NONSYMMETRIC PRECONDITIONER UPDATES IN NEWTON-KRYLOV METHODS FOR NONLINEAR SYSTEMS *

STEFANIA BELLAVIA †, DANIELE BERTACCINI ‡, AND BENEDETTA MORINI §

Abstract. Newton-Krylov methods, combination of Newton-like methods and Krylov subspace methods for solving the Newton equations, often need adequate preconditioning in order to be successful. Approximations of the Jacobian matrices are required to form preconditioners and this step is very often the dominant cost of Newton-Krylov methods. Therefore, working with preconditioners destroys in principle the "Jacobian-free" (or matrix-free) setting where the single Jacobian-vector product can be provided without forming and storing the element of the true Jacobian.

In this paper, we propose and analyze a preconditioning technique for sequences of nonsymmetric Jacobian matrices based on the update of an earlier preconditioner. The proposed strategy can be implemented in a matrix-free manner. Numerical experiments on popular test problems confirm the effectiveness of the approach in comparison with the standard ILU-preconditioned Newton-Krylov approaches.

1. Introduction and motivations. The numerical solution of large scale nonlinear systems of algebraic equations

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n,$$

is ubiquitous in models for applied sciences. Models requiring the solutions of such problems are based on partial differential equations, on systems of coupled differential equations, on hybrid differential-algebraic problems, on equilibrium problems and others. Newton and, more generally, Newton-like algorithms, are robust and deterministic approaches which are often used as a standard tool in small and medium scale problems. On the other hand, Newton methods are generally expensive when applied to large scale problems, specially compared to splitting-based methods, [28]. The most critical part of Newton-like algorithms is forming and storing an approximation to the Jacobian matrix of the underlying problem. Several variants of Newton-like algorithms targeted to save memory and computational resources have been considered through the years, [1, 3, 17, 20, 25, 26, 35, 36].

Here we base our new proposal on the so called *Jacobian-free Newton-Krylov* solvers i.e. Newton-like algorithms where the Newton correction linear equations are solved by a Krylov subspace method without requiring explicitly the Jacobian matrix, see e.g. [16, 18, 28, 36]. In particular, only the action of the Jacobian matrix on a given vector is required and it is approximated performing one function F evaluation. Unfortunately, these algorithms sometimes suffer from a slow convergence of the (inner) Krylov linear iterations and building good preconditioners for the Jacobian matrices in a matrix-free environment is not an easy task. Specifically, effective algebraic preconditioners require accessing the entries of the Jacobian matrix but often their computation can be the most expensive part of the algorithm. As a result,

*Revised version. Work supported in part by INDAM-GNCS grant "Progetti 2010", "Analisi e risoluzione iterativa di sistemi lineari di grandi dimensioni in problemi di ottimizzazione", and by grant PRIN 20083KLJEZ, MIUR, Roma, Italia.

†Dipartimento di Energetica "S. Stecco", Università di Firenze, via C. Lombroso 6/17, 50134 Firenze, Italia, stefania.bellavia@unifi.it

‡Dipartimento di Matematica e Centro Interdipartimentale per la Biostatistica e la Bioinformatica, viale della Ricerca Scientifica, 00133 Roma, bertaccini@mat.uniroma2.it

§Dipartimento di Energetica "S. Stecco", Università di Firenze, via C. Lombroso 6/17, 50134 Firenze, Italia, benedetta.morini@unifi.it

computing a preconditioner each time the Jacobian matrix changes (*recomputed* preconditioner) can be a very expensive task for large scale problems while reusing the same preconditioner on several Newton iterations (*frozen* preconditioner) may yield to a failure of the inexact Newton process. Therefore, something in between the former and the latter approach can be beneficial and the construction of a reasonable candidate preconditioner should require the approximation of as few entries of the Jacobian as possible; this purpouse can be accomplished by preconditioning strategies for sequences of linear systems.

