Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2009, Article ID 282307, 13 pages
doi:10.1155/2009/282307

Research Article

A Twisted Block Tangential Filtering Decomposition Preconditioner

Ruirui Wang,[1] Qiang Niu,[2] and Linzhang Lu[2]

[1] *School of Mathematical Sciences, Xiamen University, Xiamen 361005, China*
[2] *School of Mathematics and Computer Science, Guizhou Normal University, Guiyang 550001, China*

Correspondence should be addressed to Linzhang Lu, lzlu@xmu.edu.cn

Received 25 December 2008; Accepted 24 February 2009

Recommended by Ben T. Nohara

For block-tridiagonal linear system of equations, a variant of tangential filtering preconditioners is proposed in this paper. The new variant is based on a twisted block factorization along with certain filtering property. For practical usage, a class of composite preconditioners tested, which are constructed by combining the twisted tangential filtering decomposition preconditioner with the classical $ILU(0)$ preconditioner in a multiplicative way. The performance of the new preconditioners is compared with other classical preconditioners; the superiority and the weakness of the preconditioners are pointed out.

Copyright © 2009 Ruirui Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

In many scientific and engineer applications, for example, simulation of laser propagation in a plasma [1] and study of transport in highly heterogeneous porous media [2], we have to numerically solve certain partial differential equations in 2D or 3D. The discretization of these PDEs by finite difference or finite volume schemes usually leads to large block-tridiagonal linear system:

$$Ax = b, \tag{1.1}$$

with

$$A = \begin{pmatrix} D_1 & U_1 & & \\ L_1 & D_2 & \ddots & \\ & \ddots & \ddots & U_{m-1} \\ & & L_{m-1} & D_m \end{pmatrix} \in \mathcal{R}^{n\times n}, \qquad b \in \mathcal{R}^n. \tag{1.2}$$

Due to the size of the problem, preconditioned Krylov iterative methods have become one of the most popular choices. It is generally recognized that the efficiency of the linear systems solver heavily depends on the property of the preconditioners. Therefore, the construction of robust and efficient preconditioners has become an interesting research topic. Several incomplete block factorization preconditioners have been proposed by many researchers, see, for example, Axelsson [3–5] and Meurant [6–8]. Frequency filtering preconditioners [9–16] advocated by G. Wittum and his successor are a special kind of incomplete block factorization preconditioner. This class of preconditioning techniques has been illustrated particularly efficient for linear systems arising from the discretization of partial differential equations with discontinuous coefficients.

With the development of techniques of parallel computing, developing high-performance preconditioners that are suitable for parallel computing environment is becoming an important topic. In this paper, we propose a tangential filtering preconditioner constructed by the framework of twisted block factorization. Firstly, the constructed preconditioner has a filtering property. Secondly, the construction and solving procedures of the twisted factorization preconditioner are carried out from two sides, which can be done in parallel. The performance of the newly built preconditioner is compared with the tangential filtering preconditioner proposed in [10]. For practical applications, Achdou and Nataf [10] propose to combine the tangential filtering preconditioner with the $ILU(0)$ preconditioner. In this paper, we also consider to combine the twisted tangential filtering preconditioner with the $ILU(0)$ preconditioner in the following way:

$$M_{ITF} = \left(M_{ILU}^{-1} + M_{TBTD}^{-1} - M_{ILU}^{-1} A M_{TBTD}^{-1}\right)^{-1}. \tag{1.3}$$

The performance of several different preconditioners is compared on some linear systems generated from the discretization of boundary value problems with discontinuous coefficients. The results show that the twisted block factorization preconditioner and its corresponding preconditioner output other preconditioners on some problems.

In Section 2, we give a brief introduction of twisted block tangential filtering decomposition and then introduce the twisted block tangential filtering decomposition preconditioner. In Section 3, we analyze the properties of the preconditioner. In Section 4, we give numerical experiments to compare the performances of different types of preconditioners.

2. A Twisted Block Tangential Filtering Decomposition

The block LDU factorization of A is

$$A = \begin{bmatrix} T_1 & & & \\ L_1 & T_2 & & \\ & \ddots & \ddots & \\ & & L_{m-1} & T_m \end{bmatrix} \begin{bmatrix} T_1^{-1} & & & \\ & T_2^{-1} & & \\ & & \ddots & \\ & & & T_m^{-1} \end{bmatrix} \begin{bmatrix} T_1 & U_1 & & \\ & T_2 & \ddots & \\ & & \ddots & U_{m-1} \\ & & & T_m \end{bmatrix}, \tag{2.1}$$

where T_i, $(i = 1, \ldots, m)$ are square invertible $n_i \times n_i$ matrices. Matrices L_{i-1} (resp. U_{i-1}) are $n_i \times n_{i-1}$ (resp. $n_{i-1} \times n_i$) matrices. The matrices T_i satisfy the induction formula

$$T_i = \begin{cases} D_1, & i = 1; \\ D_i - L_{i-1} T_{i-1}^{-1} U_{i-1}, & 2 \le i \le m. \end{cases} \tag{2.2}$$

