An efficient block variant of robust structured multifrontal factorization method*

Zuo Xian-Yu(左宪禹)[a)†], Mo Ze-Yao(莫则尧)[b)‡], and Gu Tong-Xiang(谷同祥)[b)]

[a)] School of Computer and Information Engineering, Henan University, Kaifeng 475004, China
[b)] Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, Beijing 100088, China

(Received 4 December 2012; revised manuscript received 5 March 2013)

Based on the two-dimensional three-temperature (2D3T) radiation diffusion equations and its discrete system, using the block diagonal structure of the three-temperature matrix, the reordering and symbolic decomposition parts of the RSMF method are replaced with corresponding block operation in order to improve the solution efficiency. We call this block form method block RSMF (in brief, BRSMF) method. The new BRSMF method not only makes the reordering and symbolic decomposition become more effective, but also keeps the cost of numerical factorization from increasing and ensures the precision of solution very well. The theoretical analysis of the computation complexity about the new BRSMF method shows that the solution efficiency about the BRSMF method is higher than the original RSMF method. The numerical experiments also show that the new BRSMF method is more effective than the original RSMF method.

Keywords: HSS structure, low-rank property, multifrontal method, two-dimensional three-temperature radiative diffusion equations

PACS: 02.10.Ud **DOI:** 10.1088/1674-1056/22/8/080201

1. Introduction

Large sparse linear systems arise frequently from numerical and engineering problems, in particular, the discretization of partial differential equations (PDEs). Typically, there are two types of linear system solvers: direct methods and iterative methods. Direct methods are reliable and are efficient for multiple right-hand sides, but are often expensive due to the generation of fill-in or loss of sparsity. Iterative methods take good advantage of sparsity and require less storage, but may diverge or converge slowly if no effective preconditioners are available.

Assume we have a system

$$Ax = b,$$

where A is an $N \times N$ symmetric positive-definite (SPD) matrix. If A arises from the discretization of some PDEs, it may be associated with a mesh. In the direct solution of the system, A or the mesh points can be reordered so as to reduce fill-in. For example, the nested dissection ordering[1] and its generalizations can be used to get nearly optimal exact factorization complexity, which is generally $O(N^{3/2})$ in two dimensions or $O(N^2)$ in three dimensions.[2] In nested dissection, a mesh is recursively divided with separators (small sets of mesh points).

In recent years, nearly linear complexity, structured approximate factorization methods have been developed based on a low-rank property. It has been noticed that, during the direct solution of some PDEs such as elliptic equations, certain off-diagonal blocks of the intermediate dense matrices or fill-in have small numerical ranks.[3–7] This property can be used to improve the computational efficiency, with dense intermediate matrices approximated by rank structured matrices such as quasiseparable, semiseparable, or hierarchical matrices.[3,8–11] This idea is widely used in the development of new fast algorithms. Related techniques have also been shown to be very useful in high performance scientific computing.[12–14] Rank structured methods can be fully integrated into sparse matrix techniques to provide new fast solvers. Structured sparse factorization algorithms[15] are proposed based on the multifrontal method[16,17] and the robust HSS Cholesky factorization algorithm.[18]

In inertial confinement fusion, i.e., in the high energy density physics field, numerical simulation is the main research method.[19] The three-temperature linear solver of solving radiation fluid dynamics equations is the important component of the numerical simulation. In many cases, the three-temperature linear solver's CPU time accounts for more than 90% of the total simulation time. In recent years, many authors for two-dimensional three-temperature (2D3T) equations have done a lot of work and developed a lot of algorithms, for example.[20–27] With the large-scale numerical simulation development of this kind of application, the linear solver still

*Project supported by the National Natural Science Foundation of China (Grant Nos. 61202098, 61033009, 61170309, 91130024, and 11171039) and the China Tianyuan Mathematics Youth Fund (Grant No. 11226337).

†Corresponding author. E-mail: xianyu_zuo@163.com
‡Corresponding author. E-mail: zeyao_mo@iapcm.ac.cn

© 2013 Chinese Physical Society and IOP Publishing Ltd

is the main bottleneck restricting the performance of the numerical simulation. To solve these problems before, the iteration method[28] was most popular. Due to the high cost, direct method was often used as a preconditioner, for example, ILU(k) etc. This paper attempts to use the block-diagonal structure of the three-temperature matrix to improve the RSMF method, that is BRSMF method, and we use it as a three-temperature linear solver. Numerical experiments show that the new BRSMF method is more effective than the original RSMF method.

The rest of this paper is organized as follows. Section 2 reviews the robust structured multifrontal factorization method. Section 3 presents 2D3T radiation diffusion equations and its discrete system. The detailed BRSMF method is discussed in Section 4. Section 5 demonstrates the efficiency of the BRSMF method with numerical experiments in terms of the model problem and practical application problems. Section 6 gives some general remarks.

