New updates of incomplete LU factorizations and applications to large nonlinear systems

by S. Bellavia, B. Morini and M. Porcelli

Report naXys-02-2012 February 2, 2012

University of Namur
8, rempart de la Vierge, B5000 Namur (Belgium)

http://www.naxys.be

New updates of incomplete LU factorizations and applications to large nonlinear systems *

S. Bellavia†, B. Morini‡, and M. Porcelli§

February 2, 2012

Abstract

In this paper, we address the problem of preconditioning sequences of large sparse nonsymmetric systems of linear equations and present two new strategies to construct approximate updates of factorized preconditioners. Both updates are based on the availability of an incomplete LU (ILU) factorization for one matrix of the sequence and differ in the approximation of the so-called *ideal* updates. The first strategy is an approximate diagonal update of the ILU factorization; the second strategy relies on banded approximations of the factors in the ideal update. The efficiency and reliability of the proposed preconditioners are shown in the solution of nonlinear systems of equations by preconditioned inexact Newton-Krylov methods. Matrix-free implementations of the updating strategy are provided and numerical experiments are carried out on application problems.

Keywords: Preconditioning, incomplete factorizations, factorization updates, inexact Newton-Krylov methods, matrix-free environment.

1 Introduction

The need to solve large, sparse nonsymmetric systems of linear equations is frequent in numerical optimization. Such sequences are generated by Newton-type methods for the nonlinear systems appearing in equilibrium systems, macroeconometric models and the discretization of partial differential equations and integro-differential equations, see e.g. [15, 16, 17]. In this context, Newton-type methods coupled with preconditioned Krylov subspace methods and procedures to enhance convergence from arbitrary starting point are extensively used, see e.g. [1, 4, 14, 17, 22, 23].

Let $F(x) = 0$ be a system of nonlinear equations where $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ is continuously differentiable and let J be the Jacobian matrix of F. Each iteration of a globally convergent Newton-Krylov method comprises the solution of the linear system

$$J_k s = -F_k, \tag{1}$$

*Work supported in part by INDAM-GNCS grant "Progetti GNCS 2011", "Metodi numerici avanzati per problemi di ottimizzazione non lineare vincolata di grandi dimensioni"

†Dipartimento di Energetica "S. Stecco", Università di Firenze, viale G.B. Morgagni 40, 50134 Firenze, Italia. Email: stefania.bellavia@unifi.it

‡Dipartimento di Energetica "S. Stecco", Università di Firenze, viale G.B. Morgagni 40, 50134 Firenze, Italia. Email: benedetta.morini@unifi.it

§Namur Center for Complex Systems (NAXYS), FUNDP-University of Namur, 61, rue de Bruxelles, B-5000 Namur, Belgium. Email: margherita.porcelli@fundp.ac.be

where x_k is the current iteration, $F_k = F(x_k)$, $J_k = J(x_k)$, and a phase dedicated to a globalization procedure. In general, the dominant cost of the whole procedure is constituted by the solution of the sequence of linear systems (1) generated through the iterations. This observation motivates the recent growing interest in improving the solution of the overall sequence of systems by sharing some of the computational effort throughout the sequence. In fact, the cost of solving subsequent systems may be reduced by using information available from earlier systems.

Some alternative ideas have been proposed for handling the solution of a sequence of large nonsymmetric linear systems effectively. A first possibility is to recycle selected Krylov subspaces generated from previous systems, [21]. A second possibility is to improve an existing incomplete LU (ILU) preconditioner progressively within the solution of the sequence, [10]. A third possibility is to update a preconditioner computed for some specific system by inexpensive strategies; this can be done by Broyden-type rank-one updates ([8]) or by fully-algebraic procedures like those in [2, 9, 12, 13].

The preconditioner updates in [2, 9, 12, 13] are inspired by the attempt to cheaply approximate the *ideal* update to an existing preconditioner. Let J_s be the Jacobian matrix at some iteration of the Newton-Krylov method, and let

$$P_s = LDU, \tag{2}$$

be an ILU factorization of J_s where D is diagonal and L and U are lower and upper triangular matrices, respectively, with unit main diagonal. If $J_s \approx LDU$, then using the equality $J_k = J_s + (J_k - J_s)$ we get

$$J_k \approx L(D + L^{-1}(J_k - J_s)U^{-1})U. \tag{3}$$

Then, the matrix

$$P_k^I = L(D + L^{-1}(J_k - J_s)U^{-1})U, \tag{4}$$

represents the ideal updated preconditioner in the sense that, for any matrix norm $\|\cdot\|$, its accuracy $\|J_k - P_k^I\|$ for J_k is equal to the accuracy $\|J_s - P_s\|$, [12].

It is evident that the ideal update is not suitable for practical use. In general, the matrix $L^{-1}(J_k - J_s)U^{-1}$ is expensive to build; moreover P_k^I is dense and its factorization is impractical. Updating strategies are based on simple approximations to $L^{-1}(J_k - J_s)U^{-1}$ and give rise to preconditioners which are products of matrices easy to invert.

In this paper, by using structured approximations of the factors in P_k^I, we propose two techniques to form updated preconditioners. The first one is based on the approximation of $J_k - J_s$ by its diagonal. This yields to a diagonal modification of the ILU factorization and to a new strategy to form an approximate cheap factorization of the resulting matrix. The second technique employs structured approximations of $J_k - J_s$, L^{-1} and U^{-1}; in this context, we propose an algorithm to compute a fixed number of diagonals, below and above the main one, in L^{-1} and U^{-1} without the need of a complete inversion of L and U.

Unpreconditioned Newton-Krylov methods do not require forming or storing the Jacobians as their action on vectors required by the Krylov solver can be approximated by finite differences. On the other hand, the computation of an algebraic preconditioner at each nonlinear iteration is no longer a matrix-free procedure. However, a preconditioning strategy is considered to be nearly matrix-free if it has the following properties: a few full Jacobians are formed; in most Newton iterations, matrices that are reduced in complexity with respect to the full Jacobians are required; matrix-vector product approximations by finite differences can be used; see [13, 17]. It deserves to be mentioned that our *approximate updating* strategies, as well as those proposed in [2, 13], share these properties which are valuable for real applications.

The paper is organized as follows. In Section 2 we describe the updating proposals given in [2, 12]. In Section 3 we present and analyze our new strategies. In Section 4 we discuss how these

techniques can be implemented and finally, we present the results of numerical experiments in Section 5.

Notations

Unless explicitly stated, $\|\cdot\|$ denotes an arbitrary vector norm and induced matrix norm. The entries of a matrix $A \in \mathbb{R}^{n\times n}$ are denoted either as a_{ij} or $(A)_{ij}$. Given a nonnegative integer γ, $[A]_\gamma \in \mathbb{R}^{n\times n}$ indicates the banded matrix obtained extracting from A the main diagonal and γ upper and lower diagonals. If $\gamma = 0$, $[A]_0$ is the diagonal matrix formed from the elements of the diagonal of A. The off-diagonal part $A - [A]_0$ of A si denoted by the symbol $\mathit{off}(A)$. If A is lower (upper) triangular, $[A]_\gamma$ is obtained extracting from A the main diagonal and γ lower (upper) diagonals. Similarly, $\mathit{off}(A)$ is formed by the lower (upper) extra diagonals of A. Finally, we borrow notations used by Matlab in linear algebra: *diag, tril, triu, eye, zeros.*

2 Preliminaries

Let J_s and P_s be the matrices introduced in §1; hereafter, we will refer to them as the *seed* Jacobian and *seed* preconditioner, respectively. Thus, we are assuming that an incomplete factorization process of J_s can be carried out without breakdowns; this is guaranteed when the matrix has a strong diagonal, [6, 12]. For the rest of the paper we also assume that the Jacobian matrices J_k in (1) are invertible.