The block LDU factorization can be written as

$$A = (L + T) T^{-1} (U + T). \tag{2.3}$$

Similar to the block LDU factorization, the block UDL factorization of A has the form

$$A = \left(U + \widehat{T}\right) \widehat{T}^{-1} \left(L + \widehat{T}\right), \tag{2.4}$$

where $\widehat{T}$ = Blockdiag $(\widehat{T}_1, \widehat{T}_2 \cdots \widehat{T}_m)$ with $\widehat{T}_i$ satisfies

$$\widehat{T}_i = \begin{cases} D_i - U_i \widehat{T}_{i+1}^{-1} L_i, & 1 \le i \le m-1, \\ D_m, & i = m. \end{cases} \tag{2.5}$$

Then the twisted block factorization can be written as

$$A = \left(E + \widetilde{T}\right) \widetilde{T}^{-1} \left(F + \widetilde{T}\right), \tag{2.6}$$

where

$$E = \begin{pmatrix} 0 & & & & & & \\ L_1 & 0 & & & & & \\ & \ddots & \ddots & & & & \\ & & L_{j-1} & 0 & U_j & & \\ & & & & \ddots & \ddots & \\ & & & & & 0 & U_{m-1} \\ & & & & & & 0 \end{pmatrix}, \qquad F = \begin{pmatrix} 0 & U_1 & & & & & \\ & 0 & \ddots & & & & \\ & & \ddots & U_{j-1} & & & \\ & & & 0 & & & \\ & & & L_j & \ddots & & \\ & & & & \ddots & 0 & \\ & & & & & L_{m-1} & 0 \end{pmatrix}. \tag{2.7}$$

The index j satisfies $1 < j < m$, the matrix $\widetilde{T} = \text{Blockdiag}(\widetilde{T}_1, \widetilde{T}_2 \cdots \widetilde{T}_m)$ with the diagonal block $\widetilde{T}_i$ satisfies the following relationship:

$$\widetilde{T}_i = \begin{cases} D_1, & i = 1 \\ D_i - L_{i-1}\widetilde{T}_{i-1}^{-1}U_{i-1}, & 2 \le i \le j-1, \\ D_j - L_{i-1}\widetilde{T}_{i-1}^{-1}U_{i-1} - U_i\widetilde{T}_{i+1}^{-1}L_i, & i = j, \\ D_i - U_i\widetilde{T}_{i+1}^{-1}L_i, & j+1 \le i \le m-1 \\ D_m, & i = m. \end{cases} \tag{2.8}$$

Different from diagonal block matrix D_i of A, the matrices $\widetilde{T}_i$ becomes dense quickly. Therefore, factorization (2.6) cannot be used for large problems in practice. However, the framework can be used to build an incomplete twisted block-factorization preconditioner for A. Precisely, we can replace the blocks $\widetilde{T}_i$ by suitably chosen sparse or block-sparse approximations $\overline{T}_i$, $(i = 1, 2, \ldots, m)$. Then an incomplete factorization preconditioner M is constructed, which has the following form:

$$M = \left(E + \overline{T}\right)\overline{T}^{-1}\left(F + \overline{T}\right), \tag{2.9}$$

with $\overline{T} = \text{Blockdiag}\ (\overline{T}_1, \overline{T}_2 \cdots \overline{T}_m)$.

From (2.9) it is easy to see that solving linear system $Mx = f$ is equivalent to solving the following two linear systems

$$\left(E + \overline{T}\right)y = f, \qquad \left(\overline{T}^{-1}F + I\right)x = y. \tag{2.10}$$

By exploiting the structure, both of the linear systems can be solved by the forward and backward sweeps. Suppose $y = (y_1^T, y_2^T, \ldots, y_m^T)^T$, $f = (f_1^T, f_2^T, \ldots, f_m^T)^T$, and $x = (x_1^T, x_2^T, \ldots, x_m^T)^T$ according to the block structure of $\overline{T}$. Then the process of solving $Mx = f$ can be described in Algorithm 1.

