SIAM J. SCI. COMPUT.
Vol. 26, No. 3, pp. 844–863

BLOCK TRIANGULAR AND SKEW-HERMITIAN SPLITTING METHODS FOR POSITIVE-DEFINITE LINEAR SYSTEMS*

ZHONG-ZHI BAI†, GENE H. GOLUB‡, LIN-ZHANG LU§, AND JUN-FENG YIN†

Abstract. By further generalizing the concept of Hermitian (or normal) and skew-Hermitian splitting for a non-Hermitian and positive-definite matrix, we introduce a new splitting, called positive-definite and skew-Hermitian splitting (PSS), and then establish a class of PSS methods similar to the Hermitian (or normal) and skew-Hermitian splitting (HSS or NSS) method for iteratively solving the positive-definite systems of linear equations. Theoretical analysis shows that the PSS method converges unconditionally to the exact solution of the linear system, with the upper bound of its convergence factor dependent only on the spectrum of the positive-definite splitting matrix and independent of the spectrum of the skew-Hermitian splitting matrix as well as the eigenvectors of all matrices involved. When we specialize the PSS to block triangular (or triangular) and skew-Hermitian splitting (BTSS or TSS), the PSS method naturally leads to a BTSS or TSS iteration method, which may be more practical and efficient than the HSS and NSS iteration methods. Applications of the BTSS method to the linear systems of block two-by-two structures are discussed in detail. Numerical experiments further show the effectiveness of our new methods.

Key words. non-Hermitian matrix, positive-definite matrix, triangular matrix, block triangular matrix, Hermitian and skew-Hermitian splitting, splitting iteration method

AMS subject classifications. 65F10, 65F15, 65F50

DOI. 10.1137/S1064827503428114

1. Introduction.

We consider an iterative solution of the large sparse system of linear equations

$$Ax = b, \qquad A = (a_{k,j}) \in \mathbb{C}^{n\times n} \quad \text{nonsingular and} \quad b \in \mathbb{C}^n, \tag{1}$$

where $A \in \mathbb{C}^{n\times n}$ is a positive-definite complex matrix, which may be either Hermitian or non-Hermitian.

Denote by

$$H = \frac{1}{2}(A + A^*) \quad \text{and} \quad \widetilde{S} = \frac{1}{2}(A - A^*)$$

the Hermitian and skew-Hermitian parts of the matrix A, respectively. Then it immediately holds that

$$A = H + \widetilde{S},$$

*Received by the editors May 21, 2003; accepted for publication (in revised form) February 9, 2004; published electronically January 12, 2005.

http://www.siam.org/journals/sisc/26-3/42811.html

†State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, P.O. Box 2719, Beijing 100080, People's Republic of China (bzz@lsec.cc.ac.cn, yinjf@lsec.cc.ac.cn). The work of the first author was subsidized by The Special Funds for Major State Basic Research Projects G1999032803.

‡Department of Computer Science, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, CA 94305-9025 (golub@sccm.stanford.edu). The work of this author was supported in part by DOE-FC02-01ER4177.

§Department of Mathematics, Xiamen University, Xiamen 361005, People's Republic of China (lzlu@jingxian.xmu.edu.cn). The work of this author was supported by the National Natural Science Foundation of China.

which naturally leads to the *Hermitian and skew-Hermitian* splitting (HSS) of the matrix A [1, 7]. Based on the HSS and motivated by the classical *alternating direction implicit* (ADI) iteration technique [8], Bai, Golub, and Ng [1] recently presented the following HSS iteration method for solving the non-Hermitian positive-definite system of linear equations (1):

$$\begin{cases} (\alpha I + H)x^{(k+\frac{1}{2})} = (\alpha I - \widetilde{S})x^{(k)} + b, \\ (\alpha I + \widetilde{S})x^{(k+1)} = (\alpha I - H)x^{(k+\frac{1}{2})} + b, \end{cases}$$

where α is a given positive constant. Then, by further generalizing the HSS to the *normal and skew-Hermitian* splitting (NSS)

$$A = N + \widetilde{S}_o, \quad N \in \mathbb{C}^{n\times n} \quad \text{normal and} \quad \widetilde{S}_o \in \mathbb{C}^{n\times n} \quad \text{skew-Hermitian},$$

they obtained the NSS iteration method [2], which is analogous to the above HSS scheme, except that the splitting matrices H and $\widetilde{S}$ in the HSS iteration are replaced with N and $\widetilde{S}_o$, respectively.

It was demonstrated in [1, 2] that both HSS and NSS iteration methods converge unconditionally to the unique solution of the non-Hermitian positive-definite system of linear equations (1), with the bounds on their convergence being about the same as those of the conjugate gradient [9] (for HSS) and GMRES [11, 10] (for NSS) methods when they are applied to Hermitian and normal matrices, respectively. Moreover, the upper bounds of their contraction factors are dependent on only the spectrums of the Hermitian (for HSS) and the normal (for NSS) parts but are independent of the spectrums of the skew-Hermitian parts as well as the eigenvectors of all matrices involved. Numerical results show that both HSS and NSS iteration methods are very effective and robust when they are used to solve large sparse positive-definite systems of linear equations (1).

A noticeable property of this class of methods is that, by making use of matrix splittings, they split a *general* non-Hermitian positive-definite linear system into two *special* subsystems of linear equations which can be solved effectively by either direct methods (e.g., Cholesky factorization and Bunch decomposition [5, 6]) or iterative methods (e.g., Krylov subspace [10], multigrid, and multilevel methods). Hence, The HSS and NSS iteration techniques bridge the gap between the classical splitting iteration methods and the modern subspace projection and grid correction methods.

More generally, we know that any Hermitian or non-Hermitian positive-definite matrix $A \in \mathbb{C}^{n\times n}$ also possesses a splitting of the form

$$A = P + S, \quad P \in \mathbb{C}^{n\times n} \quad \text{positive-definite and} \quad S \in \mathbb{C}^{n\times n} \quad \text{skew-Hermitian}. \tag{2}$$

That is to say, A is of a *positive-definite and skew-Hermitian* (PS) splitting (2), or in short, PSS. Here, we say that $P \in \mathbb{C}^{n\times n}$ is positive definite if its Hermitian part, i.e., $\frac{1}{2}(P^* + P)$, is positive definite, and that $S \in \mathbb{C}^{n\times n}$ is skew-Hermitian if $S^* = -S$. By applying the technique of constructing HSS and NSS iterations, we can establish a class of PSS iteration methods for solving the positive-definite system of linear equations (1). Unlike both HSS and NSS methods, the new PSS method can be used to effectively compute iterative solutions of both Hermitian and non-Hermitian positive-definite linear systems. Theoretical analysis shows that the PSS iteration method preserves all properties of both HSS and NSS iteration methods. See Theorem 2.3 in section 2.

By specializing the PSS to *block triangular (or triangular) and skew-Hermitian* splittings (BTSS or TSS), the PSS method directly results in a class of BTSS or TSS iteration methods for solving the system of linear equations (1), which may be more practical and efficient than both HSS and NSS iteration methods because, for the BTSS (or TSS) method, we need only solve block triangular (or triangular) linear subsystems, rather than invert shifted positive-definite matrices, as in the PSS method, or shifted Hermitian (normal) positive-definite matrices as in the HSS (resp., NSS) method, at the first-half of each iteration step. In addition, applications of the BTSS method to linear systems of block two-by-two structures will be discussed in detail, and several numerical examples will be used to show the effectiveness of our new methods.

This paper is organized as follows. We establish the PSS iteration method and its convergence theory in section 2 and describe the BTSS iteration methods in section 3. Applications of the BTSS method to linear systems of block two-by-two structures are discussed in section 4, and numerical results are listed and analyzed in section 5. Finally, in section 6 we briefly give some concluding remarks.

2. PSS iteration method and its convergence. More precisely, the above-mentioned PSS iteration scheme based on the PSS (2) for solving the positive-definite system of linear equations (1) can be described as follows.

THE PSS ITERATION METHOD. Given an initial guess $x^{(0)} \in \mathbb{C}^n$, for $k = 0, 1, 2, \ldots$, until $\{x^{(k)}\}$ converges, compute

$$\begin{cases} (\alpha I + P)x^{(k+\frac{1}{2})} = (\alpha I - S)x^{(k)} + b, \\ (\alpha I + S)x^{(k+1)} = (\alpha I - P)x^{(k+\frac{1}{2})} + b, \end{cases}$$

where α is a given positive constant.

Evidently, just like the HSS and NSS methods, each iterate of the PSS iteration alternates between the positive-definite matrix P and the skew-Hermitian matrix S, analogously to the classical ADI iteration for partial differential equations [8, 12]. In fact, we can reverse the roles of the matrices P and S in the above PSS iteration so that we may first solve the linear system with coefficient matrix $\alpha I + S$ and then solve the linear system with coefficient matrix $\alpha I + P$.

