Preconditioner Updates for Solving Sequences of Large and Sparse Nonsymmetric Linear Systems

Duintjer Tebbens, Jurjen
2005

Institute of Computer Science
Academy of Sciences of the Czech Republic

Preconditioner Updates for Solving Sequences of Large and Sparse Nonsymmetric Linear Systems

Jurjen Duintjer Tebbens, Miroslav Tůma

Technical report No. 940

August 2005

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: +420 266 053 251, fax: +420 286 585 789,
e-mail:{tebbens,tuma,}@cs.cas.cz,

Institute of Computer Science
Academy of Sciences of the Czech Republic

Preconditioner Updates for Solving Sequences of Large and Sparse Nonsymmetric Linear Systems

Jurjen Duintjer Tebbens, Miroslav Tůma

Technical report No. 940

August 2005

Abstract:

Preconditioner updates for solving sequences of linear algebraic systems are considered. New and theoretically motivated strategies for approximating updates of factorized nonsymmetric preconditioners are presented. It is shown experimentally that some of them can be very beneficial. In particular, they are successful in significantly decreasing the number of iterations of a preconditioned iterative method for solving subsequent systems of a sequence when compared with the strategy of freezing the preconditioner computed in the first step of the sequence. In some cases, the updated preconditioners offer a rate of convergence close to the rate obtained when preconditioning with recomputed preconditioners. In addition, their application may be cheaper than that of recomputed factors. Since the updates are typically cheap and straightforward, we believe that their use is of practical interest and that they can replace recomputing preconditioners, which is often expensive, especially in parallel and matrix-free environments.

Keywords:
Preconditioned iterative methods, sparse matrices, sequences of linear algebraic systems, incomplete factorizations, factorization updates, Gauss-Jordan transformations, minimum spanning tree

PRECONDITIONER UPDATES FOR SOLVING SEQUENCES OF LARGE AND SPARSE NONSYMMETRIC LINEAR SYSTEMS

JURJEN DUINTJER TEBBENS AND MIROSLAV TŮMA*

Abstract. Preconditioner updates for solving sequences of linear algebraic systems are considered. New and theoretically motivated strategies for approximating updates of factorized nonsymmetric preconditioners are presented. It is shown experimentally that some of them can be very beneficial. In particular, they are successful in significantly decreasing the number of iterations of a preconditioned iterative method for solving subsequent systems of a sequence when compared with the strategy of freezing the preconditioner computed in the first step of the sequence. In some cases, the updated preconditioners offer a rate of convergence close to the rate obtained when preconditioning with recomputed preconditioners. In addition, their application may be cheaper than that of recomputed factors. Since the updates are typically cheap and straightforward, we believe that their use is of practical interest and that they can replace recomputing preconditioners, which is often expensive, especially in parallel and matrix-free environments.

Key words. preconditioned iterative methods, sparse matrices, sequences of linear algebraic systems, incomplete factorizations, factorization updates, Gauss-Jordan transformations, minimum spanning tree

1. Introduction.

We consider the solution of sequences of linear systems

$$A^i x = b^i,\ i = 1, \ldots, \tag{1.1}$$

where $A^i \in \mathbb{R}^{n\times n}$ are general nonsingular sparse matrices and $b^i \in \mathbb{R}^n$ are corresponding right-hand sides. Such sequences arise in many applications like computational fluid dynamics, structural mechanics, numerical optimization as well as in solving non-PDE problems. For example, a system of nonlinear equations $F(x) = 0$ for $F : \mathbb{R}^n \to \mathbb{R}^n$ solved by a Newton or Broyden-type method leads to a sequence of problems

$$J(x_i)(x_{i+1} - x_i) = -F(x_i),\ i = 1, \ldots, \tag{1.2}$$

where $J(x_i)$ is the Jacobian evaluated in the current iteration x_i or its approximation [19], [20].

The solution of such sequences of linear systems is the main bottleneck in many applications mentioned above. Iterative Krylov subspace solvers are often methods of choice when the systems are large. In many cases, these solvers must be preconditioned in order to be efficient. Computing preconditioners $M^1, M^2, \ldots$ for individual systems separately, may be very expensive. There is a strong need for reduction of costs by sharing some of the computational effort among the subsequent linear systems.

A way to reduce the overall costs for solving systems of the type (1.2) is to modify Newton's method by skipping some Jacobian evaluations as in the Shamanskii combination of Newton's method and the Newton-chord method [7], [35]. In this way we get a sequence of systems with identical matrices, and techniques for solving systems with more right-hand sides may be applied, see, e.g., [27], [31], [38], [40], provided the right-hand sides are available. However, combination of a modified Newton method with a Krylov subspace solver, called modified Newton-Krylov method (MNK), has much weaker nonlinear convergence properties than the standard Newton's method.

*Institute of Computer Science, Czech Academy of Sciences, Pod Vod. věží 2, 18207 Praha 8, Czech Republic. This work is supported by the National Program of Research "Information Society" under project 1ET400300415.

Another approach, which is usually more efficient, is based on *freezing* the preconditioner (using the same preconditioner for a sequence of linear systems), but recomputing (approximate) Jacobians A^i [32], [22], [23]. This approach is very natural in the context of a matrix-free environment, where the system matrices A^i may be available only in the form of matrix-vector products (matvecs), see also the overview of matrix-free Newton-Krylov methods in [21]. Even when a matrix is not available explicitly, it can be often estimated from results of matvecs with a suitably chosen set of test vectors. The problem of minimizing the number of test vectors, described in the pioneering paper [10], was formalized in [37] in terms of graph coloring problems of a related graph. Since then a bunch of heuristic algorithms, software and applications for estimation of sparse matrices has been developed, see, e.g., the overview [16]. In matrix-free environment, a preconditioner with a special structure based on the diagonally compensated reduction [1] can be sometimes obtained from an exactly estimated auxiliary matrix using a small number of matvecs since the graph coloring algorithms for matrix estimation with a reduced number of constraints allow this [9]. An important assumption for applicability of matrix estimation techniques is the knowledge of the matrix sparsity pattern. Fortunately, in many cases the pattern is known, e.g., from the discretization grid.

As we will demonstrate later, freezing the preconditioner need not be enough for fast convergence of preconditioned iterative methods in many practical cases. In order to decrease the overall computational effort for solving a sequence of subsequent systems we may reuse some additional information from the linear system $A^1x = b^1$. One example of such an algebraic approach in the Newton-Krylov framework is to recycle Krylov subspaces among systems of a sequence, see, e.g., [24], [29].

Our contribution is targeted to improvement of algebraic preconditioners which may be considered as a complementary approach. We propose here *approximate* updates of a preconditioner M^1 which is factorized as $LDU \approx A$ or $ZDW \approx A^{-1}$. Note that many interesting algorithms were proposed for *exact* updates of decompositions. Recent sparse updates [11], [12], [36] replace in some cases classical dense updates from, e.g., [30]. There is some recent work in approximate updates as well. For example, approximate diagonal updates of approximate inverse preconditioners which are useful for solving parabolic PDEs were proposed in [2], see also [6]. A straightforward approximate rank one update of a preconditioner can be obtained in case of a sequence of linear systems from a quasi-Newton method, as shown in the SPD case in [26], [5].

