NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2000; **00**:1–6 *Prepared using* nlaauth.cls *[Version: 2002/09/18 v1.02]*

Preconditioner updates for solving sequences of linear systems in matrix-free environment†

Jurjen Duintjer Tebbens and Miroslav Tůma*

Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 18207 Praha 8, Czech Republic.

SUMMARY

We present two new ways for preconditioning of sequences of nonsymmetric linear systems in matrix-free environment. Both approaches are fully algebraic, they are based on the general updates of incomplete LU decompositions recently introduced in [1], and they may be directly embedded into nonlinear algebraic solvers. The first of the approaches uses a new model of partial matrix estimation to compute the updates. The second approach exploits separability of the function to apply the updated factorized preconditioner via function evaluations with the discretized operator. The approaches based on preconditioner updates are experimentally compared in matrix-free environment with two other possibilities: preconditioner recomputation for each linear system of a sequence, or freezing of a reference preconditioner. It is shown in our test cases, that updating is typically the best of the three compared strategies.

KEY WORDS: preconditioned iterative methods, matrix-free environment, inexact Newton-Krylov methods, factorization updates, incomplete factorizations

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, for example, when a system of nonlinear equations is solved by a Newton or Broyden-type method [2], [3]. Krylov subspace methods are among the most successful approaches for solving the linear systems. These methods have the property that the system matrix is needed only in the form of matrix-vector products, and the explicit

*Correspondence to: Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 18207 Praha 8, Czech Republic. E-mail: {tebbens,tuma}@cs.cas.cz
†This work was supported by the project No. IAA100300802 of the Grant Agency of the Academy of Sciences of the Czech Republic. The work of the first author is also supported by project number KJB100300703 of the Grant Agency of the Academy of Sciences of the Czech Republic.

Received 2008
Revised

representation of the matrix is not necessary. If the matrix is not represented explicitly, we often say that the method is *matrix-free*.

It is widely recognized that in most cases of practical interest, Krylov subspace methods must be preconditioned in order to be efficient and robust. However most of the strong preconditioners either require the system matrix explicitly, or their computation may be rather expensive. In order to reduce the costs of the computation of preconditioners, we may reuse a preconditioner over more systems of the given sequence of systems of linear equations. In addition, the quality of the reused preconditioner may be further enhanced through updates containing information extracted from the sequence of matrices, or from previous application of the Krylov subspace method. In this paper we address the problem of solving a sequence of general nonsymmetric systems by preconditioned Krylov subspace methods, where the preconditioners are based on incomplete LU decompositions, they use general rank-n updates, and all the computations are done in matrix-free environment. Due to the costs that are related to the fact that the system matrix is not given explicitly, avoiding frequent recomputations of the preconditioner from scratch seems to be even more important in matrix-free environment than in the standard case. In the following, we will briefly summarize basic lines of previous research on matrix-free preconditioning and on solving sequences of systems of linear equations with preconditioner updates, that is, on the basic subproblems which we face.

A very popular and natural framework to solve large systems of nonlinear algebraic equations is represented by Jacobian-free Newton-Krylov (JFNK) methods which combine Newton iterations with Krylov subspace methods and assume that the Jacobian matrix is not explicitly available. Let us first mention a couple of preconditioning strategies related to matrix-free environment, many of which are targeted prevailingly for the JFNK methods. First, the preconditioners can correspond to a discretization of an operator which is simpler than the operator for evaluating the sparse system matrix, see, e.g., [4], [5], [6], [7], [8], [9]. Successful preconditioners were also proposed for solving problems in applications which provide rather dense Jacobian matrices [10], [11]. If a preconditioner is algebraic, then the fact that the system matrix is not explicitly available often implies that the preconditioner is rather simple and/or sparse. In some of such situations, the role of the preconditioner is played by the matrix diagonal or its approximation. In other situations, the preconditioning employs more sophisticated stationary iterative methods, fast FFT-based solvers, ADI methods, inner-outer schemes etc., in order to be easily applied in matrix-free environment. An early important paper which explicitly targets preconditioning in matrix-free environment is [12] with results for a model nonlinear boundary value problem, see also the applications in CFD [13]. For more details on JFNK methods and their modifications, see the overview in [8], but see also [4], [14], [15]. If we know the sparsity structure of system matrices, these matrices can be always *estimated* by matrix-vector multiplications, as we will explain later, and this estimation can be efficient as well. Then more sophisticated preconditioners such as incomplete factorizations can be based on the estimated matrices. It is also known that only a few matrix-vector multiplications may be needed to obtain an *approximate estimate* that is sufficiently accurate for the construction of a good preconditioner [16].

Preconditioner updates used for solving system sequences are traditionally based on modifications by matrices of small rank. Early work which uses the Broyden formula to update the preconditioner was introduced in [17]. The authors in [18] use rank-1 updates for both Krylov and Newton parts of the solver and apply the resulting algorithm to power system problems. Another recent combination of rank-one updated preconditioners with the

Newton method is [19], see also the references given there. A theoretically interesting approach which may be useful especially if the system matrix changes very slowly was presented for the conjugate gradient method in [20]. An important field which employs sequences of large and sparse matrices where the matrices may not be explicitly available, is smooth optimization. Limited-memory variable metric methods [21] are based on keeping an approximate Hessian matrix (or its inverse) in the form of a truncated sequence of vector pairs, which represent updates of small rank and allow to compute search vectors from linear combinations of these vectors. Because of the memory restrictions, the limited-memory updates may be replaced or enhanced by an additional simple matrix, which is regularly updated, see, e.g., [22], [23], [24], [25]. Straightforward rank-2 updates of a previous preconditioner in the sequence were proposed in [26], where the vectors involved in the updates were computed from the conjugate gradient iterative method. Simple specific diagonal preconditioning for solving the sequence of problems arising in the truncated Newton method was recently proposed in [27].

An important class of algebraically-motivated strategies to accelerate the convergence of preconditioned iterative methods, and related to our goals, is based on constructing or improving the preconditioner by adaptive spectral information obtained directly from the Krylov subspace methods, see, e.g., [28], [29], [30]. These strategies are often problem specific, but they are in general compatible with matrix-free implementations. The authors in [31] and [32] determine an invariant subspace of the system matrix associated with the eigenvalues close to the origin and remove the influence of those eigenvalues on the rate of convergence (see the extension of the latter research for adaptive enhancement of preconditioners in a sequence of systems in [33]). A similar approach has been proposed in [34]. There are a couple of other techniques to improve the preconditioner based on deflated and augmented Krylov subspace techniques, see, e.g., [34], [35], [36]. All of these techniques have also a significant potential to be applied for solving sequences of systems. Recent results on solving the sequences of systems of linear equations with explicit recycling of the information from Krylov subspaces can be found, for example, in [37], [38], [39] for computational mechanics and material sciences, and in [40] for tomographic imaging. Note that further interesting ideas which may be used for solving general sequences of linear systems are used in solvers for sequences with the same system matrix and different right-hand sides which are not simultaneously available, see, e.g., [41], [42].

Although it is possible to analyze the spectral properties of sequences of preconditioned matrices in some important special cases, in typical situations we know much less, as it is, e.g., in general JFNK methods. Updates of small rank are often restricted to specific classes of problems or nonlinear schemes as well. Therefore, cheap and generally rank-n preconditioner updates are strongly needed. Recently some new approaches to approximate preconditioner updates were introduced, see e.g. [43]. The authors in [44] propose approximate diagonal updates to solve parabolic PDEs, see also [45]. Nonsymmetric updates of general incomplete LU decompositions were considered in [1, 46], see also some results in solving real-world problems in [47]. So far, neither of these approaches have addressed the challenges related to preconditioner updates in matrix-free environment.

This paper deals with matrix-free algorithms to solve the sequences of linear systems based on the general triangular preconditioner updates introduced in [1]. In particular, two new approaches for matrix-free updates of preconditioners are proposed. The first approach is based on evaluating of all the structures needed to perform the update via an efficient matrix estimation. In particular, a novel *partial estimation* procedure, which is targeted for the matrix-

free triangular updates, is proposed. The second approach performs the updates by modifying forward or backward solves with the preconditioner, inside the iterative method. It is shown that both approaches may be efficient and robust in matrix-free environment.