We easily see that when $P \in \mathbb{C}^{n\times n}$ is normal or Hermitian, the above PSS iteration method reduces to the NSS or HSS iteration method accordingly.

In matrix-vector form, the PSS iteration can be equivalently rewritten as

$$x^{(k+1)} = M(\alpha)\, x^{(k)} + G(\alpha)\, b, \tag{3}$$

where

$$\begin{cases} M(\alpha) = (\alpha I + S)^{-1}(\alpha I - P)(\alpha I + P)^{-1}(\alpha I - S), \\ G(\alpha) = 2\alpha(\alpha I + S)^{-1}(\alpha I + P)^{-1}. \end{cases} \tag{4}$$

Thus, $M(\alpha)$ is the iteration matrix of the PSS iteration. As a matter of fact, (3) may also result from the splitting

$$A = B(\alpha) - C(\alpha)$$

of the coefficient matrix A, with

$$\begin{cases} B(\alpha) = \frac{1}{2\alpha}(\alpha I + P)(\alpha I + S) \equiv G(\alpha)^{-1}, \\ C(\alpha) = \frac{1}{2\alpha}(\alpha I - P)(\alpha I - S) \equiv G(\alpha)^{-1}M(\alpha). \end{cases}$$

To prove the convergence of the PSS iteration method, we first demonstrate the following two lemmas.

LEMMA 2.1. *Let*

$$V(\alpha) = (\alpha I - P)(\alpha I + P)^{-1}. \tag{5}$$

If $P \in \mathbb{C}^{n\times n}$ is a positive-definite matrix, then it holds that

$$\|V(\alpha)\|_2 < 1 \quad \forall \alpha > 0.$$

Proof. Because $P \in \mathbb{C}^{n\times n}$ is a positive-definite matrix (i.e., $\frac{1}{2}(P^* + P)$, the Hermitian part of P, is a Hermitian positive-definite matrix), we know that for any $y \in \mathbb{C}^n \setminus \{0\}$,

$$2\alpha\langle (P^* + P)y, y\rangle > 0 \quad \text{or} \quad \langle(\alpha P^* + \alpha P)y, y\rangle > \langle(-\alpha P^* - \alpha P)y, y\rangle$$

holds. Here $\langle\cdot,\cdot\rangle$ denotes the inner product in $\mathbb{C}^n$. It then straightforwardly follows that

$$\langle(\alpha^2 I + \alpha P^* + \alpha P + P^*P)y, y\rangle > \langle(\alpha^2 I - \alpha P^* - \alpha P + P^*P)y, y\rangle,$$

or equivalently,

$$\langle(\alpha I + P)y, (\alpha I + P)y\rangle > \langle(\alpha I - P)y, (\alpha I - P)y\rangle. \tag{6}$$

Let $x = (\alpha I + P)y$. Then we have $x \neq 0$ since $(\alpha I + P)$ is nonsingular and $y \neq 0$. Now, (6) can be equivalently written as

$$\langle x, x\rangle > \langle V(\alpha)x, V(\alpha)x\rangle.$$

That is to say, it holds that

$$\frac{\|V(\alpha)x\|_2}{\|x\|_2} < 1 \quad \forall \alpha > 0,$$

or in other words,

$$\|V(\alpha)\|_2 < 1 \quad \forall \alpha > 0. \qquad \square$$

LEMMA 2.2. *Let $S \in \mathbb{C}^{n\times n}$ be a skew-Hermitian matrix. Then for any $\alpha > 0$,*

(a) *$\alpha I + S$ is a non-Hermitian positive-definite matrix; and*

(b) *the Cayley transform $Q(\alpha) \equiv (\alpha I - S)(\alpha I + S)^{-1}$ of S is a unitary matrix.*

Proof. For the proof, see [1, 2]. $\square$

With Lemmas 2.1 and 2.2, we are now ready to prove convergence of the PSS iteration method.

THEOREM 2.3. *Let $A \in \mathbb{C}^{n\times n}$ be a positive-definite matrix, let $M(\alpha)$ defined in* (4) *be the iteration matrix of the PSS iteration, and let $V(\alpha)$ be the matrix defined in* (5)*. Then the spectral radius $\rho(M(\alpha))$ of $M(\alpha)$ is bounded by $\|V(\alpha)\|_2$. Therefore, it holds that*

$$\rho(M(\alpha)) \leq \|V(\alpha)\|_2 < 1 \quad \forall \alpha > 0;$$

i.e., the PSS iteration converges to the exact solution $x^\star \in \mathbb{C}^n$ of the system of linear equations (1)*.*

Proof. Because $S \in \mathbb{C}^{n\times n}$ is a skew-Hermitian matrix, from the definition of PSS (2) we know that $P^* + P = A^* + A$. Hence, $P \in \mathbb{C}^{n\times n}$ is also a positive-definite matrix. Moreover, by Lemma 2.2 we see that $Q(\alpha) = (\alpha I - S)(\alpha I + S)^{-1}$ is a unitary matrix.

Let

$$\widetilde{M}(\alpha) = V(\alpha)(\alpha I - S)(\alpha I + S)^{-1}.$$

Then $\widetilde{M}(\alpha)$ is similar to $M(\alpha)$. Therefore, by applying Lemma 2.1 we have

$$\begin{aligned}\rho(M(\alpha)) = \rho(\widetilde{M}(\alpha)) &\le \|\widetilde{M}(\alpha)\|_2 = \|V(\alpha)(\alpha I - S)(\alpha I + S)^{-1}\|_2 \\ &\le \|V(\alpha)\|_2\|Q(\alpha)\|_2 = \|V(\alpha)\|_2 < 1. \qquad \square\end{aligned}$$

Theorem 2.3 shows that the PSS iteration method converges unconditionally to the exact solution of the positive-definite system of linear equations (1). Moreover, the upper bound of its contraction factor is dependent on only the spectrum of the positive-definite part P but is independent of the spectrum of the skew-Hermitian part S as well as the eigenvectors of the matrices P, S, and A.

In particular, when the splitting matrix P is additionally assumed to be normal or Hermitian, Theorem 2.3 recovers the convergence theorem for the HSS and NSS iteration methods established in [1, 2], respectively.

We should point out that two important problems need to be further studied for the PSS iteration method. One is the choice of the skew-Hermitian matrix S, or the splitting of the matrix A, and another is the choice of the acceleration parameter α.

Theoretically, due to Theorem 2.3 we can choose S to be any skew-Hermitian matrix and α to be any positive constant. However, practically, in addition to the above requirements, we must choose S to be the skew-Hermitian matrix such that the linear systems with the coefficient matrices $\alpha I + P$ and $\alpha I + S$ can be solved easily and effectively, and must choose the positive constant α such that the PSS iteration converges very fast. Evidently, these two problems may be very difficult and, usually, their solutions strongly depend on the concrete structures and properties of the coefficient matrix A as well as the splitting matrices P and S.

For a positive-definite matrix $A \in \mathbb{C}^{n\times n}$ whether Hermitian or non-Hermitian, one useful choice of its PSS matrices P and S is as follows: Let $A = H + \widetilde{S}$ be the HSS of the matrix A, and let $H = D + L_H + L_H^*$ be the (block) triangular splitting of the Hermitian positive-definite matrix H, where D is the (block) diagonal matrix and L_H is the strictly (block) lower triangular matrix of H. Then we can define P and S to be

$$P = D + 2L_H \quad \text{and} \quad S = L_H^* - L_H + \widetilde{S},$$

respectively. This PS splitting directly leads to a special PSS iteration method for solving the system of linear equations (1), which only solves (block) lower triangular linear systems at the first half of each of its iteration steps (IT). Alternatively, we can also define P and S to be

$$P = D + 2L_H^* \quad \text{and} \quad S = L_H - L_H^* + \widetilde{S},$$

respectively, which immediately leads to a similar PSS iteration method.

In the next section, we will give two more practical choices of the PSS. These two special kinds of PSS are very basic. Technical combinations of them can yield

a variety of new positive-definite splittings, and hence, many practical PSS iteration methods.

According to the shifted skew-Hermitian linear subsystem

$$(\alpha I + S)x^{(k+1)} = (\alpha I + P)x^{(k+\frac{1}{2})} + b \tag{7}$$

involved in each iteration step of the PSS iteration method, we can solve it either exactly by the classical LU, QR, or Bunch decomposition [5, 6, 9], or inexactly by the modern Krylov subspace methods [10, 11]. We remark that the Krylov subspace iteration method of an optimality or Galerkin property, like GMRES, will possess a three-term recurrence form, as $(\alpha I + S)$ is now a normal matrix.