To summarize our contribution, we present new approaches to approximate updates of factorized, and general nonsymmetric preconditioners which may be useful in solving subsequent linear systems. The paper is organized as follows. In Section 2 we present an introduction into preconditioner updates, and briefly motivate and discuss possible strategies. In Section 3 we introduce new approaches with a necessary theoretical motivation. The results of numerical experiments with the new algorithms are presented and discussed in Section 4. Directions for current and future research are given in Conclusions. Throughout the paper, $\|\cdot\|$ denotes an arbitrary matrix norm.

2. Approximate updates of preconditioners. Some of the strategies to update preconditioners that we mentioned in the introduction are linked with specific classes of linear solvers (e.g. recycling Krylov subspaces) or their effectiveness depends on special properties of the involved system matrices (e.g. while exploiting symmetry). In this paper we wish to consider sequences of general, nonsymmetric

systems that are solved by preconditioned iterative methods. We address the following problems: First, how can we update, in theory, a preconditioner in such a way that the updated preconditioner is likely to be as powerful as the original one? And second, how can we approximate, in practice, such an update in order to obtain a preconditioner that is inexpensive to apply and yet useful?

In order to simplify the notation, we consider two linear systems of dimension n denoted by $Ax = b$ and $A^+x^+ = b^+$. Denote the difference matrix $A - A^+$ by B and let M be a preconditioner approximating A. The quality of the preconditioner M can be expressed by a norm of the matrix

$$A - M \tag{2.1}$$

or by some norm of one of the matrices

$$I - M^{-1}A \quad \text{or} \quad I - AM^{-1} \tag{2.2}$$

if we consider preconditioning from the left or right, respectively (see, e.g. [2]). If preconditioners are in factorized form, both (2.1) and (2.2) should be considered in practice since the preconditioners can suffer from two types of deteriorations. While the norm of the matrix (2.1) expresses *accuracy* of the preconditioner, the norms of the matrices (2.2) relate to its *stability* [8], see also [3]. In the context of derivation of preconditioner updates we will prefer to address the norm of the matrix (2.1). Potential instabilities of the updated preconditioner can be reduced by update modifications as we will show later. We have

$$\|A - M\| = \|A^+ - (M - B)\|,$$

hence $M^+ \equiv M - B$ is an updated preconditioner for A^+ of the same "level" of accuracy as M is for A. This "ideal" updated preconditioner cannot be used, in general, in practice since multiplication of vectors with $(M - B)^{-1}$ may be too expensive. There are ways, however, to approximate matvecs with $(M - B)^{-1}$. Note that there may very well exist different preconditioners that are ideal with respect to a norm of (2.1): Just consider $M^+ = M - C$ for some matrix $C \neq B$ with

$$\|A - M\| = \|A^+ - M^+\| = \|A^+ - M + C\|.$$

We will concentrate here, however, on approximation of $M^+ = M - B$ and we will assume it is nonsingular. Then we can write down its inverse with the help of the Sherman-Morrison formula.

Consider a decomposition $B = UV^T$ of the difference matrix B, where U and V are $n \times k$ matrices for some $k \leq n$. Then

$$(M^+)^{-1} = M^{-1} + M^{-1}U\left(I_k - V^TM^{-1}U\right)^{-1}V^TM^{-1}. \tag{2.3}$$

If there exists a decomposition $B = UV^T$ such that $k \ll n$, then application of $(M^+)^{-1}$ is computationally feasible. This is the case, for example, in quasi-Newton methods where the difference matrix has rank one or two [26], [5].

In general, however, we cannot assume the difference matrix has low rank and it will be necessary to *approximate* the ideal update to be able to compute matvecs with its inverse inexpensively. With respect to (2.3), the most logical way of doing so is finding a low rank approximation of B. This leads to the minimization problem

$$\min_{U,V,\, rank(UV^T)=k} \|B - UV^T\|, \tag{2.4}$$

for some $k \ll n$ and in some norm. In the Euclidean norm, the minimization problem is solved by considering the k largest singular values and corresponding singular vectors. Unfortunately, it is too expensive to compute the singular value decomposition of the large system matrices we are interested in. An option is computation of only a small number of dominant singular values and vectors (e.g. through a shift-and-inverted Lanczos process applied to B^TB) or some form of incomplete singular value decomposition (e.g. with Jacobi's method). There may be cases where the difference matrix B can be approximated inexpensively in this way, but in general computational costs may become at least as high as for the computation of a new preconditioner for A^+.

An inexpensive procedure that approximates B in the Frobenius norm results from simply choosing the k rows b_{i*} of largest 2-norm and approximating by $\sum_{i=1}^{k} e_i b_{i*}$, or, analogously, finding the k largest columns b_{*i} and approximating B with $\sum_{i=1}^{k} b_{*i} e_i^T$. In this way, we also approximate in the $\|\cdot\|_\infty$ or $\|\cdot\|_1$ norm. Clearly, these approximations need not be the best rank k approximations. In fact, in many applications the difference matrix has elements of similar magnitude along its diagonals, hence a small number of rows or columns cannot very well replace the matrix.

These considerations seem to imply that computation of a low rank approximation UV^T of B, followed by application of (2.3), does not seem to be promising. This motivated us to search for different, and often simpler, approximation strategies. In the following section we describe some options.

3. Updates of approximate decompositions. This section presents new preconditioners for a matrix $A^+ = A - B$ obtained as cheap updates from a preconditioner $M \approx A$. Assume that M is given in the form of a triangular decomposition as $M = LDU \approx A$, where L and U have unit main diagonal. The derivation of some of our updates is based on the assumption that the entries of L and U decay when moving away from the main diagonal, see the discussion on the decay of the entries of inverse factors in [4], cf. also [2]. In addition, note that sufficient diagonal dominance may often be imposed if A contains a strong transversal [28], [13], [14] such that its entries can be permuted to the main diagonal. Consider the "ideal" update $M - B$ from the previous section. If it is invertible, we can approximate, for example, as

$$(M-B)^{-1} = U^{-1}(D - L^{-1}BU^{-1})^{-1}L^{-1} \approx U^{-1}(D-B)^{-1}L^{-1}, \tag{3.1}$$

provided $D - B$ is nonsingular. Now assume $\overline{D-B}$ is a nonsingular approximation of $D - B$ that can be inverted inexpensively. Then define a preconditioner M^+ via the last expression in (3.1) as

$$M^+ = L(\overline{D-B})U. \tag{3.2}$$

The accuracy of this preconditioner can be significantly higher than the accuracy of the frozen preconditioner $M = LDU$ for A^+. In the following lemma we express the relation of these preconditioners quantitatively.

