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/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 [16]. In contrast to the RIF algorithm [9], 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. Although we describe the theory behind the factorization for general nonsymmetric matrices, in implementation and experiments we restrict for clarity and conciseness only to the case when the system matrix is symmetric and positive definite. In this case, we call the new approximate LDL^T factorization 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.

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 [17], [18]. 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 [45], [20], see also [3]. An increasing amount of attention lead to nice theoretical results for simple model problems [27], [40], [30]. Solving more complicated problems require techniques which increase robustness of preconditioner computation, and result in faster convergence of the iterative method. The published proposals to achieve this goal have included methods to increase matrix diagonal dominance, locally or globally, by systematic or ad hoc modifications of the decomposition [34], [38], [1], procedures to find a nearby matrix with a breakdown-free incomplete decomposition [2], or symmetric permutations of the system matrix [26], [5]. The improvements of basic incomplete decompositions became even more desired when understood that their variants based on dropping small entries by value are typically better than decompositions based on dropping by levels, and drop-tolerance based strategies can be really used in practice. Combining dropping by value with additional enhancements (balancing size with efficiency by sorting computed factor entries and choosing a part of them) [41] can

[*]This work was supported by Spanish grants MTM 2004-02998 and MTM 2007-64477 and by the project 1ET400300415 within the National Program of Research "Information Society".

[†]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`).

then provide preconditioners which are very powerful in many cases. Moreover, some other theoretical improvements [44], [33], or practical changes significantly [32], [42] improved iterative methods preconditioned by incomplete decompositions further on. An important breakthrough came with applying permutations forcing a strong diagonal of the system matrix [4] and with efficient sparse algorithms and implementations to perform this task [24], [25].

Increasing interest in approximate inverse preconditioners was mainly motivated by the need to have more efficient vector and parallel processing [23], [31] of iterative methods. Later it was found that sophisticated incomplete approximate inverses [29], [35], [6] offer advantages for preconditioned iterative methods also for uniprocessor and sequential implementations [7]. 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 [15], [8]. 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 make use of them during their construction or for auxiliary estimates. One step along this line brought up a new robust incomplete decomposition RIF of symmetric and positive definite matrices [9]. In this case, the triangular factor is computed directly from a factorized approximate inverse. Computation of this approximate factorized inverse is breakdown-free for all SPD matrices. Progress in solving difficult problems [12], [43] also indicates that considering matrix inverse during the factorization may be a right way to get better preconditioners. In particular, the authors in these papers show that computing estimates of factor inverses can significantly increase robustness of LU factorization.

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) [16]. There is a subtle relation between the AISM algorithm and the AINV decomposition [6] on a larger matrix [13], but we will 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. Note that even for SPD matrices, one of the factors is generally square. Although we present the insight for decomposition of general nonsymmetric matrices, later we restrict just to SPD matrices. In this case we obtain via the AISM algorithm, 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 make use the dropping rules introduced by M. Bollhöfer and Y. Saad for LU factorizations and based on the theory developed by them [12], [43].

Section 2 gives the insight into the inverse Sherman-Morrisson (ISM) decomposition and shows structure of the resulting factors. Section 3 deals with some theory which may be useful for stabilization of the approximate ISM (AISM) decomposition in general case. Section 4 presents the main outcome of the paper: the way to stabilize the direct triangular factor obtained via the AISM decomposition, by balancing dropping rules. Then we present results of the numerical experiments showing very promising behaviour of the new approach, 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 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 [16] 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 $D_{A_0} = \operatorname{diag}(r_1, \ldots, r_n)$, $r_k = 1 + y_k^T A_0^{-1} u_k = 1 + v_k^T A_0^{-1} x_k$ 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 entry 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 [16] introduces an auxiliary unit upper triangular matrix W which helps to provide a better insight into the ISM decomposition.