Remarks 2.1. Each of the solvers for $(E + \overline{T})y = f$ and $(\overline{T}^{-1}F + I)x = y$ described in Algorithm 1 involves forward and backward sweeps, and the two sweeps have no relationship with each other, so the forward and backward sweeps can be run in parallel.The procedure of constructing of $\overline{T}$ is consistent with the idea presented in [10]. Suppose we have approximation β_{i-1} of $\overline{T}_{i-1}^{-1}$ which satisfies

$$\|\overline{T}_{i-1}\beta_{i-1} - I\| \le \alpha < 1 \tag{2.11}$$

then

$$\|(\overline{T}_{i-1}\beta_{i-1} - I)^2\| \le \|(\overline{T}_{i-1}\beta_{i-1} - I)\|^2 \le \alpha^2, \tag{2.12}$$

```
(1) Solving $(E+\overline{T})y = f$
    $y_1 = \overline{T}_1^{-1} f_1,\ y_m = \overline{T}_m^{-1} f_m.$
    for $i = 2 : j-1$
        $y_i = \overline{T}_i^{-1}(f_i - L_{i-1}y_{i-1}),$
    end.
    for $i = m-1 : -1 : j+1$
        $y_i = \overline{T}_i^{-1}(f_i - U_i y_{i+1}),$
    end.
    $y_j = \overline{T}_j^{-1}(f_j - U_j y_{j+1} - L_{j-1}y_{j-1}).$
(2) Solving $(\overline{T}^{-1}F + I)x = y$
    $x_j = y_j,$
    for $i = j-1 : -1 : 1$
        $x_i = y_i - \overline{T}_i^{-1} U_i y_{i+1},$
    end
    for $i = j+1 : m$
        $x_i = y_i - \overline{T}_i^{-1} L_{i-1} y_{i-1},$
    end.
```

Algorithm 1: Solving **Mx** = **f**.

which implies

$$\left\|\overline{T}_{i-1}\left(2\beta_{i-1} - \beta_{i-1}\overline{T}_{i-1}\beta_{i-1}\right)\right\| \le \alpha^2. \tag{2.13}$$

It means that $2\beta_{i-1} - \beta_{i-1}\overline{T}_{i-1}\beta_{i-1}$ is a better approximation of $\overline{T}_{i-1}^{-1}$ than β_{i-1}. According to (2.8), we have the following formula for $\overline{T}$

$$\overline{T}_i = \begin{cases} D_1, & i = 1\ , \\ D_i - L_{i-1}\left(2\beta_{i-1} - \beta_{i-1}\overline{T}_{i-1}\beta_{i-1}\right)U_{i-1}, & 2 \le i \le j-1, \\ D_j - L_{i-1}\left(2\beta_{i-1} - \beta_{i-1}\overline{T}_{i-1}\beta_{i-1}\right)U_{i-1} - U_i\left(2\beta_{i+1} - \beta_{i+1}\overline{T}_{i+1}\beta_{i+1}\right)L_i, & i = j, \\ D_i - U_i\left(2\beta_{i+1} - \beta_{i+1}\overline{T}_{i+1}\beta_{i+1}\right)L_i, & j+1 \le i \le m-1, \\ D_m, & i = m. \end{cases} \tag{2.14}$$

Then the new block factorization preconditioner M based on the twisted factorization can be constructed by choosing β_{i-1} properly. Following the tangential filtering condition proposed in [10], a diagonal approximation β_{i-1} can be determined such that

$$(M - A)t = 0, \tag{2.15}$$

where t is a filtering vector.

Lemma 2.2. *If the matrices $\overline{T}_{i-1}(1 \leq i-1 \leq m)$ are invertible, then one has*

$$M - A = \textit{Blockdiag}\ (N_1, N_2, \ldots, N_m) \tag{2.16}$$

with

$$N_i = \begin{cases} 0, & i = 1, \\ L_{i-1}\left(\beta_{i-1}\overline{T}_{i-1} - I\right)\overline{T}_{i-1}^{-1}\left(\overline{T}_{i-1}\beta_{i-1} - I\right)U_{i-1}, & 2 \leq i \leq j-1, \\ L_{i-1}\left(\beta_{i-1}\overline{T}_{i-1} - I\right)\overline{T}_{i-1}^{-1}\left(\overline{T}_{i-1}\beta_{i-1} - I\right)U_{i-1} & \\ +U_i\left(\beta_{i+1}\overline{T}_{i+1} - I\right)\overline{T}_{i+1}^{-1}\left(\overline{T}_{i+1}\beta_{i+1} - I\right)L_i, & i = j, \\ U_i\left(\beta_{i+1}\overline{T}_{i+1} - I\right)\overline{T}_{i+1}^{-1}\left(\overline{T}_{i+1}\beta_{i+1} - I\right)L_i, & j+1 \leq i \leq m-1, \\ 0, & i = m. \end{cases} \tag{2.17}$$

