \mathrm{tr}(G_i^T G_j) \sum_{j=1}^{s} \beta_j \mathrm{tr}(F_i^T F_j) \right) \\ &\quad + \sum_{i=1}^{s}\sum_{j=1}^{s} \alpha_i\alpha_j \mathrm{tr}(G_i^T G_j) \sum_{i=1}^{s}\sum_{j=1}^{s} \beta_i\beta_j \mathrm{tr}(F_i^T F_j). \end{aligned}$$

Let's define $\widehat{G} := (\widehat{g}_{ij}), \widehat{F} := (\widehat{f}_{ij})$, where $\widehat{g}_{ij} = \mathrm{tr}(G_i^T G_j), \widehat{f}_{ij} = \mathrm{tr}(F_i^T F_j)$. To compute the traces, $\mathrm{tr}(G_i^T G_j)$ and $\mathrm{tr}(F_i^T F_j)$ $(i = 1, \cdots, s; j = i, \cdots, s)$, we can use the symmetry property: $\mathrm{tr}(G_i^T G_j) = \mathrm{tr}(G_j^T G_i)$. Note that we only need to operate on small matrices G_i, F_i to form $\widehat{G}$ and $\widehat{F}$. With these definitions, we have

$$f(\alpha,\beta) = \mathrm{tr}(\widehat{G}\widehat{F}) - 2\alpha^T \widehat{G}\widehat{F}\beta + (\alpha^T \widehat{G}\alpha)(\beta^T \widehat{F}\beta). \tag{12}$$

Then we need to solve the following minimization problem to find α and β,

$$\min_{\alpha,\beta} f(\alpha,\beta).$$

We can use the nonlinear optimization method to determine the parameters. For example, we can use function fminunc or fminsearch in MATLAB to determine the $2s$ parameters, $\alpha = (\alpha_1, \cdots, \alpha_s)^T, \beta = (\beta_1, \cdots, \beta_s)^T$. In practical problems, s is small, there are only a few parameters. In this paper, s is only 2 or 4.

Based on this basic idea, we construct the Kronecker product approximation preconditioners for the convection-diffusion problems.

3.1 Scalar convection-diffusion problem

After SUPG discretization we obtain the linear system in the form of

$$(G_1 \otimes F_1 + G_2 \otimes F_2)x = b,$$

where $G_1 = F_2 = M$, $F_1 = \frac{\nu+\delta}{6}K + \frac{h}{12}C$, and $G_2 = \frac{\nu}{6}K$. We seek for the Kronecker product approximation $G \otimes F$, where $G = \alpha_1 G_1 + \alpha_2 G_2, F = \beta_1 F_1 + \beta_2 F_2$, such that $\|G_1 \otimes F_1 + G_2 \otimes F_2 - G \otimes F\|_F$ is minimized.

We denote the linear system as $Ax = b$. The size of the original coefficient matrix is $N^2 \times N^2$, while G, F are $N \times N$. We need not to form the coefficient explicitly, we just store G_1, G_2, F_1, F_2. The matrix-vector product can be done like this: $Ax = (G_1 \otimes F_1 + G_2 \otimes F_2)x = \mathsf{vec}(F_1 X G_1^T + F_2 X G_2^T)$, where $\mathsf{vec}(X) = x$. After preconditoning, we need to compute $(G \otimes F)^{-1} z$, which can be obtained by solving $(G \otimes F)w = z$. Because G and F are both small matrices, we can use the inverse directly. Therefore $w = \mathsf{vec}(F^{-1} Z G^{-T})$, where $\mathsf{vec}(Z) = z$. Note that G and F are tridiagonal matrices. The inverse of a tridiagonal matrix can be expressed by two sequences, and there exist efficient algorithms for it (see [22] and the references therein).

3.2 Stokes problem

The preconditioner involves two approximations. The first approximation is used for the first two diagonal blocks:

$$M \otimes K + K \otimes M \approx G \otimes F.$$

The second approximation is used for the Schur complement:

$$\begin{aligned} S &\approx -\frac{3h^2}{2\nu}\Big[\frac{2}{3}\beta\nu(I \otimes T_N + T_N \otimes I) + (H_n G^{-1} H_n^T) \otimes (H_o F^{-1} H_o^T) \\ &\quad + (H_o G^{-1} H_o^T) \otimes (H_n F^{-1} H_n^T)\Big] \\ &\approx -\frac{3h^2}{2\nu}(S_1 \otimes S_2). \end{aligned}$$

Then we can choose a diagonal block preconditioner P_d and a tridiagonal block preconditioner P_t as following:

$$P_d = \frac{\nu}{6}\begin{bmatrix} G \otimes F & 0 & 0 \\ 0 & G \otimes F & 0 \\ 0 & 0 & -\frac{9h^2}{\nu^2}(S_1 \otimes S_2) \end{bmatrix},$$

$$P_t = \frac{\nu}{6}\begin{bmatrix} G \otimes F & 0 & \frac{3h}{\nu}(H_n^T \otimes H_o^T) \\ 0 & G \otimes F & \frac{3h}{\nu}(H_o^T \otimes H_n^T) \\ 0 & 0 & -\frac{9h^2}{\nu^2}(S_1 \otimes S_2) \end{bmatrix}.$$

We can derive the inverse, for example,

$$P_t^{-1} = \frac{6}{\nu}\begin{bmatrix} G^{-1} \otimes F^{-1} & 0 & \frac{\nu}{3h}(G^{-1} H_n^T S_1^{-1}) \otimes (F^{-1} H_o^T S_2^{-1}) \\ 0 & G^{-1} \otimes F^{-1} & \frac{\nu}{3h}(G^{-1} H_o^T S_1^{-1}) \otimes (F^{-1} H_n^T S_2^{-1}) \\ 0 & 0 & -\frac{\nu^2}{9h^2}(S_1^{-1} \otimes S_2^{-1}) \end{bmatrix}.$$

At each iteration step we need to solve $P_d^{-1} z$ or $P_t^{-1} z$. Because we only deal with matrices of order n, we can compute it directly.

$$P_d^{-1} z = \frac{6}{\nu}\begin{bmatrix} \mathsf{vec}(F^{-1} Z_u G^{-T}) \\ \mathsf{vec}(F^{-1} Z_v G^{-T}) \\ \mathsf{vec}(-\frac{\nu^2}{9h^2} S_2^{-1} Z_p S_1^{-T}) \end{bmatrix},$$

$$P_t^{-1} z = \frac{6}{\nu} \begin{bmatrix} \mathsf{vec}(F^{-1}(Z_u + \frac{\nu}{3h} H_o^T S_2^{-1} Z_p S_1^{-T} H_n) G^{-T}) \\ \mathsf{vec}(F^{-1}(Z_v + \frac{\nu}{3h} H_n^T S_2^{-1} Z_p S_1^{-T} H_o) G^{-T}) \\ \mathsf{vec}(-\frac{\nu^2}{9h^2} S_2^{-1} Z_p S_1^{-T}) \end{bmatrix},$$

where $z = [z_1^T, z_2^T, z_3^T]^T$, and $z_1 = \mathsf{vec}(Z_u), z_2 = \mathsf{vec}(Z_v), z_3 = \mathsf{vec}(Z_p)$.