According to the positive constant α, if $P \in \mathbb{C}^{n\times n}$ is a normal matrix, then we can compute $\widehat{\alpha}^\star = \arg\min_{\alpha>0}\{\|V(\alpha)\|_2\}$ by making use of the formula in Theorem 2.2 of [2]; if $P \in \mathbb{C}^{n\times n}$ is a general positive-definite matrix, we do not have any formula to compute a usable $\widehat{\alpha}^\star$ and hence, the upper bound of $\|V(\widehat{\alpha}^\star)\|_2$. Usually, it holds that

$$\widehat{\alpha}^\star \neq \alpha_{\rm opt} \equiv \arg\min_{\alpha>0}\{\rho(M(\alpha))\}$$

and

$$\rho(M(\widehat{\alpha}^\star)) > \rho(M(\alpha_{\rm opt})).$$

Therefore, it is important to know how to compute an approximation of $\alpha_{\rm opt}$ as accurately as possible for improving the convergence speed of the method, and it is a hard task that needs further in-depth study from the viewpoint of both theory and computations.

3. BTSS iteration methods. Without loss of generality, we assume, in this section that the coefficient matrix $A \in \mathbb{C}^{n\times n}$ of the system of linear equations (1) is partitioned into the following block m-by-m system:

$$A = (A_{\ell,j})_{m\times m} \in \mathbb{C}^{n\times n}, \quad A_{\ell,j} \in \mathbb{C}^{n_\ell \times n_j}, \quad \ell, j = 1, 2, \ldots, m,$$

where n_ℓ, $\ell = 1, 2, \ldots, m$, are positive integers satisfying $\sum_{\ell=1}^{m} n_\ell = n$.

Let D, L, and U be the block diagonal, strictly block lower triangular, and strictly block upper triangular parts of the block matrix A, respectively. Then we have

$$\begin{aligned} A &= (L + D + U^*) + (U - U^*) \equiv T_1 + S_1 \\ &= (L^* + D + U) + (L - L^*) \equiv T_2 + S_2. \end{aligned} \tag{8}$$

Clearly, T_1 and T_2 are block lower triangular and block upper triangular matrices, respectively, and both S_1 and S_2 are skew-Hermitian matrices. We will call the two splittings in (8) BTSS of the matrix A. We remark that these two splittings are both PS splittings because $T_\ell + T_\ell^* = A + A^*$ $(\ell = 1, 2)$ and $A \in \mathbb{C}^{n\times n}$ is positive definite.

If we make technical combinations of the BTSS with the HSS or the NSS, other interesting and practical cases of the PS splitting can be obtained. For example,

$$\begin{aligned} A &= \left(L + \frac{1}{2}(D + D^*) + U^*\right) + \left(\frac{1}{2}(D - D^*) + U - U^*\right) \equiv T_3 + S_3 \\ &= \left(L^* + \frac{1}{2}(D + D^*) + U\right) + \left(\frac{1}{2}(D - D^*) + L - L^*\right) \equiv T_4 + S_4 \end{aligned} \tag{9}$$

are two BTSS, which come from combinations of BTSS of the matrix A in (8) and HSS of the matrix D.

Now, with the choices

$$P = T_\ell, \quad S = S_\ell, \qquad \ell = 1, 2, 3, 4,$$

we can immediately define the corresponding BTSS iteration methods for solving the positive-definite system of linear equations (1).

We note that for these four BTSS iteration methods, we need only solve block-triangular linear subsystems, rather than invert shifted positive-definite matrices, as in the PSS iteration method, or shifted Hermitian (normal) positive-definite matrices, as in the HSS (NSS) iteration method. Moreover, the block-triangular linear subsystems can be solved recursively through solutions of the systems of linear equations

$$(\alpha I + A_{j,j})x_j = r_j, \qquad j = 1, 2, \ldots, m, \tag{10}$$

for the BTSS iteration methods associated with the splittings in (8), and

$$\left(\alpha I + \frac{1}{2}(A_{j,j} + A_{j,j}^*)\right) x_j = r_j, \qquad j = 1, 2, \ldots, m, \tag{11}$$

for those associated with the splittings in (9). Because the splitting matrices T_ℓ ($\ell = 1, 2, 3, 4$) are positive definite, the block submatrices $A_{j,j}$ ($j = 1, 2, \ldots, m$) are also positive definite; in particular, $\frac{1}{2}(A_{j,j} + A_{j,j}^*)$ ($j = 1, 2, \ldots, m$) are all Hermitian positive-definite matrices. Therefore, we may employ another BTSS iteration to solve the linear subsystems (10) and the conjugate gradient iteration to solve the linear subsystems (11) if necessary. In addition, the matrices T_ℓ, $\ell = 1, 2, 3, 4$, may be much more sparse than the matrices H and N in the HSS and NSS methods. For instance, when the matrix A is an upper Hessenberg matrix, T_ℓ and S_ℓ, $\ell = 1, 2, 3, 4$, in the BTSS (or TSS) splittings are still very sparse, but H, $\widetilde{S}$ and N, $\widetilde{S}_o$ in the HSS and NSS may be very dense. Therefore, the BTSS iteration methods may save considerably more computing costs than both HSS and NSS iteration methods. Another advantage of the BTSS iteration methods is that they can be used to solve both Hermitian and strongly non-Hermitian positive-definite systems of linear equations more effectively than both HSS and NSS iteration methods. For example, consider the non-Hermitian positive-definite system of linear equations

$$(\alpha I + \widehat{S})z = r$$

arising from HSS, NSS, and TSS iteration methods, where $\widehat{S} \in \mathbb{C}^{n\times n}$ is a skew-Hermitian matrix, α is a positive constant, and $r \in \mathbb{C}^n$ is a given right-hand-side vector. Both HSS and NSS iteration methods cannot be used to solve it; however, the BTSS iteration method may solve it very efficiently. This shows that the BTSS iteration methods have a large application area.

When D, L, and U are the (pointwise) diagonal, the (pointwise) strictly lower triangular, and the (pointwise) strictly upper triangular parts of the matrix A, we call the BTSS a *triangular and skew-Hermitian* splitting (TSS) and the BTSS iteration method a TSS iteration method.

We remark that both BTSS and TSS iteration methods are, in general, different from the HSS and NSS iteration methods. Only when D is Hermitian (normal) and $L + U^* = 0$ do the BTSS and TSS methods give the same scheme as the HSS (resp., NSS) method.

The following investigation describes formulae for approximately estimating the acceleration parameters α for the BTSS iteration methods.

For $\ell = 1, 2, 3, 4$, let $T_\ell = D_\ell + G_\ell$, where

$$D_\ell = \begin{cases} D & \text{for} \quad \ell = 1, 2, \\ \frac{1}{2}(D + D^*) & \text{for} \quad \ell = 3, 4, \end{cases} \quad \text{and} \quad G_\ell = \begin{cases} L + U^* & \text{for} \quad \ell = 1, 3, \\ L^* + U & \text{for} \quad \ell = 2, 4. \end{cases}$$

Obviously, $D_\ell (\ell = 1, 2, 3, 4)$ are block diagonal matrices and $G_\ell (\ell = 1, 2, 3, 4)$ are strictly block lower (for $\ell = 1, 3$) or upper (for $\ell = 2, 4$) triangular matrices. Therefore,

$$\begin{aligned} (\alpha I + T_\ell)^{-1} &= ((\alpha I + D_\ell) + G_\ell)^{-1} \\ &= \left[\left(I + G_\ell(\alpha I + D_\ell)^{-1}\right)(\alpha I + D_\ell)\right]^{-1} \\ &= \left[(\alpha I + D_\ell)\left(I + (\alpha I + D_\ell)^{-1} G_\ell\right)\right]^{-1} \\ &= (\alpha I + D_\ell)^{-1} \sum_{j=0}^{m-1} (-1)^j \left[G_\ell(\alpha I + D_\ell)^{-1}\right]^j \\ &= \sum_{j=0}^{m-1} (-1)^j \left[(\alpha I + D_\ell)^{-1} G_\ell\right]^j (\alpha I + D_\ell)^{-1}. \end{aligned}$$

Here, the last two equalities hold since

$$\left[G_\ell(\alpha I + D_\ell)^{-1}\right]^m = \left[(\alpha I + D_\ell)^{-1} G_\ell\right]^m = 0.$$

It then follows that

$$\begin{aligned} V_\ell(\alpha) &\equiv (\alpha I - T_\ell)(\alpha I + T_\ell)^{-1} \\ &= I - 2T_\ell(\alpha I + T_\ell)^{-1} \\ &= I - 2(D_\ell + G_\ell)(\alpha I + T_\ell)^{-1} \\ &= (\alpha I - D_\ell)(\alpha I + D_\ell)^{-1} + 2\alpha(\alpha I + D_\ell)^{-1} \sum_{j=1}^{m-1} (-1)^j \left[G_\ell(\alpha I + D_\ell)^{-1}\right]^j \\ &\approx (\alpha I - D_\ell)(\alpha I + D_\ell)^{-1} \qquad \text{(a first-order approximation)} \end{aligned}$$

and

$$\begin{aligned} \|V_\ell(\alpha)\|_2 &\approx \|(\alpha I - D_\ell)(\alpha I + D_\ell)^{-1}\|_2 \\ &= \begin{cases} \max\limits_{1 \le j \le m} \left\{ \left\| (\alpha I - A_{j,j})(\alpha I + A_{j,j})^{-1} \right\|_2 \right\} & \text{for } \ell = 1, 2, \\ \max\limits_{1 \le j \le m} \left\{ \left\| \left(\alpha I - \frac{1}{2}(A_{j,j} + A_{j,j}^*)\right) \left(\alpha I + \frac{1}{2}(A_{j,j} + A_{j,j}^*)\right)^{-1} \right\|_2 \right\} & \text{for } \ell = 3, 4. \end{cases} \end{aligned}$$