Proof. Consider the matrix $M - A$ and observe that

$$2\beta_{i-1} - \beta_{i-1}\overline{T}_{i-1}\beta_{i-1} - \overline{T}_{i-1}^{-1} = -\left(\beta_{i-1}\overline{T}_{i-1} - I\right)\overline{T}_{i-1}^{-1}\left(\overline{T}_{i-1}\beta_{i-1} - I\right), \tag{2.18}$$

thus (2.17) holds. □

Now we consider how to form a diagonal matrix β_{i-1}. Let $t = (t_1^T, t_2^T, \cdots, t_m^T)^T$ be a given vector. If there are no zero entries in the vectors $U_{i-1}t_i(2 \leq i \leq j)$ and $L_i t_i(j \leq i \leq m-1)$, then it is possible to find diagonal matrices β_{i-1} such that M produces the same effect with A when operating on the filtering vector t, that is,

$$(M - A)t = 0. \tag{2.19}$$

From (2.17), we can see that it is sufficient to make

$$\left(\overline{T}_{i-1}\beta_{i-1} - I\right)U_{i-1}t_i = 0, \quad 2 \leq i \leq j, \tag{2.20}$$

$$\left(\overline{T}_{i+1}\beta_{i+1} - I\right)L_i t_i = 0, \quad j \leq i \leq m-1. \tag{2.21}$$

These requirements can be satisfied by setting β_{i-1} as follows:

$$\beta_{i-1} = \mathrm{Diag}\left(\overline{T}_{i-1}^{-1}U_{i-1}f_i./U_{i-1}t_i\right), \quad 2 \leq i \leq j\ , \tag{2.22}$$

$$\beta_{i+1} = \mathrm{Diag}\left(\overline{T}_{i+1}^{-1}L_i t_i./L_i t_i\right), \quad j \leq i \leq m-1, \tag{2.23}$$

where ./ designs the pointwise vector division, and Diag(v) is the diagonal matrix constructed from the vector v. We refer to the preconditioner constructed by the above procedure as twisted block tangential filtering decomposition preconditioner.

3. Analysis of the Twisted Tangential Filtering Preconditioner

In this section, we restrict A to be symmetric positive definite, and use $A \succ B$ ($A \succeq B$) to denote that $A - B$ is symmetric positive definite (semidefinite). Consider the twisted tangential filtering preconditioner M formed by (2.9), which ensures the filtering property (2.19). Furthermore, the following lemma holds and it has been established in [10].

Lemma 3.1. *If $A \succ 0$, then matrices $\overline{T}_i \succeq \widetilde{T}_i$, $1 \le i \le m$. Moreover, $M \succ 0$ and $M - A \succeq 0$ hold. The proof is similar to the proof of Lemma 2.1 of [10], so it is omitted here. From Lemma 3.1, one has the following result. The proof can be found in [17, 18].*

Theorem 3.2. *Let*

$$A = M - N, \tag{3.1}$$

be the splitting of coefficient matrix A induced by the twisted tangential filtering decomposition preconditioner M, then $\rho(M^{-1}N) < 1$.

4. Numerical Expriments

In this section, we present some numerical results to test the performance of preconditioners discussed in this paper. The performance of composite preconditioners is compared with M_{ILU}. Two kinds of approaches of constructing the filtering preconditioner M are considered. The combination approach (1.3) is used for all the composite preconditioners.

Consider the boundary value problem used in [10]

$$\begin{aligned} \eta(x)u + \operatorname{div}(\mathbf{a}(x)u) - \operatorname{div}(\kappa(x)\nabla u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega_D, \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on } \partial\Omega_N, \end{aligned} \tag{4.1}$$

where $\Omega = [0,1]^n$ ($n = 2$, or 3), $\partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$. The function η, the vector field $\mathbf{a}$, and the tensor κ are the given coefficients of the partial differential operator. In 2D case, we have $\partial\Omega_D = [0,1] \times \{0,1\}$, and in 3D case, we have $\partial\Omega_D = [0,1] \times \{0,1\} \times [0,1]$. Due to the discontinuous coefficients in the PDE equation and the size of A, an efficient preconditioner plays an important role in solving (4.1) by preconditioned iterative methods.

Several types of preconditioners tested in our numerical experiments, we outline the notations as follows,

M_{ILU}: the $ILU(0)$ preconditioner;

M_{TFFD}: the tangential frequency filtering decomposition preconditioner;

M_{TBTD}: the twisted block tangential filtering decomposition preconditioner;

M_{ITF}: the composite preconditioner generated by M_{ILU} and M_{TFFD};

M_{ITB}: the composite preconditioner generated by M_{ILU} and M_{TBTD}.

Two filtering vectors are tested, the Ritz vector of A used in [10], and $[1, 1, \ldots, 1]^T$ which is used as a filtering vector in [19]. The index j is set to be $j = \mathrm{m}/2$, where α denotes the largest integer not exceeding α.