2. Robust structured multifrontal factorization method

2.1. Robust HSS Cholesky factorization

Given a dense real SPD matrix F, use the method[18] to compute an approximate Cholesky factorization $F = (\approx)LL^{\mathrm{T}}$, where L is a lower triangular HSS matrix. LL^{T} generally exists for any given approximation accuracy. The fundamental idea can be illustrated in terms of a block 2×2 SPD matrix. Factorize the $(1,1)$ block of the following matrix:

$$F = \begin{pmatrix} F_{1,1} & F_{2,1}^{\mathrm{T}} \\ F_{2,1} & F_{2,2} \end{pmatrix} = \begin{pmatrix} D_1 & \\ F_{2,1}D_1^{-\mathrm{T}} & I \end{pmatrix} \begin{pmatrix} D_1^{\mathrm{T}} & D_1^{-1}F_{2,1}^{\mathrm{T}} \\ & S \end{pmatrix},$$

where $F_{1,1} = D_1 D_1^{\mathrm{T}}$ is the Cholesky factorization of $F_{1,1}$, and $S = F_{2,2} - (F_{2,1}D_1^{-\mathrm{T}})(D_1^{-1}F_{2,1}^{\mathrm{T}})$ is the Schur complement. Compute an SVD $F_{2,1}D_1^{-\mathrm{T}} = U_2 B_1^{\mathrm{T}} U_1^{\mathrm{T}} + \hat{U}_2 \hat{B}_1^{\mathrm{T}} \hat{U}_1^{\mathrm{T}}$, where all the singular values greater than a tolerance τ are in B_1. (The number of singular values in B_1 is the off-diagonal *numerical rank* r.) Then

$$F \approx \begin{pmatrix} D_1 & \\ U_2 B_1^{\mathrm{T}} U_1^{\mathrm{T}} & I \end{pmatrix} \begin{pmatrix} D_1^{\mathrm{T}} & U_1 B_1 U_2^{\mathrm{T}} \\ & \tilde{S} \end{pmatrix},$$

where $\tilde{S}$ is an approximate Schur complement given by

$$\tilde{S} = F_{2,2} - U_2 B_1^2 U_2^{\mathrm{T}} = S + O(\tau^2),$$

that is, a positive semidefinite term is implicitly added to the Schur complement. Then a Cholesky factorization $\tilde{S} = D_2 D_2^{\mathrm{T}}$ yields

$$F \approx LL^{\mathrm{T}}, \quad L = \begin{pmatrix} D_1 & \\ U_2 B_1^{\mathrm{T}} U_1^{\mathrm{T}} & D_2 \end{pmatrix}.$$

Therefore, we obtain an approximate Cholesky factor L which is a block 2×2 HSS form. It is also shown that, with certain modifications, L can work as an effective preconditioner[18] when the low-rank property is insignificant. If the HSS rank of A for a small tolerance is large, L can be obtained by manually choosing a small rank r (and a large tolerance). The idea can be generalized to multiple blocks so that L is a general lower-triangular HSS matrix.

2.2. Nested dissection ordering

Consider the mesh used in the discretization as an undirected graph $G = (V;E)$. Each vertex $i \in V$ corresponds to a row and a column of A, and each edge $(i;j) \in E$ corresponds to entries $aij = aji \neq 0$. Nested dissection recursively divides the mesh into subregions with separators.[1,29] At the first level, a separator divides the entire grid into two subregions (plus the separator itself). The two subregions are further divided recursively. See Fig. 1.

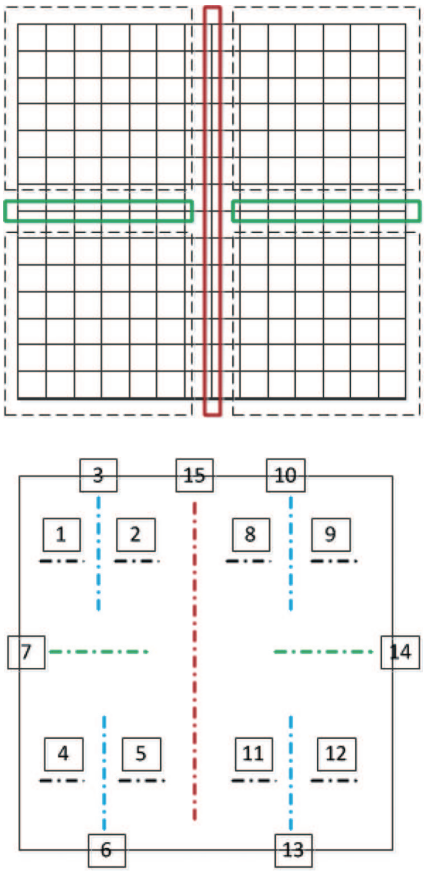


Fig. 1. (color online) Separators in nested dissection.

Lower level separators are ordered before upper level ones. During the factorization of A, the elimination of a mesh point mutually connects points which are previously connected to it.[30,31] This creates fill-in. For simplicity of presentations, we look at an $n\times n$ regular mesh such as the one in Fig.1. After reordering, the matrix A has a nonzero pattern as shown in Fig. 2. Then clearly, the nodes in the top level separator are mutually connected and form a dimension $O(n)$ dense matrix whose exact factorization costs at least $O(n^3)$ flops.

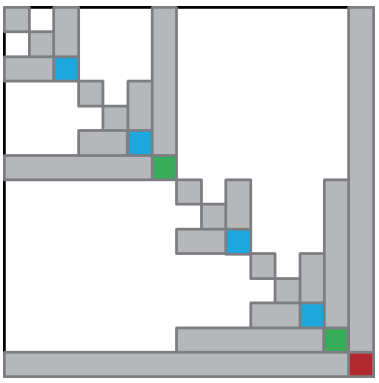


Fig. 2. (color online) Nonzero pattern of A after the nested dissection ordering.

