Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 467672, 9 pages
http://dx.doi.org/10.1155/2014/467672

Research Article

MRILDU: An Improvement to ILUT Based on Incomplete LDU Factorization and Dropping in Multiple Rows

Jian-Ping Wu[1] and Huai-Fa Ma[2,3]

[1] *College of Computer, National University of Defense Technology, Changsha 410073, China*
[2] *State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydropower Research, Beijing 100038, China*
[3] *Earthquake Engineering Research Center, China Institute of Water Resources and Hydropower Research, Beijing 100048, China*

Correspondence should be addressed to Jian-Ping Wu; wjp@nudt.edu.cn

Received 30 May 2014; Accepted 6 August 2014; Published 20 August 2014

Academic Editor: D. R. Sahu

We provide an improvement MRILDU to ILUT for general sparse linear systems in the paper. The improvement is based on the consideration that relatively large elements should be kept down as much as possible. To do so, two schemes are used. Firstly, incomplete LDU factorization is used instead of incomplete LU. Besides, multiple rows are computed at a time, and then dropping is applied to these rows to extract the relatively large elements in magnitude. Incomplete LDU is not only fairer when there are large differences between the elements of factors L and U, but also more natural for the latter dropping in multiple rows. And the dropping in multiple rows is more profitable, for there may be large differences between elements in different rows in each factor. The provided MRILDU is comparable to ILUT in storage requirement and computational complexity. And the experiments for spare linear systems from UF Sparse Matrix Collection, inertial constrained fusion simulation, numerical weather prediction, and concrete sample simulation show that it is more effective than ILUT in most cases and is not as sensitive as ILUT to the parameter p, the maximum number of nonzeros allowed in each row of a factor.

1. Introduction

The solving of sparse equations is the core issue of many scientific and engineering calculations, and the time required to solve sparse linear systems generally accounts for a large proportion in the whole numerical simulation. More seriously, with the improvement of the simulation precision, the spatial resolution becomes higher and higher, the grid points become more and more dense, and thus, the size of the linear equations eventually also becomes larger and larger. For the large-scale sparse linear equations, especially for the sparse linear system during the simulation of the three-dimensional problems, the storage requirement and the amount of computation from the traditional direct solution methods are very large; besides, the sparsity of the coefficient matrix cannot be fully utilized [1] and it is difficult to be parallelized efficiently.

The iterative method can be controlled easily and the iterative method can make full use of the sparsity of the matrix; thus the storage requirement and the amount of computation are both very small in a single iteration. In addition, it can solve the corresponding linear equations only if the computational rule of coefficient matrix-vector product is known, not having to know the specific structure or the specific elements of the coefficient matrix, which is impossible for the direct method. Based on these considerations, the iterative method, especially the Krylov subspace method, has drawn more and more attention in recent years. Because the projection scheme is adopted in the Krylov subspace method, the convergence rate is relatively quicker. However, the convergence rate of the Krylov subspace method depends on the eigenvalue distribution of the coefficient matrix; the more concentrated the eigenvalues are, the quicker the convergence will be. To accelerate the convergence rate and reduce the total computation time, efficient preconditioning techniques should be used to convert the original sparse linear equations into another with the same solution, but with

narrower eigenvalue distribution region of the coefficient matrix [1, 2].

In general, the narrower the region the eigenvalues are located in, the more effectively the number of iterations can be reduced. However, the amount of computation in a single iteration must also be less so as to make the ultimate solution process efficient. On the other hand, when preconditioning is not used, the amount of computation in a single iteration lies mainly in the operation that the coefficient matrix is applied to a vector. Therefore, an efficient preconditioner should ensure that the distribution region of the eigenvalues is improved significantly, and at the same time, the additional amount of computation in a single iteration is almost equal to that of a matrix-vector product. The incomplete factorization is one kind of such preconditioners, which aims to solve the general sparse linear equations.

Many incomplete factorizations have been developed since Meijerink and van der Vorst introduced incomplete Cholesky factorization into the conjugate gradient as a preconditioner [3]. The differences among them are mainly in the dropping rules for the elements in the factors. Further, many block forms, diagonal modifications, stabilized versions, and parallel implementations have been developed.

The simplest incomplete factorization is ILU(0). It requires that the sparse structure of the incomplete factor is the same as the original coefficient matrix [3]. The implementation of ILU(0) can be very cheap, and when the coefficient matrix is diagonally dominant or a M matrix, it is effective and robust. However, for more complex issues encountered in actual applications, ILU(0) is too rough, and thus it inspires people to design more efficient ones under the way of allowing more fill-ins. ILU(k) is one of them; it can be seen as an extension of ILU(0) [1, 2, 4], in which the fill-ins with level number greater than k are set to zero. Generally speaking, ILU(1) is quite effective. When the level threshold k is higher, the improvement is usually very small. However, with the increase of k, the amount of computation increases very fast and thus ILU(k) with k greater than 1 is not considered in general.

When the diagonal dominance of the coefficient matrix is poor, ILU(k) may drop many relatively large elements,while retaining many elements with small absolute values, making the effectiveness of the method degraded greatly for general sparse matrices. It is due to the fact that ILU(k) completely starts from the nonzero structure, while the size of element in the factors is not considered. Based on this consideration, the researchers put forward another incomplete factorization in which the dropping is applied based on the magnitude of the nonzero elements in the factors. However, if the dropping rule is based only on the magnitude of elements, the total number of nonzero elements in the derived factors is very difficult to control; thus the storage requirement and the amount of computation are very difficult to control. The double threshold incomplete factorization (ILUT), put forward by Saad, effectively deals with the above-mentioned problem. When the factorization is performed at the kth step, the elements with relatively small absolute value in the kth row of the factor are dropped according to the threshold value σ firstly and then at most p elements with the largest absolute value are retained in the nondiagonal elements in each row of incomplete factors [5].