The paper is organized as follows. In Section 2 the general preconditioner updates are briefly recapped and the two basic approaches are introduced. They are presented in detail in the subsequent Sections 3 and 4. Section 5 discusses numerical experiments for both approaches. The paper is concluded by some finalizing remarks.

2. Basic update technique and the matrix-free algorithms

The triangular preconditioner updates for nonsymmetric sequences from [1] are defined with the help of the difference between the matrix from the first (*reference*) linear system of a sequence and the *current* system matrix. Let A be the system matrix of the reference system and let A^+ be the current system matrix. If LDU is an incomplete triangular decomposition of A and $B = A - A^+$ is the difference matrix, then the triangularly updated preconditioners for the current system are defined as

$$(LD - tril(B))\,U, \qquad \text{or} \qquad L\,(DU - triu(B))\,, \tag{2.2}$$

where *tril* and *triu* denote, respectively, lower and upper triangular part of a matrix. We assume that $(LD - tril(B))$ or $(DU - triu(B))$, respectively, is nonsingular. Without loss of generality, we will use the first type of update in (2.2) in our exposition. In practice, the type of update is chosen dynamically [46, 47].

In matrix-free environment the factorization LDU has been obtained through estimating the reference matrix A, and it is stored explicitly. The update needs in addition a part of the difference matrix B, which is not given explicitly (only A has been estimated). Since the straightforward estimation of the difference matrix may be expensive, one possible strategy which we propose is based on modified matrix estimation that is reasonably cheap. In this case we use an enhanced *partial* and *approximate* matrix estimation. This approach is described in Section 3.

In Section 4 we will describe another strategy to use the triangular updates in matrix-free environment. It applies the preconditioner (2.2) without running any matrix estimation procedure other than for the reference matrix. However, this is recommendable only when function components are *separable*. Let us explain what we mean by separability in our case (cf. the concept of partial separability in optimization, e.g. in [48]). Consider a Krylov subspace method where the product of the system matrix A with a vector v is replaced by the value of a function $\mathcal{F}$ evaluated at v. We say that $\mathcal{F}$ is separable if the evaluation of $\mathcal{F}$ can be easily separated in the evaluation of its function components. That is, if the components of the function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^n$ can be written as $\mathcal{F}_i : \mathbb{R}^n \to \mathbb{R}$, where $e_i^T \mathcal{F}(v) = \mathcal{F}_i(v)$, and computing $\mathcal{F}_i(v)$ costs about an n-th part of the full function evaluation $\mathcal{F}(v)$. Note that in some cases, as they arise in complicated computations based on finite volumes or finite elements, the contributions for each volume or element are computed simultaneously, and in this case, the evaluation of a single function component costs more.

3. Matrix-free triangular updates via partial matrix estimation

This subsection describes one possible way to compute and apply the triangular updates for a sequence of linear systems. As mentioned above, in general, it is possible to get a matrix or its desired submatrices by solving the *matrix estimation problem*. Let us shortly describe the strategy and provide some basic references. The matrix estimation problem is the problem to estimate a sparse matrix by a small number of well-chosen matrix-vector multiplications (matvecs). Curtis, Powell and Reid [49] were the first to demonstrate that all nonzero entries of a sparse matrix can be estimated, given the sparsity structure, using a number of matvecs which is often much smaller than the matrix dimension. Direct computation of the entries of a generally nonsymmetric matrix B can be formulated as the following problem.

Problem 3.1. *Given the sparsity pattern of B find vectors $d_1, \ldots, d_p$ such that for each nonzero entry b_{ij} of B there is a vector $d_k, 1 \leq k \leq p$, satisfying $(Bd_k)_i = b_{ij}(d_k)_j$.*

In practice, we need to have p as small as possible so that the number of matvecs needed to obtain all nonzero entries is minimal. Subsequently, Coleman and Moré [50] demonstrated the relation of the matrix estimation problem 3.1 to the vertex *coloring* of a related graph G by a minimum number of colors. This minimum number is called the chromatic number of G. So-called direct methods for solving the matrix estimation problem for a matrix B described in Problem 3.1 use as G the intersection graph of B, that is the adjacency graph $G(B^TB)$ of B^TB. Note that for an (undirected) adjacency graph $G(C)$ of a square and symmetric matrix C we define its set of vertices as $V(G(C)) = \{1, \ldots, n\}$ and its set of edges as $E(G(C)) = \{\{i, j\} \mid c_{ij} \text{ is nonzero}\}$. A vertex coloring of the intersection graph labels every vertex with a color in such a way that no two adjacent vertices have the same color. The number of groups of vertices of the related graph with the same color then corresponds to the number of matvecs needed to estimate all entries of the matrix. A recent survey of theoretical results and techniques in this field is [51] where one can find details on many standard matrix estimation strategies. If we need to estimate only a part of a given matrix, we speak about the *partial matrix estimation problem* [51], [16].

Using the notation from above, consider matrices A and A^+ from a sequence. If we need to compute a preconditioner directly from A^+, then a straightforward strategy is to estimate A^+ entirely. When the sparsity patterns of A^+ and A are the same, we can use the same graph G to find, let us say, p color groups for both matrices (note that we typically need to use only approximate algorithms for graph coloring since the related decision problem is NP-complete [52]), and the graph coloring algorithm does not need to be rerun to estimate A^+ if we have estimated A. In this way, we need p matvecs for each estimation. If the matrix patterns in the sequence differ too much, we may need to run the graph coloring algorithm for A^+ as well, but its running time is typically smaller than the time needed for matvecs. It was demonstrated in [16] that for A^+ we can use the results of the graph coloring algorithm for a matrix with a "slightly different" sparsity pattern.

In order to use the triangular updates described above we only have to estimate, in addition to A which was estimated earlier, the upper or the lower triangular part of A^+. This leads to a special partial matrix estimation problem. Without loss of generality, consider estimation of the lower triangular part of A^+. We will formulate this problem as a standard graph coloring problem (called 1-distance graph coloring problem; the problem can be formulated also differently using a different coloring paradigm) for a graph which is different from the

intersection graph of A^+. The following theorem describes this graph.

Theorem 3.1. *Consider the graph*

$$G_T(B) = G(L_B^T L_B) \cup G_K,$$

where $G(L_B^T L_B) = (V, E)$ *is the intersection graph of the lower triangular part of the matrix* B *and* G_K *is defined as*

$$G_K = \cup_{i=1}^{n} G_i, \qquad G_i = (V_i, E_i) = (V, \{\{k, j\} |\ b_{ik} \neq 0 \wedge b_{ij} \neq 0 \wedge k \leq i < j\}).$$

If the graph $G_T(B)$ *can be colored by* p *colors, then the entries of the lower triangular part* L_B *of* B *can be computed by* p *matvecs of* B *with vectors* $d_1, \ldots d_p$ *such that for each nonzero entry* l_{ij} *of* L_B *there is a vector* $d_k, 1 \leq k \leq p$, *satisfying* $(Bd_k)_i = l_{ij}(d_k)_j$.

Proof: In other words, the theorem gives necessary conditions to solve a modified Problem 3.1 in which we have to estimate only the entries of L_B via matvecs with B. Assume that $G_T(B)$ was colored by p colors. Define the vectors $d_k, 1 \leq k \leq p$, such that

$$(d_k)_j = \begin{cases} 1 & \text{if the vertex } j \text{ has the color } k, \\ 0 & \text{otherwise.} \end{cases}$$

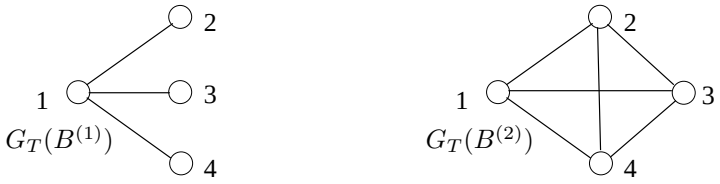
Consider a nonzero entry l_{ij} of L_B. Since there are edges $\{i, l\}$ in $G_T(B)$ for each $1 \leq l \leq n$ such that b_{il} is nonzero, we have $(Bd_k)_i = l_{ij}$ and we have the result. □

Note that the graph $G_T(B)$ contains only a subset of edges of the adjacency graph $G(B^T B)$ which should be considered to solve Problem 3.1. Consequently, in order to estimate only a triangular part of A^+ we may need a smaller number of matvecs than in the case of estimation of the whole B.