2.3. Multifrontal method

The multifrontal method[16,17] reorganizes the sparse Cholesky factorization $A = \mathscr{L}\mathscr{L}^{\mathrm{T}}$ into local factorizations of intermediate dense matrices, where $\mathscr{L}$ is lower triangular. The factorization is conducted following a tree called an elimination tree, or more generally, an assembly tree. In general, an elimination tree $\mathscr{T}$ has n nodes and a node p is the parent of i if and only if

$$p = \min\{j > i | \quad \mathscr{L}|_{j\times i} \neq 0\},$$

where $\mathscr{L}|_{j\times i}$ represents the (j,i) entry of L. Use $T[i]$ to denote the subtree of T with root i. Let $\mathscr{N}_i \equiv \{j_1, j_2, \ldots, j_d\}$ be the set of row indices of nonzeros in $\mathscr{L}_{:,i}$ (the i-th column of $\mathscr{L}$) with i excluded. The i-th *frontal matrix* is defined to be

$$\mathscr{F}_i = \begin{pmatrix} A|_{i\times i} & (A|_{\mathscr{N}_i\times i})^{\mathrm{T}} \\ A|_{\mathscr{N}_i\times i} & 0 \end{pmatrix} - \sum_{j\in\mathscr{T}[i]\setminus i} \mathscr{L}|_{(i\cup\mathscr{N}_i)\times j}(\mathscr{L}|_{(i\cup\mathscr{N}_i)\times j})^{\mathrm{T}}.$$

One step of elimination applied to $\mathscr{F}_i$ provides the column $\mathscr{L}|_{(i\cup\mathscr{N}_i)\times i}$:

$$\mathscr{F}_i = \begin{pmatrix} \mathscr{L}|_{i\times i} & 0 \\ \mathscr{L}|_{\mathscr{N}_i\times i} & I \end{pmatrix} \begin{pmatrix} (\mathscr{L}|_{i\times i})^{\mathrm{T}} & (\mathscr{L}|_{\mathscr{N}_i\times i})^{\mathrm{T}} \\ 0 & \mathscr{U}_i \end{pmatrix},$$

where $\mathscr{U}_i$ is the contribution from $\mathscr{T}[i]$ to p and is called the i-th update matrix:

$$\mathscr{U}_i = -\sum_{j\in\mathscr{T}[i]} \mathscr{L}|_{\mathscr{N}_i\times j}(\mathscr{L}|_{\mathscr{N}_i\times j})^{\mathrm{T}}.$$

Update matrices are used to form upper level frontal matrices. This process is called an extend–add operation, which matches indices and add entries, denoted by

$$\mathscr{F}_i = \begin{pmatrix} A|_{i\times i} & (A|_{\mathscr{N}_i\times i})^{\mathrm{T}} \\ A|_{\mathscr{N}_i\times i} & 0 \end{pmatrix} \circledast \mathscr{U}_{c_1} \circledast \mathscr{U}_{c_2} \circledast \cdots \circledast \mathscr{U}_{c_q},$$

where nodes $c_1, c_2, \ldots, c_q$ are the children of i in the elimination tree. The elimination process then repeats along the elimination tree.

2.4. Structured supernodal multifrontal factorization

Here, use nested dissection to reorder A and to produce a binary tree $\mathscr{T}$ as the assembly tree in a supernodal version of the multifrontal method, where each separator is treated as a node in the tree. Figure 3(a) below shows the assembly tree for the mesh in Fig. 1.

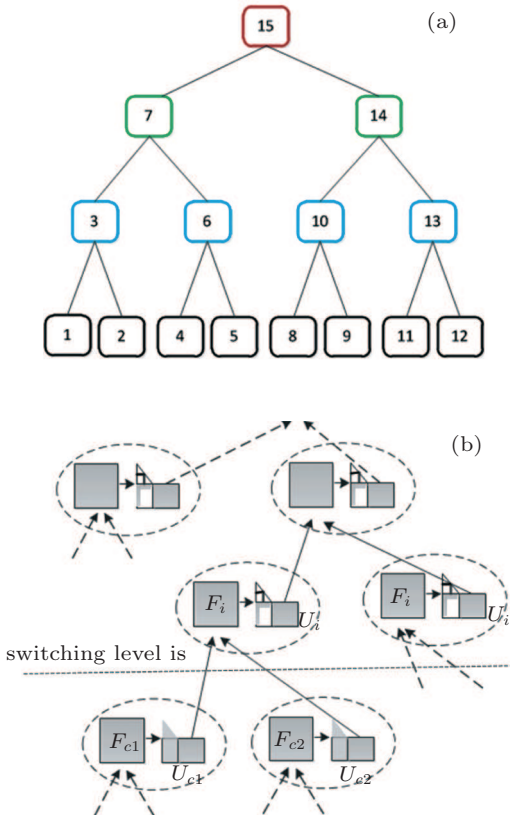


Fig. 3. (color online) (a) Supernodal elimination tree $\mathscr{T}$ and (b) robust structured multifrontal-factorization.

Assume the root of $\mathscr{T}$ is at level 0, and the leaves are at the largest level. For a separator i, let $\mathscr{N}_i \equiv \{j_1, j_2, \ldots, j_d\}$ be the set of neighbor separators of i at the same or upper levels of i in T. Also let t_j denote the index set of the neighbor j in A, and let $\hat{t}_j$ denote the subset of t_j that is connected to i due to lower level eliminations. The frontal matrix $\mathscr{F}_i$ is formed by the block form extend–add operation

$$\mathscr{F}_i = \mathscr{F}_i^0 \circledast \mathscr{U}_{c_1} \circledast \mathscr{U}_{c_2},$$
$$\mathscr{F}_i^0 \equiv \begin{pmatrix} A|_{t_i\times t_i} & (A|_{(\cup_{j=1}^d \hat{t}_j)\times t_i})^{\mathrm{T}} \\ A|_{(\cup_{j=1}^d \hat{t}_j)\times t_i} & 0 \end{pmatrix},$$

where c_1 and c_2 are the children of i. For notational convenience, rewrite $\mathscr{F}_i$ as

$$\mathscr{F}_i \equiv \begin{pmatrix} F_{i,i} & F_{\mathscr{N}_i,i}^{\mathrm{T}} \\ F_{\mathscr{N}_i,i} & F_{\mathscr{N}_i,\mathscr{N}_i} \end{pmatrix}.$$