The LU factorization is not the only way to construct incomplete factorization preconditioner. Many others can also be used. Based on A-orthogonal factorization, Benzi computed a factorization type sparse approximate inverse preconditioner AINV, and the experimental results show that it has similar quality as that of ILU preconditioner but has better potential parallelism [6]. Aiming at the sparse linear equations needing to be solved in the numerical simulation of Markov chains, an incomplete WZ factorization is introduced in reference [7] and the experimental results show that this method is quicker than the ILU factorization preconditioner [7, 8]. For the sparse linear systems with finite element structure, Vannieuwenhoven and Meerbergen provided an incomplete multiple wave-frontal LU factorization preconditioning process, which is obtained through dropping in the process, and the computation performance and the robustness are improved through full use of the dense structure of the elemental stiffness matrix and the selection of local efficient principals [9].

Up to now, various incomplete factorizations are provided. However, ILUT is one of the most widely used incomplete factorizations. Although the effectiveness of the ILUT proposed for the general sparse linear equations is very high, there is still room for improvement. Recently, Maclachlan, Osei-kuffuor, and Saad studied the measures to improve its accuracy and stability through applying compensation [10]. This paper can also be seen as an improvement to ILUT. Having realized that the dropping rules in ILUT are not fair, an improvement is proposed in this paper from the following two aspects. First, the incomplete LDU factorization is used to replace the incomplete LU factorization, so as to make the dropping in incomplete factors L and U fairer. Second, when multiple rows computed at a time for incomplete factors L and U, the dropping rules are applied, respectively, to L and U. The elements with maximum magnitude are retained in the computed rows.

2. Description of MRILDU

The ILUT preconditioner has been widely and effectively used to solve many difficult sparse linear systems. The specific descriptions of the double threshold incomplete factorization ILUT(p, σ) are shown in Algorithm 1 [2].

In Algorithm 1, three dropping strategies are used. First, in the fifth row of the described algorithm, if the absolute value of some element is less than the threshold σ, the element will be dropped and replaced by zero. Second, in the tenth row, another rule is applied to drop the elements whose absolute values are less than σ_i, where σ_i is taken as σ/t_i and t_i is equal to the quotient of the 1-norm of the ith row in the original coefficient matrix to the number of the nonzero elements in the row. Third, in the eleventh and twelfth rows, for the ith row of L, namely, the strictly lower triangular part, at most p largest elements are retained, and at the same time, for the strictly upper triangular part in the row of U, at most p largest elements are retained too. In addition, the diagonal elements are always retained.

(1) For $i = 1, \ldots, n$ Do
(2) $w := a_{i^*}$
(3) For $k = 1, \ldots, i-1$ and when $w_k \neq 0$ Do
(4) $w_k := w_k / u_{kk}$
(5) If $|w_k| < \sigma$ then $w_k := 0$
(6) If $w_k \neq 0$ then
(7) $w := w - w_k * u_{k^*}$
(8) Endif
(9) Enddo
(10) If $|w_j| < \sigma_i$ then $w_j := 0$ for $j = i+1, \ldots, n$
(11) Extract the maximum p elements from $l_{i,1:i-1}$
(12) Extract the maximum p elements from $u_{i,i+1:n}$
(13) $l_{ij} := w_j$ for $j = 1, \ldots, i-1$
(14) $u_{ij} := w_j$ for $j = i, \ldots, n$
(15) $w := 0$
(16) Enddo

Algorithm 1: Algorithm ILUT(p, σ).

(1) For $i = 1, \ldots, n$ Do
(2) $w := a_{i^*}$
(3) For $k = 1, \ldots, i-1$ and when $w_k \neq 0$ Do
(4) $\alpha := w_k$ and $w_k := \alpha / d_k$
(5) If $|w_k| < \sigma$ then $w_k := 0$
(6) If $w_k \neq 0$ then
(7) $w := w - \alpha * u_{k^*}$
(8) Endif
(9) Enddo
(10) For $j = i+1, \ldots, n$ Do
(11) $w_j := w_j / w_i$
(12) If $|w_j| < \sigma$ then $w_j := 0$
(13) Endfor
(14) $l_{ij} := w_j$ for $j = 1, \ldots, i$–1
(15) $d_i := w_i$
(16) $u_{ij} := w_j$ for $j = i+1, \ldots, n$
(17) $w := 0$
(18) If mod$(i, b) = 0$ then
(19) Extract the maximum bp elements from $l_{i-b+1:i,*}$
(20) Extract the maximum bp elements from $u_{i-b+1:i,*}$
(21) Endif
(22) Enddo

Algorithm 2: Algorithm MRILDU(b, p, σ).

The effectiveness of ILUT preconditioner originates from the following two aspects. First, the number of the nonzero elements in the incomplete factors is reduced through dropping the elements whose relative magnitude is less than σ and retaining at most p nonzero elements in each row of the factors. This reduces not only the time used in incomplete factorization, but also the preconditioned iteration time greatly. Second, when the diagonal dominance of the matrix is poor, the quality of the incomplete factors can be improved through increasing the parameter p and decreasing the parameter σ, especially if σ is equal to zero and p is equal to the order of the matrix and the related incomplete factorization is just the LU factorization.

