BALANCED INCOMPLETE FACTORIZATION *

RAFAEL BRU, JOSÉ MARÍN, JOSÉ MAS † AND M. TŮMA‡

Abstract. In this paper we present a new incomplete factorization of a square matrix into triangular factors in which we get standard LU or LDL^T factors (direct factors) and their inverses (inverse factors) at the same time. Algorithmically, we derive this method from the approach based on the Sherman-Morrison formula [18]. In contrast to the RIF algorithm [11], the direct and inverse factors here directly influence each other throughout the computation. Consequently, the algorithm to compute the approximate factors may mutually balance dropping in the factors and control their conditioning in this way. For the symmetric positive definite case, we derive the theory and present an algorithm for computing the incomplete LDL^T factorization, and discuss experimental results. We call this new approximate LDL^T factorization the Balanced Incomplete Factorization (BIF). Our experimental results confirm that this factorization is very robust and may be useful in solving difficult ill-conditioned problems by preconditioned iterative methods. Moreover, the internal coupling of computation of direct and inverse factors results in much shorter setup times (times to compute approximate decomposition) than RIF, a method of a similar and very high level of robustness. We also derive and present the theory for the general nonsymmetric case, but do not discuss its implementation.

Key words. preconditioned iterative methods, sparse matrices, incomplete decompositions, approximate inverses

1. Introduction.

We consider the linear system

$$Ax = b, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a large, sparse, regular and generally nonsymmetric matrix. One of the intensively studied problems in scientific computing is the development of efficient preconditioners for solving (1.1) by preconditioned iterative methods.

Incomplete factorizations represent a class of algebraic preconditioners which is important from both theoretical and practical points of view. Their development started by the work of Buleev at the end of fifties [19], [20]. Throughout the time, the algorithms have achieved a considerable degree of efficiency and robustness. The first incomplete factorizations were tightly connected to particular discretizations of partial differential equations by finite differences and to special matrices [49], see also [22] and [3]. An increasing amount of attention led to nice theoretical results for simple model problems [28], [43], [31]. Solving more complicated problems and enhancing the convergence of the preconditioned iterative method requires tools to improve the robustness of preconditioners. To achieve this goal, a variety of approaches have been proposed, such as techniques to increase matrix diagonal dominance, locally or globally, by systematic or ad hoc modifications of the decomposition [37], [41], [1], procedures to find a nearby matrix with a breakdown-free incomplete decomposition [2], or symmetric permutations of the system matrix [27], [7]. Combining dropping by value with additional enhancements (as balancing size of the preconditioner and its efficiency by sorting computed factor entries and choosing a part of them [45]) can then provide preconditioners which are in many cases very powerful. A new

*This work was supported by Spanish grants MTM 2004-02998 and MTM 2007-64477 and by the project No. IAA100300802 of the Grant Agency of the Academy of Sciences of the Czech Republic.

†Instituto de Matemàtica Multidisciplinar, Universitat Politècnica de València, 46022 València, Spain (rbru@imm.upv.es, jmarinma@imm.upv.es, jmasm@imm.upv.es)

‡ Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic, (tuma@cs.cas.cz).

decomposition which is in the SPD case inherently breakdown-free was proposed in [48], see also [36]. Some other practical changes have significantly improved iterative preconditioned methods by incomplete decompositions [35], [46] and led to efficient implementations. New permutations forcing a strong diagonal of the system matrix have become useful in this field as well [5], [32]. These permutations are based on efficient sparse algorithms and implementations to perform this task [25], [26]. Note that here we just sketched only a small fraction of the whole development in the field of incomplete factorizations. For more detailed review of some of these developments see the survey [3].

Increasing interest in approximate inverse preconditioners was mainly motivated by the need to enhance efficiency of iterative methods on vector and parallel computer architectures [24], [34]. Later it was found that sophisticated incomplete approximate inverses [30], [38], [8] offer advantages for preconditioned iterative methods also for uniprocessor and sequential implementations [9]. Further on we will mainly restrict to factorized approximate inverses being aware that unfactorized approximate inverses can be in some cases the methods of choice. One of the explanations that the state-of-the-art implementations of approximate inverse preconditioners can be competitive even in sequential computational environment is that they may capture long interactions among matrix entries more successfully than standard incomplete decompositions [17], [10]. This motivates not only further development of sparse approximate inverses but also a search for direct incomplete decompositions which are based on approximate inverses, or which use them as auxiliary structures. One step along this line brought up a new robust incomplete decomposition RIF of symmetric and positive definite matrices [11]. In this case, the triangular factor is computed directly from the SAINV factorized approximate inverse [11]. Computation of this approximate factorized inverse is breakdown-free for all SPD matrices. Progress in solving difficult problems also indicates that considering the matrix inverse during the factorization may be a right way to get better preconditioners [14], [47]. In particular, the authors show in these papers that computing estimates of the inverses of the factors can significantly increase robustness of incomplete LU factorizations.

In this paper we present a new incomplete factorization which computes the direct and inverse factors at the same time. This factorization is derived from the factorized approximate decomposition AISM (Approximate Inverse Sherman-Morrison) [18]. There is a subtle relation between the AISM algorithm and the AINV decomposition [6], [8] on a larger matrix [15], but we do not follow this link here. Instead we get a new surprising insight into the approximate inverse factors by closely watching the factors produced by the AISM algorithm. Although we present the insight for decomposition of general nonsymmetric matrices, later we restrict just to SPD matrices. In this case we obtain, at the same time, the factors L, D and L^{-1} of the LDL^T factorization. Computation of all these factors is interleaved and they both symbolically and numerically influence each other. Moreover, the order of the computation can be used to mutually balance their conditioning by dropping, hence the name Balanced Incomplete Factorization (BIF). In particular, we use the dropping rules introduced by M. Bollhöfer and Y. Saad for LU factorizations and based on the theory developed by them [14], [47].

Section 2 gives the insight into the inverse Sherman-Morrison (ISM) decomposition, shows the structure of its factors and presents the main theoretical result of the paper. Section 3 deals with some additional theory which may be useful for stabilization of the approximate ISM (AISM) decomposition in general case. Section 4

explains the way to stabilize the computation of the the new direct triangular factor by balancing dropping rules and gives the algorithm in the form of pseudocode. Then we present results of the numerical experiments showing very promising behaviour of the new factorization, and conclude the paper by some additional notes.

