

PRECONDITIONED ITERATIVE METHODS FOR SOLVING LINEAR LEAST SQUARES PROBLEMS*

RAFAEL BRU[†], JOSÉ MARÍN[†], JOSÉ MAS[†], AND MIROSLAV TŮMA[‡]

Abstract. New preconditioning strategies for solving $m \times n$ overdetermined large and sparse linear least squares problems using the conjugate gradient for least squares (CGLS) method are described. First, direct preconditioning of the normal equations by the balanced incomplete factorization (BIF) for symmetric and positive definite matrices is studied, and a new breakdown-free strategy is proposed. Preconditioning based on the incomplete LU factors of an $n \times n$ submatrix of the system matrix is our second approach. A new way to find this submatrix based on a specific weighted transversal problem is proposed. Numerical experiments demonstrate different algebraic and implementational features of the new approaches and put them into the context of current progress in preconditioning of CGLS. It is shown, in particular, that the robustness demonstrated earlier by the BIF preconditioning strategy transfers into the linear least squares solvers and the use of the weighted transversal helps to improve the LU-based approach.

Key words. preconditioned iterative methods, incomplete decompositions, approximate inverses, linear least squares

AMS subject classifications. Primary, 65F20, 65F50; Secondary, 15A06, 15A23

DOI. 10.1137/130931588

1. Introduction. Linear least-squares (LS) problems

$$(1.1) \quad \min_x \|b - Ax\|_2,$$

where $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) is a large and sparse matrix with full column rank, can be solved iteratively using the conjugate gradient for least squares (CGLS) method [14], which implicitly applies the conjugate gradient method to the normal equations

$$(1.2) \quad A^T Ax = A^T b.$$

For solving either large sparse systems of linear algebraic equations or LS problems, iterative methods may be preferred because they often require much less storage than their direct counterparts. Their successful application often needs a good preconditioner in order to achieve fast convergence rates. In particular, for systems of linear equations arising from discretizations of three-dimensional boundary value problems, the advantages of preconditioned iterative methods are clear and well documented in the literature. However, less knowledge about their benefits for LS problems is available for various reasons. LS problems arising from various sources may differ very much and may need differently preconditioned iterative methods. In other words, the problem of robust and efficient iterative solutions of LS problems is much harder than the iterative solution of systems of linear equations. This fact is implicitly underlined

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 2, 2013; accepted for publication (in revised form) June 18, 2014; published electronically August 19, 2014. This work was partially supported by Spanish grant MTM 2010-18674 and the project 13-06684S of the Grant agency of the Czech Republic.

<http://www.siam.org/journals/sisc/36-4/93158.html>

[†]Institut 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).

in the literature, where experiments with various strategies are often tightly restricted to classes of problems that arise in such distant areas of engineering and applied research as signal and control processing, statistics, geodesy, etc. An excellent source on general and specialized solution strategies with a comprehensive treatment of various applications is still the book [14].

In principal, three main classes of general-purpose preconditioning approaches for solving problem (1.1) were proposed, studied, and subsequently enhanced by improvements from various authors. The most traditional approach is based on the normal equations and a factorization of the symmetric positive definite matrix $A^T A$. Sources of the direct variant of this approach date back to [5]. In our context, the preconditioner can be based on an *incomplete* factorization of $A^T A$. A recent strategy of this type that also uses implicit decomposition of the inverse of $A^T A$ was recently introduced in [12]. Another type of preconditioner based on approximate inverses was recently studied in a series of papers [27, 28, 29] having a starting point in the Ph.D. thesis [26].

A lot of attention has been devoted to algorithms based on the incomplete QR decomposition of A . This approach that provides a preconditioner for CGLS represents the second main source of preconditioning strategies, but there are other closely related approaches less frequently used; see, e.g., [13]. An important theoretical feature of the QR-based approach without dropping is that we obtain $Q^T Q = I$. In practice, the QR factorization causes a significant fill-in (for a recent source of many references, see [30]), and finding a *useful* sparse incomplete QR decomposition is inherently difficult. Partly, it may be because modification and compensation strategies of the incomplete QR decomposition are much less understood than analogous techniques for the incomplete symmetric and positive definite (SPD) factorization; see, e.g., [1, 2, 8, 37, 63]. Finding appropriate tools to modify the structure or factors for the incomplete QR decomposition, whose power was shown in [52], turns out to be an important open problem in this field. An interesting contribution that discusses both theoretical and practical aspects of the Gram–Schmidt-based incomplete LQ decomposition as well as other preconditioning techniques is given in [56]. For the first attempts to develop a drop tolerance based incomplete orthogonalization by Givens rotations, see [68] and the detailed overview [67], and also the incomplete orthogonal decomposition based on static sparsity patterns in [47]. A recent theoretical overview of the incomplete QR strategies based on Givens rotations is given in [3] and in the subsequent paper [54].

One specific approach that uses an incomplete QR decomposition based on Gram–Schmidt orthogonalization with no dropping in Q is developed in [48]. This method is studied in [64] and [65], where it is shown that this approach can be also described as a specific robust incomplete Cholesky (IC) decomposition, called Cholesky incomplete modified Gram–Schmidt (CIMGS). Without going into details, its main idea can be easily described considering the decomposition as a sequence of modifications of Schur complements. If the decomposition is performed incompletely, the modifications are *semidefinite*, and it is breakdown-free. Looking carefully at CIMGS in [65], we get that, see also [6], the CIMGS method applied to the matrix $B = A^T A$ is equivalent to the IC decomposition of B based on positive semidefinite updates proposed by Tismenetsky in [63]. The resulting method is known to be rather robust [65, 8], and this is the reason that we mention it here as a representative of the incomplete QR approaches.

The third main alternative is the LU-based approach. It was introduced in [51] as a direct solution method. It consists of partitioning the system matrix A with

permuted rows as

$$PA = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix},$$

where P denotes the row permutation matrix. The LS problem is then transformed by multiplying A by A_1^{-1} from the right. The corresponding normal matrix is then $I_n + (A_2 A_1^{-1})^T A_2 A_1^{-1}$, which should be easy to solve especially when $m - n \ll n$, since the above matrix is a low rank modification of the identity matrix. The transformed LS problem can then be solved either directly with stationary iterative methods such as SOR, or with a nonsymmetric Krylov space iterative method where A_1^{-1} serves as a preconditioner. Further, A_1^{-1} can be used as a preconditioner for the normal equations. Note that if a complete decomposition of A_1 with well-conditioned L is available, then the Peters–Wilkinson method may be the method of choice [55] for solving both dense and sparse problems.

In this paper we further develop both the first and the third approaches, i.e., the incomplete decomposition to solve the normal equations (1.2) as well as the LU-based technique mentioned above, but our comparison covers also the incomplete QR decomposition CIMGS. In the first approach treated here, namely, the incomplete decomposition for the normal equations, we deal with the balanced incomplete factorization (BIF) preconditioner for symmetric and positive definite matrices from [20]; see also [19]. In particular, we present a new breakdown-free algorithm for symmetric and positive definite matrices and demonstrate that this strategy is a useful choice for solving LS problems because of its robustness.

The ILU approach has been slightly overlooked if we consider just the papers published in recent years. An explanation may be the lack of attention devoted to the choice of the submatrix A_1 used for this type of preconditioning. Our proposal is exactly in this direction. The incomplete LU decomposition is based on the generally known ILUT algorithm [57], although we have also developed the BIF preconditioner for solving nonsymmetric problems that was shown to be rather robust in [21]. The core of the strategy is the matrix reordering that chooses the leading submatrix A_1 of A . The reordering is based on an algorithm for finding the weighted transversal (matching) using the sparsity structure and magnitudes of entries of A . Such an approach was introduced and motivated for square matrices in [53], and it was efficiently implemented for nonsymmetric systems of linear equations in [33] and [34]; see also [7]. Although the reordering entails an additional small overhead and may result in instabilities for very ill-conditioned problems [14, 40], the results of numerical experiments in Tables 2 and 3 show that it sometimes leads to better performance than that of the approach based on the normal equations.

The paper is organized as follows. In section 2 we present the enhanced BIF algorithm accompanied by some theoretical results and propose its application to computing the incomplete factorization of the matrix $A^T A$. In section 3 we describe the LU-based approach and the new reordering technique used to find the submatrix A_1 . In section 4 we show the results of the numerical experiments. Section 5 outlines the main conclusions.

