Received: 6 October 2017 | Revised: 29 January 2018 | Accepted: 25 February 2018
DOI: 10.1002/nla.2173

RESEARCH ARTICLE

WILEY

Preordering saddle-point systems for sparse LDL^T factorization without pivoting

Sangye Lungten[1] | Wil H.A. Schilders[1] | Jennifer A. Scott[2,3]

[1]Department of Mathematics and Computer Science, Centre for Analysis, Scientific Computing and Applications, Eindhoven University of Technology, Den Dolech 2 Postbus 513 5600 MB Eindhoven, The Netherlands

[2]STFC Rutherford Appleton Laboratory, Didcot, Oxfordshire, OX11 0QX, UK

[3]School of Mathematical, Physical and Computational Sciences, University of Reading, Reading, RG6 6AQ, UK

Correspondence
Sangye Lungten, Department of Mathematics and Computer Science, Centre for Analysis, Scientific Computing and Applications, Eindhoven University of Technology, Den Dolech 2 Postbus 513, 5600 MB Eindhoven, The Netherlands.
Email: s.lungten@tue.nl

Summary

This paper focuses on efficiently solving large sparse symmetric indefinite systems of linear equations in saddle-point form using a fill-reducing ordering technique with a direct solver. Row and column permutations partition the saddle-point matrix into a block structure constituting a priori pivots of order 1 and 2. The partitioned matrix is compressed by treating each nonzero block as a single entry, and a fill-reducing ordering is applied to the corresponding compressed graph. It is shown that, provided the saddle-point matrix satisfies certain criteria, a block LDL^T factorization can be computed using the resulting pivot sequence without modification. Numerical results for a range of problems from practical applications using a modern sparse direct solver are presented to illustrate the effectiveness of the approach.

KEYWORDS

fill-reducing ordering, LDL^T factorization, saddle-point systems, sparse symmetric indefinite matrices

1 | INTRODUCTION

Our interest lies in solving large sparse symmetric indefinite systems of equations in saddle-point form

$$Kz = b, \quad K = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix}, \quad z = \begin{bmatrix} x \\ y \end{bmatrix}, \quad b = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{1}$$

where the (1,1) block A is an $n \times n$ symmetric positive definite (SPD) matrix, the (2,1) block B is an $m \times n$ matrix of full row rank with $m < n$, the (2,2) block C is an $m \times m$ symmetric positive semidefinite (SPSD) matrix (including the case $C = 0$), z is the solution vector, and b is given. In this paper, we focus on the case where B can be permuted to the trapezoidal form

$$P_r^T B P_c = [B_1 \;\; B_2], \tag{2}$$

where B_1 is an $m \times m$ nonsingular upper triangular matrix, and P_r and P_c are $m \times m$ and $n \times n$ permutation matrices, respectively. Define

$$P_K = \begin{bmatrix} P_c^T & \\ & P_r^T \end{bmatrix},$$

and then premultiplying K by P_K and postmultiplying by P_K^T, we obtain another system of the form (1), given by

$$\tilde{K}\tilde{z} = \tilde{b}, \quad \tilde{K} = \begin{bmatrix} P_c^T A P_c & P_c^T B^T P_r \\ P_r^T B P_c & -P_r^T C P_r \end{bmatrix}, \quad P_K^T \tilde{z} = z, \quad P_K^T \tilde{b} = b. \tag{3}$$

It is this system that we then solve. For simplicity of notation, we omit the ˜ in our discussion.

Linear systems of saddle-point type arise in a wide variety of applications throughout computational science and engineering, and their efficient solution has been the study of extensive research. A comprehensive review of the work done prior to 2005 is given by Benzi et al.,[1] and because of the ubiquitous nature of saddle-point systems and the challenges they pose, new algorithms and results continue to be presented in the literature. Systems of the form (1) that also satisfy (2) occur in a number of important practical applications. Such systems include the class of $\mathcal{F}$ matrices, where $C = 0$ and each column of B has at most two entries, and if there are two entries, they sum to zero. Such a B is sometimes called a gradient matrix.[2] Many of these matrices are related to topological network problems. For example, application of Kirchhoff's current law and Ohm's resistor law to a resistor network leads to a saddle-point matrix of the form (1) with B being a reduced node–arc incidence matrix with each column containing at most two nonzero entries $\{-1, 1\}$ (see Section 2.1 below). Another source is a water distribution pipe network analysis in which B is again a node–arc incidence matrix. Such network systems are nonlinear and are solved by using the Newton iteration method; see the work of Elhay et al.[3] These need a fast and robust linear solver because a saddle-point system has to be solved at each iteration. Other problems coming from practical applications that satisfy (2) are included in The University of Florida Sparse Matrix Collection[4]; we employ some of these in our numerical experiments (see Section 5).

The most common direct solution method for solving sparse symmetric indefinite linear systems involves factorizing K into the form

$$K = LDL^T,$$

where D is a diagonal matrix with 1×1 and 2×2 blocks, and L is a sparse unit lower triangular matrix. In practice, a more general factorization of the form

$$P^T SKSP = LDL^T$$

is computed, where S is a diagonal scaling matrix and P is a permutation matrix (or, more generally, a product of permutation matrices) that holds the pivot sequence (elimination order). It is the choice of S and P that determines the sparsity of L, as well as the accuracy and stability of the numerical factorization.

Before the factorization commences, P is normally computed using one of the many available fill-reducing ordering algorithms (a variant of nested dissection[5] or minimum degree[6-8] is most usually employed). These make the implicit assumption that the diagonal entries of K are present. An analyze phase uses the chosen pivot sequence to set up data structures for the subsequent factorization. A key difference between a sparse direct solver for SPD systems and one for symmetric indefinite systems is that the former can choose P on the basis of the sparsity pattern of K and then use it without change during the factorization, whereas the latter may need to modify it to incorporate pivoting to ensure that the factorization exists and to maintain numerical stability. In particular, for a saddle-point matrix with $C = 0$, following a fill-reducing ordering, variables corresponding to the $(2, 2)$ block may be chosen as pivot candidates before the diagonal entry has filled in. In this case, either the pivot must be delayed until later in the factorization or a suitable partner for use as a 2×2 pivot must be sought. A nonzero pivot candidate may also have to be delayed if it is small compared with the other entries in its column. Delaying a pivot leads to P being modified. Not only does modifying P contribute significantly to the complexity of the development of a sparse indefinite solver, it also adds overheads in terms of both time and memory requirements when the solver is run. These overheads occur in the search for a suitable pivot at each stage of the factorization and then in the handling of candidate pivots that are found to be unsuitable. Furthermore, pivoting means there is less scope for achieving parallelism and hence a reduction in performance when compared with the positive definite case (see the study of Hogg[9] for recent work on this for solving symmetric indefinite sparse linear systems on modern CPU/GPU architectures).

Our interest is in finding a permutation P such that $PSKSP^T$ can be factorized stably without the need for numerical pivoting and without modifying the entries of K while still limiting the number of entries in the factor L. This problem has been considered for $\mathcal{F}$ matrices by Tůma[10] and de Niet et al.[2] For general saddle-point problems, Bridson[11] split the nodes of the adjacency graph of K into two disjoint sets: A-nodes that correspond to the diagonal entries of K and C-nodes corresponding to the remaining diagonal entries. He then modified a sparsity-preserving ordering so that a C-node is ordered only after all its A-node neighbors have been ordered. Using this so-called constraint ordering, provided A is semidefinite and B is of full row rank, the LDL^T factorization can be shown to exist. Moreover, the pivots associated with the A-nodes are guaranteed to be positive, and those associated with C-nodes are guaranteed to be negative. By rescaling, $L \leftarrow L|D|^{1/2}$ and $D \leftarrow \text{sign}(D) = \text{diag}(\pm 1)$, the diagonal matrix is fully determined in advance by the structure of the problem, independent of the numerical values. This constrained ordering allows a Cholesky factorization code to be modified to perform the factorization of the indefinite K without numerical pivoting. Experiments reported by Hogg et al.[12] demonstrate that, compared with using a nested dissection ordering and modifying it during the factorization to maintain stability, the constrained ordering leads to a significantly denser factor L, and the flop counts to compute it are greater.