2. Structure of the ISM decomposition. This section describes the new factorization via the exact inverse Sherman-Morrison (ISM) decomposition. Suppose that the general nonsymmetric matrix A can be written as

$$A = A_0 + \sum_{k=1}^{n} x_k y_k^T$$

where A_0 is a nonsingular matrix and $\{x_k\}_{k=1}^n$ and $\{y_k\}_{k=1}^n$ are two sets of vectors in $\mathbb{R}^n$. We recall that the inverse of a matrix when using the Sherman-Morrison formula (see [18] for details), is given by

$$A_0^{-1} - A^{-1} = A_0^{-1} U_{A_0} D_{A_0}^{-1} V_{A_0}^T A_0^{-1},$$

where U_{A_0} and V_{A_0} have the column vectors u_k and v_k given by

$$u_k = x_k - \sum_{i=1}^{k-1} \frac{v_i^T A_0^{-1} x_k}{r_i} u_i \quad \text{and} \quad v_k = y_k - \sum_{i=1}^{k-1} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i,$$

respectively, and $D_{A_0} = \mathrm{diag}(r_1, \ldots, r_n)$, $r_k = 1 + y_k^T A_0^{-1} u_k = 1 + v_k^T A_0^{-1} x_k$ for $k = 1, 2, \ldots, n$.

When we choose, for simplicity,

$$A_0 = sI_n, \quad s > 0, \quad x_k = e_k \quad \text{and} \quad y_k = (a^k - a_0^k)^T,$$

where a^k stands for the $k-$th row of the matrix A and a_0^k stands for the $k-$th row of the matrix A_0, we obtain from above

$$u_k = x_k - \sum_{i=1}^{k-1} \frac{(v_i)_k}{sr_i} u_i \quad \text{and} \quad v_k = y_k - \sum_{i=1}^{k-1} \frac{y_k^T u_i}{sr_i} v_i, \tag{2.1}$$

where $(v_i)_k$ denotes the $k-$th component of the vector v_i. Then we have

$$A^{-1} = s^{-1} I - s^{-2} U_s D_s^{-1} V_s^T,$$

which expresses the inverse Sherman-Morrison (ISM) decomposition in the matrix form, where the subscript s denotes the potential dependence of the factors on the parameter s. The following lemma from [18] introduces an auxiliary unit upper triangular matrix W which helps to provide a better insight into the ISM decomposition.

LEMMA 2.1. *[18] Let U_s, V_s and $D_s = \mathrm{diag}(r_1^s, \ldots, r_n^s)$ be the matrices computed by the exact factorization algorithm ISM for some $s > 0$. Let U, V and $D = \mathrm{diag}(r_1^1, \ldots, r_n^1)$ be the matrices computed by the exact factorization algorithm ISM for $s = 1$. Then,*

$$U_s = U, \tag{2.2}$$

$$V_s = V - (s-1)W, \tag{2.3}$$

$$D_s = s^{-1}D, \tag{2.4}$$

where the $k-$th column of W is

$$w_k = x_k - \sum_{i=1}^{k-1} \frac{y_k^T u_i}{r_i^1} w_i. \tag{2.5}$$

From the construction of matrices U and W it follows that both these matrices are unit upper triangular. The following theorem shows the structure of the matrix V_s more in detail.

THEOREM 2.2. *Let there exist the exact ISM decomposition*

$$A^{-1} = s^{-1}I - s^{-2}U_s D_s^{-1} V_s^T \tag{2.6}$$

for some $s > 0$. Let W be the upper triangular matrix defined above. Then $V_s^T = DU^{-1} - sW^T$.

Proof. From (2.6) and Lemma 2.1 we get

$$\begin{aligned} s^{-1}I - A^{-1} = s^{-2}U_s D_s^{-1} V_s^T = U(s^{-1}D_s^{-1})(s^{-1}V_s^T) = \\ = UD^{-1}(s^{-1}V^T - (1 - s^{-1})W^T). \end{aligned} \tag{2.7}$$

Taking limit $s \to \infty$ we arrive at

$$A^{-1} = UD^{-1}W^T. \tag{2.8}$$

From (2.6) and (2.8) we then get

$$UD^{-1}W^T = s^{-1}I - s^{-1}UD^{-1}V_s^T.$$

That is

$$UD^{-1}V_s^T = I - sUD^{-1}W^T.$$

Consequently,

$$V_s^T = DU^{-1} - sW^T. \tag{2.9}$$

□

Using the introduced notation for the ISM factors U, V and W computed for $s = 1$ we arrive at the following corollary.

COROLLARY 2.3. *Let there exist the exact ISM decomposition (2.6) for some s and and let $A = \bar{L}\bar{D}\bar{U}$ be the LDU decomposition of A. Moreover, let W be defined as in (2.5). Then*

$$\bar{L} = W^{-T} \quad \textit{and} \quad \bar{D}\bar{U} = DU^{-1}. \tag{2.10}$$

Moreover,

$$\bar{D} = D, \qquad \bar{U} = U^{-1}.$$

Proof. Note that from (2.8) we have

$$A = W^{-T} D U^{-1}.$$

Then, the result easily follows from the uniqueness of the triangular decompositions of A and the structure of V_s described in Theorem 2.2. □

For clarity, the structure of V_s for a given s can be written as follows, using the introduced notation and the assumption of Theorem 2.2.

COROLLARY 2.4. *We have*

$$V_s = \bar{U}^T \bar{D} - s\bar{L}^{-T}. \tag{2.11}$$

In particular, for SPD A we have

$$V_s = \bar{L}\bar{D} - s\bar{L}^{-T}. \tag{2.12}$$

Pictorially, we have (separately for the *strict triangular* parts of V_s and its *diagonal*)

$$V_s = \begin{bmatrix} \ddots & & -sW^T \equiv -s\bar{L}^{-T} \\ & \ddots & \\ \bar{U}^T\bar{D} & & \ddots \end{bmatrix}, \qquad \operatorname{diag}(V_s) = \bar{D} - sI. \tag{2.13}$$

Hence, we arrived at a surprising result related to the structure of V_s. It stores both inverse and direct triangular factors of the matrix A. One of them is scaled by a scalar, the other is scaled by the diagonal matrix $D \equiv \bar{D}$. This fact and the way how we get them together inside the ISM algorithm have important consequences for preconditioning iterative methods. Moreover, for an SPD matrix A we have all the information from U in V as well.

The following section will briefly discuss the approximate ISM decomposition. In particular, we will mention the role of the parameter s which provides an additional degree of freedom which we have in the ISM framework. We are motivated to discuss this subject here since the AISM procedure is guaranteed to be breakdown-free only for M-matrices and H-matrices [18], [21]. Such results are parallel to those related to the existence of ILU, see [43] and [42]. Although the main line of this paper covers stabilization of a special case of the algorithm by sophisticated dropping rules, we are also interested in further ways which may contribute to the efficiency of the method, especially in the nonsymmetric case.

3. Approximate ISM decomposition. As mentioned above, this section is devoted to getting more insight into the role of the parameter s in the exact and approximate ISM (AISM) decompositions. First, note that a redefinition of $s > 0$ does not influence the breakdown-free property of the exact ISM decomposition. Namely, we have the following simple proposition which is a corollary of Lemma 2.1.

