Proceedings of the Project Review, Geo-Mathematical Imaging Group (Purdue University, West Lafayette IN), Vol. 1 (2013) pp. 131-138.

NONSYMMETRIC STRUCTURED MULTIFRONTAL METHODS FOR SPARSE MATRICES WITH APPLICATION TO SOLVING THE HELMHOLTZ EQUATION

ZIXING XIN*, JIANLIN XIA†, MAARTEN V. DE HOOP‡, STEPHEN CAULEY§, AND VENKATARAMANAN BALAKRISHNAN¶

Abstract. Some structured multifrontal methods for nonsymmetric sparse matrices are developed, which are applicable to discretized PDEs such as Helmholtz equations on finite difference or irregular meshes. Unlike various existing structured direct solvers, which often focus on symmetric positive definite or symmetric matrices, our methods consider the issues for nonsymmetric cases, which often arise due to the nature of the problem or the boundary conditions. We accommodate the nonsymmetry in both the ordering and the factorization. Multiple variations are giving, including a randomized nonsymmetric multifrontal method. Both static and local pivoting strategies for the structured factorizations are discussed for the stability purpose.

1. Introduction. Solving large sparse linear systems of equations is an important issue in many engineering and scientific problems. Here, we consider the following equation:

$$Ax = b$$

where A is an $n \times n$ nonsymmetric (or non-Hermitian) matrix arising from the discretization of PDEs on regular or irregular meshes.

Among various direct methods, the multifrontal method [16] has good data locality and takes advantage of dense matrix kernels. Significant progress has been made since its initial development in [7]. This method and its variations have been implemented in many high performance computing packages.

Generally, a major hurdle in sparse direct solutions is fill-in. Thus, A is usually first reordered with methods such as nested dissection (ND) [9] or minimal degree (MD) [17] methods. It has been shown that ND generally has better performance than MD for large problems. For symmetric matrices, the adjacency graph of A preserves the matrix structure and can be utilized to obtain the ordering of original matrix. Each separator on the graph corresponds to a local reordering of the original matrix. Finally, all the separators form a tree which serves as the assembly tree [7] in multifrontal method. For nonsymmetric matrices, the adjacency graph of AA^T or $A + A^T$ is often considered. Explicitly forming the structure of AA^T can be very time consuming when n is large. Thus, we choose to reorder A based on the non-zero pattern of $A + A^T$, as often done [6].

In recent years, structured multifrontal methods [20, 22] have been developed, and combine the multifrontal method with rank structures such as hierarchically semiseparable (HSS) forms [3, 23]. These methods produce approximate factorizations and can break the classical factorization complexity bounds. But the methods are mostly tested on symmetric positive definite, symmetric, or pattern symmetric matrices.

The main contribution of this paper is to extend the approaches in [20, 22] to nonsymmetric matrices. Since the major building pieces are mostly available, we try to put them together and put forward a practical nonsymmetric structured multifrontal code. For various problems, especially those where the nonsymmetry is due to the boundary conditions, the inherent rank properties are similar to the symmetric cases. Nonsymmetric extensions of the methods in [20, 22] are shown. We also give the major steps for generalizing a latest randomized multifrontal method in [21] to the nonsymmetric case.

*Department of Mathematics, Purdue University, West Lafayette, IN 47907 (zxin@purdue.edu)
†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (xiaj@math.purdue.edu)
‡Department of Mathematics, Purdue University, West Lafayette, IN 47907 (mdehoop@math.purdue.edu)
§Athinoula A. Martinos Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital, Harvard University, Charlestown, MA 02129 ((stcauley@nmr.mgh.harvard.edu).
¶School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 ((ragu@ecn.purdue.edu)

We also consider various pivoting issues for structured factorizations. For example, we can use static pivoting [14] to permute large entries on the diagonal so as to improve the numerical stability of the factorization. We also discuss the potential of improving the stability of HSS factorizations.

Our solver is applied to Helmholtz equations discretized on both finite difference and finite element meshes. We show the efficiency advantage over the standard multifrontal method.

The remaining sections are organized as follows. We briefly review HSS structures and ULV factorizations in Section 2. We show the nonsymmetric structured multifrontal methods in Section 3, after a review of the basic idea of existing structured multifrontal methods. In Section 4, we discuss the pivoting and stability issues. Section 5 includes our numerical experiments on Helmholtz problems on both regular and irregular meshes. Finally, we draw our conclusions in Section 6.