We now overview the approximate updating of P_s proposed in the papers [2, 12] and point out the sources of errors introduced in the replacement of P_k^I with a matrix easier to invert.

A fundamental aspect of the updating procedures is the possibility of approximating L^{-1} and U^{-1} or to have such inverses available. This issue is tackled in different ways in view of the kind of the computed seed preconditioner, see [6]. The preconditioner P_s we are currently considering can be classified as *implicit*, as its application requires the solution of linear systems. Another possibility is the availability of an *explicit* preconditioner, that is a preconditioner that approximates J_s^{-1} and that can be applied by performing matrix-vector products.

The practical updating techniques proposed by Duintjer Tebbens and Tuma in [12] are based on the implicit preconditioner P_s given in (2). They rely on suitable approximations to $J_k - J_s$ and on the assumption that either L or U more or less approximates the identity matrix. If L is close to the identity matrix and L^{-1} is neglected in (3), such expression becomes

$$J_k \approx L(\, D + (J_k - J_s)U^{-1}\,)U = L(\, DU + (J_k - J_s)\,), \tag{5}$$

while assuming U to be close to the identity, and neglecting U^{-1} it follows

$$J_k \approx L(D + L^{-1}(J_k - J_s))U = (LD + (J_k - J_s))U. \tag{6}$$

Several options have been proposed in [12] to replace $DU + (J_k - J_s)$ or $LD + (J_k - J_s)$ by a nonsingular and easily invertible approximation. One option is to approximate $J_k - J_s$ in (5) or (6) by its upper/lower triangular part which gives rise to

$$P_k = L(\, DU + triu(J_k - J_s)), \tag{7}$$

$$P_k = (LD + tril(J_k - J_s))U, \tag{8}$$

respectively. Thus, the required information on J_k is reduced in complexity with respect to the full matrix. Furthermore, the application of P_k does not involve any factorization and consists in solving two triangular systems. This strategy tacitly assumes that one triangular part of $J_k - J_s$

dominates the other and a typical situation of this kind arises in CFD model problems, [9]. Interestingly, nearly matrix-free implementations of this approach are possible and the explicit knowledge of the upper/lower triangular part of J_k is not required, see [13].

Since important information may be lost retaining only one triangular part of $J_k - J_s$, an alternative option replaces $DU + (J_k - J_s)$ in (5) or $LD + (J_k - J_s)$ in (6) by an unstructured matrix with inverse available in closed form. Such unstructured approximation is constructed by using products of Gauss-Jordan transformations, [12].

The introduced updated preconditioners have the potential to be more effective than reusing P_s (*frozen preconditioner*) under the above mentioned assumptions on L and U and under the assumption that the used approximation of $J_k - J_s$ is accurate enough, [12, Lemma 2.1]. An analogous result can be stated with respect to a *recomputed preconditioner* for J_k, [12, Theorem 2.2].

We conclude this section with the method of updating the seed preconditioner presented by Bellavia et al. in [2]. The proposed strategy is based on the observation that P_k^I requires the knowledge of L^{-1} and U^{-1}; hence it uses an explicit seed preconditioner, i.e. an incomplete factorization for J_s^{-1} of the form

$$J_s^{-1} \approx W\, D^{-1} Z^T, \tag{9}$$

where D is a diagonal matrix, and W and Z are sparse unit upper triangular matrices. A possibility to construct this sparse approximate inverse preconditioner is to use the incomplete generalized Gram-Schmidt orthogonalization process with respect to the bilinear form associated to J_s given in [6]. Alternatively, one can first compute an ILU factorization of J_s and then approximately invert L and U; the latter task can be performed by solving a triangular system for each column of L^{-1} and U^{-1}, [7].

Exploiting factorization (9) and the relation $J_s \approx Z^{-T}\, DW^{-1}$, equality $J_k = J_s + (J_k - J_s)$ yields

$$J_k \approx Z^{-T} \left(D + Z^T (J_k - J_s) W\right) W^{-1}.$$

Then, replacing $J_k - J_s$ by the banded matrix $[J_k - J_s]_\delta$, and $Z^T[J_k - J_s]_\delta W$ by the banded matrix $[Z^T[J_k - J_s]_\delta W]_\beta$, for some nonnegative β and δ, the updated explicit preconditioner for J_k is

$$P_k = W \left(D + [Z^T[J_k - J_s]_\delta W]_\beta\right)^{-1} Z^T. \tag{10}$$

Note that P_k approximates J_k^{-1} and that only the main diagonal and δ lower and upper diagonals of J_k need to be computed for its construction. When $\beta = 0$, the middle factor in P_k is diagonal and the application of the preconditioner is straightforward; on the other hand if $\beta > 0$ the application of P_k requires the solution of banded linear systems with matrix $D + [Z^T[J_k - J_s]_\delta W]_\beta$. In terms of computational cost, only small values of β and δ are viable and if β is nonnull, direct methods for banded systems are convenient.

The effectiveness of this approach depends on two issues. First, it is assumed that the approximation of J_s^{-1} in terms of sparse factors is possible. This means that many entries of J_s^{-1} are small in magnitude and in its factorization many entries can be discarded while retaining accuracy. Second, the accuracy in the replacement of the ideal update depends on the magnitude of the elements dropped away in $J_k - J_s$ and in $Z^T[J_k - J_s]_\delta W$. These two issues are related in the sense that explicit preconditioners are satisfactory when the entries of J_s^{-1}, Z and W decay away from the diagonals, and banded approximations of matrices tend to contain most of their large entries, [7].

The preconditioner (10) can be more efficient than the frozen preconditioner and than a recomputed preconditioner for J_k under the assumption that the seed preconditioner and the

approximations $[J_k - J_s]_\delta$, and $[Z^T[J_k - J_s]_\delta W]_\beta$ to $J_k - J_s$, and $Z^T[J_k - J_s]_\delta W$ respectively, are both accurate, see [2, §4.3].

3 New techniques based on structured updates

Our techniques attempt to approximate P_k^I by extracting banded parts of the matrices appearing in it.

The first procedure takes the main diagonal of the matrix $J_k - J_s$ and does not involve L^{-1} and U^{-1}; the resulting preconditioner is named Diagonally Updated ILU (DU_ILU) factorization. The second procedure allows for larger bandwidths and require a partial knowledge of L^{-1} and U^{-1}; we will refer to resulting preconditioner as Banded Updated ILU (BU_ILU) preconditioner.

The quality of the preconditioners will depend on the size of discarded quantities from P_k^I; if such quantities are small, the updated preconditioners are expected to be powerful. Remarkably, some information on the size of $J_k - J_s$ is problem-independent and can be obtained from the convergence theory of Newton-Krylov methods. Specifically, suppose that J is Lipschitz continuous in a convex set containing a solution of the nonlinear systems and that the iterate x_s such that $J_s = J(x_s)$ is close enough to such solution to guarantee local convergence of the Newton method. Then, the iterates x_k subsequent to x_s converge to the solution and $\|J_k - J_s\| = O(\|x_k - x_s\|)$ is small.