LEMMA 3.1. *Let $||A - LDU|| = \varepsilon||A|| < ||B||$. Then the preconditioner from (3.2) satisfies*

$$\begin{aligned} ||\ A^+ - M^+|| &\le ||A^+ - LDU|| \frac{||L(D - \overline{D-B})U - B|| + \varepsilon||A||}{||B|| - \varepsilon||A||} \le \\ &\le ||A^+ - LDU|| \frac{\|L\|\,\|D - B - \overline{D-B}\|\,\|U\| + ||L-I||\,\|BU\| + \|B\|\,||U-I|| + \varepsilon||A||}{||B|| - \varepsilon||A||}. \end{aligned}$$

PROOF: We get directly

$$
\begin{aligned}
&|| \; A^+ - M^+|| = ||A - B - L(\overline{D-B})U|| = ||(A - LDU) - L(\overline{D-B} - D)U - B|| \\
&\leq \left(\varepsilon||A|| + ||L(D - \overline{D-B})U - B||\right) \frac{||B|| - \varepsilon||A||}{||B|| - \varepsilon||A||} \\
&\leq \left(\varepsilon||A|| + ||L(D - \overline{D-B})U - B||\right) \frac{||(A - LDU) - B||}{||B|| - \varepsilon||A||} \\
&\leq ||A^+ - LDU|| \frac{||L(D - \overline{D-B})U - B|| + \varepsilon||A||}{||B|| - \varepsilon||A||} \\
&\leq ||A^+ - LDU|| \frac{||L(D - B - \overline{D-B})U + LBU - B|| + \varepsilon||A||}{||B|| - \varepsilon||A||} \\
&\leq ||A^+ - LDU|| \frac{\|L\| \, \|D - B - \overline{D-B}\| \, \|U\| + ||L - I|| \, \|BU\| + \|B\| \, ||U - I|| + \varepsilon||A||}{||B|| - \varepsilon||A||}.
\end{aligned}
$$

□

Clearly, the bound for the norm $||A^+ - M^+||$ is small if $\overline{D-B}$ is close to $D-B$, and if A is strongly diagonally dominant since then $||L-I||$ and $||U-I||$ tend to be small. This is best seen in the rightmost bound, but the other bound is sharper.

In the symmetric case, the preconditioner M^+ changes to $M^+ = L(\overline{D-B})L^T$, hence symmetry is preserved. However, we are here primarily interested in the non-symmetric case, and in this case it may be advantageous to assume that only one of the two factors L, U is close to the identity matrix, instead of both. Without loss of generality, we choose L and thus we can approximate it as

$$
(M - B)^{-1} = (DU - L^{-1}B)^{-1}L^{-1} \approx (DU - B)^{-1}L^{-1}, \tag{3.3}
$$

if $DU - B$ is nonsingular. If $\overline{DU-B}$ denotes a nonsingular and easily invertible approximation of $DU - B$, then we define M^+ by

$$
M^+ = L(\overline{DU-B}). \tag{3.4}
$$

For the choice of M^+ from (3.4), Lemma 3.1 translates to

LEMMA 3.2. *Let $||A - LDU|| = \varepsilon||A|| < ||B||$. Then the preconditioner from (3.4) satisfies*

$$
\begin{aligned}
||A^+ - M^+|| &\leq ||A^+ - LDU|| \frac{||L(DU - \overline{DU-B}) - B|| + \varepsilon||A||}{||B|| - \varepsilon||A||} \\
&\leq ||A^+ - LDU|| \frac{\|L\| \, \|DU - B - \overline{DU-B}\| + ||L - I|| \, \|B\| + \varepsilon||A||}{||B|| - \varepsilon||A||}.
\end{aligned}
$$

Again, a good approximation $\overline{DU-B}$, combined with a nearly diagonal factor L, may yield a powerful preconditioner.

In the following subsection we propose approximations of $DU - B$ that can be efficiently computed and that lead to preconditioners that are inexpensive to apply. We will always assume that the approximation $\overline{DU-B}$ is nonsingular. In the section thereafter we address possible instability of the proposed preconditioners and mention a way to overcome it. Of course, all techniques we treat can be analogously formulated for updates of the form $(\overline{LD-B})U$.

3.1. Proposed preconditioner updates. We now propose basic choices for the approximation $\overline{DU - B}$ in (3.4). We will see that the introduced algorithms can be used to approximate the matrix $\overline{D - B}$ for the preconditioner (3.2) as well.

3.1.1. Triangular updates. Here we describe choices which are directly obtained from sparsification of $DU - B$. A very simple choice of $\overline{DU - B}$ for M^+ in (3.4) is

$$\overline{DU - B} \equiv triu(DU - B), \tag{3.5}$$

where *triu* denotes the upper triangle (including the main diagonal). From Lemma 3.2 we get that M^+ is accurate, assuming $L \approx I$, if the upper triangle of B contains an important part of the whole difference matrix B. This seems to be the case if the difference matrix is rather nonsymmetric as in upwind/downwind perturbations in nonlinear convection-diffusion problems. We show in section 4 that this preconditioner can be very powerful in such applications. Even in cases when this M^+ can be readily applied, we might consider ways to improve efficiency of the backward solve by sparsification if the factor U is rather dense. This sparsification is well justified if the update brings new important information represented by its large entries. In our experiments with a nonlinear convection-diffusion problem, we tried to enhance the effectiveness of M^+ only by structured sparsifications, considering the most important subdiagonals. Denoting by the subindices $[i_1, \ldots, i_l]$ the l upper subdiagonals that start in columns $i_1, \ldots, i_l$ (and the main diagonal by the subindex 0) we considered also choices of the form

$$\overline{DU - B} \equiv (DU - B)_{[i_1,\ldots,i_l]}. \tag{3.6}$$

In particular, if the entries of B dominate those of DU (in magnitude) we may choose only indices corresponding to upper subdiagonals of B (the difference matrix is sparse). Further simplification eventually leads to

$$\overline{DU - B} \equiv \mathrm{diag}(DU - B), \tag{3.7}$$

which still yields a useful update in some applications and which is a straightforward generalization of the approach from [2] for solving a more general problem.

Similarly, we can consider triangular approximations $\overline{D - B}$ of $D - B$ in the preconditioner (3.2), but this is theoretically and, as we will see, also experimentally inferior with respect to the above mentioned choices.

The costs to compute M^+ in (3.4) for all the choices (3.5), (3.6) and (3.7) are negligible. The additional cost for applying the preconditioner from (3.5) is one triangular sweep with the triangle of B, if we store B and U separately and do not merge them. Applying other strategies is even cheaper.

The presented strategies are strongly based on confining the update to the upper (or, equivalently, lower) triangle. Whereas numerical experiments seem to indicate this makes sense, there may be applications where it is necessary to take into account both triangles of the difference matrix. The next subsection is devoted exactly to this case.

3.1.2. General updates. Here we introduce a strategy to approximate $DU - B$ by a generally non-triangular but easily invertible matrix. Denote the matrix $\mathrm{diag}(\overline{DU - B})$ by $\tilde{D}$, and $\tilde{D}^{-1}(\tilde{D} - \overline{DU - B})$ denote by $\tilde{B}$. Then $\tilde{B}$ has zero diagonal and we can write

$$\overline{DU - B} = \tilde{D}(I - \tilde{B}). \tag{3.8}$$

To motivate the scaling transformation in (3.8) consider the case when $\tilde{B} = \beta e_i e_j^T$, for some $1 \le i, j \le n, i \ne j$, and recall we assume $\overline{DU - B}$ is nonsingular, hence so is $I - \tilde{B}$. Then we get

$$(I - \tilde{B})^{-1} = I + \beta e_i e_j^T / (1 - \beta e_j^T e_i) = I + \beta e_i e_j^T. \tag{3.9}$$

The matrix (3.9) is equal to a unit matrix modified by an offdiagonal entry β at the position (i, j). That is, $(I - \tilde{B})$ is a special Gauss-Jordan transformation [17], and it has a fill-in free inverse.