LEMMA 2.1. *[16] Let U_s, V_s and $D_s = \operatorname{diag}(r_1^s, \ldots, r_n^s)$ be the matrices computed by the exact factorization algorithm ISM for some $s > 0$. Let U, V and $D = \operatorname{diag}(r_1^1, \ldots, r_n^1)$ 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 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 let the LDU decomposition of A be written as $A = \bar{L}\bar{D}\bar{U}$. Moreover, let W be defined as in (2.5). Then*

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

Moreover,

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

Proof. Note that from (2.8) we have

$$A = W^{-T}DU^{-1}$$

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

$$V_s = \begin{bmatrix} \ddots & & -sW^T \equiv -s\bar{L}^{-T} \\ & \ddots & \\ \bar{U}^T\bar{D} & & \ddots \end{bmatrix}. \tag{2.13}$$

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 has 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 [16], [19]. Such results are parallel to those related to the existence of ILU, see [40] and [39]. Although the main line of this paper covers stabilization of a special case of the algorithm by sophisticated dropping rules, we are 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 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). $\square$

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 the both triangular factors* which we will call the *scaling rule* provides useful value of s. We intended to point this fact out but we will not follow its implications here. 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 approximated 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

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

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 different dropping. We typically need to use stronger modifications. Nevertheless, based on the strong connection between ILU and AISM decompositions, we can proceed similarly as in [39] 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. A contemporary use of this iterative strategy for the ILU decomposition one can find in [37]. 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 SPD problems taking into account that extension to the nonsymmetric problems can be done along the same lines as the symmetric AINV decomposition was extended to its nonsymmetric counterpart in [6]. Our restriction is motivated by the desire to describe one particular preconditioning approach more in detail.

Balanced incomplete factorization (BIF) $\bar{L}\bar{D}\bar{L}^T$ of an SPD matrix is called the algorithm to construct incompletely the lower triangular matrix $\bar{L}$ stored in the matrix V_s and diagonal matrix $\bar{D}$ from (2.13) using the formulas (2.1) for $k = 1, \ldots, n$. Incompleteness is controlled by the dropping rules described below which mutually balance direct and inverse triangular factors $\bar{L}$ and $\bar{L}^{-1}$. Note that the scaled entries of U are contained in V_s and can be retrieved from there.

Let us describe our balancing dual dropping rules for decomposition which help us to produce high quality incomplete factors.

Relations between direct factors from LU decomposition and inverse factors from sparse factorized inverses were studied by M. Bollhöfer and Y. Saad in [14]. Robust dropping rules based on the resulting analysis were used in [11] and [12]. Suppose we have an incomplete decomposition $A = \bar{L}\bar{D}\bar{L}^T$, and we intend to apply it as a preconditioner of a Krylov space method. It is well known, that an important role in the preconditioned method is played by the transformed matrix which can be written as $\bar{L}^{-1}A\bar{L}^{-T}$. Consider dropping by value and denote the drop tolerance by τ. In [14] it is justified to drop the entries $\bar{l}_{jk}$ of a column k of $\bar{L}$ if they satisfy

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

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

The BIF algorithm of an SPD matrix computes at the same time, in addition to $\bar{D}$, the incomplete $\bar{L}$ and incomplete $\bar{L}^{-1}$. Theorem 2.2 enables us to apply the dropping rule (4.1) directly if we consider the BIF algorithm as a way to get $\bar{L}$. Consider for a moment, that our goal is to use also $\bar{L}^{-1}$. Denote $\ell_{jk} = (L^{-1})_{jk}$. Then (4.1) applied to direct computation of the inverse factor $\bar{L}^{-1}$ reads as that we drop entries ℓ_{jk} of the column of k of $\bar{L}^{-1}$ if they satisfy