The issue of preconditioning sequences of linear systems has been investigated in several papers. We recall approaches based on Quasi-Newton updates [8, 31], recycling Krylov subspace techniques [34], partial matrix estimation [19], approximate update of factorized preconditioners [5, 9, 10, 13, 22, 23, 30]. Focusing on the last mentioned approach, the cited papers differ for the sequence of matrices considered and the techniques implemented but share the following key features. If a factorized preconditioner for a (*reference*) matrix of the sequence is known, then a factorized updated approximation for a subsequent matrix can be algebraically derived; in theory, such updated preconditioner requires the difference between the current and the reference matrices. In most cases, forming the ideal update is not of practical interest because of the computational cost and the fact that one term of the factorization is dense. To cope with these difficulties the ideal update is approximated, e.g. by using structured approximations to the matrices involved.

The approach given in [13, 22, 23] deserves a deeper analysis. To our knowledge these are the only papers dealing with approximate preconditioner updates for sequences of nonsymmetric linear systems arising in Newton-Krylov solvers. The updating technique given in [13, 22, 23] was inspired by [5, 9] and it is based on an LDU incomplete factorization of the reference matrix. Duintjer Tebbens and Tůma assume that either the factor L or U more or less approximate the identity matrix [22]. Then, the updates proposed neglect one of the two factors L or U and take into account one triangular part of the difference matrix between the current and the reference Jacobian. In fact, it is typically assumed that the matrices have a strong diagonal; this may occur in various applications, e.g. in some upwind/downwind discretization schemes. For different applications, the condition imposed on the L and U factors may be too strong. Interestingly, the proposed strategy can be implemented in a matrix-free manner as discussed in [23].

We propose a new preconditioning technique for the sequence of linear systems arising in Newton-Krylov algorithms. Our procedure generalizes the updates presented in [5, 9] for incomplete factorizations of shifted matrices and in [10] for symmetric matrices applied to an image restoration problem. It consists of two steps. First, we compute a preconditioner for a reference Jacobian in the form of an incomplete inverse factorization. Second, in the progress of the nonlinear iterations we perform cheap updates of such preconditioner. This is achieved by a banded approximation of both the difference between the current Jacobian and the reference Jacobian and the arising updating term. As we will show, the cheapest implementation allows the construction of the preconditioner with a computational effort similar to that of matrix-free settings, i.e. linear in the number of function F evaluations.

The proposed preconditioning technique is quite general. It can update any incomplete factorization of the inverse of the reference Jacobian, e.g. an inverse ILU technique [38] or an AINV preconditioner for nonsymmetric matrices [6]. Further, it can be readily employed in the solution of arbitrary sequences of nonsymmetric linear

systems such as those arising in affine scaling methods for large bound constrained nonlinear systems [4], electronic structure calculation [34], quantum chromodynamics [34].

Our technique differs considerably from the proposals by Duintjer Tebbens and Tůma and may be an efficient alternative approach. In particular, our procedure relies on approximate inverse preconditioners and therefore the application of the preconditioner can be efficiently implemented on parallel computers. Further, the difference matrix and the arising updating term are approximated by extracting the main diagonal and zero or more diagonals on both sides instead of triangular parts as in [13, 22, 23].

In §2 we describe the Inexact Newton-Krylov setting we are working on; in §3 we discuss preconditioning for Newton-Krylov algorithms and in §4 we introduce our matrix-free updating strategy. §4.3 reports an analysis of the properties and behaviour of our preconditioner along with a theoretical comparison of our preconditioner with the frozen and recomputed preconditioners. Finally, in §5 we report some numerical results and comparisons with a Matlab implementation of the paradigm implemented in the widely used *NITSOL* software package [36].

Throughout the paper, $\|\cdot\|$ denotes an arbitrary vector or matrix norm. The specific use of 1-norm and Euclidean norm will be indicated as $\|\cdot\|_1$, $\|\cdot\|_2$.

2. Newton-Krylov methods. Consider the system of nonlinear equations

$$F(x) = 0, \tag{2.1}$$

where $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ is continuously differentiable. Let $F = (F_1, F_2, \ldots, F_n)^T$, $F_i : \mathbb{R}^n \mapsto \mathbb{R}$, $i = 1, \ldots, n$, and J denote the Jacobian matrix.