For $\ell = 3, 4$, because $\frac{1}{2}(A_{j,j} + A_{j,j}^*)$ $(j = 1, 2, \ldots, m)$ are Hermitian matrices we immediately have

$$\begin{aligned} \widehat{\alpha}^\star &= \arg\min_{\alpha > 0} \|V_\ell(\alpha)\|_2 \qquad (\ell = 3, 4) \\ &\approx \arg\min_{\alpha > 0} \max_{1 \le j \le m} \left\{ \left\| \left(\alpha I - \frac{1}{2}(A_{j,j} + A_{j,j}^*)\right) \left(\alpha I + \frac{1}{2}(A_{j,j} + A_{j,j}^*)\right)^{-1} \right\|_2 \right\} \\ &= \arg\min_{\alpha > 0} \max_{1 \le j \le m} \max_{1 \le k \le n_j} \left\{ \left| \frac{\alpha - \lambda_k^{(j)}}{\alpha + \lambda_k^{(j)}} \right| \right\} \\ &= \sqrt{\lambda_{\min}^{(j_o)} \lambda_{\max}^{(j_o)}}, \end{aligned}$$

where $\lambda_k^{(j)}$ $(k = 1, 2, \ldots, n_j)$ are eigenvalues of the matrix $\frac{1}{2}(A_{j,j} + A_{j,j}^*)$, $\lambda_{\min}^{(j_o)}$ and $\lambda_{\max}^{(j_o)}$ are the smallest and the largest eigenvalues of the matrix $\frac{1}{2}(A_{j,j} + A_{j,j}^*)$, $j_o = \arg\max_{1\le j\le m} \kappa(\frac{1}{2}(A_{j,j} + A_{j,j}^*))$, and $\kappa(\cdot)$ denotes the spectral condition number of the corresponding matrix.

In particular, when $m = n$ and $A_{j,j}$ $(j = 1, 2, \ldots, m)$ are entries, we have for $\ell = 1, 2, 3, 4$ that

$$\widehat{\alpha}^\star = \arg\min_{\alpha>0} \max_{1\le j\le n} \left|\frac{\alpha - a_{j,j}}{\alpha + a_{j,j}}\right| = \sqrt{a_{\min} a_{\max}},$$

where

$$a_{\min} = \min_{1\le j\le n} \{a_{j,j}\}, \quad a_{\max} = \max_{1\le j\le n} \{a_{j,j}\}.$$

4. Applications. We now give applications of the BTSS iteration methods to the systems of linear equations (1) whose coefficient matrices possess the block two-by-two structure

$$A = \begin{bmatrix} W & F \\ E & N \end{bmatrix},$$

where $W \in \mathbb{C}^{q\times q}$ and $N \in \mathbb{C}^{(n-q)\times(n-q)}$ are positive-definite complex submatrices such that $A \in \mathbb{C}^{n\times n}$ is a positive-definite matrix. We may further assume that both W and N are Hermitian; otherwise we can consider the BTSS iteration methods to be induced by the BTSS in (9).

According to (8) and (9) we know that the BTSS of the block two-by-two matrix $A \in \mathbb{C}^{n\times n}$ are

$$\begin{aligned}
A &= \begin{bmatrix} W & 0 \\ E + F^* & N \end{bmatrix} + \begin{bmatrix} 0 & F \\ -F^* & 0 \end{bmatrix} \equiv T_1 + S_1 \\
&= \begin{bmatrix} W & E^* + F \\ 0 & N \end{bmatrix} + \begin{bmatrix} 0 & -E^* \\ E & 0 \end{bmatrix} \equiv T_2 + S_2 \\
&= \begin{bmatrix} \frac{1}{2}(W + W^*) & 0 \\ E + F^* & \frac{1}{2}(N + N^*) \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(W - W^*) & F \\ -F^* & \frac{1}{2}(N - N^*) \end{bmatrix} \equiv T_3 + S_3 \\
&= \begin{bmatrix} \frac{1}{2}(W + W^*) & E^* + F \\ 0 & \frac{1}{2}(N + N^*) \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(W - W^*) & -E^* \\ E & \frac{1}{2}(N - N^*) \end{bmatrix} \equiv T_4 + S_4.
\end{aligned}$$

Given an initial guess $x^{(0)} \in \mathbb{C}^n$, the BTSS iterations compute sequences $\{x^{(k)}\}$ as follows:

$$\begin{cases} (\alpha I + T_\ell)x^{(k+\frac{1}{2})} = (\alpha I - S_\ell)x^{(k)} + b, \\ (\alpha I + S_\ell)x^{(k+1)} = (\alpha I - T_\ell)x^{(k+\frac{1}{2})} + b, \end{cases} \qquad \ell = 1, 2, 3, 4.$$

As an example, in the following we investigate only the BTSS iteration for the case $\ell = 1$, because the other three cases can be discussed analogously. Moreover, without loss of generality, we could assume that both W and N are Hermitian if necessary; otherwise we can consider the BTSS iteration methods to be induced by the BTSS for $\ell = 3, 4$.

Let x and b be correspondingly partitioned into blocks as

$$x = \begin{bmatrix} u \\ p \end{bmatrix}, \qquad b = \begin{bmatrix} f \\ g \end{bmatrix},$$

where $u, f \in \mathbb{C}^q$ and $p, g \in \mathbb{C}^{n-q}$. The first half-step of the BTSS iteration requires the solution of linear systems of the form

$$(\alpha I + W)u^{(k+\frac{1}{2})} = \alpha u^{(k)} + f - Fp^{(k)}. \tag{12}$$

Once the solution $u^{(k+\frac{1}{2})}$ of (12) has been obtained, we compute

$$(\alpha I + N)p^{(k+\frac{1}{2})} = -(E + F^*)u^{(k+\frac{1}{2})} + \alpha p^{(k)} + g + F^* u^{(k)}. \tag{13}$$

Note that the coefficient matrices in (12) and (13) are positive definite (or Hermitian positive definite, respectively), if the matrices W and N are. Therefore, the linear systems (12) and (13) can be solved by any algorithm for positive-definite systems (e.g., a PSS method) when the matrices W and N are positive definite, or by any algorithm for Hermitian positive-definite systems (e.g., a sparse Cholesky factorization or the conjugate gradient method [9]) when the matrices W and N are Hermitian positive definite.

The second half-step of the BTSS iteration requires the solution of linear systems of the form

$$\begin{cases} \alpha u^{(k+1)} + Fp^{(k+1)} = (\alpha I - W)u^{(k+\frac{1}{2})} + f, \\ -F^* u^{(k+1)} + \alpha p^{(k+1)} = -(E + F^*)u^{(k+\frac{1}{2})} + (\alpha I - N)p^{(k+\frac{1}{2})} + g. \end{cases} \tag{14}$$

This linear system can be solved in various ways, including the conjugate-gradient-like method discussed in [7, 10] and the BTSS (or TSS) method. Additionally, when $n \leq 2q$, we may first solve the Hermitian positive-definite system of linear equations

$$(\alpha^2 I + F^* F)p^{(k+1)} = -(\alpha E + F^* W)u^{(k+\frac{1}{2})} + \alpha(\alpha I - N)p^{(k+\frac{1}{2})} + F^* f + \alpha g,$$

and then compute

$$u^{(k+1)} = \frac{1}{\alpha}(-Fp^{(k+1)} + (\alpha I - W)u^{(k+\frac{1}{2})} + f),$$

and when $n \geq 2q$, we may first solve the Hermitian positive-definite system of linear equations

$$\begin{aligned}(\alpha^2 I + FF^*)u^{(k+1)} = {} & (\alpha(\alpha I - W) + F(E + F^*))u^{(k+\frac{1}{2})} \\ & -F(\alpha I - N)p^{(k+\frac{1}{2})} + \alpha f - Fg,\end{aligned}$$

and then compute

$$p^{(k+1)} = \frac{1}{\alpha}(F^* u^{(k+1)} - (E + F^*)u^{(k+\frac{1}{2})} + (\alpha I - N)p^{(k+\frac{1}{2})} + g).$$

We remark that, unlike the HSS and PHSS[1] iteration methods in [3, 4], the BTSS iteration methods are divergent for any $\alpha > 0$ when they are applied to the saddle-point problem

$$Ax \equiv \begin{bmatrix} W & F \\ F^* & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \equiv b,$$

