Electronic Transactions on Numerical Analysis.
Volume 54, pp. 370–391, 2021.
Copyright © 2021, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol54s370

ETNA
Kent State University **and**
Johann Radon Institute (RICAM)

A TWO-LEVEL ITERATIVE SCHEME FOR GENERAL SPARSE LINEAR SYSTEMS BASED ON APPROXIMATE SKEW-SYMMETRIZERS[*]

MURAT MANGUOĞLU[†] AND VOLKER MEHRMANN[‡]

Abstract. We propose a two-level iterative scheme for solving general sparse linear systems. The proposed scheme consists of a sparse preconditioner that increases the norm of the skew-symmetric part relative to the rest and makes the main diagonal of the coefficient matrix as close to the identity as possible so that the preconditioned system is as close to a shifted skew-symmetric matrix as possible. The preconditioned system is then solved via a particular Minimal Residual Method for Shifted Skew-Symmetric Systems (MRS). This leads to a two-level (inner and outer) iterative scheme where the MRS has short-term recurrences and satisfies an optimality condition. A preconditioner for the inner system is designed via a skew-symmetry-preserving deflation strategy based on the skew-Lanczos process. We demonstrate the robustness of the proposed scheme on sparse matrices from various applications.

Key words. symmetrizer, skew-symmetrizer, Krylov subspace method, shifted skew-symmetric system, skew-Lanczos method

AMS subject classifications. 65F08, 65F10, 65F50

1. Introduction.

We discuss the numerical solution of general linear systems

$$Ax = b,$$

where $A \in \mathbb{R}^{n\times n}$ is a general large sparse invertible matrix. If the coefficient matrix is symmetric and positive definite or symmetric and indefinite, then one can use the Conjugate Gradient algorithm or the recently proposed two-level iterative scheme in [23], respectively. In this paper, we propose a new robust two-level black-box scheme for solving general systems without any assumption on the symmetry or definiteness of the coefficient matrix. In contrast to most other iterative methods where preconditioning is often used to symmetrize the system and to lower the condition number, our new approach consists of an initial step which makes the system close to an identity-plus-skew-symmetric matrix that leads to an effective shifted skew-symmetric preconditioner. Both the preconditioned system and the application of the preconditioner is approached by an iterative method so that the method is a two-level (inner-outer) iterative scheme.

Our main motivation to study identity-plus-skew-symmetric preconditioners are linear systems arising in the time discretization of dissipative Hamiltonian differential equations of the form

$$E\dot{z} = (J - R)\,z + f(t), \qquad z(t_0) = z_0,$$

where $\dot{z}$ denotes the derivative with respect to time, J is a skew-symmetric matrix, R is symmetric positive semidefinite, and E is the symmetric positive semidefinite Hessian of a quadratic energy functional (Hamiltonian) $\mathcal{H}(z) = \frac{1}{2}z^T Ez$; see, e.g., [2, 10, 14, 20, 31, 32] for such systems in different physical domains and applications. If one discretizes such

[*]Received September 15, 2020. Accepted March 17, 2021. Published online on April 29, 2021. Recommended by M. Hochstenbach. The first author was supported by Alexander von Humboldt Foundation for a research stay at TU Berlin and the BAGEP Award of the Science Academy. The second author was supported by Deutsche Forschungemeinschaft through collaborative research center SFB TRR 154 Project B03.

[†]Institut für Mathematik, Technische Universität Berlin, 10623 Berlin, Germany. Present address: Department of Computer Engineering, Middle East Technical University, 06800 Ankara, Turkey (`manguoglu@ceng.metu.edu.tr`).

[‡]Institut für Mathematik, Technische Universität Berlin, 10623 Berlin, Germany (`mehrmann@math.tu-berlin.de`).

systems in time, e.g., with the implicit Euler method, and sets $z_k = z(t_k)$, then in each time step t_k one has to solve a linear system of the form

$$(E - h(J - R))z_{k+1} = Ez_k + hf(t_k). \tag{1.1}$$

Similar linear systems arise also when other discretization schemes are used. We note that if the right-hand side vector or the coefficient matrix or both are slowly changing, then it is possible to use *recycling* Krylov subspace methods [25]. In (1.1), the right-hand side vector is changing, therefore our proposed method can easily adopt recycling Krylov subspace methods as the outer iterative scheme, which we leave as a future work.

The matrix $A = E + h(R - J)$ has a positive (semi)definite symmetric part $M = E + hR$. If M is positive definite, then with a two-sided preconditioning with the Cholesky factor L of $M = LL^T$, the matrix $L^{-1}AL^{-T}$ has the form $I + \widetilde{J}$, where $\widetilde{J} = hL^{-1}AL^{-T}$ is skew-symmetric [5, 35]. For such systems, structure-exploiting Krylov subspace methods with three-term recurrences were derived and analyzed in [5, 19, 22, 29, 35]. Given a general square matrix A, symmetrizers from right or left are, respectively, symmetric matrices S_r or S_l such that $AS_r = S_r^T A^T$ or $S_l A = A^T S_l^T$. Existing algorithms for constructing dense and exact symmetrizers are studied and summarized in [8]. In this paper, however, we construct two-sided preconditioners so that the preconditioned systems have the form $D + \hat{J}$, where D is diagonal and close to the identity and $\hat{J}$ is close to a skew-symmetric matrix (*approximate shifted skew-symmetrizers (ASSS)*). To this preconditioned system we then apply a two-level iterative method, where the inner iteration is a skew-symmetric Krylov subspace method. We assume that both A and $S \in \{S_r, S_l\}$ are sparse and nonsymmetric, with S having a user-defined sparsity structure. The sparse ASSS preconditioner is obtained by first applying a nonsymmetric permutation and scaling and then solving a sparse overdetermined linear least squares (LLS) problem to obtain S. Similar approaches for dense symmetrizers [8] or algorithms for improving the structural symmetry in the context of sparse direct solvers, as proposed in [26, 30], do not have the latter property.

We note that while it is possible to obtain and use either S_r or S_l, in our experience, the numerical results did not differ much for the test problems. Therefore, in the rest of the paper we use the right variant and hereafter S refers to S_r.

The paper is organized as follows. The proposed sparse approximate skew-symmetrizer is introduced in Section 2, a two-level Krylov subspace method based on the skew-symmetrizer is introduced in Section 3, numerical results to show the robustness of the proposed method and to show the timings for a large-scale problem are presented in Section 4 and Section 5, respectively, and the conclusions follow in Section 6.

2. A sparse approximate shifted skew-symmetrizing preconditioner.

Given a sparse invertible matrix $A \in \mathbb{R}^{n\times n}$, then in order to achieve our goal of constructing a sparse approximate shifted skew-symmetrizing (ASSS) preconditioner, we first apply diagonal scalings (D_r, D_c) and a row permutation ($\mathcal{P}$),

$$\bar{A} = \mathcal{P}D_r A D_c \tag{2.1}$$

such that the diagonal entries of $\bar{A}$ have modulus one and the off-diagonal elements are of modulus less than or equal to one. Such a permutation and scaling procedure is well established in the code MC64 of the Harwell Subroutine Library (HSL) [18], and it is called *the maximum product transversal with scaling*. It solves a weighted bipartite matching problem, and the resulting matrix $\bar{A}$ is guaranteed to contain a zero-free main diagonal if A is structurally nonsingular [9]. For a detailed description of this permutation and scaling, we refer the reader to [24], where it was originally proposed to reduce the required amount of pivoting for dense Gaussian elimination.

After this, we look for a sparse matrix S such that

$$(\bar{A}S)_{i,j} = -(\bar{A}S)_{j,i}, \qquad \text{for } i \neq j, \tag{2.2}$$

and

$$(\bar{A}S)_{i,i} = 1, \qquad \text{for } i = 1, 2, \ldots, n, \tag{2.3}$$

where S can have various sparsity structures, such as being diagonal, tridiagonal, banded, having the sparsity of $\bar{A}$, or any structure defined by the user.

The described problem can be formulated as a sparse over-determined LLS problem, where by (2.2), each nonzero in the strictly upper triangular part of $|\bar{A}S| + |\bar{A}S|^T$ defines a constraint of the LLS problem and additional n constraints are obtained via (2.3). Let nz be the number of nonzeros in the strictly upper triangular part of $|\bar{A}S| + |\bar{A}S|^T$ and $nnz(S)$ be the number of nonzeros in S. Then the LLS problem has $nnz(S)$ unknowns and $nz + n$ equations, and if $nz + n > nnz(S)$, then the problem is overdetermined.

As a first example of a sparsity structure, let us assume that $S = \text{diag}(s_{1,1}, \ldots, s_{n,n})$, so that $nnz(S) = n$. Then, (2.2) and (2.3) are given by

$$\bar{a}_{i,j}s_{j,j} + \bar{a}_{j,i}s_{i,i} = 0,$$

and

$$\bar{a}_{i,i}s_{i,i} = 1,$$

respectively. With $s = [s_{1,1}, s_{2,2}, \ldots, s_{n,n}]^T$, the resulting overdetermined system is given by

$$\begin{bmatrix} B_u \\ B_l \end{bmatrix} s = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}, \tag{2.4}$$

where $\mathbf{0}$ and $\mathbf{1}$ are vectors of all zeros of size nz and all ones of size n, respectively. B_u is a sparse matrix of size $nz \times n$, where each row has only two nonzeros, $\bar{a}_{i,j}$ and $\bar{a}_{j,i}$, in its ith and jth columns, respectively, while B_l is just the diagonal of $\bar{a}_{i,i}$. So with

$$(s) := \left\| \begin{bmatrix} B_u \\ B_l \end{bmatrix} s - \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \right\|_2^2,$$

the unique solution of the LLS problem is obtained by computing $\min_s f(s)$. The unique solution can be obtained via a direct or iterative sparse LLS solver. To gain more flexibility with respect to the importance of the two constraints, we introduce a weighting parameter $(\gamma > 0)$ and solve the weighted problem

$$f(s,\gamma) = \|B_u s\|_2^2 + \gamma \|B_l s - \mathbf{1}\|_2^2 = \left\|(\bar{A}S) + (\bar{A}S)^T\right\|_F^2 + \gamma \left\|\mathcal{D}(\bar{A}S) - I\right\|_F^2, \tag{2.5}$$

where $\mathcal{D}(X)$ denotes a diagonal matrix whose diagonal entries are those of X.