Note that h/ν is the cell Reynolds number. Hence, the tridiagonal block preconditioner P_t can be regarded as a modification of the diagonal block preconditioner P_d depending on the cell Reynolds number.

3.3 Oseen problem

If we use MAC finite difference discretization on Oseen problem, where the discrete velocities and pressures are defined on a staggered grid, the discrete Laplace operator and the discrete convection operator in one direction will result in a block involving a summation of $(2n+1)$ Kronecker products. We can use the similar approach to Oseen problem. But we first need to approximate the convection term. For example,

$$W_i^X \approx d_i^X \begin{bmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 0 \end{bmatrix}, \quad \text{where } d_i^X = \frac{1}{n-2} \sum_{k=1}^{n-2} w_{k,i}^X.$$

Here the convection term is still kept as antisymmetric. Then we have

$$\sum_{i=1}^{n} \left(e_i e_i^T \otimes W_i^X \right) \approx \begin{bmatrix} d_1^X & & & \\ & d_2^X & & \\ & & \ddots & \\ & & & d_n^x \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 0 \end{bmatrix} := D_n^X \otimes C_{n-1}.$$

Hence we can obtain the approximation for convection terms.

$$\begin{aligned} N_1 &\approx D_n^X \otimes C_{n-1} + C_n \otimes D_{n-1}^Y, \\ N_2 &\approx \widehat{D}_{n-1}^X \otimes C_{n-1} + C_n \otimes \widehat{D}_n^Y, \end{aligned}$$

where D_{n-1}^Y, $\widehat{D}_{n-1}^X$, $\widehat{D}_n^Y$ are diagonal matrices defined in a similar way for D_n^X. The convection terms have the similar forms as (10) and (11) for the case of constant wind.

Secondly we approximate the diffusion and convection term. We assume that

$$\begin{aligned} \nu A_1 + \frac{h}{2} N_1 &\approx \nu (G_U \otimes F_U), \\ \nu A_2 + \frac{h}{2} N_2 &\approx \nu (G_V \otimes F_V). \end{aligned}$$

Note that G_U, F_U, G_V and F_V are still tridiagonal matrices. Their inverses can be efficiently computed [22].

In the third step, we approximate the Schur complement.

$$\begin{aligned} S &\approx -h^2/\nu[G_U^{-1} \otimes (H_o F_U^{-1} H_o^T) + (H_o G_V^{-1} H_o^T) \otimes F_V^{-1}] \\ &\approx -h^2/\nu(S_1 \otimes S_2). \end{aligned}$$

As in Stokes problem, we can construct diagonal block preconditioner and tridiagonal block preconditioner as follows.

$$P_d = \nu \begin{bmatrix} G_U \otimes F_U & 0 & 0 \\ 0 & G_V \otimes F_V & 0 \\ 0 & 0 & -\frac{h^2}{\nu^2}(S_1 \otimes S_2) \end{bmatrix},$$

$$P_t = \nu \begin{bmatrix} G_U \otimes F_U & 0 & \frac{h}{\nu}(I_n \otimes H_o^T) \\ 0 & G_V \otimes F_V & \frac{h}{\nu}(H_o^T \otimes I_n) \\ 0 & 0 & -\frac{h^2}{\nu^2}(S_1 \otimes S_2) \end{bmatrix}.$$

We can easily check the inverse of P_t.

$$P_t^{-1} = \frac{1}{\nu} \begin{bmatrix} G_U^{-1} \otimes F_U^{-1} & 0 & \frac{\nu}{h}(G_U^{-1} S_1^{-1}) \otimes (F_U^{-1} H_o^T S_2^{-1}) \\ 0 & G_V^{-1} \otimes F_V^{-1} & \frac{\nu}{h}(G_V^{-1} H_o^T S_1^{-1}) \otimes (F_V^{-1} S_2^{-1}) \\ 0 & 0 & -\frac{\nu^2}{h^2}(S_1^{-1} \otimes S_2^{-1}) \end{bmatrix}$$

In practical computation we need matrix-vector products $P_d^{-1} z$ and $P_t^{-1} z$.

$$P_d^{-1} z = \frac{1}{\nu} \begin{bmatrix} \mathsf{vec}(F_U^{-1} Z_u G_U^{-T}) \\ \mathsf{vec}(F_V^{-1} Z_v G_V^{-T}) \\ \mathsf{vec}(-\frac{\nu^2}{h^2} S_2^{-1} Z_p S_1^{-T}) \end{bmatrix}.$$

$$P_t^{-1} z = \frac{1}{\nu} \begin{bmatrix} \mathsf{vec}(F_U^{-1}(Z_u + \frac{\nu}{h} H_o^T S_2^{-1} Z_p S_1^{-T}) G_U^{-T}) \\ \mathsf{vec}(F_V^{-1}(Z_v + \frac{\nu}{h} S_2^{-1} Z_p S_1^{-T} H_o) G_V^{-T}) \\ \mathsf{vec}(-\frac{\nu^2}{h^2} S_2^{-1} Z_p S_1^{-T}) \end{bmatrix}.$$

Again we find that the tridiagonal block preconditioner P_t is a modification of the diagonal block preconditioner P_d based on the cell Reynolds number h/ν.

For Oseen problem, we can construct the constraint preconditioner in the following form,

$$P_c = \begin{bmatrix} \nu(G_U \otimes F_U) & 0 & h(I_n \otimes H_o^T) \\ 0 & \nu(G_V \otimes F_V) & h(H_o^T \otimes I_n) \\ h(I_n \otimes H_o) & h(H_o \otimes I_n) & 0 \end{bmatrix}.$$

But this matrix is rank-one deficient, and cannot be used as preconditioner directly. Using previous approximation of Schur complement, we can give its approximate inverse in form as follows.