Another aspect of the estimation of a triangular part of a matrix is that it depends on the matrix reordering since the graph construction depends on it. We will demonstrate it in the following Example 3.1, where we show two differently reordered arrow matrices $B^{(1)}, B^{(2)}$ and the corresponding graphs $G_T(B^{(1)}), G_T(B^{(2)})$ from Theorem 3.1.

Example 3.1.

$$B^{(1)} = \begin{pmatrix} * & * & * & * \\ * & * & & \\ * & & * & \\ * & & & * \end{pmatrix} \qquad B^{(2)} = \begin{pmatrix} * & & & * \\ & * & & * \\ & & * & * \\ * & * & * & * \end{pmatrix}$$



Clearly, $G_T(B^{(1)})$ can be colored by two colors but $G_T(B^{(2)})$ needs four colors. The next theorem shows that sometimes we can increase our chances to decrease the number of matvecs

$$\begin{pmatrix} * & * & * & & & \\ * & * & & * & & \\ * & & * & & * & * \\ & * & & * & & * \\ & & * & & * & * \\ & & * & * & * & * \end{pmatrix}$$

Figure 3.1. Matrix B after the Cuthill-McKee reordering for which its graph $G_T(B)$ needs more colors than the graph $G_T(\hat{B})$, where $\hat{B}$ we get from B after the symmetric reversal of its columns and rows.

needed for the estimation of a triangular part of the matrix by an appropriate ordering of the matrix. In the following we will compare the Cuthill-McKee and the Reverse Cuthill-McKee (RCM) reorderings [53].

Theorem 3.2. *Assume that the irreducible matrix B with symmetric sparsity pattern was reordered by the Cuthill-McKee reordering. Further assume that the following condition applies: if $b_{ij} \neq 0$ for some i, j, $1 \leq j \leq i \leq n$ then $b_{lj} \neq 0$ for all l, $j \leq l \leq i$ (envelope assumption). Denote by $\hat{B}$ the matrix which we obtain from B by reversing the order of rows and columns with respect to B, that is, $\hat{B}$ corresponds to the original matrix reordered by the related RCM reordering. Then the chromatic number of $G_T(B)$ is not larger than the chromatic number $G_T(\hat{B})$.*

Proof: We will use induction on i. Let us first define $f_i = \min\{j|\ b_{ij} \neq 0\}$ for $1 \leq i \leq n$. If B is reordered by the Cuthill-McKee reordering then we know that $f_i \leq f_j$ if $1 \leq i \leq j \leq n$ (monotone envelope property) and $f_i < i$ for $1 < i \leq n$ [54].

The assertion is trivially valid for $i = 1$. Consider $i > 1$. Assume that we border the matrix of dimension $i-1$ by a row i from the bottom and a column i from the right. Let j be the minimum index such that $b_{ij} \neq 0$. The nonzero entries in the i-th row induce a complete subgraph in $G_T(B)$. Because of the envelope assumption, this complete subgraph must be in $G_T(\hat{B})$ as well since the nonzeros b_{lj} for $j \leq l \leq i$ induce a clique. Consequently, $G_T(\hat{B})$ has all edges from $G_T(B)$ and its chromatic number is not smaller than the chromatic number of $G_T(B)$. □

Note that CM/RCM reorderings are often used to preprocess a matrix of linear systems solved by preconditioned iterative methods. One motivation in the nonsymmetric case is that such reorderings may be very beneficial for the stability of the incomplete decomposition [55]. Nevertheless, Theorem 3.2 is not valid without the envelope assumption. Figure 3.1 shows an example of a matrix B after the Cuthill-McKee reordering. The chromatic number of $G_T(B)$ is five. B reordered by the related RCM needs only four colors. Despite this counterexample, Theorem 3.2 gives an idea of useful reorderings even when the envelope assumption does not hold.

Another important component of the proposed strategy, in addition to the estimation techniques, is the prefiltration of the matrix from which the preconditioner is computed. The prefiltration is based on the sparsity pattern of the reference matrix.

Let us summarize crucial points of our approach. The complete matrix-free preconditioned iterative method with the updates to solve a sequence of linear systems needs to estimate the reference matrix, and the triangular parts of the remaining matrices, so that they could be

used in the updates. These two tasks are performed via the following algorithm.

Algorithm 3.1. PARTIAL MATRIX ESTIMATION FOR TRIANGULAR PRECONDITIONER UPDATES. ***Input:*** *Matrix sequence* $A^{(0)}, A^{(1)}, \ldots, A^{(n)}$ *and the sparsity pattern* $\mathcal{S}(A^{(0)})$.

1. **Estimation.** *Estimate* $A^{(0)}$ *using* $\mathcal{S}(A^{(0)})$.
2. **Initial factorization.** *Factorize* $A^{(0)}$ *such that* $A^{(0)} \approx LDU$.
3. **Sparsification.** *Filtrate* $A^{(0)}$ *to get* $\overline{A^{(0)}}$ *and its sparsity pattern* $\mathcal{S}(\overline{A^{(0)}})$.
4. ***for*** $k = 0, \ldots, n$
   Estimate the lower triangular part of $A^{(i)}$ *needed for the update based on the coloring of* $G_T(\overline{A^{(0)}})$.
   end for

Note that the triangular part of A_0 is estimated twice. First, it is estimated with the original sparsity pattern. Second, the filtrated pattern $\mathcal{S}(\overline{A^{(0)}})$ is used. The updates then use the lower triangular part of the matrices $B^{(i)} = A^{(0)} - A^{(i)}$, for $i = 1, \ldots, n$ computed with the *filtrated and approximate* sparsity pattern $\mathcal{S}(\overline{A^{(0)}})$. Since the estimation adds some error to the computed matrix entries, it is important to distribute this error in all the approximate matrices in the same way. In addition, we observed experimentally that it is better to use in the updates an estimate to $A^{(0)}$ obtained in Step 4 of Algorithm 3.1, under the same conditions as the other estimates, than an estimate from Step 1. This is the reason to have the loop in Step 4 starting from 0. As for the sequential graph coloring heuristic, it tries to balance the error among the groups of columns of the same color as proposed in [16].

4. Matrix-free updates in the separable case

This section describes the approach for applying matrix-free triangular preconditioner updates when the function components are separable, as described in Section 2. Assume for the moment that the triangular part of the matrix A and its incomplete LU decomposition are available explicitly (for simplicity, we hide the diagonal factor of the decomposition in L). In practice, these quantities are computed for the reference system of the sequence. Let the current matrix A^+ be given implicitly in the form of its action on vectors, expressed by the function evaluation $\mathcal{F}^+(\cdot)$, where $\mathcal{F}^+ : \mathbb{R}^n \to \mathbb{R}^n$ and let $\mathcal{F}_i^+$ be the i-th component of $\mathcal{F}^+$. When the function components are separable we show how to avoid most estimation of A^+ and thus keep memory small at the same time. The strategy used to apply the updated preconditioner

$$(L - tril(B))\, U \tag{4.3}$$

is summarized in Algorithm 4.1.