For a general sparse S, the LLS problem is formulated in a similar way as in the diagonal case. The set of constraints is defined for each nonzero (i, j) in the strictly upper (or lower) triangular nonzero pattern of the matrix $|\bar{A}S| + |\bar{A}S|^T$ via (2.2) (using MATLAB column notation) via

$$\bar{A}_{i,:}S_{:,j} + \bar{A}_{j,:}S_{:,i} = 0,$$

and the diagonal constraints are obtained, for $i = 1, 2, \ldots, n$, via (2.3). Note that one needs to map the nonzero entries of S to a vector to form the LLS problem and map it back to S after obtaining the solution vector. This can be done using the sparse matrix storage format. In Appendix A, we present a MATLAB implementation which stores the nonzeros of sparse matrices in column-major order, i.e., compressed sparse column format.

3. A bilevel iterative scheme.

Given a general sparse linear system

$$Ax = b, \tag{3.1}$$

where $A \in \mathbb{R}^{n\times n}$ is nonsingular. As discussed in the introduction, many preconditioners are either applied or aim for a symmetric or symmetric positive definite system, since for these we have short recurrences in Krylov subspace methods like the conjugate gradient method. Only very few algorithms focus on skew-symmetric or shifted skew-symmetric structures. In this section we present the theoretical basis for an algorithm that preprocesses the system such that the coefficient matrix is as close as possible to a shifted skew-symmetric matrix and then use the shifted skew-symmetric part of the matrix as preconditioner applying it as an iterative solver with short recurrences and optimality property that requires only one inner product per iteration.

Consider the splitting of the coefficient matrix into its symmetric and skew-symmetric parts

$$A = M + J,$$

where $M = M^T$ and $J = -J^T$. (In applications, J often is a matrix of small norm). If M is positive definite, then one can precondition the system by computing the Cholesky factorization $M = LL^T$ and solve the modified system

$$(I + L^{-1}JL^{-T})L^T x = L^{-1}b,$$

where $L^{-1}JL^{-T}$ is again skew-symmetric. However, in general, M is not positive definite; it may be indefinite or singular. In this case we propose a black-box algorithm that employs an ASSS preconditioner. This is a two-level procedure in which we first apply the discussed nonsymmetric row permutation and scaling to obtain a zero-free diagonal with diagonal entries of modulus one and off-diagonal entries of modulus less than or equal to one. The second step applies a sparse matrix S obtained via the algorithm described in Section 2 by solving a sparse LLS problem. After ASSS preconditioning, the modified system has the form

$$\widehat{A}\widehat{x} = \widehat{b}, \tag{3.2}$$

where $\widehat{A} = \mathcal{P}D_rAD_cS$, $\widehat{x} = S^{-1}D_c^{-1}x$, and $\widehat{b} = \mathcal{P}D_rb$. Let $\widehat{M} = \frac{\widehat{A}+\widehat{A}^T}{2}$ and $\widehat{J} = \frac{\widehat{A}-\widehat{A}^T}{2}$. Note that due to the ASSS preconditioning, even though $\widehat{M}$ is not guaranteed to be positive definite, it has eigenvalues clustered around 1 and typically very few negative eigenvalues. Furthermore, the skew-symmetric part $\widehat{J}$ is more dominant now. One can now compute a Bunch-Kaufman-Parlett factorization [4], $\widehat{M} = \widehat{L}\widehat{D}\widehat{L}^T$ (where $\widehat{L}$ is a sparse lower triangular matrix and $\widehat{D}$ is block-diagonal matrix with either 1×1 or 2×2 blocks) and modify the factorization to obtain

$$|\widehat{M}| = \widehat{L}|\widehat{D}|\widehat{L}^T,$$

where, as in [34], $|\widehat{D}| = V|\Lambda|V^T$ if $\widehat{D}$ has a spectral decomposition $V\Lambda V^T$. Then, $|\widehat{D}|$ has a Cholesky factorization $L_{|\widehat{D}|}L_{|\widehat{D}|}^T$ since it is positive definite. Setting $\mathcal{L} := \widehat{L}L_{|\widehat{D}|}$ and multiplying (3.2) from the left with $\mathcal{L}^{-1}$ and inserting $I = \mathcal{L}^{-T}\mathcal{L}^T$, we obtain the system $\mathcal{A}\mathrm{x} = \mathrm{b}$, where $\mathcal{A} = \mathcal{L}^{-1}\widehat{A}\mathcal{L}^{-T}$, $\mathrm{x} = \mathcal{L}^T\widehat{x}$ and $\mathrm{b} = \mathcal{L}^{-1}\widehat{b}$. We note that $\mathcal{A}$ can be split as

$$\mathcal{A} = \underbrace{(L_{|\widehat{D}|}^{-1}\widehat{D}L_{|\widehat{D}|}^{-T} - I)}_{\mathcal{M}_r} + (I + \underbrace{\mathcal{L}^{-1}\widehat{J}\mathcal{L}^{-T}}_{\mathcal{J}}),$$

where the rank of $\mathcal{M}_r$ is equal to the number of negative eigenvalues of $\widehat{D}$, which is expected to be very small, and $I+\mathcal{J}$ is a shifted skew-symmetric matrix. Furthermore, $\mathcal{M}_r$ is symmetric and block-diagonal with only a few nonzero blocks of size either 1×1 or 2×2, and is of rank $r \ll n$. The 1×1 blocks have the value -2, and the 2×2 blocks have eigenvalues $\{-2, 0\}$. Due to the (almost) diagonal and low-rank structure of $\mathcal{M}_r$, it is easy to obtain a symmetric low-rank decomposition

$$\mathcal{M}_r = U_r \Sigma_r U_r^T, \tag{3.3}$$

where $\Sigma_r = -2I_r$ and U_r is a sparse (with either one or two nonzero entries per column) $n \times r$ orthogonal matrix. A pseudocode for computing such low-rank decomposition is presented in Algorithm 1.

Algorithm 1 Sparse low-rank decomposition of $\mathcal{M}_r = U_r \Sigma_r U_r^T$.

```
Input: $\mathcal{M}_r \in \mathbb{R}^{n\times n}$, $r$ (rank of $\mathcal{M}_r$), Ind (set of indices of nonzeros of $\mathcal{M}_r$)
$\Sigma_r \leftarrow 0, U_r \leftarrow 0, U \leftarrow 0$
$i \leftarrow 1, j \leftarrow 1$
$\mathcal{M}'_r \leftarrow \mathcal{M}_\nabla(\text{Ind}, \text{Ind})$
while $(i < r)$ do
  if $(\mathcal{M}'_r(i, i+1) = 0)$ then
    $\Sigma_r(j,j) \leftarrow \mathcal{M}'_r(i,i)$
    $U(:,j) \leftarrow e_i$
    $i \leftarrow i+1$
  else
    Compute the eigenpair $\{\lambda_2, v_2\}$ of $\mathcal{M}'_r(i:i+1, i:i+1)$
    $\Sigma_r(j,j) \leftarrow \lambda_2$
    $U(:,j) \leftarrow [e_i, e_{i+1}]v_2$
    $i \leftarrow i+2$
  end if
  $j \leftarrow j+1$
end while
if (i = r) then
  $\Sigma_r(j,j) \leftarrow \mathcal{M}'_r(i,i)$
  $U(:,j) \leftarrow e_i$
end if
$U_r(\text{Ind}, :) \leftarrow U$
Output: $U_r \in \mathbb{R}^{n\times r}, \Sigma_r \in \mathbb{R}^{r\times r}$
```

The cost of this last step is $O(r)$ arithmetic operations since it only needs to work with a submatrix of $\mathcal{M}_r$ corresponding to the indices of the nonzero entries. Using this factorization, we obtain $\mathcal{A} = U_r \Sigma_r U_r^T + \mathcal{S}$, where $\mathcal{S} = I + \mathcal{J}$, so that $\mathcal{A}$ is a shifted skew-symmetric matrix with a low-rank perturbation. Using the Sherman-Morrison-Woodbury formula [13], we theoretically have the exact inverse

$$\mathcal{A}^{-1} = \mathcal{S}^{-1} - \mathcal{S}^{-1}U_r(\Sigma_r^{-1} + U_r^T\mathcal{S}^{-1}U_r)^{-1}U_r^T\mathcal{S}^{-1},$$

which can be applied to the right-hand side vector b to obtain x, by solving only shifted skew-symmetric linear systems.