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

In order to apply the rules (4.1) and (4.2) we need to consider the implementation more in detail. Consider the left-looking implementation of formulas (2.1). That is, the implementation which in each step computes one column of U and one column of V_s. In particular, in the $k-$th step we compute one diagonal entry, the $k-$th row of $\bar{L}^{-1}$ and the scaled $k-$th column of $\bar{L}$. Moreover, all these items are stored in V_s. Using Corollary 2.4 we can see that (4.1) for dropping entries in $v_{k:n,k}$ for a fixed column index k reduces to dropping the entries v_{jk} for $j > k$ such that

$$|v_{jk}| \parallel e_k^T v_{1:k,k} \parallel \leq s d_k \tau.$$

The rule for dropping entries in the k-th row of $\bar{L}^{-1}$ then reduces to dropping those entries v_{jk} with $j < k$ which satisfy

$$v_{jk} \parallel e_j^T v_{k,1:j} \operatorname{diag}(d_1, \ldots, d_j)^{-1} \parallel \leq s\tau.$$

Clearly, both rules can be directly applied since they use only information previously computed, and the algorithm obtained will be referred as BIF.

The coupled computation of direct and inverse factors explains the accumulated experience with AISM that U and V_s may profit from different dropping tolerance used for them since they store mathematically rather different quantities. In addition, the computation also points out that different drop tolerances should be used for different parts of columns of V_s, at least, if more simple 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 obtained by RIF [9] we can observe important differences. RIF provides in one step different quantities than BIF. In RIF, an approximate row of $\bar{L}^{-1}$ and an approximate row of $\bar{L}$ are evaluated in each step. But the flow of information is only one-directional. Underlying SAINV process is primary, RIF is a side-product based on multipliers for SAINV updates [9]. In contrast, computation of direct and inverse factors in BIF is *coupled*. Both factors $\bar{L}^{-1}$ and $\bar{L}$ use information from each other during the whole computation. Thus, if a good quality of one of the sparse factors happens, this may positively influence the decomposition. Might be, this observation lead to further the use of BIF in applications, where the system matrix sparsity seems to be critical, as in solving least-squares problems [10].

In the following section we will present a couple of experimental results showing great potential of the new algorithm.

5. Numerical experiments.

This section is devoted to numerical experiments with the new factorization which is used as a preconditioner of the conjugate gradient 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, it can be considered as an improvement of 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

TABLE 1
Test problems

Matrix	n	nnz	Application	Source
BCSSTK35	30,237	1,450,163	Automobile seat frame	U. of Florida [22]
VANBODY	47,072	1,191,985	Van body model	The PARASOL project
CT20STIF	52,329	1,375,396	Engine block	U. of Florida [22]
CFD1	70,656	949,510	CFD pressure matrix	U. of Florida [22]
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 [22]
ENGINE	143,571	2,424,822	Engine head	R. Kouhia [36]
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

used in our comparison is the RIF preconditioner, which may be considered as one of the methods of choice among robust approaches [9]. 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. In some sense, our experiments present a focused extension of that considered in [9].

Our implementation of BIF is left-looking. In this implementation we compute in each step one column of the matrix V_s. In order to have fully sparse implementation we need to have access to both columns and rows of V_s. The matrix V_s is formed and stored by columns in a standard way, but we store, in addition, at most $lsize$ largest entries for each row of V_s. This additional space is introduced for fast sparse computation of dot products. Note that then the row and column structure of V_s do not necessarily need to correspond each other. In all our experiments we chose $lsize = 10$. To our surprise, the results were nearly insensitive to the choice of $lsize$. Note that this fact is in contrast to the use of dual dropping directly in the AINV decomposition [6], where we typically get very different sizes of the factor rows. Uniform bound $lsize$ on their number 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 conforms to our note that we did not optimize 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 seem to be quite ill-conditioned. Note that we have considered here some test problems used in [9] and extended this choice by larger problems. For each matrix we provide the problem size n, the number nnz 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 [28].

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

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
HOOD	666	27.7
INLINE_1	764	101.
LDOOR	810	145.