$$P_c^{-1} \approx \frac{1}{\nu} \begin{bmatrix} C_{11} & C_{12} & \frac{\nu}{h} C_{13} \\ C_{21} & C_{22} & \frac{\nu}{h} C_{23} \\ \frac{\nu}{h} C_{31} & \frac{\nu}{h} C_{32} & -\frac{\nu^2}{h^2} C_{33} \end{bmatrix},$$

where

$$\begin{aligned}
C_{11} &= G_U^{-1} \otimes F_U^{-1} - (G_U^{-1} S_1^{-1} G_U^{-1}) \otimes (F_U^{-1} H_o^T S_2^{-1} H_o F_U^{-1}),\\
C_{12} &= -(G_U^{-1} S_1^{-1} H_o G_V^{-1}) \otimes (F_U^{-1} H_o^T S_2^{-1} F_V^{-1}),\\
C_{13} &= (G_U^{-1} S_1^{-1}) \otimes (F_U^{-1} H_o^T S_2^{-1}),\\
C_{21} &= -(G_V^{-1} H_o^T S_1^{-1} G_U^{-1}) \otimes (F_V^{-1} S_2^{-1} H_o F_U^{-1}),\\
C_{22} &= G_V^{-1} \otimes F_V^{-1} - (G_V^{-1} H_o^T S_1^{-1} H_o G_V^{-1}) \otimes (F_V^{-1} S_2^{-1} F_V^{-1}),\\
C_{23} &= (G_V^{-1} H_o^T S_1^{-1}) \otimes (F_V^{-1} S_2^{-1}),\\
C_{31} &= (S_1^{-1} G_U^{-1}) \otimes (S_2^{-1} H_o F_U^{-1}),\\
C_{32} &= (S_1^{-1} H_o G_V^{-1}) \otimes (S_2^{-1} F_V^{-1})\\
C_{33} &= S_1^{-1} \otimes S_2^{-1}.
\end{aligned}$$

Here we use approximate Schur complement, and this approximate inverse will be nonsingular. Let's define $T_V := H_o F_U^{-1} Z_u G_U^{-T} + F_V^{-1} Z_v G_V^{-T} H_o^T$. Then the matrix-vector product can be expressed as

$$P_c^{-1} z \approx \frac{1}{\nu} \begin{bmatrix} \mathsf{vec}(F_U^{-1}[Z_u + H_o^T S_2^{-1}(\frac{\nu}{h} Z_p - T_V) S_1^{-T}] G_U^{-T}) \\ \mathsf{vec}(F_V^{-1}[Z_v + S_2^{-1}(\frac{\nu}{h} Z_p - T_V) S_1^{-T} H_o] G_V^{-T}) \\ \mathsf{vec}(-\frac{\nu}{h} S_2^{-1}(\frac{\nu}{h} Z_p - T_V) S_1^{-T}) \end{bmatrix}.$$

From this formula, we can see that the constraint preconditioner P_c is a modification of the tridiagonal block preconditioner P_t. Hence, we can expect that the constraint preconditioner is better than the tridiagonal block preconditioner and diagonal block preconditioner.

Besides, we can delete the last row and column to make the coefficient matrix nonsingular. Then we introduce another kind of constraint preconditioner [32]. First we represent the coefficient matrix of (7) as

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_{11} & B^T \\ B & 0 \end{bmatrix}, \text{ where } \mathcal{A}_{11} = \begin{bmatrix} \nu A_1 + \frac{h}{2} N_1 & 0 \\ 0 & \nu A_2 + \frac{h}{2} N_2 \end{bmatrix}.$$

Then we use the following constraint preconditioner:

$$\mathcal{P}_c = \begin{bmatrix} \widetilde{\mathcal{A}}_{11} & B^T \\ B & 0 \end{bmatrix}, \text{ where } \widetilde{\mathcal{A}}_{11} = \begin{bmatrix} G_U \otimes F_U & 0 \\ 0 & G_V \otimes F_V \end{bmatrix}.$$

With the preconditioner $\mathcal{P}_c$, we need to solve the equation in the form of $\mathcal{P}_c z = y$. And the following decomposition is needed.

$$\begin{bmatrix} \widetilde{\mathcal{A}}_{11} & B^T \\ B & 0 \end{bmatrix}^{-1} = \begin{bmatrix} I & -\widetilde{\mathcal{A}}_{11}^{-1} B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \widetilde{\mathcal{A}}_{11}^{-1} & 0 \\ 0 & -(B \widetilde{\mathcal{A}}_{11}^{-1} B^T)^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B \widetilde{\mathcal{A}}_{11}^{-1} & I \end{bmatrix}.$$

In the inner iteration, we can use GMRES or BiCGStab to solve $(B \widetilde{\mathcal{A}}_{11}^{-1} B^T) x = b$. But in the numercial examples we do not use this strategy.

For the Navier-Stokes problem, it is a nonlinear case, and it needs the linearization. If we use Picard iteration, we need to solve an Oseen problem at each iteration step. So the similar techniques in this subsection can be applied to Navier-Stokes problem.

4 Numerical Tests

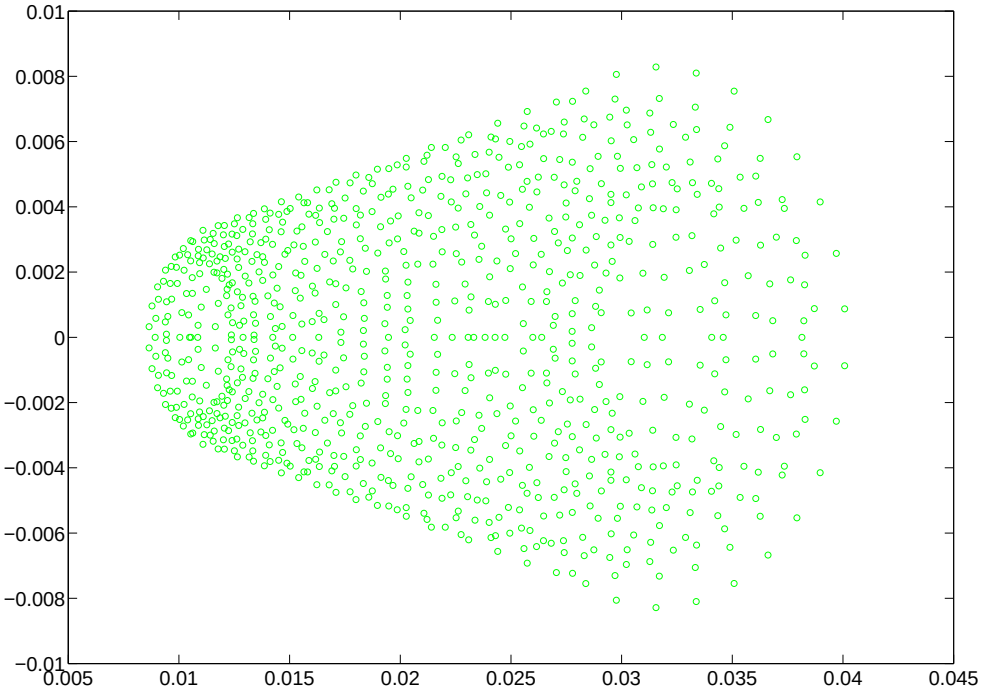
In this section we perform numerical tests (MATLAB R2006b; Intel(R) Core(TM)2 CPU 6600@2.40GHz, 3.21GB SDRAM) on the scalar convection-diffusion problem, Stokes problem, Ossen problem, and Navier-Stokes problem with our Kronecker product approximation (KPA) preconditoners, and compare KPA with ILU(t) or ILU('0'), where t is the drop tolerance, and ILU('0') is the incomplete LU of a sparse matrix with 0 level of fill-in. When the coefficient matrix is rank-one deficient, ILU often yields a singular upper triangular matrix, and can not be used as a preconditioner. But the KPA preconditoners are almost always nonsingular. Besides, we do not form the coefficient matrix explicitly, one important advantage of KPA is that we only need to process with matrices of order n although the original coefficient matrix is of order n^2. This results in less memory requirement.