In the structured multifrontal method, set a switching level l_s so that if a separator i is at level l of T and $l > l_s$, use exact Cholesky factorizations, and otherwise, use HSS Cholesky

factorizations. See Fig. 3(b). Similar to Ref. [7], this can help minimize the cost, which is smaller than using structured factorizations at all levels as in Ref. [32] by a factor up to $O(\log_n)$. This is justified by Theorem 1 below. Here, we only need to describe the structured factorization part, that is the robust structured multifrontal factorization method (in brief, RSMF). The RSMF method can be presented in algorithm form as:

Algorithm 1 (Robust structured multifrontal factorization[15])

1) Mesh ordering: Use nested dissection to order the mesh points. Build a post-ordering elimination tree with m nodes.

2) Symbolic factorization: Decide the switching level l_s. Based on the connections of separators and off-diagonal numerical ranks (or storage restriction), predict the structure of the factor. Allocate storage for update matrix stacks and the structured factor.

3) Numerical factorization: For separators $i = 1, \ldots, m$

a) If i is a leaf node, set i to be the i-th initial frontal matrix.

b) If i is at level $l_i > l_s$

b1) Apply the traditional Cholesky factorization to partially factorize $\mathscr{F}_{ii}$ until separator i is eliminated.

b2) Compute $\mathscr{U}_i$, the Schur complement.

else

b3) Apply the HSS structured Cholesky factorization to partially factorize $\mathscr{F}_{ii}$ until separator i is eliminated.

b4) Compute $\mathscr{U}_i$ with a low-rank update.

c) If i is a left node, push $\mathscr{U}_i$ onto the update matrix stack.

d) If i is a right node, pop an update matrix $\mathscr{U}_j$ from the update matrix stack (j is the sibling of i). Use the traditional extend–add procedure to obtain the frontal matrix of the parent node of i.

The following theorem gives the complexity of the Algorithm 1.

Theorem 1[15] Assume Algorithm 1 is used to factorize a discretized matrix A on an $n \times n$ regular mesh, and the elimination tree is a complete binary tree. Also assume the numerical ranks of all HSS off-diagonal blocks of the frontal matrices in the factorization are bounded by r. Then the complexity of Algorithm 1 is $O(rn^2 \log_2 n)$, if the elimination tree has $l = 2|\log_2 n|$ levels with the switch level l_s chosen, such that the cost for the bottom level standard factorizations is the same as the cost for upper level structured factorizations. In such a situation, the storage for the structured factor is $O(n^2 \log_2(r \log_2 n))$.

3. 2D3T radiation diffusion equations and its discrete system

The 2D3T radiation diffusion equations are defined as[25]

$$\begin{cases} C_{ve}\dfrac{\partial T_e}{\partial t} - \dfrac{1}{\rho}\nabla\cdot(K_e\nabla T_e) = \omega_{er}(T_r - T_e) + \omega_{ei}(T_i - T_e), \\ C_{vr}\dfrac{\partial T_r}{\partial t} - \dfrac{1}{\rho}\nabla\cdot(K_r\nabla T_r) = \omega_{er}(T_e - T_r), \\ C_{vi}\dfrac{\partial T_i}{\partial t} - \dfrac{1}{\rho}\nabla\cdot(K_i\nabla T_i) = \omega_{ei}(T_e - T_i) \end{cases} \tag{1}$$

with $(x,y) \in \Omega_{xy}$ and $0 \le t \le T_{\max}$. Equation (1) are the electron, photon, and ion temperature equations, respectively. The computational domain consists of several subdomains, and each containing some one material. In equations (1),

E1a) T_α (α = e, r, i) are the temperatures needed to be calculated;

E1b) $C_{v\alpha}$ (α = e, r, i) are isochore specific heat coefficients;

E1c) ρ is the density of the material;

E1d) $K_\alpha = K_\alpha(\rho, T_\alpha)$ (α = e, r, i) are the heat conduction coefficients; and

E1e) ω_{er} and ω_{ei} are the energy exchange coefficients between electron and photon, and electron and ion, respectively.

Specifically, $C_{v\alpha}$, K_α, and $\omega_{e\alpha}$ are defined, respectively, as follows:

$$C_{v\alpha} = \begin{cases} c_e, & \alpha = \mathrm{e}, \\ c_r T_r^3, & \alpha = \mathrm{r}, \\ c_i, & \alpha = \mathrm{i}, \end{cases} \qquad K_\alpha = \begin{cases} A_e T_e^{5/2}, & \alpha = \mathrm{e}, \\ A_r T_r^{3+\delta}, & \alpha = \mathrm{r}, \\ A_i T_i^{5/2}, & \alpha = \mathrm{i}, \end{cases}$$

$$\omega_{e\alpha} = \begin{cases} A_{er}\rho T_e^{-1/2}, & \alpha = \mathrm{r}, \\ A_{ei}\rho T_{ei}^{-2/3}, & \alpha = \mathrm{i}, \end{cases}$$

where c_α are constants, A_α, $A_{e\alpha}$, β and ρ are continuous in the interior of the material.

The boundary and initial conditions are:

(i) boundary conditions:

i-1 rigid walls: $K_\alpha \nabla T_\alpha \cdot \boldsymbol{n} = 0$, $\alpha = \mathrm{e,r,i}$, where $\boldsymbol{n}$ is the outer normal vector along boundary;

i-2 free surfaces: $K_\alpha \nabla T_\alpha \cdot \boldsymbol{n} = 0$, $\alpha = \mathrm{e,i}$; $T_\mathrm{r} = T_\mathrm{r}(t,x,y)|_{(x,y)\in\partial\Omega_{xy}}$.