In the recent years, there has been considerable interest in using matching-based orderings (MBOs) to obtain orderings (and scalings) for sparse matrices. For unsymmetric matrices, maximum weighted matching algorithms are used to

move large entries onto the diagonal. The idea is that these will potentially provide stable candidate pivots, and the number of delayed pivots during the subsequent factorization will be reduced. In the symmetric case, symmetry needs to be preserved, but a symmetric permutation leaves the diagonal unchanged. Thus, the aim for a general symmetric matrix $K = \{k_{ij}\}$ is to permute a large off-diagonal entry k_{ij} close to the diagonal so that the 2×2 block $\begin{bmatrix} k_{ii} & k_{ij} \\ k_{ij} & k_{jj} \end{bmatrix}$ is potentially a good 2×2 candidate pivot. Duff et al.[13] noticed that the cycle structure of the permutation associated with the unsymmetric maximum weighted matching $\mathcal{M}$ can be exploited to obtain such a permutation P_s. This has been explored further by Duff et al.[14] and, among others, Hagemann et al.[15] and Schenk et al.[16,17]

A maximum weighted matching $\mathcal{M}$ is first computed. Any diagonal entries that are in the matching are immediately considered as potential 1×1 pivots and are held in a set $\mathcal{M}_1$. A set $\mathcal{M}_2$ of potential 2×2 pivots is then built by expressing the computed permutation in terms of its component cycles. A cycle of length 1 corresponds to an entry k_{ii} in the matching. A cycle of length 2 corresponds to two nodes i and j, where k_{ij} and k_{ji} are both in the matching. r potential 2×2 pivots can be extracted from even cycles of length $2r$ or from odd cycles of length $2k+1$. To combine the resulting permutation with a fill-reducing ordering, the adjacency graph of $P_s^T K P_s$ is compressed and an ordering is applied to the compressed graph. In the compression step, the union of the sparsity structure of the two rows and columns corresponding to a potential 2×2 pivot is built and used as the structure of a single row and column in the compressed matrix. A fill-reducing ordering is applied to the (weighted) compressed graph, and the resulting permutation is expanded to a permutation P_f for the original matrix. The final permutation is the product $P = P_f P_s$.

Hogg et al.[12] reported on the use of MBOs for solving tough general indefinite systems and extended their use to rank-deficient problems.[18] They found that while MBOs can substantially reduce the number of delayed pivots, it may still be necessary to perform some numerical pivoting. Moreover, computing an MBO can add a significant computational cost, and because the values of the entries of the matrix are used in its computation, if a sequence of problems with the same sparsity pattern needs to be factorized, the pivot order may have to be recomputed for each problem.

The main aim of this paper is to obtain a fill-reducing permutation P by exploiting the structure and properties of B so that $PSKSP^T$ can be factorized without modifications to the pivot sequence. To achieve this, B is permuted to the trapezoidal form (2). This form is used to obtain a block saddle-point matrix with m 2×2 blocks and $(n-m)1 \times 1$ blocks on the diagonal. In a previous work, Lungten et al[19,20] used this form and took it as their pivot sequence m 2×2 pivots followed by $(n-m)1 \times 1$ pivots. They proved that when $C = 0$ and subject to the matrix B_1 having large diagonal entries, the resulting factorization is stable. Furthermore, they showed that the work needed to compute L using this sequence can be limited by exploiting the fact that some of the blocks are unchanged during the factorization. However, a key disadvantage is that there can be substantially more fill-in in L compared with standard fill-reducing orderings that ignore the block structure, resulting in higher memory requirements and greater solve times (see the work of Lungten[21] and Section 5 below). In this paper, we propose a new approach that aims to improve the earlier work by treating each nonzero block of the block saddle-point matrix as a single entry to determine a compressed adjacency graph and then by applying a fill-reducing ordering to the compressed graph. The application of a fill-reducing ordering mixes up the order of the 1×1 and 2×2 pivots and leads to less fill. We show that Schur complement updates of this mixed pivot sequence exist provided that B_1 is nonsingular; this allows us to prove the existence of the factorization without modifications to the pivot sequence.

The outline of the rest of this paper is as follows. In Section 2, we consider permuting the matrix B to trapezoidal form, looking first at reduced node–arc incidence matrices and then more general matrices. Our new ordering algorithm, which we call Block Approximate Minimum Degree (BAMD), is presented in Section 3, and in Section 4, we prove that by using the BAMD pivot sequence, the LDL^T factorization exists. Numerical results are presented in Section 5. These include comparisons in terms of fill and the backward error with an MBO. Finally, in Section 6, some concluding remarks are given.

2 | PERMUTATION OF *B* TO TRAPEZOIDAL FORM

2.1 | Reduced node–arc incidence matrices

We first consider the saddle-point systems that arise in the network analysis of electronic circuits and water distribution pipe networks. In such systems, the matrix B is a reduced node–arc incidence matrix. Consider a connected directed graph (or network) $\mathcal{G}(V, E)$ with $m+1$ nodes $V = \{\eta_0, \eta_1, \ldots, \eta_m\}$ and n arcs (or edges) $E = \{\xi_1, \xi_2, \ldots, \xi_n\}$. The node–arc incidence matrix of $\mathcal{G}$ is an $(m+1) \times n$ matrix $\hat{B}$ with entries

$$\hat{b}_{ij} = \begin{cases} 1 & \text{if } \eta_i \text{ is the initial node of arc } \xi_j \\ -1 & \text{if } \eta_i \text{ is the terminal node of arc } \xi_j \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the rows of $\hat{B}$ correspond to nodes and the columns to arcs.

Saddle-point systems arising from network problems are made consistent by grounding a node, say, η_0, and by removing the corresponding row of the node–arc incidence matrix. The resulting $m \times n$ matrix B is a reduced node–arc incidence matrix. The columns that had entries in row η_0 have only one entry, whereas all other columns have exactly two entries (one of which is 1 and the other is -1). Starting from the ground node, a breadth-first search type algorithm to permute B to upper trapezoidal form is developed by Lungten et al.[22] This technique is based on connected star subgraphs and aims to obtain an upper triangular matrix B_1 such that B_1^{-1} is sparse. A star graph of order k is a tree with k nodes such that one node (referred to as the *central node*) is of degree $k-1$, and the remaining $k-1$ nodes are of degree 1; these $k-1$ nodes are neighbors of the central node. A modified version of algorithm 2 in the work of Lungten et al.[22] is presented in Algorithm 1 (see also algorithm 3.2.2 in the thesis of Lungten[21]). Here (and elsewhere), e_i is the ith unit vector, and $P_r(i)$ and $P_c(i)$ denote the ith columns of the permutation matrices P_r and P_c, respectively. The graph $\mathcal{G}(V,\ E)$ associated with an $m \times n$ reduced node–arc incidence matrix B can contain q $(1 \le q \le m-1)$ star subgraphs connected to each other. The first star subgraph of $\mathcal{G}(V,\ E)$ is the one with the central node $\eta_c = \eta_0$ that is grounded; k is initialized to 1. A neighbor η_{i_k} of η_c and the corresponding arc $\xi_{j_k} = (\eta_c,\ \eta_{i_k})$ are determined, and the corresponding columns i_k and j_k are permuted with column k of the permutation matrices P_r and P_c, respectively. η_{i_k} is then appended to the set W, and ξ_{j_k} is removed from E and k is incremented. The process is repeated for any remaining neighbors of η_c. One of the neighbors of the first central node is selected as the central node of the second star subgraph; q points to next central node, and the algorithm continues with this new central node (no search is needed to find the next central node).