TABLE 3
Comparison of the RIF and BIF preconditioners

Matrix	RIF		BIF	
	size / *t_p*	*CG_its*/ *t_CG*	*size* / *t_p*	*CG_its*/ *t_CG*
BCSSTK35	262120/ 0.50	60/ 0.47	255292/0.16	45/ 0.38
VANBODY	80576/ 0.75	8/ 0.09	82230/0.08	7/ 0.08
CT20STIF	63716/ 1.00	62/ 0.69	66172/0.06	61/ 0.67
CFD1	700523/11.5	287/ 4.08	734801/0.83	291/ 4.38
OILPAN	139959/ 1.20	134/ 1.92	116282/0.17	141/ 1.97
X104	†	†	1225831/0.88	44/ 1.89
CFD2	810560/ 5.40	372/ 8.40	889647/0.62	389/ 8.55
ENGINE	145570/ 1.48	213/ 4.95	148131/0.25	208/ 4.88
HOOD	368729/ 3.36	320/16.1	339062/0.60	319/16.3
INLINE_1	651393/16.4	149/22.4	662897/1.65	156/23.3
LDOOR	1229157/13.9	167/34.9	1277133/2.27	168/36.0

$b = Ae$, 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 preconditioner. For each method we report the number of nonzeros in the incomplete L factor (*size*), the time for constructing the preconditioner (*t_p*), the number of PCG iterations (*CG_its*) and the time for the iterative solution phase (*t_CG*).

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 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

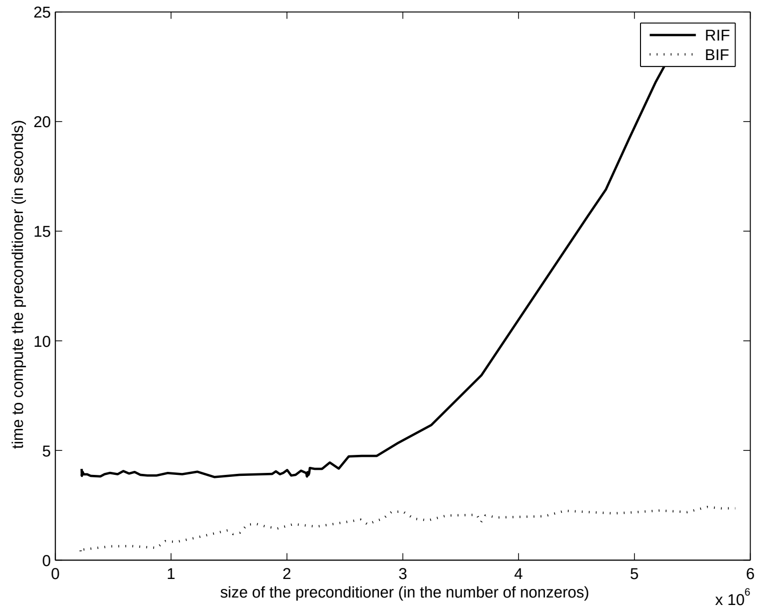

FIG. 5.1. *Sizes of RIF and BIF preconditioners (in numbers of their nonzeros) versus time to construct them (in seconds).*

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 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, and 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 computation. This intermediate memory is typically larger than the additional memory needed for the new approach.

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 a matrix PWTK (stiffness matrix from a pressurized wind tunnel, from the University of Florida collection [22]). Its dimension is 217,918, and it has 5,926,171 nonzeros in its lower triangular part. To solve the problem, JCG did 1178 iterations in 44.8 seconds. This matrix was, for example, used to show behavior of the LDL^T direct solver in [21].