Given the current iteration x_k, Newton method computes a step s as the solution to the Newton equation

$$J_k s = -F_k, \tag{2.2}$$

where $F_k = F(x_k)$, $J_k = J(x_k)$, and then sets $x_{k+1} = x_k + s$. On the other hand, Newton-Krylov methods solve the linear system (2.2) approximately by using a Krylov method and find a step s_k such that

$$\|J_k s_k + F_k\|_2 \leq \eta_k \|F_k\|_2, \tag{2.3}$$

with $\eta_k \in [0, 1)$. Clearly, these methods belong to the class of the Inexact Newton methods [20] and, among the variety of contributions in this framework we cite the papers [1, 3, 4, 17, 20, 25, 26, 35, 36].

The so-called forcing term η_k controls the level of accuracy in the solution of the Newton equation. Solving (2.2) approximately is beneficial when the dimension n of the problem is large and when x_k is far from a solution. In practice, the role of η_k is to avoid a pointless accuracy in the computation of the steps while retaining fast local convergence. Thus, it is convenient to require a modest accuracy far away from a solution and higher accuracy when approaching a solution, [26].

The convergence of Newton and Inexact Newton methods is local, i.e. convergence is guaranteed if the initial iterate x_0 is sufficiently near a solution. Globalization techniques improve the likelihood of convergence from arbitrary starting point, see e.g. [2, 25, 29], and most of them fall into two classes: linesearch methods and trust-region methods. Focusing on linesearch strategies, they have been widely used in the

context of Newton-Krylov methods and impose a sufficient decrease in the value of $\|F\|_2$. Such strategies follow from the observation that any step p satisfying

$$\|J_k p + F_k\|_2 \leq \eta_k \|F_k\|_2, \tag{2.4}$$

is a descent direction for $\|F\|_2$ at x_k, [25, Lemma 7.1]. A well-known strategy proposed in [25] implements a test for accepting the inexact Newton step s_k; if s_k fails to satisfy the test then backtracking is performed. The resulting Linesearch Newton-Krylov Algorithm is stated below.

Algorithm 2.1: LINESEARCH NEWTON-KRYLOV.

Given x_0, $\eta_{max} \in [0, 1)$, $\alpha \in (0, 1)$, $0 < \sigma_0 < \sigma_1 < 1$.
For $k = 0, 1 \ldots$ until convergence do:
  Choose $\eta_k \in [0, \eta_{max}]$.
  Compute a step s_k by applying the Krylov method to (2.2) and terminate when (2.3) holds.
  Set $\bar{s}_k = s_k$ and $\bar{\eta}_k = \eta_k$.
  While $\|F(x_k + \bar{s}_k)\|_2 \geq (1 - \alpha(1 - \bar{\eta}_k)) \|F_k\|_2$
    Choose $\sigma \in [\sigma_0, \sigma_1]$.
    Update $\bar{s}_k = \sigma \bar{s}_k$ and $\bar{\eta}_k = 1 - \sigma(1 - \bar{\eta}_k)$.
  Set $x_{k+1} = x_k + \bar{s}_k$.

At each iteration of Algorithm 2.1, an initial inexact Newton step s_k at level η_k is tried. If it does not provide a sufficient decrease in the value of $\|F\|_2$, then vectors $\bar{s}_k$ are tried until a satisfactory step is found. It is important to note that $\bar{s}_k$ is still an inexact step as it satisfies (2.3) with forcing term $\bar{\eta}_k$. The above procedure is a general framework for linesearch method; Inexact Newton methods that incorporate a linesearch procedure based on the first Goldstein-Armijo condition (the "alpha-condition") are a special case of the algorithm considered, [25].

If J_k is invertible and $\|F_k\|_2 \neq 0$, the Linesearch Newton-Krylov Algorithm does not break down. In particular, a step s_k can be found; then, the analysis conducted in [25] shows that the while-loop terminates in a finite number of steps. The principal convergence result for the Algorithm 2.1 is the following.

THEOREM 2.1. *Assume that the Linesearch Newton-Krylov Algorithm does not break down. If x^* is a limit point of $\{x_k\}$ such that $J(x^*)$ is nonsingular, then $F(x^*) = 0$ and $x_k \to x^*$. Furthermore, the initial s_k and η_k are accepted without modification in the while-loop for all sufficiently large k.*

If J is Lipschitz continuous at x^*, then if $\eta_k \to 0$, the convergence is q-superlinear, and if $\eta_k \leq \Gamma \|F_k\|_2$ for some $\Gamma > 0$, the convergence is of q-order 2.