2. The BIF for the normal equations. As mentioned, there is a long list of previous attempts to precondition the normal equations with various types of IC factorizations. The preconditioning strategy we propose here uses the LDL^T decomposition arising from the (shifted) $(I - (A^T A)^{-1})^{-1}$ -biconjugation applied to the normal equations [20]; see equation (4) in [25]. Our approach, in contrast to standard

IC decompositions, computes also an approximation of the inverse Cholesky factor of $A^T A$. It is well known that because of the larger conditioning of $A^T A$, some standard approaches become less stable [14]. One possibility to obtain a robust decomposition of a matrix is to exploit properties of its inverse. This strategy was shown to be successful for standard incomplete LU decompositions of nonsymmetric matrices [16, 17, 18] when the estimates related to the inverse of such matrices are used to monitor and control dropping of the decomposition. A different algorithm that also uses information from the inverse is the robust incomplete factorization (RIF) introduced in [11] that has been also been applied to solve LS problems in [12]. Recent work [26], see also [29], proposed $A^T A$ -biconjugation, which uses L^{-1} , the inverse of the L factor of the Cholesky factorization, as an auxiliary intermediate quantity. It was derived in an interesting way starting with explicit expressions for the Moore–Penrose pseudoinverse originally proposed by Greville [42] instead of applying the approximate inverse preconditioner (AINV) directly to $A^T A$ [9]. It is worth noting that Greville’s method for computing the pseudoinverse can also be derived from a general framework given in [23, Theorem 3.1.3], but we do not give the details since this connection is not needed in this work.

Our method is based on a biconjugation process that allows the computation of a decomposition of the inverse of a given matrix; see [19]. When applied to a symmetric positive definite matrix $B = A^T A$, it computes the factors Z , D_s , and V satisfying

$$(2.1) \quad s^{-1}I - B^{-1} = s^{-2}ZD_s^{-1}V^T,$$

where $s > 0$ is a given scalar. Algorithmically, the columns of Z and V are computed from

$$(2.2) \quad z_k = e_k - \sum_{i=1}^{k-1} \frac{v_i^T e_k}{sr_i} z_i \quad \text{and} \quad v_k = y_k - \sum_{i=1}^{k-1} \frac{y_k^T z_i}{sr_i} v_i$$

for $k = 1, 2, \dots, n$, where e_k is k th unit vector, $y_k = (b^k - se^k)^T$, b^k denote the k th row of B , and $r_i = 1 + v_{ii}/s$ are the elements of the diagonal matrix D_s . It was proved in [20] that the decomposition $B = LDL^T$ and the decomposition (2.1) satisfy

$$Z = L^{-T}, \quad V = LD - sL^{-T}, \quad \text{and} \quad D_s = s^{-1}D.$$

Thus, the Cholesky factorization of B and its inverse can be computed simultaneously with the biconjugation process described. Furthermore, both the direct and the inverse factors are influenced by their counterparts since the computations can be coupled in two ways. The motivation that lies behind the process is to obtain more robust incomplete decompositions. This coupling was first performed in [20] via synchronized dropping in the approximations of L and L^{-1} based on the relations derived in [18]. Later on, the approach proposed in [21] extended this coupled decomposition to the nonsymmetric case and the resulting approach was again found rather robust and efficient. Both papers also contain algorithmic schemes and a description of the implementation that is based on sparse data structures and incompleteness by dropping.

To design an algebraic preconditioning algorithm to solve least squares problems, we have to consider additional constraints. First, note that the system matrix $B = A^T A$ of the normal equations is often significantly denser than A . This implies that aggressive dropping to keep the preconditioner sparse must be applied. As a consequence, the standard incomplete decompositions may become unstable and the convergence of the iterative method can fail. Also, incomplete decompositions that

are parametrized often need fine tuning of the parameters to avoid breakdown, in particular, for the normal equations. Summarizing this, devising robust preconditioning algorithms is even more crucial for solving the normal equations by preconditioned iterative methods. This is the reason why we propose the use of the BIF preconditioner which allows the coupled computation of both the direct and inverse factors of B , and a special care is devoted to computation of diagonal entries.

It was shown that the balanced incomplete decomposition is breakdown-free for H-matrices, but it may fail for symmetric and positive definite matrices. We present a new version for the symmetric positive definite matrices satisfying the breakdown-free property. It is obtained by reformulating the computation of pivots using the quadratic form defined by the symmetric and positive definite matrix B , and keeping the computation of both factors in V mutually dependent. The new compact scheme of the balanced incomplete decomposition is displayed in Algorithm 2.1 as a dense code without dropping.

Let us note that the columns of $Z = L^{-T}$ and the diagonal elements of D are contained in the upper triangular part of V ; see [20]. More precisely, the k th columns of Z and the upper part of V are related by $z_k = -[v_{1:k-1,k}^T/s, 0, \dots, 0]^T + e_k$, and the k th pivot is $d_k = v_{kk} + s$. This is the reason why Algorithm 2.1 is described only in terms of the matrix V using equation (2.8) of [2] and the equalities given below in the proof of Theorem 2.1.

ALGORITHM 2.1.

for $k = 1 : n$ **do**

$v_k = b_k - se_k$

for $i = 1 : k - 1$ **do**

$$v_{1:i-1,k} = v_{1:i-1,k} - \frac{b_k^T z_i}{z_i^T B z_i} v_{1:i-1,i}$$

$$v_{i,k} = s \frac{b_k^T z_i}{z_i^T B z_i}$$

$$v_{k:n,k} = v_{k:n,k} - \frac{b_k^T z_i}{z_i^T B z_i} v_{k:n,i}$$

end for

$$z_k = -[v_{1:k-1,k}^T/s, 0, \dots, 0]^T + e_k$$

end for

This algorithm computes the LDL^T decomposition of B as it is stated in the following theorem.

THEOREM 2.1. *Algorithm 2.1 obtains the matrix $V = LD - sL^{-T}$ where L and D are the Cholesky factors of $B = LDL^T$. Furthermore, the computation of V is breakdown-free for any dropping of off-diagonal entries in V .*

Proof. The only change in Algorithm 2.1 with respect to the BIF algorithm is the computation of the denominator in the equations. Since $Z = L^{-T}$ and z_i in Algorithm 2.1 is its i th column, it follows that

$$z_i^T B z_i = z_i^T LDL^T z_i = e_i^T D e_i = d_i = sr_i.$$

Then the computation of the columns of the matrix V by the algorithm is just (2.2), where the pivot d_i is computed in another way. Further, since B is symmetric and positive definite and $z_i \neq 0$, at least its i th entry is nonzero (equal to 1), we have that $z_i^T B z_i > 0$ even if dropping is applied. $\square$

To further improve the quality of the preconditioner, we introduce a modification following the ideas proposed by Tismenetsky [63] that we briefly describe here. This

approach assumes that each column l_k of the exact factor L is split as $l_k = \bar{l}_k + \hat{l}_k$ such that $\bar{l}_k$ and $\hat{l}_k$ have orthogonal patterns. Then the Tismenetsky algorithm decomposes in each step the matrix increased by $E = \hat{L}\hat{L}^T$, where E is an additional positive semidefinite local error matrix. E typically reflects the entries of L that are small in some sense and kept in $\hat{L}$. The same splitting as in the Tismenetsky decomposition can be applied also to BIF for computation of the lower triangular part of V . Assume that each lower triangular part $v_{k:n,k}$ of the column k of V is expressed as a direct sum $v_{k:n,k} = \bar{v}_{k:n,k} + \hat{v}_{k:n,k}$ such that $\bar{v}_{k:n,k}$ and $\hat{v}_{k:n,k}$ have orthogonal patterns. Theorem 2.2 formally describes the Tismenetsky decomposition based on the lower triangular part of V that is breakdown-free.

THEOREM 2.2. *Consider Algorithm 2.1 applied to a symmetric and positive definite matrix B with the computation of $v_{k:n,k}$ replaced by*

$$(2.3) \quad v_{k:n,k} = v_{k:n,k} - \sum_{\substack{i < k \\ \bar{v}_{ki} \neq 0}} \frac{\bar{v}_{ki}}{v_{ii} + s} (\bar{v}_{k:n,i} + \hat{v}_{k:n,i}) - \sum_{\substack{i < k \\ \hat{v}_{ki} \neq 0}} \frac{\hat{v}_{ki}}{v_{ii} + s} \bar{v}_{k:n,i}$$

for $k = 1, \dots, n$. Let $D = \text{diag}(V) + sI$ and let L be the unitary lower triangular matrix such that has the same strictly lower triangular part as VD^{-1} . Let $\hat{B} = LDL^T$ with $\hat{B} = B + E$ for some positive semidefinite matrix E . Then, the computation of V is breakdown-free.