Algorithm 4.1. APPLICATION OF TRIANGULAR PRECONDITIONER UPDATES IN THE SEPARABLE CASE. ***Input:*** *Explicitly stored matrices L, U and $tril(A)$ and the function components of $\mathcal{F}^+$ which represent A^+ implicitly.*

1. **Initialization.** *Find the main diagonal $\{a_{11}^+, \ldots, a_{nn}^+\}$ of A^+ before running the iterative method. It can be found by computing*

$$a_{ii}^+ = \mathcal{F}_i^+(e_i), \qquad 1 \le i \le n.$$

2. **Forward solve in each iteration.** *Use the following* mixed explicit-implicit *strategy: Split the lower triangular matrix of (4.3) as $L - tril(B) = E + tril(A^+)$. That is, $E \equiv L - tril(A)$ is stored explicitly, and the implicit part $tril(A^+)$ contains entries of the new system matrix. We then have to solve triangular systems of the form*

$$\left(E + tril(A^+)\right) z = y,$$

which yields the forward solve loop

$$z_i = \frac{y_i - \sum_{j<i} e_{ij} z_j - \sum_{j<i} a_{ij}^+ z_j}{e_{ii} + a_{ii}^+}, \qquad i = 1, 2, \ldots, n. \tag{4.4}$$

Note that the values e_{ii} and a_{ii}^+ in the denominator are known. In the numerator of (4.4), the first sum can be computed explicitly and the second sum can be computed by the function evaluation

$$\sum_{j<i} a_{ij}^+ z_j = \mathcal{F}_i^+ \left((z_1, \ldots, z_{i-1}, 0, \ldots, 0)^T\right). \tag{4.5}$$

3. **Backward solve in each iteration.** *This is a trivial step since the matrix U in (4.3) has been stored explicitly.*

The costs to find the main diagonal in Step 1 (initialization) correspond approximately to the costs of one full function evaluation. Note that the diagonal of $B = A - A^+$ is known in advance in some applications. In particular, if the diagonal does not change, $\mathrm{diag}(B)$ is the zero matrix. Similarly, in Step 2, the whole forward-solve loop requires n partial evaluations (4.5). In total, it approximately gives the cost of one additional full function evaluation per solve step of the preconditioned iterative method.

Let us compare this approach with the strategy which recomputes the preconditioner for each system of a given sequence. With updates applied according to Algorithm 4.1, only the main diagonal of A^+ needs to be estimated (in the initialization). If we would recompute the preconditioner, on the other hand, we would have to estimate the matrix A^+, and also to compute the incomplete factorization. However, *application* of the update using Algorithm 4.1 could be more expensive than applying a new factorization if we would need a similar number of iterations since an extra full function evaluation in each forward solve is needed.

Let us mention that this additional function evaluation with Algorithm 4.1 can be in the case of slow convergence of the preconditioned iterative method subsequently replaced by the following explicit evaluations. Let *stril* denote the strict lower triangular part. To obtain the entries of $stril(A^+)$, note that every forward solve with (4.5) gives us an equation in the

Table 4.1. Costs of one non-matrix-free explicit preconditioning step

type	initialization	solve step	memory
Recomp	$A^+ \approx L^+U^+$	solves with L^+, U^+	A^+, L^+, U^+
Update	—	solves with $L, U, tril(B)$	$A^+, tril(A), L, U$

unknowns a^+_{ij}, $j < i$. Thus after the first complete forward solve cycle we have one equation for the entries of all rows in $stril(A^+)$ and rows of $stril(A^+)$ with one entry can be evaluated (and merged with the corresponding rows of L as far as the entries are in the same columns). Similarly, after the second complete forward solve cycle we get two equations for the entries of all rows in $stril(A^+)$ and thus rows with two entries are explicitly known (and can possibly be merged with rows of L). If a row of $stril(A^+)$ contains k entries then k forward solves are needed to obtain enough linear equations to compute the entries of the row. Here we do not address the problem of non-singularity of the system from which the entries of A^+ are computed as well as other practical problems which may be faced during the implementation.

Let us return to the computation of A, which is the reference matrix of the given sequence. In our experiments, A is estimated. But, for completeness, if the estimation of A is not efficient and the decomposition has been obtained in a different way, or if the sparsity patterns of $tril(A)$ and L differ so much that the storage costs would grow unacceptably, one may also use the function components of $\mathcal{F}$ to operate with $tril(A)$ in the forward solve. Then, formally, the explicit part E of the forward solve consists of L only. In (4.4) there are three sums of which the last two are computed implicitly. The whole forward solve then would cost about two full function evaluations in total (which can again be eliminated with the process described in the previous paragraph).

Finally, note that the function separability could even motivate replacement of the graph coloring-based estimation by direct evaluations of the non-zero entries according to the simple formula

$$a_{ij} = \mathcal{F}_i(e_j). \tag{4.6}$$

In addition, specific preconditioners may not even need all matrix entries to be evaluated [16]. Apart from a few experiments in the next section, we will not further follow the ideas mentioned in the latest three paragraphs.

We conclude this section by an overview of costs for the described strategies. Let us distinguish three cases. Table 4.1 summarizes the costs and memory for one preconditioning step with recomputation (denoted as "Recomp") and with the update (2.2) ("Update"), respectively, in the standard, non matrix-free environment where the system matrices are explicitly given. With the recomputation strategy, we denote the approximate LU factors of A^+ by L^+, U^+. Tables 4.2 and 4.3 present the costs and memory for one preconditioning step with recomputation and update, respectively, in matrix-free environment. Table 4.2 addresses the approach from Section 3. We denote matrix estimations and function evaluations by $est(.)$ and $eval(.)$, respectively. Table 4.3 summarizes the costs for the approach from Section 4 which exploits the separability of $\mathcal{F}$ into components assuming that the function components can be computed in about an nth part of the cost of a full function evaluation. To emphasize the difference between the strategies, it is assumed that both $tril(A^+)$ and $tril(A)$ are given only

Table 4.2. Costs of one matrix-free preconditioning step based on Algorithm 3.1

type	initialization	solve step	memory
Recomp	$est(A^+)$, $A^+ \approx L^+U^+$	solves with L^+, U^+	L^+, U^+
Update	$est(tril(A^+))$	solves with $L, U, tril(B)$	$tril(A^+), tril(A), L, U$

Table 4.3. Costs of one matrix-free preconditioning step based on Algorithm 4.1

type	initialization	solve step	memory
Recomp	$est(A^+)$, $A^+ \approx L^+U^+$	solves with L^+, U^+	L^+, U^+
Update	$est(diag(A^+))$	solves with L, U, eval$(\mathcal{F}, \mathcal{F}^+)$	L, U

implicitly.

The tables provide only a rough comparison. In particular, the amount of overlap between the sparsity patterns of L and $tril(B)$ may have an important influence on the storage and application costs. In some cases A and A^+ may have not only somewhat different sparsity patterns, but also completely different sizes. Our second example in Section 5 shows that the updates can be efficient even in such a situation.

5. Numerical experiments

This section is devoted to numerical experiments illustrating the techniques from Section 3 and 4. In particular, we consider two test problems for experiments with the approach based on partial matrix estimation and two problems covering the separable case. We attempted to use a variety of ILU decompositions and we performed tests with GMRES as well as with BiCGSTAB. The first two problems consider fixed sequences of linear systems generated from nonlinear solvers. The next two problems result from a Newton-type method with a flexible stopping criterion for the linear system solution. Here the preconditioners with the triangular updates were fully embedded into the nonlinear solver. The convergence of the resulting matrix-free Newton-Krylov method strongly depends on the accuracy of linear system solution and varies with the type of preconditioner that is used. All experiments were implemented in Fortran 95 on Intel Pentium-based machines.

In all experiments we use the standard difference approximation of the Jacobian of the function F that is to be minimized to avoid storage of the Jacobian. I.e., a matvec with the Jacobian, Av, is replaced by

$$\mathcal{F}(v) \equiv \frac{F(x + \epsilon \cdot v/\|v\|) - F(x)}{\epsilon}, \tag{5.7}$$

for some small $\epsilon > 0$, where x is the vector at which the Jacobian is approximated.

The first set of experiments is devoted to solving a sequence of linear problems arising during the computation of a constitutive model from structural mechanics provided by Karsten Quint. More precisely, a small strain metal viscoplasticity model was developed for a rectangular plate

of length 100, width 21.2 and height 9.62 cm with a hole in the middle. When applying the Multilevel-Newton algorithm, every time-step contains an inner loop that requires the solution of nonlinear systems. For more details on the parameters of the material and of the Multilevel-Newton algorithm which were used, we refer to the description of the first application in [56]. We consider here a sequence of linear systems from a randomly chosen time-step in the middle of the simulation process and discretization with 1 350 quadratic elements in most of the domain with a somewhat finer grid in the center. This sequence consists of 8 linear systems of dimension 4 936 with matrices containing about 315 000 nonzeros.