Proof. See [25, 26]. □

We explicitly note that under the above standing conditions, asymptotic convergence to the solution x^* is determined by the ultimate behaviour of the sequence $\{\eta_k\}$.

Another important feature of the Newton-Krylov methods is that they require no matrix storage if a transpose-free Krylov method is used. In fact, a transpose-free Krylov method applied to (2.2) requires only the action of the Jacobian J on a vector v and Jacobian-vector products can be approximated using finite-difference formulas.

Formulas of orders one, two, and four are available, see e.g. [36, §2.2]. A first-order approximation is given by

$$F'(x)v \approx \frac{F(x+\epsilon v) - F(x)}{\epsilon}, \tag{2.5}$$

where the scalar ϵ is chosen on standard arguments to have a trade-off between floating point errors and accuracy requirements, [16, 36]. The resulting approach is referred to as "matrix-free" since it does not require forming or storing the Jacobian. Remarkably, this approach still maintains the fast local convergence properties stated in Theorem 2.1, [16].

3. Preconditioning Newton-Krylov methods. Good performance of the Linesearch Newton-Krylov Algorithm is strongly affected by effective preconditioning of the Newton equations. Usually, a maximum number of linear iterations is allowed in the solution of (2.2). If the Krylov method converges slowly, the initial step s_k satisfying (2.3) may not be found within the prescribed iterations. In this case, either a failure is declared or the algorithm proceeds with the last step, say s_*, computed by the linear solver. In practice, let

$$s_k = s_*, \qquad \eta_k = \frac{\|J_k s_* + F_k\|_2}{\|F_k\|_2}.$$

Provided that η_k is less than one, then s_k is a descent direction for $\|F\|_2$ and backtracking can be performed along it, [3]. However, this step may be only a weak descent direction and too many backtracking reductions may be necessary; as a consequence a practical backtracking routine may declare failure before an acceptable step is found.

Unfortunately, algebraic preconditioning requires forming the Jacobian matrices or approximating them and the resulting Newton-Krylov procedure is no longer matrix-free. On the other hand, the preconditioning strategy is considered to be matrix-free, or nearly matrix-free, if it needs matrices that are reduced in complexity with respect to the full Jacobian and still takes advantage of matrix-vector product approximations by finite differences, see e.g., [28, pp. 362, 374].

The excellent survey [28] includes a review on preconditioning Newton-Krylov methods with ILU factorizations, multilevel Schwarz-type methods, multigrid and structure-based approaches. These strategies require a high computational effort to access to the Jacobians with the exception of Krylov iterative solvers used as preconditioners and few cases using the special structure of the problem.

Updated preconditioners for sequences of nonsymmetric linear systems were motivated by the need to avoid the expensive computation of a new preconditioner. Proper implementations of the updating strategies allow to improve upon the frozen preconditioners and obtain numerical performances which compare favorably with those of a recomputed preconditioner.

4. Matrix free updating preconditioners. To reduce the computational cost of preconditioning a sequence of Newton equations, the quality of a preconditioner from an earlier Newton equation can be enhanced through updates containing information on the current Jacobian. For sake of efficiency, the updating strategy should require a small overhead, be simple to implement and interface with existing pieces of codes.

In this section, we present a procedure for updating the factorization of a reference preconditioner over several subsequent iterations. For specific options, this turns out

to be matrix-free and the computational effort is similar to that of matrix-free settings, requiring only the evaluation of the n components of function F.

The idea is an evolution of what has been proposed in [5, 9] for special sequences of symmetric matrices differing from one another by a diagonal matrix. In order to build up an approximation for the Jacobian matrix J_k, consider an invertible reference Jacobian J_{seed} and a factorization of J_{seed}^{-1}

$$J_{seed}^{-1} = W\, D^{-1} Z^T, \tag{4.1}$$

where D is a diagonal matrix, W and Z are upper triangular matrices, possibly permuted, with ones on the main diagonal.