Proof. In exact arithmetic, Algorithm 2.1 computes a matrix V such that $V = LD - sL^{-T}$. Equation (2.8) of [21] states $b_k^T z_i = l_{ki} d_i = v_{ki}$ and also $d_i = v_{ii} + s$. Then the modification in (2.3) is just the Tismenetsky algorithm for any choice of the splitting of vectors $v_k = \bar{v}_k + \hat{v}_k$ if the entry v_{kk} remains in $\bar{v}_k$. Then, by the proposition stated on page 336 of [63], the new algorithm is breakdown-free. $\square$

The exact update in (2.3) removes coupling of the computation of the direct and inverse factors in V that we consider to be important for the decomposition robustness. In our experiments, we use Algorithm 2.2, where the update (2.3) is reformulated in the sense of Theorem 2.1 and allows safe incomplete implementation. The coupling of the upper and lower triangular parts of V is restored via the bilinear form that makes the incomplete decomposition also breakdown-free.

ALGORITHM 2.2.

for $k = 1 : n$ **do**

$$v_k = b_k - s e_k$$

for $i = 1 : k - 1$ **do**

$$v_{1:i-1,k} = v_{1:i-1,k} - \frac{b_k^T z_i}{z_i^T B z_i} v_{1:i-1,i}$$

$$v_{i,k} = s \frac{b_k^T z_i}{z_i^T B z_i}$$

$$v_{k:n,k} = v_{k:n,k} - \sum_{\substack{i < k \\ \bar{v}_{ki} \neq 0}} \frac{\bar{v}_{ki}}{z_i^T B z_i} (\bar{v}_{k:n,i} + \hat{v}_{k:n,i}) - \sum_{\substack{i < k \\ \hat{v}_{ki} \neq 0}} \frac{\hat{v}_{ki}}{z_i^T B z_i} \bar{v}_{k:n,i}$$

end for

$$z_k = -[v_{1:k-1,k}^T / s, 0, \dots, 0]^T + e_k$$

Split $v_{k:n,k}$ in two vectors, $\bar{v}_{k:n,k}$ and $\hat{v}_{k:n,k}$ with orthogonal pattern such that

$$v_{k:n,k} = \bar{v}_{k:n,k} + \hat{v}_{k:n,k}$$

end for

3. LU preconditioning for least squares problems. The class of LU preconditioners for the least-squares problems based on splitting of A with permuted

rows

$$(3.1) \quad PA = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$$

mentioned in the introduction was thoroughly considered in [15]. It was noted there that such preconditioned iterative methods have very good convergence properties provided that the permutation matrix P is chosen such that the matrix AA_1^{-1} is well-conditioned. The resulting Krylov space method for the preconditioned system was deeply studied in [39]. It was also shown in [15] that a simple reformulation of the LU preconditioned algorithm can be easily adapted to solve generalized least-squares problems.

Our main goal here is to show that the LU preconditioner based on an incomplete decomposition can sometimes successfully compete with the other methods, as our experiments in section 4 show. The proposed strategy is based on finding a good reordering P of A such that A_1 has a stable incomplete decomposition $A_1 \approx L_1 U_1$ and $PA(L_1 U_1)^{-1}$ is well-conditioned. To find the reordering, we use the concept of maximum transversal of a matrix. Moreover, after computing the transversal we still keep the possibility to choose the pivot from the whole matrix A when the incomplete LU decomposition of A_1 is actually performed.

We recall that a transversal M of a $m \times n$ sparse matrix A is a subset of its nonzero entries such that no two of them are in the same row and also no two of them are in the same column. Being M , a transversal is a necessary and sufficient condition for permuting the subset M to the diagonal. M is called a *maximum transversal* if its cardinality is as large as possible. If A has full column rank, then all maximum transversals have n elements, thus obtaining a maximum transversal is a way to find a permutation P of A such that the n diagonal entries of PA are the nonzero entries of the transversal. An efficient implementation of the basic algorithm that finds the maximum transversal [46] was given in [36]; see also a recent survey [32]. The corresponding problem in graph theory is often called the maximum bipartite matching problem.

To enforce stronger diagonal dominance of the permuted matrix and therefore try to satisfy the two desirable properties of the splitting (3.1) stated above, additional constraints on the choice of the transversal M are imposed. Taking into account the magnitude of the entries for square nonsymmetric matrices, convert this problem into a variant of the weighted bipartite matching problem [33, 34, 53], also referred to as the assignment problem [22]. Experiments with preconditioned iterative methods can be found in [7]. Its application to symmetric and indefinite systems has been studied in [35, 43, 60]. Here we introduce a transversal-based reordering for rectangular matrices of size $m \times n$, $m \geq n$.

A well known strategy that has been used for square matrices considers the maximum transversal $M = \{a_{p(1),1}, \dots, a_{p(n),n}\}$, such that the product

$$(3.2) \quad \prod_{j=1}^n |a_{p(j),j}|$$

is maximized over all possible bijective maps $p : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$. Here we are interested mainly in generally rectangular matrices, but it is still worth noting that if A is a generalized diagonally dominant matrix (M- or H-matrix), then the maximum of the product (3.2) is attained by the transversal formed by the main diagonal entries, that is, no reordering is needed.

THEOREM 3.1.

(a) Let A be an H -matrix with nonsingular comparison matrix. Then

$$\prod_{j=1}^n |a_{jj}| \geq \prod_{j=1}^n |a_{p(j),j}| \quad \text{for any permutation } p \text{ of } \{1, \dots, n\}.$$

(b) If there is a permutation matrix P such that PA is an H -matrix, then the algorithm that finds a maximum transversal maximizing the product (3.2) provides a permutation matrix P' such that $P'A$ is an H -matrix.

Proof.

(a) Since A is an H -matrix there exists a diagonal matrix D with positive diagonal entries such that

$$|a_{jj}| d_j \geq \sum_{k \neq j} |a_{kj}| d_k.$$

Let p be an arbitrary permutation of the set $\{1, \dots, n\}$, different from the principal one. Then for each j , one has $|a_{jj}| d_j \geq |a_{p(j),j}| d_{p(j)}$. Then

$$\prod_{j=1}^n |a_{jj}| d_j \geq \prod_{j=1}^n |a_{p(j),j}| d_{p(j)}$$

and the result follows.

(b) It is an easy consequence of part (a), since the product of the absolute values of the diagonal entries of the H -matrix PA maximizes the product (3.2). $\square$

As is well known, the condition of being generalized strictly diagonally dominant guarantees the successful computation of many preconditioners. According to Theorem 3.1, the strategy of maximizing the product (3.2) reorders matrices such that they are in most cases more diagonally dominant.

In [33] it is proved that maximizing the product (3.2) is equivalent to minimizing the sum of weights

$$(3.3) \quad \sum_{j=1}^n |c_{p(j),j}|$$

defined as

$$(3.4) \quad c_{ij} = \begin{cases} \log \bar{a}_j - \log |a_{ij}|, & a_{ij} \neq 0, \\ \infty, & a_{ij} = 0, \end{cases}$$

where $\bar{a}_j = \max_i |a_{ij}|$ is the maximum magnitude of an entry of the j th column of A . To take into account other problem features such as the matrix sparsity, the definition of the weights in (3.4) may be modified.

However, in the rectangular case, $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, this definition of the weights c_{ij} may not be suitable for the following reason. To decompose the $n \times n$ leading submatrix A_1 of PA , the row dominance is more important than the column one, because we concentrate on obtaining a well-conditioned matrix A_1 and also in the stability of its incomplete LU decomposition. Observe that even in the case that A contains a submatrix that is an H -matrix, the above strategy does not guarantee finding that submatrix, as can be seen in Example 1.

Then, we redefine the weights as

$$(3.5) \quad c_{ij} = \begin{cases} \log \bar{a}_i - \log |a_{ij}|, & a_{ij} \neq 0, \\ \infty, & a_{ij} = 0, \end{cases}$$

where $\bar{a}_i = \max_j |a_{ij}|$ is the maximum magnitude of an entry of the i th row of A . The injective map $p : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ that minimizes the sum (3.3) is computed, and entries of the corresponding transversal are permuted to the diagonal of A to provide the permuted matrix PA . Now, the product of diagonal entries is not maximized over the set of all row permutations of A .

Example 1. Consider the following matrix:

$$\left(\begin{array}{c} A_1 \\ \hline A_2 \end{array} \right) = \left(\begin{array}{ccc} 1 & & \\ & 2 & \\ & & 3 \\ \hline 5 & 6 & 8 \\ 3 & 5 & 6 \\ 2 & 3 & 4 \end{array} \right).$$

Our method (3.5) leads to choosing the well-conditioned submatrix formed by the first three rows of A to form A_1 instead of choosing the last ones (with entries of larger magnitudes) as strategy (3.4) does. Clearly, A_1 is an M-matrix, and it is better conditioned and has more stable factorization than A_2 , which is not an M-matrix. Observe that AA_1^{-1} has condition number 8.29, which is smaller than the one of AA_2^{-1} , which is 13.59.