Algorithm 1 Permutes a reduced node-arc incidence matrix B to trapezoidal form $[B_1\ B_2]$, where B_1 is upper triangular matrix and B_1^{-1} is sparse.

Input: An $m \times n$ $(m < n)$ reduced node-arc incidence matrix B of full row rank and its corresponding directed graph $\mathcal{G}(V,\ E)$ with $V = \{\eta_0, \eta_1, \dots, \eta_m\}$ and $E = \{\xi_1, \xi_2, \dots, \xi_n\}$, with ground node η_0.

Output: Permutation matrices P_r and P_c such that $P_r^T B P_c = [B_1\ \ B_2]$, where B_1 is an $m \times m$ upper triangular matrix and B_2 is an $m \times (n-m)$ matrix such that B_1^{-1} is sparse.

```
Set q = 1, k = 1, W = {η_0}, η_c = η_0. Set P_r = I_m and P_c = I_n.
while k ≤ m, do
    Find ξ_{j_k} = (η_c, η_{i_k}) such that η_{i_k} ∉ W
    if ξ_{j_k} ≠ φ, then
        Permute columns k and i_k of P_r
        Permute columns k and j_k of P_c
        W ← W ∪ {η_{i_k}}
        E ← E \ {ξ_{j_k}}
        k ← k + 1
    else
        q ← q + 1
        Select a new η_c ∈ W to be the central node of the qth star subgraph.
    end if
end while
```

It is of interest to note that in the work of Lungten et al.,[22] the permutation focuses only on obtaining an upper triangular B_1 (the remaining $n-m$ columns in B_2 are ordered randomly), Algorithm 1 additionally obtains B_2 with a banded structure. This is illustrated by the example in Figure 1.

2.2 | More general matrices

We now consider more general matrices B. We could employ a sparse QR algorithm to transform B to trapezoidal form as in the work of Schilder.[23] However, our interest is transforming B using permutations so that the number of entries in the matrix $\tilde{K}$ given by (3) is the same as in the original K (1) (using a QR factorization to transform K can lead to a dense $(1,1)$ and/or a dense $(2,2)$ block).

Given the $m \times n$ sparse matrix $B = [b_{ij}]$, we associate a bipartite graph $\mathcal{G}_B(V_r \cup V_c, E)$ in which the node sets $V_r = \{row_1, row_2, \dots, row_m\}$ and $V_c = \{col_1, col_2, \dots, col_n\}$ correspond to the rows and columns of B; there is a directed edge $\xi : row_i \to col_j$ of weight b_{ij} whenever $b_{ij} \neq 0$. An edge subset $\mathcal{M} \subseteq E$ is called a *matching* if no two edges in $\mathcal{M}$ are incident to the same node. In matrix terms, a matching corresponds to a set of nonzero entries with no two in the same row or column. A node is *matched* if there is an edge in the matching incident on the node. The *cardinality* of a matching is the number of edges in it. A *maximum matching* is a matching of maximum cardinality. If B is of full row rank and $m < n$, the maximum cardinality is m. Let V_{mr} and V_{mc} denote the row and column node sets corresponding to a maximum matching, respectively.

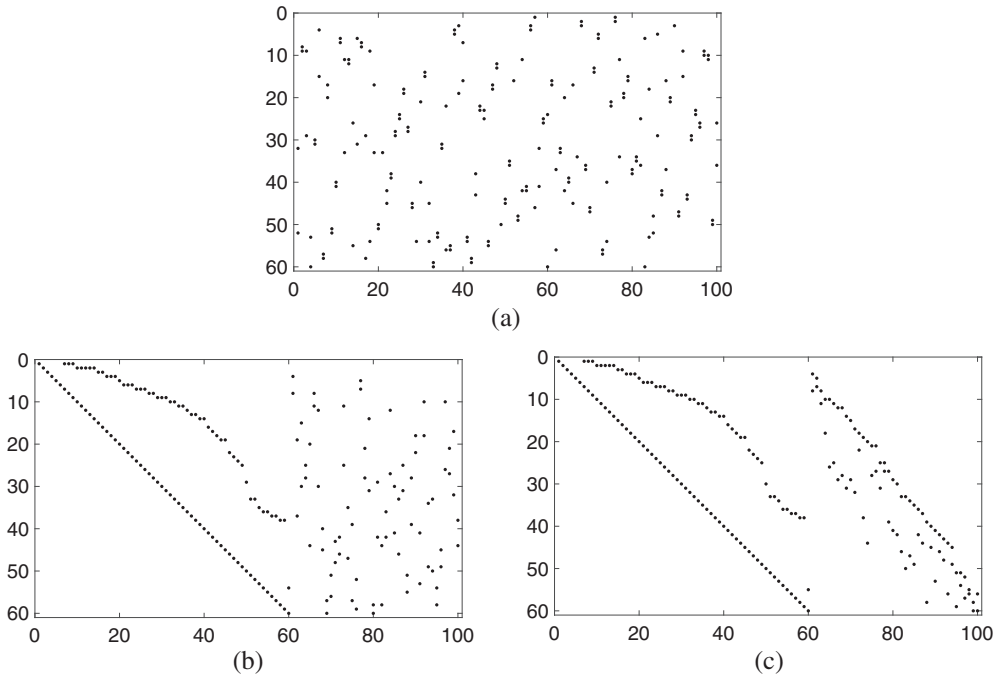

FIGURE 1 Example illustrating two different permutations of a reduced node–arc incidence matrix to upper trapezoidal form. (a) Original reduced node–arc incidence matrix B of order 60×100. (b) Trapezoidal form obtained using algorithm 2 in the work of Lungten et al.[22] (c) Trapezoidal form with banded B_2 obtained using Algorithm 1

To permute B, we use a simple minimum degree technique based on the following *degree-one principle*.

The degree-one principle

Let $\mathcal{G}_B(V_r \cup V_c, E)$ be the bipartite graph of an $m \times n$ $(m < n)$ sparse matrix B of full row rank. B can be permuted to trapezoidal form if, for $k = 1, \ldots, n-1$, the bipartite graph of $B^{(k)}$ has at least one node $j_k \in V_c$ of degree one, where $B^{(1)} = B$, and $B^{(k+1)}$ is the $(m-k) \times (n-k)$ matrix obtained by removing column j_k and the corresponding row from $B^{(k)}$.