Based on this observation, in the following we will try to find approximations $\overline{DU - B}$ such that $\overline{DU - B} \approx DU - B$ and that the scaled matrix $I - \tilde{B}$ can be written as a product of Gauss-Jordan transformations

$$(I - e_{i_1} \tilde{b}_{i_1 *})(I - e_{i_2} \tilde{b}_{i_2 *}) \dots (I - e_{i_K} \tilde{b}_{i_K *}), \tag{3.10}$$

where $\tilde{B} = (\tilde{b})_{ij}$. Denote the sparsity structure of a row i of $\tilde{B}$ (with zero diagonal) by $row(i)$, that is, $row(i) = \{k | i \ne k \wedge \tilde{b}_{ik} \ne 0\}$. The multiplication $(I - \tilde{B})v$ for a given vector v is very cheap, as stated in Observation 3.1.

OBSERVATION 3.1. *The number of operations for multiplying a vector by a matrix of the form (3.10) or its inverse is at most* $2 \sum_{j=1}^{K} |row(i_j)|$.

It is well known that if B is an upper triangular matrix, then we can set $\overline{DU - B} = DU - B$ since any unit triangular matrix $I - \tilde{B}$ from (3.8) can be trivially written as a product of $n - 1$ elementary triangular matrices. For example, $I - \tilde{B} = R_{n-1} \dots R_1$ with $R_i = I - e_i \tilde{b}_{i*}$ for $i = 1, \dots, n - 1$ if $I - \tilde{B}$ is a unit *upper* triangular matrix. In some other cases, only a part of a matrix can be written as a product of Gauss-Jordan transformations. Consider, e.g., the case of a unit lower triangular matrix $I - \tilde{B}$ with k full additional subdiagonals in its upper triangular part determined by the nonzero entries in its first row $\tilde{b}_{1,l+1}, \dots, \tilde{b}_{1,l+k}$. Then a product of Gauss-Jordan transformations with a fill-in free inverse can cover the first l rows of the matrix, l rows starting from the row $2l + k$ and so on. If $n \bmod 2l + k - 1 = 0$ then only $l/(2l + k - 1)$ percent of the rows can be covered by one such product.

Changes in a sequence of matrices restricted to a couple of diagonals are rather frequent. Nevertheless, if the difference has to be reasonably well approximated by one sweep of the form (3.10) the choice of Gauss-Jordan transformations may be more sophisticated. The following theorem shows a necessary and sufficient condition for the existence of a decomposition of $I - \tilde{B}$ of the form (3.10).

THEOREM 3.1. *Let* $I - \tilde{B} = I - \sum_{j_l : l = 1, \dots, K} e_{j_l} \tilde{b}_{j_l *}$. *Then*

$$I - \tilde{B} = (I - e_{i_1} \tilde{b}_{i_1 *})(I - e_{i_2} \tilde{b}_{i_2 *}) \dots (I - e_{i_K} \tilde{b}_{i_K *}) \tag{3.11}$$

if and only if

$$i_l \notin \bigcup_{k=1}^{l-1} row(i_k) \text{ for } 2 \le l \le K \tag{3.12}$$

for all $i_1, \dots, i_K$ *such that* $\{j_1, \dots, j_K\} = \{i_1, \dots, i_K\}$.

Proof: The equivalence of (3.11) and (3.12) follows from the orthogonality of the unit vector e_{i_l} with respect to all $\tilde{b}_{i_k *}$ for $k < l$, $1 \le l \le K$. □

An elegant systematic way to get an update based on Gauss-Jordan transformations can be described by a bipartite graph model $G(DU - B) = (R, C, E)$ of $DU - B$ where $R = \{1, \dots, n\}$, $C = \{1', \dots, n'\}$ and $E = \{(i, j') | (DU - B)_{ij} \ne 0\}$.

THEOREM 3.2. *Consider a spanning forest $T = (V_T, E_T)$ of $G(DU-B)$ such that $\{(i,i')|1 \le i \le n\} \subseteq E_T$. Then the matrix $\overline{DU-B} \in \mathbb{R}^{n\times n}$ with the entries defined by $\overline{(DU-B)}_{ij} = \begin{cases} (DU-B)_{ij} & \text{if } (i,j') \in E_T \\ 0 & \text{otherwise} \end{cases}$, scaled by its diagonal entries as in (3.8), can be expressed as a product of the form (3.10).*

Proof: First consider the case when the spanning forest T is not connected. Components of T induce a block diagonal splitting of $\overline{DU-B}$, and matrices corresponding to individual blocks can be mutually multiplied in any order without causing any fill-in. Consequently, we can assume without loss of generality that T is connected and that T is a spanning tree. In the following we will show how to form the sequence of Gauss-Jordan transformations from the left to the right.

Our assumption implies that T contains at most $n-1$ edges (i,j') with $i \neq j$. There exists a free row vertex $i \in R$ in T which is in T incident only to the edge (i,i') such that there is an edge $(k,i') \in E_T$ for some k. Set $i_1 = i$. Then remove from T the vertices $i \in R$, $i' \in C$ and all edges incident to them. Clearly, the updated tree T contains a free row vertex again. By repeating the choice of free row vertices and updates T in this way we get the sequence $i_1, \ldots, i_{n-1}$. If we rewrite as $I - \tilde{B}$ the matrix $\overline{DU-B}$ scaled by its diagonal we have $I-\tilde{B} = (I-e_{i_1}\tilde{b}_{i_1*})(I-e_{i_2}\tilde{b}_{i_2*})\ldots(I-e_{i_{n-1}}\tilde{b}_{i_{n-1}*})$ which proves the theorem. □

Theorem 3.2 implies the following algorithmic strategy to find a matrix $\overline{DU-B}$ which would approximate $DU-B$ and could be expressed as a product of Gauss-Jordan transformations.

ALGORITHM 3.1. *Algorithm to find $\overline{DU-B}$ such that (3.11) is satisfied based on a bipartite graph of $DU-B$.*

(1) Find a spanning forest $T = (V_T, E_T)$ of $G(DU-B)$ of maximum weight with edge weights $w_{ij} = |(DU-B)_{ij}|$ for $(i,j') \in E_T$ such that $\{(i,i')|1 \le i \le n\} \subseteq E_T$.

(2) Find the entries of $\tilde{B}$ (and corresponding entries of $\overline{DU-B}$ of the difference matrix) as well as a feasible ordering of Gauss-Jordan factors for $i_1, \ldots, i_{n-1}$ in (3.10).

(3) For each $k = 2, \ldots, n$ add to $\overline{DU-B}$ all entries $(DU-B)_{i_k l}$ of $DU-B$ such that $l \in \{i_1, \ldots, i_{k-1}\}$.

Note that in the last step of Algorithm 3.1 we possibly put into $\overline{DU-B}$ much more nonzero entries than the $2n-1$ entries provided by the weighted spanning forest. The condition is based on Theorem 3.1. The complexity of the weighted minimum spanning forest (here we need, in fact, a weighted maximum forest) is $O(m \log m)$ for the Kruskal algorithm [18] and $O(n + m \log m)$ for the Prim algorithm [33], where m is the number of edges in the graph G. While in some cases the algorithms may seem time consuming, this procedure can provide useful updates. Note, in addition, that we start with the partial spanning tree with the set of edges $\{(i,i')|1 \le i \le n\}$.