4.1 Scalar case

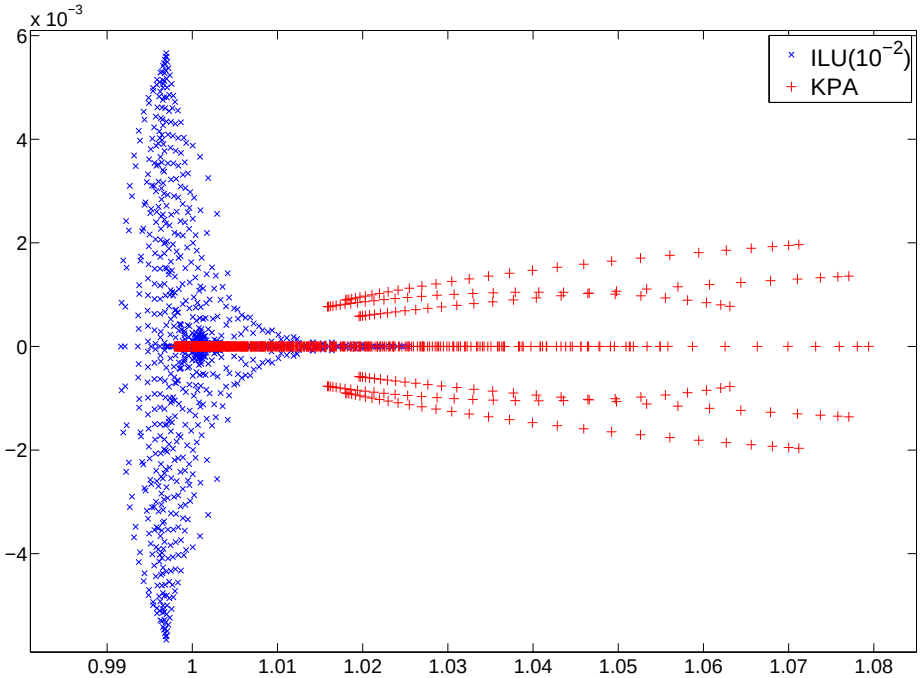
We take $\nu = 0.0001$, $n = 30$. The spectrum of original coefficient matrix and preconditioned matrix are shown in Figure 1. The spectrum of the preconditioned matrix is bounded around 1. The spectrum of the original matrix is also bounded, but it is much nearer to the origin, which delays the convergence. The convergence curve of GMRES on the case where $\nu = 0.0001$, $n = 15$ is given in Figure 2. The KPA preconditioner is as good as ILU(10^{-2}), but it needs less flops and storage. BiCGStab on the case where $\nu = 0.0001$, $n = 49$ shows the similar behavior (see Figure 3). We can see that KPA preconditioner is very efficient for this case. Here we do not use the efficient algorithm for the inverse of the tridiagonal matrix. We use the MATLAB function `inv` directly.

4.2 Stokes problem

Here we consider the Stokes flow in a rectangle domain $\Omega = (0, 1) \times (0, 1)$ with Dirichlet boundary condition: $u = v = 0$ on $x = 0, x = 1, y = 0$; $u = 1$, $v = 0$ on the top boundary $y = 1$. Figure 4 illustrates the spectral distribution of the coefficient matrix for the case where $\nu = 0.01$, with 32×32 grids and local stabilization $\beta = 0.25$. The spectrum of the original matrix spread like a curve in $[0, 3000] \times [-0.05, 0.1]$, while the spectrum of the preconditioned matrix is clustered around 1. So we can expect the preconditioned linear systems have better convergence behaviors. In fact, without preconditioning, GMRES(20) needs about 4000 iteration steps to converge. The number of iteration steps is 6 or 8 times larger than that of KPA preconditioners. Figure 5(a) shows the corresponding residual curves of GMRES(20). Compared with its original residual curve, both KPA preconditioner and ILU preconditioner greatly improve the convergence. Here ILU(t) is also quite good, but remember that it will suffer from fill-ins and needs more flops to construct. What's more, since the coefficient matrix is singular here, sometimes ILU yields singular upper triangular U, and cannot be used as a preconditioner. For example, ILU('0') for the case with 16×16 grids gives a singular U and cannot be used for preconditioning. ILU(10^{-1}) works, but it is not as good as KPA preconditioners in this case (see Figure 5(b)). Note that GMRES(20) also converges without preconditioning for this case, but it needs about 1200 iteration steps. The streamlines and pressure contour are given in Figure 6, which is solved with tridiagonal KPA preconditioner.



(a) Original matrix



(b) Preconditioned matrices

Figure 1: Spectra of the coefficient matrices for the scalar case

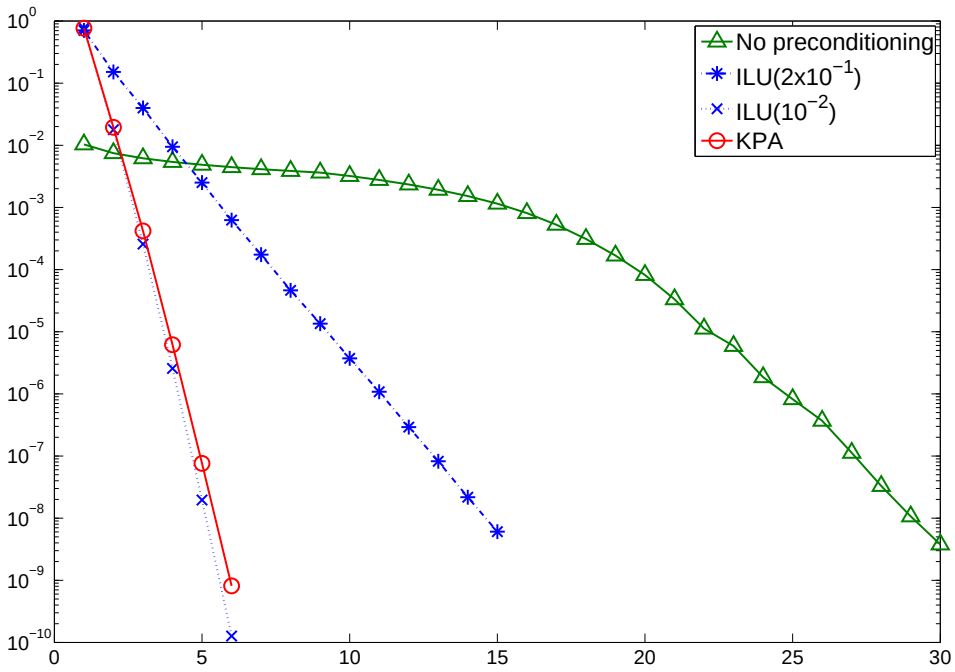


Figure 2: GMRES for the scalar case (16×16, $\nu = 0.0001$)