Figure 5.1 shows the time to compute the preconditioners of different sizes. Clearly, the setup time is much smaller for all sizes of preconditioners. While we were able to choose parameters for RIF to make it even larger (following the increase

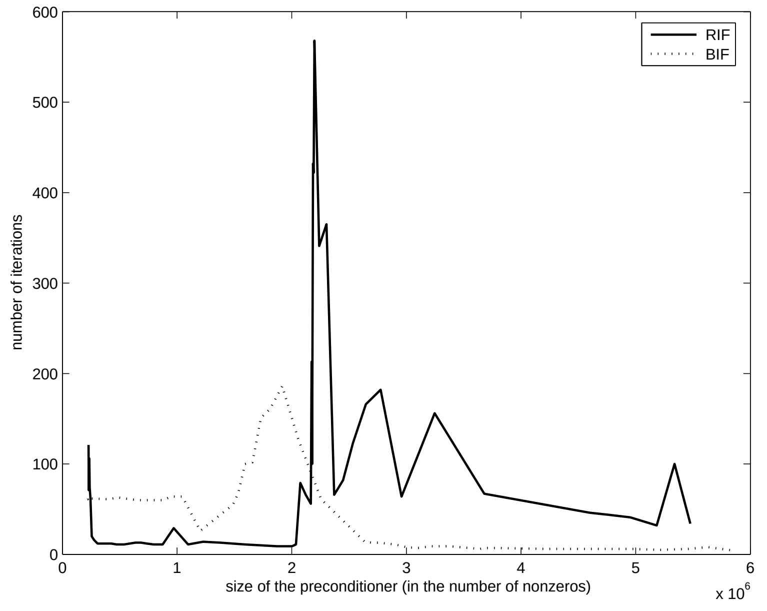

FIG. 5.2. *Sizes of RIF and BIF preconditioners (in numbers of their nonzeros) versus number of iterations for the conjugate gradient method preconditioned by them.*

in the setup time) this was not the case of BIF. Its density is naturally limited by the size of the row structures of V_s on one side, and by the dropping rules based on the norms of rows of $\hat{L}$ and $\hat{L}^{-1}$ on the other side.

Figure 5.2 shows the dependence of the number of iterations of the conjugate gradient 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 is to point out that the new approach is practically robust for solving difficult problems for which direct methods take considerable amount of time. BIF does this, even when not proved to be breakdown-free. We believe that this is due

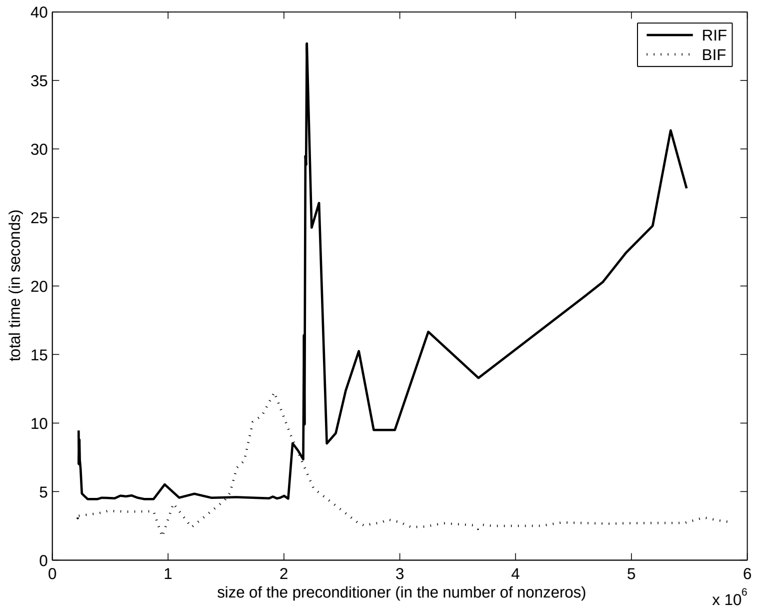

FIG. 5.3. *Sizes of RIF and BIF preconditioners (in numbers of their nonzeros) versus the total time (the time to compute the preconditioner plus the time for the conjugate gradient method).*

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. [10]).

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 algorithm of this new factorization closely couples computation of both the factors L and 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 L and 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.

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. C. Haws, and M. Tůma. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM J. Sci. Comput.*, 22(4):1333–1353, 2000.