In practice, for large-scale sparse systems, it is too expensive to compute the full LDL^T factorization of $\widehat{M}$. Instead, an incomplete factorization $\widetilde{M} = \widetilde{L}\widetilde{D}\widetilde{L}^T$ can be utilized together with the Cholesky factorization of $|\widetilde{D}| = L_{|\widetilde{D}|}L_{|\widetilde{D}|}^T$, where $\widetilde{\mathcal{L}} = \widetilde{L}L_{|\widetilde{D}|}$. This leads to a modified system,

$$\widetilde{\mathcal{L}}^{-1}\widehat{A}\widetilde{\mathcal{L}}^{-T}\widetilde{\mathcal{L}}^T\widehat{x} = \widetilde{\mathcal{L}}^{-1}\widehat{b},$$

which then is solved iteratively using a Krylov subspace method with the preconditioner

$$P = \underbrace{(L_{|\widetilde{D}|}^{-1}\widetilde{D}L_{|\widetilde{D}|}^{-T} - I)}_{\widetilde{\mathcal{M}}_r} + (I + \underbrace{\widetilde{\mathcal{L}}^{-1}\widehat{J}\widetilde{\mathcal{L}}^{-T}}_{\widetilde{\mathcal{J}}}) \tag{3.4}$$

or alternatively, if r is zero, with a preconditioner

$$P = \widetilde{\mathcal{S}} = I + \widetilde{\mathcal{J}}. \tag{3.5}$$

One can in principle even apply P^{-1} exactly as described earlier. However, in a practical implementation utilizing $\widetilde{\mathcal{S}}^{-1}$ via a direct solver is expensive. Therefore one can apply it inexactly by solving shifted skew-symmetric systems iteratively, where the coefficient matrix is $\widetilde{\mathcal{S}}$. This gives rise to an inner-outer iterative scheme. In addition to solving a shifted skew-symmetric system (where the coefficient matrix does not have to be formed explicitly) with a single right-hand side vector, applying P^{-1} requires sparse matrix-vector/vector-vector operations, the solution of a dense $r \times r$ system, as well as the one-time cost of computing a low-rank decomposition of $\widetilde{\mathcal{M}}_r = \widetilde{U}_r\widetilde{\Sigma}_r\widetilde{U}_r^T$ and solving a shifted skew-symmetric system with multiple right-hand side vectors. The convergence rate of the outer Krylov subspace method depends on the spectrum of the preconditioned coefficient matrix $P^{-1}\widetilde{\mathcal{L}}^{-1}\widehat{A}\widetilde{\mathcal{L}}^{-T}$. The incomplete factorization of $\widehat{M}$ is an approximation such that $\widehat{M} = \widetilde{L}\widetilde{D}\widetilde{L}^T + E$, where E is a small-norm error matrix. Assuming that we apply P^{-1} exactly, the preconditioned coefficient matrix is $P^{-1}\widetilde{\mathcal{L}}^{-1}\widehat{A}\widetilde{\mathcal{L}}^{-T} = I + P^{-1}\widetilde{\mathcal{L}}^{-1}E\widetilde{\mathcal{L}}^{-T}$. Due to the sparse ASSS preconditioning step, $\widehat{M}$ is already close to the identity and $\widehat{J}$ is dominant. Therefore, the norm of the perturbation of the preconditioned matrix from the identity ($\|P^{-1}\widetilde{\mathcal{L}}^{-1}E\widetilde{\mathcal{L}}^{-T}\|$) is expected to be small.

3.1. Solution of sparse shifted skew-symmetric systems. An application of the described preconditioners involves the solution of linear systems, where the coefficient matrix $I + \widetilde{\mathcal{J}}$ is shifted skew-symmetric. Specifically, we are interested in the iterative solution of such systems. While general algorithms such as Biconjugate Gradient Stabilized (BiCGSTAB) [33], Generalized Minimal Residual (GMRES) [27], Quasi-Minimal Residual (QMR) [12], and Transpose-Free Quasi-Minimal Residual (TFQMR) [11] can be used, there are some iterative solvers available for shifted skew-symmetric systems such as CGW [5, 29, 35] and the Minimal Residual Method for Shifted Skew-Symmetric Systems (MRS) [19, 22]. We use MRS since it has a short recurrence and satisfies an optimality property. Furthermore, MRS requires only one inner product per iteration [19], which would be a great advantage if the algorithm is implemented in parallel since inner products require all-to-all reduction operations which create synchronization points. In addition to shifted skew-symmetric systems with one right-hand side vector, we also need to solve such systems with multiple right-hand side vectors. As far as we know, there is currently no “block” MRS available.

Even though block Krylov methods are more amenable to breakdown, there are also ways to avoid the breakdown (for example, block-CG; see [21]). We instead implemented a version of the MRS algorithm based on simultaneous iterations for multiple right-hand side vectors, which is given in Appendix B. In the proposed scheme, the convergence rate of the MRS iteration depends on the spectrum of the shifted skew-symmetric coefficient matrix $I + \widetilde{\mathcal{J}}$. In the next section, we propose a technique for improving this spectrum while preserving its shifted skew-symmetry.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

3.2. Improving the spectrum of shifted skew-symmetric systems via deflation. One disadvantage of the MRS algorithm is that if a preconditioner is used, then the preconditioned system should also be shifted skew-symmetric, which might not be easy to obtain. Therefore, we propose an alternative deflation strategy to improve the number of iterations of MRS. For a shifted skew-symmetric system

$$(I + \widetilde{\mathcal{J}})z = y, \tag{3.6}$$

we eliminate the extreme eigenvalues of $I + \widetilde{\mathcal{J}}$ by applying k iterations ($k \ll n$) of the skew-Lanczos process on $\widetilde{\mathcal{J}}$; see [16, 19, 22]. A pseudocode for this procedure is presented in Algorithm 2.

Algorithm 2 Skew-Lanczos procedure.

```
**Input:** $\widetilde{\mathcal{J}} \in \mathbb{R}^{n\times n}$ ($\widetilde{\mathcal{J}} = -\widetilde{\mathcal{J}}^T$) and $k$.
Let $q_1$ be an arbitrary vector $\in \mathbb{R}^n$
$q_1 \leftarrow q_1/\|q_1\|_2$
$z \leftarrow \widetilde{\mathcal{J}} q_1$
$\alpha_1 \leftarrow \|z\|_2$
**if** $\alpha_1 \neq 0$ **then**
  $q_2 \leftarrow -z/\alpha_1$
  **for** $i = 2$ to $k-1$ **do**
    $z \leftarrow \widetilde{\mathcal{J}} q_i - \alpha_{i-1} q_{i-1}$
    $\alpha_i \leftarrow \|z\|_2$
    **if** $\alpha_i = 0$ **then**
      break
    **end if**
    $q_{i+1} \leftarrow -z/\alpha_i$
  **end for**
**end if**
**Output:** $Q_k = [q_1, q_2, \ldots, q_k] \in \mathbb{R}^{n\times k}$ and $\tau = [\alpha_1, \alpha_2, \ldots, \alpha_{k-1}]^T \in \mathbb{R}^{k-1}$
```

Considering the resulting matrices

$$S_k = \begin{bmatrix} 0 & \alpha_1 & & 0 \\ -\alpha_1 & \ddots & \ddots & \\ & \ddots & \ddots & \alpha_{k-1} \\ 0 & & -\alpha_{k-1} & 0 \end{bmatrix}, \qquad Q_k = \left[q_1, q_2, \ldots, q_k\right],$$

we deflate the system in (3.6) by forming

$$[(I + \widetilde{\mathcal{J}}) - Q_k S_k Q_k^T + Q_k S_k Q_k^T]z = y$$

so that $Q_k^T \widetilde{\mathcal{J}} Q_k = S_k$, where Q_k is $n \times k$ with $Q_k^T Q_k = I$ and S_k is a tridiagonal skew-symmetric $k \times k$ matrix. Let $\bar{\mathcal{J}} = \widetilde{\mathcal{J}} - Q_k S_k Q_k^T$, which is still skew-symmetric and in which the largest (in modulus) eigenvalues have been set to zero. Then the system in (3.6) can be written as a low-rank perturbation of a shifted skew-symmetric system

$$[(I + \bar{\mathcal{J}}) + Q_k S_k Q_k^T]z = y,$$

which can be handled again by the Sherman-Morrison-Woodbury formula. In fact, this low-rank perturbation can be combined with the low-rank perturbation in (3.4), i.e., the preconditioner P can be rewritten as

$$P = \left[Q_k, \widetilde{U}_r\right] \begin{bmatrix} S_k & \\ & \widetilde{\Sigma}_r \end{bmatrix} \begin{bmatrix} Q_k^T \\ \widetilde{U}_r^T \end{bmatrix} + \bar{\mathcal{S}}, \tag{3.7}$$

where $\bar{\mathcal{S}} = I + \bar{\mathcal{J}}$. Then, the preconditioner can be applied directly as via

$$P^{-1} = \bar{\mathcal{S}}^{-1} - \bar{\mathcal{S}}^{-1}\bar{U}_{r+k}\left(\bar{\Sigma}_{r+k} + \bar{U}_{r+k}^T\bar{\mathcal{S}}^{-1}\bar{U}_{r+k}\right)^{-1}\bar{U}_{r+k}^T\bar{\mathcal{S}}^{-1}, \tag{3.8}$$

where $\bar{U}_{r+k} = \left[Q_k, \widetilde{U}_r\right]$ and $\bar{\Sigma}_{r+k} = \begin{bmatrix} S_k & \\ & \widetilde{\Sigma}_r \end{bmatrix}$. Note that P is the same preconditioner as in (3.4) except that the perturbation is of rank $r + k$ now and the shifted skew-symmetric matrix ($\bar{\mathcal{S}}$) has a better spectrum; see Section 4.4.