In this case, separate computation of the function components would be expensive, and we used the strategy based on partial matrix estimations. To solve the problem, we use the GMRES(40) method preconditioned by ILUT [57]. We present results of several experiments differing in parameters provided to the preconditioner. In particular, we would like to show that the matrix-free strategy from Algorithm 3.1 is successful over large variations in the preconditioner density. Let us remind that Algorithm 3.1 evaluates in its loop also a less accurate approximation of the triangular part of $A^{(0)}$ using the filtrated pattern $\mathcal{S}(\overline{A^{(0)}})$ which we use to compute the updates, although we have a more accurate $A^{(0)}$ available. $A^{(0)}$ was filtrated in Algorithm 3.1 such that all the entries with magnitude smaller than half of the magnitude of the largest entry in their rows were dropped.

Tables 5.4-5.8 present the results in terms of number of iterations and needed matrix-vector multiplications (that is, the function evaluations (5.7)). We compare the three possible computational strategies: preconditioner recomputation by matrix estimation for each system (Recomp), preconditioner computation only for the reference matrix (Freeze), and preconditioning with the triangular updates based on Algorithm 3.1. The average number of nonzeros of the factorizations is denoted with "Psize". The column "fevals" gives the number of function evaluations needed for matrix estimation and "overall fevals" presents the number of iterations plus the number of fevals for estimations. The two fevals numbers for $A^{(0)}$ in the column "Updated" correspond to its estimation with full and filtrated pattern. The first fact which we observe is that the updating strategy works very well in terms of iteration counts, and it seems from this point of view to be the best option. Let us also remind the experimental dependence of timings and iteration counts presented for the triangular updates in [1] if we assume similar sizes of preconditioners used in the compared strategies. Consequently, it is clear that the updates are very often able to recover a lot of the information missing in the LU decomposition of the reference matrix. Except for Table 5.8, both recomputation and updates are much better than the freezing strategy. Table 5.8 shows the reversed situation, where the freezing strategy is the best of all. But note that in this case the preconditioner is rather dense. One could assume that the additional information provided by the sparsified difference matrix does not seem to be sufficient to improve the preconditioner in this case.

For the next set of experiments we have chosen a sequence of problems provided by Reijo Kouhia. The sequence represents linear systems from the discretization of nonlinear heat transfer on a 50×50 quadrilateral mesh discretized with 9-node Lagrange biquadratic elements. The system matrices are of dimension 9 801 with 152 881 nonzero entries. In this case, the individual matrices of the sequence are numerically strongly different. Consequently, the sizes of the ILUT preconditioners differ strongly as well. Table 5.9 presents in the column denoted by "Psize" the sizes of the recomputed preconditioner. In order to have the comparison fair to all the options, we compute the preconditioner as follows. The option "Recomp" estimates the matrix and computes the ILUT(0.5,10) preconditioner in the form $A^+ \approx L^+U^+$ for

Table 5.4. *Number of iterations and function evaluations for solving preconditioned linear systems from the structural mechanics problem with ILUT(0.01,5).*

ILUT(0.01,5), Psize ≈ 260 000						
Matrix	Recomp		Freeze		Updated	
	its	fevals	its	fevals	its	fevals
$A^{(0)}$	343	89	343	89	343	89+25
$A^{(1)}$	172	89	623	0	237	25
$A^{(2)}$	201	89	694	0	298	25
$A^{(3)}$	294	89	723	0	285	25
$A^{(4)}$	298	89	799	0	334	25
$A^{(5)}$	386	89	708	0	320	25
$A^{(6)}$	348	89	714	0	318	25
$A^{(7)}$	317	89	717	0	318	25
overall fevals	3 071		5 321		2 942	

Table 5.5. *Number of iterations and function evaluations for solving preconditioned linear systems from the structural mechanics problem with ILUT(0.001,20).*

ILUT(0.001,20), Psize ≈ 404 000						
Matrix	Recomp		Freeze		Updated	
	its	fevals	its	fevals	its	fevals
$A^{(0)}$	187	89	187	89	187	89+25
$A^{(1)}$	89	89	393	0	146	25
$A^{(2)}$	126	89	448	0	182	25
$A^{(3)}$	221	89	480	0	184	25
$A^{(4)}$	234	89	513	0	190	25
$A^{(5)}$	193	89	487	0	196	25
$A^{(6)}$	178	89	521	0	196	25
$A^{(7)}$	246	89	521	0	196	25
overall fevals	2 186		3 639		1 966	

each linear system. The other options use the ILUT(α,10) decomposition of $A^{(0)}$ where α is chosen such that the decomposition is of similar size as the decomposition $A^+ \approx L^+U^+$. This strategy was chosen because the sizes of ILUT(0.5,10) decompositions for the matrices in the sequence significantly change, and we are interested in evaluating the power of the updated preconditioners which would contain a similar amount of information, measured by the preconditioner size. These experiments use the BiCGStab iterative method and the "overall fevals" row sums together the following items: twice the number of iterations, and the number of matvecs needed for estimations.

As mentioned above, the next two test problems cover the separable function case. In

Table 5.6. *Number of iterations and function evaluations for solving preconditioned linear systems from the structural mechanics problem with* $ILUT(10^{-4}, 30)$.

ILUT$(10^{-4}, 30)$, Psize $\approx 550\,000$						
Matrix	Recomp		Freeze		Updated	
	its	fevals	its	fevals	its	fevals
$A^{(0)}$	85	89	85	89	85	89+25
$A^{(1)}$	59	89	233	0	78	25
$A^{(2)}$	72	89	313	0	84	25
$A^{(3)}$	78	89	344	0	85	25
$A^{(4)}$	78	89	289	0	108	25
$A^{(5)}$	78	89	289	0	108	25
$A^{(6)}$	79	89	318	0	108	25
$A^{(7)}$	86	89	318	0	108	25
overall fevals	1 327		2 278		1 253	

Table 5.7. *Number of iterations and function evaluations for solving preconditioned linear systems from the structural mechanics problem with* $ILUT(10^{-5}, 50)$.

ILUT$(10^{-5}, 50)$, Psize $\approx 812\,000$						
Matrix	Recomp		Freeze		Updated	
	its	fevals	its	fevals	its	fevals
$A^{(0)}$	65	89	65	89	65	89+25
$A^{(1)}$	31	89	128	0	52	25
$A^{(2)}$	35	89	163	0	45	25
$A^{(3)}$	35	89	237	0	45	25
$A^{(4)}$	37	89	167	0	52	25
$A^{(5)}$	38	89	169	0	51	25
$A^{(6)}$	37	89	168	0	51	25
$A^{(7)}$	50	89	168	0	51	25
overall fevals	1 040		1 354		901	

the first example, the function with easily separable components is represented by a two-dimensional nonlinear convection-diffusion model problem with finite difference discretization. The convection-diffusion model problem has the form (see, e.g. [2])

$$-\Delta u + Cu\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) = f(x,y), \quad f(x,y) = 2000x(1-x)y(1-y), \tag{5.8}$$

where $C > 0$ is the Reynold number, and it is discretized on the unit square. The standard five-point discretization stencil (central difference) provides separable components of F of the

Table 5.8. *Number of iterations and function evaluations for solving preconditioned linear systems from the structural mechanics problem with* $ILUT(10^{-6}, 70)$.