Although ILUT is effective, it still has deficiencies in practical applications. First, it is very difficult to specify the best values for the parameters in ILUT. If p is too small or σ is too large, the effectiveness of the preconditioner will be very poor. However, when p is very large and σ is very small, although the effectiveness of the obtained preconditioner is good, the cost of the construction of the preconditioner and the overhead in a single iteration are both very large, which is unbearable. Second, when the magnitudes of the elements in each row of the matrix differ greatly, the magnitudes of the elements in each row of the incomplete factors L and U may also differ greatly. However, ILUT almost averagely retains the same number of nonzero elements in each row and this may drop some relatively large elements while retaining many relatively small elements. Although the scheme to retain relatively large elements is not always superior to that to retain relatively small elements, the existing experiments show that it has more advantages to retain relatively large elements in general, especially for diagonally dominant matrices.

Based on the above considerations, here we propose an improvement to ILUT, which is multirow ILDU (MRILDU). The idea can be outlined as follows. Perform incomplete LDU factorization for the matrix, compute multiple rows of factors L and U every time, and then apply dropping strategy to the obtained rows of L and U. It has the following two advantages to use the incomplete LDU factorization to replace the incomplete LU factorization. First, when LDU factorization is applied, the elements in L and the elements in U have been, respectively, proportioned to the diagonal elements in the same column or same row and thus it is more equitable to use the same dropping strategy in L and U. Second, because the elements in L have already been scaled, it is more equitable for each row when the dropping strategy is used for multiple rows. Furthermore, when multiple rows are computed at a time and then dropping rule is applied, the unified rule can be used to drop nonzero elements in these rows. Compared to the algorithm that one row is computed at a time and simultaneously the dropping strategy is applied, MRILDU is more favorable in retaining the elements with relatively large magnitudes in the incomplete factors. Assuming that every time b rows of the matrix are factorized and then the dropping strategy is used for them, MRILDU(b, p, σ) can be specifically described in Algorithm 2.

During the implementation, the algorithm MRILDU (b, p, σ) needs to efficiently solve the following three problems. The first one is the linear combination of the sparse vectors on step 7. The second one is to select several elements with the largest magnitude from the given vectors on step 19 and step 20, and the third one is that the elements in ith row of L must be accessed in ascending order on step 3. It can be found from the comparison of the steps in Algorithms 2 and 1 that the first problem encountered in the algorithm MRILDU(b, p, σ) is the same as that of ILUT(p, σ). Therefore, the same technology can be used. See literatures [2, 4] for specific implementation details.

For the second problem, we adopt the quick sort method which is slightly different from that in ILUT(p, σ). When the sorting is applied to step 19 and step 20 in MRILDU(b, p, σ), the exchange operation is not applied in deed, while another

integer array is used to track the sorting process, and ultimately its first bp elements are used to record the positions of the elements with the largest magnitude. After the end of the sorting, the elements on other positions in the column number array are set to zero, and those elements are dropped accordingly. In this way, the remaining elements can be preserved directly without having to spend too much time to determine the row numbers of the nonzero elements.

For the third problem, from its appearance, it should be completely identical in MRILDU and in ILUT. However, it needs to note that, in MRILDU, b rows are taken as a whole at a time to use the dropping strategy. We may as well assume that the dropping strategy is used to the rows from the kth to the $k + b - 1$th altogether, when the kth row is computed, similar to ILUT; the elements in the final incomplete factor U are used. However, when the rows from $k + 1$th to $k + b - 1$th are computed, the elements in the $k - 1$th row as well as the rows in front of it in the factor U are the elements in the eventual incomplete factor. But when the elements in the rows from the kth to $k + b - 2$th are used, these elements are only temporary values, and the drooping strategy has not been applied yet. In order to facilitate the calculation, initially, bn additional storage units are allocated in advance for each incomplete factor and in the calculation the elements of continuous b rows are stored according to the format of the $k-1$th and the previous rows to facilitate the implementation of the algorithm. When the dropping strategy is applied, the elements in these b rows are processed directly. The dropped elements are set to zero, and after that, these zero elements are dropped completely and their positions are filled with the subsequent nonzero elements.

In the following, we analyzed the differences of the storage requirement and the amount of computation of MRILDU from ILUT. In ILUT, every time one row is computed and p nonzero elements with largest magnitudes are selected and retained from the current row of each triangular factor. There may be only at most n nonzero elements in each row of the factors before applying the dropping rule. Therefore, it can be considered that the temporary space complexity is $O(n)$, which needs to be allocated additionally. In MRILDU, the nonzero elements in b rows need to be stored before applying dropping rule, and thus, the temporary space complexity which needs to be allocated additionally may be up to $O(bn)$. As previously described, the storage space has been allocated in advance. For the final incomplete factors, due to the fact that the storage units of MRILDU(b, p, σ) and ILUT(p, σ) are both $(2p + 1)n$ at most, therefore, the storage requirements are almost invariant, especially when p is small or σ is large. Accordingly in the preconditioned iteration, the computing time in a single iteration will also be almost the same. On the other hand, the quality of MRILDU is higher in general; that is to say, when MRILDU is used, the time spent in iteration will be less than that when ILUT is used. Therefore, if the construction of the preconditioner is not considered, the time elapsed in the iteration itself will be less with MRILDU than that with ILUT.

The time complexity of the construction of MRILDU differs ILUT mainly from two origins. First, in ILUT, the dropping is applied for one row each time. After the dropping is completed, the retained elements of the row are stored in the incomplete factor. When the calculation is applied for the subsequent rows, the elements in front of the current row in the factor U need to be used and these elements are in the final incomplete factor. In MRILDU, b rows of the factors are calculated at a time. It may be assumed that they are the rows from the kth to $k + b - 1$th. When applying dropping to these b rows as a whole, the calculation process is as described previously. Therefore, the additional temporary storage units are greater than that in ILUT in general. At the same time, the amount of computation will be also slightly larger than that with ILUT.