4. Numerical results.

4.1. Implementation details for the numerical experiments. As a baseline of comparison, we implemented a robust general iterative scheme that was proposed in [3]. It uses the permutation and scalings given in (2.1) followed by a symmetric permutation. We use Reverse Cutthill-McKee (RCM) reordering since RCM reordered matrices have better robustness in subsequent applications of ILU-type preconditioners [3]. After the symmetric permutation, we use ILU preconditioners with no fill-in (ILU(0)), with pivoting and threshold of 10^{-1} (ILUTP(10^{-1})) and 10^{-2} (ILUTP(10^{-2})), of MATLAB. We call this method *MPS-RCM*, and it is implemented in MATLAB R2018a.

Our new method is also implemented in MATLAB R2018a in two stages: preprocessing and iterative solution. In the preprocessing stage, we obtain the sparse ASSS preconditioner, where we just need the coefficient matrix to obtain the permutation and scalings by calling HSL-MC64 via its MATLAB interface. This is followed by solving the LLS problem in (2.4), which we do directly via the MATLAB backslash operation. Then, we compute an incomplete Bunch-Kaufman-Parlett factorization of $\widehat{M}$ via the MATLAB interface of the *sym-ildl software package* [15]. We use its default parameters except that we disable any further scalings. Similar to ILUTP, we use thresholds of 10^{-1} and 10^{-2} and allow any fill-in, and, similar to ILU(0), we allow as many nonzeros as the original matrix per column with no threshold-based dropping. We call these methods ILDL(10^{-1}), ILDL(10^{-2}), and $\sim$ILDL(0), respectively. We compute the low-rank factorization in (3.3) and apply a few steps of the skew-Lanczos process to deflate the shifted skew-symmetric part of the coefficient matrix. Finally, we iteratively solve the shifted skew-symmetric linear system of equations that arise in (3.8) with multiple right-hand side vectors via MRS,

$$\bar{\mathcal{S}}X = \bar{U}_{r+k},$$

and form the $(r + k) \times (r + k)$ dense matrix

$$\bar{\Sigma}_{r+k} + \bar{U}_{r+k}^T\bar{\mathcal{S}}^{-1}\bar{U}_{r+k}$$

explicitly. All of these preprocessing steps do not require the right-hand side vector, and they are done only once if a sequence of linear systems with the same coefficient matrix but with different right-hand side vectors needs to be solved.

After preprocessing, the linear system of equation in (3.2) is solved via a Krylov subspace method with the preconditioner in (3.7). At each iteration of the Krylov subspace method, the inverse of the preconditioner is applied as in (3.8). This requires the solution of a shifted skew-symmetric linear system. We use the MRS method for those shifted skew-symmetric systems.

For the outer Krylov subspace method, some alternatives are BiCGSTAB, GMRES, and TFQMR, even though they often behave almost the same [3], GMRES requires a restart parameter that defies our objective toward obtaining a *black-box* solver and BiCGSTAB has erratic convergence. Alternatively, TFQMR has a smoother convergence and does not require restarting. We observe that TFQMR can stagnate, which is also noted in [36]. Therefore, as a challenge for our new approach, we use TFQMR for both our proposed scheme and MPS-RCM. The stopping criterion for TFQMR is set to 10^{-5}, and for the inner MRS iterations of the proposed scheme, we use the same stopping criterion. The right-hand side is determined from the solution vector of all ones.

We note that even though we use MATLAB's built-in functions as much as possible in implementing the proposed iterative scheme, MPS-RCM is entirely using the built-in functions of MATLAB or efficient external libraries. Therefore, no fair comparison in terms of the running times in MATLAB is currently possible. An efficient and parallel implementation of the proposed scheme requires a lower-level programming language such as C/C++ due to the low-level algorithmic and data structural details that need to be addressed efficiently. For example, the proposed scheme needs efficient accessing of rows and columns of a sparse matrix. At first glance, one might tend to store the matrix both in Compressed Sparse Row and Column formats, however, this approach is doubling the memory requirement. Therefore, a new storage scheme without much increase in the memory requirements is needed. Also, efficient and parallel implementation of sparse matrix-vector multiplications, where the coefficient matrix is symmetric (and shifted skew-symmetric) and parallel sparse triangular backward/forward sweeps are challenging problems. These are still active research areas by themselves [1, 6]. Therefore, we leave these issues as future work and focus on the robustness of the proposed scheme in MATLAB.

4.2. Test problems. In this section we give the selection criterion and describe the matrices that we use for numerical experiments. MPS-RCM makes incomplete LU-based preconditioned iterative solvers very robust. Therefore, to identify the most challenging problems, we use MPS-RCM to choose a highly indefinite and challenging set of 10 problems from the SuiteSparse Matrix Collection [7], in which at least one instance of MPS-RCM fails due to failure of incomplete factorization or stagnation of the Krylov subspace method. Properties of the test problems and their sparsity plots are given in Table 4.1 and Figure 4.1, respectively. All chosen problems are (numerically) nonsymmetric, and only a few of them are structurally symmetric. Here, structural symmetry means that if the element (i, j) is nonzero, then the element (j, i) is also nonzero. (The sparsity pattern of the matrix is symmetric.) Numerical symmetry implies structural symmetry but not vice-versa.

The matrices Bp_200 and bp_600 come from a sequence of simplex basis matrices in Linear Programming. West0989 and west1505 arise in a chemical engineering plant model with seven and eleven stage column sections, respectively. Rajat19 is a circuit simulation problem. Rdb1250l, rdb3200l, and rdb5000 occur in a reaction-diffusion Brusselator model. Chebyshev2 is an integration matrix using the Chebyshev method for solving fourth-order semilinear initial boundary value problems, and finally, Orani678 arises in the economic modeling of Australia.

4.3. Effectiveness of the shifted skew-symmetrizer. The structure of the approximate skew-symmetrizer (S) can be arbitrary. We experimented with a simple diagonal (S_d) and a tridiagonal (S_t) structure. In Table 4.2, the dimensions and the number of nonzeros for the LLS problem in (2.4) are given. After that, we obtain a shifted skew-symmetrized matrix

TABLE 4.1
Size (n), number of nonzeros (nnz), structural symmetry (Struct. S.), numerical symmetry (Num. S.), and the problem domains of the test problems.

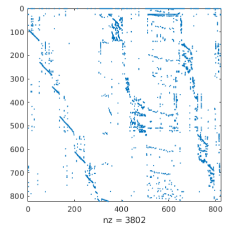
Matrix	n	nnz	Struct. S.	Num. S.	Problem Domain
bp_200	822	3802	–	–	Optimization
bp_600	822	4172	–	–	Optimization
west0989	989	3518	–	–	Chemical process simulation
rajat19	1157	3699	–	–	Circuit simulation
rdb1250l	1250	3802	✓	–	Computational fluid dynamics
west1505	1505	5414	–	–	Chemical process simulation
chebyshev2	2053	18 447	–	–	Structural
orani678	2529	90 158	–	–	Economics
rdb3200l	3200	18 880	✓	–	Computational fluid dynamics
rdb5000	5000	29 600	✓	–	Computational fluid dynamics

($\widehat{A}$). To evaluate the effectiveness of the scaling and permutation followed by the approximate skew-symmetrizer, we use three metrics: the skew-symmetry of the off-diagonals, the distance of the main diagonal to the identity, and the condition number. In Table 4.3, we depict these for the original matrix, for matrices after MC64 scaling and permutation, and for those followed by applying S_d or S_t, which we call "Original", "MC64", "MC64+S_d", and "MC64+S_t", respectively. As expected, for most cases MC64+S_t has successfully improved the skew-symmetry of the off-diagonal part compared to the original matrix. One exception is chebyshev2, which has a condition number of order 10^{15}, and MC64+S_t has improved both the condition number and the main diagonal. For all test problems, MC64+S_t has improved the main diagonal, and for 8 of 10 cases, it has also improved the condition number compared to the original matrix. In all cases, S_t improves the skew-symmetry of the off-diagonal part and the diagonal compared to S_d except chebyshev2. The condition number becomes worse for 6 cases out of 10 using S_t. However, this is not an issue since we further precondition the system in our proposed method.

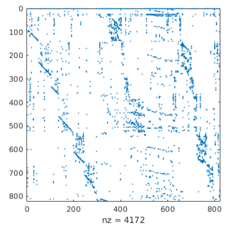
The spectra of the original, reordered, and skew-symmetrized matrices are given in Figure 4.2. S_t (shown in red) does a better job moving the real part of most eigenvalues to the right half complex plane and clustering them around one, compared to S_d. Therefore, in the following numerical experiments we use S_t.

4.4. Effectiveness of the deflation. In Figure 4.3, we present the spectrum of the original $I + \widetilde{\mathcal{J}}$ (by using incomplete LDL^T factorization of $\widehat{H}$ with zero fill-in) and of the deflated shifted skew-symmetric $I + \bar{\mathcal{J}}$ after 10, 20, and 50 iterations of the skew-Lanczos process for all test matrices. We note that the scale of the real axis is negligible for all cases and the spectrum is purely imaginary. Even though case-by-case fine-tuning is possible by looking at the spectra, we observe that deflating with 20 vectors gives a meaningful balance between the number of iterations and the orthogonality between the Lanczos vectors for the test problems since for 50 vectors the spectrum is worse and for 10 vectors it does not improve it as much as for 20 vectors. Therefore, in the following experiments, we use 20 skew-Lanczos vectors.