(ii) concatenation conditions across material interfaces: T_α, $K_\alpha \nabla T_\alpha \cdot \boldsymbol{n}$ are continuous over interfaces, $\alpha = \mathrm{e,r,i}$, $\boldsymbol{n}$ is the outer normal vector;

(iii) initial conditions:

$$T_\alpha(0,x,y) = T_\alpha^0(x,y), \quad \alpha = \mathrm{e,i,r}.$$

The system energy of unit mass is defined by $E = E_\mathrm{e} + E_\mathrm{r} + E_\mathrm{i}$, where $E_\mathrm{e} = c_\mathrm{e} T_\mathrm{e}$, $E_\mathrm{i} = c_\mathrm{i} T_\mathrm{i}$, $E_\mathrm{r} = \frac{1}{4} c_\mathrm{r} T_\mathrm{r}^4$.

Consider the rectangular computing domain $\Omega_{xy} = [0,A] \times [0,B]$ where the up boundary is the free surface and the

rest are the rigid walls. The 2D3T radiation diffusion equations (1) are discretized by the fully finite volume method. Specifically, the domain Ω_{xy} is zoned into $L \times M$ meshes by some lines paralleling to the axes, where the step size in x direction is $h_1 = A/L$ and in y direction is $h_2 = B/M$. The temperatures, energy and density are defined at the center points of the meshes. The 2D3T radiation diffusion equations are a highly nonlinear system of equations. Combining nonlinear iterative methods, for example, Newton or Picard iterative method,[20] a linear algebra system of equations should be solved in each nonlinear iterative step, where the coefficients matrix has the following structure:

$$\begin{pmatrix} A_1 & S_1 & & & & \\ X_1 & A_2 & S_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & X_{M-2} & A_{M-1} & S_{M-1} & \\ & & & X_{M-1} & A_M & \end{pmatrix}_{N\times N},$$

where $N = 3LM$ and the matrices $S_m, X_m, (m = 1, 2, \ldots, M)$ have the following structure:

$$S_m = \begin{pmatrix} S_{m,1} & & & \\ & S_{m,2} & & \\ & & \ddots & \\ & & & S_{m,L} \end{pmatrix},$$

$$X_m = \begin{pmatrix} X_{m,1} & & & \\ & X_{m,2} & & \\ & & \ddots & \\ & & & X_{m,L} \end{pmatrix},$$

and the matrices $A_m, (m = 1, 2, \ldots, M)$ have the following structure:

$$A_m = \begin{pmatrix} A_{m,1} & Y_{m,1} & & & & \\ Z_{m,1} & A_{m,2} & Y_{m,2} & & & \\ & \ddots & \ddots & \ddots & & \\ & & Z_{m,L-2} & A_{m,L-1} & Y_{m,L-1} & \\ & & & Z_{m,L-1} & A_{m,L} & \end{pmatrix}.$$

In the above matrices, each $S_{m,l}, X_{m,l}, Y_{m,l}, Z_{m,l}$ are all 3×3 diagonal matrices, and $A_{m,l}$ are 3×3 matrices with the following structure:

$$A_{m,l} = \begin{pmatrix} * & * & * \\ * & * & 0 \\ * & 0 & * \end{pmatrix}.$$

4. BRSMF method

In this section, we mainly use the block-diagonal structure of the three-temperature matrix to improve the RSMF method. The fundamental idea of the BRSMF method can be illustrated in terms of a simple example of 2D3T equations based on 3×3 rectangular meshes. Using the implicit finite volume method, we can get the nonzero structure of the three-temperature matrix in Fig. 4.

By the RSMF method, we know that matrix reordering and symbolic factorization must be executed before numerical factorization. Reordering is to reduce the number of fill-in produced in the process of numerical factorization. Using symbolic factorization is to realize numerical factorization quickly. Based on the adjacency graph of matrix, the RSMF method realizes matrix reordering by using nested dissection. Figure 4 demonstrates the adjacency graph of the matrix corresponding Fig. 5.

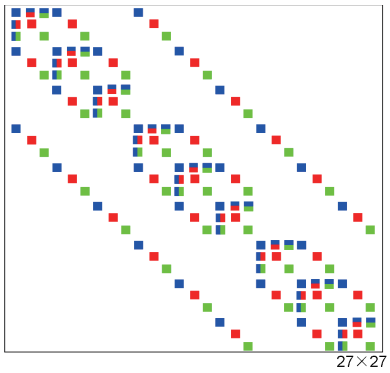


Fig. 4. (color online) Nonzero structure of three-temperature matrix.

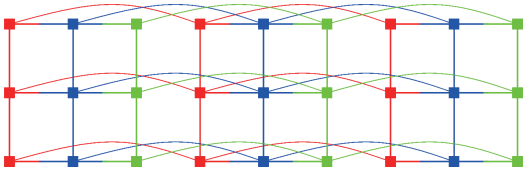


Fig. 5. (color online) Adjacency graph of matrix corresponding to Fig. 4.

Although the number of practical vertexes is 9 in the discretization mesh, the number of vertexes in the adjacency graph, Fig. 5, is 27. The reason for this is the coupling relationship of electron, photon, and ion temperatures, that is, each vertex corresponding to three unknowns. When we use the RSMF method to solve the linear system of equations, reordering is executed justly based on this adjacency graph. From Fig. 5, we know that the adjacency subgraphs? corresponding electron, photon, and ion temperatures are the same, if we remove the coupling relationship among electron, photon, and ion temperatures. Figure 6 demonstrates the adjacency subgraph corresponding to each temperature.