Consider the 6×8 matrix B in Figure 2a and its associated bipartite graph $\mathcal{G}_B$ in Figure 2b. The first column node with degree one is $j_1 = 2$; it is matched with the row node $i_1 = 4$. Deleting j_1 and i_1 removes edges

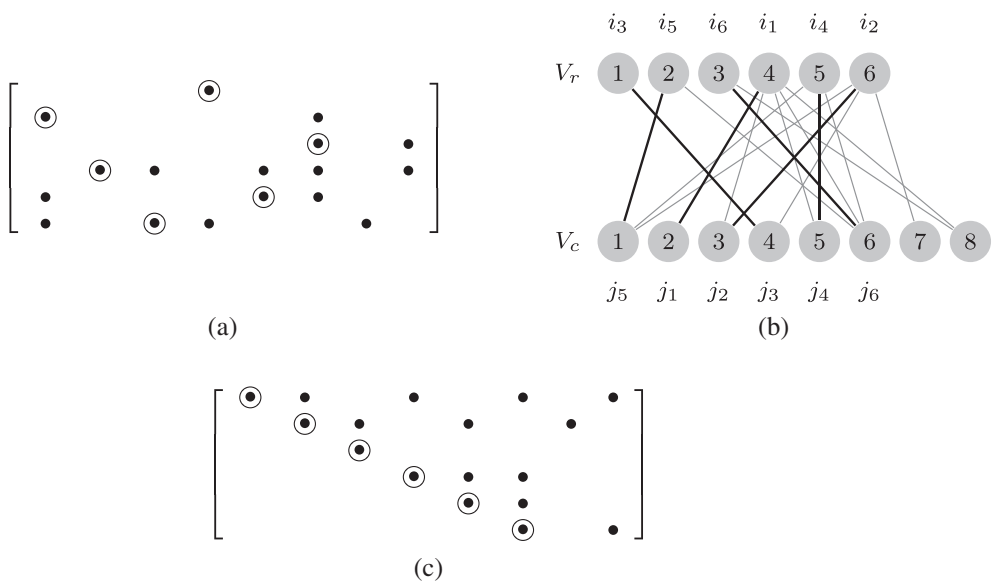

FIGURE 2 Permutation based on the degree-one principle. (a) B is matrix of dimensions 6×8 of full rank. (b) $\mathcal{G}_B$ is the bipartite graph. The thick edge lines connect the matched row and column nodes (corresponding to the circled nonzero entries in B). (c) The trapezoidal form with a 6×6 upper triangular matrix B_1 and a 6×2 rectangular matrix B_2, where $P_r = [e_4\, e_6\, e_1\, e_5\, e_2\, e_3]$ and $P_c = [e_2\, e_3\, e_4\, e_5\, e_1\, e_6\, e_7\, e_8]$ are the row and column permutation matrices, respectively

$\{(4,2),(4,3),(4,5),(4,6),(4,8)\}$. Column node $j_2 = 3$ now has degree one. It is matched with the row node $i_2 = 6$. Repeating the process gives a matching $\mathcal{M} = \{(4,1),(6,3),(1,4),(5,5),(2,1),(3,6)\}$ together with row and column matched node sets $V_{mr} = \{4,6,1,5,2,3\}$ and $V_{mc} = \{2,3,4,5,1,6\}$. Using the ordered sets V_{mr} and $V_{mc} \cup (V_c \setminus V_{mc})$, permutation matrices P_r and P_c of order m and n, respectively, can be defined to obtain the trapezoidal form in Figure 2c.

The steps are summarized in Algorithm 2. Here, for $row_i \in V_r$, $N(row_i)$ denotes the set of column nodes $col_k \in V_c$ that are neighbors of row_i (i.e., the edges $(row_i, col_k) \in E$), and for $col_j \in V_c$, $deg(col_j)$ is the number of row nodes $row_i \in V_r$ for which $(row_i, col_j) \in E$. The algorithm continues until either $E = \phi$ or all columns of the reduced matrix have degree greater than 1. If this happens after k steps, the permuted matrix is of the form

$$P_r^T B P_c = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}, \tag{4}$$

where B_{11} is $k \times k$ upper triangular, B_{12} is $k \times (n-k)$, and the $(m-k) \times (n-k)$ block B_{22} has columns of degree greater than one. A QR decomposition of B_{22} can be used to complete the transformation of B to trapezoidal form.

Algorithm 2 The degree-one principle matching algorithm for permuting an $m \times n$ $(m < n)$ sparse matrix B of full row rank with at least one column with a single entry to the form (4).

Require: B and its bipartite graph $G_B(V_r \cup V_c,\ E)$.

Ensure: Permutation matrices P_r and P_c such that $P_r^T B P_c$ is of the form (4), where B_{11} is an $k \times k$ upper triangular matrix for some k, $1 \le k \le m$.

```
Initialize V_mc = φ and V_mr = φ. Set k = 0. Set P_r = I_m and P_c = I_n.
For each col_j ∈ V_c, compute deg(col_j)
while there exists(row_i, col_j) ∈ E such that deg(col_j) = 1, do
    k ← k + 1
    V_mr ← V_mr ∪ {row_i}
    V_mc ← V_mc ∪ {col_j}
    Permute columns k and row_i of P_r
    Permute columns k and col_j of P_c
    E ← E \ ⋃ {(row_i, col_k) : col_k ∈ N(row_i)}
    For each col_j ∈ V_c \ V_mc, update deg(col_j)
end while
```

Remark 1. In practice, rather than finding a single column of degree 1 at each step, all the columns of degree 1 are found at once, and the updates to the degrees are then done after all such columns and their matched rows have been removed.

Remark 2. An advantage of Algorithm 1 compared with Algorithm 2 is that, for the former, B_1^{-1} is sparse. Note also that Algorithm 1 has to look for a ground node η_0 to use as the first central node only once; having found this node, the algorithm proceeds without having to search the remaining nodes looking for a new central node. The disadvantage of Algorithm 1 is that it is only applicable to reduced node–arc matrices.

3 | BAMD ORDERING

Having permuted B so that B_1 is upper triangular, the permuted saddle-point matrix can be written as

$$K = \begin{bmatrix} A_{11} & A_{12} & B_1^T \\ A_{21} & A_{22} & B_2^T \\ B_1 & B_2 & -C \end{bmatrix}, \tag{5}$$

where $A = [A_{11}\, A_{12};\ A_{21}\, A_{22}]$ with $A_{12} = A_{21}^T$, and $B = [B_1\, B_2]$. Let $A = [a_{ij}]$, $B = [b_{ij}]$, and $C = [c_{ij}]$. If we now define a permutation matrix P of order $n+m$ by

$$P = [e_1\ e_{n+1}\ e_2\ e_{n+2} \cdots e_m\ e_{n+m}\ e_{m+1} \cdots e_n],$$

where e_i is the ith unit vector of length $n+m$, then applying P to K, we obtain the block structure

$$P^T K P = [K_{ij}],$$

where K_{ij} is either a 2×2, 2×1, 1×2 or 1×1 block given by

$$K_{ij} = \begin{cases} \begin{bmatrix} a_{ii} & b_{ii} \\ b_{ii} & -c_{ii} \end{bmatrix}, 1 \le i = j \le m; & \begin{bmatrix} a_{ij} & b_{ji} \\ 0 & -c_{ij} \end{bmatrix}, 1 \le j < i \le m; \\ \begin{bmatrix} a_{ij} & 0 \\ b_{ij} & c_{ij} \end{bmatrix}, \ 1 \le i < j \le m; & \begin{bmatrix} a_{ij} \\ b_{ij} \end{bmatrix}, 1 \le i \le m < j \le n; \\ [a_{ij}\, b_{ji}], \quad 1 \le j \le m < i \le n; & [a_{ii}], \quad m < i, j \le n. \end{cases} \tag{6}$$

There are exactly m 2×2 and $(n-m)$ 1×1 diagonal blocks K_{ii} that form "a priori" pivots. Lungten et al.[19,20] showed that in the important and special case that B_1 is nonsingular and upper triangular and $C = 0$, using this pivot sequence results in a stable factorization provided that the entries of B_1 satisfy the following condition:

$$|b_{kk}| \ge \{|b_{kj}|, \ j = k+1, \ldots, n\}, \qquad \text{for } k = 1, \ldots, m. \tag{7}$$

If B is a node–arc incidence or gradient matrix, then (7) is clearly satisfied. Lungten et al. also showed that some of the blocks K_{ij} remain unchanged within the L factor, limiting the work needed to compute the factorization.