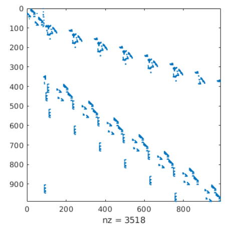
4.5. Iterative solution of general sparse linear systems. In the following numerical experiments we use the proposed method as described earlier. For the proposed method, the number of Lanczos vectors is set to 20. Compared to the nondeflated version of the proposed method, the number of MRS iterations is $30.8\,\%$ lower on average when the inner system is deflated with 20 Lanczos vectors for the test problems. We have also experimented with the skew-Lanczos method with full reorthogonalization for 20 Lanczos vectors and the



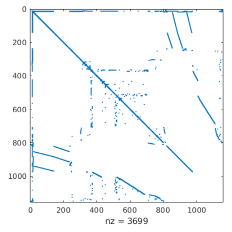
(a) bp_200



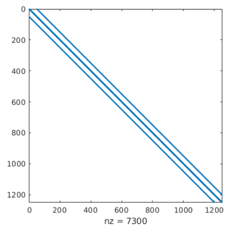
(b) bp_600



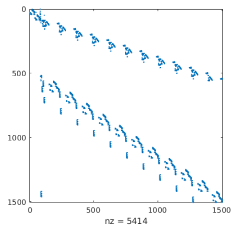
(c) west0989



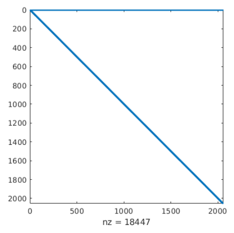
(d) rajat19



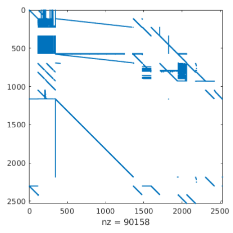
(e) rdb1250l



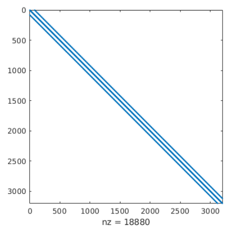
(f) west1505



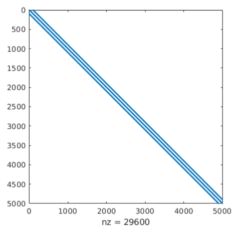
(g) chebyshev2



(h) orani678



(i) rdb3200l



(j) rdb5000

FIG. 4.1. *Sparsity structures of the test matrices.*

improvement in the number of iterations is negligible for the test problems. Since it is used just as a preconditioner, we believe that in general it may not be necessary in the skew-Lanczos process to fully or partially reorthogonalize. Therefore, we only report the proposed method with deflation using 20 Lanczos vectors without any reorthogonalization.

In Table 4.4, the exact ranks of $\widetilde{\mathcal{M}}_r$ for various incomplete factorizations as well as the percentages of the rank with respect to the matrix dimension are given. For the set of test problems, the largest one is 56, which is only 3.7 % of the matrix dimension. They are roughly invariant for incomplete factorization for all problems except rdbl1250l. For the cases when the rank is zero, we use the preconditioner in (3.5); otherwise we use the preconditioner in (3.4).

TABLE 4.2
Number of rows ($nz+n$), columns ($nnz(S)$), and nonzeros (nnz) of the coefficient matrix in the LLS problem (2.4) associated with the test problems.

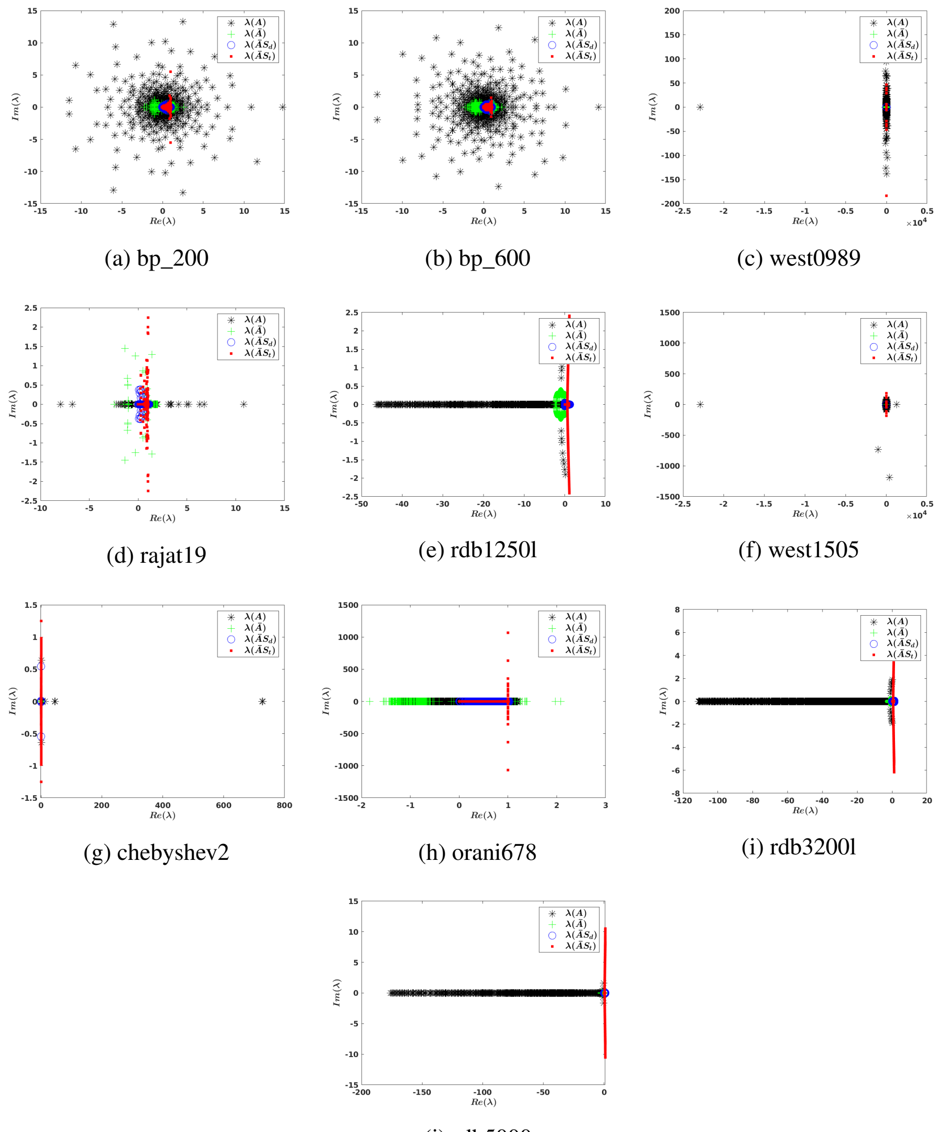
	S_d			S_t		
Matrix	$nz+n$	$nnz(S_d)$	nnz	$nz+n$	$nnz(S_t)$	nnz
bp_200	3727	822	3802	8131	2464	11 404
bp_600	4007	822	4172	9604	2464	12 514
west0989	3491	989	3518	7529	2965	10 549
rajat19	3425	1157	3699	7422	3469	11 095
rdb1250l	4275	1250	7300	8570	3748	21 892
west1505	5382	1505	5414	11 598	4513	16 237
chebyshev2	14 344	2053	18 447	20 481	6157	55 331
orani678	87 358	2529	90 158	11 852	7585	270 465
rdb3200l	11 040	3200	18 880	22 115	9598	56 632
rdb5000	17 300	5000	29 600	34 645	14 998	88 792

We have also experimented with the preconditioner in (3.5) for problems where the rank is not zero but relatively small. However, the number of iterations has increased significantly even for the case where the rank is equal to one (rdbl3200l with ILDL(10^{-2})). We have tried to solve the LLS problem with the regularization parameters (2.5) $\gamma = 0.1, 1$, and 10. Since $\gamma = 0.1$ and 10 are worse in terms of the number of iterations, we only report the results with $\gamma = 1$.

In Table 4.5, the number of TFQMR iterations for the proposed method and MPS-RCM are given. For the proposed method, we also give the average number of inner iterations in parenthesis. As seen in the table, for all test problems and regardless of the choice of the dropping tolerance, the proposed method succeeds. On the other hand, MPS-RCM fails for $80\,\%$, $90\,\%$, and $20\,\%$ of the problems using ILU(0), ILUTP(10^{-1}), and ILUTP(10^{-2}), respectively. For ILU(0) the majority of the failures are due to TFQMR stagnating. For ILUTP(10^{-1}) it happens because of a zero pivot is encountered during factorization. As expected, when the dropping tolerance decreases to 10^{-2}, the number of failures decreases since the incomplete factorization is more like a direct solver. Hence, MPS-RCM becomes more robust. When it does not fail, the required number of TFQMR iterations for MPS-RCM is quite low except for rdb5000. The proposed method, on the other hand, is robust regardless of the quality of the incomplete factorization for the test problems. There are no failures during the incomplete factorization and no failures of the iterative scheme. The required number of TFQMR iterations improves as a more accurate incomplete factorization is used. The number of iterations if ILDL(10^{-2}) is used are comparable to those obtained by MPS-RCM with ILUTP(10^{-2}). Except for two cases (rdb3200l and rdb5000) for which the proposed method is significantly better and for two other cases (orani678 and rdb1250l) for which MPS-RCM is significantly better. For the proposed method, the number of average inner MRS iterations is not dependent on the choice of the incomplete factorization for all test problems except for rdb1250l; in this case, it is almost halved when ILDL(10^{-1}) is used compared to ~ILDL(0) and ILDL(10^{-2}). We believe this is due to the fact that for the same matrix using ILDL(10^{-1}), the rank of $\widetilde{\mathcal{M}}_r$ is 1, which is much smaller than those of ~ILDL(0) and ILDL(10^{-1}) (36 and 26, respectively).