The second origin is the different dimension of the vectors to be split and the different number of elements which need to be extracted from them. For ILUT, if there are q nonzeros in each of the rows from the kth to the $k + b - 1$th before dropping is applied, the time complexity for each row is about $O(q/p \log q)$ when the quick sort is used to extract the required p elements from the factor. Therefore, the time complexity for b rows is $O(bq/p \log q)$. For MRILDU, because the dropping strategy is applied to bp rows, this operation is equivalent to the extraction of bp elements from bq elements. Thus, the time complexity should be $O\{bq/(bp) \log(bq)\}$, namely, $O\{q/p \log(bq)\}$. This shows that just from the perspective of extracting the elements with largest magnitudes, it seems that MRILDU will be quicker. However, it needs to note that, when the rows from the $k+1$th to the $k+b-1$th are computed, the temporarily storage units in each row are likely more than those in ILUT. Therefore, even if MRILDU will be slightly quicker, it is not very significant, especially when b is large.

Furthermore, it can be proved that similar to other ILU factorizations, when A is a M matrix or diagonally dominant, MRILDU can always continue, and the lower right corner submatrix to be factorized will also be always a M matrix or diagonally dominant. For each of these two issues, after A is split into LDU and A − LDU, the iterative method with LDU as the iterative matrix will be always convergent. Correspondingly, the condition number of the preconditioned coefficient matrix will be better than that of the original coefficient matrix.

3. Numerical Experiments

In this section, for several sparse linear equations from some scientific engineering applications, the proposed MRILDU is compared to ILUT. The experiments are applied on the high performance server with Intel Xeon CPU E5-2670 0 @ 2.60 GHz CPU, 20480 KB cache, and main memory of 48 G. The operating system is Red Hat 4.4.5-6 Linux. The Intel Fortran compiler ifort 11.1.059 is used as compiler and -O3 is used for optimization.

In this paper, for convenience, the number of iterations will be denoted by IT, and the time used for the construction of the preconditioner and the iteration process are denoted by CTME and ITME, respectively. The total solution time is denoted by TTME, which is the sum of CTME and ITME. NNZ represents the number of nonzero elements in the

Table 1: Dimension (N) and number of nonzero (NNZ) test matrices.

Matrix	N	NNZ	Application field
af_shell10	1508065	27090195	Sheet metal forming
bone010	986703	36326514	Model reduction
ldoor	952203	42493817	Structural problem
thermal2	1228045	4904179	Thermal problem
airfoil_2d	14214	259688	CFD problem
cvxbqp1	50000	349968	Optimization problem
sparsine	50000	1548988	Structural problem
orsirr_1	1030	6858	CFD problem
pores_2	1224	9613	CFD problem
saylr3	1000	3750	CFD problem
sherman3	5005	20033	CFD problem

Table 2: Iteration results for matrix af_shell10.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		7	6813992	0.2050	1.2517	1.4566
MRILDU	5	1	6	8311004	0.2420	1.1074	1.3494
MRILDU	5	2	5	8529756	0.2464	0.9494	1.1958
MRILDU	5	5	5	8313907	0.2228	0.9362	1.1590
ILUT	10		6	9848598	0.2054	1.1158	1.3213
MRILDU	10	1	4	11282919	0.2162	0.8019	1.0181
MRILDU	10	2	3	11369132	0.2150	0.6045	0.8196
MRILDU	10	5	4	11579261	0.1928	0.7934	0.9863
ILUT	20		6	10464841	0.2027	1.1415	1.3442
MRILDU	20	1	2	11746902	0.1872	0.4282	0.6154
MRILDU	20	2	2	11749200	0.1876	0.4273	0.6149
MRILDU	20	5	2	11749244	0.1855	0.4318	0.6173

Table 3: Iteration results for matrix bone010.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		16	4900335	0.3119	2.5586	2.8704
MRILDU	5	1	12	5884440	0.4067	1.9592	2.3659
MRILDU	5	2	12	5894775	0.4346	1.9649	2.3995
MRILDU	5	5	11	5915403	0.4694	1.8480	2.3174
ILUT	10		9	9702708	0.3530	1.6085	1.9615
MRILDU	10	1	8	10686813	0.4344	1.4531	1.8875
MRILDU	10	2	8	10722972	0.4445	1.4701	1.9146
MRILDU	10	5	7	10795936	0.4915	1.2933	1.7848
ILUT	20		5	18784893	0.3951	1.0752	1.4703
MRILDU	20	1	5	19768998	0.4477	1.1030	1.5507
MRILDU	20	2	4	19817915	0.4725	0.8938	1.3663
MRILDU	20	5	4	19923484	0.5748	0.9048	1.4797

preconditioner. All the time results are in seconds. In all iterations, the initial solution vector is selected as all zeros.

3.1. Sparse Linear Systems from Some Test Matrices. The coefficient matrices of the sparse linear equations here come from the UF sparse matrix collection (http://www.cise.ufl.edu/research/sparse/matrices/index.html) website, and the information of these matrices can be briefly described in Table 1. Among them, the last four matrices starting from orsirr_1 can also be downloaded from the website Martrix-market (http://math.nist.gov/MatrixMarket/browse.html). The right-hand side vectors are obtained through applying the matrix to the given true solution vector, in which the ith component is $x_i = i/n$, where n is the order of the matrix. During the iteration process, BiCGSTAB is used, and the convergence criterion is that the 2-norm of the residual vector is reduced by 10 orders of magnitude.