Table 1: Results for Case 1: the advection-diffusion problem with a rotating velocity in two dimensions. Top resuls are using $\mathbf{t} = (\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1})^{\mathbf{T}}$, bottom resuls are using Ritz vector corresponding to the smallest eigenvalue of $\mathbf{A}$.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	60	$1.31e-10$	43	$6.50e-12$	44	$1.10e-11$	18	$4.18e-11$	16	$2.94e-11$
100	108	$5.40e-10$	63	$8.00e-12$	64	$1.69e-11$	26	$1.57e-10$	23	$7.52e-11$
200	186	$1.15e-9$	90	$1.10e-11$	91	$2.81e-11$	37	$5.67e-10$	32	$3.44e-10$
300	×	×	110	$1.50e-11$	111	$2.72e-11$	45	$6.88e-10$	39	$8.56e-10$
50	60	$1.31e-10$	43	$5.50e-12$	44	$1.26e-11$	18	$5.22e-11$	16	$3.43e-11$
100	108	$5.40e-10$	62	$9.30e-12$	64	$1.49e-11$	26	$1.82e-10$	23	$6.79e-11$
200	186	$1.15e-9$	90	$1.02e-11$	91	$3.05e-11$	37	$5.72e-10$	33	$1.57e-10$
300	×	×	110	$1.83e-11$	111	$3.00e-11$	45	$6.70e-10$	39	$7.10e-10$

The linear systems are solved by $FGMRES$ [20] method preconditioned by the previously mentioned preconditioners. The algorithm is unrestarted and the maximum Krylov subspace is set to be 200. For comparison reasons, the number of iterations of the $ILU(0)$ preconditioned $FGMRES$ method for constructing the filtering test-vector has been chosen to be 20. The algorithm is stopped whenever the relative norm $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\| / \|\mathbf{b}\|$ is less than 10^{-12}. The exact solution is generated randomly. In the following tables, $iters$ denotes the number of iterations, $error$ denotes the infinite norm of the difference between the final approximate solution and the exact solution. We use "×" to denote that the method fails to converge within 200 iterations. For preconditioners M_{ILU}, M_{TFFD}, and M_{TBTD}, every iteration requires only one preconditioner solve, so the total preconditioner solves are equal to the iteration number. For the composite preconditioners, assuming that the $ILU(0)$ preconditioner has the same cost with the filtering preconditioner, so the costs for composite preconditioner is twice of the iteration. All the experiments are performed in MATLAB [21]. The codes have not been optimized for the highest efficiency and therefore we do not report the time, but we outline the number of iterations.

The considered boundary value problems (4.1) are discretized on a regular Cartesian grid with a cell-centred finite volume scheme. Full up-winding is used for the convective term in the partial differential equation. The following five different cases are considered.

Case 1. The advection-diffusion problem with a rotating velocity in two dimensions.

The tensor κ is the identity, and the velocity is $\mathbf{a} = (2\pi(x_2 - 0.5),\ 2\pi(x_1 - 0.5))^T$. The function η is zero. The uniform grid with $n \times n$ nodes ($n = 50, 100, 200, 300$) are tested respectively. Table 1 displays the results obtained by using different preconditioners. The $ILU(0)$ preconditioner needs the most iterations to converges. The preconditioners M_{TFFD} and M_{TBTD} have better performances compared with $ILU(0)$. The composite preconditioners are more efficient and M_{ITB} works a little better than M_{ITF}. When changing the filtering vector, the iteration numbers have a small change, but the Ritz vector needs additional 20 steps to calculate. Figure 1 depicts the convergence curves of FGMRES method preconditioned by several different preconditioners. The filtering vector is set to be $t = (1, 1, \ldots, 1)^T$. We can see that the composite preconditioners are efficient, the FGMRES method preconditioned by M_{TFFD} and M_{TBTD} produces nearly the same convergence curve.

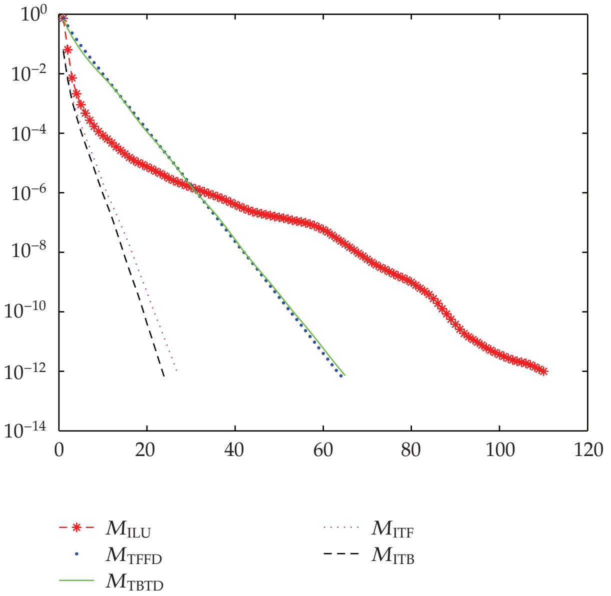

Figure 1: Convergence history for FGMRES when $1/h = 100$ for the advection-diffusion problem with a rotating velocity in two dimensions.

Case 2. Nonhomogenous problems with large jumps in the coefficients in two dimensions.