2. Hierarchically semiseparable structure and ULV factorization. In our structured multifrontal method for a nonsymmetric sparse matrix A, the intermediate dense matrices are approximated by hierarchically semiseparable (HSS) forms. In this section, we give a brief review of HSS representations and HSS ULV factorizations. Refer to [3, 23] for more details.

For a matrix F, we partition it into 2^l block rows and 2^l block columns corresponding to a certain level l of a full binary tree T with k levels. The block rows and columns without the diagonal blocks are called HSS blocks. Each HSS block is associated with a node in the binary tree. Suppose the nodes of T are in their postordering and j is a non-leaf node with children c_1 and c_2 in T. If there exist matrices $D_j, U_j, V_j, R_j, W_j, B_j$ satisfying the following recursions:

$$D_j = \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1} V_{c_2}^T \\ U_{c_2} B_{c_2} V_{c_1}^T & D_{c_2} \end{pmatrix}, \quad U_j = \begin{pmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{pmatrix}, \quad V_j = \begin{pmatrix} V_{c_1} W_{c_1} \\ V_{c_2} W_{c_2} \end{pmatrix},$$

then we say that F is in an HSS form. The matrices $D_j, U_j, V_j, R_j, W_j, B_j$ are called the HSS generators. T is called an HSS tree.

We say F has a low-rank property if its HSS blocks have small ranks or numerical ranks. To take advantage of this property, we use ULV factorizations as discussed in [3, 23]. First, we sparsify the HSS blocks by applying QL factorizations to U_j. After the diagonal generators D_j are then updated and partially factorized conformably. Some leading pieces in the diagonal blocks can then be eliminated, and the remaining pieces are gathered to form a reduced HSS form [20]. The process is repeated until the root of the HSS tree is reached. The ULV factorization costs $O(r^2 N)$ flops, where r is the maximum rank of all the HSS blocks (called HSS rank), and N is the matrix size.

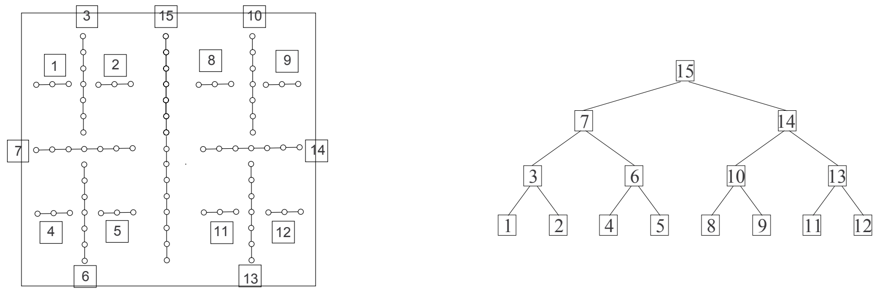
The ULV HSS solution [20] is performed with the sequence of factors from the factorization stage. This includes a forward substitution stage and a backward substitution stage, which propagates information bottom-up and then top-down along T. The solution cost is $O(rN)$.

3. Nonsymmetric structured multifrontal methods.

3.1. Nested dissection ordering for directed graphs. In sparse direct solutions, an adjacency graph G for a (nonsymmetric) matrix A is defined by creating an directed edge pointing from vertices i to j connect if $a_{ij} \neq 0$ [10]. LU factorization can be viewed as a process of eliminating the vertices one by one in the graph. Eliminating a vertex will connect its neighbor vertices. In the corresponding LU factorization process, many new nonzeros (or fill-in) may be introduced into the Schur complements.

To reduce fill-in, we use nested dissection [9] to first reorder the vertices in the graph. The main idea is to divide G recursively with separators or small sets of vertices. For instance, if a separator divides the graph into two sets of vertices V_1 and V_2, the vertices in V_1 or V_2 are disjoint. Figure 1 shows the separators in nested dissection and the associated tree structure, as used in [22]. This tree will be used in our factorizations as the assembly tree [7]. The separators are ordered following a bottom-up way in the tree.

Here, although G is directed, the node separators are can be obtained in the same way (see Figure 2). That is, we can apply standard nested dissection to the graph of $A + A^T$, for simplicity, as often used [6].