A factorization for the inverse of the current invertible Jacobian matrix J_k can be obtained as follows. Letting

$$\Delta_k = J_k - J_{seed}, \quad E_k = Z^T\, \Delta_k\, W, \tag{4.2}$$

we get

$$\begin{aligned} J_k^{-1} &= (J_{seed} + \Delta_k)^{-1} \\ &= (Z^{-T} D W^{-1} + Z^{-T} E_k W^{-1})^{-1} \\ &= W\ (D + E_k)^{-1}\ Z^T. \end{aligned} \tag{4.3}$$

Often, the factors Z and W are rather dense and their computation and storage are too expensive to be feasible. On the other hand, it is often possible to compute sparse approximations $\tilde{Z}$ and $\tilde{W}$ for Z and W which can provide a good factorized sparse preconditioner for the inverse of J_{seed}. This gives rise to

$$P_{seed} = \tilde{W}\, \tilde{D}^{-1} \tilde{Z}^T, \tag{4.4}$$

where $\tilde{D}$ is a nonsingular diagonal approximation to D.

Consequently, we can construct a candidate factorized sparse inverse preconditioner P_k for J_k^{-1} using P_{seed} and a sparsification of matrices E_k and Δ_k. In fact, we set

$$P_k = \tilde{W}\ \left(\tilde{D} + \tilde{E}_k\right)^{-1}\ \tilde{Z}^T, \tag{4.5}$$

where

$$\tilde{E}_k = g(\tilde{Z}^T\, \tilde{\Delta}_k\, \tilde{W}), \quad \tilde{\Delta}_k = f(\Delta_k). \tag{4.6}$$

Here f and g are functions extracting selected components of their matrix arguments.

In the following sections 4.1, 4.2 we describe a viable implementation of the updating process. Specifically, we discuss how to generate a preconditioner P_{seed} for J_{seed} and choose the functions g and f in order to avoid computing the entire Jacobian matrix J_k and facing with full matrices. We also discuss how to avoid breakdowns in the updates when the resulting preconditioner is nearly singular. Then in section 4.3 we analyze the quality of the updated preconditioners.

4.1. A factorized sparse approximate inverse for J_{seed}. In order to get a usable sequence of approximations for the Jacobian matrices J_k, it is mandatory to provide a good starting preconditioner. Let us suppose that an approximate factorization for a possibly permuted version of J_{seed}^{-1} exists; to simplify the treatise, we omit the presence of permutations. Factors for the approximate inverse preconditioner (4.4) can be computed from various decompositions. Here we describe two viable approaches, one based on the inverse ILU technique proposed in [38] and the other based on AINV preconditioners, [6].

The inverse ILU technique. The *reference* preconditioner P_{seed} can be provided using an inverse ILU technique. This amounts to compute first a *threshold* ILU factorization, see e.g., [37, Chapter 10]

$$J_{seed} \approx \tilde{L}\,\tilde{D}\,\tilde{U}^T, \tag{4.7}$$

where $\tilde{L}$ and $\tilde{U}$ are unit lower triangular and $\tilde{D}$ is a diagonal matrix and then to approximate the factorization of the inverse of the Jacobian matrix

$$J_{seed}^{-1} = W D^{-1} Z^T.$$

Letting $\tilde{Z}$ and $\tilde{W}$ be sparse matrices such that

$$\tilde{Z} \approx \hat{Z} = \tilde{L}^{-T}, \quad \tilde{W} \approx \hat{W} = \tilde{U}^{-T}, \tag{4.8}$$

P_{seed} takes the form

$$P_{seed} = \tilde{W}\tilde{D}^{-1}\tilde{Z}^T. \tag{4.9}$$

Clearly, this procedure is not feasible if the reference ILU factorization breaks down or if it is very ill-conditioned. This can happen in the case of, e.g., strongly indefinite matrices. A way to circumvent this relies often in allowing more fill-in.