Moreover, since our goal is to compute an incomplete factorization of the chosen submatrix, sparsity is another point to be considered. We propose to modify the weights (3.5) as

$$(3.6) \quad \bar{c}_{ij} = \theta c_{ij} + (1 - \theta) \left(\frac{r_i}{r_{\max}} \right) c_{ij},$$

where r_i is number of nonzero entries in row i , $r_{\max} = \max_i r_i$ and $0 \leq \theta \leq 1$. For values of θ greater than 0.5, the magnitude of the matrix entries is emphasized.

4. Numerical experiments. In this section we present the results of some numerical experiments aimed at assessing the performance of the BIF preconditioner for least squares problems and the new approach to find a useful splitting (3.1) for the LU-based preconditioner. We also compare them with some other well known methods. All codes developed for the tests were written in FORTRAN 95 and have been compiled with Intel Fortran Composer XE 2013. For the experiments we used one processor of an Intel Core2 Q6700 (2.66GHz, 4GB RAM).

The number of rows m , number of columns n together with a short description of each matrix source are summarized in Table 1. All matrices can be found in the Tim Davis collection of sparse matrices [31]. Note that because of typographical reasons, the full matrix names were in Tables 2 and 3 abbreviated. With respect to the matrices representing constraints for the simplex method of linear programming, we note that they proceed from specific types of problems, see [4], and they may be difficult to use especially for combinatorial preprocessings since their nonzero entries are often chosen from a very restricted set of numerical values. Note that these matrices had to be transposed in order to have $m > n$, and in some cases they were automatically regularized by removing linearly dependent columns. Regularization by removing their last column was also needed for the popular animal breeding matrices, although we have not found anywhere in the literature that these matrices do not have full column rank.

The stopping criterion used for CGLS was based on the backward error. Namely, we stopped the iterations as soon as the residual 2-norm of the k th approximation of

TABLE 1
Test problems.

Matrix	m	n	nz	Application
S	3,140	1,987	8,510	animal breeding
PHOTOGRAMMETRY2	4,472	936	37,056	photogrammetry
M	9,397	6,119	25,013	animal breeding
LP_MAROS_R7	9,408	3,136	144,848	linear programming
LP_DFL001	12,230	6,071	35,632	linear programming
STORMG2-27	14,441	37,485	94,274	linear programming
TESTBIG	17,613	31,223	61,639	linear programming
LP_OSA_07	25,067	1,118	144,812	linear programming
L	28,254	17,263	75,018	animal breeding
KEMELMACHER	28,452	9,693	100,875	computer graphics/vision
LP_OSA_14	54,797	2,337	317,097	linear programming
DELTAX	68,600	21,961	247,424	example from S. Toledo
LANDMARK	71,952	2,704	1,151,232	problem from V. Pereyra
LP_OSA_30	104,374	4,350	604,488	linear programming
LP_KEN_18	154,699	105,127	358,171	linear programming
MESH_DEFORM	234,023	9,393	853,829	image mesh deformation
IMAGE_INTERP	240,000	120,000	711,683	image editing problem
SLS	1,748,122	62,729	6,804,304	statistics

the solution vector was smaller than $\alpha(\|A\|_2\|x_k\|_2 + \|b\|_2)$. Note that this is only a sufficient condition for convergence, see, e.g., the discussion in [24]. The constant α was set to 10^{-6} for all problems except for the matrix LANDMARK for which 10^{-4} was used because of its lower final attainable accuracy. In all cases the initial guess was $x_0 = 0$, and the right-hand side b was chosen so that the solution was the vector of all 1's.

Before showing and discussing the results of experiments, let us describe the way we present them. Arranging the results for more matrices and methods into tables is often useful, but it may not reveal some visible and important differences among the considered approaches. In particular, robustness of the methods is not easily described by tables. This is the reason why we provide, for some matrices, graphs containing a more detailed view visualizing some characteristic features of the methods that would otherwise stay hidden.

Tables 2 and 3 contain results for CGLS preconditioned by the four different incomplete decompositions of $B = A^T A$: BIF, IC, AINV, and CIMGS. The implementation of the new breakdown-free BIF algorithm for the normal equations, summarized in section 2, is based on [20]. In addition, in each step of the decomposition, we computed the diagonal entries of the preconditioner from the bilinear form in a straightforward way by an additional matrix-vector product and a dot product. The BIF shift parameter s was fixed to 1 as in [20], and data structure for rows of the matrix V_s stored at most $lsize = 10$ of the largest nonzero magnitudes. Further, additional size of the intermediate memory used for each $\hat{v}_k$ was fixed to $lsize/2$. IC denotes the left-looking implementation of a standard drop-tolerance based IC decomposition that was used in [12] to evaluate the RIF algorithm. AINV is the factorized approximate inverse preconditioner [9]. Its right-looking implementation is used since the computational complexity for sparse preconditioners should depend less on the matrix $A^T A$, which may be rather dense. Thus, the AINV computation depends more on the sparsity of the computed approximate factor of the inverse of $A^T A$. Finally, we consider CIMGS, which is the incomplete QR factorization in which Q is exact and only the factor R is computed incompletely. For the sake of efficiency, we used for CIMGS the left-looking implementation proposed by Kaporin in [49], which uses

TABLE 2
Sizes and iteration counts of the BIF, IC, AINV, and CIMGS preconditioners.

Matrix	BIF		IC		AINV		CIMGS	
	size	its	size	its	size	its	size	its
S	6,538 49,413	101 96	10,741 41,550	108 12	6,668 54,526	93 25	6,785 52,144	613 7
PHOTO2	1,283 39,427	90 84	1,385 39,076	73 133	1,359 36,657	112 857	1,303 59,578	‡ 38
M	6,239 141,982	182 142	7,633 94,847	176 17	7,408 93,856	174 42	6,897 113,736	‡ 249
MAR_R7	18,469 317,000	9 5	18,642 339,081	8 2	18,645 298,517	7 4	21,111 342,635	71 2
DFL001	11,835 236,996	254 140	10,018 299,992	307 471	11,460 392,724	258 485	20,200 1,906,518	‡ 16
STRG2-27	26,671 86,370	327 174	26,824 90,160	149 221	32,685 1,995,888	‡ ‡	32,232 7,728,742	‡ ‡
TESTBIG	63,777 139,117	80 57	78,391 170,825	23 16	50,457 145,324	97 ‡	103,974 2,039,655	‡ ‡
OSA_07	1,969 2,057	19 19	2,162 3,515	29 42	2,824 4,533	88 55	3,363 55,288	256 11
L	40,289 145,093	167 167	21,802 208,634	196 26	46,237 213,759	153 55	30,908 626,399	‡ ‡
KEMELM	72,649 264,143	193 193	83,702 282,019	‡ ‡	76,560 338,038	‡ ‡	71,461 122,725	116 63
OSA_14	2,873 3,276	46 19	2,740 7,156	25 45	4,531 7,323	37 71	4,587 117,933	227 16
DELTAX	172,992 344,066	146 118	165,613 318,554	161 137	180,488 314,330	150 128	165,432 335,724	‡ ‡
LMARK	24,675 105,577	51 26	28,636 122,692	10 ‡	4,856 341,544	‡ ‡	24,337 98,334	‡ 2
OSA_30	4,954 5,289	30 20	4,773 7,652	27 32	4,354 8,501	39 33	22,781 217,051	215 20
KEN_18	198,210 1,013,707	469 438	423,006 1,662,023	396 429	128,779 253,550	316 354	235,711 1,003,559	‡ 674
MESH_DF	119,070 241,574	115 105	118,555 235,124	‡ 5	12,297 43,712	182 279	137,227 261,049	384 47
IMG_IN	1,174,032 1,575,791	15 10	1,241,194 1,352,400	‡ ‡	‡ ‡	‡ ‡	1,077,003 1,541,568	133 70
SLS	66,605 80,444	53 25	65,355 71,462	60 47	62,729 69,838	79 55	9,547,789 9,795,156	‡ 36

two drop tolerances. The smaller one is fixed to 10^{-4} , and it simply means that the nonzero fill-in entries smaller than this value are used only to modify the diagonal of the preconditioner. The larger drop tolerance controls the size of the preconditioner.