(i) Separators in nested dissection (ii) Corresponding assembly tree

FIG. 1. *Nested dissection for an undirected graph and the corresponding assembly tree.*

(i) Undirected mesh (ii) Directed mesh

FIG. 2. *A separator (marked with a dashed line) for an undirected graph and a directed one.*

3.2. Nonsymmetric multifrontal method. The multifrontal method [7, 16] is a very important direct method for matrix factorizations. We briefly review it here. Define is an elimination tree $\mathcal{T}$ with n nodes for A, where node p is the parent of j if and only if $p = \min\{i > j | L_{ij} + U_{ji} \neq 0\}$, where L and U are the LU factors of A [11]. The factorization of A is then organized in a bottom-up order along $\mathcal{T}$.

For a leaf node i, define

$$\mathcal{F}_i \equiv \mathcal{F}_i^0 = \left(\begin{array}{c|ccc} A_{ii} & A_{i,q_1} & \cdots & A_{i,q_n} \\ \hline A_{p_1,i} & & & \\ \vdots & & 0 & \\ A_{p_k,i} & & & \end{array} \right), \tag{3.1}$$

where p_j and q_j are appropriate nonzero row and column indices, respectively. $\mathcal{F}_i$ is called a frontal matrix associated with node i. The elimination of i from the graph results in the Schur complement

$$\mathcal{U}_i = - \begin{pmatrix} A_{p_1 i} \\ \vdots \\ A_{p_k i} \end{pmatrix} A_{ii}^{-1} \begin{pmatrix} A_{iq_1} & \cdots & A_{iq_n} \end{pmatrix},$$

which is called the update matrix.

For a nonleaf node i, the frontal matrix $\mathcal{F}_i$ is formed by some components of A and the update matrices from the children nodes using extend-add operations.

$$\mathcal{F}_i = \mathcal{F}_i^0 \bigoplus \mathcal{U}_{c_1} \bigoplus \mathcal{U}_{c_2}$$

where c_1, c_2 are the two children of i. The extend-add operation here is to match the index between the matrices by permutation and inserting zero rows/columns. It can be illustrated as follows:

$$\begin{array}{c} 2 \\ 5 \end{array} \begin{pmatrix} p & q \\ u & v \end{pmatrix} \bigoplus \begin{array}{c} 7 \\ 2 \end{array} \begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{array}{c} 2 \\ 5 \\ 7 \end{array} \begin{pmatrix} p & q & 0 \\ u & v & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{array}{c} 2 \\ 5 \\ 7 \end{array} \begin{pmatrix} y & 0 & z \\ 0 & 0 & 0 \\ w & 0 & x \end{pmatrix}.$$

Similarly, eliminate i and compute $\mathcal{U}_i$. The process then repeats. More details on the multifrontal method for a nonsymmetric A can be found in [6].

3.3. Nonsymmetric structured multifrontal method. Here, we show an direct extension of the supernodal structured multifrontal method in [20] to the nonsymmetric matrix A. Nested dissection is applied to $A + A^T$ and an assembly tree $\mathcal{T}$ is formed.

As in [20], below certain level $\mathbf{l}_s$ (called a switching level) of $\mathcal{T}$, dense local matrix operations are used. Above $\mathbf{l}_s$, HSS operations are applied instead. The switching level is selected so that the total complexity is optimized [22]. The HSS operations are described as follows.

Without loss of generality, assume $\mathcal{F}_\mathbf{i}$ corresponds to an HSS tree with $k+2$ nodes, and the children of the root $k+2$ are k and $k+1$. Also assume that the HSS form of $\mathcal{F}_\mathbf{i}$ (after a direct HSS construction) looks like

$$\mathcal{F}_\mathbf{i} = \begin{pmatrix} H & U_k B_k V_{k+1}^T \\ U_{k+1} B_{k+1} V_k^T & D_{k+1} \end{pmatrix}, \tag{3.2}$$

where H is a smaller HSS form. Apply the ULV factorization in [3, 23] to H:

$$H = L_\mathbf{i} S_\mathbf{i}.$$

The resulting update matrix looks like

$$\mathcal{U}_\mathbf{i} = D_{k+1} - U_{k+1} B_{k+1} (V_k^T H^{-1} U_k) B_k V_{k+1}^T.$$