The evaluation of $\tilde{L}^{-1}$ and $\tilde{U}^{-1}$ can be performed by $2n$ triangular solves or by using the analytical expression of the rows of the inverse of an unit upper triangular matrix, [38]. Since $\hat{Z}$, $\hat{W}$ can be full even if $\tilde{L}$ and $\tilde{U}$ are sparse, sparsity in the factors $\tilde{Z}$, $\tilde{W}$ can be obtained by dropping entries of $\tilde{L}^{-1}$ and $\tilde{U}^{-1}$ on the basis of a level-of fill scheme or of a drop tolerance [7]. Under some hypothesis on J_{seed}, like, e.g., diagonal dominance or being M-matrix or a suitable reordering with lumping etc., we can ensure a fast decay of the entries of the inverses. Nonetheless, sparsification of $\hat{Z}$ and $\hat{W}$ even with very slow (or no) decay of the entries away from the main diagonal can result in surprisingly good results in practice, [5]. Effective practical schemes for computing $\tilde{Z}$ and $\tilde{W}$ consist in dropping fill as soon as it occurs within a sparse vector update; at this regard we refer to the numerical fill drop strategy of Algorithm 3 in [38]. In order to control the number of fills, we can easily impose a restriction on the allowed number of nonzero entries. This approach is effective on shared memory machines since the only concurrent access to the same memory location is a read type access. The inversion and sparsification part of the underlying algorithm is perfectly scalable because there are no dependencies between the calculation of the columns of the triangular factors, [38].

The AINV method. Another method for computing an approximate factorization of the inverse of the reference Jacobian (4.9) is the sparse AINV preconditioner described in [6]. The algorithm is based on a biconjugate Gram–Schmidt orthogonalization process with respect to the bilinear form associated with J_{seed} and requires the computation of n (sparse) matrix vector products with J_{seed} and J_{seed}^T. Sparsity in the inverse factors is obtained by carrying out the biconjugation process incompletely; a characterization of the fill-in occurring in the AINV procedure is given in [6]. As noted in [7] the preconditioner construction phase offers limited opportunity for parallelization.

The incomplete factorization for the inverse of the reference Jacobian matrix is again expressed as in (4.9).

4.2. The updates, in practice. Let us consider functions f and g in (4.6) of the form

$$\text{band}(A, k_l, k_u), \tag{4.10}$$

a banded approximation of a square matrix A with k_l lower and k_u upper diagonals, respectively. In (4.6) we sparsify the matrix Δ_k and let $\tilde{\Delta}_k$ be given by

$$\tilde{\Delta}_k = \text{band}(\Delta_k, k_l, k_u),$$

for some $0 \leq k_l, k_u \leq n$. In fact, using a few diagonals of J_k to form $\tilde{\Delta}_k$ significantly reduces the storage requirement and the cost of forming the preconditioner. The matrix $\tilde{\Delta}_k$ has

$$nz_{\tilde{\Delta}} = (k_l + k_u + 1)n - \frac{k_l(k_l+1)}{2} - \frac{k_u(k_u+1)}{2},$$

nonzero elements and the direct computation of the entries of $\tilde{\Delta}_k$ can be achieved via function evaluations. In particular, from (2.5) it follows that individual elements $[J_k]_{i,j}$ of the Jacobian can be computed as

$$[J_k]_{i,j} \approx \frac{F_i(x_k + \epsilon e_j) - F_i(x_k)}{\epsilon}, \quad 1 \leq i, j \leq n, \tag{4.11}$$

where e_j denotes the jth unit vector. The cost of this formula is the scalar function F_i evaluation. Therefore, in case a full function F evaluation costs roughly n scalar F_i, $i = 1, \ldots, n$ evaluations, if $k_l = k_u = 0$, the diagonal elements of J_k can be computed as in (4.11) at the cost of approximately one full function evaluation. Analogously, in case $\tilde{\Delta}_k$ is a matrix with small row and column bandwidth, a number of scalar function evaluations equal to the number $nz_{\tilde{\Delta}}$ of nonzero elements are required; this cost corresponds to approximately $k_l + k_u + 1$ full function evaluations. The previous analysis may yield to an underestimation of the costs when several components of F contain a same expression requiring a substantial amount of computation, [24].