ILUT$(10^{-6}, 70)$, Psize $\approx 950\,000$						
Matrix	Recomp		Freeze		Updated	
	its	fevals	its	fevals	its	fevals
$A^{(0)}$	32	89	32	89	32	89+25
$A^{(1)}$	21	89	78	0	54	25
$A^{(2)}$	28	89	88	0	38	25
$A^{(3)}$	24	89	101	0	39	25
$A^{(4)}$	26	89	92	0	38	25
$A^{(5)}$	26	89	87	0	38	25
$A^{(6)}$	26	89	86	0	38	25
$A^{(7)}$	28	89	86	0	38	25
overall fevals	923		739		804	

Table 5.9. *Number of iterations and function evaluations for solving preconditioned linear systems from nonlinear heat transfer problem. Recomputed factorizations for each matrix use ILUT(0.5,10), frozen and updated preconditioners use an* $ILUT(\alpha, 10)$ *preconditioner of the reference problem which has similar size as the recomputed preconditioner.*

ILUT preconditioner as described							
Matrix	Psize	Recomp		Freeze		Updated	
		its	matvecs	its	matvecs	its	matvecs
$A^{(0)}$	29 271	97	29	97	29	97	29+12
$A^{(1)}$	72 916	121	29	2 363	0	112	12
$A^{(2)}$	61 115	117	29	1 207	0	110	12
$A^{(3)}$	56 239	126	29	729	0	122	12
$A^{(4)}$	53 644	116	29	525	0	122	12
$A^{(5)}$	52 510	127	29	322	0	128	12
$A^{(6)}$	52 146	116	29	263	0	130	12
$A^{(7)}$	52 067	130	29	164	0	141	12
$A^{(8)}$	52 063	142	29	223	0	144	12
$A^{(9)}$	52 063	139	29	223	0	137	12
$A^{(10)}$	52 063	136	29	222	0	131	12
$A^{(11)}$	52 024	140	29	222	0	131	12
overall matvecs		3 710		5 410		3 183	

simple form

$$F_k(x) \equiv \frac{4x_k - x_{k-1} - x_{k+1} - x_{k+N} - x_{k-N}}{h^2}$$
$$-Cx_k \frac{x_{k-1} + x_{k+1} + x_{k+N} + x_{k-N}}{h} - f_k, \quad 1 \le k \le n,$$

where f_k is the discretization of f and N is the number of inner nodes. To solve $F(x) = 0$ we use the inexact Newton-Krylov method where the Krylov subspace method is BiCGSTAB and the stopping criterion of the iterative method is chosen adaptively; for details about this inexact Newton-Krylov method, see [58]. The final matrix-free solver was embedded into the UFO-software [58] for nonlinear problems. The initial approximation is the discretization of $u_0(x, y) = 0$.

The preconditioner we use in the experiments is ILU(0), which has the same sparsity pattern as the matrix it preconditions. Note that with an ILU(0)-factorization the triangular factors L and U have exactly the same sparsity pattern as $tril(B)$ and $triu(B)$. We will exploit this when applying the updates (see below). In our experiments, ILU(0) was computed by rows by adapting Saad's ILUT code, see [57].

In the tables 5.10-5.15 we display results for several choices of C and different grid sizes. As before, 'Freeze' denotes the case where the ILU(0) factorization is computed for the reference linear system only, and it is reused for all the subsequent linear systems. In the column "Recomp", ILU(0) is computed for each linear system separately. The last two columns present results with preconditioners updated by lower or upper triangular updates. For these experiments we present timings for the individual strategies because it would be difficult to present a fair comparison without them (the number of function evaluations cannot be used because the evaluations are split into function component evaluations).

The ILU(0) factorizations are computed from the estimations of Jacobians obtained by direct function component evaluations according to (4.6). We observed that in our case this is slightly faster than running the graph coloring estimation. However, in this model example the difference is marginal. The graph coloring algorithm yields 7 colors, hence the matrix is estimated with about $7n$ function component evaluations. With (4.6) we need $nnz \approx 5n$ function component evaluations. In addition, the graph coloring algorithm for this problem is very fast. For the largest problem we tested, $n = 96\,100$, the average time for estimation with graph coloring is 0.27 seconds, for estimation with (4.6) we need 0.13 seconds on average and the graph coloring algorithm, which needs to be run only once, takes 0.3 seconds. These are negligible time savings compared to the duration of the whole solution process. However, with more complex sparsity patterns the situation may be different.

The triangular parts $tril(B)$ and $triu(B)$ have been obtained directly by (4.6) as well and they are merged with L or U before the iterative solver is applied. We observed experimentally that this is about as fast as performing two steps of mixed explicit-implicit solves with Algorithm 4.1 and merging the entries computed from the two-by-two linear systems provided by the implicit matvecs (4.5) as described in Section 4. However, using mixed explicit-implicit solves throughout the whole linear solution process is clearly slower. One mixed explicit-implicit solve in the dimension $n = 96\,100$ costs about 0.07 seconds whereas a solve where entries are merged takes about 0.02 seconds. This difference may become smaller when updating a factorization which has the sparsity pattern very different from that of $tril(B)$ or $triu(B)$. The main difference is probably caused by the fact that a function component evaluation as

Table 5.10. *Preconditioning strategies for convection-diffusion problem (5.8) with* $n = 8\,182$ *and* $C = 10$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	159	124	129	160
Newton iterations	7	7	7	7
FCE (in thousands)	2 956	2 630	2 615	3 117
overall time in seconds	1.98	3.55	1.92	2.36

Table 5.11. *Preconditioning strategies for convection-diffusion problem (5.8) with* $n = 8\,182$ *and* $C = 50$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	410	122	153	186
Newton iterations	9	9	9	9
FCE (in thousands)	7 168	2 824	3 198	3 732
overall time in seconds	4.39	4.29	2.25	2.73

in (4.5) is more expensive than the explicit sum $\sum_{j<i} e_{ij} z_j$ in (4.4). Nevertheless, there may be applications where one cannot afford to store $tril(B)$ or $triu(B)$ in addition to the factors L and U and using mixed explicit-implicit solves would be the only option available.

The results are expressed by the total number of BiCGStab iterations, the total number of nonlinear Newton steps and the total time in seconds needed to reduce $\|F(x)\|$ to the value 10^{-15}. For smaller dimensions we also display the total number of function component evaluations (denoted by FCE) in thousands. The results show that recomputation always yields the smallest number of BiCGStab iterations but it is clearly outperformed by the strategy which uses the updates when the time is considered. Even the freezing strategy, for which BiCGStab performs poorly, is faster than recomputation in many test cases. The repeated computation of the ILU(0) factorization seems to be relatively expensive in this example of a matrix-free implementation. For example, with $n = 44\,521$ the average time to compute ILU(0) is 3.7 seconds. This represents a considerable part of the total solution time. The important time losses caused by recomputation are already observed in Table 5.10: Recomputation and the lower triangular updates yield nearly the same number of BiCGStab iterations but the updates make the strategy close to twice as fast. As is the case with the number of BiCGStab steps, the number of function component evaluations does not seem to influence significantly the overall performance either (note that additional function evaluations are performed after the solution of the linear system during the line search). The influence of the Reynold number C seems to be modest. Only the freezing strategy is rather sensitive to the choice of C. In order to have the experiments realistic from the physical point of view, we did not experiment with the choice of C for larger problems and used the value of the Reynold number from the case where the updates performed worst for $n = 8\,182$, i.e. $C = 100$. As for the difference between lower and upper triangular updates, we display here both strategies but in practice one would choose between these two adaptively according to the triangular part of B with larger magnitude of its norm, see also [1].

Table 5.12. *Preconditioning strategies for convection-diffusion problem (5.8) with* $n = 8\,182$ *and* $C = 100$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	1 648	111	261	1 203
Newton iterations	11	10	10	11
FCE (in thousands)	27 361	2 750	5 036	20 386
overall time in seconds	15.85	4.69	3.42	13.62

Table 5.13. *Preconditioning strategies for convection-diffusion problem (5.8) with* $n = 22\,801$ *and* $C = 100$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	1 550	228	289	334
Newton iterations	10	9	9	9
FCE (in thousands)	72 393	12 808	15 228	17 280
overall time in seconds	44.7	19.4	11.7	12.7

Table 5.14. *Preconditioning strategies for convection-diffusion problem (5.8) with* $n = 44\,521$ *and* $C = 100$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	740	400	432	528
Newton iterations	9	9	9	9
overall time in seconds	45.9	56.3	31.1	39.1

Table 5.15. *Preconditioning strategies for convection-diffusion problem (5.8) with* $n = 96\,100$ *and* $C = 100$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	1 369	704	911	778
Newton iterations	9	9	9	9
overall time in seconds	176	190	133	121

Our last test problem illustrates some further effects of the preconditioner updates in matrix-free environment with separable function components. This problem is also taken from the UFO-test software and represents a two-dimensional driven cavity problem of the form

$$\Delta\Delta u + C\left(\frac{\partial u}{\partial y}\frac{\partial \Delta u}{\partial x} - \frac{\partial u}{\partial x}\frac{\partial \Delta u}{\partial y}\right) = 0, \tag{5.9}$$

on the unit square, discretized by 13-point finite differences on a shifted uniform grid [59].