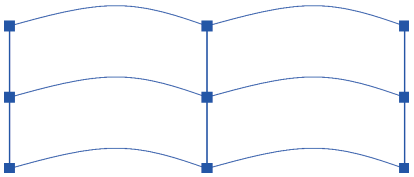


Fig. 6. (color online) Adjacency subgraph corresponding to each temperature.

From the above analyses, we know that electron, photon, and ion temperatures have the same nonzero structure in the original global matrix without the coupling relationships among them, that is, their adjacency relationships are the same. Therefore, if the corresponding three vertexes with electron, photon, and ion temperatures are looked at as one supernode, one adjacency graph with 9 supernodes will correspond to the original adjacency graph in Fig. 5, which is called the auxiliary adjacency graph. The corresponding matrix with the auxiliary adjacency graph is called the auxiliary matrix. Figures 7 and 8 demonstrate the auxiliary adjacency graph and auxiliary matrix respectively. The diagonal elements and the off-diagonal elements of the auxiliary matrix are all 3×3 matrices, which have the following nonzero structure respectively

$$\begin{pmatrix} * & * & * \\ * & * & 0 \\ * & 0 & * \end{pmatrix}, \quad \begin{pmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix}.$$

So far, the nested dissection reordering could be executed based on the auxiliary matrix and its adjacency graph. Therefore, the scale of the problem is one-third of the original, that is, the number of the original adjacency graph in Fig. 5 is three times as much as the auxiliary adjacency graph. Symbolic factorization is executed along with the elimination tree generated using the auxiliary adjacency graph. In the numerical factorization stage, the original numerical elements corresponding with the coupling relationship and the adjacency relationship of themselves should be involved in the numerical computation. Thus, the size of each separator produced in the BRSMF method should be expanded three times. The BRSMF method has the following form:

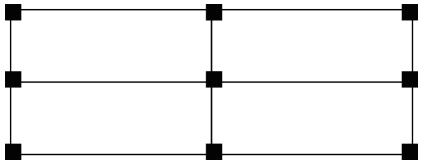


Fig. 7. Auxiliary adjacency graph.

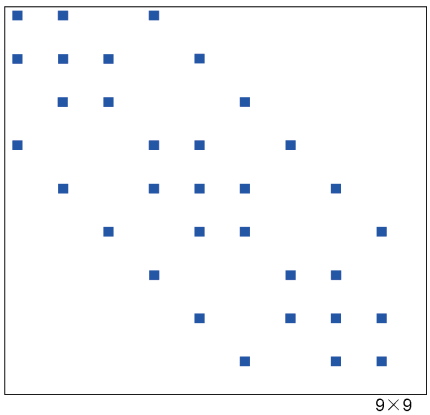


Fig. 8. (color online) Auxiliary matrix.

Algorithm 2 (Block robust structured multifrontal factorization, BRSMF)

1) Mesh ordering: Use nested dissection to order the adjacency graph of the auxiliary matrix. Build a post-ordering elimination tree with m nodes.

2) Symbolic factorization: Using the elimination tree of the auxiliary adjacency graph, decide the switching level l_s. Based on the connections of separators and off-diagonal numerical ranks (or storage restriction), predict the structure of the factor. Allocate storage for update matrix stacks and the structured factor.

3) Numerical factorization: For separators $i=1,\ldots,m$ expand the dimensions of matrices $\mathscr{F}_i$ and $\mathscr{U}_i$ three times, that is, the corresponding numerical elements in the original matrix involved in the computation.

a) If i is a leaf node, set $\mathscr{F}_i$ to be the i-th initial frontal matrix.

b) If i is at level $l_i > l_s$

b1) Apply the traditional Cholesky factorization to partially factorize $\mathscr{F}_{ii}$ until separator i is eliminated.

b2) Compute $\mathscr{U}_i$, the Schur complement.

else

b3) Apply the HSS structured Cholesky factorization to partially factorize $\mathscr{F}_{ii}$ until separator i is eliminated.

b4) Compute $\mathscr{U}_i$ with a low-rank update.

c) If i is a left node, push $\mathscr{U}_i$ onto the update matrix stack.

d) If i is a right node, pop an update matrix $\mathscr{U}_j$ from the update matrix stack (j is the sibling of i). Use the traditional extend–add procedure to obtain the frontal matrix of the parent node of i.

About the complexity of Algorithm 2, we have the following conclusions.

In Algorithm 2, the corresponding three vertexes with electron, photon, and ion temperatures are looked at as one supernode. Therefore, the scale of the problem is one-third of the original. The adjacency graph and matrix based on supernodes are called the *auxiliary adjacency graph* and *auxiliary matrix* respectively.

As the nested dissection reordering in Algorithm 2 is executed based on the auxiliary matrix and its adjacency graph, the computation complexity is one-third of the nested dissection reordering in Algorithm 1.

In Algorithm 2, the symbolic factorization is executed along with the elimination tree generated using the auxiliary adjacency graph. At the same time, the symbolic factorization is executed without the coupling relationships among electron, photon, and ion temperatures. The computation complexity of the symbolic factorization in Algorithm 2 is less than one-third of the corresponding part in Algorithm 1.

In Algorithm 2, the numerical factorization is executed based on the original numerical elements corresponding to the

coupling relationship and the adjacency relationship of themselves, that is, electron, photon, and ion temperatures. The computation complexity of the numerical factorization in Algorithm 2 and Algorithm 1 has the same order of magnitude, that is, $O(rn^2 \log_2 n)$. But, from the following numerical experiments, we can see that the computation time of the numerical factorization in Algorithm 2 is always less than the time in Algorithm 1 with the increase of problem size.