As proven in [20], this computation can be quickly done. In fact, the ULV factorization produces a reduced matrix $\hat{D}_k$ for H, and $\mathcal{F}_\mathbf{i}$ is reduced to

$$\begin{pmatrix} \hat{D}_k & \hat{U}_k B_k V_{k+1}^T \\ U_{k+1} B_{k+1} \hat{V}_k^T & D_{k+1} \end{pmatrix}. \tag{3.3}$$

Thus [20],

$$\mathcal{U}_\mathbf{i} = D_{k+1} - U_{k+1} B_{k+1} (\hat{V}_k^T \hat{D}_k^{-1} \hat{U}_k) B_k V_{k+1}^T. \tag{3.4}$$

$\mathcal{U}_\mathbf{i}$ is then passed to the parent node in the assembly tree as in the traditional multifrontal method. Repeating the process until the root in the assembly tree is reached, then we have the final structured factorization:

$$A = \mathbf{LU}.$$

After the factorization, the solution stage is very similar to that in [20], except that the forward and backward substitutions use different blocks.

If the conditions in [20] on the off-diagonal rank patterns [19] of the frontal matrices hold, then we can similarly show that the nonsymmetric structured multifrontal method costs about $O(n)$ flops for 2D problems discretized on a $N \times N$ regular mesh ($n = N^2$) and $O(n^{4/3})$ for 3D problems discretized on a $N \times N \times N$ regular mesh ($n = N^3$). The solution cost is about $O(n)$ for both 2D and 3D, with about $O(n)$ storage for the factors.

3.4. Randomized nonsymmetric structured multifrontal method. Similarly, we can give a nonsymmetric variation for the randomized multifrontal method in [21]. We sketch the major framework as follows.

1. Starting from the nodes $\mathbf{i}$ at the switching level, compute the products

$$Y_\mathbf{i} = \mathcal{U}_\mathbf{i} X_i, \quad Z_\mathbf{i} = \mathcal{U}_\mathbf{i}^T X_i,$$

   where X_i is a skinny matrix of random vectors with the column size roughly equal to the HSS rank of $\mathcal{F}_\mathbf{i}$. More details on choosing the column size can be found in [13, 15].

2. For a node $\mathbf{i}$ above the switching level, compute two skinny extend-add operations

$$Y_{\mathbf{i}} = (\mathcal{F}_{\mathbf{i}}^0 X_i) \ddagger Y_{\mathbf{c}_1} \ddagger Y_{\mathbf{c}_2}, \quad Z_{\mathbf{i}} = ((\mathcal{F}_{\mathbf{i}}^0)^T X_i) \ddagger Z_{\mathbf{c}_1} \ddagger Z_{\mathbf{c}_2},$$

   where $\mathcal{F}_{\mathbf{i}}^0$ is extracted from A similarly to that in (3.1), and the operator $\ddagger$ denotes a skinny extend-add operation, which needs to be performed only on the rows of the participating skinny matrices.
3. Form selected entries of $\mathcal{F}_{\mathbf{i}}$ based on $\mathcal{F}_{\mathbf{i}}^0$ and the HSS forms $\mathcal{F}_{\mathbf{c}_1}$ and $\mathcal{F}_{\mathbf{c}_2}$.
4. Use the above entries of $\mathcal{F}_{\mathbf{i}}$ and $Y_{\mathbf{i}}$ and $Z_{\mathbf{i}}$ to construct an HSS approximation to $\mathcal{F}_{\mathbf{i}}$ as in [18, 24].
5. Partially factorize $\mathcal{F}_{\mathbf{i}}$ as in (3.2)–(3.4).

Then repeat the procedure. A significant benefit of this scheme is that the blocks passed along the assembly tree and also in the extend-add operations are only matrix-vector products, instead of large dense update matrices.

4. Pivoting in structured factorizations. As in standard LU factorizations, pivoting is often needed to ensure the stability. We can follow the strategies in [6, 8, 14]. In fact, static pivoting can be introduced into our method before nested dissection. Static pivoting usually permutes the large entries to the diagonal which turns out to be a weighted bipartite matching problem. There are different criterions for choosing the permutation matrix, such as maximizing the product or the sum of the diagonal entries [8]. It is found in [14] that it achieves the best performance in maximizing the product of the diagonal entries while scaling the matrix so that the diagonal entries are the largest in the columns and are all equal to one.