5. Large scale problems.

We have introduced a general-purpose black-box scheme and shown that it is robust. There are, however, additional storage and computational costs associated with forming and solving a large LLS problem for computing ASSS, and depending



(a) bp_200 (b) bp_600 (c) west0989

(d) rajat19 (e) rdb1250l (f) west1505

(g) chebyshev2 (h) orani678 (i) rdb3200l

(j) rdb5000

FIG. 4.2. *Spectrum of the original matrix after applying MC64 and shifted skew-symmetrizers.*

on the sparsity structure of the preconditioner, ASSS might introduce additional fill-in. These costs can be amortized not only if multiple linear systems with the same coefficient matrix have to be solved, but also significantly reduced for problems in which the coefficient matrix is shifted skew-symmetric or very close to shifted skew-symmetric form. To illustrate this point, we study the effectiveness of the proposed method for large-scale problems that arise in interior-point optimizations. The matrices are obtained from the SuiteSparse Matrix Collection [7] from a matrix set called Schenk_I BMNA. These systems arise in damped Newton iteration while solving an interior-point optimization problem [28]. The coefficient matrices of the

TABLE 4.3

The effect of MC64 and the shifted skew-symmetrizer. In the table Skew-symmetry, Diagonal and Cond denote $\|(X - X^T)/2\|_F/\|X - \mathcal{D}(X)\|_F$, $\|\mathcal{D}(X) - I\|_F$ *and the condition number of* X*, respectively and rounded to one decimal place where* X *is either the original matrix after applying MC64, MC64 followed by* S_d *or MC64 followed by* S_t *in which* S_d *and* S_t *are diagonal and tridiagonal shifted skew-symetrizers, respectively.*

Matrix		Original	MC64	MC64+S_d	MC64+S_t
bp_200	Skew-symmetry	70.7 %	71 %	70.7 %	86.2 %
	Diagonal	28.7	35.8	12.2	11.6
	Cond	6.4×10^6	4.0×10^2	4.8×10^2	1.0×10^4
bp_600	Skew-symmetry	70.7 %	70.7 %	71.3 %	74.8 %
	Diagonal	28.7	36.6	13.2	12.7
	Cond	1.5×10^6	3.2×10^2	3.5×10^2	7.8×10^3
west0989	Skew-symmetry	70.7 %	70.7 %	70.7 %	100 %
	Diagonal	2.3×10^4	28.8	13.4	12.6
	Cond	9.9×10^{11}	6.7×10^3	8.3×10^3	9.3×10^5
rajat19	Skew-symmetry	28.4 %	67.1 %	68.1 %	83.4 %
	Diagonal	33.9	29.5	11.9	9.8
	Cond	1.1×10^{10}	2.3×10^{10}	1.1×10^{11}	5.7×10^{10}
rdb1250l	Skew-symmetry	49 %	56.2 %	55.4 %	97.9 %
	Diagonal	690.8	70.7	17.2	9.8
	Cond	4.7×10^2	4.9×10^2	3.6×10^2	3.1×10^2
west1505	Skew-symmetry	70.7 %	70.7 %	70.7 %	100 %
	Diagonal	2.3×10^4	35.9	16.6	15.7
	Cond	1.6×10^{12}	8.8×10^3	1.1×10^4	1.2×10^6
chebyshev2	Skew-symmetry	70.7 %	11.8 %	7.6 %	37.4 %
	Diagonal	898.5	3.5	21.9	21.9
	Cond	5.5×10^{15}	8.6×10^9	2.9×10^{10}	1.5×10^{10}
orani678	Skew-symmetry	70.7 %	70.8 %	70.6 %	100 %
	Diagonal	53.5	97.5	15.2	15.1
	Cond	9.6×10^3	7.5×10^3	1.2×10^4	6.4×10^6
rdb3200l	Skew-symmetry	21.9 %	27.9 %	27.1 %	99.7 %
	Diagonal	2.5×10^3	113.1	20.6	12
	Cond	1.1×10^3	9×10^2	8.2×10^2	7.3×10^2
rdb5000	Skew-symmetry	14.4 %	18.3 %	17.9 %	99.9 %
	Diagonal	4.8×10^3	141.4	24.6	14.1
	Cond	4.4×10^3	3×10^3	2.8×10^3	3.6×10^3

linear systems (3.1) are symmetric indefinite of size $n \times n$ with the 2×2 block structure [28]

$$A = \begin{bmatrix} D_1 & B \\ B^T & -D_2 \end{bmatrix},$$

where $D_1 \in \mathbb{R}^{n_1 \times n_1}$ and $D_2 \in \mathbb{R}^{n_2 \times n_2}$ are diagonal positive definite and $B \in \mathbb{R}^{n_1 \times n_2}$. These systems are not easy to solve neither directly [28] nor iteratively [17]. We select a representative subset that consists of the three largest problems, and their properties are given in Table 5.1. A sparsity plot of the largest coefficient matrix (c-big) is provided in

TABLE 4.4
Rank of $\widetilde{\mathcal{M}}_r$ and its percentage with respect to the matrix dimension ($\frac{r}{n} \times 100$) rounded to one decimal place in parenthesis.

Matrix	∼ILDL(0)	ILDL(10^{-1})	ILDL(10^{-2})
bp_200	33 (4 %)	32 (3.8 %)	32 (3.8 %)
bp_600	48 (5.8 %)	46 (5.6 %)	45 (5.5 %)
west0989	35 (3.5 %)	32 (3.2 %)	35 (3.5 %)
rajat19	38 (3.3 %)	38 (3.3 %)	38 (3.3 %)
rdb1250l	36 (2.9 %)	1 (0.1 %)	26 (2.1 %)
west1505	54 (3.6 %)	55 (3.7 %)	56 (3.7 %)
chebyshev2	5 (2 %)	7 (0.3 %)	7 (0.3 %)
orani678	19 (0.8 %)	26 (1 %)	28 (1.1 %)
rdb3200l	0 (0 %)	0 (0 %)	1 (0 %)
rdb5000	0 (0 %)	0 (0 %)	0 (0 %)

TABLE 4.5
Number of TFQMR iterations. The average number of inner MRS iterations for the proposed method rounded to one decimal place is given in parenthesis. ∗: TFQMR stagnated, †: TFQMR reached the maximum number of iterations (2000) without reaching the required relative residual, ‡: zero pivot is encountered during factorization.

	Proposed method			**MPS-RCM**		
Matrix	∼ILDL(0)	ILDL(10^{-1})	ILDL(10^{-2})	ILU(0)	ILUTP(10^{-1})	ILUTP(10^{-2})
bp_200	26(53.6)	88(54.4)	3(55.8)	†	‡	2
bp_600	15(65.7)	33(66.6)	4(67)	†	‡	2
west0989	74(134.6)	45(106.9)	3(105.5)	∗	20	1
rajat19	10(35.6)	46(35.8)	17(35.2)	‡	‡	‡
rdb1250l	90(66.7)	10(37.1)	49(71.8)	∗	‡	10
west1505	57(162.9)	75(166.9)	1(163)	†	‡	1
chebyshev2	10(15.9)	15(12)	14(15.6)	1	‡	‡
orani678	15(229.3)	13(235.9)	12(249)	15	‡	4
rdb3200l	9(63.6)	15(61)	3(69.1)	∗	‡	20
rdb5000	8(100.6)	16(101.5)	6(100.9)	∗	†	129

Figure 5.1; all matrices in the set have a similar sparsity structure. The application of the ASSS preconditioner is strongly simplified by multiplying both sides of (3.1) from the left by

$$\begin{bmatrix} I_{n_1 \times n_1} & 0 \\ 0 & -I_{n_2 \times n_2} \end{bmatrix}$$

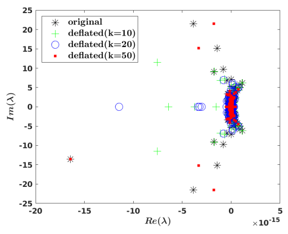
and obtaining the modified system $\widehat{A}\widehat{x} = \widehat{b}$, where $\widehat{x} = x$,

$$\widehat{A} = \begin{bmatrix} D_1 & B \\ -B^T & D_2 \end{bmatrix},$$

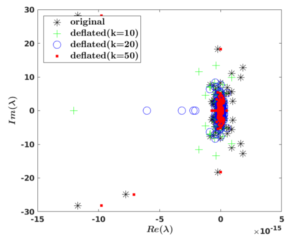
and, if $b^T = [b_1^T, b_2^T]$ (b_1 and b_2 are vectors of size n_1 and n_2, respectively), then $\widehat{b}^T = [b_1^T, -b_2^T]$. As in Section 3, we consider splitting the coefficient matrix into it symmetric and skew-symmetric parts $\widehat{A} = \widehat{M} + \widehat{J}$, where

$$\widehat{M} = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}, \qquad \widehat{J} = \begin{bmatrix} 0 & B \\ -B^T & 0 \end{bmatrix}.$$