PROPOSITION 3.1. *Let A be a square matrix. The ISM decomposition of A exists for the positive parameter s if and only if the ISM exists for any other parameter $t > 0$.*

Proof. Observe from (2.4) that for every two positive values of the ISM parameter

$$D_t = \frac{s}{t} D_s. \tag{3.1}$$

Therefore, if the ISM decomposition exists for some $s > 0$, then it exists for any positive parameter t. Moreover, the smaller s, the bigger diagonal entries (pivots). □

The real hint for the choice of s in the ISM method follows from the structure of V_s in (2.13). Namely, the parameter s influences mutual scaling of the factors stored in the lower and upper triangles of V_s, respectively. The rule to *equate approximately norms of both triangular factors*, which we will call the *scaling rule*, can be used to compute the value of s. Although we express this fact, we will not follow its implications here. Instead, we prefer to show the potential of the new approach in its most basic form.

When constructing the AISM preconditioner a dropping is used to obtain factors U, D_s and V_s. Consider now two AISM decompositions with different parameters s and t and with the same dropping rules. Further, we assume that we do not drop the diagonal entries of the factors U and V. The following Theorem is easy to be proved.

THEOREM 3.2. *Let A be a square matrix such that there exist the AISM decomposition, with the same dropping rules, $\tilde{A}_s^{-1} = s^{-1}I - s^{-2}\tilde{U}_s\tilde{D}_s^{-1}\tilde{V}_s^T$ and $\tilde{A}_t^{-1} = t^{-1}I - t^{-2}\tilde{U}_t\tilde{D}_t^{-1}\tilde{V}_t^T$ for two parameters s and t, respectively. Then,*

$$\tilde{U}_s = \tilde{U}_t, \quad s\tilde{D}_s = t\tilde{D}_t \text{ and } \operatorname{tril}(\tilde{V}_s) = \operatorname{tril}(\tilde{V}_t)$$

where tril *means the strict lower triangular part of the corresponding matrix.*

Using Theorem 3.2 we can bound the difference between two approximate inverses of A induced by the two different parameters s and t for the AISM decomposition, and observe where the dependence on the parameter takes place. We can write for the two computed inverses

$$\tilde{A}_s^{-1} = s^{-1}I - s^{-2}\tilde{U}_s\tilde{D}_s^{-1}\tilde{V}_s^T \qquad \text{and} \qquad \tilde{A}_t^{-1} = t^{-1}I - t^{-2}\tilde{U}_t\tilde{D}_t^{-1}\tilde{V}_t^T$$

Recall the relations in Theorem 3.2 and note that from equation (2.9) we have

$$\tilde{V}_s^T = \tilde{D}\tilde{U}^{-1} - Z_s \text{ and } \tilde{V}_t^T = \tilde{D}\tilde{U}^{-1} - Z_t, \tag{3.2}$$

where Z_t and Z_s represent the lower triangular part of $\tilde{V}_t^T$ and $\tilde{V}_s^T$, respectively. Here $\operatorname{diag} Z_t = tI$ and $\operatorname{diag} Z_s = sI$. Then we can write

$$\begin{aligned}
\tilde{A}_t^{-1} - \tilde{A}_s^{-1} &= \left(\frac{1}{t} - \frac{1}{s}\right) I - \frac{1}{t}\tilde{U}_t t^{-1}\tilde{D}_t^{-1}\tilde{V}_t^T + \frac{1}{s}\tilde{U}_s s^{-1}\tilde{D}_s^{-1}\tilde{V}_s^T \\
&= \left(\frac{1}{t} - \frac{1}{s}\right) I - \frac{1}{t}\tilde{U}\tilde{D}^{-1}\tilde{V}_t^T + \frac{1}{s}\tilde{U}\tilde{D}^{-1}\tilde{V}_s^T \\
&= \left(\frac{1}{t} - \frac{1}{s}\right) I - \tilde{U}\tilde{D}^{-1}\left(\frac{1}{t}\tilde{V}_t^T - \frac{1}{s}\tilde{V}_s^T\right)
\end{aligned}$$

by equation (3.2)

$$= \tilde{U}\tilde{D}^{-1}\left(\frac{1}{t}\tilde{Z}_t - \frac{1}{s}\tilde{Z}_s\right).$$

Taking norms we arrive at

$$\|\tilde{A}_t^{-1} - \tilde{A}_s^{-1}\| \leq \|\tilde{U}\|\|\tilde{D}^{-1}\| \left\| \frac{1}{t}\tilde{Z}_t - \frac{1}{s}\tilde{Z}_s \right\|.$$

It means that the proximity of the approximated inverses for two parameters is related to the difference betweeen the lower triangular parts of $\tilde{V}_t^T$ and $\tilde{V}_s^T$ (which can be expressed via the matrix W^T) divided by the corresponding parameter. As above, this confirms the scaling role of the parameter s.

As we have seen above, if we face a breakdown, the new choice of s may not be sufficient to get a successful decomposition even if it influences a different dropping. We typically need to use stronger modifications of A. Nevertheless, based on the strong connection between ILU and AISM decompositions, we can proceed similarly as in [42] for ILU, as stated in the following simple result.

LEMMA 3.3. *Let A be a square matrix such that the AISM decomposition does not exist. Then, the matrix $A(\alpha) = A + \alpha I$ has AISM decomposition for some $\alpha > 0$ large enough.*

The result implies that in practice, the decomposition can be based on an iterative process in which we may increase the parameter α until a given maximum value, since the quality of the preconditioner could be adversely affected for large α. A contemporary use of this iterative strategy for incomplete decompositions can be found in [40] where the quality of the preconditioner is analyzed as a function of the shift value. Note this iterative process enables to apply easily the scaling rule mentioned above if we keep track of the norms of computed factors. Further potential improvement based on the dynamic choice of the scaling parameter is out of scope of this paper. In our experiments we used another way for improving incomplete factors, which is described below.

4. Balancing Incomplete Factorization in the SPD case. In this and the subsequent section we will restrict ourselves to the symmetric and positive case. Later, we will give a few comments on practical issues related to solving general nonsymmetric problems. Our restriction is motivated by the desire to describe one particular preconditioning approach more in detail.