[1] Preconditioned Hermitian and skew-Hermitian splitting.

where $W \in \mathbb{C}^{q\times q}$ is Hermitian positive definite, $F \in \mathbb{C}^{q\times(n-q)}$ is of full column rank, and $2q \geq n$. More concretely, when $W = I$, following a derivation analogous to that in [3], we obtain that the eigenvalues of the BTSS iteration matrix

$$M_1(\alpha) = (\alpha I + S_1)^{-1}(\alpha I - T_1)(\alpha I + T_1)^{-1}(\alpha I - S_1)$$

are $\frac{\alpha-1}{\alpha+1}$ with multiplicity $2q - n$ and

$$\frac{1}{(\alpha+1)(\alpha^2+\sigma_k^2)}(\alpha(\alpha^2+3\sigma_k^2) \pm \sqrt{(\alpha^2+\sigma_k^2)^2 + 4\alpha^2\sigma_k^2(\alpha^2+2\sigma_k^2)}),\ k = 1, 2, \ldots, n-q,$$

where $\sigma_k (k = 1, 2, \ldots, n - q)$ are the positive singular values of the matrix F. Therefore,

$$\rho(M_1(\alpha)) = \max_{1\leq k\leq n-q}\left\{\frac{\alpha(\alpha^2+3\sigma_k^2) + \sqrt{(\alpha^2+\sigma_k^2)^2 + 4\alpha^2\sigma_k^2(\alpha^2+2\sigma_k^2)}}{(\alpha+1)(\alpha^2+\sigma_k^2)}\right\} > 1$$

for any $\alpha > 0$. However, we see that all eigenvalues of the iteration matrix $M_1(\alpha)$ are real, and the eigenvalues $\frac{\alpha-1}{\alpha+1}$ and

$$\frac{1}{(\alpha+1)(\alpha^2+\sigma_k^2)}(\alpha(\alpha^2+3\sigma_k^2) - \sqrt{(\alpha^2+\sigma_k^2)^2 + 4\alpha^2\sigma_k^2(\alpha^2+2\sigma_k^2)}),\ k = 1, 2, \ldots, n-q,$$

are within the interval $(-1, 1)$, while the other eigenvalues

$$\frac{1}{(\alpha+1)(\alpha^2+\sigma_k^2)}(\alpha(\alpha^2+3\sigma_k^2) + \sqrt{(\alpha^2+\sigma_k^2)^2 + 4\alpha^2\sigma_k^2(\alpha^2+2\sigma_k^2)}),\ k = 1, 2, \ldots, n-q,$$

are within the interval $(1, +\infty)$.

5. Numerical experiments. We use two examples to numerically examine feasibility and effectiveness of our new methods.

Without special claim (i.e., if we do not introduce new definitions), all our tests are started from random vectors, performed in MATLAB with machine precision 10^{-16}, and terminated when the current iterate satisfies $\|r^{(k)}\|_2/\|r^{(0)}\|_2 < 10^{-5}$, where $r^{(k)}$ is the residual of the current, say kth, iteration. The right-hand-side vector b is computed from $b = Ax^\star$, where $x^\star$ is the exact solution of the system of linear equations (1), and is a randomly generated vector.

Example 5.1 (see [1]). Consider the system of linear equations (1), for which $A \in \mathbb{R}^{n\times n}$ is the upwind difference matrix of the two-dimensional convection-diffusion equation

$$-(u_{xx} + u_{yy}) + q \cdot \exp(x + y)(xu_x + yu_y) = f(x, y)$$

on the unit square $\Omega = [0, 1] \times [0, 1]$ with the homogeneous Dirichlet boundary conditions. The stepsizes along both x and y directions are the same, i.e., $h = \frac{1}{m+1}$.

In our computations, we use the TSS iteration, resulting from the TSS $A = T_1 + S_1$ in (8) as the test method, and solve the shifted skew-Hermitian linear subsystem (7) at each TSS iteration step by the Gaussian elimination method.

In Figures 1 and 2, we plot the eigenvalues of the iteration matrices of both HSS and TSS methods when $q = 1$ and $m = 24,\ 32$, respectively. It is clear that the eigenvalue distributions of the HSS and TSS iteration matrices are quite different: the eigenvalues of the HSS iteration matrix are tightly clustered around the real axis

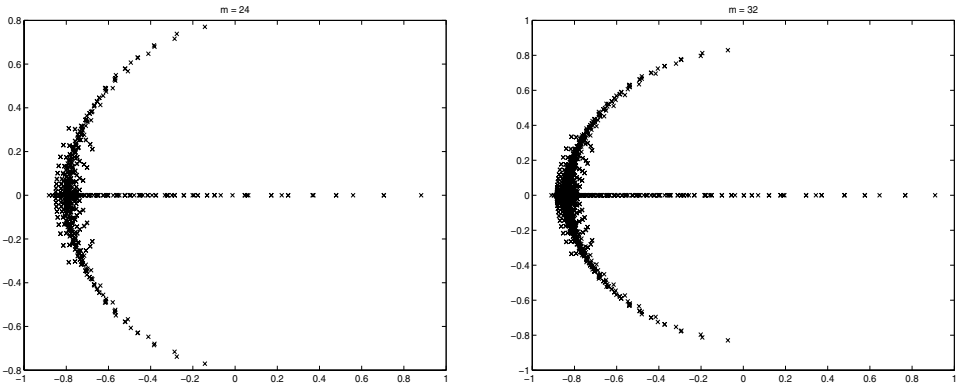

FIG. 1. *The eigenvalue distributions of the HSS iteration matrices when* $q = 1$, *and* $m = 24$ *(left) and* $m = 32$ *(right), for Example* 5.1.

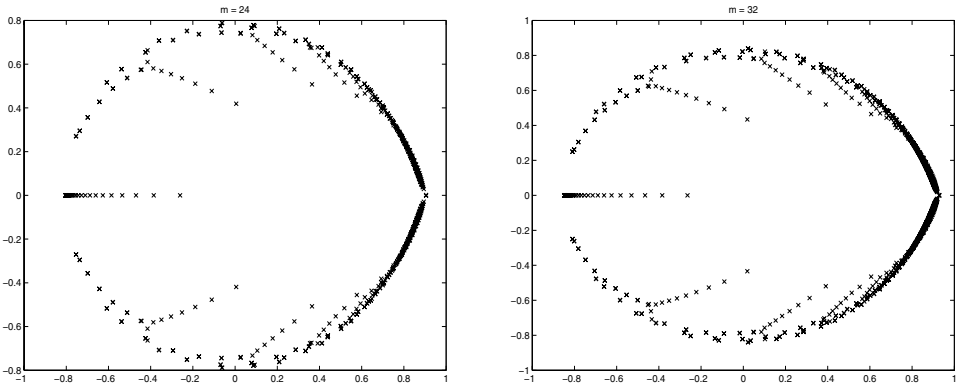

FIG. 2. *The eigenvalue distributions of the TSS iteration matrices when* $q = 1$, *and* $m = 24$ *(left) and* $m = 32$ *(right), for Example* 5.1.

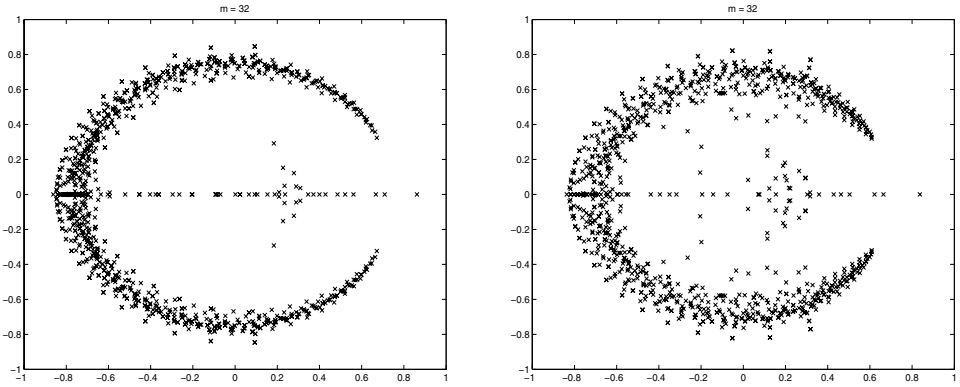

FIG. 3. *The eigenvalue distributions of the HSS iteration matrices when* $m = 32$, *and* $q = 6$ *(left) and* $q = 9$ *(right), for Example* 5.1.