3.1 The DU_ILU preconditioner

Let Σ_k be the diagonal matrix

$$\Sigma_k = [J_k - J_s]_0, \tag{11}$$

and σ_{ii}^k, $i = 1, \ldots, n$, be its diagonal entries. The DU_ILU preconditioner is built by replacing $J_k - J_s$ with Σ_k in the ideal preconditioner (4). As a results, P_k^I is approximated by

$$LDU + \Sigma_k,$$

which amounts to a diagonal modification of the available ILU factorization. The factorization of matrix $LDU + \Sigma_k$ is impractical in the context of the updates as its cost is comparable to that of a recomputed preconditioner. We do not factorize it exactly but, inspired by [3], form a cheap approximate factorization; we describe this approach in the following algorithm.

The matrix P_k computed by Algorithm 3.1 is our DU_ILU preconditioner. It takes the place of $LDU + \Sigma_k$ and shows some nice computational features. First, it is expressed in factorized form. Second, the cost to form P_k is low: the computation of D_k is negligible while the computation of L_k and U_k consists in scaling the nonzero entries of L and U. Third, by construction, $s_{ii}^k \in (0, 1]$, $i = 1, \ldots, n$, and it is known that the conditioning of the matrices L_k and U_k is at least as good as the conditioning of L and U respectively, [18]. Further algorithmic issues are postponed to Section 4.

Let now analyze the quality of the DU_ILU preconditioner. We have

$$\begin{aligned} \|J_k - P_k\| &\leq \|J_k - (LDU + \Sigma_k)\| + \|LDU + \Sigma_k - P_k\| \\ &\leq \|J_s - P_s\| + \|J_k - J_s - \Sigma_k\| + \|LDU + \Sigma_k - P_k\|. \end{aligned} \tag{16}$$

The norm $\|J_s - LDU\|$ measures the accuracy of the seed preconditioner while the second and third term in (16) can be ascribed to the DU_ILU procedure. In fact, the DU_ILU algorithm

Algorithm 3.1: DU_ILU PRECONDITIONER

Given $J_s, P_s = LDU$ of dimension $n \times n$.

1. Compute Σ_k in (11).
2. Set

$$D_k = D + \Sigma_k, \tag{12}$$

$$S_k = diag(s_{11}^k, \ldots, s_{nn}^k), \qquad s_{ii}^k = \frac{|d_{ii}|}{|d_{ii}| + |\sigma_{ii}^k|},\ i = 1, \ldots, n, \tag{13}$$

$$L_k = eye(n), \qquad U_k = eye(n), \tag{14}$$
$$off\,(L_k) = off\,(L)S_k, \qquad off\,(U_k) = S_k off\,(U).$$

3. Let

$$P_k = L_k D_k U_k. \tag{15}$$

introduces the second term in (16) by neglecting the off-diagonal part of $J_k - J_s$ and the third term in (16) by replacing $LDU + \Sigma_k$ with P_k. As shown below, P_k is accurate for $LDU + \Sigma_k$ as long as Σ_k is sufficiently small. Hence, when P_s is accurate for J_s and the sequence of matrices $\{J_k\}$ is slowly varying, i.e. $\|J_k - J_s\|$ is small, P_k is expected to be accurate.

To analyze the accuracy of P_k as an approximation to $LDU + \Sigma_k$, we introduce the error matrix

$$E_k = LDU + \Sigma_k - P_k, \tag{17}$$

and provide the expression for its entries $E_k = (e_{ij}^k)$. An analogous notation for the entries of D_k, L_k, U_k, is used. First, consider the diagonal entries of E_k,

$$e_{ii}^k = \sum_{r=1}^{i-1} l_{ir} u_{ri} d_{rr} + d_{ii} + \sigma_{ii}^k - \sum_{r=1}^{i-1} l_{ir}^k u_{ri}^k d_{rr}^k - d_{ii}^k.$$

By (12) and (15),

$$e_{ii}^k = \sum_{r=1}^{i-1} l_{ir} u_{ri} (d_{rr} - (s_{rr}^k)^2 d_{rr}^k),$$

i.e.

$$[E_k]_0 = [off\,(L)(D - S_k^2 D_k) off\,(U)]_0. \tag{18}$$

Second, we give the analytical form of the off-diagonal entries of E_k, $e_{ij}^k = (LDU)_{ij} - (L_k D_k U_k)_{ij}$. Suppose $i > j$, then by using (15) and the fact that L and U have unit diagonal, we get

$$\begin{aligned} e_{ij}^k &= \sum_{r=1}^{j} l_{ir} u_{rj} d_{rr} - \sum_{r=1}^{j} l_{ir}^k u_{rj}^k d_{rr}^k \\ &= \sum_{r=1}^{j-1} l_{ir} u_{rj} (d_{rr} - (s_{rr}^k)^2 d_{rr}^k) + l_{ij}(d_{jj} - s_{jj}^k d_{jj}^k). \end{aligned}$$

Proceeding analogously for the entries e_{ij} with $i < j$, we can conclude that

$$E_k - [E_k]_0 = off(off(L)(D - S_k^2 D_k)off(U)) + off(L)(D - S_k D_k) + (D - S_k D_k)off(U),$$

and by (18)

$$E_k = off(L)(D - S_k^2 D_k)off(U) + off(L)(D - S_k D_k) + (D - S_k D_k)off(U). \tag{19}$$

The following lemma expresses the relation between E_k and Σ_k and indicates that for $\|\Sigma_k\|$ small enough, P_k is an accurate approximation to $LDU + \Sigma_k$.

Theorem 3.1 *Let P_k and E_k be defined in (15) and (17). Then, there exists a positive scalar c, independent of k, such that $\|E_k\| \leq c\|\Sigma_k\|$.*

Proof. By taking into account (19), it is sufficient to prove that

$$\|D - S_k^2 D_k\| \leq \hat{c}\|\Sigma_k\|, \qquad \|D - S_k D_k\| \leq \hat{c}\|\Sigma_k\|,$$

for some positive $\hat{c}$. We proceed entrywise and start supposing that $d_{ii}\sigma_{ii}^k > 0$. Then, $s_{ii}^k = d_{ii}/(d_{ii} + \sigma_{ii}^k)$, and

$$\begin{aligned} &\left|d_{ii} - (s_{ii}^k)^2 d_{ii}^k\right| = \left|\frac{d_{ii}}{d_{ii} + \sigma_{ii}^k}\ \sigma_{ii}^k\right| \leq \left|\sigma_{ii}^k\right|, \\ &d_{ii} - s_{ii}^k d_{ii}^k = 0. \end{aligned}$$

In case $d_{ii}\sigma_{ii}^k < 0$, without loss of generality suppose $d_{ii} > 0$ and $\sigma_{ii}^k < 0$. Then, $s_{ii}^k = d_{ii}/(d_{ii} - \sigma_{ii}^k)$, and

$$\begin{aligned} &\left|d_{ii} - (s_{ii}^k)^2 d_{ii}^k\right| = \left|\frac{d_{ii}}{d_{ii} - \sigma_{ii}^k}\left(3 + \frac{2\sigma_{ii}^k}{d_{ii} - \sigma_{ii}^k}\right)\ \sigma_{ii}^k\right| \leq 5\left|\sigma_{ii}^k\right|, \\ &|d_{ii} - s_{ii}^k d_{ii}^k| = \left|\frac{2d_{ii}}{d_{ii} - \sigma_{ii}^k}\ \sigma_{ii}^k\right| \leq 2\left|\sigma_{ii}^k\right|. \end{aligned}$$

Thus, the proof is completed. □

3.2 Banded approximate inverses

The two approaches presented in §2 tacitly assume that L^{-1} and U^{-1} contain many entries which are small in magnitude.