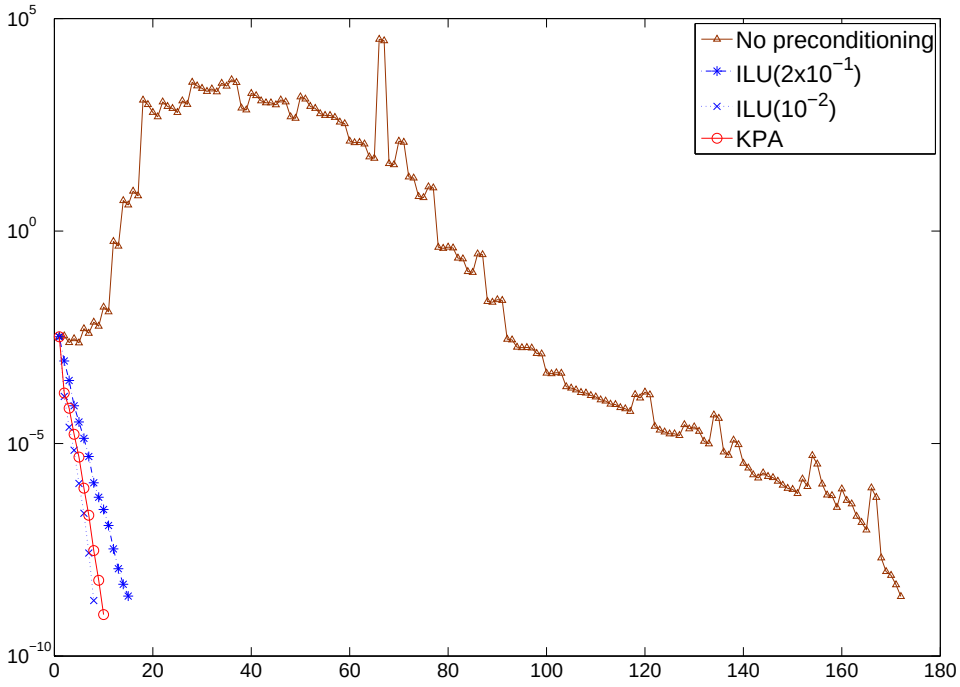
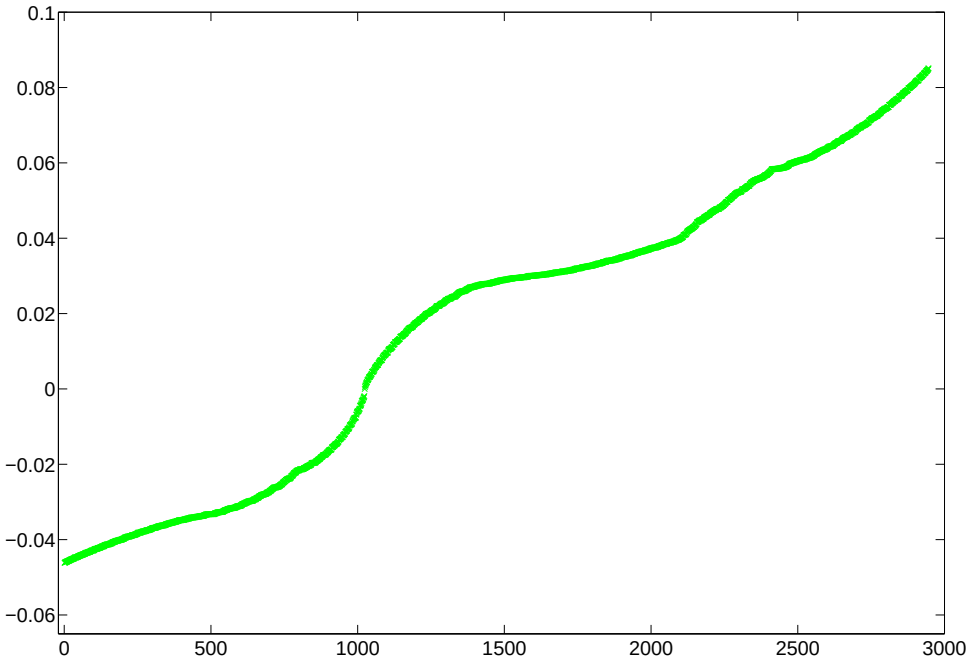
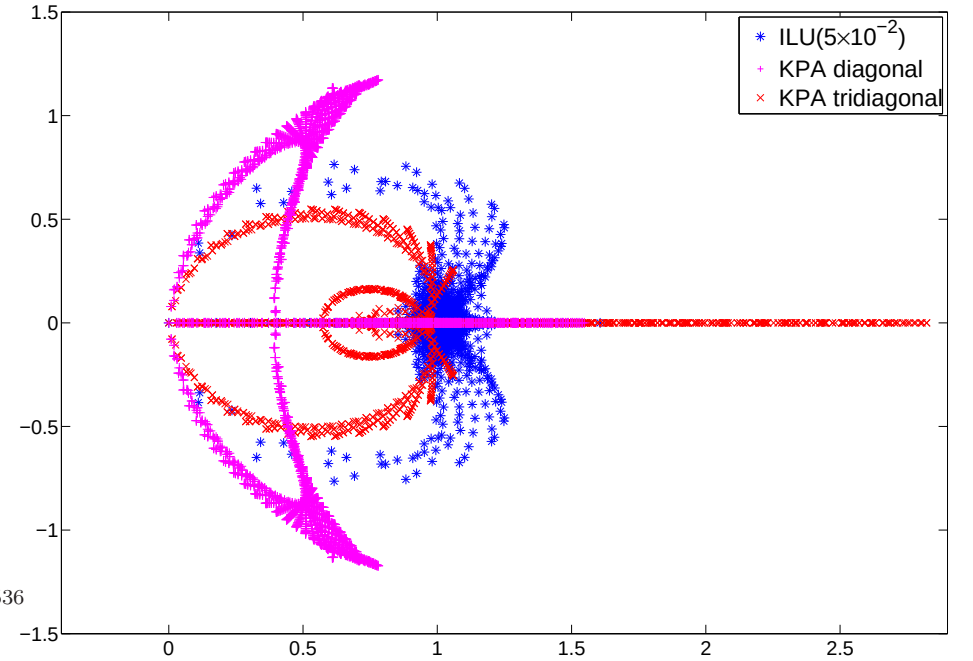