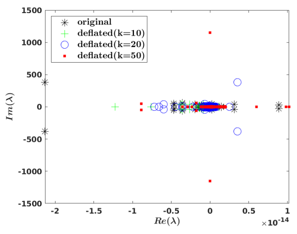
Here, $\widehat{M}$ is not only positive definite but also a diagonal matrix. The rest of the proposed method proceeds as described in Section 3, except that, with $\widehat{M}$ being a diagonal matrix, one



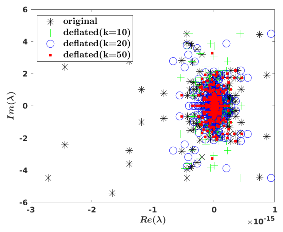
(a) bp_200



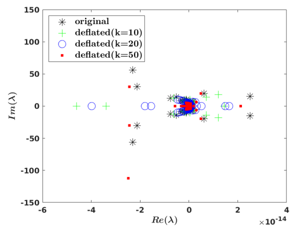
(b) bp_600



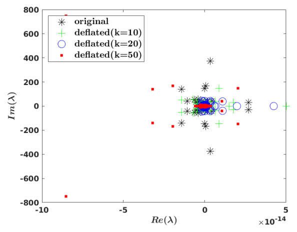
(c) west0989



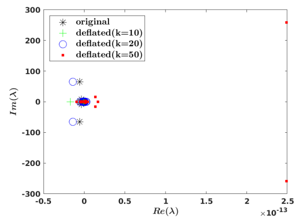
(d) rajat19



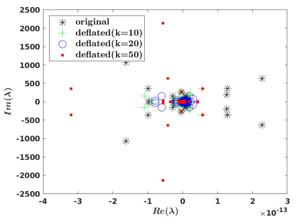
(e) rdb1250l



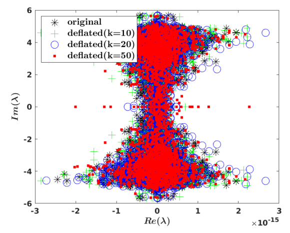
(f) west1505



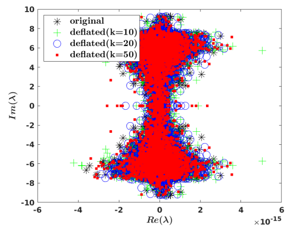
(g) chebyshev2



(h) orani678



(i) rdb3200l



(j) rdb5000

FIG. 4.3. *Spectrum of the original skew-symmetric matrix and after deflation.*

can compute its full Cholesky factorization $\widehat{M} = \widehat{L}\widehat{L}^T$ with a cost of only $O(n)$ arithmetic operations. Then, we solve the modified system $(I + \widehat{L}^{-1}\widehat{J}\widehat{L}^{-T})\widehat{L}^T\widehat{x} = \widehat{L}^{-1}\widehat{b}$ iteratively via TFQMR using the preconditioner $\widehat{P} = I + \widehat{L}^{-1}\widehat{J}\widehat{L}^{-T}$, where the systems involving the preconditioner are solved via MRS. Since the preconditioner is exact, instead of deflating to improve the convergence rate, we relax the inner stopping criterion and stop the inner iterations if the norm of the relative residual is smaller than ϵ_{in}. For comparison, we use MPS-RCM with ILU and ILUTP preconditioners (as described in Section 4.1). Since the coefficient matrix is symmetric, we use an incomplete LDL^T factorization as preconditioner; we call this method *ILDL*. For the incomplete LU and LDL^T factorizations, we use the MATLAB implementation

TABLE 5.1
Size (n), block sizes (n_1 and n_2), and number of nonzeros (nnz) of the large problems.

Matrix	n	n_1	n_2	nnz
c-big	345 241	201 877	143 364	2 340 859
c-73	169 422	86 417	83 005	1 279 274
c-73b	169 422	86 417	83 005	1 279 274

and the *sym-ildl package* [15], respectively, with their default parameters, but we vary the dropping threshold. We use the same stopping criterion of 10^{-5} as in Section 4.1 for the outer (TFQMR) iterations for all solvers. We obtain the following results on a server with 4 $\times$ 2.3 GHz AMD Opteron 6376 processors, where each processor has 16 cores (64 cores total) with a total 128 GB of memory and using MATLAB R2019b. We obtain the timing results as the sole user of the server. In Table 5.2, the number of iterations are given for the proposed method, MPS-RCM and ILDL. The methods MPS-RCM and ILDL fail in 5 cases each, while our new method converges in all cases even when the inner stopping tolerance is 10^{-1}. The failure of MPS-RCM is due to encountering a zero pivot during the incomplete factorization, and for ILDL it is due to either TFQMR stagnating or reaching the maximum number of iterations (10 000) without converging. MPS-RCM requires fewer iterations than ILDL. The amount of additional storage required by each solver for the full and incomplete factors are given in Table 5.3, where MPS-RCM requires more memory compared to ILDL, while the new method does not need any additional incomplete factorizations for these problems. For comparison, we include a direct solver (sparse full LDL^T factorization) that is available in MATLAB. It is chosen automatically by the MATLAB backslash operation if the input matrix is sparse symmetric and indefinite. Before computing the sparse LDL^T factorization, we apply the MATLAB symmetric approximate minimum degree reordering. We also compare the required solution time for each of these methods in Table 5.4. Total run times (including any reordering, scaling, factorization, and iterative or triangular solution time) reported in the table are the average of 5 runs. Generally, the run times are proportional to the number of nonzeros in the factors and the number of iterations except for MPS-RCM with ILUPT(10^{-2}), where the number of nonzeros of the incomplete factors are lower for c-73 compared to c-73b, but the required solution time is higher since the computation of the incomplete factorization requires more time even though the resulting factors are sparser. As expected, the direct solver is faster for smaller problems, while for the largest problem (c-big) the new method is the fastest.

6. Conclusions.

A robust two-level iterative scheme has been presented for solving general sparse linear system of equations. The robustness of the scheme is shown on challenging matrices that arise in various problems which are obtained from the SuiteSparse Matrix Collection. While it requires some additional preprocessing steps, the results presented in this paper indicate that the proposed scheme significantly improves the robustness of iterative methods for general sparse linear systems compared to existing methods. This result holds irrespectively of the quality of the incomplete factorization, even for a challenging set of test problems. The proposed scheme requires some additional memory, but this is shown to be kept within a small percentage of the problem size. Furthermore, for large-scale systems that arise in interior-point optimizations, we show that the memory requirements of the proposed scheme can be reduced without any degradation of its robustness. We believe that with the introduction of the proposed scheme, iterative solvers will become much more viable alternatives for solving problems in chemical engineering, optimizations, economics, etc. Its efficient parallel implementation requires addressing some algorithmic challenges which we leave as future

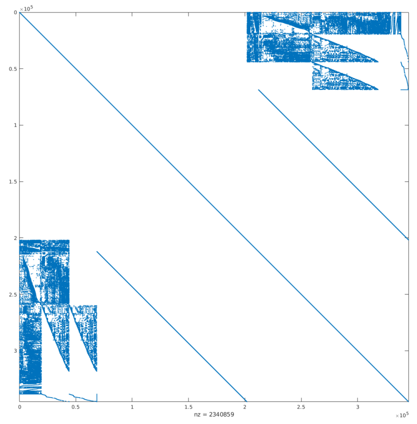


FIG. 5.1. *Sparsity structure of c-big.*

TABLE 5.2
Number of TFQMR iterations. The average number of inner MRS iterations for the proposed method rounded to one decimal place is given in parenthesis. $*$: TFQMR stagnated, †: TFQMR reached the maximum number of iterations (10 000) without reaching the required relative residual, ‡: zero pivot is encountered during factorization.

		Matrix	c-big	c-73	c-73b
New method	$\epsilon_{in} = 10^{-1}$		$2(786.4)$	$2(855.2)$	$2(2,139.4)$
	$\epsilon_{in} = 10^{-2}$		$1(2570.3)$	$1(2102)$	$2(4681.5)$
	$\epsilon_{in} = 10^{-3}$		$1(3856.5)$	$1(3325)$	$1(4885.5)$
MPS-RCM	ILU(0)		‡	‡	‡
	ILUTP(10^{-1})		56	‡	‡
	ILUTP(10^{-2})		4	29	30
ILDL	ILDL(10^{-1})		†	†	†
	ILDL(10^{-2})		1771	$*$	960
	ILDL(10^{-3})		2553	$*$	303

TABLE 5.3
Number of nonzeros of the full and incomplete LU and LDL^T factors (‡: indicates that a zero pivot is encountered during factorization).

		Matrix	c-big	c-73	c-73b
MPS-RCM	ILU(0)		‡	‡	‡
	ILUTP(10^{-1})		2 865 786	‡	‡
	ILUTP(10^{-2})		8 744 791	3 734 670	6 822 622
ILDL	ILDL(10^{-1})		1 421 973	944 198	773 600
	ILDL(10^{-2})		3 299 471	1 224 634	1 103 864
	ILDL(10^{-3})		4 267 310	1 254 394	1 278 301
Direct	LDL^T		37 378 804	2 143 127	1 919 452

TABLE 5.4
Total time (seconds) rounded to one decimal place. for each entry the method is executed 5 times and the average running time is reported. Numbers in bold indicate the best time. ∗: TFQMR stagnated, †: TFQMR reached the maximum number of iterations (10 000) without reaching the required relative residual, ‡: zero pivot is encountered during factorization.