Usually, the inverse of a sparse matrix A is dense, and it is not known in advance a good sparsity pattern for an approximation to A^{-1}. Possible exceptions are large classes of matrices where the entries of the inverse tend to zero away from the main diagonal. Classes of matrices with decaying inverses were detected and analyzed in several papers: banded symmetric positive definite and indefinite matrices, [11, 19]; nonsymmetric block tridiagonal banded matrices, [20]; matrices of the form $f(A)$ where A is symmetric and banded and f is analytic, [5]. Typically, the rate of decay of the entries of A^{-1} is fast if A is diagonally dominant.

For the mentioned matrices, it appears justified the approximation of the inverse by a banded matrix. This approach was applied to preconditioning techniques for symmetric matrices in the papers [5, 24]. We proceed along such lines and introduce a suitable technique for building band approximate inverses of L and U and of the middle factor in P_k^I.

Our first contribution is an algorithm for computing exactly the matrices $[L^{-1}]_\gamma$ and $[U^{-1}]_\gamma$ without the complete inversion of L and U. The closed form of $[L^{-1}]_\gamma$ and $[U^{-1}]_\gamma$ is derived algebraically as follows. Consider the upper unit triangular matrix U, and let u_i^T, $i = 1, \ldots, n-1$, be ith row of matrix $\mathit{off}(U)$, i.e. $u_i^T = (0, \ldots, 0, u_{ii+1}, \ldots, u_{in})$. Analogously, let v_i^T, $i = 1, \ldots, n-1$, be the ith row of matrix $\mathit{off}(U^{-1})$. Trivially,

$$U^{-1} = I + \sum_{i=1}^{n-1} e_i v_i^T, \tag{20}$$

where e_i is the i-th vector of the canonical basis. By using the closed form of the inverses of the elementary matrices,

$$U^{-1} = \left(\prod_{i=n-1}^{1} (I + e_i u_i^T) \right)^{-1} = \prod_{i=1}^{n-1} (I - e_i u_i^T).$$

Note that

$$\begin{aligned} U^{-1} &= \prod_{i=2}^{n-1} (I - e_i u_i^T) - e_1 u_1^T \prod_{i=2}^{n-1} (I - e_i u_i^T) \\ &= \prod_{i=3}^{n-1} (I - e_i u_i^T) - e_1 u_1^T \prod_{i=2}^{n-1} (I - e_i u_i^T) - e_2 u_2^T \prod_{i=3}^{n-1} (I - e_i u_i^T) \\ &= \cdots \\ &= I - \sum_{j=1}^{n-1} e_j u_j^T \prod_{i=j+1}^{n-1} (I - e_i u_i^T), \end{aligned} \tag{21}$$

with $\prod_{j=n}^{n-1} (I - e_j u_j^T) = I$. Then, comparing (20) with (21), we get

$$v_i^T = -u_i^T \prod_{j=i+1}^{n-1} (I - e_j u_j^T), \qquad i = 1, \ldots, n-1.$$

From the sparsity pattern of the matrices $(I - e_j u_j^T)$, it follows that the product $u_i^T (I - e_{i+1} u_{i+1}^T)$ modifies the components of u_i of indices $i+2, \ldots, n$; letting $u_i^{(1)} = u_i^T (I - e_{i+1} u_{i+1}^T)$, the product $(u_i^1)^T (I - e_{j+2} u_{j+2}^T)$ modifies the components of $u_i^{(1)}$ of indices $i+3, \ldots, n$; etc. This has the effect that the l-th entry of v_i^T, $l = i+1, \ldots, i+\gamma$, coincide with l-th entry of

$$-u_i^T \prod_{j=i+1}^{\min\{i+\gamma, n\}-1} (I - e_j u_j^T).$$

Interestingly, the above expressions indicate that each row of $[U^{-1}]_\gamma$ can be computed independently from the others and such computation is exact without the need of a complete inversion of U. Algorithm 3.2 below sketches the details of the computation of matrix $[U^{-1}]_\gamma$ and is written borrowing from Matlab the notations to handle arrays and matrices.

Algorithm 3.2: COMPUTATION OF $[U^{-1}]_\gamma$

Given $U \in \mathbb{R}^{n\times n}$ upper unit triangular, γ nonnegative integer.

1. $V = eye(n)$.
2. For $i = 1, \ldots, n-1$
   $l = \min\{i+\gamma, n\}$.
   $V(i, i+1:l) = -U(i, i+1:l)$.
   For $j = i+1, \ldots, l-1$
   $w = [zeros(1, j-i), U(j, j+1:l)]$.
   $V(i, i+1:l) = V(i, i+1:l) - V(i, j)w$.
   end
   end
3. $[U^{-1}]_\gamma = V$.

An alternative algorithm for computing $[U^{-1}]_\gamma$ was presented in [24, Algorithm CHOL]. It evaluates the rows of $[U^{-1}]_\gamma$ from the $(n-1)$-th to the first sequentially, whereas in Algorithm 3.2 the rows can be calculated in parallel.

Clearly, the computation of $[L^{-1}]_\gamma$ with L lower unit triangular, can be performed by applying Algorithm 3.2 to the transpose of the matrix L and then transposing again the resultant banded matrix.

Algorithm 3.2 is used to build our BU_ILU preconditioner as sketched in Algorithm 3.3.

Algorithm 3.3: BU_ILU PRECONDITIONER

Given $J_s, P_s = LDU$, β, γ, δ nonnegative integers.

1. Compute $[L^{-1}]_\gamma$ and $[U^{-1}]_\gamma$ by Algorithm 3.2.
2. Compute $[J_k - J_s]_\delta$.
3. Set

$$P_k = L \left[D + [L^{-1}]_\gamma [J_k - J_s]_\delta [U^{-1}]_\gamma \right]_\beta U. \tag{22}$$

Details on the implementation of Algorithm 3.3 are postponed to Section 4. Here we observe that the way to apply P_k depends on the value of β. The most favorable case is $\beta = 0$ wherein one diagonal and two triangular systems must be solved. Otherwise, the factorization of the banded middle factor in P_k is necessary.

We further mention that Algorithm 3.3 provides a preconditioner for J_k using information from both the lower and upper tridiagonal part of the current Jacobian, differently from the approach given in [12] where either the lower or the upper part of J_k is discarded.

We conclude this section providing an upper bound to the distance of the BU_ILU preconditioner from J_k. In the next theorem we show that such distance depends on the quality of P_s for J_s and on the distance between J_k and J_s. Then, an accurate seed preconditioner and a slowly varying sequence $\{J_k\}$ can yield an accurate BU_ILU preconditioner.

Theorem 3.2 *Let P_k be the* BU_ILU *preconditioner in (22). Then,*

$$\|J_k - P_k\| \leq \|J_s - LDU\| + \bar{c}\|J_k - J_s\|, \tag{23}$$

where $\|\cdot\|$ denotes the 1 or the infinity norm and $\bar{c}$ is some positive scalar independent of k.

Proof. As β is nonnegative, it trivially follows

$$P_k = LDU + L\left[[L^{-1}]_\gamma [J_k - J_s]_\delta [U^{-1}]_\gamma\right]_\beta U,$$

and

$$\|J_k - P_k\| \leq \|J_s - LDU\| + \|J_k - J_s\| + \|L\left[[L^{-1}]_\gamma [J_k - J_s]_\delta [U^{-1}]_\gamma\right]_\beta U\|.$$

By the properties of the 1 or the infinity norm, we have $\|[A]_\gamma\| \leq \|A\|$, for any matrix A and nonnegative γ. As a consequence, we get (23) with $\bar{c} = 1 + cond(L)cond(U)$ and $cond(\cdot)$ being the condition number of a matrix. □

4 Practical preconditioned Newton-Krylov algorithms

In this section we discuss the main implementation aspects of the DU_ILU and BU_ILU procedures embedded in Newton-Krylov methods for the nonlinear system $F(x) = 0$. We address issues related to the nearly matrix-free implementations and the possible breakdowns of the algorithms proposed. Further, we give some details on the implementation of the Newton-Krylov solver used in our numerical experiments and outline the adopted strategy to refresh the seed preconditioner when the performance of the updating strategy deteriorates.