The coefficients η and $\mathbf{a}$ are both zero. The tensor κ is isotropic and discontinuous. It jumps from the constant value 10^3 in the ring $1/2\sqrt{2} \le |x-c| \le 1/2$, $c = (1/2, 1/2)^T$, to 1 outside. We tested uniform grids with $n \times n$ nodes, $n = 100, 200, 300, 400$. Table 2 displays the results obtained by using different preconditioners.

Case 3. Skyscraper problems.

The tensor κ is isotropic and discontinuous. The domain contains many zones of high permeability which are isolated from each other. Let $[x]$ denote the integer value of x. In 2D, we have

$$\kappa(x) = \begin{cases} 10^3 * ([10 * x_2] + 1), & \text{if } [10 * x_i] = 0 \mod (2),\ i = 1, 2, \\ 1, & \text{otherwise}, \end{cases} \tag{4.2}$$

and in 3D

$$\kappa(x) = \begin{cases} 10^3 * ([10 * x_2] + 1), & \text{if } [10 * x_i] = 0 \mod (2),\ i = 1, 2, 3, \\ 1, & \text{otherwise}. \end{cases} \tag{4.3}$$

The coefficients η and $\mathbf{a}$ are both zero. Tables 3 and 4 display the results obtained by using different preconditioners for 2D and 3D problems.

Table 2: Results for Case 2: Nonhomogenous problems with large jumps in the coefficients in two dimensions. Top results are using $\mathbf{t} = (\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1})^{\mathbf{T}}$ as filtering vector, bottom results are using Ritz vector corresponding to the smallest eigenvalue of **A**.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	61	6.62e − 11	43	5.67e − 12	44	1.07e − 11	18	2.27e − 11	16	9.90e − 11
100	108	4.71e − 10	62	6.91e − 12	64	2.23e − 11	26	1.43e − 10	23	1.01e − 10
200	187	1.54e − 9	89	9.23e − 12	92	2.14e − 11	37	3.10e − 10	33	1.92e − 10
300	×	×	109	1.81e − 11	112	3.33e − 11	45	6.06e − 10	40	7.31e − 10
50	61	6.62e − 11	43	5.44e − 12	45	8.40e − 12	18	3.14e − 11	16	3.29e − 11
100	108	4.71e − 10	62	9.16e − 12	64	1.85e − 11	26	1.07e − 10	23	1.08e − 10
200	187	1.54e − 9	89	1.03e − 11	92	1.75e − 11	37	2.73e − 10	33	1.76e − 10
300	×	×	109	1.41e − 11	112	3.28e − 11	45	5.94e − 10	40	4.80e − 10

Table 3: Results for Case 3: skyscraper problems. Top results are in 2D and bottom results are in 3D, $\mathbf{t} = (\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1})^{\mathbf{T}}$.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	152	2.01e − 7	×	×	×	×	16	4.20e − 8	15	3.93e − 8
100	×	×	×	×	×	×	26	2.63e − 7	24	4.86e − 7
200	×	×	×	×	×	×	39	9.50e − 7	37	1.90e − 6
300	×	×	×	×	×	×	47	2.77e − 6	44	2.64e − 6
10	19	4.57e − 12	16	2.40e − 12	17	1.86e − 12	8	7.08e − 13	7	1.66e − 11
15	190	1.07e − 8	×	×	×	×	13	9.91e − 9	25	8.92e − 9
20	132	2.57e − 8	×	×	×	×	11	5.13e − 9	20	6.21e − 9
30	×	×	×	×	×	×	14	6.91e − 8	13	8.84e − 8

Table 4: Results for Case 3: skyscraper problems. Top results are in 2D and bottom results are in 3D, **t** is set to be the Ritz vector corresponding to the smallest eigenvalue of **A**.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*i ters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	152	2.01e − 7	×	×	×	×	16	8.91e − 8	15	1.67e − 7
100	×	×	×	×	×	×	26	2.29e − 7	25	3.13e − 7
200	×	×	×	×	×	×	38	1.74e − 6	37	2.48e − 6
300	×	×	×	×	×	×	47	3.52e − 6	44	3.28e − 6
10	19	4.57e − 12	16	4.77e − 12	17	2.56e − 12	8	1.77e − 012	8	7.86e − 13
15	190	1.07e − 8	×	×	×	×	14	2.16e − 9	24	1.60e − 8
20	132	2.57e − 8	×	×	×	×	11	6.43e − 9	20	3.61e − 9
30	×	×	×	×	×	×	14	5.60e − 8	14	2.37e − 8

Case 4. Convective skyscraper problems.

The same happens with the Skyscraper problems except that the velocity field is changed to be $\mathbf{a} = (1000, 1000, 1000)^T$. The tested results are displayed in Tables 5 and 6.