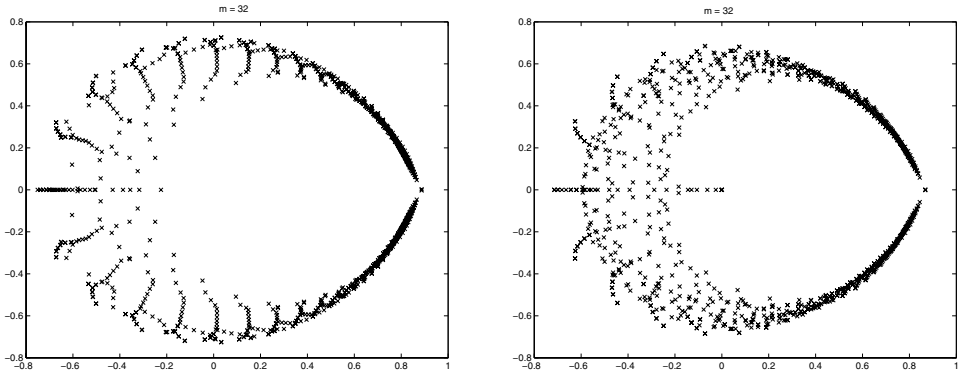

FIG. 4. *The eigenvalue distributions of the TSS iteration matrices when* $m = 32$, *and* $q = 6$ *(left) and* $q = 9$ *(right), for Example* 5.1.

TABLE 1
$\alpha_{\exp}$ *versus* $\rho(M(\alpha_{\exp}))$ *for Example* 5.1 *(*$q = 1$*).*

m		8	16	24	32	64
TSS	$\alpha_{\exp}$	1.118	0.619	0.424	0.322	0.163
	$\rho(M(\alpha_{\exp}))$	0.723	0.858	0.905	0.929	0.964
HSS	$\alpha_{\exp}$	1.054	0.595	0.413	0.316	0.163
	$\rho(M(\alpha_{\exp}))$	0.706	0.837	0.882	0.909	0.953

TABLE 2
IT and CPU for Example 5.1 *(*$q = 1$*).*

m		8	16	24	32	64
TSS	IT	31	56	83	113	234
	CPU	0.009	0.453	4.78	48.095	75.948
HSS	IT	24	51	82	108	214
	CPU	0.019	0.825	17.954	87.906	115.606
Speed-up		2.11	1.82	3.76	1.83	1.52

and a circular arc on the complex plane, while those of the TSS iteration matrix are clustered closely around the real axis and a circle; however, the distribution domains of the eigenvalues of both iteration matrices are very similar. This observation is further confirmed by Figures 3 and 4 for which $m = 32$ and $q = 6,\ 9$, respectively. In particular, from these figures we can see that when q becomes larger, the shapes and domains of the eigenvalue distributions of the HSS and TSS iteration matrices become more similar.

In Table 1, we list the experimentally optimal parameters $\alpha_{\exp}$ and the corresponding spectral radii $\rho(M(\alpha_{\exp}))$ of the iteration matrices $M(\alpha_{\exp})$ of both TSS and HSS methods for several m when $q = 1$. The data show that when m is increasing, $\alpha_{\exp}$ is decreasing, while $\rho(M(\alpha_{\exp}))$ is increasing, for both TSS and HSS methods. $\alpha_{\exp}$ and $\rho(M(\alpha_{\exp}))$ of TSS are larger than those of HSS, correspondingly, for all of our tested m. This straightforwardly implies that *the number of IT* of TSS may be larger than that of HSS. *However, because TSS has a smaller computational workload than HSS at each of the IT, the actual* computing time *(CPU) of TSS may be less than that of HSS.* These facts are confirmed by the numerical results in Table 2.

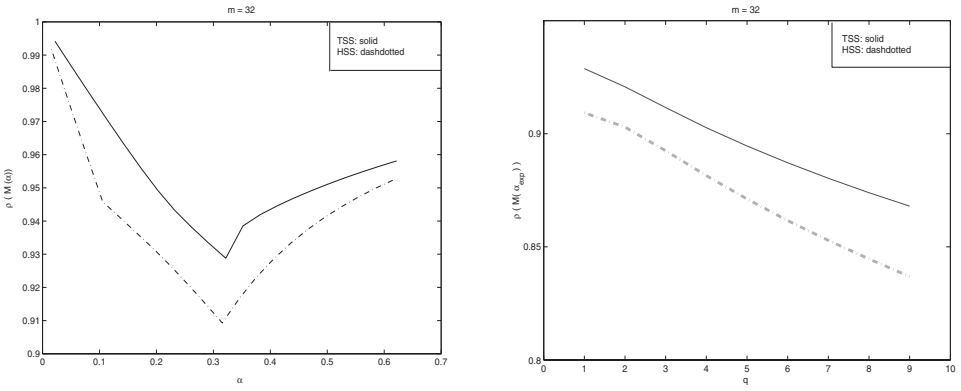

FIG. 5. *Curves of $\rho(M(\alpha))$ versus α with $q = 1$(left) and $\rho(M(\alpha_{\exp}))$ versus q (right) for the HSS and TSS iteration matrices when $m = 32$ for Example* 5.1.

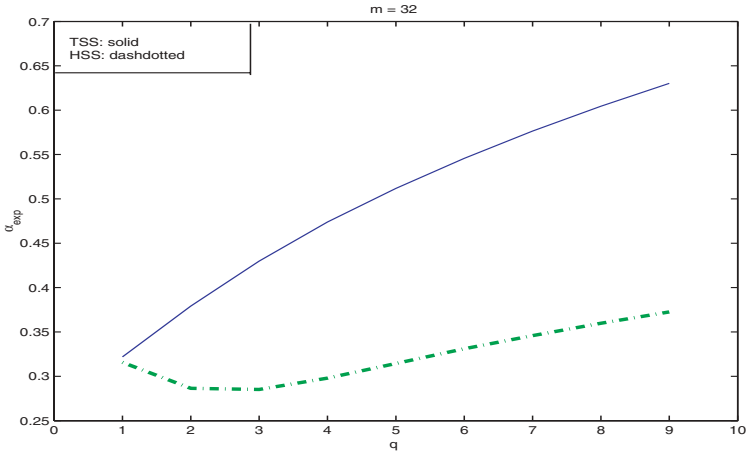

FIG. 6. *Curves of $\alpha_{\exp}$ versus q for the HSS and the TSS iteration matrices when $m = 32$ for Example* 5.1.

In fact, the speed-up of TSS with respect to HSS is quite noticeable, where we define it by

$$\text{speed-up} = \frac{\text{CPU of HSS method}}{\text{CPU of TSS (or BTSS) method}}.$$

In Table 2, the speed-up is at least 1.52 for $m = 64$, and it even achieves 3.76 for $m = 24$.

As we have mentioned in sections 2 and 3, computing the optimal parameter α_{opt} for both HSS and TSS is very difficult and is usually problem dependent. In Figure 5 we depict the curves of $\rho(M(\alpha))$ with respect to α, and $\rho(M(\alpha_{\exp}))$ with respect to q, and in Figure 6 we depict the curves of $\alpha_{\exp}$ with respect to q, when $m = 32$, for both TSS and HSS iteration methods, to intuitively show these functional relationships. Evidently, from Figure 5 we see that $\rho(M(\alpha))$ attains the minimum at about $\alpha = \alpha_{\exp}$, that $\rho(M(\alpha_{\exp}))$ monotonically decreases when q is increasing, and that both $\rho(M(\alpha))$ and $\rho(M(\alpha_{\exp}))$ for TSS are larger than those for HSS for all α

TABLE 3
$\alpha_{\exp}$ *and* $\rho(M(\alpha_{\exp}))$ *when* $m = 32$ *for Example* 5.1.

q		1	2	3	4	5	6	7	8	9
TSS	$\alpha_{\exp}$	0.322	0.379	0.430	0.474	0.512	0.546	0.576	0.605	0.630
	$\rho(M(\alpha_{\exp}))$	0.929	0.921	0.912	0.903	0.895	0.887	0.880	0.874	0.868
HSS	$\alpha_{\exp}$	0.316	0.286	0.285	0.298	0.315	0.331	0.346	0.360	0.373
	$\rho(M(\alpha_{\exp}))$	0.909	0.903	0.892	0.882	0.871	0.862	0.853	0.845	0.837

TABLE 4
IT and CPU when $m = 32$ *for Example* 5.1.

q		1	2	3	4	5	6	7	8	9
TSS	IT	113	98	95	91	88	83	79	79	76
	CPU	48.095	38.567	39.224	35.889	35.161	18.116	17.741	17.379	16.330
HSS	IT	108	106	99	91	82	83	71	72	65
	CPU	87.906	91.372	85.389	72.429	37.056	38.673	32.981	31.427	29.013
Speed-up		1.83	2.37	2.18	2.02	1.05	2.13	1.86	1.81	1.78

and q, respectively. These facts are further confirmed by the numerical results listed in Table 3. It then follows that the IT of TSS will be larger than those of HSS. However, because TSS has a smaller computational workload than HSS at each of the IT, the actual CPU of TSS may be less than that of HSS, which straightforwardly implies that TSS is, practically, more efficient than HSS. These facts are confirmed by the numerical results in Table 4. In fact, in Table 4 the speed-up of TSS with respect to HSS is quite noticeable; it is at least 1.78 for all tested values of q, except for $q = 5$ whose speed-up is 1.05, and it even achieves 2.18 for $q = 3$. In addition, from Figure 6 we observe that $\alpha_{\exp}$ monotonically increases when q is increasing for both TSS and HSS iteration methods, $\alpha_{\exp}$ for TSS is larger than that for HSS for all q, and the gap between the $\alpha_{\exp}$'s for TSS and HSS also becomes larger when q becomes larger.