The results with ILUT and MRILDU are listed in Table 2. In all the tests of this subsection, the threshold value σ is taken as $1E-3$ all the time.

As can be seen from Tables 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12, when the same p and σ are used, the number of iterations with MRILDU is less than that with ILUT in general; namely, the effectiveness of MRILDU is higher. Besides, when p is small, due to the fact that the number of iterations with MRILDU is less than that with ILUT, the time elapsed for iteration is also less than that with ILUT. However, for some matrices, because the number of the nonzero elements retained by MRILDU is relatively more, while the number of iterations is not reduced significantly, especially when p is large, the time elapsed for iteration is slightly longer than that with ILUT. From the perspective of total execution time including preconditioner construction and iteration process, although it is difficult to say that MRILDU is absolutely better than ILUT, MRILDU is superior in general. Furthermore, when ILUT is used, it is difficult to solve the linear systems for some matrices. However, when MRILDU is used, the solution is more robust with slightly larger b. Moreover, the experiments also show that when b is large, MRILDU is not as sensitive as ILUT to the parameter p.

3.2. Energy Equation in Inertial Constrained Fusion Simulation. The energy equations in the numerical simulation of inertial confinement fusion can be described as in the literatures [11, 12]. When the discretization is applied to the equations, the velocity and coordinates are given on grid points, while the temperature, density, pressure, and energy are given at the center of the cells. When uniform quadrilateral mesh is used to discretize the continuous equations, we

Table 4: Iteration results for matrix ldoor.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		19	4165454	0.3516	2.0167	2.3683
MRILDU	5	1	17	5030085	0.4738	1.8135	2.2873
MRILDU	5	2	15	5220618	0.4855	1.6805	2.1660
MRILDU	5	5	13	5360138	0.4806	1.5297	2.0103
ILUT	10		10	7533577	0.3755	1.1645	1.5400
MRILDU	10	1	9	8365112	0.4880	1.0859	1.5740
MRILDU	10	2	7	8683459	0.4947	0.8785	1.3732
MRILDU	10	5	7	8979981	0.4938	0.9068	1.4007
ILUT	20		5	12443336	0.3878	0.6864	1.0742
MRILDU	20	1	5	13180555	0.4608	0.7038	1.1647
MRILDU	20	2	5	13565302	0.4595	0.7209	1.1804
MRILDU	20	5	5	13932275	0.4477	0.7354	1.1830

Table 5: Iteration results for matrix thermal2.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		6	3589294	0.0808	0.6483	0.7291
MRILDU	5	1	3	4701667	0.0791	0.3485	0.4276
MRILDU	5	2	3	4836374	0.0781	0.3535	0.4316
MRILDU	5	5	3	4881973	0.0766	0.3549	0.4315
ILUT	10		6	3785538	0.0799	0.6612	0.7411
MRILDU	10	1	2	4896078	0.0680	0.2475	0.3154
MRILDU	10	2	2	4896078	0.0717	0.2476	0.3193
MRILDU	10	5	2	4896078	0.0706	0.2474	0.3180
ILUT	20		6	3785538	0.0799	0.6616	0.7416
MRILDU	20	1	2	4896078	0.0687	0.2477	0.3164
MRILDU	20	2	2	4896078	0.0723	0.2474	0.3198
MRILDU	20	5	2	4896078	0.0705	0.2474	0.3180

Table 6: Iteration results for matrix airfoil_2d.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		45	140933	0.0119	0.0583	0.0702
MRILDU	5	1	38	155280	0.0178	0.0486	0.0664
MRILDU	5	2	41	155863	0.0188	0.0563	0.0751
MRILDU	5	5	38	155992	0.0188	0.0561	0.0750
ILUT	10		22	276559	0.0286	0.0354	0.0640
MRILDU	10	1	21	289850	0.0326	0.0327	0.0653
MRILDU	10	2	22	294696	0.0334	0.0372	0.0706
MRILDU	10	5	21	297000	0.0335	0.0382	0.0718
ILUT	20		13	491044	0.0509	0.0291	0.0800
MRILDU	20	1	14	455303	0.0415	0.0302	0.0717
MRILDU	20	2	14	457858	0.0399	0.0306	0.0706
MRILDU	20	5	14	458851	0.0397	0.0306	0.0703

Table 7: Iteration results for matrix cvxbqp1.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		9	144410	0.0021	0.0206	0.0227
MRILDU	5	1	3	188828	0.0023	0.0077	0.0100
MRILDU	5	2	2	197127	0.0022	0.0055	0.0078
MRILDU	5	5	2	198783	0.0023	0.0055	0.0077
ILUT	10		9	155499	0.0020	0.0207	0.0227
MRILDU	10	1	1	199884	0.0021	0.0032	0.0053
MRILDU	10	2	1	199884	0.0021	0.0032	0.0053
MRILDU	10	5	1	199884	0.0022	0.0032	0.0054
ILUT	20		9	155499	0.0020	0.0207	0.0227
MRILDU	20	1	1	199884	0.0021	0.0032	0.0053
MRILDU	20	2	1	199884	0.0021	0.0032	0.0053
MRILDU	20	5	1	199884	0.0022	0.0032	0.0054