Balanced incomplete factorization (BIF) $\hat{L}\hat{D}\hat{L}^T$ of an SPD matrix is called the algorithm to construct the lower triangular matrix $\hat{L}$ and the diagonal matrix $\hat{D}$ which approximate the matrices $\bar{L}$ and $\bar{D}$, respectively, using incompletely the formulas (2.1) for $k = 1, \ldots, n$. Both the incomplete factors $\hat{D}$ and $\hat{L}$ are stored via the approximation $\hat{V}_s$ to the matrix V_s as shown in (2.13). In addition, we compute in this way the matrix $\widehat{L^{-1}}$ which is an approximation to $\bar{L}^{-1}$. The incompleteness is controlled by the dropping rules described below which mutually balance magnitudes of entries in the approximate direct and inverse triangular factors $\hat{L}$ and $\widehat{L^{-1}}$. Note that the scaled entries of the approximation $\hat{U}$ to U are also contained in $\hat{V}_s$ and can be retrieved from there. Once the matrix $\hat{V}_s$ is computed, only its scaled lower triangular part and diagonal, which represent $\hat{L}$ and $\hat{D}$, are used in the iterative method.

Let us describe our balancing dual dropping rules. Relations between direct factors from LU decomposition and inverse factors from sparse factorized inverses were studied by M. Bollhöfer and Y. Saad in [16]. Robust dropping rules based on the resulting analysis were used in [13] and [14]. Suppose we have an incomplete decomposition $A \approx \hat{L}\hat{D}\hat{L}^T$, and we intend to apply it as a preconditioner for the conjugate gradient (CG) method. It is well known, that an important role in the

preconditioned method is played by the transformed matrix which can be written as $\hat{L}^{-1}A\hat{L}^{-T}$. Consider dropping by value and denote the drop tolerance by τ. In [16] it is justified to drop the entries $\hat{l}_{jk}$ of a column k of $\hat{L}$ if they satisfy

$$|\hat{l}_{jk}| \parallel e_k^T \bar{L}^{-1} \parallel \leq \tau. \tag{4.1}$$

Namely, for dropping in the $k-$th column of $\hat{L}$ we need to know the norm of the $k-$th row of $\bar{L}^{-1}$. This dropping rule then implies that the entries of the exact inverse $\hat{L}^{-1}$ of the computed incomplete factor $\hat{L}$ of the incomplete LDL^T factorization are close to the corresponding entries of the directly computed factorized approximate inverse.

As mentioned above, the BIF algorithm applied to the SPD matrix A computes at the same time, in addition to $\hat{D}$, both the incomplete factors $\hat{L}$ and $\widehat{L^{-1}}$. Consider for a moment, that our goal is to use also $\widehat{L^{-1}}$. Denote $\hat{\ell}_{jk} = (\widehat{L^{-1}})_{jk}$. Then (4.1) applied to direct computation of the incomplete inverse factor $\widehat{L^{-1}}$ implies that we drop entries $\hat{\ell}_{jk}$ of the column k of $\widehat{L^{-1}}$ if they satisfy

$$|\hat{\ell}_{jk}| \parallel e_k^T \bar{L} \parallel \leq \tau \tag{4.2}$$

Theorem 2.2 enables us to apply both dropping rules (4.1) and (4.2) directly if we replace the exact quantities inside the norms by their approximations computed throughout.

The basic scheme of the dense left-looking implementation of the BIF algorithm with the dropping rules based on (4.1) and (4.2) is given below as Algorithm 4.1.

ALGORITHM 4.1. BIF algorithm

Input: $A = (a_{i,j})$ (SPD matrix), *dropv* (drop tolerance for the auxiliary matrix $\hat{V} = (v_{i,j})$), *dropu* (drop tolerance for the auxiliary matrix $\hat{U} = (u_{i,j})$), s (scaling parameter)

Output: *Factors* $\hat{L}$ *and* $\hat{D}$ *of incomplete* LDL^T *factorization of* A.

for $k = 1, \ldots, n$

$v_{1:n,k} = a_{k,1:n}^T, \quad v_{k,k} = v_{k,k} - s$ (initialize the k-th column of $\hat{V}$)

$norm\_l_{1:n} = 0$ (initialize norms of rows of $\hat{L}$)

for $i = 1, \ldots, k-1$ (update the k-th columns of $\hat{U}$ and $\hat{V}$)

$$v_{1:n,k} = v_{1:n,k} - \frac{a_{k,1:n}u_{1:n,i}}{sd_i} v_{1:n,i}$$

$$u_{1:n,k} = u_{1:n,k} - \frac{v_{k,i}}{sd_i} u_{1:n,i}$$

end for

$d_k = v_{k,k}/s + 1.0$

$norm\_invl_k = \sqrt{v_{1:k-1,k}^T v_{1:k-1,k} + s^2}/s$

$temp = 1.0/(d_k d_k)$

for $i = k+1, \ldots, n$ (update norms of rows of L)

$norm\_l_i = norm\_l_i + v_{i,k} v_{i,k}\, temp$

end for

$norm\_l_k = \sqrt{norm\_l_k + 1.0}$

Keep in $u_{1:n,k}$ *only the entries with absolute value larger than dropu*

(standard dropping in $\hat{U}$)

Keep in $v_{1:k-1,k}$ *only the entries* v_{ik} *with magnitudes larger than* $dropv/norm\_l_i$

(first part of BIF dropping in $\hat{V}$)

Keep in $v_{k+1,n}$ only the entries with magnitudes larger than $dropv*d_k/norm\_invl_k$
(second part of BIF dropping in $\hat{V}$)

end for

Return *$\hat{D} = diag(\hat{V} + sI)$ and $\hat{L}$, which is the lower triangular part of $\hat{V} + sI$ scaled by $\hat{D}^{-1}$ from the right.*

In Algorithm 4.1 we also explicitly deal with the auxiliary matrix $\hat{U}$ which is an approximation to U. As mentioned in Section 2, the vector $u_{1:n,i}$ used to update $v_{1:n,k}$ can be equivalently replaced by the vector $col = [v_{1:i-1,i}^T/s, -1.0, zeros_{i+1:n}^T]^T$, where $zeros$ is the vector of all zeros of the dimension n. If we do this substitution, the matrix $\hat{U}$ is not needed at all. Even when in our experiments we obtained the same results with or without $\hat{U}$, we mention it here for explanatory reasons and possible future floating-point analysis of the two different algorithmic possibilities. The vector of norms of the rows of $\hat{L}$, which is denoted by $norm\_l$, is updated throughout the algorithm. The vector of norms of the rows of $\widehat{L^{-1}}$ denoted by $norm\_invl$ is computed throughout from the upper triangular part of $\hat{V}_s$.