Table 5: Results for Case 4: convective skyscraper problems. Top results are in 2D, bottom results are in 3D, $\mathbf{t} = (\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1})^{\mathbf{T}}$.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	95	$3.84e-9$	139	$3.43e-9$	×	×	13	$2.96e-10$	13	$2.92-10$
100	173	$2.81e-8$	×	×	×	×	18	$1.26e-9$	21	$5.08e-9$
200	×	×	×	×	×	×	26	$2.31e-9$	28	$3.05e-8$
300	×	×	×	×	×	×	27	$3.22e-8$	34	$8.32e-8$
10	13	$2.80e-9$	12	$1.24e-9$	12	$1.82e-9$	6	$4.65e-10$	5	$6.61e-9$
15	92	$3.03e-10$	54	$4.61e-10$	×	×	30	$8.45e-11$	17	$6.55e-11$
20	71	$2.60e-10$	34	$7.25e-10$	117	$1.67e-9$	9	$7.99e-12$	18	$9.51e-10$
30	116	$1.59e-9$	105	$2.15e-9$	×	×	33	$1.0e-10$	14	$1.30e-9$

Table 6: Results for Case 4: convective skyscraper problems. Top results are in 2D, bottom results are in 3D, $\mathbf{t}$ is set to be the ritz vector corresponding to the smallest eigenvalue of $\mathbf{A}$.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	95	$3.84e-9$	139	$3.43e-9$	×	×	13	$5.13e-10$	13	$4.99e-10$
100	173	$2.81e-8$	×	×	×	×	18	$1.67e-9$	21	$7.82e-9$
200	×	×	×	×	×	×	25	$4.41e-9$	28	$2.13e-8$
300	×	×	×	×	×	×	28	$1.85e-8$	34	$5.74e-8$
10	13	$2.80e-9$	12	$2.39e-9$	12	$3.83e-9$	6	$2.31e-10$	5	$4.56e-9$
15	92	$3.03e-10$	56	$3.17e-10$	×	×	29	$4.51e-10$	17	$4.83e-11$
20	71	$2.60e-10$	34	$6.41e-10$	121	$8.30e-10$	9	$2.78e-11$	18	$6.38e-10$
30	116	$1.59e-9$	108	$1.78e-9$	×	×	33	$3.19e-10$	15	$3.96e-10$

Table 7: Results for Case 5: anisotropic layers. Top results are in 2D, bottom results are in 3D, $\mathbf{t} = (\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1})^{\mathbf{T}}$.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	99	$5.88e-8$	53	$9.38e-9$	47	$1.61e-8$	11	$2.44e-8$	10	$3.78e-8$
100	190	$6.73e-7$	76	$6.22e-8$	72	$1.22e-8$	17	$6.23e-7$	16	$7.38e-8$
200	×	×	110	$3.30e-8$	103	$3.45e-8$	29	$2.30e-6$	27	$1.05e-7$
300	×	×	136	$7.95e-9$	127	$2.08e-8$	40	$1.04e-7$	37	$1.22e-7$
20	27	$1.43e-7$	24	$2.22e-8$	22	$2.66e-8$	10	$1.43e-8$	9	$1.22e-8$
30	34	$2.55e-7$	27	$3.31e-8$	26	$3.97e-8$	11	$4.87e-8$	10	$5.98e-8$
40	40	$8.14e-7$	28	$1.98e-8$	29	$4.97e-8$	11	$1.75e-7$	11	$7.12e-8$

Case 5. Anisotropic layers.

The domain is made of 10 anisotropic layers with jumps of up to four orders of magnitude and an anisotropy ratio of up to 10^3 in each layer. For 3D problem, the cube is divided in to 10 layers parallel to $z = 0$, of size 0.1, in which the coefficients are constant. The coefficient κ_x in the ith layer is given by $v(i)$, the latter being the ith component of the vector $v = [\alpha, \beta, \alpha, \beta, \alpha, \beta, \gamma, \alpha, \alpha]$, where $\alpha = 1$, $\beta = 10^2$ and $\gamma = 10^4$. We have $\kappa_y = 10\kappa_x$ and $\kappa_z = 1000\kappa_x$. The velocity field is zero. Numerical results are shown in Tables 7 and 8.

Table 8: Results for Case 5: anisotropic layers. Top results are in 2D, bottom results are in 3D, **t** is set to be the Ritz vector corresponding to the smallest eigenvalue of **A**.

	M_{ILU}		M_{TFFD}		M_{TBTD}		M_{ITF}		M_{ITB}	
$1/h$	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*	*iters*	*error*
50	99	$5.88e-8$	53	$4.91e-9$	48	$5.58e-9$	11	$1.85e-8$	10	$3.78e-8$
100	190	$6.73e-7$	77	$1.26e-8$	72	$1.53e-8$	18	$2.77e-7$	16	$7.84e-7$
200	×	×	110	$1.13e-8$	103	$3.03e-8$	29	$3.85e-8$	27	$8.38e-8$
300	×	×	136	$1.69e-8$	126	$1.55e-8$	40	$1.09e-7$	37	$7.40e-8$
20	27	$1.43e-7$	25	$1.54e-8$	22	$1.73e-8$	10	$4.75e-9$	9	$1.09e-8$
30	34	$2.55e-7$	27	$2.71e-8$	27	$2.65e-8$	11	$7.30e-8$	10	$5.71e-8$
40	40	$8.14e-7$	28	$2.48e-8$	29	$3.11e-8$	11	$2.82e-7$	11	$1.15e-7$