Figure 3: BiCGStab for the scalar case (50×50, $\nu = 0.0001$)

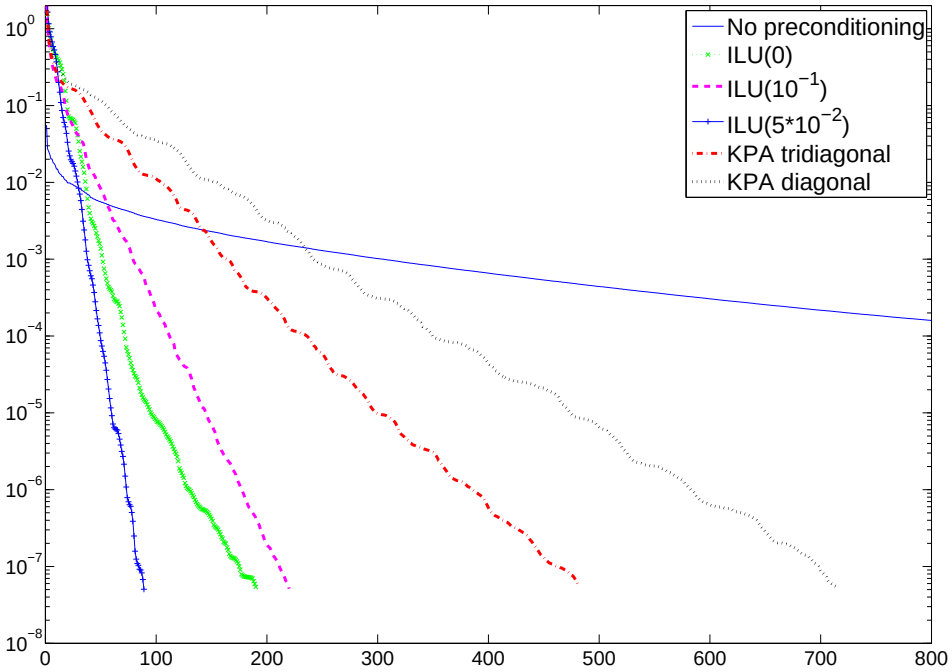


(a) Original matrix

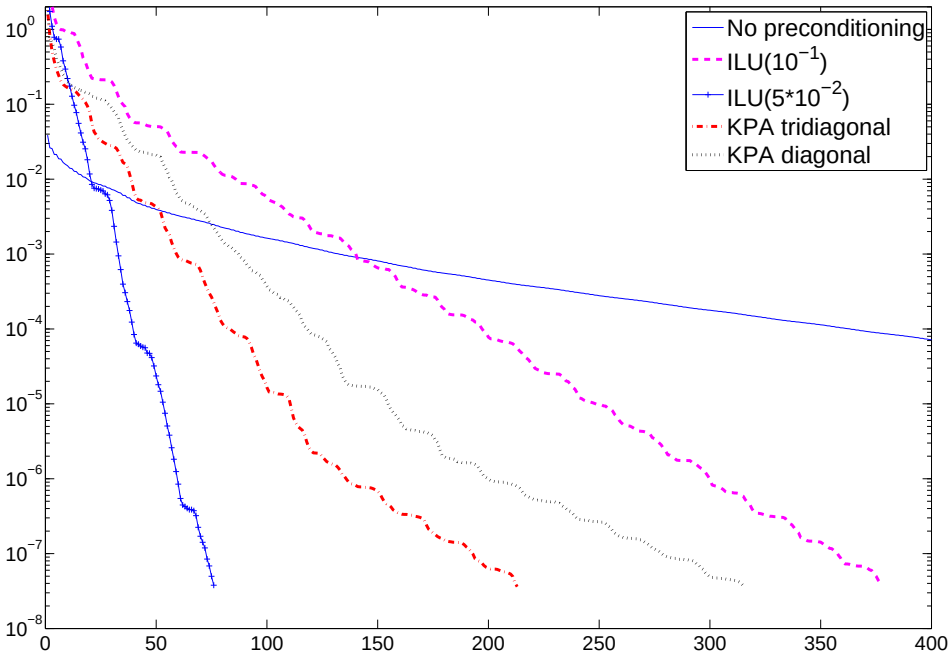


(b) Preconditioned matrices

Figure 4: Spectral distributions of Stokes problem (32×32, $\nu = 0.01$)

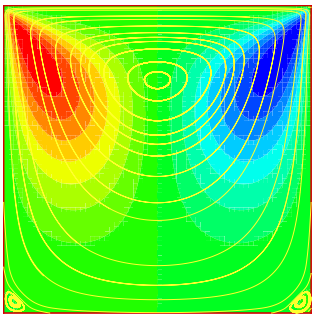


(a) 32×32 grids

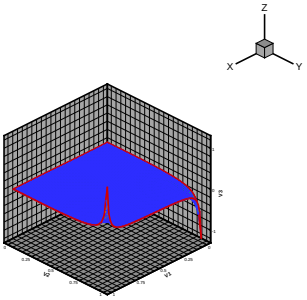


(b) 16×16 grids

Figure 5: Residual curves of GMRES(20) for Stokes problem ($\nu = 0.01$)



(a) Streamlines and vertical velocity contour



(b) Pressure contour

Figure 6: Stokes problem (64×64, $\nu = 0.01$)

4.3 Oseen problem

Let $\mathbf{w} = (w_1, w_2)^T$ denote the wind. We choose $\mathbf{w}$, such that it is the image of $[2y(1-x^2), -2x(1-y^2)]$ under the linear mapping from $(-1,1) \times (-1,1)$ to Ω. That is,

$$\mathbf{w} = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} 8x(1-x)(2y-1) \\ -8y(1-y)(2x-1) \end{bmatrix}.$$

Here we consider the case where $\nu = 1$ with 32×32 grids, and use full GMRES to solve it. With KPA preconditioners, the corresponding spectrum is more compact. Note that the coefficient matrix is rank-one deficient, ILU with larger drop tolerence often generates a singular upper triangular part, and cannot be used as preconditioners. For example, ILU(10^{-1}) and ILU(10^{-2}) do not work in this case, and ILU('0') cannot be used to construct the preconditioner either. In Figure 7, we need to choose a drop tolerance as small as 5×10^{-3}, while the number of nonzero elements in the incomplete factors are about 6.5 times larger than the number of nonzeros in the original matrix though the reordering is used to reduce the fill-ins. But the KPA preconditoners are almost always nonsingular and easy to construct. We can also see that the KPA constraint precondtioner is better than the KPA block tridiagonal preconditioner, and the latter is better than the KPA block diagonal preconditioner, which coincides with our former analysis.