The ideal approximation of the "correction" $\tilde{E}_k$ in (4.6) is diagonal, so there are no subsidiary linear systems to be solved in the application of the updated preconditioner (4.5). A true matrix-free setting, in the sense of computational complexity in time and space can be reached by a diagonal approximation for J_k, generating a diagonal $\tilde{\Delta}_k$ and then producing $\tilde{E}_k$ by extracting just the main diagonal from the product $\tilde{E}_k = \tilde{Z}^T \tilde{\Delta}_k \tilde{W}$. This correspond to

$$\tilde{\Delta}_k = \text{band}(\Delta_k, 0, 0), \quad \tilde{E}_k = \text{band}(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}, 0, 0).$$

Choosing $1 \leq k_l, k_u \ll n$, $\tilde{D} + \tilde{E}_k$ has row bandwidth $k_l + k_u + 1$ and a cheap factorization can be computed if it admits an LU decomposition (L has k_l lower bandwidth and U has k_u upper bandwidth). Alternatively, the Givens QR decomposition gives rise to a matrix R with $k_l + k_u$ superdiagonals and requires about $2n(k_l + k_u)^2$ flops, see e.g. [14, p. 221].

More insight into the construction of matrix $\tilde{E}_k$, we need the elements

$$[\tilde{E}_k]_{ij} = [\tilde{Z}]_{i,:} \, [\tilde{\Delta}_k \tilde{W}]_{:,j}, \tag{4.12}$$

where $i = 1, \ldots, n$, $j = \max\{i - k_l, 1\}, \ldots, \min\{i + k_u, n\}$, $[\tilde{Z}]_{i,:}$ denotes the ith row of $\tilde{Z}$ and $[\tilde{\Delta}_k \tilde{W}]_{:,j}$ represents the jth column of $\tilde{\Delta}_k \tilde{W}$. By construction $\tilde{\Delta}_k$ is a band

matrix and $\tilde{Z}$ and $\tilde{W}$ are triangular sparse matrices. A first approach to compute $\tilde{E}_k$ is based on the construction of $\tilde{\Delta}_k$ as specified above. Then, computing $[\tilde{\Delta}_k \tilde{W}]_{:,j}$ amounts to forming a linear combination of a few sparse columns of $\tilde{\Delta}_k$ corresponding to the nonzero entries in the jth column of $\tilde{W}$. Analogously, the scalar products in (4.12) are sparse-sparse mode scalar products. In the limiting case $k_l = k_u = 0$, the diagonal entries of $\tilde{E}_k$ require n scalar vector products; the ith of such products involves sparse vectors with nonzero components in the first i positions.

A second approach where $\tilde{\Delta}_k$ is not explicitly needed is suggested in [23]. Taking into account the structure of $\tilde{\Delta}_k$ and $\tilde{W}$, the vector $[\tilde{\Delta}_k \tilde{W}]_{:,j}$ has at most $j + k_l$ nonzero elements, i.e. $[\tilde{\Delta}_k \tilde{W}]_{i,j}$, $i = 1, \ldots, \min\{j + k_l, n\}$. These elements can be evaluated by finite differences through at most $j + k_l$ scalar function evaluations; in fact, let us consider the entry $[\tilde{\Delta}_k \tilde{W}]_{i,j}$ and let us assume for sake of simplicity that $i - k_l \geq 1$ and $i - k_u \leq n$. Then,

$$[\tilde{\Delta}_k \tilde{W}]_{i,j} = \sum_{l=i-k_l}^{i+k_u} ([J_k]_{i,l} - [J_{seed}]_{i,l})[\tilde{W}]_{l,j}$$

and

$$\sum_{l=i-k_l}^{i+k_u} [J_k]_{i,l}[\tilde{W}]_{l,j} = (J_k \bar{w})_i$$

where $\bar{w} \in \mathbb{R}^n$ is given by $\bar{w} = (0, \ldots, 0, [\tilde{W}]_{i-k_l,j}, \ldots, [\tilde{W}]_{i+k_u,j}, 0, \ldots, 0)^T$. Thus, the term $(J_k \bar{w})_i$ can be approximated by finite differences and requires the evaluation of the ith component of the function F at $x_k + \epsilon \bar{w}$, for some scalar ϵ. Moreover, it is important to note from (4.12) that only the components of $[\tilde{\Delta}_k \tilde{W}]_{:,j}$ corresponding to nonzero entries of $[\tilde{Z}]_{i,:}$, $i = \max\{j - k_l, 1\}, \ldots, \min\{j + k_u, n\}$, are needed.