Table 8: Iteration results for matrix sparsine.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		16	229231	0.0121	0.0741	0.0861
MRILDU	5	1	14	278049	0.0168	0.0658	0.0826
MRILDU	5	2	12	283896	0.0175	0.0593	0.0768
MRILDU	5	5	12	286417	0.0172	0.0611	0.0782
ILUT	10		13	410600	0.0130	0.0684	0.0814
MRILDU	10	1	9	459413	0.0169	0.0493	0.0662
MRILDU	10	2	9	469141	0.0175	0.0509	0.0685
MRILDU	10	5	9	473756	0.0165	0.0511	0.0677
ILUT	20		10	631734	0.0127	0.0712	0.0839
MRILDU	20	1	6	680387	0.0123	0.0456	0.0580
MRILDU	20	2	4	709044	0.0125	0.0324	0.0449
MRILDU	20	5	4	723248	0.0125	0.0322	0.0447

Table 9: Iteration results for matrix orsirr_1.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		167	5333	0.0002	0.0094	0.0097
MRILDU	5	1	10	6712	0.0004	0.0008	0.0011
MRILDU	5	2	11	6782	0.0004	0.0008	0.0012
MRILDU	5	5	10	6932	0.0004	0.0008	0.0012
ILUT	10		165	5678	0.0003	0.0095	0.0098
MRILDU	10	1	10	7517	0.0004	0.0008	0.0012
MRILDU	10	2	11	7698	0.0004	0.0009	0.0013
MRILDU	10	5	10	7931	0.0004	0.0008	0.0012
ILUT	20		165	5961	0.0003	0.0097	0.0100
MRILDU	20	1	10	7839	0.0004	0.0008	0.0012
MRILDU	20	2	9	8100	0.0004	0.0007	0.0011
MRILDU	20	5	11	8183	0.0004	0.0009	0.0013

can derive the discrete nonlinear equations. When Newton iteration is used to solve the nonlinear system and natural ordering is used to grid points, the problem is converted into the solution of block tridiagonal linear equations which corresponds to Jacobi matrices [12].

In this subsection, we perform experiments for two typical sparse matrices (namely, MAT17 and MAT20) extracted from the discrete solution of the two-dimensional three temperature energy equations. The size of the related discrete grid is 21 by 75. Since each grid point corresponds to three temperature variables, namely, electron, ion, and photon, the number of the dimensions of the equations is 4,725. In the experiments, the stop criterion is that the Euclid norm of the residual vector is reduced by 10 orders of magnitude.

Table 10: Iteration results for matrix pores_2.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		95	7339	0.0003	0.0071	0.0074
MRILDU	5	1	56	9587	0.0004	0.0045	0.0049
MRILDU	5	2	48	11435	0.0006	0.0041	0.0048
MRILDU	5	5	30	11315	0.0007	0.0028	0.0035
ILUT	10		58	9778	0.0005	0.0050	0.0055
MRILDU	10	1	28	14520	0.0008	0.0027	0.0035
MRILDU	10	2	24	17193	0.0011	0.0026	0.0037
MRILDU	10	5	20	17206	0.0012	0.0022	0.0034
ILUT	20		59	10002	0.0005	0.0050	0.0055
MRILDU	20	1	19	19384	0.0011	0.0022	0.0033
MRILDU	20	2	17	21137	0.0012	0.0021	0.0033
MRILDU	20	5	17	21307	0.0013	0.0021	0.0034

Table 11: Iteration results for matrix saylr3.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		45	4940	0.0004	0.0028	0.0032
MRILDU	5	1	9	6104	0.0006	0.0006	0.0013
MRILDU	5	2	7	7602	0.0008	0.0006	0.0014
MRILDU	5	5	7	8560	0.0009	0.0007	0.0015
ILUT	10		43	7455	0.0006	0.0030	0.0037
MRILDU	10	1	6	9374	0.0010	0.0006	0.0015
MRILDU	10	2	5	11002	0.0010	0.0005	0.0016
MRILDU	10	5	5	11828	0.0010	0.0006	0.0016
ILUT	20		47	8968	0.0007	0.0036	0.0043
MRILDU	20	1	5	12024	0.0010	0.0006	0.0016
MRILDU	20	2	5	12430	0.0010	0.0006	0.0016
MRILDU	20	5	5	12466	0.0010	0.0006	0.0016

Table 12: Iteration results for matrix sherman3.

	p	b	IT	NNZ	CTME	ITME	TTME
ILUT	5		137	28459	0.0021	0.0365	0.0386
MRILDU	5	1	44	32097	0.0029	0.0121	0.0150
MRILDU	5	2	44	33180	0.0033	0.0124	0.0158
MRILDU	5	5	30	35526	0.0038	0.0092	0.0130
ILUT	10		128	50481	0.0051	0.0404	0.0456
MRILDU	10	1	28	55298	0.0060	0.0089	0.0150
MRILDU	10	2	27	56084	0.0064	0.0092	0.0156
MRILDU	10	5	23	58137	0.0066	0.0084	0.0150
ILUT	20		113	85790	0.0091	0.0462	0.0553
MRILDU	20	1	18	88174	0.0096	0.0073	0.0169
MRILDU	20	2	17	88770	0.0098	0.0072	0.0170
MRILDU	20	5	18	90454	0.0100	0.0080	0.0179

For matrices MAT17 and MAT20, the results are listed in Tables 13 and 14, respectively. In these experiments, the parameter σ is taken as $1E-3$.