4.4 Navier-Stokes problem

In every Picard iteration, we need to solve an Oseen problem. So the strategies in Oseen problem can be used here. Since the constraint preconditioner and the block tridiagonal preconditioner are better than block diagonal preconditioner, and are easier to construct than ILU, we only test the constraint preconditioner and the block tridiagonal preconditioner here. For the case where $\nu = 0.01$, with 64×64 girds, we need 11 Picard iteration steps. The residual curves of each

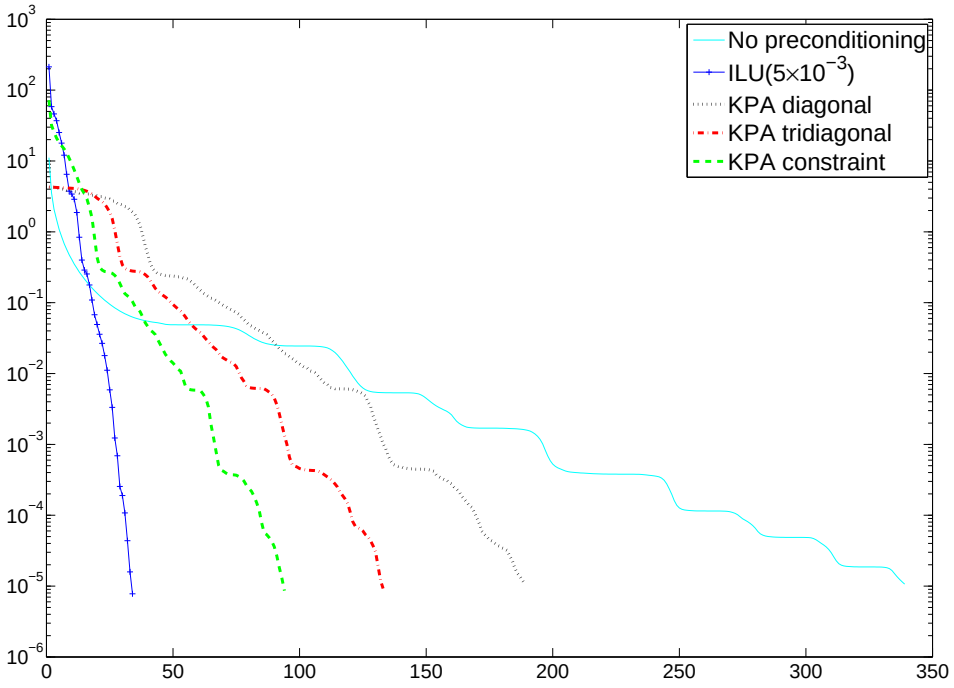


Figure 7: Residuals of GMRES for Oseen problem (32×32, $\nu = 1$)

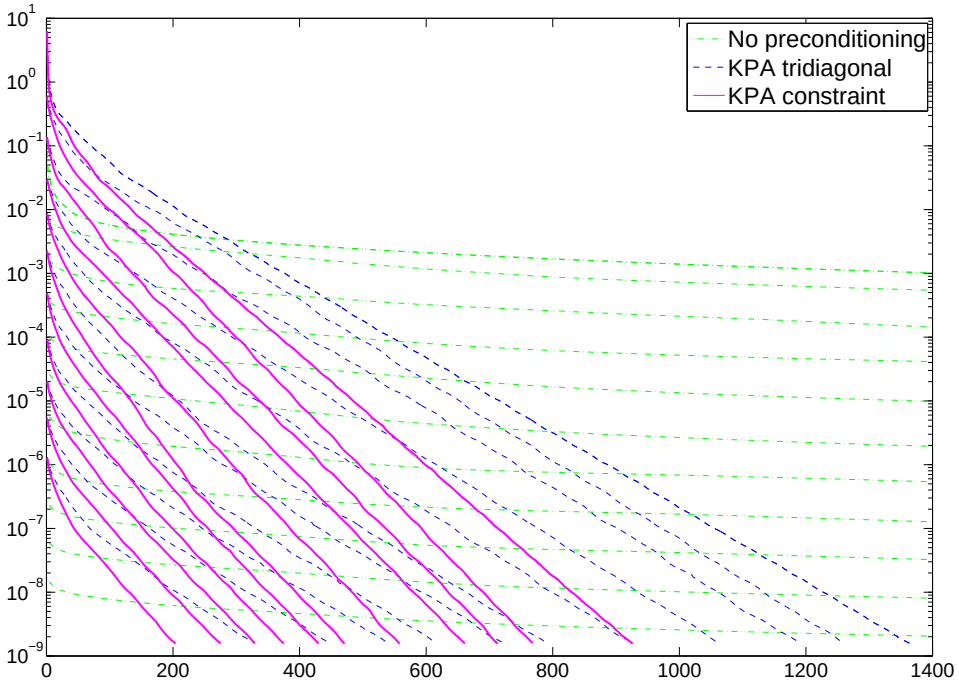
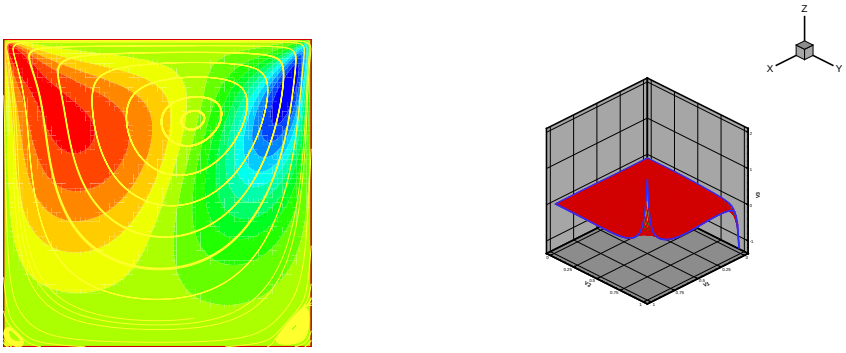


Figure 8: Residuals of GMRES(20) for Navier-Stokes problem (64×64, $\nu = 0.01$)

Picard iteration are illustrated in Figure 8. Without preconditioning, GMRES(20) needs more than 30000 iterations to converge for the linear systems corresponding to the first Picard iteration step. As the Picard iteration converges, the linear systems converge more quickly, but it still needs about 2000 iterations for the last Picard iteration step. Compared by the computation time, we find that GMRES(20) with tridiagonal preconditioner is 15 times faster than GMRES(20) without preconditioning, and GMRES(20) with constraint preconditioner is 24 times faster. And again the constraint preconditioner has better convergence behavior than the block tridiagonal preconditioner. With the constraint preconditioner, we obtain the converged velocity field and pressure field as shown in Figure 9.