The key disadvantage of employing this pivot sequence in which all m 2×2 pivots precede the 1×1 pivots is that it can lead to significantly more fill-in in the factors than is necessary. To reduce the fill, we need to combine the preselection of pivot blocks with a fill-reducing ordering. We can do this as for the MBOs that we described in the Introduction. That is, we compress the adjacency graph of PKP^T by considering each block as a single entity and merging the sparsity patterns of the rows and columns belonging to a 2×2 diagonal block; a fill-reducing ordering is then applied to the compressed graph. When an approximate minimum degree (AMD) ordering[6] is used on the compressed graph, we refer to this as BAMD ordering.

We observe that when ordering the compressed graph, we do not employ a weighting when a row of the compressed graph corresponds to two rows of the original matrix. In their work on MBOs, Hogg et al.[12] found that this offered little advantage.

4 | EXISTENCE OF THE FACTORIZATION

To prove the existence of the factorization, we use the following well-known result.

Lemma 1. *Partition an $n \times n$ SPD matrix A into the block form*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where A_{11} is $m \times m$ $(1 \le m < n)$, $A_{21} = A_{12}^T$ is $n-m \times m$, and A_{22} is $n-m \times n-m$. Then, the Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is SPD.

Recall that the a priori pivot sequence comprises 2×2 pivots formed by taking rows and columns of A_{11}, B_1, and C, and 1×1 pivots that are the elements of A_{22}; the fill-reducing ordering of the compressed graph permutes the order of these pivots. At each stage of the factorization, all the remaining pivots must be updated. Each updated 2×2 pivot has one of the following forms:

$$\begin{bmatrix} \alpha & \beta \\ \beta & -\gamma \end{bmatrix}, \quad \begin{bmatrix} \alpha & \beta \\ \beta & 0 \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} \alpha & 0 \\ 0 & -\gamma \end{bmatrix}, \tag{8}$$

where, if B_1 is nonsingular and triangular, α, β, and γ, which are from the Schur complement updates of A, B, and C, respectively, are nonzero. This is a result of the following theorem, which shows that a 2×2 pivot updated by the Schur complement of a 1×1 pivot is nonsingular and vice versa.

Theorem 1. *Let K be the $(n+m) \times (n+m)$ saddle-point matrix (5), where the $m \times n$ matrix $B = [B_1 B_2]$ is of full rank with B_1 being an $m \times m$ nonsingular matrix; the $n \times n$ SPD matrix A is partitioned conformally and C is an $m \times m$ SPSD matrix* (including $C = 0$). Let K be permuted as follows:

$$\left[\begin{array}{cc|c} A_{11} & B_1^T & A_{12} \\ B_1 & -C & B_2 \\ \hline A_{21} & B_2^T & A_{22} \end{array}\right].$$

Then, the Schur complement S_{indef} of the symmetric indefinite matrix $\begin{bmatrix} A_{11} & B_1^T \\ B_1 & -C \end{bmatrix}$ and the Schur complement S_{SPD} of the SPD matrix A_{22} are nonsingular.

Proof. From Lemma 1, the Schur complements

$$S_1 = A_{22} - A_{21}A_{11}^{-1}A_{12} \text{ and } S_2 = A_{11} - A_{12}A_{22}^{-1}A_{21}$$

are SPD. Now, let

$$S_{A_{11}} = C + B_1 A_{11}^{-1} B_1^T. \tag{9}$$

Because C is SPSD, A is SPD, and B_1 is nonsingular, $S_{A_{11}}$ is SPD and thus nonsingular. Therefore,

$$\begin{aligned} S_{\text{indef}} &= A_{22} - \begin{bmatrix} A_{21} & B_2^T \end{bmatrix} \begin{bmatrix} A_{11} & B_1^T \\ B_1 & -C \end{bmatrix}^{-1} \begin{bmatrix} A_{12} \\ B_2 \end{bmatrix} \\ &= A_{22} - \begin{bmatrix} A_{21} & B_2^T \end{bmatrix} \begin{bmatrix} A_{11}^{-1} - A_{11}^{-1}B_1^T S_{A_{11}}^{-1} B_1 A_{11}^{-1} & A_{11}^{-1}B_1^T S_{A_{11}}^{-1} \\ S_{A_{11}}^{-1} B_1 A_{11}^{-1} & -S_{A_{11}}^{-1} \end{bmatrix} \begin{bmatrix} A_{12} \\ B_2 \end{bmatrix} \\ &= A_{22} - A_{21}A_{11}^{-1}A_{12} + A_{21}A_{11}^{-1}B_1^T S_{A_{11}}^{-1} B_1 A_{11}^{-1} A_{12} - A_{21}A_{11}^{-1}B_1^T S_{A_{11}}^{-1} B_2 \\ &\quad - B_2^T S_{A_{11}}^{-1} B_1 A_{11}^{-1} A_{12} + B_2^T S_{A_{11}}^{-1} B_2. \end{aligned}$$

i.e.,

$$S_{\text{indef}} = S_1 + \left(A_{21}A_{11}^{-1}B_1^T - B_2^T\right) S_{A_{11}}^{-1} \left(B_1 A_{11}^{-1} A_{12} - B_2\right). \tag{10}$$

The right-hand side of (10) is the sum of SPD and SPSD matrices, and hence, S_{indef} is SPD and nonsingular.

The Schur complement of A_{22} is

$$S_{\text{SPD}} = \begin{bmatrix} A_{11} & B_1^T \\ B_1 & -C \end{bmatrix} - \begin{bmatrix} A_{12} \\ B_2 \end{bmatrix} A_{22}^{-1} \begin{bmatrix} A_{21} & B_2^T \end{bmatrix} = \begin{bmatrix} S_2 & \hat{B}^T \\ \hat{B} & -\hat{C} \end{bmatrix},$$

where $\hat{B} = B_1 - B_2 A_{22}^{-1} A_{21}$ and $\hat{C} = C + B_2 A_{22}^{-1} B_2^T$. The Schur complement of S_2 is

$$S_{\text{C}} = -\hat{C} - \hat{B} S_2^{-1} \hat{B}^T = -(C + G),$$

where $G = B_2 A_{22}^{-1} B_2^T + \hat{B} S_2^{-1} \hat{B}^T$. We need to show that S_{C} is nonsingular. It suffices to show that S_{C} is negative definite by showing that G is SPD. Define a block permutation matrix P of order n by

$$P = \begin{bmatrix} 0 & I_m \\ I_{n-m} & 0 \end{bmatrix}.$$

It is easy to see that $G = BP(P^T A P)^{-1} P^T B^T$, which is SPD. □

Remark 3. Theorem 1 proves that if B_1 is nonsingular, then the factorization of the permuted saddle-point system using the BAMD pivot sequence exists. If an ordering such as the matching-based `MC64` ordering[24] is applied to B alone, it is possible to obtain a B_1 with large entries on the diagonal. However, such orderings do not guarantee that B_1 is nonsingular, and consequently, the LDL^T factorization may not exist without modifications to the pivot sequence. We have performed numerical experimentation that confirms this.

Remark 4. If C is SPD, the requirement that B_1 is nonsingular is not needed to prove the existence of the Schur complements S_{indef} and S_{SPD}.