In Figure 7 we depict the curves of IT and CPU with respect to q for both TSS and HSS iteration methods. We see that for a small q, say $q < 4$, the IT of TSS are less than those of HSS; while for a large q, say $q \geq 4$, the situation is reversed. However, CPU of TSS is always less than that of HSS for all of our tested q. This clearly shows that TSS is much more effective than HSS in actual computations.

To compare the computing efficiency of TSS as an iteration scheme and as a preconditioner to the Krylov subspace methods, such as GMRES(ℓ) ("ℓ" denotes the number of the restarting steps) and BiCGSTAB, we choose the starting vector $x^{(0)} = (\sin(1), \sin(2), \ldots, \sin(n))^T$ and the stopping criterion $\|r^{(k)}\|_2/\|r^{(0)}\|_2 < 10^{-4}$. Similarly, the right-hand-side vector b is computed from $b = Ax^\star$, where $x^\star$ is the exact solution of the system of linear equations (1), and is a randomly generated vector. The numerical results with respect to IT, CPU, and RES (relative residual error defined by RES $= \|r^{(k)}\|_2/\|r^{(0)}\|_2$) are listed in Tables 5 and 6. Here, we use "–" to represent that the corresponding iteration cannot achieve the above-mentioned stopping criterion after $10m$ iteration steps.

From Table 5 we see that, as an iteration scheme, TSS outperforms both GMRES (ℓ) ($\ell = 5, 10, 15$, and 20) and BiCGSTAB from the aspect of both IT and CPU. Also from Table 6 we observe that as a preconditioner TSS improves the numerical behavior

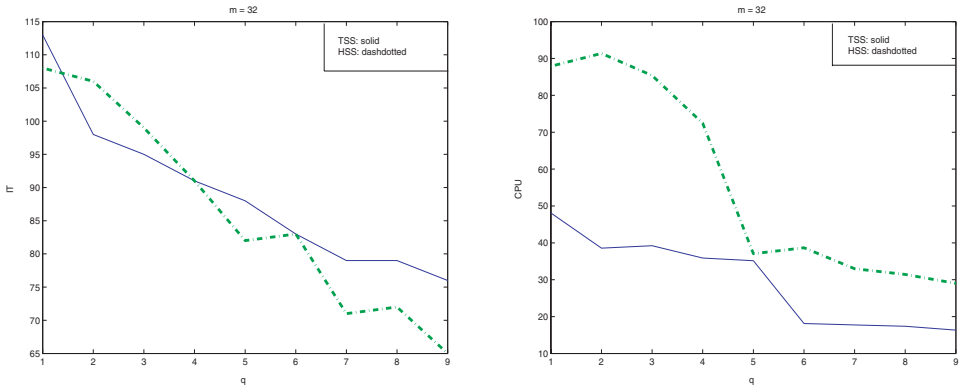

FIG. 7. *Curves of IT versus q (left) and CPU versus q (right) for the HSS and the TSS iteration methods when* $\alpha = \alpha_{\exp}$ *and* $m = 32$ *for Example* 5.1.

TABLE 5
IT, CPU, and RES for Example 5.1.

q		7		8		9	
m		32	64	32	64	32	64
TSS	$\alpha_{\exp}$	0.5762	0.3072	0.6041	0.3231	0.6303	0.3379
	IT	40	80	38	76	38	73
	CPU	0.35	19.96	0.33	19.00	0.32	18.96
GMRES(5)	IT	201	419	458	309	–	–
	CPU	1.54	65.52	3.51	55.07	–	–
	RES	–	–	–	–	4.32e-2	1.14e-2
GMRES(10)	IT	144	201	181	309	387	–
	CPU	0.95	34.80	1.19	55.07	2.58	–
	RES	–	–	–	–	–	8.76e-3
GMRES(15)	IT	151	178	161	309	200	346
	CPU	0.97	22.47	1.03	55.07	1.25	62.57
	RES	–	–	–	–	–	–
GMRES(20)	IT	120	163	178	231	179	218
	CPU	0.72	20.21	1.08	37.83	1.10	34.48
BiCGSTAB	IT	50	85	53	82	60	86
	CPU	0.54	27.34	0.57	26.67	0.65	25.11

of both GMRES(ℓ) ($\ell = 5, 10, 15$, and 20) and BiCGSTAB considerably more than both ILU (incomplete LU factorization with no fill-in) and UGS (unsymmetric Gauss–Seidel iteration). See [10, 12].

As a matter of fact, TSS requires much less computational storage than GMRES(ℓ) and BiCGSTAB, and it possesses better convergence properties than UGS. In addition, compared with ILU and UGS, TSS can lead to a high-quality preconditioning matrix. This is why TSS is superior to GMRES(ℓ) and BiCGSTAB as an iteration scheme, as well as to ILU and UGS as a preconditioner.

Example 5.2. Consider the system of linear equations (1), for which

$$A = \begin{bmatrix} W & F\Omega \\ -F^T & N \end{bmatrix}, \text{ where } \quad W \in \mathbb{R}^{q\times q} \quad \text{and} \quad N, \Omega \in \mathbb{R}^{(n-q)\times(n-q)}, \quad \text{with} \quad 2q > n.$$

We define the matrices $W = (w_{k,j})$, $N = (n_{k,j})$, $F = (f_{k,j})$, and $\Omega = \text{diag}(\omega_1, \ldots,$

TABLE 6
IT and CPU when $q = 7$ for Example 5.1.

Methods	m	Preconditioners					
		TSS		ILU		UGS	
		IT	CPU	IT	CPU	IT	CPU
GMRES(5)	32	24	0.18	39	1.78	561	3.45
	64	28	3.66	140	96.70	362	142.74
GMRES(10)	32	22	0.12	27	1.59	410	2.41
	64	28	3.15	100	23.34	335	112.81
GMRES(15)	32	23	0.12	25	0.91	263	1.38
	64	28	2.90	58	32.31	391	123.82
GMRES(20)	32	20	0.09	19	0.65	189	0.97
	64	26	2.73	55	29.73	280	84.91
BiCGSTAB	32	14	0.12	21	1.22	101	0.92
	64	15	2.83	51	11.47	151	87.80

ω_{n-q}) as follows:

$$w_{k,j} = \begin{cases} k+1 & \text{for} \quad j = k, \\ 1 & \text{for} \quad |k-j| = 1, \\ 0 & \text{otherwise}, \end{cases} \qquad k, j = 1, 2, \ldots, q,$$

$$n_{k,j} = \begin{cases} k+1 & \text{for} \quad j = k, \\ 1 & \text{for} \quad |k-j| = 1, \\ 0 & \text{otherwise}, \end{cases} \qquad k, j = 1, 2, \ldots, n-q,$$

$$f_{k,j} = \begin{cases} j & \text{for} \quad k = j + 2q - n, \\ 0 & \text{otherwise}, \end{cases} \qquad k = 1, 2, \ldots, q; j = 1, 2, \ldots, n-q,$$

and

$$\omega_k = \frac{1}{k}, \qquad k = 1, 2, \ldots, n-q.$$

Note that $A \in \mathbb{R}^{n \times n}$ is a nonsymmetric positive-definite matrix. We use the BTSS iteration method defined by (12)–(14) to solve the system of linear equations (1) and solve the shifted skew-Hermitian linear subsystem at each BTSS iteration step by the strategy specified in section 4. In our computations, we choose $q = \frac{9}{10}n$.

In Figures 8 and 9, we plot the eigenvalues of the iteration matrices of both HSS and BTSS methods when $n = 400$ and $n = 800$, respectively. Analogously, it is clear that the eigenvalue distributions of the HSS and BTSS iteration matrices are quite different: the eigenvalues of the HSS iteration matrix are clustered on the real axis and a circular arc on the complex plane, while those of the BTSS iteration matrix are clustered on the real axis and a circular arc as well as located in a triangular area; however, the distribution domain of the eigenvalues of the BTSS iteration matrix is considerably smaller than that of the HSS iteration matrix, in particular, along the direction of the imaginary axis.