Let us give some details on the sparse implementation which we used in our experiments. Note that we will not discuss the use of matrix $\hat{U}$ which does not need to be used. In each major step, for $k = 1, \ldots, n$, we compute one column of the matrix $\hat{V}_s$. In order to have fully sparse implementation of the BIF algorithm we need to store sparse $\hat{V}_s$ such that we have a fast access to both its columns and rows. The access to its columns is needed since the update of $v_{1:n,k}$ for $i = 1, \ldots, k-1$ is performed by substracting a linear combination of the previous columns of $\hat{V}_s$. In order to avoid the full loop from 1 to $k-1$ in this update we need to know which of these previous columns have at least one nonzero entry in the same position as $a_{k,1:n}$, which is the k-th row of A. Note that $a_{k,1:n}$ is accessed in the dot products used for the update of columns of $\hat{V}_s$. Such information is easy to get if we have a fast access to the rows of $\hat{V}_s$. More in detail, we get the indices of the columns of $\hat{V}_s$ with a nonzero contribution to $v_{1:n,k}$ by unifying the structures of the rows of the current $\hat{V}_s$ which correspond to the indices of nonzero entries of $a_{k,1:k-1}$ being less than k.

Consequently, we store $\hat{V}_s$ by columns in a standard compressed sparse by row (CSR) format [33], cf. also [44]. This data structure is reallocated if needed. In addition, we store at most $lsize$ entries with largest magnitudes of each row of $\hat{V}_s$ (having thus the space for $\hat{V}_s$ stored by rows bounded by $n * lsize$). Once a column v_k is evaluated and sparsified by dropping we check its nonzero entries, compare their magnitudes with those which are stored in their rows and keep only those with largest magnitudes. Clearly, the two different data structures which store $\hat{V}_s$ by columns and rows, respectively, do not necessarily contain the same information. Once the computation of columns of $\hat{V}_s$, dropping its entries and update of the incomplete rowwise representation of $\hat{V}_s$ are explained, the sparse implementation of the remaining lines of Algorithm 4.1 is straightforward.

The coupled computation of direct and inverse factors reflects the accumulated experience with AISM that $\hat{U}$ and $\hat{V}_s$ may profit from different drop tolerances used for them since they store mathematically rather different quantities. Taking into account the theory above, different drop tolerances should be used for different parts of columns of $\hat{V}_s$, if simpler dropping rules would be applied.

If we compare the direct and inverse factors computed by the left-looking implementation of the BIF algorithm with the factors which are obtained by RIF [11], we observe important differences. RIF provides in each step different quantities than

BIF. Namely, a row of the approximation $\widehat{L^{-1}}$ to $\bar{L}^{-1}$ and a row of the approximation $\hat{L}$ to $\bar{L}$, are evaluated in each step of the left-looking RIF algorithm. The factor $\hat{L}$ obtained in RIF as a side-product of the SAINV algorithm [11] does not have any influence on the computation of the factor $\widehat{L^{-1}}$. In contrast, computation of direct and inverse factors in BIF is *coupled*, both factors $\widehat{L^{-1}}$ and $\hat{L}$ use information from each other during the whole computation and can positively influence each other.

In the following section we will present a few experimental results showing the great potential of the new algorithm. Even when we do not cover here the nonsymmetric case let us give a few comments on its implementation. First, to get both incomplete lower and upper triangular factors of the LU decomposition, we need to run the process for both A and A^T, similarly as in getting approximate inverses by biconjugation [8]. If we interleave these processes, we have always available partially formed approximate factors and approximations of their inverses. Consequently, dropping rules motivated by [13] could be then applied, but we potentially double memory demands. Or, we can compute the incomplete upper triangular factor from A, the incomplete lower triangular factor from A^T, and propose a different dropping strategy. Our preliminary experience points out that the scaling rule may be taken into account.

5. Numerical experiments. This section is devoted to numerical experiments with the new factorization which is used as a preconditioner for the CG method. In particular, we are interested in solving large and ill-conditioned problems. The main goal of this section is to show that the new approach is practically robust for solving these problems, and rather cheap to compute. We will see that, in general, the new approach provides better results compared to those obtained with the RIF preconditioner.

As a baseline method we report results for Jacobi preconditioning which may give an idea of difficulty of the test problems. A natural competitor of the new approach used in our comparison is the RIF preconditioner, which may be considered as one of the methods of choice among robust approaches [11]. Moreover, the RIF preconditioner is also based on factorized approximate inverses, and the comparison may be useful in order to find similarities and contrasts between both approaches. Note that standard preconditioners based on drop tolerances failed on most of the problems, or we were not able to find parameters which would force them to run. Standard level-based preconditioners including IC(0) also failed or were extremely inefficient, and we do not report results with them as well. Note that for most of the test problems the factor $\hat{L}$ from IC(0), which has the same sparsity pattern as the lower triangular part of A, may be a very poor approximation of the exact Cholesky factor $\bar{L}$ of A. This can result in the preconditioned iterative method which would be rather inefficient. In addition, if A is not M-matrix or H-matrix, the decomposition can even break down. In this case, the Jacobi preconditioner may be often preferred, especially in parallel environment, and/or if we perform only a few iterations, since the Jacobi preconditioned conjugate gradient method is known to converge under weak assumptions (in exact arithmetic), and it often converges in a reasonable amount of iterations. In some sense, our experiments present a focused extension of that considered in [11].

Our sparse left-looking implementation was described in the previous Section. Note that in all our experiments we chose $lsize = 10$. To our surprise, the number of iterations of the preconditioned CG (PCG) method was largely insensitive to the choice of $lsize$. Note that this fact is in contrast with dual dropping in the AINV

TABLE 1
Test problems

Matrix	n	nz	Application	Source
BCSSTK35	30,237	1,450,163	Automobile seat frame	U. of Florida [23]
VANBODY	47,072	1,191,985	Van body model	The PARASOL project
CT20STIF	52,329	1,375,396	Engine block	U. of Florida [23]
CFD1	70,656	949,510	CFD pressure matrix	U. of Florida [23]
OILPAN	73,752	1,835,470	Car olipan	The PARASOL project
X104	108,384	5,138,004	Beam joint	The PARASOL project
CFD2	123,440	1,605,669	CFD pressure matrix	U. of Florida [23]
ENGINE	143,571	2,424,822	Engine head	R. Kouhia [39]
PWTK	217,918	5,926,171	Pressurize wind tunnel	U. of Florida [23]
HOOD	220,542	5,494,489	Car hood	The PARASOL project
INLINE_1	503,712	18,660,027	Inline skater	The PARASOL project
LDOOR	952,203	23,737,339	Large door	The PARASOL project

decomposition [8], where we typically get very different numbers of nonzero entries in the factor rows. Uniform bound $lsize$ on these numbers may spoil the efficiency of AINV. The fact that the parameter $lsize$ is used only for an auxiliary data structure in BIF seems to be crucial for the difference in behavior between BIF and AINV. As a further simplification in our implementation, we always use $s = 1$, and this corresponds to our goal not to optimize the performance of the preconditioners in the other way than by balancing. Our matrices used natural ordering and were not initially scaled.