From the tests results presented in this paper, we can see that the composite preconditioners have better performance than using just a single preconditioner. The FGMRES method preconditioned by M_{TBTD} produces nearly the same results as by preconditioner M_{TFFD}, and also for the composite preconditioners M_{ITB} and M_{ITF}. However, the preconditioner proposed in this paper has the advantage of parallel computation. For Advection-diffusion and nonhomogeneous problems, there is a little difference between using *ones* or Ritz vector as the filtering vector. Considering the additional costs for Ritz vector, it is reasonable to use the *ones* as filtering vector.

5. Conclusion

In this paper, we introduce a new variant of tangential filtering decomposite preconditioner M_{TBTD}, which is based on the twisted factorization of the coefficient matrix A. The new one is comparable to the preconditioner M_{TFFD} presented in [10]. Considering the process of preconditioning with M_{TBTD} described in Algorithm 1, the preconditioner M_{TBTD} is superior to M_{TFFD} for its parallel property. And for the same reason, the composite preconditioner M_{ITB} surpasses M_{ITF}.

Acknowledgment

This work is supported by the National Natural Science Foundation of China nos. 10531080.

References

[1] S. Desroziers, F. Nataf, and R. Sentis, “Simulation of laser propagation in a plasma with a frequency wave equation,” *Journal of Computational Physics*, vol. 227, no. 4, pp. 2610–2625, 2008.

[2] B. Aksoylu and H. Klie, “A family of physics-based preconditioners for solving elliptic equations on highly heterogeneous media,” *Applied Numerical Mathematics*, vol. 59, no. 6, pp. 1159–1186, 2009.

[3] O. Axelsson and L. Kolotilina, “Diagonally compensated reduction and related preconditioning methods,” *Numerical Linear Algebra with Applications*, vol. 1, no. 2, pp. 155–177, 1994.

[4] O. Axelsson and H. Lu, “On eigenvalue estimates for block incomplete factorization methods,” *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 4, pp. 1074–1085, 1995.

[5] O. Axelsson and G. Lindskog, “On the eigenvalue distribution of a class of preconditioning methods,” *Numerische Mathematik*, vol. 48, no. 5, pp. 479–498, 1986.

[6] G. Meurant, *Computer Solution of Large Linear Systems*, vol. 28 of *Studies in Mathematics and Its Applications*, North-Holland, Amsterdam, The Netherlands, 1999.

[7] G. Meurant, “A review on the inverse of symmetric tridiagonal and block tridiagonal matrices,” *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 3, pp. 707–728, 1992.

[8] P. Concus, G. H. Golub, and G. Meurant, “Block preconditioning for the conjugate gradient method,” *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 1, pp. 220–252, 1985.

[9] Y. Achdou and F. Nataf, “An iterated tangential filtering decomposition,” *Numerical Linear Algebra with Applications*, vol. 10, no. 5-6, pp. 511–539, 2003.

[10] Y. Achdou and F. Nataf, “Low frequency tangential filtering decomposition,” *Numerical Linear Algebra with Applications*, vol. 14, no. 2, pp. 129–147, 2007.

[11] A. Buzdin, “Tangential decomposition,” *Computing*, vol. 61, no. 3, pp. 257–276, 1998.

[12] A. Buzdin and G. Wittum, “Two-frequency decomposition,” *Numerische Mathematik*, vol. 97, no. 2, pp. 269–295, 2004.

[13] C. Wagner, “Tangential frequency filtering decompositions for symmetric matrices,” *Numerische Mathematik*, vol. 78, no. 1, pp. 119–142, 1997.

[14] C. Wagner, “Tangential frequency filtering decompositions for unsymmetric matrices,” *Numerische Mathematik*, vol. 78, no. 1, pp. 143–163, 1997.

[15] C. Wagner and G. Wittum, “Adaptive filtering,” *Numerische Mathematik*, vol. 78, no. 2, pp. 305–328, 1997.

[16] G. Wittum, *Filternde Zerlegungen. Schnelle Löser für grosse Gleichungssysteme*, Teubner Skripten zur Numerik, B. G. Teubner, Stuttgart, Germany, 1992.

[17] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, vol. 95 of *Applied Mathematical Sciences*, Springer, New York, NY, USA, 1994.

[18] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

[19] L. Grigori, F. Nataf, and Q. Niu, “Two sides tangential filtering decomposition,” June 2008, http://hal.inria.fr/inria-00286595/fr.

[20] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, Mass, USA, 1996.

[21] The MathWorks, Inc., *MATLAB 7*, September 2004.