5. Numerical experiments

5.1. Example 1

We first use the standard 5-point discretized stencil to demonstrate the BRSMF method's efficiency, where A is a block 5-diagonal sparse SPD matrix:

$$A = \begin{pmatrix} T & -I & & 0 \\ -I & \ddots & \ddots & \\ & \ddots & \ddots & -I \\ 0 & & -I & T \end{pmatrix},$$

$$T = \begin{pmatrix} D & -I & & 0 \\ -I & \ddots & \ddots & \\ & \ddots & \ddots & -I \\ 0 & & -I & D \end{pmatrix},$$

$$D = \begin{pmatrix} 6 & -1 & -1 \\ -1 & 5 & 0 \\ -1 & 0 & 5 \end{pmatrix}.$$

Here, we let the matrix size N range from 120^2 to 960^2, total level from 9 to 12, switching level from 6 to 8, the tolerance τ be 1.0×10^{-6}, the initial guess $x_0 = (0, 0, \ldots, 0)^{\mathrm{T}}$ and the exact solution be $x^* = (1, 1, \ldots, 1)^{\mathrm{T}}$. The minimal number of vertexes involved in a separator for the RSMF method and BRSMF method are 30 and 10 respectively.

Tables 1 and 2 demonstrate the experimental results of the RSMF and BRSMF methods for different problem sizes respectively, where T_{total} is the total execution time, T_{ND} is the nested dissection reordering time, T_{sym} and T_{num} are symbolic and numerical factorization time respectively, and T_{sub} is the forward and backward substitution time. Results show that the improvements of nested dissection reordering and symbolic factorization are very notable. At the same time, the time of numerical factorization has been improved effectively. We can see that the improvement becomes more and more remarkable with the increase of problem size. The percentage of improvement of the BRSMF method relative to the RSMF method for the total execution time is demonstrated in Fig. 9, that is, $1 - T_{\mathrm{BR}}/T_{\mathrm{R}}$, where T_{BR} and T_{R} are the total execution time for the BRSMF method and the RSMF method respectively. Figure 9 show that the whole solving efficiency increases about one time for larger problem size.

Table 1. Experimental results of RSMF execution time.

Mesh size	RSMF				
N	T_{total}	T_{ND}	T_{sym}	T_{num}	T_{sub}
120×120	0.234	0.105	0.400×10^{-1}	0.781×10^{-1}	0.117×10^{-1}
240×240	0.119×10^{1}	0.309	0.277	0.543	0.625×10^{-1}
480×480	0.752×10^{1}	0.143×10^{1}	0.236×10^{1}	0.346×10^{1}	0.266
960×960	0.493×10^{2}	0.809×10^{1}	0.213×10^{2}	0.189×10^{2}	0.992

Table 2. Experimental results of BRSMF execution time.

Mesh size	BRSMF				
N	T_{total}	T_{ND}	T_{sym}	T_{num}	T_{sub}
120×120	0.152	0.430×10^{-1}	0.078×10^{-1}	0.898×10^{-1}	0.117×10^{-1}
240×240	0.742	0.133	0.051	0.496	0.625×10^{-1}
480×480	0.381×10^{1}	0.426	0.269	0.285×10^{1}	0.262
960×960	0.227×10^{2}	0.268×10^{1}	0.244×10^{1}	0.166×10^{2}	0.984

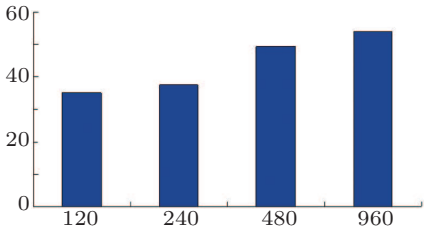


Fig. 9. (color online) The percentage of improvement of BRSMF method (%).

Table 3. Comparison of the Flops time and solution accuracy.

Mesh size	RSMF		BRSMF	
N	Flops	Rel.	Flops	Rel.
120×120	0.131×10^{9}	0.358×10^{-6}	0.132×10^{9}	0.264×10^{-6}
240×240	0.135×10^{10}	0.190×10^{-6}	0.135×10^{10}	0.346×10^{-6}
480×480	0.929×10^{10}	0.128×10^{-5}	0.904×10^{10}	0.115×10^{-5}
960×960	0.476E+11	0.751×10^{-5}	0.465E+11	0.667×10^{-5}

For different problem sizes, flops of numerical factoriza-

tion and solution accuracy of both methods are compared in Table 3. The results show that the BRSMF method not only keeps the cost of numerical factorization from increasing, but ensures the precision of solution very well.

5.2. Example 2

For comparing the efficiency of both RSMF and BRSMF methods, we extract four SPD three-temperature linear systems of equations from LARED-H program involved in JASMIN.[33] Four linear systems are numbered Mat1, Mat2, Mat3, and Mat4 respectively, whose dimensions are all 27849. Table 4 demonstrates the experimental results of both methods, where the total level l is 10, the switching level l_s is 7, the tolerance τ is 1.0×10^{-3}, the initial guess $x_0=(0,0,\ldots,0)^{\mathrm{T}}$ and the exact solution is $x^*=(1,1,\ldots,1)^{\mathrm{T}}$. The minimal number of vertexes involved in a separator for the RSMF method and the BRSMF method are 30 and 10 respectively.

Table 4. Performance comparison of both methods.