The test matrices are listed in Table 1. Many of them are quite ill-conditioned. Note that we have considered here some test problems used in [11] and extended this choice by larger problems. For each matrix we provide the problem size n, the number nz of nonzeros in the lower triangular part, the application field in which the matrix was created, and the source. Note that the matrices from the Parasol project are currently available from a depository in RAL described in [29].

Each problem was solved by the preconditioned conjugate gradient method for a relative decrease 10^{-6} of the system backward error, allowing a maximum of 2,000 iterations. For the experiments we used an artificial right-hand side computed as $b = Ae$, where e is the vector of all ones. The initial guess was the vector of all zeros. The computations have been performed using one processor Intel Pentium 4 (3GHz, 1GB RAM). The codes written in Fortran 90, have been compiled with Compaq Visual Fortran 6.6a.

In Table 2 we present results obtained with the Jacobi preconditioner, that is solving the diagonally-scaled problem. For each problem we provide the number of iterations of the preconditioned conjugate gradient method as well as the elapsed user time for the computation obtained with a system function *etime*.

Table 3 presents the results obtained with RIF and BIF preconditioners. For each method we report the ratio which we get if the number of nonzero entries in the approximate factor $\hat{L}$ is divided by the number of nonzero entries in the lower triangular part of A. This ratio is denoted by $relsize$. Further we show the time for constructing the preconditioner (t_p), the number of PCG iterations (its) and the time for the iterative solution phase (t_it).

The parameters to apply the dropping rules for both RIF and BIF were chosen so as to obtain preconditioners with very similar size. In all cases we considered rather

TABLE 2
JCG preconditioning

Matrix	JCG its	JCG time
BCSSTK35	464	2.38
VANBODY	605	4.81
CT20STIF	389	3.69
CFD1	814	6.20
OILPAN	736	8.95
X104	1100	33.4
CFD2	854	10.8
ENGINE	686	13.8
PWTK	1178	44.8
HOOD	666	27.7
INLINE_1	764	101.
LDOOR	810	145.

TABLE 3
Comparison of the RIF and BIF preconditioners

Matrix	RIF				BIF			
	relsize	*t_p*	*its*	*t_it*	*relsize*	*t_p*	*its*	*t_it*
BCSSTK35	0.18	0.50	60	0.47	0.18	0.16	45	0.38
VANBODY	0.07	0.75	8	0.09	0.07	0.08	7	0.08
CT20STIF	0.05	1.00	62	0.69	0.05	0.06	61	0.67
CFD1	0.74	11.5	287	4.08	0.77	0.83	291	4.38
OILPAN	0.08	1.20	134	1.92	0.17	0.17	141	1.97
X104	†	†	†	†	0.24	0.88	44	1.89
CFD2	0.50	5.40	372	8.40	0.55	0.62	389	8.55
ENGINE	0.06	1.48	213	4.95	0.06	0.25	208	4.88
PWTK	0.21	4.03	14	0.81	0.20	1.01	26	1.39
HOOD	0.07	3.36	320	16.1	0.06	0.60	319	16.3
INLINE_1	0.04	16.4	149	22.4	0.04	1.65	156	23.3
LDOOR	0.05	13.9	167	34.9	0.05	2.27	168	36.0

sparse preconditioners in order to show that a reasonable efficiency can be obtained even with a small amount of additional information extracted from the matrix. Note that we did not do any tuning of preconditioners for optimal performance. We can see that both preconditioners seem to be similarly robust. Apart from one case when the computation of RIF failed (for a wide spectrum of parameters) since the underlying SAINV process [4] needed excessive amount of memory, we consider both approaches efficient, having similar degree of robustness. As for the timings, the new preconditioner is much cheaper to compute. This can be possibly explained by combination of two effects. First, BIF is based on cache-efficient, and primarily, columnwise implementation whereas in RIF we switch between two matrices stored in different data structures. The BIF implementation is principially much simpler. Second, we believe that it is mainly caused by the dropping rules which make the factors more stable, producing larger diagonal entries and generating thus less fill-in. If we would consider total timings, the new BIF approach would be a clear overall winner. Note that we do not consider here the intermediate memory for the SAINV process hidden inside RIF

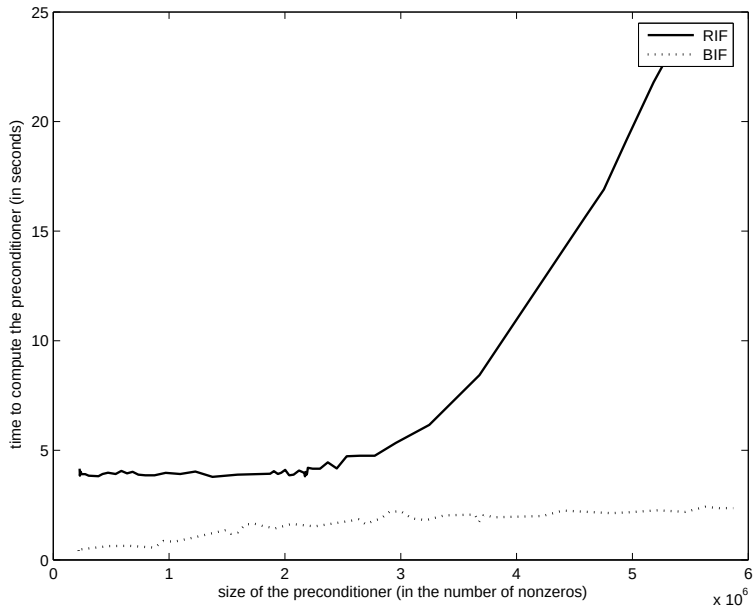

FIG. 5.1. *Sizes of RIF and BIF preconditioners (in numbers of their nonzeros) versus time to construct them (in seconds) for the matrix PWTK.*

computation. We observed that this intermediate memory is typically larger than the additional memory needed for the new approach.

We also performed some experiments with a direct method. For this purpose we used the MA57 code from the Harwell Subroutine Library with the native AMD reordering. The code was able to solve all our linear systems except for the two largest ones. Even when this particular direct method provided larger timings in most test cases, we believe that one can tune both approaches to get very similar results from the efficiency point of view for the considered problem sizes. But we also believe that the classes of direct and iterative methods offer very different merits. They are in many senses complementary, and their comparison should be always purpose-oriented. Note that one of the principial features of the direct methods is that they can get the solution with a small backward error. In contrast, in the case of the preconditioned conjugate gradients we typically need and get only a rough approximation of the solution, where its approximativeness is measured relatively, e.g., with respect to the original backward error norm or the original residual norm.

Comparison of preconditioners is always a multivariate problem. The results represented via a table, even if the corresponding numbers are carefully selected, may not tell us the whole story. In the following we will pay attention at the results obtained for the matrix PWTK.