Another approach to find a suitable approximation $\overline{DU-B}$ with $I-\tilde{B}$ based on Gauss-Jordan transformations is to use a simple greedy procedure based on Theorem 3.1. It is described in Algorithm 3.2. Consider a sequential choice of indices $i_1, \ldots, i_K$, where $K \le n-1$ will be determined by the algorithm. In each step we keep and update a set of *candidate rows* $\mathcal{R}$ initialized by $\{1, \ldots, n\}$. After choosing a row i we remove from $\mathcal{R}$ all the rows $j \in \mathcal{R}$ for which $\tilde{b}_{ij} \neq 0$. Here the sparsification of $DU-B$

(removal of its "small" entries) does not need to be done in advance. Instead, it can be performed on-the-fly when running the algorithm.

ALGORITHM 3.2. *Algorithm to determine Gauss-Jordan transformations such that the scaled matrix* $\overline{DU - B}$ *can be written in the form (3.11).*

(1) *set* $\mathcal{R} = \{1, \ldots, n\}$, $K = 0$
(2) *for* $k = 1, \ldots, n$ *do*
(3) *set* $row(k) = \{i | i \neq k \wedge |(DU - B)_{ki}| > tol\}$
(4) *set* $p_k = \sum_{j \in row(k)} |(DU - B)_{kj}|$
(5) *end for*
(6) *while* $\mathcal{R} \neq 0$ *do*
(7) *choose a row* $i \in \mathcal{R}$ *maximizing* $p_i - \sum_{j \in \mathcal{R} \cap row(i)} p_j$
(8) *set* $K = K + 1$, $i_K = i$
(9) *set* $\mathcal{R} = \mathcal{R} \backslash \{row(i_K) \cup i\}$
(10) *end while*

The indices of Gauss-Jordan transformations provided by Algorithm 3.2 then determine the approximation in (3.8) with $I - \tilde{B}$ equal to the product (3.10).

Both algorithmic strategies explained in this subsection can be directly applied to approximate $D - B$ for (3.2) by Gauss-Jordan transformations replacing $DU - B$ by $D - B$ in Algorithms 3.1 and 3.2, and scaling the approximation similarly.

3.2. Coping with possible instabilities. So far, we have assumed all choices (3.5), (3.6), (3.7) and those generated by Algorithms 3.1 and 3.2 are nonsingular but this need not be so. Even worse, the application of (3.4) can become unstable for nonsingular choices of $\overline{DU - B}$ in the same way as incomplete decomposition can be unstable. For triangular updates this may happen whenever the off-diagonal entries of $DU - B$ are significantly larger than diagonal entries, even for perfectly stable original incomplete decompositions $M = LDU$. Applying stabilization strategies to the initial system, such as finding a maximal transversal [3], cannot guarantee to overcome the instability encountered here. We suggest a slight modification of (3.4) that is very natural in the context of updating preconditioners.

The "ideal" update of section 2 is $M^+ = LDU - B$. As LDU approximates A, we have

$$M^+ = LDU - B \approx A - B = A^+,$$

hence we may expect M^+ is far from being singular provided LDU is close to A. Moreover, this M^+ inherits diagonal dominance of A^+. If A^+ fails to be diagonally dominant, we can preprocess it with an appropriate permutation. It seems reasonable to modify (3.4) as

$$M^+ = L\,\overline{(DU - L^{-1}B)}, \tag{3.13}$$

where $\overline{(DU - L^{-1}B)}$ is close to $(DU - L^{-1}B)$. Certainly, the approximation of $(DU - L^{-1}B)$ is more expensive than the approximation of $(DU - B)$ treated before. Nevertheless, in practice it can be still feasible.

A very cheap approximation of $(DU - L^{-1}B)$ is obtained by scaling the upper triangle of B as

$$\overline{(DU - L^{-1}B)} \equiv DU - \mathrm{diag}(||Le_1||, \ldots, ||Le_n||)^{-1} triu(B). \tag{3.14}$$

Another choice,

$$\overline{(DU - L^{-1}B)} \equiv DU - triu(L^{-1}B), \tag{3.15}$$

seems expensive at first sight due to the product $L^{-1}B$. But exploiting the sparsity of B, the triangularity of L and the fact that we need only one triangle of the product, computing $triu(L^{-1}B)$ can be done effectively. For example, if B arises from standard 2D central difference discretization, then the number of multiplications needed is at most $n(\sqrt{n}+1)$. For localized large entries in the update matrix, approximation to $triu(L^{-1}B)$ can be even cheaper. Straightforward reduction of these costs follows from considering

$$\overline{DU - L^{-1}B} \equiv (DU - L^{-1}B)_{[i_1,\ldots,i_l]}, \tag{3.16}$$

for a small number of positions i_1 to i_l. It is easy to see that when the positions are chosen to correspond to the nonzero upper subdiagonals of B, then the computation of this approximation of $DU - L^{-1}B$ is comparable to executing one matvec with B.

Finally, in analogy with the preceding subsection, it may be advantageous to take into account both triangles of the update. This can be achieved by applying Algorithms 3.1 or 3.2 to

$$(DU - L^{-1}B)_{[i_1,\ldots,i_l]}, \tag{3.17}$$

where we allow negative indexes to denote *lower* subdiagonals starting on the corresponding row (as in Matlab).

The next section is devoted to numerical experiments with the most promising updates introduced in the paper.

4. Numerical experiments. In this section we report on numerical experiments aimed at assessing the performance of preconditioned Krylov subspace methods for solving sequences of systems of linear algebraic equations where preconditioners are updated instead of recomputed. The sequences were generated with the optimization software from [25]. We always consider the entire sequence of linear problems needed to reduce the nonlinear residuals to the tolerance defined in [25]. The software uses Fortran 90 double precision arithmetic. Preconditioners and their updates were written both in Fortran90 (ILUT preconditioners) and in Matlab (drop tolerance-based ILU), and we present results of codes from both environments. The codes were run on computers with Intel processors.

As an accelerator, the BiCGSTAB [39] iterative method was used. We also performed experiments with the restarted GMRES method [34]. The results were similar and we do not report on them here. Iterations were stopped when the Euclidean norm of the residual was decreased by 7 orders of magnitude. Nevertheless, in our experiments we observed close to linear behavior of convergence curves of the preconditioned iterative method. Therefore, we might expect qualitatively the same results for weaker or nonuniform stopping criteria used in nonlinear solvers. Except for one experiment, the BiCGSTAB method was preconditioned from the right.

Most of the experiments which we present here are related to the following two nonlinear problems. The first of them is the two-dimensional nonlinear convection-diffusion problem (Kelley, 1995)

$$\Delta u - Ru\nabla u = 2000x(1-x)y(1-y),\ R = 50, \tag{4.1}$$

TABLE 1
Number of iterations for preconditioned nonlinear convection-diffusion problem with a frozen preconditioner.

A	M	*its/NO*
A^1	M^1	21
A^2	M^1	29
A^3	M^1	39
A^4	M^1	52
A^5	M^1	77
A^6	M^1	80
A^7	M^1	102
A^8	M^1	102
A^9	M^1	98
A^{10}	M^1	101
A^{11}	M^1	99
A^{1-11}	M^{1-11}	21 ± 5

on the unit square, which was discretized by 5-point finite differences on a uniform grid. The initial approximation was the discretization of $u_0(x, y) = 0$. We chose a moderate Reynolds number in order to avoid potential discretization problems such as the necessity to add stabilization terms.