In Table 7, we list the experimentally optimal parameters $\alpha_{\exp}$ and the corresponding spectral radii $\rho(M(\alpha_{\exp}))$ of the iteration matrices $M(\alpha_{\exp})$ of both BTSS

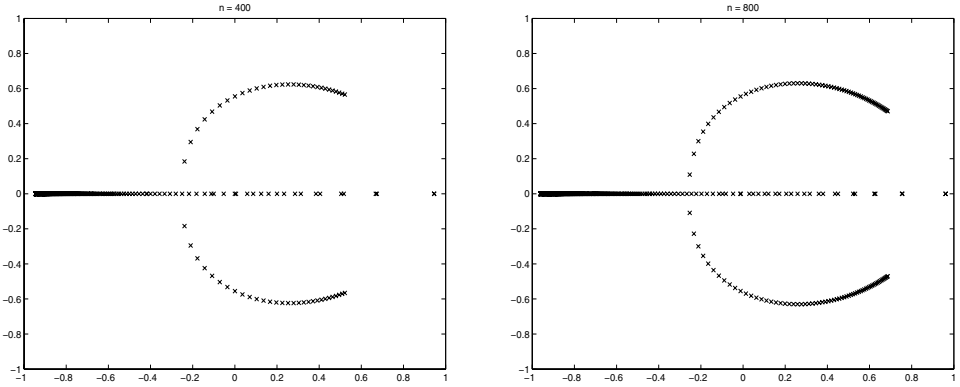

FIG. 8. *The eigenvalue distributions of the HSS iteration matrices when* $n = 400$ *(left) and* $n = 800$ *(right) for Example* 5.2.

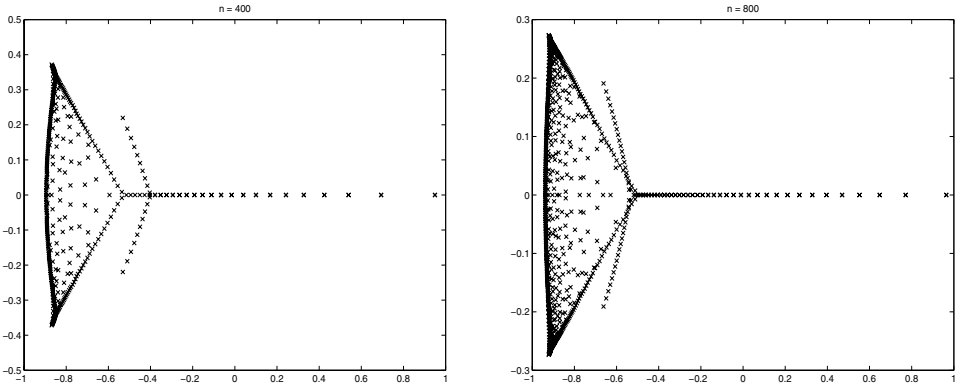

FIG. 9. *The eigenvalue distributions of the BTSS iteration matrices when* $n = 400$ *(left) and* $n = 800$ *(right) for Example* 5.2.

and HSS methods for several n. The data show that when n is increasing, $\alpha_{\exp}$ and $\rho(M(\alpha_{\exp}))$ are increasing for both BTSS and HSS methods. $\rho(M(\alpha_{\exp}))$ of BTSS is slightly larger than that of HSS, correspondingly, for all our tested n. This straightforwardly implies that the IT of BTSS may be comparable to that of HSS. Because BTSS has a smaller computational workload than HSS at each of the IT, the actual CPU of BTSS may be much less than that of HSS. Therefore, BTSS will be much more efficient than HSS. This fact is further confirmed by the numerical results in Table 8, where the speed-up is at least 1.75 for $n = 800$, and it even achieves 2.41 for $n = 200$.

TABLE 7
α_{exp} *and* $\rho(M(\alpha_{exp}))$ *for Example* 5.2.

n		100	200	400	800	1600
BTSS	α_{exp}	4.865	6.874	9.713	13.733	19.418
	$\rho(M(\alpha_{exp}))$	0.901	0.929	0.949	0.964	0.974
HSS	α_{exp}	4.476	6.351	8.999	12.736	18.018
	$\rho(M(\alpha_{exp}))$	0.896	0.924	0.946	0.961	0.972

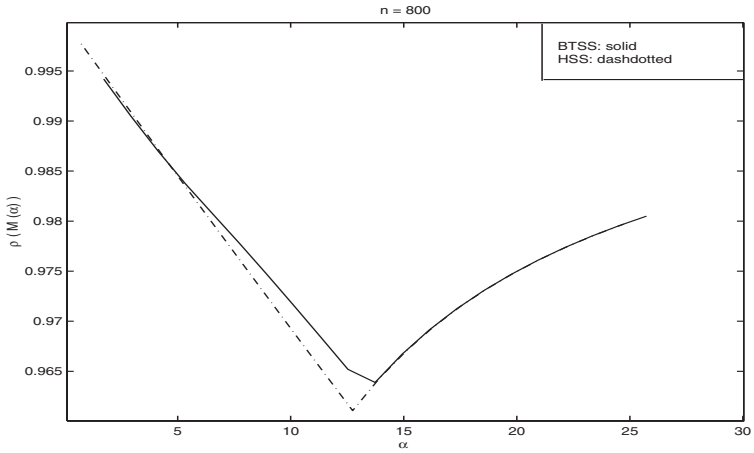

FIG. 10. *Curves of $\rho(M(\alpha))$ versus α for the HSS and the BTSS iteration matrices when $n = 800$ for Example* 5.2.

TABLE 8
IT and CPU for Example 5.2.

n		100	200	400	800	1600
BTSS	IT	66	101	145	210	293
	CPU	0.064	0.402	2.991	39.869	208.375
HSS	IT	70	97	134	192	269
	CPU	0.133	0.967	6.635	69.744	394.706
Speed-up		2.08	2.41	2.22	1.75	1.89

In Figure 10 we depict the curves of $\rho(M(\alpha))$, with respect to α when $n = 800$, for both BTSS and HSS iteration methods to intuitively show this functional relationship. Evidently, we see that $\rho(M(\alpha))$ attains the minimum at about $\alpha = \alpha_{\exp}$, and the spectral radius of the BTSS iteration matrix is almost equal to that of the HSS iteration matrix when α becomes larger than $\alpha_{\exp}$ of BTSS.

6. Concluding remarks. We have further developed the HSS and NSS iteration methods and established a more general framework of iteration methods based on the PSS of the positive-definite coefficient matrix of the system of linear equations. We have proved the convergence of the PSS iteration method and showed that it inherits all advantages of both HSS and NSS iteration methods. Several special examples of the PSS method, i.e., the TSS (or BTSS) methods, have been described, and numerical examples have been implemented to show that in the senses of computational storage and CPU time, the TSS and BTSS iteration methods are much more practical and effective as iteration schemes than the HSS iteration method, as well as the Krylov subspace methods such as GMRES(ℓ) and BiCGSTAB, and they are also much more practical and effective as preconditioners than the ILU factorization and the UGS iteration. However, we should mention that the questions of how to optimally choose the iteration parameters and how to efficiently solve the shifted skew-Hermitian linear

systems involved in these iteration methods are very practical and interesting problems that need further in-depth study.

REFERENCES

[1] Z.-Z. Bai, G.H. Golub, and M.K. Ng, *Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 603–626.

[2] Z.-Z. Bai, G.H. Golub, and M.K. Ng, *On Successive-Overrelaxation Acceleration of the Hermitian and Skew-Hermitian Splitting Iteration*, available online at http://www-sccm.stanford.edu/wrap/pub-tech.html.

[3] Z.-Z. Bai, G.H. Golub, and J.-Y. Pan, *Preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive semidefinite linear systems*, Numer. Math., 98 (2004), pp. 1–32.

[4] M. Benzi and G.H. Golub, *An Iterative Method for Generalized Saddle Point Problems*, Technical Report SCCM-02-14, Scientific Computing and Computational Mathematics Program, Department of Computer Science, Stanford University, Stanford, CA, 2002. Available online at http://www-sccm.stanford.edu/wrap/pub-tech.html

[5] J.R. Bunch, *A note on the stable decomposition of skew-symmetric matrices*, Math. Comput., 38 (1982), pp. 475–479.

[6] J.R. Bunch, *Stable algorithms for solving symmetric and skew-symmetric systems*, Bull. Austral. Math. Soc., 26 (1982), pp. 107–119.

[7] P. Concus and G.H. Golub, *A generalized conjugate gradient method for non-symmetric systems of linear equations*, in Computing Methods in Applied Sciences and Engineering, Lecture Notes in Econom. and Math. Systems 134, R. Glowinski and J.R. Lions eds., Springer-Verlag, Berlin, 1976, pp. 56–65. Also available online at http://www-sccm.stanford.edu/wrap/pub-other.html

[8] J. Douglas, Jr. and H. Rachford, Jr., *Alternating direction methods for three space variables*, Numer. Math., 4 (1956), pp. 41–63.

[9] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[10] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

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

[12] R.S. Varga, *Matrix Iterative Analysis*, 2nd revised and expanded ed., Ser. Comput. Math. 27, Springer-Verlag, Berlin, 2000.