TABLE 1 Test problems

Identifier	n	m	$nz(K)$	C
c-55	19,121	13,659	403,450	$-\delta I_m$
c-58	22,461	15,134	552,551	$-\delta I_m$
c-62	25,158	16,573	559,341	$-\delta I_m$
c-70	39,302	29,622	658,986	$-\delta I_m$
c-72	47,950	36,114	707,546	$-\delta I_m$
c-73	86,417	83,005	1,279,274	$-\delta I_m$
c-big	201,877	143,364	2,340,859	$-\delta I_m$
tuma1	13,360	9,607	87,760	0
tuma2	7,514	5,477	49,365	0
mario001	23,130	15,304	204,912	0
mario002	234,128	155,746	2,097,566	0
helm3d01	30,060	2,166	428,444	0
k1_san	46,954	20,804	559,774	0
d_pretok	129,160	53,569	1,641,666	0
S3D-15	11,520	4,095	122,298	0
S3D-18	19,494	6,858	208,158	0
S3D-24	45,000	15,624	484,044	0
S3D-32	104,544	35,936	1,130,772	0
RNB6	21,208	13,167	106,034	0
RNC1	58,054	36,392	290,264	0
RNC3	12,222	7,631	61,104	0
RNC4	7,459	4,656	37,289	0
RNC6	16,551	19,775	82,749	0
WN6	8,584	8,392	42,916	0
WN7	14,830	12,523	74,130	0
WN8	19,647	17,971	98,205	0

Note. n and m denote the order of A and C (see (1)), and $nz(K)$ is the number of entries in K.

5 | NUMERICAL EXPERIMENTS

In this section, we present numerical results to illustrate the effectiveness of our proposed ordering algorithm BAMD for solving sparse saddle-point systems (1). Our test matrices are listed in Table 1. They all satisfy $n + m > 12,000$, and in each case, the $(2, 1)$ block B can be permuted to the trapezoidal form (2). The problems come from a range of application areas. The c-xx problems are interior-point optimization matrices and are taken from the University of Florida Sparse Matrix Collection.[4] For these examples, $C = -\delta I_m$, where $\delta = 10^{-8}$; for all other examples, $C = 0$. The problems tuma1 to d_pretok are finite element models and are also from the University of Florida Sparse Matrix Collection.[4] The S3D-xx examples are generated using finite difference methods for Stokes equation in three dimensions[2]; the RNxx examples are from industrial resistor network analysis[25]; and the WNx examples are water distribution pipe networks.[3]

The numerical experiments are performed on a MacBook Pro Retina, 64-bit OS X EI Capitan with a 2.6 GHz Intel Core i5 using MATLAB R2016a (9.0.0.341360). The right-hand-side vector b is computed so that the exact solution is $z = [1, \ldots, 1]^T$. In the following, the scaled residual ϵ_{rb} is given by

$$\epsilon_{\text{rb}} = \frac{\|Kz - b\|}{\|K\|\|z\| + \|b\|},$$

with the infinity norm. The computed solution is only accepted if ϵ_{rb} is less than $tol = 10^{-13}$; where necessary, up to 20 steps of iterative refinement are performed. If ϵ_{rb} remains greater than tol after iterative refinement, then we record a failure. We define the fill ratio to be

$$fill(L) = nz(L)/nz(K_L),$$

where $nz(K_L)$ and $nz(L)$ denote the number of entries in the lower triangular part of K and in L, respectively. Although not explicitly reported here, we also always check the forward errors of our computed solutions.

We use the MATLAB interface to the state-of-the-art sparse direct solver `HSL_MA97`.[26,27] `HSL_MA97` implements a multifrontal algorithm and, for indefinite systems, employs threshold partial pivoting to ensure that all entries of L satisfy $l_{ij} < u^{-1}$, where the threshold parameter $u \in [0, 0.5]$ is under the control of the user (the default setting is 0.01). If u is

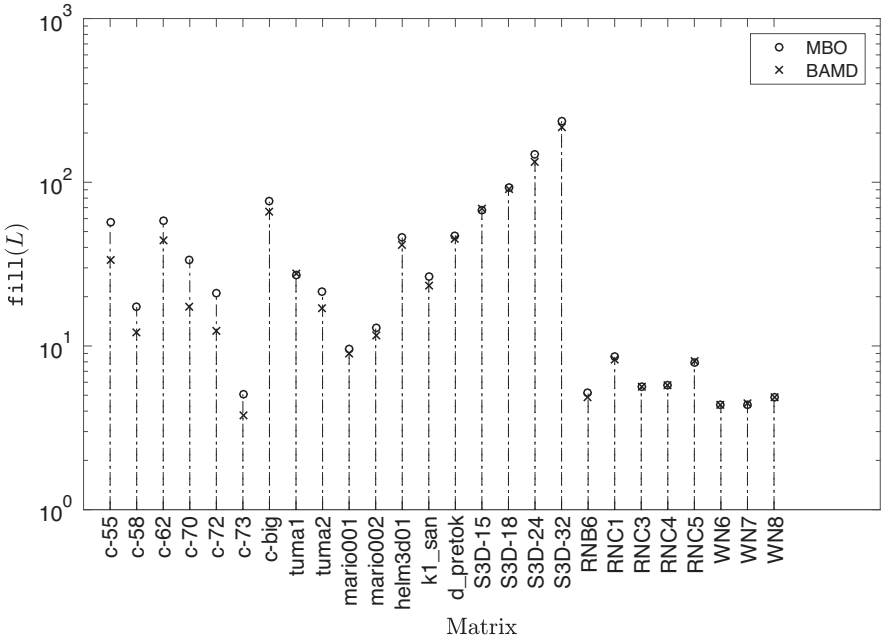

FIGURE 3 Sparsity of the factor for the matching-based ordering (MBO) and BAMD orderings for case (i) (default u)

chosen to be small, then the number of pivots that are delayed during the factorization will generally be small, minimizing the fill-in in L (i.e., the chosen pivot sequence is used with little or no modification), but for a general fill-reducing pivot sequence, the factorization is potentially unstable. Increasing u gives a greater guarantee of stability but at the possible cost of increased fill-in in L. Note that it is important that the entries of K are well scaled before the factorization commences. `HSL_MA97` offers a number of scaling options; in our experiments, we use the `MC64` scaling.

`HSL_MA97` includes a number of ordering options. Our interest is comparing our proposed BAMD ordering algorithm with the MBO offered by `HSL_MA97`. In particular, we employ the MBO with AMD on the compressed graph (the MATLAB interface setting is `control.ordering = 7`). We report on two cases: (i) default settings ($u = 0.01$) and (ii) threshold $u = 0.0$. With the latter setting, numerical pivoting is "switched off", and pivots are only delayed if they are (approximately) zero.

Sparsity results and the scaled residuals for the MBO and BAMD orderings for case (i) are reported in Figures 3 and 4.

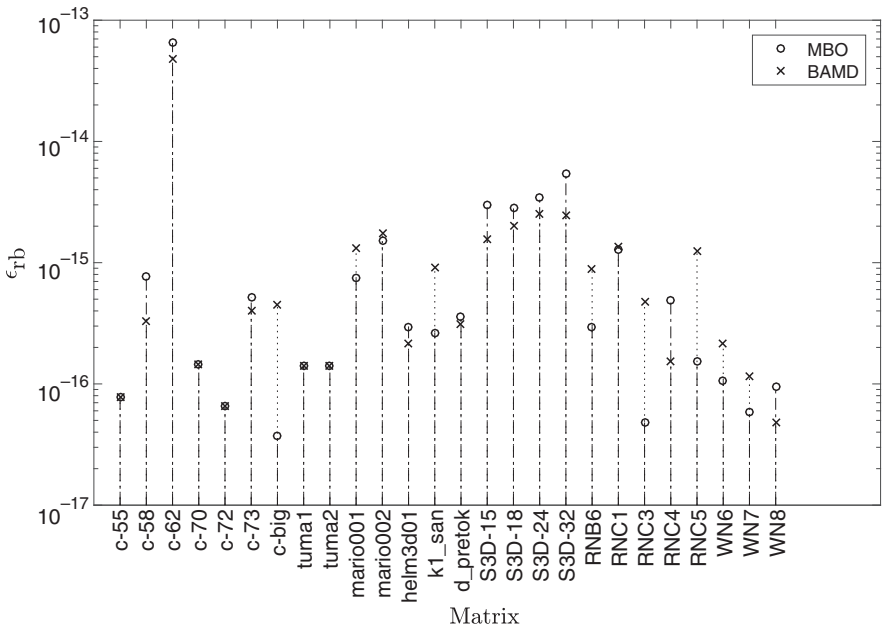

FIGURE 4 Relative backward errors ϵ_{rb} for the matching-based ordering (MBO) and BAMD orderings for case (i) (default u)