The boundary conditions are given by $u = 0$ on $\partial\Omega$ and $\partial u(0,y)/\partial x = 0$, $\partial u(1,y)/\partial x = 0$, $\partial u(x,0)/\partial x = 0$ and $\partial u(x,1)/\partial x = 1$. The initial approximation is the discretization of $u_0(x,y) = 0$.

In this example the main problem with recomputing the preconditioner is that it becomes unstable as the Newton iteration proceeds (this is caused by the nature of this nonlinear problem). The first system matrix leads to the most stable factorization, hence freezing or updating it may be a better idea than recomputing. We show this for two choices of parameters: In Table 5.16 the dimension is 3 364 and $C = 100$, in Table 5.17 the dimension is 3 721 and $C = 50$ (for larger problems the initial preconditioner is also unstable). The recomputed preconditioners are too unstable for convergence of the Newton process. We did not use here ILU(0) but allowed changes of the sparsity pattern and adding more fill-in into the incomplete factors. These tables use ILUT(1,0.01); this yields a factorization with 49 185 non-zeros for Table 5.16 where the system matrices have 42 576 non-zero entries. In Table 5.17 the system matrices have 47 157 non-zeros and ILUT(1,0.01) has 54 474. Both frozen and updated preconditioners are able to solve the nonlinear problem; updating the lower triangular factor performs a little worse than freezing, but updating the upper triangular factor of the preconditioner performs better than freezing.

In the driven cavity problem we also compare the performance of the mixed implicit/explicit forward solves from Algorithm 4.1 with fully explicit solves where the entries of $tril(A^+)$ were obtained from (4.6). Recall that in the previous example, mixed implicit/explicit forward solves were clearly slower than explicit solves. Here we use ILU-factorizations different from ILU(0). The overall cost of a forward solve, composed of solves with L and with $tril(B)$, starts to be dominated by the solves with L as the size of L grows. We demonstrate this effect in Table 5.18 for the case $n = 3\,721$ with $C = 50$ using lower triangular preconditioner updates. The table displays the total number of iterations and the overall timing in seconds (in the same column) to solve the whole sequence in dependence of the size of the initial factorization. We observe that the mixed implicit/explicit strategy is in general only slightly slower than the explicit strategy. Sometimes it is even faster (the ILU(14,0.01)-factorization yields a BiCGStab breakdown). Note that for the ILU(1,0.01)-factorization from Table 5.17 the difference between the two strategies is more significant: For instance, the explicit upper triangular updates from the table needed 2.28 seconds to solve the nonlinear problem with 557 BiCGStab iterations whereas the mixed implicit/explicit solves needed 3.82 seconds for 592 BiCGStab iterations. Let us also mention that to guarantee convergence of BiCGStab with preconditioners recomputed from scratch it is necessary to use rather dense factorizations. In our test cases, the sparsest factorization for which BiCGStab with recomputed preconditioners converges is ILUT(20,0.01). The number of its nonzeros is in this case about four times that of the system matrix.

As in the previous example, we measured the difference between matrix estimation with graph coloring and estimation with (4.6). The structure of the system matrices is more complicated than for the five-diagonal matrices of the previous test problem; the number of computed color groups is 18, hence we need about $18n$ against $13n$ function component evaluations with (4.6). Still the difference for the overall computation time is negligible. For instance, solving the sequence from Table 5.16 with upper triangular updates and with graph color estimation takes 1.6 seconds, i.e. only 0.1 seconds more than estimation with (4.6).

Table 5.16. *Preconditioning strategies for driven cavity problem (5.9) with ILUT(1, 0.01),* $n = 3\,364$ *and* $C = 100$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	1 212	div.	1 305	436
Newton iterations	5	div.	5	5
overall time in seconds	3.3	–	4.1	1.5

Table 5.17. *Preconditioning strategies for driven cavity problem (5.9) with ILUT(1, 0.01),* $n = 3\,721$ *and* $C = 50$

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	743	div.	1 328	557
Newton iterations	5	div.	6	5
overall time in seconds	2.5	–	5.2	2.3

Table 5.18. *Explicit and implicit solves with lower triangular preconditioner updates for driven cavity problem (5.9) with* $n = 3\,721$ *and* $C = 50$

	Psize	Explicit solves	Mixed explicit/implicit solves
ILUT$(10, 0.01)$	120 837	312/2.18	199/1.97
ILUT$(12, 0.01)$	135 552	159/1.41	151/1.69
ILUT$(14, 0.01)$	150 268	—	—
ILUT$(16, 0.01)$	164 961	127/1.41	128/1.68
ILUT$(18, 0.01)$	179 655	119/1.43	115/1.64
ILUT$(20, 0.01)$	194 330	134/1.65	142/2.03
ILUT$(22, 0.01)$	209 009	107/1.53	103/1.69

6. Conclusions

We have presented theoretical results and numerical experiments related to matrix-free strategies for solving sequences of linear systems by preconditioned iterative methods. In particular, we introduced two new approaches to apply triangular updates for enhancing the solver of the sequences. The experiments in matrix-free environment appear to confirm that the proposed strategies are typically the best of all compared possibilities. Moreover, the updates can be easily embedded into matrix-free nonlinear solvers.

7. Acknowledgment

We would like to gratefully acknowledge the help of Stefan Hartmann, Reijo Kouhia and Karsten Quint who provided the test problems. We are also very much indebted to Ladislav Lukšan for helping us to embed the updated preconditioners into the UFO software.

REFERENCES

1. Duintjer Tebbens J, Tůma M. Efficient preconditioning of sequences of nonsymmetric linear systems. *SIAM J. Sci. Comput.* 2007; **29**(5):1918–1941.
2. Kelley CT. *Iterative Methods for Linear and Nonlinear Equations*. SIAM: Philadelphia, 1995.
3. Kelley CT. *Solving nonlinear equations with Newton's method*. Fundamentals of Algorithms, Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, 2003.
4. Brown PN, Saad Y. Hybrid Krylov methods for solving systems of nonlinear equations. *SIAM J. Sci. Stat. Comput.* 1990; **11**:450–481.
5. Mousseau VA, Knoll DA, Rider WJ. Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion. *J. Comput. Phys.* 2000; **160**:743–765.
6. Keyes D. Terascale implicit methods for partial differential equations. *Contemporary Methematics, AMS, Providence* 2001; **306**:29–84.
7. Reisner J, Mousseau VA, Wyszogrodzki AA, Knoll DA. An efficient physics-based preconditioner for the fully implicit solution of small-scale thermally driven atmospheric flows. *J. Comput. Phys.* 2003; **189**:30–44.
8. Knoll DA, Keyes D. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *J. Comp. Phys.* 2004; **193**:357–397.
9. Bernsen E, Dijkstra HA, Wubs FW. A method to reduce the spin-up time of ocean models. *Ocean Modell.* 2008; **20**:380–392.
10. Li X, Primeau F. A fast Newton-Krylov solver for seasonally varying global ocean biogeochemistry models. *Ocean Modell.* 2008; **to appear**.
11. Khatiwala S. Fast spin up of ocean biogeochemical models using matrix-free Newton-Krylov. *Ocean Modelling* 2008; **23**:121–129.
12. Chan TF, Jackson KR. Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms. *SIAM J. Sci. Statist. Comput.* 1984; **5**(3):533–542.
13. Luo H, Baum JD, Löhner R. A fast, matrix-free implicit method for compressible flows on unstructured grids. *J. Comp. Physics* 1998; **146**(2):664–690.
14. Knoll DA, McHugh PR. Newton-Krylov methods applied to a system of convection-reaction-diffusion equations. *Comput. Phys. Commun.* 1995; **88**:141–160.
15. Knoll DA, McHugh PR, Keyes DE. Newton-Krylov methods for low Mach number compresible combustion. *AIAA Journal* 1996; **34**:961–967.
16. Cullum J, Tůma M. Matrix-free preconditioning using partial matrix estimation. *BIT Numer. Math.* 2006; **46**:711–729.
17. Choquet R. A matrix-free preconditioner applied to CFD. *Technical Report No. 940*, INRIA, France 1995.
18. Chen Y, Shen C. A Jacobian-free Newton-GMRES(m) method with adaptive preconditioner and its application for power flow calculations. *IEEE Transactions on Power Systems* 2006; **21**(3):1096–1103.
19. Bergamaschi L, Bru R, Martínez A, Putti M. Quasi-Newton preconditioners for the inexact Newton method. *ETNA* 2006; **23**:63–74.
20. Dunagan J, Harvey NJA. Iteratively constructing preconditioners via the conjugate gradient method. *STOC'07 (39th annual ACM Symposium on Theory of Computing, San Diego, CA, 2007)*. ACM: New York, 2007; 207–216.
21. Nocedal J. Updating quasi-Newton matrices with limited storage. *Math. Comp.* 1980; **35**(151):773–782.
22. Gilbert JC, Lemaréchal C. Some numerical experiments with variable-storage quasi-Newton algorithms. *Math. Programming* 1989; **45**(3, (Ser. B)):407–435.
23. Zhu M, Nazareth JL, Wolkowicz H. The quasi-Cauchy relation and diagonal updating. *SIAM J. Optim.* 1999; **9**(4):1192–1204.
24. Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. *Math. Programming* 1989; **45**(3, (Ser. B)):503–528.
25. Veersé F, Auroux D, Fisher M. Limited memory BFGS diagonal preconditioners for a data assimilation problem in meteorology. *Optimization and Engineering* 2000; **1**(3):323–339.