Table 2 reports the number of nonzeros in the incomplete factor (under size) and number of preconditioned CGLS iterations (under its). The complementary Table 3 reports the time to construct the preconditioner (under t_p) and the time for the iterative solution phase (under t_{it}). In boldface, the best achieved timings (adding time for the decomposition and for the preconditioned CGLS) for each problem are highlighted although in some cases the differences are very small. For each matrix we report two experiments. One corresponds to a very sparse preconditioner and the other to a denser one. We always intend to couple the number of iterations with the size of the preconditioner; see also the concept of efficiency in such comparisons introduced in [61] that couples the size of the preconditioner and iteration count together into a common quantity. Therefore, this way of presenting the results is different than the approach used in [54], where the authors are interested in getting the best preconditioner not depending on its size.

TABLE 3

Timings to for the preconditioner and preconditioned CGLS for the BIF, IC, AINV, and CIMGS preconditioners.

Matrix	BIF		IC		AINV		CIMGS	
	t_p	t_{it}	t_p	t_{it}	t_p	t_{it}	t_p	t_{it}
S	0.05	0.13	0.02	0.17	0.12	0.13	0.02	0.86
	0.20	0.24	0.08	0.05	0.45	0.08	0.33	0.03
PHOTO2	0.05	0.20	0.03	0.16	0.16	0.23	0.03	‡
	0.35	0.27	0.06	0.42	0.44	2.54	0.09	0.16
M	0.03	0.44	0.05	0.43	0.13	0.42	0.03	‡
	0.98	0.89	0.23	0.09	0.69	0.21	0.27	1.36
MAR_R7	0.92	0.06	0.27	0.05	0.55	0.05	0.30	0.42
	4.11	0.08	0.80	0.03	4.49	0.06	2.15	0.03
DFL001	0.17	0.94	0.05	1.12	0.17	0.94	0.11	‡
	2.56	1.37	2.09	5.32	4.02	6.60	98.60	0.84
STRG2-27	0.44	2.31	0.11	1.06	0.44	‡	0.44	‡
	1.34	1.55	0.32	1.98	88.30	‡	172.00	‡
TESTBIG	7.36	0.56	1.72	0.18	5.24	0.64	1.53	‡
	13.90	0.52	3.72	0.17	1.85	‡	52.20	‡
OSA_07	0.06	0.13	0.05	0.19	0.20	0.55	0.06	1.58
	0.07	0.12	0.06	0.27	0.22	0.33	0.09	0.09
L	0.28	1.11	0.09	1.20	0.34	0.90	0.03	‡
	1.42	1.61	0.73	0.31	2.29	0.50	3.81	‡
KEMELM	2.55	1.47	0.14	‡	0.25	‡	0.13	0.56
	11.10	2.46	0.40	‡	1.16	‡	0.31	0.56
OSA_14	0.16	0.59	0.13	0.34	0.39	0.47	0.11	2.86
	0.16	0.25	0.11	0.59	0.52	0.89	0.20	0.27
DELTAX	14.10	2.64	0.92	2.86	2.89	2.70	0.75	‡
	26.40	2.66	1.77	3.02	5.80	2.71	1.09	‡
LMARK	1.14	1.75	0.62	0.41	0.70	‡	0.59	‡
	2.64	1.03	0.72	‡	2.26	‡	0.81	0.12
OSA_30	0.29	0.72	0.25	0.64	0.36	0.92	0.24	5.17
	0.30	0.49	0.25	0.76	0.90	0.78	0.40	0.61
KEN_18	0.90	14.20	0.72	15.00	0.58	8.69	0.25	‡
	4.95	22.70	2.93	30.70	2.00	11.10	2.68	32.70
MESH_DF	1.14	4.63	0.34	‡	0.36	6.82	0.33	15.60
	1.85	4.57	0.47	0.28	0.41	10.2	0.55	2.12
IMG_IN	46.3	1.11	5.38	‡	‡	‡	0.71	9.08
	75.4	0.87	6.35	‡	‡	‡	1.21	5.66
SLS	4.17	16.6	3.51	18.20	4.06	24.00	‡	‡
	4.24	7.91	3.46	14.40	5.68	16.80	458.00	20.30

An important goal of the experiments was also to demonstrate completely different robustness of the approaches with respect to the sizes of generated preconditioners. The preconditioners sharing the same line in the tables were chosen such that they have approximately the same sizes for the compared approaches, but it was not possible to achieve this in some cases. Then, a significantly lower size of the preconditioner means that an algorithm was not able to generate a larger preconditioner, possibly also because of the implementation. A significantly larger size for a particular method means that it was not possible to obtain a useful preconditioner of the size similar to other methods because of some numerical problems (breakdowns in construction or nonconvergence of the preconditioned iterative method). For example, BIF preconditioners for the OSA matrices are always generated very sparse and they are surprisingly still very efficient. Limited internal memory in BIF here may help to interrupt some fill-in dependencies in the decomposition very early, and the final factor is then always sparse [20]. The implementation of BIF that limits memory for

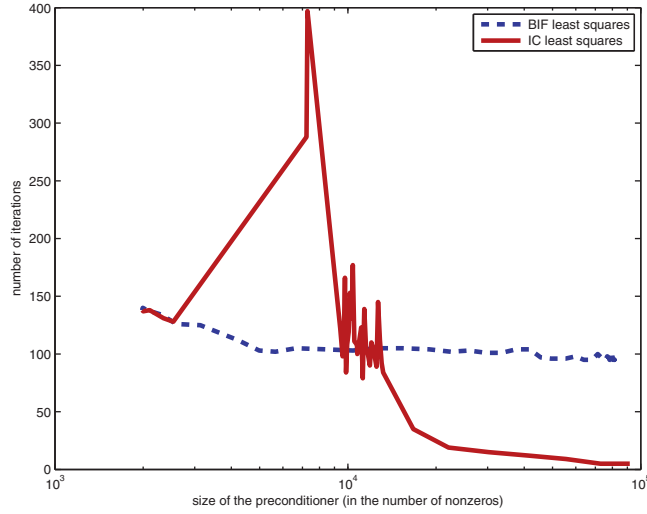


FIG. 1. Comparison of the BIF and IC preconditioned CGLS for the matrix S from the animal breeding package.

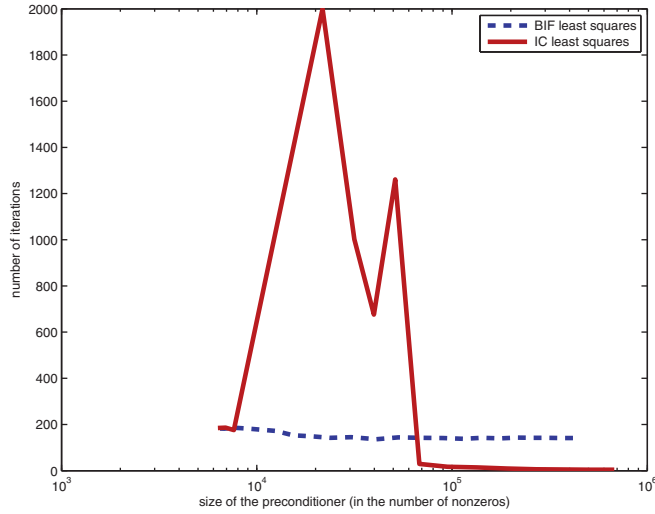


FIG. 2. Comparison of the BIF and IC preconditioned CGLS for the matrix M from the animal breeding package.

the rows of V may not generally allow one to obtain preconditioners that imply very small iteration counts. Later we show that IC is rather susceptible to breakdowns or maybe needs a large number of iterations for significant sizes of the preconditioner, but we tried to avoid these unstable cases in Tables 2 and 3. To get a more comprehensive view of the behavior of the IC preconditioner including possible instabilities, one should consult instead Figures 1, 2, and 3.

It appears from the tables that the performance of BIF and IC is similar from the point of view of the rate of convergence of the preconditioned CGLS method. Time to compute BIF is higher than for IC, but not prohibitive. It can sometimes be as high as for AINV. Note that the time to compute the BIF factors for larger problems is smaller than for AINV because of the internal memory restriction for

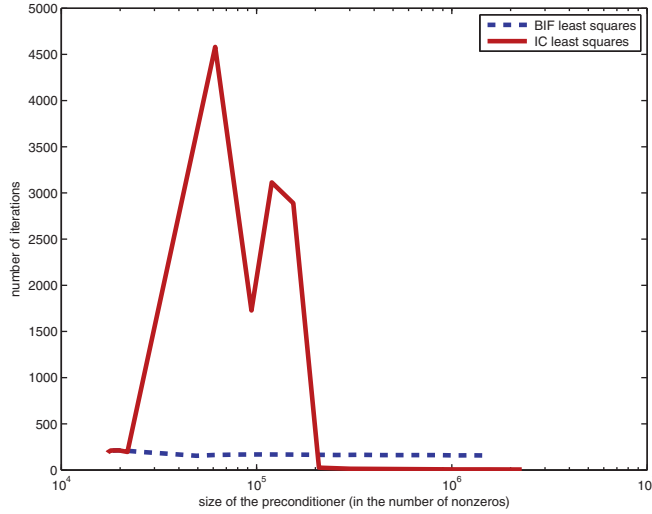


FIG. 3. Comparison of the BIF and IC preconditioned CGLS for the matrix L from the animal breeding package.