		Matrix	c-big	c-73	c-73b
New method	$\epsilon_{in} = 10^{-1}$		**55.7**	34.6	85.3
	$\epsilon_{in} = 10^{-2}$		112.9	47.9	139.9
	$\epsilon_{in} = 10^{-3}$		115.2	51.5	74.4
MPS-RCM	ILU(0)		‡	‡	‡
	ILUTP(10^{-1})		62.8	‡	‡
	ILUTP(10^{-2})		1162.4	105.4	33.8
ILDL	ILDL(10^{-1})		†	†	†
	ILDL(10^{-2})		2092.4	∗	106.7
	ILDL(10^{-3})		2979.0	∗	39.9
Direct	LDL^T		65.5	**2.0**	**2.0**

work.

Acknowledgments. We thank two anonymous referees and the editor Michiel E. Hochstenbach for their constructive comments and suggestions.

Appendix A. Shifted skew-symmetrizer MATLAB code.

```
function S = skew_symmetrize_sparse_right(A, n, gamma, S)

AA = spones(A) * spones(S);

U = triu(AA + AA',1);

neq = nnz(U);

[I, J] = find(U);

II = find(S);

nu = length(II);

S(II) = [1:nu];

B = sparse(neq + n, nu);

for i = 1:neq
    ASij = spones(A(I(i), :))'.* spones(S(:,J(i)));
    ASji = spones(A(J(i), :))'.* spones(S(:,I(i)));

    INDij = find(ASij);
    B(i, S(INDij, J(i))) = A(I(i), INDij);

    INDji = find(ASji);
    B(i, S(INDji, I(i))) = A(J(i), INDji);
end

for i = 1:n
    ASii = spones(A(i, :))'.* spones(S(:,i));
    INDii = find(ASii);
    B(i + neq, S(INDii, i)) = sqrt(gamma) * A(i, INDii);
end

```

```
v = [zeros(neq, 1); sqrt(gamma) * ones(n, 1)];

x = B \ v;

S(II) = x;

return
```

Appendix B. Shifted skew-symmetric iterative solver for multiple right-hand side vectors based on simultaneous MRS iterations.

```
function [x, its, res, relres_hist] = mrs(alpha, S, b, maxit, tol)

[n, nrhs] = size(b)

x = zeros(n, nrhs);
r = b;

for j = 1:nrhs
    r0(j) = norm(r(:, j), 2);
    relres_hist(j) = [r0(j) / r0(j)];
    s(j) = r0(j);
    q(:, j) = r(:, j) / s(j);
    beta(j) = 0;
    theta1(j) = alpha;
    c_old(j) = 1;
    s_old(j) = 0;
    delta(j) = 0;
    delta_old(j) = 0;
end
q_old = zeros(n, nrhs);
p_old = zeros(n, nrhs);
p_old2 = zeros(n, nrhs);

for i = 1:maxit
    q_new = S * q + q_old * diag(beta);
    q_old = q;
    for j = 1:nrhs
        beta(j) = norm(q_new(:, j), 2);
        if (beta(j) = 0)
            q(:, j) = q_new(:, j) / beta(j)
        end
    end
    theta = sqrt(theta1.* theta1 + beta.* beta);
    c_k = theta1./ theta;
    s_k = beta./ theta;
    delta = - s_old.* beta;
    theta1 = c_old.* theta;

    p = (q_old - p_old2 * diag(delta_old)) * diag(1./ theta);

    x = x + p * diag(s) * diag(c_k);
    s = - s.* s_k;

    relres_hist = [relres_hist; abs(s)./ r0];

    if (max(abs(s)./ r0) < tol)
        its = i
        for j = 1:nrhs
            res(j) = norm(b(:, j) - alpha * x(:, j) - S * x(:, j), 2);
        end
        relres = res./ r0;
        break;
    end

    p_old2 = p_old;
    p_old = p;
    s_old = s_k;
    c_old = c_k;
    delta_old = delta;
end
```

REFERENCES

[1] C. ALAPPAT, A. BASERMANN, A. R. BISHOP, H. FEHSKE, G. HAGER, O. SCHENK, J. THIES, AND G. WELLEIN, *A recursive algebraic coloring technique for hardware-efficient symmetric sparse matrix-vector multiplication*, ACM Trans. Parallel Comput., 7 (2020), Art. No. 19, 37 pages.

[2] C. BEATTIE, V. MEHRMANN, H. XU, AND H. ZWART, *Linear port-Hamiltonian descriptor systems*, Math. Control Signals Systems, 30 (2018), Art. No. 17, 27 pages.

[3] M. BENZI, J. C. HAWS, AND M. TŮMA, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[4] J. R. BUNCH, L. KAUFMAN, AND B. N. PARLETT, *Decomposition of a symmetric matrix*, Numer. Math., 27 (1976), pp. 95–109.

[5] P. CONCUS AND G. H. GOLUB, *A generalized conjugate gradient method for nonsymmetric systems of linear equations*, in Computing Methods in Applied Sciences and Engineering, Part 1, Lecture Notes in Econom. and Math. Systems, Vol. 134, Springer, Berlin, 1976, pp. 56–65.

[6] I. ÇUĞU AND M. MANGUOĞLU, *A parallel multithreaded sparse triangular linear system solver*, Comput. Math. Appl., 80 (2020), pp. 371–385.

[7] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), Art. No. 1, 25 pages.

[8] F. DOPICO AND F. UHLIG, *Computing matrix symmetrizers, part 2: new methods using eigendata and linear means; a comparison*, Linear Algebra Appl., 504 (2016), pp. 590–622.

[9] I. S. DUFF AND J. KOSTER, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996.

[10] H. EGGER, *Structure preserving approximation of dissipative evolution problems*, Numer. Math., 143 (2019), pp. 85–106.

[11] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[12] R. W. FREUND AND N. M. NACHTIGAL, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[13] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[14] N. GRÄBNER, V. MEHRMANN, S. QURAISHI, C. SCHRÖDER, AND U. VON WAGNER, *Numerical methods for parametric model reduction in the simulation of disc brake squeal*, ZAMM Z. Angew. Math. Mech., 96 (2016), pp. 1388–1405.

[15] C. GREIF, S. HE, AND P. LIU, *SYM-ILDL: Incomplete LDLT factorization of symmetric indefinite and skew-symmetric matrices*, ACM Trans. Math. Software, 44 (2017), Art. No. 1, 21 pages.

[16] C. GREIF AND J. M. VARAH, *Iterative solution of skew-symmetric linear systems*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 584–601.

[17] M. HAGEMANN AND O. SCHENK, *Weighted matchings for preconditioning symmetric indefinite linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 403–420.

[18] HSL, *A collection of Fortran codes for large scale scientific computation*, 2007. http://www.hsl.rl.ac.uk.

[19] R. IDEMA AND C. VUIK, *A minimal residual method for shifted skew-symmetric systems*, Tech. Rep. 07-09, Delft University of Technology, Delft, 2007.

[20] B. JACOB AND H. J. ZWART, *Linear Port-Hamiltonian Systems on Infinite-Dimensional Spaces*, Birkhäuser, Basel, 2012.

[21] H. JI AND Y. LI, *A breakdown-free block conjugate gradient method*, BIT, 57 (2017), pp. 379–403.

[22] E. JIANG, *Algorithm for solving shifted skew-symmetric linear system*, Front. Math. China, 2 (2007), pp. 227–242.

[23] M. MANGUOĞLU AND V. MEHRMANN, *A robust iterative scheme for symmetric indefinite systems*, SIAM J. Sci. Comput., 41 (2019), pp. A1733–A1752.

[24] M. OLSCHOWKA AND A. NEUMAIER, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.

[25] M. L. PARKS, E. DE STURLER, G. MACKEY, D. D. JOHNSON, AND S. MAITI, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 1651–1674.

[26] R. PORTASE AND B. UÇAR, *On matrix symmetrization and sparse direct solvers*, Research Report RR-8977, Inria-Research Centre Grenoble–Rhône-Alpes, November 2016.

[27] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[28] O. SCHENK AND K. GÄRTNER, *On fast factorization pivoting methods for sparse symmetric indefinite systems*, Electron. Trans. Numer. Anal., 23 (2006), pp. 158–179. http://etna.ricam.oeaw.ac.at/vol.23.2006/pp158-179.dir/pp158-179.pdf

[29] D. B. SZYLD AND O. B. WIDLUND, *Variational analysis of some conjugate gradient methods*, East-West J. Numer. Math., 1 (1993), pp. 51–74.

[30] B. UÇAR, *Heuristics for a matrix symmetrization problem*, in Parallel Processing and Applied Mathematics, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, eds., Lecture Notes in Computer Science vol. 4967, Springer, Berlin, 2008, pp. 718–727.

[31] A. J. VAN DER SCHAFT, *Port-Hamiltonian differential-algebraic systems*, in Surveys in Differential-Algebraic Equations. I, A. Ilchmann and T. Reis, eds., Differ.-Algebr. Equ. Forum, Springer, Heidelberg, 2013, pp. 173–226.

[32] A. J. VAN DER SCHAFT AND D. JELTSEMA, *Port-Hamiltonian systems theory: an introductory overview*, Foundations and Trends in Systems and Control, 1 (2014), pp. 173–378.

[33] H. A. VAN DER VORST, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[34] E. VECHARYNSKI AND A. V. KNYAZEV, *Absolute value preconditioning for symmetric indefinite linear systems*, SIAM J. Sci. Comput., 35 (2013), pp. A696–A718.

[35] O. WIDLUND, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.

[36] J. ZHANG, *Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications*, Comput. Methods Appl. Mech. Engrg., 189 (2000), pp. 825–840.