Our second nonlinear test problem is a finite difference analogue of the following porous media nonlinear equation [15] solved over the unit square with zero Dirichlet boundary conditions

$$\nabla u^2 + R\left(\frac{\partial u^3}{\partial x} + f(x, y)\right) = 0. \tag{4.2}$$

The function $f(x, y)$ was evaluated in order to have a simple analytic form of the solution, namely $u = x(x-1)y(y-1)$. This enabled us to check that our sequence solves not only the discretized problem but also the physical one. The initial approximation was a discretization of $u_0(x, y) = 1 - xy$. As above, $R = 50$. We chose this second test problem in order to demonstrate potential instabilities in updated decompositions.

Table 1 contains results for the nonlinear convection-diffusion problem (4.1) discretized on a 70×70 grid in case of *freezing* the preconditioner. Namely, we used the same ILUT(0.1,5) preconditioner for all the systems. By "A" we denote the matrix with the superscript denoting its order in the sequence, by "M" we denote the preconditioner with the superscript denoting its order in the sequence, and "its / NO" denotes the number of BiCGSTAB iterations preconditioned by M without any updates. The last line of the table shows that iteration counts for solving systems with preconditioners computed from their system matrices, that is A^i preconditioned by M^i for $i = 1, \ldots, 11$, do not deteriorate. They stay always approximately equal to 21. Clearly, we can conclude that freezing one preconditioner may not be enough for getting efficiently preconditioned iterative methods for all the systems.

In Table 2 we present results obtained for two simple *linear* 2D convection-diffusion problems discretized with a 5-point discretization stencil. The two problems are related by a 1D constant convection shift. More precisely, the matrix of the first system is the 2D Laplacian, and the second problem is obtained from the first one by a modification of entries by shifts on its main diagonal and one upper subdiagonal.

TABLE 2
Number of iterations for 2D linear convection-diffusion for a problem with a 1D convection shift with/without an update based on GJ transformations.

$shift$	its/NO	its/GJ
0.1	34	33
0.2	36	35
0.3	39	34
0.4	38	33
0.5	44	34
0.6	56	38
0.7	53	34
0.8	63	34
0.9	69	30

These shifts are constant along these diagonals and they have the same value and opposite signs. This is an idealized case of what may happen when solving a sequence of problems from a nonlinear convection-diffusion problem such as (4.1). The dimension of the problem is $n = 10000$. By "shift" we denote the shift value, by "its / NO" we denote the number of iterations for solving the second, shifted system by BiCGSTAB preconditioned by the ILUT(0.1,5) preconditioner for the first system. For solving the shifted system we also updated the preconditioner according to (3.2), based on the incomplete factors from the first system. The matrix $(D - B)$, where B contains one diagonal and one subdiagonal, is written as a product of Gauss-Jordan (GJ) transformations. By "its / GJ" we denote the corresponding number of iterations needed by the BiCGSTAB method. Note that in this case $D - B$ is triangular, hence we could as well multiply with $(D - B)^{-1}$ through backward solves. Table 2 presents one of more experiments of this kind which we performed, and it represents the situation when one of the systems was more difficult to solve and needed more iterations than the other one. Solving the system with recomputed preconditioners needs 35 ± 1 iterations. Clearly, the GJ-based update of the preconditioner is rather helpful for larger shifts. In addition, the preconditioner is very cheap both to compute and to apply. If both systems were very easy to solve (with a weak convection) then the update did not bring an overall improvement.

Excellent behavior of the triangular (TR) updates with the choice (3.5) for the preconditioner M^+ from (3.4) is demonstrated by results in Table 3 for solving the nonlinear convection-diffusion problem (4.1) on the 70×70 grid. Under "A / M" we display which system matrix and preconditioner from the sequence of problems we used. The two other columns give the number of iterations in case of no updates ("its / NO") and triangular updates ("its / TR") (3.5), respectively. As a preconditioner we used ILUT(0.1,5). In this case, the number of nonzeros in the system matrices was 24220, the size of the incomplete LU decomposition was 19320. There is a strong overlap between the location of the nonzeros in B and in the preconditioner. Consequently, we could spare some space by merging $triu(B)$ into DU but we did not do this. Instead, we kept the update matrix implicitly. In any case, we can conclude that the updated preconditioner perfectly replaces the one which would be obtained by a new ILUT decomposition.

In Table 4 we compare the efficiency of approaching $(DU - B)$ according to (3.5) (by TR updates (3.4)) and $(D - B)$ according to (3.2) (with GJ updates using

TABLE 3
Number of iterations for solving preconditioned linear systems of a nonlinear convection-diffusion problem with no updates and triangular updates, respectively.

A/M	its / NO	its / TR
A^1 / M^1	21	21
A^2 / M^1	29	25
A^3 / M^1	39	27
A^4 / M^1	52	25
A^5 / M^1	77	25
A^6 / M^1	80	26
A^7 / M^1	102	26
A^8 / M^1	102	27
A^9 / M^1	98	27
A^{10} / M^1	101	26
A^{11} / M^1	99	26
A^{1-11} / M^{1-11}	21 ± 5	—

Algorithm 3.2) for Problem 4.1 discretized on a 50×50 grid. The table presents results with left-preconditioned BiCGSTAB for another preconditioner M^1, namely $ILU(10^{-3})$ from Matlab. Both L and U have approximately 35000 nonzeros, the TR update uses the same space, the GJ update adds 5500 nonzeros. Both the TR and GJ updates are very cheap since they need a very small number of additional operations in each iteration. In addition to the notation from Table 3 we use "its / GJ" to denote the number of iterations when GJ updates of M^1 were used. Algorithm 3.2 used the parameter $tol = 0.1$. Clearly, the approach of (3.5) is more powerful in this case than the approach of (3.2). This confirms the theory from Section 3 according to which it is advantageous to exploit at only one of the triangular factors that it is close to diagonal. In the Frobenius norm we have here

$$\frac{\|I - L\|}{\|L\|} = 0.434 = \frac{\|I - U\|}{\|U\|} .$$

If we would apply (3.5) with GJ updates and (3.2) with TR updates we would probably obtain similar results.

Tables 5 and 6 present results for the same problem but now the BiCGSTAB method is preconditioned from the right. GJ transformations computed by Algorithm 3.2 approximate $(DU - B)$ in this case, the TR update is used as above, according to (3.5). The results in Table 5 were computed for the $ILU(10^{-2})$ and $ILU(10^{-3})$ preconditioners whereas the results in Table 6 were computed for $ILU(10^{-1})$ to get an idea how successful the GJ updates can be when a very sparse matrix is to be covered by Gauss-Jordan transformations. It is readily seen from Theorem 3.1 that the sparser a matrix, the larger the potential part that can be covered by Gauss-Jordan transformations. This also motivated our relatively large drop tolerance $tol = 0.1$ in Algorithm 3.2. With $ILU(10^{-2})$ and $ILU(10^{-3})$ as initial preconditioners TR updates needed less iterations than GJ updates. In addition, TR updates are computed without any effort whereas some costs are made while computing GJ updates with Algorithm 3.2. But GJ updates are advantageous for a different reason: The sizes of the factors L and U are for the drop tolerances 10^{-3}, 10^{-2} and 10^{-1} equal to approximately 35000, 12000 and 5000, respectively. The sizes of the GJ updates

TABLE 4
Number of iterations for the preconditioned nonlinear convection-diffusion problem with preconditioner updated by GJ updates applied to approximate $(D - B)$ for (3.2) and TR updates (3.5) for (3.4), respectively.