V mentioned above. On the other hand, the timings, especially for larger matrices, are increased by the breakdown-free computation of the diagonal entries with the full bilinear form. It pays off from the point of view of robustness since it is the only method that successfully solves all the considered problems. CGLS preconditioned by AINV can sometimes be rather competitive for sparser preconditioners. AINV turns out to be slower when a denser incomplete decomposition is needed. This behavior corresponds to our expectation that sparse approximate inverses often easily capture characteristic features of the matrix, but in order to get higher accuracy, they may need more nonzeros than IC. But the difference may not be always prohibitive, and even denser AINV preconditioners may be useful, in particular, in parallel implementations [10]. Results for the CIMGS preconditioner also agree with previous computational experience [49, 62, 8, 66]. In most cases, CIMGS cannot generate preconditioners which would be sparse and powerful at the same time. If more fill-in is allowed, the method may converge very fast. Since the incomplete QR decomposition hidden behind CIMGS does not drop elements in Q , the whole construction may take significant time.

Now, let us have a closer look at the experiments for some of the matrices. Preconditioning with IC and BIF of the CGLS method using the three matrices from the animal breeding package is depicted on Figures 1–3. The IC preconditioner for these matrices shows significant instability for a large range of drop tolerances. The behavior of BIF corresponds to our expectations on its robustness. Further, we can see that if the IC preconditioner works, it can be faster than the BIF strategy in many cases. Summarizing this observation, IC is often faster but it is more often unstable than the BIF approach. Let us emphasize that if the preconditioners would be compared only by taking a few results for the tables, we would not see the whole story. An important question is the proportion in which the algorithm and the implementation contribute to this picture. One of the characteristic features of our BIF implementation mentioned in [20] and above is that we store the factor V by rows in a fixed space that does not allow to obtain very large preconditioners. The experiments for matrices LP_KEN_18 (see Figure 4) and LP_DFL001 (see Figure 5) demonstrate this

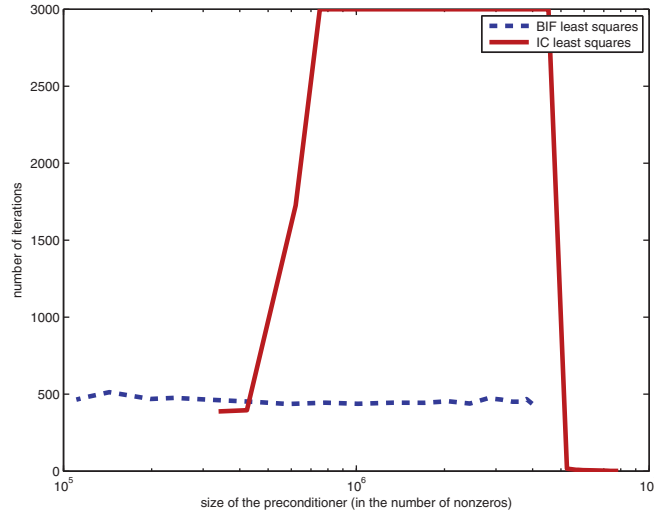


FIG. 4. Comparison of the BIF preconditioning and ICT preconditioning of CGLS for the matrix *LP_KEN_18*.

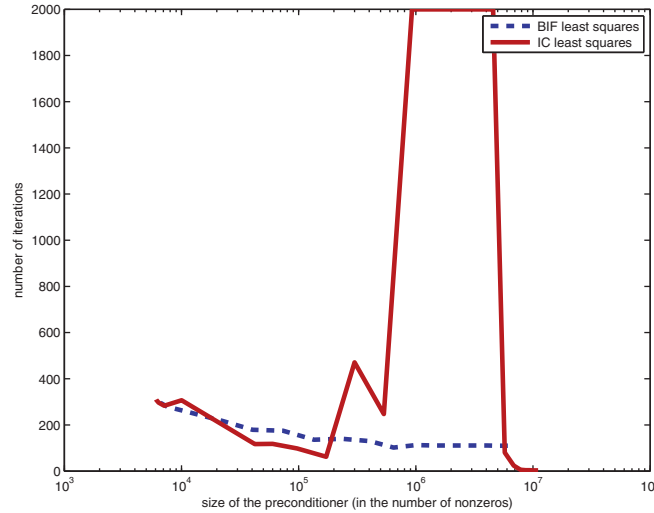


FIG. 5. Comparison of the BIF and IC preconditioning of CGLS for the matrix *LP_DFL001*.

fact. In particular, a closer look at the iteration counts for BIF and IC shows again that IC is not stable with respect to the size of the preconditioner, in particular for preconditioners of medium size. Note that in both cases, CGLS preconditioned by IC was unable to converge within a maximum number of 5,000 iterations. These figures once more confirm that the information contained in the figures may reveal more of the real complexity of the comparison, and in some sense, is also complementarity of the different approaches.

We also show results of all the considered methods graphically in one figure. Figure 6 compares the iteration counts for the matrix *HIRLAM* ($m = 1,385,270$, $n = 452,200$, $nz = 2,713,200$) [45] from the meteorological application that was also used

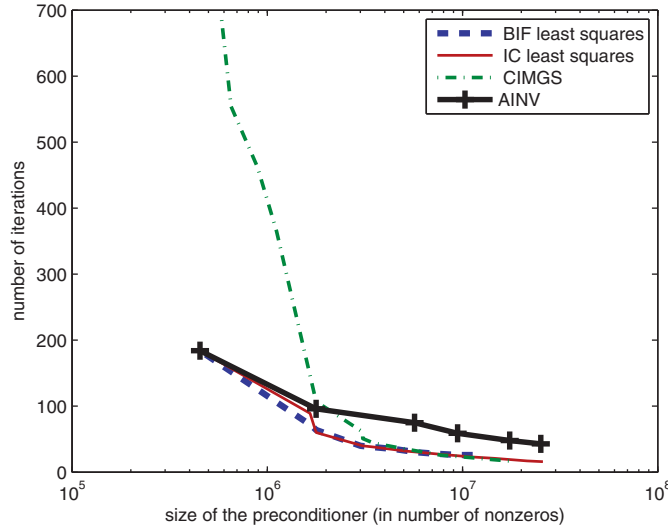


FIG. 6. Comparison of LS preconditioned by the four considered preconditioners for the matrix *HIRLAM*.

in the experiments in [12]. None of the tested methods is unstable for this matrix. IC works rather well, and BIF works also reasonably well, although it is sometimes slower. One of the main messages of this figure is to confirm that CIMGS is very good once the size of the preconditioner is sufficiently large, and it may be very poor if the preconditioner is kept sparse. AINV also solves the problem rather well if the decomposition is kept sparse. Otherwise, AINV needs consistently more iterations, but neither approach is prohibitive.

The second set of results represents the LU-based preconditioning, which is the strategy that decomposes only a submatrix A_1 of A . Clearly, this type of preconditioning may be rather weak whenever A_1 represents only a small part of the structural and numerical properties of A . One may think it will happen whenever $m \gg n$, but the experiments show that there is not clear evidence of this implication. Tables show only a small part of our test set, namely, the matrices that lead to convergence. We are interested in showing that the preprocessing based on the weighted transversal for rectangular matrices may bring advantages. We will see that such preprocessing turns out to be a useful tool for developing better techniques to solve LS problems iteratively that may be further enhanced by a multilevel approach; see [52].

Tables 4 and 5 present results of CGLS preconditioned by A_1^{-1} . For both experiments with the LU preconditioner, we used the dual dropping algorithm ILUT with drop tolerance τ equal to 0.1, allowing at most ten nonzeros in a row of the ILUT preconditioner. This choice of parameters is a compromise for the set of matrices of various sizes and from various applications. Our preliminary experience that considered smaller τ and more allowed fill-in lead for larger matrices to high computational times. In both cases, we preprocessed the matrices by the described weighted transversal strategy that chooses A_1 . We have also used the related nonsymmetric scaling described in [53] and [33], and the weights for the weighted transversal were fixed as in (3.6) to $\theta = 0.55$. Nevertheless, this setting for the reordered and scaled test matrices was not critical since the algorithm was in many cases insensitive to changes in θ .