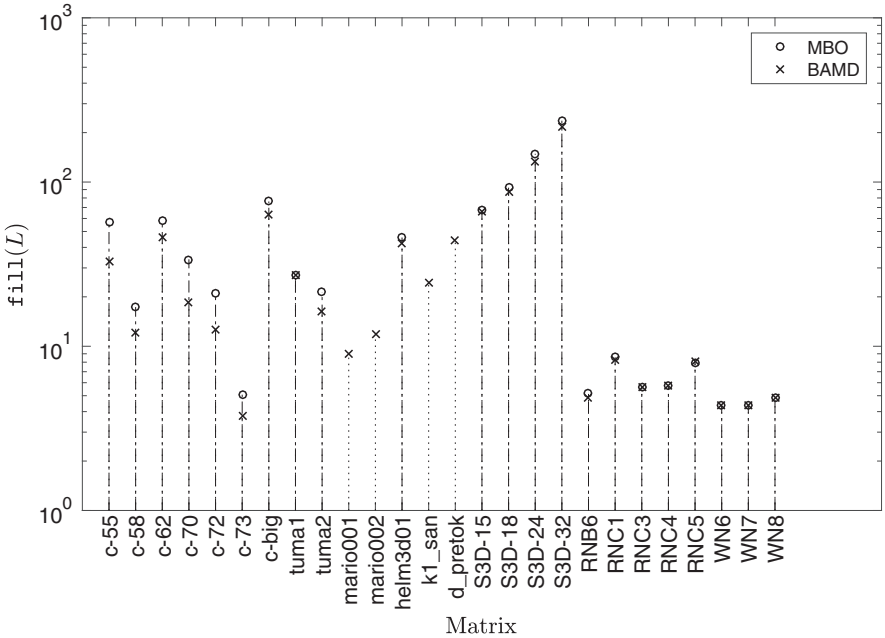

FIGURE 5 Sparsity of the factor for the matching-based ordering (MBO) and BAMD orderings for case (ii) ($u = 0.0$)

It can be observed that the fill ratio for the two orderings is similar, with the BAMD resulting in less fill for the c-xx examples. In each instance, a single step of iterative refinement is sufficient to reduce the scaled residual to be less than *tol*, confirming that the default setting for the threshold pivoting parameter u leads to stable factorizations for our test examples.

Analogous results for case (ii) are shown in Figures 5 and 6, where an ordering leads to a failure, and no result is plotted. Again, for all the problems, BAMD requires at most one step of iterative refinement to achieve the requested accuracy. However, for MBO, there are four failures (problems mario001, mario002, k1_san, and d_pretok). These results confirm that while using an MBO limits the need for numerical pivoting, pivoting is still needed for some "tough" practical cases. However, using a BAMD guarantees the existence of the LDL^T factorization without pivoting. Furthermore, the level of fill it produces is comparable (or less) with that for MBO. We conclude that BAMD can offer an attractive ordering for saddle-point systems for which B is permuted to trapezoidal form.

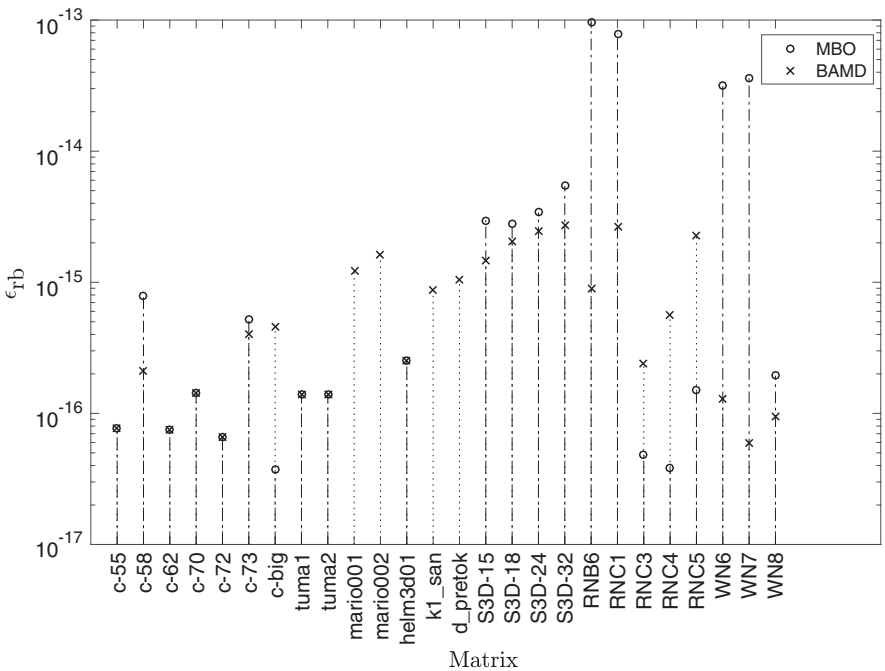

FIGURE 6 Relative backward errors ϵ_{rb} for the matching-based ordering (MBO) and BAMD orderings for case (ii) ($u = 0.0$)

TABLE 2 Sparsity of the factor for the 2F1 and BAMD orderings

Identifier				2F1		BAMD	
	n	m	$nz(K)$	$nz(L)$	$fill(L)$	$nz(L)$	$fill(L)$
c-55	19,121	13,659	403,450	71,117,712	326	7,177,301	33
c-58	22,461	15,134	552,551	112,668,327	382	3,564,490	12
tuma1	13,360	9,607	87,760	6,684,202	132	1,368,774	27
mario002	234,128	155,746	2,097,566	675,010,515	579	13,788,854	12
S3D-15	11,520	4,095	122,298	59,741,873	893	4,426,057	66
RNB6	21,208	13,167	106,034	174,348,828	2,740	310,044	4.9
RNC4	7,459	4,656	37,289	16,911,441	756	127,938	5.7
WN6	8,584	8,392	42,916	1,627,607	63	112,118	4.4

Note. $nz()$ denotes the number of entries in the matrix, and $fill(L)$ is the fill ratio.

We remark that we have performed tests on problems of the form (1) from the University of Florida Sparse Matrix Collection for which B cannot be permuted to trapezoidal form. In this case, we used the sparse QR algorithm and then ran the BAMD ordering. We compared our results with employing the MBO ordering, using default u and $u = 0.0$. We found similar levels of fill in the factors for both orderings, and there were no failures. We conclude that the sparse QR algorithm can be used to extend the applicability of the BAMD ordering (but it adds to the total computational cost).

Finally, we compare the BAMD ordering with that of Lungten et al[19,20] in which all m 2×2 pivots precede the 1×1 pivots; we refer to this as the 2F1 ordering. Results are presented in Table 2 for a subset of our test problems. We are unable to run the 2F1 ordering on some of the larger examples because of insufficient memory. The results clearly demonstrate that requiring that all the 2×2 pivots are used first is too restrictive as it leads to unacceptable fill-in in the factors.