As can be seen from Tables 13 and 14, although MRILDU with $b = 1$ corresponds to ILUT, similar to the previous experiments, the experimental results also show that

Table 13: Iteration results for MAT17.

	p	b	IT	CTME	ITME	TTME
ILUT	5		22	0.0005	0.0051	0.0057
MRILDU	5	1	12	0.0007	0.0028	0.0036
MRILDU	5	5	8	0.0012	0.0021	0.0033
MRILDU	5	10	7	0.0012	0.0019	0.0030
ILUT	10		17	0.0009	0.0043	0.0052
MRILDU	10	1	9	0.0011	0.0023	0.0034
MRILDU	10	5	6	0.0013	0.0016	0.0029
MRILDU	10	10	6	0.0012	0.0016	0.0028

Table 14: Iteration results for MAT20.

	p	b	IT	CTME	ITME	TTME
ILUT	5		17	0.0007	0.0042	0.0049
MRILDU	5	1	12	0.0008	0.0029	0.0037
MRILDU	5	5	7	0.0014	0.0020	0.0034
MRILDU	5	10	7	0.0013	0.0020	0.0034
ILUT	10		14	0.0012	0.0038	0.0050
MRILDU	10	1	8	0.0012	0.0022	0.0034
MRILDU	10	5	7	0.0014	0.0020	0.0034
MRILDU	10	10	7	0.0014	0.0020	0.0034

MRILDU with $b = 1$ has great advantages and both the number of iterations and the time used in iteration are reduced significantly. Furthermore, when the parameter p is given and the parameter b increases, the convergence rate of the iteration is improved gradually; however, when the parameter b is increased to a certain extent, further improvement is trivial when it continues to increase. Moreover, it can be noted that after the parameter b is increased to a certain extent, the quality of MRILDU will be very close for different parameters p.

3.3. Helmholtz Equation in Numerical Weather Prediction Model. When the finite-difference model is used to perform the numerical weather forecast, the sparse linear equations related to the pressure deviation in the three-dimensional space need to be solved on each time step, which are called discrete Helmholtz equations [13]. In this paper, we perform experiments for the grid point model GRAPES [14] developed by Chinese Academy of Meteorological Sciences. In this model, the full compressible atmosphere motion system is used as the control equation, the semi-implicit semi-Lagrangian scheme is used as discrete scheme for time derivative, and the discretization adopts Arakawa-C grid in horizontal direction and Charney-Phillips grid in vertical direction. See literature [14] for details.

Although the Helmholtz equations are diagonally dominant, the diagonal dominance is very weak. For this kind of sparse linear equations, the GCR iteration is often used to solve the equations [2], but the solution is very time-consuming, which occupies a large proportion in the whole numerical simulation time. For example, in GRAPES, when the diagonal scaling GCR is used and the grid size is $144 \times 73 \times 31$, namely, the number of the grid points in the latitude,

Table 15: Iteration results for linear systems from GRAPES.

	p	b	Avg. #iters	Avg. time
ILUT	5		104.5833	4.6204
MRILDU	5	1	42.1771	2.5964
MRILDU	5	5	39.8021	2.5092
MRILDU	5	10	39.3854	2.4978
ILUT	10		54.8333	3.3918
MRILDU	10	1	26.2604	2.2461
MRILDU	10	5	25.8125	2.2306
MRILDU	10	10	25.6354	2.2229

longitude, and vertical direction is 144, 73 and 31, respectively, this proportion can be up to about 70%.

In the experiments, the grid size used is 144 by 73 by 31, the preconditioned GCR is used to perform the iterations, and the stop criterion is that the 2-norm of the residual vector is less than $1E-8$. Due to the fact that the coefficient matrix of the sparse linear equations remains unchanged in the simulation process, the preconditioner just needs to be constructed in the setting stage and its information can be directly referenced in the solution processes of the sparse linear equations at subsequent time steps. During the simulation, the length of time step is taken as 1,800 seconds and the experiment is integrated for one day, namely, 96 time steps. Thus, there are 96 linear systems to be solved in all.

The experimental results can be described as in Table 15, where the average number of iterations and the average iteration time for the solution of a linear system with preconditioned GCR are provided. The parameter σ is taken as $1E-6$ in all tests.

As can be seen from Table 15, MRILDU is significantly superior to ILUT, and this is mainly thanks to the facts that the incomplete ILDU factorization is used to replace the ILU factorization and the dropping strategy for the elements in U in MRILDU is different from that in ILUT. At the same time, it benefits partly from the multirow strategy.

3.4. Mesoscale Simulation of Concrete Sample. In the mesoscale simulation of concrete specimens, the specimens are seen as three-phase composite materials which are composed of aggregate, mortar, and the interface between them and the finite element method is used. In this paper, we perform experiments for a cubic wet-sieved specimen and the size of the specimen is 550 mm by 150 mm by 150 mm. See literature [8] for specific descriptions. There are 71,013 discrete nodes and 78,800 finite elements in all, respectively. The two supporting columns are, respectively, located at the places where $x = -0.15$ m and $z = 0$ m as well as $x = 0.3$ m and $z = 0$ m, namely, the places which are, respectively, 0.05 m away from the left and right boundaries at the bottom of the specimen, while the two loading columns are located at the places where $x = 0$ m and $z = 0.15$ m as well as $x = 0.15$ m and $z = 0.15$ m, respectively, namely, the places which are, respectively, 0.2 m away from the left and right boundaries at the top of the specimen.