Figure 5.1 shows the time to compute the preconditioners of different sizes.

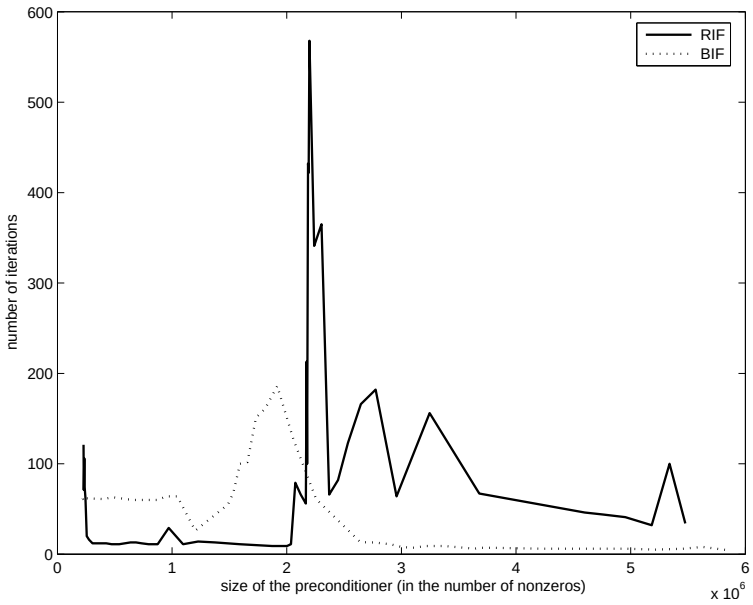

FIG. 5.2. *Sizes of RIF and BIF preconditioners (in numbers of their nonzeros) versus number of iterations for the conjugate gradient method preconditioned by them.*

Clearly, the setup time is much smaller for all sizes of preconditioners. While we were able to choose parameters for RIF to make the preconditioner density even larger (following the increase in the setup time) this was not the case of BIF. Its density is naturally limited by the size of the row structures of $\hat{V}_s$ on one side, and by the dropping rules based on the norms of rows of $\hat{L}$ and $\widehat{L^{-1}}$ on the other side.

Figure 5.2 shows the dependence of the number of iterations of the PCG method on the size of the preconditioner. We can see that there are regions of sizes of the preconditioners for which the RIF preconditioner is more efficient than the BIF preconditioner in terms of the number of iterations for the same size. The other effect visible in Figure 5.2 is the more uniform behavior of the curve for BIF. The jumps in the curve seem to have more limited amplitudes. We believe that this may be attributed to the relative dropping used in BIF. In contrast to the results presented in Table 3, the intervals of preconditioner sizes for which RIF and BIF have a very small number of PCG iterations are rather different for this matrix.

Finally, Figure 5.3 shows dependence of the total time on the sizes of preconditioners for RIF and BIF. Here, by the total time we denote the sum of the time to compute the preconditioner and the time for PCG. Due to the very small setup time, BIF is here better, and sometimes much better, for most of its sizes. What we consider even more important is that its behavior is more uniform. The main goal of this section was to point out that the new approach is practically robust for solving difficult

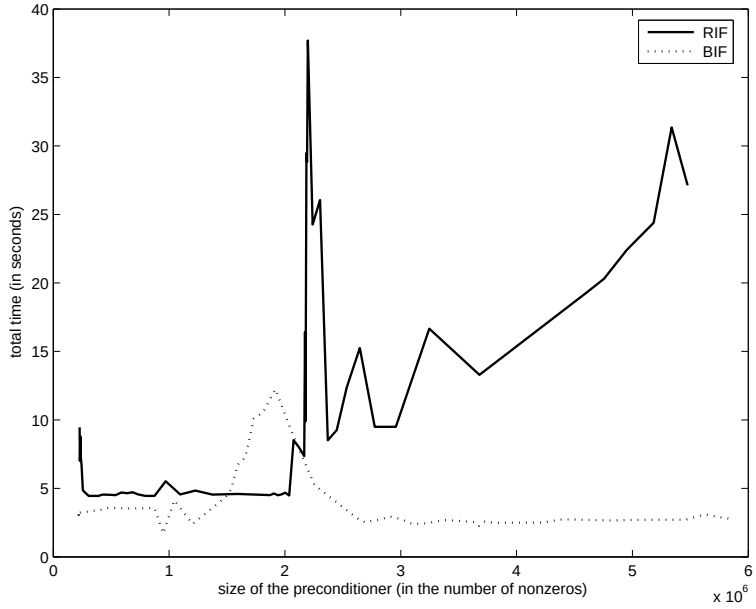

FIG. 5.3. *Sizes of RIF and BIF preconditioners (in numbers of their nonzeros) versus the total time (the time to compute the preconditioner plus the time for the conjugate gradient method).*

problems. BIF seems to satisfy this, even when not proved to be breakdown-free. We believe that this behaviour is also due to the robust dropping based on the results of M. Böllhofer and Y. Saad. We can thus obtain a high-quality preconditioner even in the case when the SAINV process hidden inside the RIF computation generates very high fill-in as in the case of matrix X104. We believe that the favourable properties of BIF experimentally shown for solving symmetric and positive definite systems may be very important in future extensions to nonsymmetric systems, block implementations and iterative solvers of linear least-squares problems (cf. [12]).

6. Conclusions and future work. We have introduced a new incomplete LDL^T factorization of symmetric and positive definite systems by carefully examining the AISM preconditioner. The new factorization is surprisingly simple. The algorithm of this new factorization closely couples computation of both the approximate factors $\hat{L}$ and $\widehat{L^{-1}}$ which influence each other during the course of computation. We have shown that the dropping rules developed by M. Böllhofer and Y. Saad can be easily applied into the computational algorithm, and therefore the growth in both $\hat{L}$ and $\widehat{L^{-1}}$ can be balanced. This balancing gives the name to the new approach: balanced incomplete factorization (BIF). Numerical experiments suggest that the new technique is reliable and can be considered as a complementary method to the RIF preconditioner, but having much faster setup than RIF. The extensions to preconditioning

nonsymmetric systems and linear least squares are currently under investigation.

7. Acknowledgment. We would like to thank to the anonymous referees and to Michele Benzi for their comments on the previous version which helped to improve the paper.