Sometimes, local pivoting within the frontal matrices may be sufficient. That is, before the HSS construction for $\mathcal{F}_{\mathbf{i}}$, we try to bring high rank blocks as close to the diagonal as possible. This happens to be the fundamental idea behind the rank-revealing and strong rank-revealing QR (RRQR) or LU factorizations [12]. A basic idea is, instead of compressing just an HSS block in the HSS construction, we compress the entire block row or column. The strong RRQR strategy in [12] is used so as to make the determinants of the diagonal blocks as large as possible. This not only enhances the stability, but also reduces the HSS rank.

A alternative local pivoting strategy is to still use the standard HSS construction. Then during the ULV factorization, an ill-conditioned diagonal block is not eliminated. Instead, it is directly merged with the blocks from the sibling node. This is repeated until the diagonal block is large enough and becomes well conditioned.

5. Applications to discretized Helmholtz equations. In this section, we show the applications of our method to (non-Hermitian) Helmholtz equations on regular or irregular meshes. Here, nonsymmetry often occurs due to the nature of the problem or certain boundary conditions. The structured multifrontal LU factorization in Section 3.3 (denoted `NEW`) is compared with the exact multifrontal LU factorization (denoted `MF`). In the experiments, we set the relative tolerance of compression in HSS constructions to be $\tau = 10^{-6}$. For the following examples, satisfactory accuracies are obtained without pivoting.

First, for the finite difference discretization with $N \times N$ meshes, we choose several mesh dimensions, which increase by a factor of $\sqrt{2}$ each time, so that n doubles every time. The number of dense local factorization levels is chosen to be 9. As can be seen in Table 5.1, when n doubles, the factorization cost of `NEW` increases by a factor close to 2. The costs are also plotted in Figure 3. The increase of the factorization cost of `MF` is much bigger. The relative residuals γ of the methods are reported in Table 5.2.

Our another example is a Helmholtz equation discretized on an unstructured mesh as shown in Figure 4(i), as obtained from [5]. The nonzero pattern of the corresponding matrix is in Figure 4(ii). Again, we compare `NEW` and `MF` on this problem. See Table 5.3.

TABLE 5.1
Flops counts (ξ_{fact}) of MF and NEW for regular meshes, where $\mathbf{l}_{\max}$ is the total number of levels in the assembly tree and $\mathbf{l}_{\max} - \mathbf{l}_s = 9$ is the number of bottom levels of dense factorizations.

n		255^2	360^2	511^2	723^2
$\mathbf{l}_{\max}$		13	14	15	16
ξ_{fact}	MF	$6.45E8$	$1.87E9$	$5.43E9$	$1.83E10$
	NEW	$5.33E8$	$1.33E9$	$3.10E9$	$7.45E9$

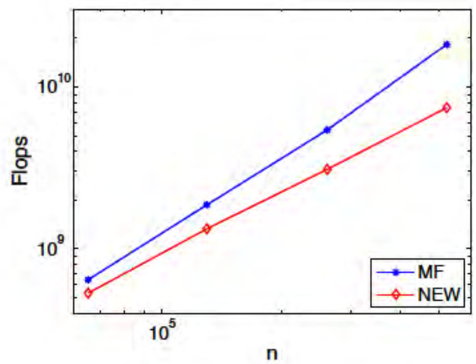
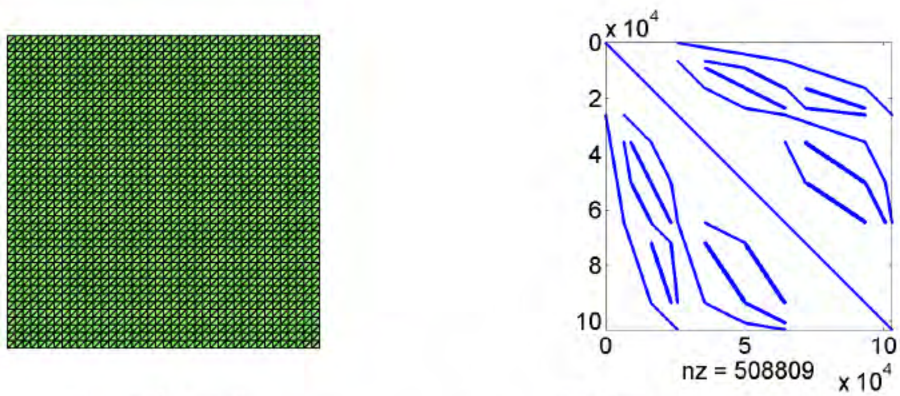