Table 16: Iteration results for static loading of concrete sample.

	b	Avg. #iters	Avg. time
ILUT		142.6517	13.7294
MRILDU	1	144.1124	12.9234
MRILDU	10	143.1742	12.2766
MRILDU	20	142.6517	10.9980

In this subsection, for the linear equations encountered in the static loading test, we compare the efficiency of ILUT and MRILDU. The load is increased step by step, and the increased load for each step is 0.25 kN. When loaded to the 59th step, the damaged elements will appear in the concrete specimen and the specimen is completely damaged at the 94th step. If some damaged element appears at some loading step, it may need to solve multiple sparse linear equations with the same coefficient matrix at this step, to correct the displacements. Therefore, there are 178 sparse linear systems to be solved in all during the whole simulation process. Although the obtained sparse linear systems are symmetric positive definite, due to the fact that ILUT and MRILDU are not symmetric, BiCGSTAB is selected as the iterative method. The stop criterion is that the 2-norm of the residual vector is reduced by six orders of magnitude.

In Table 16, for the sparse linear systems occurring in the static loading simulation, the average number of iterations and the average time elapsed for iteration of each sparse linear system are provided. In the tests, the parameter p is taken as 25 and the parameter σ is taken as $1E-3$. Due to the fact that when there are no damaged elements, the global stiffness matrix, namely, the coefficient matrix of sparse linear equations, remains unchanged and even if there are damaged elements, the coefficient matrix is also unchanged during the corrections at each loading step. As long as the coefficient matrix is unchanged, the preconditioner does not need to reconstruct. Therefore, the time consumed by the construction of the preconditioner is relatively trivial and thus the time used to construct the preconditioner is not listed in the table, while only the iteration results are listed.

As can be seen from Table 16, for sparse linear equations which need to be solved in the mesoscale numerical simulation of concrete specimen, MRILDU does not have advantages in the number of iterations compared to ILUT, but the average iteration time used for each linear system is improved slightly. This may be due to the fact that the utilization ratio of Cache is better when MRILDU is used.

4. Conclusion

In this paper, aiming at the disadvantage that ILUT may drop some relatively large elements and retain relatively small elements, we propose an improved version MRILDU based on the following two techniques. First, the incomplete LDU factorization is used to replace the incomplete LU factorization and the same dropping strategy is used for the elements in L and U, so as to make the dropping rules be more equitable for L and U. Second, multiple rows are factorized

before each dropping, so as to effectively deal with great differences between the diagonal dominance and to retain the relatively large elements, thereby improving the quality of incomplete factorization. The experimental results show that when the same parameters are used, the number of iterations with MRILDU will be significantly smaller than that with ILUT. And in most cases, the iteration time and total solution time with MRILDU can be reduced in general. Furthermore, MRILDU is not as sensitive as ILUT is to the parameter p, and thus, in the case that the parameter p cannot effectively be determined, the advantages of MRILDU are more significant.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This project is supported by the National Natural Science Foundation of China under Grants nos. 61379022 and 51079164, China Water Conservation Special no. 201201053, and Research Special of China Institute of Water Resources and Hydropower Research no. KJ1242.

References

[1] M. Benzi, "Preconditioning techniques for large linear systems: a survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[2] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, Mass, USA, 1996.

[3] J. A. Meijerink and H. A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Mathematics of Computation*, vol. 31, no. 137, pp. 148–162, 1977.

[4] I. Gustafsson, "A class of first order factorization methods," *BIT*, vol. 18, no. 2, pp. 142–156, 1978.

[5] Y. Saad, "ILUT: a dual threshold incomplete *LU* factorization," *Numerical Linear Algebra with Applications*, vol. 1, no. 4, pp. 387–402, 1994.

[6] M. Benzi and M. Tüma, "A robust incomplete factorization preconditioner for positive definite matrices," *Numerical Linear Algebra with Applications*, vol. 99, pp. 1–20, 2001.

[7] B. Bylina and J. Bylina, "Incomplete WZ factorization as an alternative method of preconditioning for solving Markov chains," in *Parallel Processing and Applied Mathematics: 7th International Conference, PPAM 2007, Gdansk, Poland, September 9–12, 2007*, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds., vol. 4967 of *Lecture Notes in Computer Science*, pp. 99–107, Springer, Berlin, Germany, 2008.

[8] B. Bylina and J. Bylina, "The incomplete factorization preconditioners applied to the GMRES(m) method for solving Markov chains," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '11)*, pp. 423–430, September 2011.

[9] N. Vannieuwenhoven and K. Meerbergen, "IMF: an incomplete multifrontal *LU*-factorization for element-structured sparse linear systems," *SIAM Journal on Scientific Computing*, vol. 35, no. 1, pp. A270–A293, 2013.

[10] S. MacLachlan, D. Osei-Kuffuor, and Y. Saad, "Modification and compensation strategies for threshold-based incomplete factorizations," *SIAM Journal on Scientific Computing*, vol. 34, no. 1, pp. A48–A75, 2012.

[11] S. Weber, P.-H. Maire, R. Loubère et al., "A transport simulation code for inertial confinement fusion relevant laser-plasma interaction," *Computer Physics Communications*, vol. 168, no. 3, pp. 141–158, 2005.

[12] T. X. Gu, Z. H. Dai, X. D. Hang, S. Fu, and X. Liu, "Efficient algebraic methods for two-dimensional energy equations with three temperatures," *Chinese Journal of Computational Physics*, vol. 22, no. 6, pp. 471–483, 2005.

[13] L. Li, W. Xue, R. Ranjan, and Z. Jin, "A scalable Helmholtz solver in GRAPES over large-scale multicore cluster," *Concurrency Computation Practice and Experience*, vol. 25, no. 12, pp. 1722–1737, 2013.

[14] D. H. Chen, J. S. Xue, X. S. Yang et al., "New generation of multi-scale NWP system (GRAPES): general scientific design," *Chinese Science Bulletin*, vol. 53, no. 22, pp. 3433–3445, 2008.