We begin showing that our preconditioners can be implemented in a nearly matrix-free manner. To form the algebraic seed preconditioner, the full seed Jacobian has to be formed. Thereafter, it is required the knowledge of the matrices $[J_k]_\delta$, $\delta \geq 0$, which are reduced in complexity with respect to the full Jacobians. It is clearly possible to build and store their entries by finite differences, [17]. An implementation of the BU_ILU preconditioner alternative to approximating $[J_k]_\delta$ and then performing the product $[J_k - J_s]_\delta [U^{-1}]_\gamma$, is to directly approximate such product by finite differences. In fact, proceeding as described in [13], once $[U^{-1}]_\gamma$ has been formed, each column of the matrix $\Phi = [J_k]_\delta [U^{-1}]_\gamma$ can be computed element-wise as follows. Assuming for sake of simplicity that $i - \delta \geq 1$ and $i + \delta \leq n$, and taking into account the structure of $[J_k]_\delta$, the entries ϕ_{ij} of Φ have the form

$$\phi_{ij} = \sum_{l=i-\delta}^{i+\delta} (J_k)_{i,l} (U^{-1})_{l,j}.$$

Then, by the structure of $[U^{-1}]_\gamma$, the expression of ϕ_{ij} reduces to

$$\phi_{ij} = \sum_{l \in L} (J_k)_{i,l} (U^{-1})_{l,j}, \tag{24}$$

where $L = \{l : l \in [l_m, l_M], l_m = \max\{i - \delta, j - \gamma\}, l_M = \min\{i + \delta, j\}\}$. If L is empty, i.e. $l_m > l_M$, then $\phi_{ij} = 0$. Otherwise, (24) can be written as

$$\phi_{ij} = w_i, \qquad w = (w_1, w_2, \ldots, w_n)^T = J_k \bar{u},$$

with $\bar{u} = (0, \ldots, 0, (U^{-1})_{l_m j}, \ldots, (U^{-1})_{l_M j}, 0, \ldots, 0)^T \in \mathbb{R}^n$ and w_i can be approximated by finite differences which amounts to evaluate the ith component of F at $x_k + \epsilon\bar{u}$, for some scalar ϵ. We conclude noting that only the components of $[J_k - J_s]_\delta [U^{-1}]_\gamma$ corresponding to nonzero entries of $[L^{-1}]_\gamma$ in the computation of $[L^{-1}]_\gamma [J_k - J_s]_\delta [U^{-1}]_\gamma$ are required.

Another issue that deserves consideration is the potential need to freeze the updated preconditioner or to compute a new seed preconditioner from scratch (*refreshed preconditioner*). In fact, Algorithms 3.1 and 3.3 need to be safeguarded against the risk of singular or nearly singular middle factors in the updated preconditioners, i.e. D_k in (15) and $\left[\ D + [L^{-1}]_\gamma\, [J_k - J_s]_\delta\, [U^{-1}]_\gamma\ \right]_\beta$ in (22). The construction of a singular preconditioner implies a *breakdown* in the updating procedure but it is unlikely that the mentioned middle factors will be exactly singular. On the other hand, the application of the updated preconditioner to a vector may indicate a serious ill-conditioning of such matrix. In this case, it is necessary to abandon the updated preconditioner and either replace it by the one used in the previous Newton equation or to refresh the seed Jacobian and preconditioner.

The occurrence of a nearly singular middle factor in the updated preconditioners is monitored for the DU_ILU procedure by the condition

$$\min_{i=1,\ldots,n} |(D_k)_{ii}| \leq \tau \|J_s\|_1, \tag{25}$$

for some small positive τ, see [2]. An analogous control is performed in the BU_ILU preconditioner whenever $\beta = 0$ is used in (22). If such condition is met, the candidate updated preconditioner is abandoned and replaced by the one used in the previous Newton iteration; in other words, the preconditioner from the previous Newton iteration is frozen.

Moreover, the effectiveness of the preconditioned Krylov method in solving the linear systems (1) may also suggest the need for refreshing the preconditioner. In a Newton-Krylov method, the linear system (1) is solved approximately and the computed step s_k is required to satisfy

$$\|J_k s_k + F_k\|_2 \leq \eta_k \|F_k\|_2, \tag{26}$$

with $\eta_k \in [0, 1)$. If the approximate updating strategy DU_ILU or BU_ILU neglects important information from the ideal update P_k^I, the preconditioned Krylov method is expected to slow down. For this reason, a practical scheme must combine the updating algorithms DU_ILU and BU_ILU with the recomputation of the preconditioner when the quality of the updates deteriorates, [2, 9]. We follow the strategy to refresh the seed preconditioner, i.e. to compute a new seed Jacobian and a new seed preconditioner, when the Krylov method fails.

To enlarge the convergence region of the Newton-Krylov solver, we embedded it into a backtracking strategy. Then, once the step s_k is computed, a steplenght α is selected in order to satisfy the following decrease condition:

$$\|F(x_k + \alpha s_k)\|_2 < (1 - \alpha\lambda)\|F_k\|_2, \quad \lambda \in (0, 1), \tag{27}$$

and the new iterate is $x_{k+1} = x_k + \alpha s_k$. To compute the steplenght α, a backtracking strategy is usually performed and we adopted the three point parabolic rule [23]. If the backtracking strategy fails in producing an acceptable step within $\texttt{NBT}_{\texttt{max}}$ backtracks, we left the iterate unchanged, i.e. we set $x_{k+1} = x_k$ and refresh the preconditioner.

5 Numerical results

In this section we show the numerical performance of the DU_ILU and BU_ILU preconditioned Newton-Krylov solvers described in the previous section. Our aim is to assess the reliability and

efficiency of the new proposals in a matrix-free setting, also in comparison with the techniques presented in [12, 13].

5.1 Implementation details

The sequence of linear systems (1) is solved by the Krylov solver BiCGSTAB supplied by Matlab and right preconditioning. Starting from the null initial guess, a maximum of $\texttt{LI}_{\texttt{max}} = 400$ linear iterations is allowed.

In order to operate in a matrix-free manner, the seed Jacobians are computed by finite differences and the matrices $[J_k]_\delta$ are built by using finite differences and stored. The action of the Jacobians J_k on vectors required by BiCGSTAB is approximated by finite differences.

The seed preconditioners is constructed by using the `luinc` Matlab function which produces two different kinds of incomplete LU factorizations: the drop tolerance and the 0 level of fill-in factorizations. We use the drop tolerance factorization with tolerances depending on the problem and force diagonal pivoting.