A/M	its / NO	its / GJ	its / TR
A^1 / M^1	5	5	5
A^2 / M^1	31	36	14
A^3 / M^1	51	40	13
A^4 / M^1	71	51	15
A^5 / M^1	91	59	15
A^6 / M^1	97	63	14
A^7 / M^1	100	64	16
A^8 / M^1	97	70	16
A^9 / M^1	103	65	16
A^{10} / M^1	100	76	17
A^{11} / M^1	99	71	16

TABLE 5
Number of iterations for the preconditioned nonlinear convection-diffusion problem with preconditioner updated by Gauss-Jordan updates applied to $(DU - B)$ and triangular updates, respectively

A / M	$ILU(10^{-3})$			$ILU(10^{-2})$		
	its / NO	its / GJ	its / TR	its / NO	its / GJ	its / TR
A^1 / M^1	6	6	6	13	13	13
A^2 / M^1	37	17	14	32	20	17
A^3 / M^1	61	18	16	58	23	17
A^4 / M^1	78	21	17	89	24	17
A^5 / M^1	104	21	17	127	23	17
A^6 / M^1	109	23	16	131	24	18
A^7 / M^1	132	21	17	182	25	18
A^8 / M^1	132	23	18	172	26	19
A^9 / M^1	113	22	16	157	22	18
A^{10} / M^1	120	22	19	166	24	18
A^{11} / M^1	127	22	17	163	24	19

are in the range $< 6700, 7800 >$ for all tolerances. Taking this into account, we can conclude that application of GJ based preconditioners is cheaper than application of TR updates, especially for the drop tolerance 10^{-3}. This is because in case of GJ updates we replace each solve with the factor U by a very cheap procedure and save the substitution step with U in each solve.

For $ILU(10^{-2})$ the departure of L from identity can be expressed as

$$\frac{\|I - L\|}{\|L\|} = 0.423$$

in the Frobenius norm. For $ILU(10^{-1})$ we have

$$\frac{\|I - L\|}{\|L\|} = 0.397.$$

Finally, we noted that the scaling (3.14) improves slightly, but not significantly, the

TABLE 6
Number of iterations for the preconditioned nonlinear convection-diffusion problem with preconditioner updated by Gauss-Jordan updates applied to $(DU - B)$ and by triangular updates, respectively, different preconditioner than in Table 5

A / M	$ILU(10^{-1})$		
	its / NO	its / GJ	its / TR
A^1 / M^1	24	24	24
A^2 / M^1	27	26	24
A^3 / M^1	38	27	21
A^4 / M^1	47	25	23
A^5 / M^1	52	22	23
A^6 / M^1	58	21	22
A^7 / M^1	68	22	23
A^8 / M^1	91	24	24
A^9 / M^1	70	20	23
A^{10} / M^1	68	22	24
A^{11} / M^1	76	24	25

power of the TR updates.

Potential instabilities of the basic updating procedures are demonstrated on the results from the difficult nonlinear Problem 4.2. Table 7 presents overview results with different GJ-based and TR-based updates on a 50×50 grid. In order to show more variants of updates we present results for one matrix A^+ (A^{10}) of the sequence. The preconditioner is Matlab-generated ILU(10^{-3}) which provides powerful but costly preconditioning. Its recomputation is rather expensive. It has a much larger number of nonzeros (approximately 100 000) than the original matrix (12300). In the Frobenius norm, $\frac{\|I-L\|}{\|L\|} = 0.349$. Although cheaper ILU decompositions are often more practical, we chose these parameters in order to get a clear view on various variants of the updates in case of instabilities. Preconditioning of the matrix A^+ by the simple GJ preconditioner based on approximation of $DU - B$ for (3.4) with the ILU preconditioner for A^1 is not successful since the iterative method does not converge. The same is true for the TR update (3.5). In both cases the updated preconditioners fail to be stable due to lack of diagonal dominance. By "Update_type" we denote the variant of the update which was used to overcome the problem. The matrix in this column was used to provide a triangular or a GJ-based update. If the matrix was triangular, TR update was used. Otherwise, we computed a GJ update using Algorithm 3.2. The only exception is the last row of Table 7 in which we used the GJ update based on Algorithm 3.1 where the weighted spanning tree was computed by Kruskal's algorithm [18]. Before running the algorithm, the matrix was sparsified with the drop tolerance $\tau = 0.2$. As mentioned above, superscripts of methods in Table 3 on rows 3 to 8 denote the restriction to certain subdiagonals (e.g., 0: main diagonal, -50: -50-th subdiagonal) given in the MATLAB notation. The triangular or GJ-updates were chosen from these subdiagonals only. In the GJ case, all the matrices were sparsified on-the-fly using the drop tolerance $\tau = 0.2$ before the GJ updates were computed. The size of all the GJ updates was around 7200. Since they effectively replaced half of the substitution with the much larger factor U (seven times), and there are two solves in each step of BiCGSTAB, the updated procedure is competitive in more cases. We can see here that the spanning tree-based update is not so powerful as some other

TABLE 7
Number of iterations for solving systems from Problem 4.2 with the system matrix $A^+ = A^{10}$ for different variants of an update of the preconditioner for A based on triangular and Gauss-Jordan transformations.

$Update\_type$	its
$DU - triu(L^{-1}B)$	45
$DU - diag(L^{-1}B)$	∞
$(DU - L^{-1}B)_{[0,1,50]}$	48
$(DU - L^{-1}B)_{[0,1]}$	78
$(DU - L^{-1}B)_{[0]}$	101
$(DU - L^{-1}B)_{[-1:1,50]}$	50
$(DU - L^{-1}B)_{[-50,-1:1,50]}$	44
$(DU - L^{-1}B)_{[-50,-10:10,50]}$	43
Algorithm 3.1	81

TABLE 8
Number of iterations for solving systems from Problem 4.2 with the system matrix A^+ for stabilized TR and GJ updates explained in the text.

A / M	its / NO	its / GJ	its / TR
A^1 / M^1	3	3	3
A^2 / M^1	7	8	5
A^3 / M^1	10	22	11
A^4 / M^1	16	14	13
A^5 / M^1	26	18	17
A^6 / M^1	35	21	20
A^7 / M^1	51	29	25
A^8 / M^1	51	32	33
A^9 / M^1	∞	50	43
A^{10} / M^1	∞	45	49
A^{11} / M^1	∞	40	39
A^{12} / M^1	∞	44	42
A^{13} / M^1	∞	39	44
A^{14} / M^1	∞	44	44
A^{15} / M^1	∞	39	43
A^{16} / M^1	∞	43	48

approaches. On the other hand, we consider as the best variants those which restrict additional computations only to a part of $L^{-1}B$, since this product may be sometimes time-consuming. Strengths and weaknesses of the updates for the Problem 4.2 are clearly visible from Table 8. Here we present comparison of results for two stabilized updates, namely the TR update based on the matrix $(DU - L^{-1}B)_{[0,1,50]}$ and the GJ update based on approximating $(DU - L^{-1}B)_{[-50,-1:1,50]}$ using Algorithm 3.2. The notation of columns is the same as above. The nonstabilized updates (that is those omitting the factor L^{-1} in the update) could be used only for preconditioning the first three systems and provided similar results as the stabilized ones. We can see that both types of updates should be used only if the preconditioners M are very costly to recompute. In addition, the computation of the update which involves the sparsified product $L^{-1}B$ is cheap only if at least one of the factors L or B is sparse,

or if, for example, nonzeros in B are local (their indices can be embedded into a short interval).