[5] 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.

[6] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19(3):968–994, 1998.

[7] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math.*, 30(2-3):305–340, 1999.

[8] M. Benzi and M. Tůma. Orderings for factorized sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21(5):1851–1868, 2000.

[9] M. Benzi and M. Tůma. A robust incomplete factorization preconditioner for positive definite matrices. *Numer. Linear Algebra Appl.*, 10(5-6):385–400, 2003.

[10] 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.

[11] M. Bollhöfer. A robust ILU with pivoting based on monitoring the growth of the inverse factors. *Linear Algebra Appl.*, 338:201–218, 2001.

[12] 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.

[13] M. Bollhöfer. personal communication, 2004.

[14] M. Bollhöfer and Y. Saad. On the relations between ILUs and factored approximate inverses. *SIAM J. Matrix Anal. Appl.*, 24(1):219–237, 2002.

[15] R. Bridson and W.-P. Tang. Ordering, anisotropy, and factored sparse approximate inverses. *SIAM J. Sci. Comput.*, 21(3):867–882, 1999.

[16] 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.

[17] N. I. Buleev. A numerical method for solving two-dimensional diffusion equations. *Atomnaja Energija*, 6:338–340, 1959.

[18] N. I. Buleev. A numerical method for solving two-dimensional and three-dimensional diffusion equations. *Matematičeskij Sbornik*, 51:227–238, 1960.

[19] 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.

[20] T. F. Chan and H. A. van der Vorst. An explicit preconditioner for the conjugate gradient method. 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.

[21] T. A. Davis. Algorithm 849: a concise sparse Cholesky factorization package. *ACM Trans. Math. Software*, 31(4):587–591, 2005.

[22] T. A. Davis. *University of Florida Sparse Matrix Collection.* available online at http://www.cise.ufl.edu/∼davis/sparse/, NA Digest, vol. 94, issue 42, October 1994.

[23] 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.

[24] 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.

[25] 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.

[26] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29:635–657, 1989.

[27] T. Dupont, R. P. Kendall, and H. H. Jr. Rachford. An approximate factorization procedure for solving self-adjoint elliptic difference equations. *SIAM J. Numer. Anal.*, 5:559–573, 1968.

[28] N. I. M. Gould, Y. Hu, and J. A. Scott. Complete results from a numerical evaluation of sparse direct solvers for the solution of large, sparse, symmetric linear systems of equations. Numerical Analysis Internal Report 2005-1, Rutherford Appleton Laboratory, 2005.

[29] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM*

J. Sci. Comput., 18(3):838–853, 1997.
[30] I. Gustafsson. A class of first order factorization methods. *BIT*, 18(2):142–156, 1978.
[31] O. G. Johnson, C. A. Micchelli, and G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM J. Numer. Anal.*, 20(2):362–376, 1983.
[32] M. T. Jones and P. E. Plassmann. An improved incomplete Cholesky factorization. *ACM Trans. Math. Software*, 21(1):5–17, 1995.
[33] 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.
[34] 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.
[35] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings. I. Theory. *SIAM J. Matrix Anal. Appl.*, 14(1):45–58, 1993.
[36] R. Kouhia. *Sparse matrices web page.* http://www.hut.fi/~kouhia/sparse.html, available online, 2001.
[37] 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.
[38] 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.
[39] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comp.*, 34:473–497, 1980.
[40] 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.
[41] Y. Saad. ILUT: a dual threshold incomplete LU factorization. *Numer. Linear Algebra Appl.*, 1(4):387–402, 1994.
[42] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, 1996.
[43] 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.
[44] M. Tismenetsky. A new preconditioning technique for solving large sparse linear systems. *Lin. Alg. Appl.*, 154–156:331–353, 1991.
[45] R. S. Varga. Factorizations and normalized iterative methods. In *Boundary problems in differential equations,*, pages 121–142, Madison, WI, 1960. University of Wisconsin Press.