We chose the forcing term η_k in (26) in order to achieve the desirable fast local convergence near a solution and, the same time, to minimize the oversolving. Following the results by Eisenstat and Walker [23], we set $\eta_0 = \eta_{max} = 10^{-2}$ and used the so-called adaptive *Choice* 2

$$\eta_k = \chi \left(\frac{\|F_{k+1}\|_2}{\|F_k\|_2} \right)^2, \ k \geq 1, \tag{28}$$

with $\chi = 0.9$ and safeguard

$$\eta_k = \max\{\eta_k, \chi \bar{\eta}_{k-1}^2\},$$

if $\chi \bar{\eta}_{k-1}^2 > 0.1$. Then, the additional safeguard $\eta_k = \min\{\eta_k, \eta_{max}\}$ is imposed. Concerning the backtracking strategy, we set $\texttt{NBT}_{\texttt{max}} = 20$ and $\lambda = 10^{-4}$ in (28).

A successful termination of the Newton-Krylov solver is declared when

$$\|F_k\|_2 \leq 10^{-8}, \tag{29}$$

within a maximum of 100 nonlinear iterations. A failure is declared otherwise. A failure is also declared when two consecutive backtracking failures occur because, despite the preconditioner refresh, it was not possible to satisfy the decrease condition (27).

In order to have a fair comparison in terms of CPU time among the codes, we implemented their most time-consuming part as Fortran 90 mex-files with Matlab interface. In particular, we implemented a Fortran mex-file version of Algorithm 3.1 which resulted very efficient since we could take advantage of the Compressed Column Format used by Matlab to store sparse matrices. Moreover, we employed a mex-file implementation of Step 2 of [13, Algorithm 4.1] which carries out the matrix-free application of the preconditioners (8). The matrix-free implementation of the preconditioner (7) has been developed analogously.

All the tests were performed on an Intel Xeon (TM) 3.4 GHz, 1GB RAM using Matlab 7.6 and machine precision $\epsilon_m \approx 2 \times 10^{-16}$.

5.2 Experiments

We considered four problems widely used in literature (see e.g. [2]): the Nonlinear Convection-Diffusion problem (`NCD`), the Flow in a Porous Medium problem (`FPM`), the CounterCurrent Reactor problem (`CCR`) and the 2D Driven Cavity problem (`2DC`). Three of them, namely problems `NCD`, `FPM` and `2DC`, arise from the discretization of PDE problems. In all four problems, the

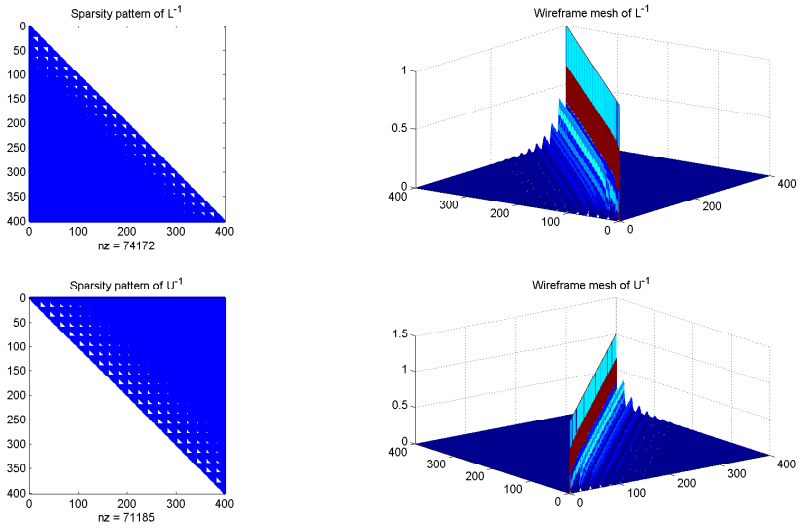

Figure 1: The nonlinear convection-diffusion problem: sparsity pattern (on the left) and wireframe mesh (on the right) of the inverses of the L and U factors obtained from the ILU factorization of the Jacobian at the starting point ($n = 400$).

dimension n of the nonlinear systems can be varied and we considered the values: 4900, 8100, 10000, 15625, 22500.

Problem `NCD` has been solved setting the Reynolds number equal to 250, 500, and 1000, while problem `2DC` has been solved with the smaller Reynolds numbers 50, 100, and 150.

The seed preconditioner in problem `CCR` was computed with the drop tolerance 10^{-1} while in problems `NCD` and `FPM` the value 10^{-2} was employed. Problem `2DC` requires a tighter drop tolerance to build an efficient preconditioner and the tolerance used is 10^{-3}. For all the other parameters, we adopted the choices used in [2].

Figures 1 and 2 show for problems `NCD` and `2DC` the sparsity pattern (on the left) and the wireframe mesh (on the right) of the inverses of the L and U factors obtained from the ILU factorization of the Jacobian at the starting point. We remark that these inverses appear almost dense but the fast decay of the elements away from the main diagonal is clear and motivates the use of our BU_ILU strategy to solve these problems. Plots similar to those of Figure 1 have been observed for problems `CCR` and `FPM` but they are not reported here.

We give statistic of the runs in tables where the column's headers have the following meaning:

- `L_IT`: Total number of BiCGSTAB iterations.
- `NL_IT`: Number of nonlinear iterations.
- `Time`: Overall execution time (in seconds) of the Newton-Krylov algorithm.
- `RN`: Number of times the preconditioner was refreshed.
- `Re`: Reynolds number.

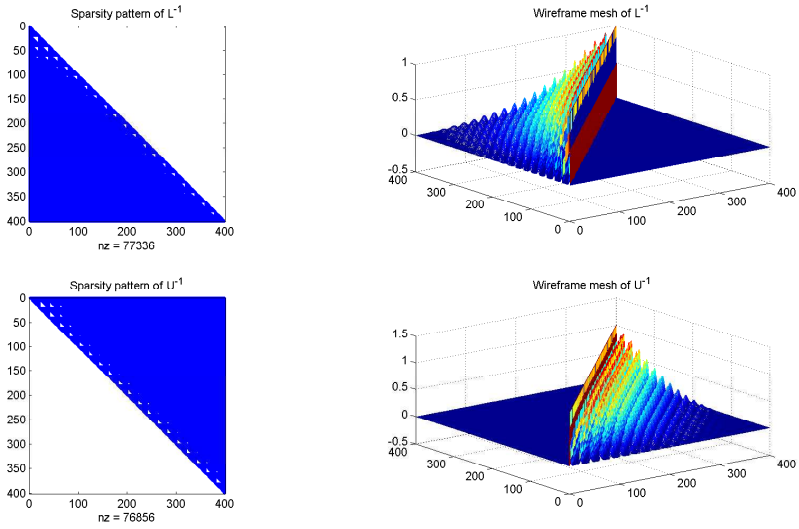

Figure 2: The 2D driven cavity problem: sparsity pattern (on the left) and wireframe mesh (on the right) of the inverses of the L and U factors obtained from the ILU factorization of the Jacobian at the starting point ($n = 400, Re = 50$).

First, we focus on problems `NCD`, `FPM` and `CCR` and compare the performance of DU_ILU preconditioner with that of BU_ILU with $\delta = 0$, $\beta = 0$ and $\gamma = 20$. Making some comments on the choice of the parameters for BU_ILU preconditioner, we first note that only the diagonal of the current preconditioner J_k has to be computed. Second, the application of BU_ILU does not require the factorization of the middle factor in (22) because $\beta = 0$. Finally, we approximate L^{-1} and U^{-1} retaining 20 of their lower and upper diagonals, respectively; experiments with $\gamma = 50$ were carried out but meaningful differences in term of BiCGSTAB iterations and number of preconditioner's refreshes were not registered.