26. Morales JL, Nocedal J. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM J. Optim.* 2000; **10**(4):1079–1096 (electronic).
27. Roma M. Dynamic scaling based preconditioning for truncated Newton methods in large scale unconstrained optimization. *Optimization Methods and Software* 2006; **20**:693–713.
28. Serra Capizzano S, Tablino Possio C. High-order finite difference schemes and Toeplitz based preconditioners for elliptic problems. *Electron. Trans. Numer. Anal.* 2000; **11**:55–84 (electronic).
29. Serra Capizzano S, Tablino Possio C. Preconditioning strategies for 2D finite difference matrix sequences. *Electron. Trans. Numer. Anal.* 2003; **16**:1–29 (electronic).
30. Bertaccini D, Golub GH, Serra Capizzano S, Tablino Possio C. Preconditioned HSS methods for the solution of non-Hermitian positive definite linear systems and applications to the discrete convection-diffusion equation. *Numer. Math.* 2005; **99**(3):441–484.
31. Kharchenko SA, Yeremin AY. Eigenvalue translation based preconditioners for the GMRES(k) method. *Numer. Linear Algebra Appl.* 1995; **2**(1):51–77.
32. Baglama J, Calvetti D, Golub GH, Reichel L. Adaptively preconditioned GMRES algorithms. *SIAM J. Sci. Comput.* 1998; **20**:243–269.
33. Loghin D, Ruiz D, Touhami A. Adaptive preconditioners for nonlinear systems of equations. *J. Comput. Appl. Math.* 2006; **189**(1-2):362–374.
34. Erhel J, Burrage K, Pohl B. Restarted GMRES preconditioned by deflation. *J. Comput. Appl. Math.* 1996; **69**(2):303–318.
35. Morgan RB. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.* 1995; **16**(4):1154–1171.
36. Saad Y, Yeung M, Erhel J, Guyomarc'h F. A deflated version of the conjugate gradient algorithm. *SIAM J. Sci. Comput.* 2000; **21**(5). Iterative methods for solving systems of algebraic equations (Copper Mountain, CO, 1998).
37. Parks ML, de Sturler E, Mackey G, Johnson DD, Maiti S. Recycling Krylov subspaces for sequences of linear systems. *Technical Report UIUCDCS-R-2004-2421*, University of Illinois 2004.
38. Wang S, de Sturler E, Paulino GH. Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. *Internat. J. Numer. Methods Engrg.* 2007; **69**(12):2441–2468.
39. de Sturler E, Le C, Wang S, Paulino G. Large scale topology optimization using preconditioned Krylov subspace recycling and continuous approximation of material distribution. *Multiscale and Functionally Graded Materials 2006 (M&FGM 2006), Oahu Island (Hawaii), 15-18 October 2006*, G H Paulino, M-J Pindera, R H Dodds, Jr, F A Rochinha, E Dave, and L Chen, (ed.). AIP Conference Proceedings, 2008; 279–284.
40. Kilmer ME, de Sturler E. Recycling subspace information for diffuse optical tomography. *SIAM J. Sci. Comput.* 2006; **27**(6):2140–2166 (electronic).
41. Golub GH, Ruiz D, Touhami A. A hybrid approach combining Chebyshev filter and conjugate gradient for solving linear systems with multiple right-hand sides. *SIAM J. Matrix Anal. Appl.* 2007; **29**(3):774–795 (electronic).
42. Giraud L, Gratton S, Martin E. Incremental spectral preconditioners for sequences of linear systems. *Appl. Numer. Math.* 2007; **57**(11-12):1164–1180.
43. Meurant G. On the incomplete Cholesky decomposition of a class of perturbed matrices. *SIAM J. Sci. Comput.* 2001; **23**(2):419–429 (electronic). Copper Mountain Conference (2000).
44. Benzi M, Bertaccini D. Approximate inverse preconditioning for shifted linear systems. *BIT* 2003; **43**(2):231–244.
45. Bertaccini D. Efficient preconditioning for sequences of parametric complex symmetric linear systems. *Electronic Transactions on Numerical Mathematics* 2004; **18**:49–64.
46. Duintjer Tebbens J, Tůma M. Improving triangular preconditioner updates for nonsymmetric linear systems. *LNCS* 2008; **4818**:737–744.
47. Birken P, Duintjer Tebbens J, Meister A, Tůma M. Preconditioner updates applied to CFD model problems. *Appl. Num. Math.* 2008; **58**(11):1628–1641.
48. Griewank A, Toint PL. On the unconstrained optimization of partially separable functions. *Nonlinear optimization, 1981 (Cambridge, 1981)*. NATO Conf. Ser. II: Systems Sci., Academic Press: London, 1982; 301–312.
49. Curtis AR, Powell MJD, Reid JK. On the estimation of sparse Jacobian matrices. *J. Inst. Maths. Applics.* 1974; **13**:117–119.
50. Coleman TF, Moré JJ. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.* 1983; **20**:187–209.
51. Gebremedhin AH, Manne F, Pothen A. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review* 2005; **47**:629–705.
52. Garey MR, Johnson DS. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Freeman & Co., 1979.

53. Cuthill EH, McKee J. Reducing the bandwidth of sparse symmetric matrices. *Proc.* 24^{th} *National Conference of the ACM*, ACM Press, 1969; 157–172.
54. Liu JWH, Sherman AH. Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. *SIAM J. Numer. Anal.* 1976; **13**:198–213.
55. Benzi M, Szyld DB, van Duin A. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM J. Sci. Comput.* 1999; **20**(5):1652–1670.
56. Hartmann S, Duintjer Tebbens J, Quint K, Meister A. Iterative solvers within sequences of large linear systems in non-linear structural mechanics. *Prepint submitted in* 2008.
57. Saad Y. ILUT: a dual threshold incomplete *LU* factorization. *Numer. Linear Algebra Appl.* 1994; **1**(4):387–402.
58. Lukšan L, Tůma M, Vlček J, Ramešová N, Šiška M, Hartman J, Matonoha C. UFO 2004 - interactive system for universal functional optimization. *Technical Report V-923*, ICS AS CR 2004.
59. Kaporin IE, Axelsson O. On a class of nonlinear equation solvers based on the residual norm reduction over a sequence of affine subspaces. *SIAM J. Sci. Comput.* 1995; **16**(1):228–249.