FIG. 3. *Flops counts (ξ_{fact}) of MF and NEW.*

TABLE 5.2
Relative residuals ($\gamma = \frac{\|Ax-b\|_2}{\|b\|_2}$) of NEW.

n	255^2	360^2	511^2	723^2
$\mathbf{l}_{\max}$	13	14	15	16
γ	$1.45E-8$	$1.55E-8$	$2.41E-8$	$2.29E-8$



(i) Irregular mesh (ii) Nonzero pattern of the matrix

FIG. 4. *The mesh and the nonzero pattern of a discretized Helmholtz problem from [5].*

Table 5.3
Flops counts (ξ_{fact}) of MF and NEW for irregular meshes, where $\mathbf{l}_{\max}$ is the total number of levels in the assembly tree and $\mathbf{l}_{\max} - \mathbf{l}_s = 10$ is the number of bottom levels of dense factorizations.

n		$103,041$	$160,801$	$284,089$	$641,601$
$l_{\max}$		15	16	16	17
ξ_{fact}	MF	$2.62E9$	$5.18E9$	$1.25E10$	$4.36E10$
	NEW	$2.35E9$	$4.36E9$	$1.04E10$	$3.25E10$

6. Conclusions. We show structured multifrontal methods for nonsymmetric sparse matrices, based on various existing techniques. The methods applicable on both structured mesh and unstructured meshes. The idea of reduced matrix is used to save the costs. Both static and local pivoting issues are discussed for the structured factorizations. The numerical experiments show that the new method performs better than the exact multifrontal method. The application to discretized Helmholtz equations is shown.

Parallel implementations of the methods will be given in the near future so as to solver larger problems. More general applications will also be considered.

Acknowledgements. We thank Yuanzhe Xi for helping with the paper. We are grateful to Long Chen for providing a routine in iFEM for the finite element discretization of the Helmholtz equation. The research of Jianlin Xia was supported in part by NSF grants DMS-1115572 and CHE-0957024.

REFERENCES

[1] M. Bebendorf, *Efficient inversion of Galerkin matrices of general second-order elliptic differential operators with nonsmooth coefficients*, Math. Comp., 74 (2005), pp. 1179–1199.
[2] M. Bebendorf and W. Hackbusch, *Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-Coefficients*, Numer. Math., 95 (2003), pp. 1–28.
[3] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, *A fast solver for hss representations via sparse matrices*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 67–81.
[4] S. Chandrasekaran, P. Dewilde, M. Gu, and N. Somasunderam, *On the numerical rank of the off-diagonal blocks of schur complements of discretized elliptic pdes*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2261–2290.
[5] L. Chen, *iFEM: an integrated finite element methods package in MATLAB*, Technical Report, University of California at Irvine, 2009, http://math.uci.edu/~chenlong/Papers/iFEMpaper.pdf.
[6] T. A. Davis and I. S. Duff, *An unsymmetric-pattern multifrontal method for sparse LU factorization*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 140–158.
[7] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.
[8] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.
[9] J. A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.
[10] J. A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[11] J. R. Gilbert and J. W. H. Liu, *Elimination structures for unsymmetric sparse LU factors*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 334–352.
[12] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong-rank revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.
[13] N. Halko, P.G. Martinsson, and J. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.
[14] X. S. Li and J. W. Demmel, *A scalable sparse direct solver using static pivoting*, in Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, (1999), pp. 22–24.
[15] E. Liberty, F. Woolfe, P. G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 20167–20172.
[16] J. W. H. Liu, *The multifrontal method for sparse matrix solution: Theory and practice*, SIAM Review, 34 (1992), pp. 82–109.
[17] H. M. Markowitz, *The elimination form of the inverse and its application to linear programming*, Management Sci. (1957), pp. 255–269.

[18] P. G. Martinsson, *A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix*, SIAM. J. Matrix Anal. Appl., 32 (2011), pp. 1251–1274.
[19] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410.
[20] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.
[21] J. Xia, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 197–227.
[22] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1382–1411.
[23] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.
[24] J. Xia, Y. Xi, and M. Gu, *A superfast structured solver for Toeplitz linear systems via randomized sampling*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 837–858.