TABLE 4
Preconditioning CGLS based on decomposition of A_1 using the weighted transversal.

Matrix	t_p	t_{it}	size	its
S	0.01	0.20	5,368	149
M	0.02	0.87	16,812	294
LP_MAROS_R7	0.10	0.02	3,136	24
L	0.02	4.19	43,785	588
TESTBIG	0.02	0.19	28,018	26
LP_OSA_07	0.02	0.39	1,118	69
LP_OSA_14	0.02	0.84	2,337	71
LP_OSA_30	0.06	1.53	4,350	69

TABLE 5
Preconditioning CGLS based on decomposition of A_1 based on the weighted transversal and dynamic partial pivoting.

Matrix	t_p	t_{it}	size	its
S	0.02	0.20	5,271	128
M	0.03	0.80	15,990	266
LP_MAROS_R7	0.02	0.17	15,565	26
KEMELMACHER	0.21	2.15	30,195	315
TESTBIG	0.04	0.75	45,777	104
LP_OSA_07	0.01	0.34	1,118	56
L	0.03	2.87	42,980	400
LP_OSA_14	0.03	0.71	2,337	56
LP_OSA_30	0.06	1.29	4,350	56
SLS	0.65	45.9	62735	151

Table 4 depicts results with the standard ILUT decomposition of A_1 without any additional pivoting. Then, for experiments in Table 5, ILUT uses also the partial pivoting for finding the pivot of the largest magnitude in each column searching among the nonzero entries from all m rows of A . This pivoting dynamically modifies the original choice of A_1 .

We can see that once the transversal based strategy is applied, the LU preconditioning is able to improve some of the iteration counts. Sometimes it does not improve the results, but in general it turns out to be helpful. Consider, e.g., results for the matrix TESTBIG in Table 4. Here we get the convergence in 26 iterations for a much smaller preconditioner than in the algorithms based on the normal equations. Clearly, in this case the submatrix A_1 was able to capture important characteristic features of A having the dimension around two thirds of the dimension of A , but also, for some other matrices, very efficient runs with respect to the preconditioners based on the normal equations were observed, as for the matrices LP_MAROS_R7 and LP_OSA_07 that are solved faster. Also note that CGLS performed badly in other cases for which m is not much greater than n , as for the animal breeding matrices.

The experiments given in the first part of this section targeting method robustness may point out an explanation. Decomposing only a submatrix of A that may be rather sparse can be considered a partial direct decomposition that does not imply breakdown but also does not help much. Simply, the method can be far from transforming CGLS into a convergent iterative method. Our point of view is that this is just the main practical drawback of the LU preconditioner for solving LS problems. Despite the observed behavior that could be caused by the overall instability of the preconditioned CGLS, let us note that LU preconditioning may be a method of choice for the solution of weighted least squares as shown in [15]. Another important advantage is that it

is an algorithm that straightforwardly avoids forming $A^T A$ in the construction of a preconditioner. This feature can be useful because of relatively low memory demands. Further, since it uses the original block A_2 , the preconditioner turns out to be more suitable for the matrix-free framework if A_1 can be determined directly from the application. But, taking into account also the fact that we are able to solve only a part of the problems in this way, LU preconditioning is usually less efficient than direct preconditioning of the normal equations. An open question is whether LU preconditioning may be more powerful if another preconditioner is applied on the top of the matrix $A_2 A_1^{-1}$ or when we apply the incomplete decomposition repeatedly to $A_2 A_1^{-1}$ for A with $m > 2n$.

Let us finalize this section with some comments on additional experience obtained with the complete LU preconditioning. In some cases we were able to show that even the direct solver used to compute the complete LU decomposition of a square submatrix may be slower whenever used as a preconditioner. Nevertheless, a preferable way to use direct solvers for solving sparse LS problems is to keep the factor L as well-conditioned as possible and to apply them to a reformulated system as pointed out in [55]; see also [58, 59]. Therefore, we do not refer to these experiments here. Also note that the strategy proposed here is a result of tests with other approaches including some crash procedures for linear programming; see, e.g., [41] and P4/P5 preprocessings [44, 38, 50]. We have found that all these alternative preprocessings need additional procedures to improve them. Their investigation is one of our future tasks.

5. Conclusions and future work. In this paper we have described new preconditioning strategies for solving sparse LS problems by preconditioned iterative methods. In particular, we have evaluated the performance of BIF preconditioning for the CGLS method. Moreover, we have further stabilized the BIF algorithm to make it breakdown-free for symmetric and positive definite matrices. Our experimental results show the robustness of the BIF algorithm for solving LS problems that is in line with previous work for systems of linear equations. Although solving LS problems iteratively is rather hard, and up to now there has been no computational strategy that would iteratively solve systems arising from a broad pool of various applications, we probably do not exaggerate by saying that the BIF preconditioning belongs to the family of algebraic preconditioners of choice, as recently found for RIF.

Further, we introduced a new preprocessing strategy for incomplete LU preconditioning following earlier work proposed by Läuchli. It is based on the computation of weighted maximum transversals by rows and also takes into account the sparsity of the leading submatrix A_1 . Our experiments show that this reordering improves the conditioning and sparse structure of this submatrix.

Our future work in the field of the LU preconditioning will consider stabilization of the crash techniques developed for the simplex method of linear programming by some postprocessing, multiple application of the Läuchli splitting, and combination with the other techniques. In particular, we would like to consider hierarchical methods to solve LS problems, continuing the pioneering work [52]. We intend to make the resulting code publicly available as that of its predecessor, the RIF method.

Acknowledgments. We would like to thank the editor and anonymous referees. Their constructive comments helped us to improve the paper significantly. Parts of this paper were written while the fourth author was visiting the Universitat Politècnica de València: the hospitality and support are greatly appreciated.

REFERENCES

- [1] M. A. AJIZ AND A. JENNINGS, *A robust incomplete Cholesky-conjugate gradient algorithm*, Int. J. Numer. Methods Engrg., 20 (1984), pp. 949–966.
- [2] O. AXELSSON AND L. Y. KOLOTILINA, *Diagonally compensated reduction and related preconditioning methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 155–177.
- [3] Z.-Z. BAI, I. S. DUFF, AND A. J. WATHEN, *A class of incomplete orthogonal factorization methods. I: Methods and theories*, BIT, 41 (2001), pp. 53–70.
- [4] S. BELLAVIA, J. GONDZIO, AND B. MORINI, *A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems*, SIAM J. Sci. Comput., 35 (2013).
- [5] C. BENOIT, *Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés a un système d'équations linéaires en nombre inférieur a celui des inconnues. Application de la méthode a la résolution d'un système défini d'équations linéaires*, Bulletin Géodésique, 2 (1924), pp. 5–77.
- [6] M. BENZI, *private communication*, 2000.
- [7] M. BENZI, J. C. HAWS, AND M. TŮMA, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.
- [8] M. BENZI, R. KOUHIA, AND M. TŮMA, *Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 6533–6554.
- [9] M. BENZI, C. D. MEYER, AND M. TŮMA, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.
- [10] M. BENZI AND M. TŮMA, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.
- [11] M. BENZI AND M. TŮMA, *A robust incomplete factorization preconditioner for positive definite matrices*, Numer. Linear Algebra Appl., 10 (2003), pp. 385–400.
- [12] M. BENZI AND M. TŮMA, *A robust preconditioner with low memory requirements for large sparse least squares problems*, SIAM J. Sci. Comput., 25 (2003), pp. 499–512.
- [13] Å. BJÖRCK, *SSOR preconditioning methods for sparse least squares problems*, in Proceedings of the Computer Science and Statistics 12th Annual Symposium on the Interface, University of Waterloo, Waterloo, Canada, 1979, pp. 21–25.
- [14] Å. BJÖRCK, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
- [15] A. BJÖRCK AND J. Y. YUAN, *Preconditioners for least squares problems by LU factorization*, Electron. Trans. Numer. Anal., 8 (1999), pp. 26–35.
- [16] M. BOLLHÖFER, *A robust ILU with pivoting based on monitoring the growth of the inverse factors*, Linear Algebra Appl., 338 (2001), pp. 201–218.
- [17] M. BOLLHÖFER, *A robust and efficient ILU that incorporates the growth of the inverse triangular factors*, SIAM J. Sci. Comput., 25 (2003), pp. 86–103.
- [18] M. BOLLHÖFER AND Y. SAAD, *On the relations between ILUs and factored approximate inverses*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 219–237.
- [19] R. BRU, J. CERDÁN, J. MARÍN, AND J. MAS, *Preconditioning sparse nonsymmetric linear systems with the Sherman–Morrison formula*, SIAM J. Sci. Comput., 25 (2003), pp. 701–715.
- [20] R. BRU, J. MARÍN, J. MAS, AND M. TŮMA, *Balanced incomplete factorization*, SIAM J. Sci. Comput., 30 (2008), pp. 2302–2318.
- [21] R. BRU, J. MARÍN, J. MAS, AND M. TŮMA, *Improved balanced incomplete factorization*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2431–2452.
- [22] R. BURKARD, M. DELL'AMICO, AND S. MARTELLO, *Assignment Problems*, SIAM, Philadelphia, 2009.
- [23] S. L. CAMPBELL AND C. D. MEYER, JR., *Generalized Inverses of Linear Transformations*, Pitman, London, 1979.
- [24] X.-W. CHANG, C. C. PAIGE, AND D. TITLEY-PÉLOQUIN, *Stopping criteria for the iterative solution of linear least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 831–852.
- [25] M. T. CHU, R. E. FUNDERLIC, AND G. H. GOLUB, *A rank-one reduction formula and its applications to matrix factorizations*, SIAM Rev., 37 (1995), pp. 512–530.
- [26] X. CUI, *Approximate Generalized Inverse Preconditioning Methods for Least Squares Problems*, Ph.D. thesis, Department of Informatics, School of Multidisciplinary Sciences, The Graduate University for Advanced Studies (Sokendai), Hayama, Japan, 2009.
- [27] X. CUI AND K. HAYAMI, *Generalized approximate inverse preconditioners for least squares problems*, Jpn. J. Ind. Appl. Math., 26 (2009), pp. 1–14.