6 | CONCLUDING REMARKS

In the recent years, driven by the need to develop direct solvers for efficiently solving sparse indefinite symmetric linear systems on modern parallel computing platforms, there has been an interest in the development of new ordering strategies that can choose a pivot sequence before the factorization commences and that can then be used with minimal (or, ideally, without) modification during the factorization (see, e.g., other works[9,15,28, 29]). Because data movement can be more expensive than numerical operations, it can be advantageous to perform more operations (and possibly allow more fill-in in the factor) than are performed by a traditional serial code. MBOs have been developed that limit the changes needed to the pivot sequence while leading to more fill. However, as our results confirm, such orderings do not remove the need for pivoting. Furthermore, they are computed using the numerical values of the matrix entries, and so, if more than one matrix with the same (or almost the same) sparsity pattern is to be factorized, MBOs have the disadvantage of potentially needing to be recomputed for each example.

In this paper, we have focused on a new fill-reducing ordering algorithm that can be used to solve symmetric indefinite saddle-point systems without the need for pivoting during the numerical factorization. The ordering is computed using only the sparsity structure. We have considered a class of saddle-point matrices in which the $(2, 1)$ block B can be permuted to trapezoidal form $B = [B_1, B_2]$, where B_1 is a nonsingular triangular matrix. We have discussed permuting B so that B_1 is upper triangular, but our proposed ordering is also applicable if B_1 is lower triangular. Using the diagonal entries of B_1, the rows and columns of the saddle-point matrix are partitioned into a block structure constituting a priori pivots of order 1 and 2. The partitioned matrix is compressed, and a fill-reducing ordering is applied to the resulting graph. Based on this strategy, we have shown that a block LDL^T factorization can be computed without having to modify the preselected pivot sequence. In our experiments, we reported on using AMD applied to the compressed graph; in some cases, the fill-in in L may be reduced by employing other orderings (in particular, a nested dissection ordering could be used).

Finally, we remark that Scott et al.[30] recently found that, for symmetric indefinite saddle-point systems, preordering the matrix using an MBO and then computing its incomplete factorization resulted in a higher quality preconditioner than preordering with a minimum degree or nested dissection ordering. A possible future investigation is to look at whether the preconditioner quality can be further improved using our proposed new ordering strategy. Another future direction is to look at other ways of preordering B to try and extend the applicability of our approach to more general saddle-point systems.

ORCID

Sangye Lungten http://orcid.org/0000-0002-7262-5486
Jennifer A. Scott http://orcid.org/0000-0003-2130-1091

REFERENCES

1. Benzi M, Golub GH, Liesen J. Numerical solution of saddle point problems. Acta Numer. 2005;14:1–137.
2. de Niet AC, Wubs FW. Numerically stable LDL^T -factorization of F-type saddle point matrices. IMA J Numer Anal. 2009;29(1):208–234.
3. Elhay S, Simpson AR, Deuerlein J, Alexander B, Schilders WHA. Reformulate co-tree flows method competitive with the global gradient algorithm for solving water distribution system equations. J Water Resour Plan Manag. 2014;140(12):04014040–1–04014040–10.
4. Davis TA, Hu Y. The University of Florida Sparse Matrix Collection. ACM Trans Math Softw. 2011;38(1):1–25.
5. George A. Nested dissection of a regular finite-element mesh. SIAM J Numer Anal. 1973;10(2):345–363.
6. Amestoy PR, Davis TA, Duff IS. An approximate minimum degree ordering algorithm. SIAM J Matrix Anal Appl. 1996;17(4):886–905.
7. Liu JWH. Modification of the minimum degree algorithm by multiple elimination. ACM Trans Math Softw. 1985;11(2):141–153.
8. Tinney WF, Walker JW. Direct solutions of sparse network equations by optimally ordered triangular factorization. Proc IEEE. 1967;55(11):1801–1809.
9. Hogg JD. A new sparse LDL^T solver using a posteriori threshold pivoting. Didcot, UK: Rutherford Appleton Laboratory; 2016. RAL-TR-2016-017.
10. Tůma M. A note on the LDL^T decomposition of matrices from saddle-point problems. SIAM J Matrix Anal Appl. 2002;23(4):903–925.
11. Bridson R. An ordering method for the direct solution of saddle-point matrices. Unpublished preprint. 2007. Available from: http://www.cs.ubc.ca/ ˜rbridson/kktdirect/
12. Hogg JD, Scott JA. Pivoting strategies for tough sparse indefinite systems. ACM Trans Math Softw. 2013;40(1):1–19.
13. Duff IS, Gilbert JR. Maximum-weighted matching and block pivoting for symmetric indefinite systems. Paper presented at: Householder Symposium on Numerical Linear Algebra; 2002 Jun 17–21; Peebles, Scotland. 2002. http://personal.strath.ac.uk/a.ramage/householder/
14. Duff IS, Pralet S. Strategies for scaling and pivoting for sparse symmetric indefinite problems. SIAM J Matrix Anal Appl. 2005;27(2):313–340.
15. Hagemann M, Schenk O. Weighted matchings for preconditioning symmetric indefinite linear systems. SIAM J Sci Comput. 2006;28(2):403–420.
16. Schenk O, Gärtner K. On fast factorization pivoting methods for symmetric indefinite systems. Electron Trans Numer Anal. 2006;23:158–179.
17. Schenk O, Wächter A, Hagemann M. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale non-convex interior-point optimization. Comput Optim Appl. 2007;36(2–3):321–341.
18. Hogg JD, Scott JA. Optimal weighted matchings for rank-deficient sparse matrices. SIAM J Matrix Anal Appl. 2013;34(4):1431–1447.
19. Lungten S, Schilders WHA, Maubach JML. Threshold incomplete factorization constraint preconditioners fpr saddle-point matrices. Linear Algebra Appl. 2018;545:76–107.
20. Lungten S, Schilders WHA, Maubach JML. Sparse block factorization of saddle point matrices. Linear Algebra Appl. 2016;502:214–242.
21. Lungten S. Solution methods for indefinite systems of linear equations in saddle-point form [PhD thesis]. Eindhoven, The Netherlands: Technische Universiteit Eindhoven; 2016.
22. Lungten S, Schilders WHA, Maubach JML. Sparse inverse incidence matrices for Schilders' factorization applied to resistor network modeling. Numer Algebra Control Optim. 2014;4(3):227–239.
23. Schilders WHA. Solution of indefinite linear systems using an LQ decomposition for the linear constraints. Linear Algebra Appl. 2009;431(3–4):381–395.
24. Duff IS, Koster J. On algorithms for permuting large entries to the diagonal of a sparse matrix. SIAM J Matrix Anal Appl. 2001;22(4):973–996.
25. Rommes J, Schilders WHA. Efficient methods for large resistor networks. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2010;29(1):28–39.
26. Hogg JD, Scott JA. HSL_MA 97: A bit-compatible multifrontal code for sparse symmetric systems. Didcot, UK: Rutherford Appleton Laboratory; 2011. Technical report RAL-TR-2011-024.
27. HSL. A collection of Fortran codes for large-scale scientific computation; 2016. Available from: http://www.hsl.rl.ac.uk/
28. Duff IS, Pralet S. Towards a stable mixed pivoting strategy for the sequential and parallel solution of sparse symmetric indefinite systems. SIAM J Matrix Anal Appl. 2007;29(3):1007–1024.
29. Hogg JD, Scott JA. Compressed threshold pivoting strategies for tough sparse indefinite systems. SIAM J Matrix Anal Appl. 2014;35(2):783–817.
30. Scott JA, Tůma M. Improving the stability and robustness of incomplete symmetric indefinite factorization preconditioners. Numer Linear Algebra Appl. 2017;24(5).

How to cite this article: Lungten S, Schilders WHA, Scott JA. Preordering saddle-point systems for sparse LDL^T factorization without pivoting. *Numer Linear Algebra Appl*. 2018;e2173. https://doi.org/10.1002/nla.2173