We performed also some experiments where our nonlinear problems where discretized by upwind schemes, leading to triangular difference matrices. As one can guess, the results for solving the linear problems were rather good, but we typically needed more nonlinear iterations. Also, our intention here was to show that TR and GJ updates can be very useful even for more general than triangular difference matrices, or systems in which the changes in nonlinear steps are mostly in one matrix triangle.

5. Conclusions. In this paper we proposed a couple of algebraic procedures which may be useful for solving sequences of systems of linear equations. The numerical experiments confirm that our updated preconditioners can be rather successful in practice, and the updates can often replace recomputation of preconditioners. In many cases, one would like to make the overall number of operations smaller with simple updates, and our experiments confirm that this is possible. Nevertheless, there can be also different, and sometimes very strong, reasons for avoiding preconditioner recomputations. In matrix-free and/or parallel environments, which are nowadays quite common, any recomputation of a preconditioner may be expensive. This is especially true for strong algebraic preconditioners which are used for solving difficult problems.

An interesting problem which was not considered here is to choose triangular updates which correspond to the sparsity pattern and sizes of entries of the difference matrix *differently* for each system of the sequence. Namely, we have the option to choose in each separate step either a lower, or an upper triangular, or even a different update. A closely related problem which we will consider in the future is to find a nonsymmetric permutation which would transform the system matrices into a form which could be more suitable for one particular triangular or GJ-based update.

Acknowledgment. The authors thank Ladislav Lukšan for providing the software [25] and for useful instructions to work with it.

REFERENCES

[1] O. Axelsson and L. Yu. Kolotilina. Diagonally compensated reduction and related preconditioning methods. *Numer. Linear Algebra Appl.*, 1:155–177, 1994.

[2] M. Benzi and D. Bertaccini. Approximate inverse preconditioning for shifted linear systems. *BIT Numer. Math.*, 43:231–244, 2003.

[3] M. Benzi, J.C. Haws, and M. Tůma. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM J. Sci. Comput.*, 21:1333–1353, 2000.

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

[5] L. Bergamaschi, R. Bru, A. Martinez, and M. Putti. Quasi-Newton preconditioners for the inexact Newton method. In *Abstract book of the 2005 International Conference On Preconditioning Techniques, Atlanta*, May 19–21, 2005.

[6] D. Bertaccini. Efficient preconditioning for sequences of parametric complex symmetric linear systems. *Electronic Transactions on Numerical Mathematics*, 18:49–64, 2004.

[7] R. P. Brent. Some efficient algorithms for solving systems of nonlinear equations. *SIAM J. Numer. Anal.*, 10:327–344, 1973.

[8] E. Chow and Y. Saad. Experimental study of ILU preconditioners for indefinite matrices. *J. Comput. Appl. Math.*, 86:387–414, 1997.

[9] J. Cullum and M. Tůma. Matrix-free preconditioning using partial matrix estimation. Technical Report V-898, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2003.

[10] A.R. Curtis, M.J.D. Powell, and J.K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974.
[11] T.A. Davis and W.W. Hager. Modifying a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 20:606–627, 1999.
[12] T.A. Davis and W.W. Hager. Multiple-rank modifications of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 22:997–1013, 2001.
[13] 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.*, 20:889–901, 1999.
[14] I. S. Duff and J. Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM J. Matrix Anal. Appl.*, 22:973–996, 2001.
[15] S.C. Eisenstat and H.F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comput.*, 17:16–32, 1996.
[16] A.H. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review, to appear*, 2005.
[17] G. H. Golub and C. F. Van Loan. *Matrix Computations. 3rd ed.* The Johns Hopkins University Press, Baltimore and London, 1996.
[18] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. of the AMS*, 2:48–50, 1956.
[19] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.
[20] C.T. Kelley. *Solving Nonlinear Equations with Newton's Method*. SIAM, Philadelphia, 2003.
[21] D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *J. Comp. Phys.*, 193:357–397, 2004.
[22] D.A. Knoll and P.R. McHugh. Newton-Krylov methods applied to a system of convection-reaction-diffusion equations. *Comput. Phys. Commun.*, 88:141–160, 1995.
[23] D.A. Knoll, P.R. McHugh, and D.E. Keyes. Newton-Krylov methods for low Mach number compresible combustion. *AIAA Journal*, 34:961–967, 1996.
[24] D. Loghin, D. Ruiz, and A. Touhami. Adaptive preconditioners for nonlinear systems of equations. Technical Report TR/PA/04/42, Cerfacs, Toulouse, 2004.
[25] L. Lukšan, M. Tůma, J. Vlček, N. Ramešová, M. Šiška, J. Hartman, and C. Matonoha. UFO 2004 - interactive system for universal functional optimization. Technical Report V-923, ICS AS CR, 2004.
[26] J.L. Morales and J. Nocedal. Automatic preconditioning by limited-memory quasi-Newton updates. *SIAM J. Opt.*, 10:1079–1096, 2000.
[27] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.*, 29:293–322, 1980.
[28] M. Olschowka and A. Neumaier. A new pivoting strategy for Gaussian elimination. *Linear Algebra Appl.*, 240:131–151, 1996.
[29] M.L. Parks, E. de Sturler, G. Mackey, D.D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. Technical Report UIUCDCS-R-2004-2421, University of Illinois, 2004.
[30] P.E. Gill, W. Murray and M.A. Saunders. Methods for computing and modifying the LDV factors of a matrix. *Math. Comput.*, 29(132), 1975.
[31] P.F. Fischer. Projection techniques for iterative solution of $Ax = b$ with successive right-hand sides. *Comp. Meth. Appl. Mech. Engng.*, 163:193–204, 1998.
[32] P.N. Brown and Y. Saad. Hybrid Krylov methods for solving systems of nonlinear equations. *SIAM J. Sci. Stat. Comput.*, 11:450–481, 1990.
[33] R.C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technology Journal*, 36:1389–1401, 1957.
[34] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.
[35] V.E. Shamanskii. A modification of Newton's method. *Ukrain. Mat. Z.*, 19:1333–1338, 1967.
[36] T.A. Davis and W.W. Hager. Row modification of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 26:621–639, 2005.
[37] T.F. Coleman and J.J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20:187–209, 1983.
[38] V. Simoncini and E. Gallopoulos. An iterative method for nonsymmeric systems with multiple right-hand sides. *SIAM J. Sci. Comput.*, 16:917–933, 1995.
[39] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.
[40] B. Vital. *Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur*. PhD thesis, Université de Rennes I., Rennes, 1990.