- [28] X. CUI, K. HAYAMI, AND J.-F. YIN, *Greville's method for preconditioning least squares problems*, in Proceedings of Algorithm 2009 Conference on Scientific Computing, 2009, pp. 440–448.
- [29] X. CUI, K. HAYAMI, AND J.-F. YIN, *Greville's method for preconditioning least squares problems*, Adv. Comput. Math., 35 (2011), pp. 243–269.
- [30] T. A. DAVIS, *Algorithm 915: Multifrontal multithreaded rank-revealing sparse QR factorization*, ACM Trans. Math. Softw., 38 (2011), pp. 8(1)–8(22).
- [31] T. A. DAVIS, *University of Florida Sparse Matrix Collection*, <http://www.cise.ufl.edu/research/sparse/matrices/> (1994).
- [32] I. S. DUFF, K. KAYA, AND B. UÇAR, *Design, implementation, and analysis of maximum transversal algorithms*, ACM Trans. Math. Software, 38 (2011), pp. 13:1–13:31.
- [33] I. S. DUFF AND J. KOSTER, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.
- [34] I. S. DUFF AND J. KOSTER, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996.
- [35] I. S. DUFF AND S. PRALET, *Towards stable mixed pivoting strategies for the sequential and parallel solution of sparse symmetric indefinite systems*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1007–1024.
- [36] I. S. DUFF AND T. WIBERG, *Remarks on implementations of $O(n^{1/2}\tau)$ assignment algorithms*, ACM Trans. Math. Software, 14 (1988), pp. 267–287.
- [37] V. ELKHOUT, *On the existence problem of incomplete factorisation methods*, Technical report UT-CS-99-435, Department of Computer Science, University of Tennessee, 1999.
- [38] A. M. ERISMAN, R. G. GRIMES, J. G. LEWIS, AND W. G. POOLE, JR., *A structurally stable modification of Hellerman–Rarick's P^4 algorithm for reordering unsymmetric sparse matrices*, SIAM J. Numer. Anal., 2 (1985), pp. 369–385.
- [39] R. FREUND, *A note on two block-SOR methods for sparse least squares problems*, Linear Algebra Appl., 88/89 (1987), pp. 211–221.
- [40] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.
- [41] N. I. M. GOULD AND J. K. REID, *New crash procedures for large systems of linear constraints*, Math. Program., 45 (1989), pp. 475–501.
- [42] T. N. E. GREVILLE, *Some applications of the pseudoinverse of a matrix*, SIAM Rev., 2 (1960), pp. 15–22.
- [43] M. HAGEMANN AND O. SCHENK, *Weighted matchings for preconditioning symmetric indefinite linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 403–420.
- [44] E. HELLERMAN AND D. RARICK, *The partitioned preassigned pivot procedure (P^4)*, in Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972, pp. 67–76.
- [45] A. HOLSTAD AND I. LIE, *On the Computation of Mass Conservative Wind and Vertical Velocity Fields*, Technical report 141, The Norwegian Meteorological Institute, Oslo, Norway, 2002.
- [46] J. E. HOPCROFT AND R. M. KARP, *A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*, in the Conference Record of the Twelfth Annual Symposium on Switching and Automata Theory, IEEE, East Lansing, MI, 1971.
- [47] D. JAMES, *Conjugate Gradient Methods for Constrained Least Squares Problems (Least Squares)*, Ph.D. thesis, University of North Carolina, 1990.
- [48] A. JENNINGS AND M. A. AJIZ, *Incomplete methods for solving $A^T A x = b$* , SIAM J. Sci. Stat. Comput., 5 (1984), pp. 978–987.
- [49] I. E. KAPORIN, *High quality preconditioning of a general symmetric positive definite matrix based on its $U^T U + U^T R + R^T U$ decomposition*, Numer. Linear Algebra Appl., 5 (1998), pp. 483–509.
- [50] J. L. KENNINGTON AND R. A. K. MOHAMED, *Recovery from numerical instability during basis reinversion*, Comput. Optim. Appl., 8 (1997), pp. 57–71.
- [51] P. LÄUCHLI, *Jordan-Elimination und Ausgleichung nach kleinsten Quadraten*, Numer. Math., 3 (1961), pp. 226–240.
- [52] N. LI AND Y. SAAD, *MIQR: A multilevel incomplete QR preconditioner for large sparse least-squares problems*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 524–550.
- [53] A. NEUMAIER AND M. OLSCHOWKA, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.
- [54] A. T. PAPADOPOULOS, I. S. DUFF, AND A. J. WATHEN, *A class of incomplete orthogonal factorization methods. II: Implementation and results*, BIT, 45 (2005), pp. 159–179.
- [55] G. PETERS AND J. H. WILKINSON, *The least squares problem and pseudo-inverse*, Comput. J., 131 (1970), pp. 309–316.

- [56] Y. SAAD, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.
- [57] Y. SAAD, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
- [58] M. A. SAUNDERS, *Sparse least squares problems by conjugate gradients: A comparison of preconditioning methods*, in Proceedings of Computer Science and Statistics: 12th Annual Conference on the Interface, Waterloo, Canada, 1979.
- [59] M. A. SAUNDERS, *LUSOL: Sparse LU Factorization Package*, version 7.0, 2008, <http://web.stanford.edu/group/SOL/download.html>.
- [60] O. SCHENK, A. WÄCHTER, AND M. HAGEMANN, *Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization*, J. Comput. Optim. Appl., 36 (2007), pp. 321–341.
- [61] J. SCOTT AND M. TŮMA, *The importance of structure in incomplete factorization preconditioners*, BIT, 51 (2011), pp. 385–404.
- [62] M. SUARJANA AND K. H. LAW, *A robust incomplete factorization based on value and space constraints*, Int. J. Numer. Methods Eng., 38 (1995), pp. 1703–1719.
- [63] M. TISMENETSKY, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154–156 (1991), pp. 331–353.
- [64] X. WANG, *Incomplete Factorization Preconditioning for Linear Least Squares Problems*, Ph.D. Thesis, Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL, 1993.
- [65] X. WANG, K. A. GALLIVAN, AND R. BRAMLEY, *CIMGS: An incomplete orthogonal factorization preconditioner*, SIAM J. Sci. Comput., 18 (1997), pp. 516–536.
- [66] I. YAMAZAKI, Z. BAI, W. CHEN, AND R. SCALETTAR, *A high-quality preconditioning technique for multi-length-scale symmetric positive definite linear systems*, Numer. Math. Theory Methods Appl., 2 (2009), pp. 469–484.
- [67] Z. ZLATEV, *Computational Methods for General Sparse Matrices*, Kluwer, Dordrecht, 1991.
- [68] Z. ZLATEV AND H. B. NIELSEN, *Solving large and sparse linear least-squares problems by conjugate gradient algorithms*, Comput. Math. Appl., 15 (1988), pp. 185–202.