Tables 1-3 display the results obtained. Remarkably, both updating procedures work well retaining only the diagonal of the current Jacobian and require a small number of preconditioner's refresh, except for problem `FPM`, with BU_ILU preconditioner and the larger grids, where 3 or 4 refreshes are required. Focusing on the number of linear iterations and preconditioner's refreshes it is observed that DU_ILU outperforms BU_ILU in the solution of problems `FPM` and `CCR`. In terms of computational time, Newton-Krylov method with DU_ILU is the most effective in 21 runs out of 25 and in several cases the savings in time are large. In the four runs where BU_ILU is faster, DU_ILU requires one extra preconditioner's refresh.

The need to freeze the preconditioner because condition (25) is met never occurred in problem `NCD` while it occurred once in BU_ILU with problem `CCR` and n= 15625. On the contrary, freezing the preconditioner was crucial to get convergence of the nonlinear procedure in problem `FPM`; this fact confirms the experience in [2].

Summarizing the computational testing reported above, as long as information on the diagonal of the current Jacobian is sufficient for updating, DU_ILU strategy seems more effective than BU_ILU. Such good behaviour and the relevant feature of being independent of the choice of parameters make the DU_ILU strategy promising for practical applications.

Problem NCD									
		DU_ILU				BU_ILU			
Re	n	L_IT	NL_IT	Time	RN	L_IT	NL_IT	Time	RN
250	4900	549	13	22.76	1	635	13	24.89	1
	8100	621	13	53.05	1	561	12	53.48	1
	10000	601	13	75.24	1	569	12	77.55	1
	15625	1108	11	178.11	1	1311	11	136.33	0
	22500	1270	11	282.24	0	916	11	374.71	1
500	4900	671	14	27.39	1	819	15	27.62	1
	8100	768	16	58.25	1	895	16	62.63	1
	10000	976	16	108.92	2	621	15	82.45	1
	15625	755	15	185.1	1	664	15	187.43	1
	22500	668	14	387.72	1	632	14	391.54	1
1000	4900	1204	16	34.54	2	1079	15	32.69	1
	8100	799	16	71.17	2	1217	16	81.96	2
	10000	1010	16	106.99	2	865	16	107.1	2
	15625	675	17	238.61	2	1000	17	251.83	2
	22500	1281	15	423.13	1	875	15	407.51	1

Table 1: The nonlinear convection-diffusion problem: results with with DU_ILU preconditioner and BU_ILU preconditioner, $\gamma = 20$, $\delta = \beta = 0$, varying Re and n.

Problem FPM								
	DU_ILU				BU_ILU			
n	L_IT	NL_IT	Time	RN	L_IT	NL_IT	Time	RN
4900	154	9	26.84	0	481	10	47.36	1
8100	230	11	89.5	0	581	12	140.93	1
10000	660	12	237.52	1	1039	14	358.05	3
15625	819	12	815.21	2	1639	13	1042.81	3
22500	1018	13	1578.8	1	1849	15	2572.29	4

Table 2: The flow in a porous medium problem: results with DU_ILU precondition with and BU_ILU preconditioner, $\gamma = 20$, $\delta = \beta = 0$, varying n.

Problem CCR								
	DU_ILU				BU_ILU			
n	L_IT	NL_IT	Time	RN	L_IT	NL_IT	Time	RN
4900	718	12	24.79	1	800	11	33.57	2
8100	661	11	55.55	1	729	10	73.64	2
10000	690	11	81.67	1	946	10	111.74	2
15625	352	12	185.03	1	1240	13	274.57	2
22500	504	11	391.62	1	1249	12	565.03	2

Table 3: The countercurrent reactor problem: results with DU_ILU preconditioner and with BU_ILU preconditioner, $\gamma = 20$, $\delta = \beta = 0$, varying n.

Problem 2DC with BU_ILU									
	Re=50			Re=100			Re=150		
n	L_IT	NL_IT	Time	L_IT	NL_IT	Time	L_IT	NL_IT	Time
4900	147	5	26.07	251	6	29.69	650	7	39.49
8100	398	5	66.28	744	6	81.27	993	6	88.93
10000	676	5	104.56	1000	6	123.36	1218	6	131.68
15625	821	5	238.94	1309	6	284.09	1431	7	300.47
22500	971	5	484.18	1436	6	558.25	2120	7	647.75

Table 4: The 2D driven cavity problem: results with BU_ILU preconditioner, $\gamma = 50$, $\delta = \beta = 1$, varying Re and n.

Although DU_ILU strategy has proved to be successful, BU_ILU procedure has a wider application. In fact, there are cases where DU_ILU cannot be applied. One example is problem 2DC where the diagonal of the Jacobian is constant and DU_ILU preconditioner coincides with the seed preconditioner. Analogously, the BU_ILU preconditioner with $\delta = 0$ reduces to the seed preconditioner.

We now present results for problem 2DC obtained with BU_ILU strategy and $\delta = 1$, that is the tridiagonal part of the current Jacobian is used for the updating. The sequences of linear systems arising in the solution of problem 2DC are extremely difficult to be solved. Sparse preconditioners caused BiCGSTAB to stagnate at the first nonlinear iteration and for this reason, as already mentioned, the drop tolerance used to compute the seed preconditioner was 10^{-3}. Further, in [12] it is pointed out that the refreshed and recomputed seed ILU preconditioners for the sequences arising in the solution of this problem, deteriorate in the progress of nonlinear iterations. Since recomputing the preconditioner should be avoided, we modified the Newton-Krylov solver, in that we inhibited the refresh of the preconditioner when the Krylov method fails and we proceeded with the last computed iterate produced by BiCGSTAB. Finally, we extracted a larger number of diagonal from L^{-1} and U^{-1} than for the previous problems and used $\gamma = 50$. This is also motivated by the slower decay of the off-diagonal elements of L^{-1} and U^{-1} than in previous problems, see Figure 2.

We performed experiments testing the behaviour of BU_ILU with $\beta = 0$ and $\beta = 1$. In most of the runs, the latter value of β resulted to be the most convenient in terms of number of linear iterations. Moreover, although a tridiagonal system has to be solved in order to apply the preconditioner, the Newton-Krylov method with BU_ILU and $\beta = 1$ is fastest in 11 runs out of 15. Therefore, in Table 4 we show results obtained with this value of β.

From Table 4 it is quite evident that the sequences become more difficult to be solved by preconditioned BiCGSTAB as the Reynold number and the grid dimension increase. The Newton-Krylov solver is successfully in all the runs but, when the two largest grid values are employed, in some nonlinear iterations BiCGSTAB fails in satisfying the stopping criterion (26) within the maximum allowed 400 iterations.

We conclude this section showing the numerical behaviour of the updating techniques proposed in [12] and described in Section 2. Two different matrix-free implementations of updatings (7) and (8) have been proposed in [12, 13].097890 We limit ourselves to consider the approach described in [13, Algorithm 4.1].

Algorithm 4.1 in [13] exploits separability of the components of function F and applies the updated factorized preconditioner via function evaluations. Following this approach the upper/lower tridiagonal part of the Jacobian is not explicitly needed and only the diagonal of the current preconditioner needs to be evaluated and stored. The cost of one application of the

preconditioner is given by n scalar F–evaluations corresponding to all the scalar component of F. Hence, if the cost of the n scalar F–evaluations is roughly the cost of one full F evaluation, each iteration of BiCGSTAB preconditioned requires four F-evaluations: two for applying the preconditioner and two for approximating the action of the Jacobian on a vector (BiCGSTAB requires two matrix-vector products and two applications of the preconditioner in every iteration). Thus we can draw the conclusion that, in terms of F-evaluations the cost of one iteration of BiCGSTAB preconditioned by (7) or (8) update is twice the cost of BiCGSTAB preconditioned by our updating procedures.