Matix	RSMF			BRSMF		
	T_{total}	Rel.	Flops	T_{total}	Rel.	Flops
Mat1	0.625	0.186×10^{-15}	0.407×10^{9}	0.258	0.187×10^{-15}	0.344×10^{9}
Mat2	0.617	0.117×10^{-11}	0.382×10^{9}	0.266	0.256×10^{-11}	0.355×10^{9}
Mat3	0.594	0.173×10^{-11}	0.382×10^{9}	0.273	0.701×10^{-10}	0.354×10^{9}
Mat4	0.594	0.389×10^{-11}	0.382×10^{9}	0.266	0.541×10^{-10}	0.354×10^{9}

From the results in Table 4, the percentage of improvement of the total execution time achieves about 50%, and the numerical factorization time is also improved effectively. Figure 10 illustrates the percentage of the total time improvement. Moreover, the results further verify that the BRSMF method not only keeps the cost of numerical factorization from increasing, but ensures the precision of solution very well.

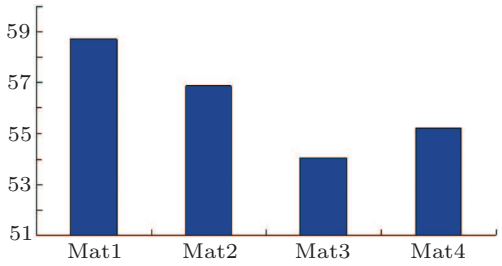


Fig. 10. (color online) The percentage of improvement of the BRSMF method (%).

6. Conclusion

In this paper, we proposed the BRSMF method by making use of the block diagonal structure of the three-temperature matrix. As the nested dissection reordering and symbolic decomposition in the BRSMF method are executed based on the auxiliary adjacency graph and auxiliary matrix, respectively, the total solving efficiency is improved significantly. From the numerical experiments, we can see that the computation time of the reordering in the BRSMF method is about one-third of the time in the RSMF method, the computation time of the symbolic factorization in the BRSMF method is less than one-third of the time in the RSMF method and the computation time of the numerical factorization in the BRSMF method is always less than the time in the RSMF method with the increase of problem size. These results show that numerical experiments agree with our theoretical analysis and the improvement becomes more and more remarkable with the increase of problem size.

References

[1] George J A 1973 *SIAM J. Numer. Anal.* **10** 345
[2] Hoffman A J, Martin M S and Rose D J 1973 *SIAM J. Numer. Anal.* **10** 364
[3] Bebendorf M and Hackbusch W 2003 *Numer. Math.* **95** 1
[4] Bebendorf M 2005 *Math. Comp.* **74** 1179
[5] Chandrasekaran S, Dewilde P and Gu M 2010 *SIAM J. Matrix Anal. Appl.* **31** 2261
[6] Grasedyck L, Kriemann R and Le Borne S 2006 *Springer LNCSE* **55** 661
[7] Xia J L, Chandrasekaran S, Gu M and Li X 2009 *SIAM J. Matrix Anal. Appl.* **31** 1382
[8] Börm S, Grasedyck L and Hackbusch W 2002 *Computing* **69** 1
[9] Eidelman Y and Gohberg If 1999 *Integral Equations Operator Theory* **34** 293
[10] Hackbusch W 1999 *Computing* **62** 89
[11] Vandebril R, Van Barel M, Golub G and Mastronardi N 2005 *A Bibliography on Semiseparable Matrices Calcolo* **42** 249
[12] Polizzi E and Sameh A 2006 *Parallel Comput.* **32** 177
[13] Sambavarama S R, Sarin V, Sameh A and Grama A 2003 *Parallel Comput.* **29** 1261
[14] Wang S, Li X S, Xia J L, Situ Y and de Hoop M V 2012 *Revised SIAM J. Sci. Comput.*
[15] Xia J L 2012 *High-Perform. Sci. Comput.* (Berlin: Springer) p. 199
[16] Duff I S and Reid J K 1983 *ACM Bans. Math. Software* **9** 302
[17] Liu J W H 1992 *SIAM Rev.* **34** 82
[18] Xia J L and Gu M 2010 *SIAM J. Matrix Anal. Appl.* **31** 2899
[19] Pei W B 2007 *J. Comm. Comput. Phys.* **2** 255
[20] An H B, Mo Z Y, Xu X W and Liu X 2009 *J. Comput. Phys.* **228** 3268
[21] Baldwin C, Brown P N, Falgout R D, Graziani F and Jones J 1999 *J. Comput. Phys.* **154** 1
[22] Fu S W, Fu H Q, Shen L J, Huang S K and Chen G N 1998 *Chin. J. Comput. Phys.* **15** 489 (in Chinese)
[23] Gu T X, Dai Z H, Hang X D, Fu S W and Liu X P 2005 *Chin. J. Comput. Phys.* **22** 471 (in Chinese)
[24] Mo Z Y, Fu S W and Shen L J 2000 *Chin. J. Comput. Phys.* **17** 625 (in Chinese)
[25] Mo Z Y, Shen L J and Wittum G 2004 *Int. J. Comp. Math.* **81** 361
[26] Zhang R P, Yu X J and Zhu J 2012 *Chin. Phys. Lett.* **29** 110201
[27] Zhao G Z, Yu X J and Zhang R P 2013 *Chin. Phys. B* **22** 020202
[28] Wu G C 2012 *Chin. Phys. B* **21** 120504
[29] METIS http://glaros.dtc.umn.edu/ gkhome/views/metis
[30] Parter S V 1961 *SIAM Rev.* **3** 119
[31] Tewarson R P 1967 *SIAM Rev.* **9** 91
[32] Schmitz P and Ying L 2012 http://www.ma.utexas.edu/users/lexing/publications/direct2d.pdf
[33] Mo Z Y, Zhang A Q, Cao X L, Liu Q K, Xu X W, An H B, Pei W B and Zhu S P 2010 *Front. Comp. Sci. China* **4** 480