Proceedings of the Project Review, Geo-Mathematical Imaging Group (Purdue University, West Lafayette IN), Vol. 1 (2013) pp. 139-156.

SUPERFAST STRUCTURED SELECTED INVERSION FOR LARGE SPARSE MATRICES

JIANLIN XIA*, YUANZHE XI†, STEPHEN CAULEY‡, AND VENKATARAMANAN BALAKRISHNAN§

Abstract. We propose a structured selected inversion method for extracting the diagonal (and certain off-diagonal) blocks of the inverse of a sparse symmetric matrix A, using the multifrontal method and rank structures. A structured multifrontal LDL factorization is computed for A with a forward traversal of the assembly tree, which yields a sequence of data-sparse factors. The factors are used in a backward traversal of the tree for the structured inversion. We show that, when A arises from the discretization of certain PDEs, the intermediate matrices in the inversion can be approximated by hierarchically semiseparable (HSS) or low-rank matrices. Due to the data sparsity, the inversion has nearly $O(n)$ complexity for some 2D and 3D discretized matrices (and is thus said to be superfast), after about $O(n)$ and $O(n^{4/3})$ flops, respectively, for the structured factorization, where n is the size of the matrix. The memory requirement is also about $O(n)$. In comparison, existing inversion methods cost $O(n^{1.5})$ in 2D and $O(n^2)$ in 3D for both the factorization and the selected inversion of the matrix, with $O(n \log n)$ and $O(n^{4/3})$ memory, respectively. Numerical tests on various PDEs and sparse matrices from a sparse matrix collection are done to demonstrate the performance.

Key words. Structured selected inversion, structured multifrontal method, data sparsity, low-rank property, linear complexity, reduced matrix

AMS subject classifications. 15A23, 65F05, 65F30, 65F50

1. Introduction.

Extracting selected entries of the inverse of a sparse matrix, often called selected inversion, is critical in many scientific computing problems. Examples include uncertainty quantification in risk analysis [2], electronic structure calculations within the density functional theory framework [14], and the quantum mechanical modeling of nanotransistors and the atomistic level simulation of silicon nanowires [4]. In these examples, the diagonal entries of the matrix inverse are needed. In some other applications such as condition estimations [3], certain off-diagonal entries are also desired. The aim of this paper is to present an efficient method for computing the diagonal (denoted $\mathrm{diag}(A^{-1})$) as well as the diagonal blocks of A^{-1} for an $n \times n$ large sparse symmetric matrix A. The method also produces some off-diagonal blocks of A^{-1}, and can be modified to compute the off-diagonal entries. For convenience, we usually just mention $\mathrm{diag}(A^{-1})$.

If A is also diagonally dominant and/or positive definite, A^{-1} may have many small entries. Based on this property, a probing method is proposed in [18]. It exploits the pattern of the sparsified matrix inverse together with some standard graph theories, and computes $\mathrm{diag}(A^{-1})$ by solving a sequence of linear systems with a preconditioned Krylov subspace algorithm. Later, several approaches are proposed for more general matrices. The fast inverse with nested dissection (FIND) method in [11] and the selected inversion method in [13] use domain decomposition and compute some hierarchical Schur complements of the interior points for each subdomain. This is followed by the extraction of the diagonal entries in a top-down pass. The Selinv method in [14, 15] uses a supernode left-looking LDL factorization of A to improve the efficiency. The method in [1] focuses on the computation of a subset of A^{-1} by accessing only part of the factors where the LU or LDL factorization of A is held in out-of-core storage. The methods in [1, 11, 13, 15] all belong to the class of direct methods. For iterative methods, a Lanczos type algorithm is first used in [20]. Later, a divide-and-conquer (DC) method and a domain decomposition (DD) method are presented in [19]. The DC method assumes that the matrix can be decomposed into a 2×2 block-diagonal matrix and a low-rank matrix recursively, where the decomposed problem is solved and corrected by the

*Department of Mathematics, Purdue University, West Lafayette, IN 47907, xiaj@math.purdue.edu. The research of Jianlin Xia was supported in part by NSF grants DMS-1115572 and CHE-0957024.

†Department of Mathematics, Purdue University, West Lafayette, IN 47907, yxi@math.purdue.edu.

‡Athinoula A. Martinos Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital, Harvard University, Charlestown, MA 02129, stcauley@nmr.mgh.harvard.edu.

§School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, ragu@ecn.purdue.edu.