We performed runs using both updates (7) and (8). The performance of the Newton-Krylov algorithm heavily depends on the chosen update; here for each problem we report on the results obtained by the best performing one. The preconditioner (8) outperformed the preconditioner (7) in the solution of problems `NCD`, `FPM` and `2DC` whereas for problem `CCR`, (7) was the best.

We do not report the elapsed CPU time because, despite the employment of Fortran 90 mex-files to implement the algorithm, we observed unexpected large values. In fact, in our experience the evaluation of n scalar functions implemented as Fortran mex-files and run in Matlab environment turned out to be more time consuming than the evaluation of one vector value F. As a consequence, we use the number of linear and nonlinear iterations and the number of F-evaluations as a measure of comparison between the preconditioners (7) and (8) and our proposals DU_ILU and BU_ILU.

Computational experience indicate that the Newton-Krylov method combined with the preconditioners (7) and (8) is less robust than with our strategies in the solution of problem `2DC`. Specifically the Newton-Krylov algorithm fails in all runs with problem `2DC` except four cases: $n = 4900$, 8100, 10000 and $Re = 50$, $n = 8100$ and $Re = 100$. The failures of the Newton-Krylov solver depend on the failures of preconditioned BiCGSTAB.

Statistics of the runs obtained solving problems `NCD`, `FPM` and `CCR` are collected in Tables 5 and 6. We did not report statistics on problem `2DC` as the number of runs successfully solved is small.

Comparing Table 5 with Table 1, we note that in the majority of the runs the updating (8) requires a lower number of preconditioner's refresh and the number of BiCGSTAB iterations is in most of the runs lower than half of the number of iterations executed by BiCGSTAB preconditioned by DU_ILU and BU_ILU . Therefore, although BiCGSTAB preconditioned by (8) requires about two additional F-evaluations per iteration, we can draw the conclusion that this updating strategy is less costly than ours.

Results for problems `FPM` and `CCR` seems to indicate an advantage of our procedures over the updates (7) and (8) with respect to the number of F-evaluations and preconditioner's refresh. Specifically, Table 6 highlights that in the solution of `FPM` problem, updating (8) requires a large number of preconditioner's refresh. Most of the times, these refreshes are performed because BICGSTAB returns a failure as ill-conditioning of the preconditioner has been detected. Finally, comparing the the results in Tables 3 and 6 for problem `CCR`, we see that the updating (7) performs better than BU_ILU in terms of F evaluations but is less convenient than DU_ILU strategy in terms of F-evaluations and number of preconditioner's refresh.

References

[1] S. Bellavia, S. Berrone, *Globalization strategies for Newton-Krylov methods for stabilized FEM discretization of Navier-Stokes equations*, Journal of Computational Physics, 226 (2007), pp. 2317-2340.

	Problem NCD with Update (8)								
	Re = 250			Re = 500			Re = 1000		
n	L_IT	NL_IT	RN	L_IT	NL_IT	RN	L_IT	NL_IT	RN
4900	206	12	0	643	15	1	636	15	1
8100	213	12	0	640	16	1	666	14	1
10000	212	12	0	351	14	0	712	15	1
15625	225	11	0	319	14	0	871	15	1
22500	264	11	0	327	13	0	496	14	0

Table 5: The nonlinear convection-diffusion problem: results with (8) preconditioner varying Re and n.

	Problem FPM with Update (8)			Problem CCR with Update (7)		
n	L_IT	NL_IT	RN	L_IT	NL_IT	RN
4900	154	13	2	443	12	2
8100	1172	15	4	440	12	2
10000	1094	17	5	440	12	2
15625	754	15	4	440	12	2
22500	1076	20	6	438	12	2

Table 6: The flow in a porous medium problem: results with (8) preconditioner on on the left. The countercurrent reactor problem: results with (7) preconditioner on the right.

[2] S. Bellavia, D. Bertaccini, B. Morini, *Nonsymmetric preconditioner updates in Newton-Krylov methods for nonlinear systems*, SIAM J. Sci. Comput., 33 (2011), pp. 2595-2619.

[3] S. Bellavia, V. De Simone, D. di Serafino, and B. Morini, *Efficient preconditioner updates for shifted linear systems*, SIAM J. Sci. Comput., 33 (2011), pp. 1785–1809.

[4] S. Bellavia, B. Morini, *A globally convergent Newton-GMRES subspace method for systems of nonlinear equations*, SIAM J. Sci. Comput., 23 (2001), pp. 940–960.

[5] M. Benzi, G.H. Golub, *Bounds of the entries of matrix functions with applications to preconditioning*, BIT, 39 (1999), pp. 417-438.

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

[7] M. Benzi, M. Tůma, *A Comparative Study of Sparse Approximate Inverse Preconditioners*, Appl. Numer. Math., 30 (1999), 305–340.

[8] L. Bergamaschi, R. Bru, A. Martinez, M. Putti, *Quasi-Newton preconditioners for the inexact Newton method*, Electronic Trans. Num. Anal., 23 (2006) pp. 76–87.

[9] P. Birken, J. D. Tebbens, A. Meister and M. Tuma, *Preconditioner updates applied to CFD model problems*, Appl. Numer. Math., 58 (2008), pp. 1628–1641.

[10] C. Calgaro, J.P. Chehab, Y.Saad, *Incremental incomplete ILU factorizations with applications*, Numer. Linear Algebra Appl., 17 (2010), pp. 811-837.

[11] S. Demko, W.F. Moss, P.W. Smith, *Decay rates for inverses of band matrices*, Math. Comput., 43 (1984), pp. 491-499.

[12] J. Duintjer Tebbens, M. Tůma, *Efficient Preconditioning of Sequences of Nonsymmetric Linear Systems*, SIAM J. Sci. Comput., 29 (2007), pp. 1918–1941.

[13] J. Duintjer Tebbens, M. Tůma, *Preconditioner Updates for Solving Sequences of Linear Systems in Matrix-free Environment*, Numer. Linear Algebra Appl., 17 (2010), pp. 997-1019.

[14] S.C. Eisenstat, H.F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

[15] C.A. Floudas et al., *Handbook of test problems in local and global optimization*, Kluwer Academic Publishers, Nonconvex Optimization and its Applications, 33, 1999.

[16] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, *Optimization with PDE Constraints*, Springer-Verlag, New York Inc. 2009.

[17] D.A. Knoll, D.E. Keyes, *Jacobian-free Newton-Krylov methods, a survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357-397.

[18] F. Lemeire, *Bounds for condition numbers of triangular and trapezoid matrices*, BIT, 15 (1975), pp. 58–64.

[19] G. Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix. Anal. Appl., 13 (1992), pp. 707-728.

[20] R. Nabben, *Decay rates of the inverse of nonsymmetric tridiagonal and band matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 820-837.

[21] M.L. Parks, E. de Sturler, G. Mackey, D.D. Johnson, S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 1651-1674.

[22] R.P. Pawlowski, J.N. Shadid, J.P. Simonis, H.F. Walker, *Globalization techniques for Newton-Krylov methods and applications to the fully-coupled solution of the Navier-Stokes equations*, SIAM Review, 48 (2006), pp. 700-721.

[23] M. Pernice, H.F. Walker, *NITSOL: a new iterative solver for nonlinear systems*, SIAM Journal Sci Comput., 19 (1998), pp. 302-318.

[24] P.S. Vassilevski, *On some ways of approximating inverses of banded matrices in connection with deriving preconditioners based on incomplete block factorizations*, Computing, 43 (1990), pp. 277-296.