REFERENCES

[1] M. A. Ajiz and A. Jennings. A robust incomplete Choleski-conjugate gradient algorithm. *Internat. J. Numer. Methods Engrg.*, 20(5):949–966, 1984.
[2] O. Axelsson and L. Kolotilina. Diagonally compensated reduction and related preconditioning methods. *Numer. Linear Algebra Appl.*, 1(2):155–177, 1994.
[3] M. Benzi. Preconditioning techniques for large linear systems: a survey. *J. Comput. Phys.*, 182(2):418–477, 2002.
[4] M. Benzi, J. K. Cullum, and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM J. Sci. Comput.*, 22(4):1318–1332, 2000.
[5] M. Benzi, J. C. Haws, and M. Tůma. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM J. Sci. Comput.*, 22(4):1333–1353, 2000.
[6] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17(5):1135–1149, 1996.
[7] M. Benzi, D. B. Szyld, and A. van Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM J. Sci. Comput.*, 20(5):1652–1670, 1999.
[8] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19(3):968–994, 1998.
[9] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math.*, 30(2-3):305–340, 1999.
[10] M. Benzi and M. Tůma. Orderings for factorized sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21(5):1851–1868, 2000.
[11] M. Benzi and M. Tůma. A robust incomplete factorization preconditioner for positive definite matrices. *Numer. Linear Algebra Appl.*, 10(5-6):385–400, 2003.
[12] M. Benzi and M. Tůma. A robust preconditioner with low memory requirements for large sparse least squares problems. *SIAM J. Sci. Comput.*, 25(2):499–512, 2003.
[13] M. Bollhöfer. A robust ILU with pivoting based on monitoring the growth of the inverse factors. *Linear Algebra Appl.*, 338:201–218, 2001.
[14] M. Bollhöfer. A robust and efficient ILU that incorporates the growth of the inverse triangular factors. *SIAM J. Sci. Comput.*, 25(1):86–103, 2003.
[15] M. Bollhöfer. Personal communication, 2004.
[16] M. Bollhöfer and Y. Saad. On the relations between ILUs and factored approximate inverses. *SIAM J. Matrix Anal. Appl.*, 24(1):219–237, 2002.
[17] R. Bridson and W.-P. Tang. Ordering, anisotropy, and factored sparse approximate inverses. *SIAM J. Sci. Comput.*, 21(3):867–882, 1999.
[18] R. Bru, J. Cerdán, J. Marín, and J. Mas. Preconditioning sparse nonsymmetric linear systems with the Sherman-Morrison formula. *SIAM J. Sci. Comput.*, 25(2):701–715, 2003.
[19] N. I. Buleev. A numerical method for solving two-dimensional diffusion equations. *Atomnaja Energija*, 6:338–340, 1959.
[20] N. I. Buleev. A numerical method for solving two-dimensional and three-dimensional diffusion equations. *Matematičeskij Sbornik*, 51:227–238, 1960.
[21] J. Cerdán, T. Faraj, J. Marín, and J Mas. A block approximate inverse preconditioner for sparse nonsymmetric linear systems. Technical Report No. TR-IMM2005/04, Polytechnic University of Valencia, Spain, 2005.
[22] T. F. Chan and H. A. van der Vorst. Approximate and incomplete factorizations. In *Parallel Numerical Algorithms, ICASE/LaRC Interdisciplinary Series in Science and Engineering IV. Centenary Conference, D.E. Keyes, A. Sameh and V. Venkatakrishnan, eds.*, pages 167–202, Dordrecht, 1997. Kluver Academic Publishers.
[23] T. A. Davis. The University of Florida Sparse Matrix Collection. Tech. Report REP-2007-298, CISE, University of Florida, 2007. Available online at `http://www.cise.ufl.edu/~davis`.
[24] P. F. Dubois, A. Greenbaum, and G. H. Rodrigue. Approximating the inverse of a matrix for use on iterative algorithms on vector processors. *Computing*, 22:257–268, 1979.
[25] 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.*, 20:889–901, 1999.

[26] I. S. Duff and J. Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM J. Matrix Anal.*, 22:973–996, 2001.
[27] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29:635–657, 1989.
[28] T. Dupont, R. P. Kendall, and H. H. Jr. Rachford. An approximate factorization procedure for the solving self-adjoint elliptic difference equations. *SIAM J. Numer. Anal.*, 5:559–573, 1968.
[29] N. I. M. Gould, Y. Hu, and J. A. Scott. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. *ACM Trans. Math. Software, article 10 (electronic)*, 33(2), 2007.
[30] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3):838–853, 1997.
[31] I. Gustafsson. A class of first order factorization methods. *BIT*, 18(2):142–156, 1978.
[32] M. Hagemann and O. Schenk. Weighted matchings for preconditioning symmetric indefinite linear systems. *SIAM J. Sci. Comput.*, 28(2):403–420, 2006.
[33] A. Jennings. A compact storage scheme for the solution of symmetric linear simultaneous equations. *Computing J.*, 9:281–285, 1966.
[34] O. G. Johnson, C. A. Micchelli, and G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM J. Numer. Anal.*, 20(2):362–376, 1983.
[35] M. T. Jones and P. E. Plassmann. An improved incomplete Cholesky factorization. *ACM Trans. Math. Software*, 21(1):5–17, 1995.
[36] I. E. Kaporin. High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$ decomposition. *Numer. Linear Algebra Appl.*, 5:483–509, 1998.
[37] D. S. Kershaw. The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *J. Comp. Phys.*, 26:43–65, 1978.
[38] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings. I. Theory. *SIAM J. Matrix Anal. Appl.*, 14(1):45–58, 1993.
[39] R. Kouhia. *Sparse matrices web page*. available online at http://www.hut.fi/∼kouhia/sparse.html, 2001.
[40] I. Lee, P. Raghavan, and E. G. Ng. Effective preconditioning through ordering interleaved with incomplete factorization. *SIAM J. Matrix Anal. Appl.*, 27(4):1069–1088, 2006.
[41] T. A. Manteuffel. Shifted incomplete Cholesky factorization. In I.S. Duff and G. W. Stewart, editors, *Sparse Matrix Proceedings 1978*, Philadelphia, PA, 1979. SIAM Publications.
[42] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comp.*, 34:473–497, 1980.
[43] 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. *Math. Comp.*, 31:148–162, 1977.
[44] Y. Saad. SPARSKIT: A basic tool kit for sparse matrix computations. Technical Report 90–20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990.
[45] Y. Saad. ILUT: a dual threshold incomplete *LU* factorization. *Numer. Linear Algebra Appl.*, 1(4):387–402, 1994.
[46] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Co., Boston, 1996.
[47] O. Schenk, M. Bollhöfer, and R. A. Römer. On large-scale diagonalization techniques for the Anderson model of localization. *SIAM J. Sci. Comput.*, 28(3):963–983, 2006.
[48] M. Tismenetsky. A new preconditioning technique for solving large sparse linear systems. *Lin. Alg. Appl.*, 154–156:331–353, 1991.
[49] R. S. Varga. Factorizations and normalized iterative methods. In *Boundary problems in differential equations*, pages 121–142, Madison, WI, 1960. University of Wisconsin Press.