(a) Streamlines and vertical velocity contour

(b) Pressure contour

Figure 9: Navier-Stokes problem (64×64, $\nu = 0.01$)

5 Conclusions

In this paper, we investigate the structure of the coefficient matrices of the convection-diffusion model problems, and construct the preconditioners based on the Kronecker product structure. This kind of problem specific preconditioner is efficient. The numerical tests on the scalar convection-diffusion problem, Stokes problem, Ossen problem, and Navier-Stokes problem show that the Kronecker product approximation (KPA) preconditoners accelerate the convergence. We also find that the constraint preconditioner can be regarded as the modification of the tridiagonal block preconditioner, and the tridiagonal block preconditioner is the modification of the diagonal block preconditioner based on the cell Reynolds number. This explains why the constraint preconditioner is usually better.

Our Kronecker product approximation (KPA) preconditioner has several advantages. Firstly, KPA preconditioner is easy to construct. We always operate on small matrices. We even do not need to form the coefficient matrix explicitly. Hence less flops are needed. Secondly, there is no trouble with fill-ins, and less storage is used. We only need to store X and Y, though $X \otimes Y$ is

dense. What's more, it is robust. The approximations X and Y are always nonsingular in our numerical tests, even though the original coefficient matrix is singular, while ILU often fails and cannot be used because the coefficient matrices arised from Stokes problem, Oseen problem and Navier-Stokes problem are rank-one deficient.

Though we only consider 2D case here, we also have the Kronecker product structure in the coefficient matrix in 3D case, but each term has the form $A \otimes B \otimes C$. For our problems, we can use other difference schemes, or apply it to non-uniform structured grids, but the expressions will not be as neat as uniform grids. Here we only consider the convection-diffusion model problems. For a more general case, we can use $A \approx \sum G_i \otimes F_i$ and DWT [11], where A comes from typical integral equations of potential theory.

References

[1] A. Ben-Israel, T. N. E. Greville, Generalized Inverses: Theory and Applications, 2nd Edition, Springer-Verlag, New York, 2003.

[2] M. Benzi, G. H. Golub and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1-137.

[3] M. Benzi, M. Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968-994.

[4] M. Benzi, M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30(1999), pp. 305-340.

[5] P. N. Brown, H. F. Walker, *GMRES on (nearly) singular systems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 37-51.

[6] Z. -H. Cao, *A note on constraint preconditioning for nonsymmetric indefnite matrices*, SIAM J. Matrix Anal. Appl. 24 (2002), pp. 121-125.

[7] Z. -H. Cao, *A class of constraint preconditioners for nonsymmetric saddle point matrices*, Numerische Mathematik, 103 (2006), pp. 47-61.

[8] Z. -H. Cao, *Comparison of performance of iterative methods for singular and nonsingular saddle point linear systems arising from Navier-Stokes equations*, Appl. Math. Comput., 174 (2006), pp. 630-642.

[9] Z. -H. Cao, *Augmentation block preconditioners for saddle point-type matrices with singular (1,1) blocks*, Numer. Linear Algebra Appl., to appear in 2008.

[10] H. Elman, *Preconditioning for the steady-state Navier-Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299-1316.

[11] J. M. Ford, E. E. Tyrtyshnikov, *Combining Kronecker product approximation with discrete wavelet transforms to solve dense, function-related linear systems*, SIAM J. Sci. Comput., 25 (2003), pp. 961-981.

[12] G. H. Golub and C. Van Loan, *Matrix Computations*, 3rd edition, Johns Hopkins University Press, Baltimore and London, 1996.

[13] G. H. Golub, C. Greif, J. M. Varah, *An algebraic analysis of a block diagonal preconditioner for saddle point systems*, SIAM. J. Matrix Anal. Appl. 27 (2005), pp. 779-792.

[14] N. I. M. Gould, M. E. Hribar, J. Nocedal, *On the solution of equality constrained quadratic programming problems arising in optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 1376-1395.

[15] C. Greif, D. Schötzau, *Preconditioners for saddle point linear systems with highly singular (1,1) blocks*, Electronic Transactions on Numerical Analysis, 22 (2006), pp. 141-121.

[16] M. Grote, T. Huckle, *Parallel preconditioning with sparse approximate inverse*, SIAM J. Sci. Comput., 18 (1997), pp. 838-853.

[17] C. Keller, N. I. M. Gould, A. J. Wathen, *Constraint preconditioning for indefnite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300-1317.

[18] A. Klawonn, *Block-triangular preconditioners for saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 172-184.

[19] L. Yu Kolotilina, A. Yu Yeremin, *Factorized sparse approximate inverse preconditionings*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45-58.

[20] A. N. Langville, W. J. Stewart, *A Kronecker product approximate preconditioner for SANs*, Numer. Lin. Algebra Appl., 11 (2004), pp. 723-752.

[21] J. Liesen, Z. Strakos, *GMRES convergence analysis for a convection-diffusion model problem*, SIAM J. Sci. Comput., 26 (2005), pp. 1989-2009.

[22] G. Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707-728.

[23] K. W. Morton, Numerical Solution of Convection-Diffusion Problems, Chapman & Hall, London, 1996.

[24] N. Pitsianis, C. F. Van Loan, Approximation with Kronecker products, in Linear Algebra for Large Scale and Real Time Applications, M. S. Moonen, G. H. Golub (eds), Kluwer Academics, Boston, 1993, pp. 293-314.

[25] A. Quarteroni, A. Valli, Numerical Approximation of Partial Differential Equations, Springer Ser. Comput. Math. 23, Springer-Verlag, Berlin, 1994.

[26] M. Rozloznik, V. Simoncini, *Krylov subspace methods for saddle point problems with indefnite preconditioning*, SIAM J. Matrix Anal. Appl., 24 (2004), pp. 368-391.

[27] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856-869.

[28] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994),pp. 387-402.

[29] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830-847.

[30] E. de Sturler, J. Liesen, *Block-diagonal and constraint preconditioners for nonsymmetric indefnite linear systems, Part I: Theory*, SIAM J. Sci. Comput., 26 (2005), pp. 1598-1619.

[31] G. Wang, Y. Wei and S. Qiao, *Generalized Inverses: Theory and Computations*, Science Press, Beijing, 2004.

[32] H. Xiang, Iterative methods and perturbation analysis of structured linear systems, PhD thesis (in Chinese), School of Mathematical Sciences, Fudan University, Shanghai, China, June 2006.

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399