A breakdown of the updated approximation (4.5) occurs whenever $\tilde{D} + \tilde{E}_k$ is singular. Using $k_l = k_u = 0$, we can easily monitor a possible breakdown in the preconditioning strategy by checking the entries in $\tilde{D} + \tilde{E}_k$. If a diagonal entry of $\tilde{D} + \tilde{E}_k$ is close to zero, we can apply a diagonal shift of matrix $\tilde{D} + \tilde{E}_k$ or inhibit the update as follows. Let $P_k = \tilde{W}(\tilde{D} + \tilde{E}_k)^{-1}\tilde{Z}^T$ in (4.5) be the candidate preconditioner for matrix J_k and denote by $[\tilde{D} + \tilde{E}_k]_{i,i}$ the ith diagonal element of matrix $\tilde{D} + \tilde{E}_k$. If

$$\min_{i=1,\ldots,n} \left| [\tilde{D} + \tilde{E}_k]_{i,i} \right| > \omega, \tag{4.13}$$

for a prescribed tolerance ω, then the candidate preconditioner P_k is updated. Otherwise P_k is abandoned and replaced by the one used in the previous Newton equation. The above control prevents updates with small pivots and works well in practice as shown in §5.

4.3. Approximations and convergence analysis. In this section we analyze the quality of the proposed updated preconditioners. Some results will be provided working with P_k^{-1} rather than P_k, although in practice only P_k is employed.

Intuitively, the effectiveness of the preconditioner P_k in (4.5) depends on the magnitude of the elements discarded by the functions f and g in (4.6). For this reason, we introduce the functions

$$o_f(A) = A - f(A), \quad o_g(A) = A - g(A),$$

where A is a matrix, f and g are the functions of the form (4.10) used in (4.6). In practice, $o_f(A)$ and $o_g(A)$ consist of the elements of A which are not selected by functions f and g, respectively. Using these functions, we provide a formal expression of the matrices $\tilde{\Delta}_k$ and $\tilde{E}_k$ given in (4.6). Since

$$\begin{aligned}\tilde{E}_k &= g(\tilde{Z}^T\,\tilde{\Delta}_k\,\tilde{W})\\ &= g(\tilde{Z}^T\,\Delta_k\,\tilde{W}) - g(\tilde{Z}^T\,o_f(\Delta_k)\,\tilde{W})\\ &= \tilde{Z}^T\,\Delta_k\,\tilde{W} - o_g(\tilde{Z}^T\,\Delta_k\,\tilde{W}) - g(\tilde{Z}^T\,o_f(\Delta_k)\,\tilde{W}),\end{aligned}$$

letting

$$\Theta_1 = -o_g(\tilde{Z}^T\,\Delta_k\,\tilde{W}), \quad \Theta_2 = -g(\tilde{Z}^T\,o_f(\Delta_k)\,\tilde{W}), \tag{4.14}$$

we obtain

$$\tilde{E}_k = \tilde{Z}^T\,\Delta_k\,\tilde{W} + \Theta_1 + \Theta_2. \tag{4.15}$$

Following [22, Lemma 2.1], the analysis is carried assuming that P_{seed}^{-1} is close to J_{seed}. In the following we let

$$\nu = \|\tilde{Z}^{-T}\|\,\|\tilde{W}^{-1}\|. \tag{4.16}$$

THEOREM 4.1. *Let P_{seed} and P_k be given in (4.4) and (4.5). Assume that P_{seed} satisfies*

$$\|J_{seed} - P_{seed}^{-1}\| = \epsilon\|J_{seed}\| < \|J_{seed} - J_k\|, \tag{4.17}$$

for some positive ϵ. Then

$$\|J_k - P_k^{-1}\| \leq \frac{\epsilon\|J_{seed}\| + \nu(\|\Theta_1\| + \|\Theta_2\|)}{\|J_k - J_{seed}\| - \epsilon\|J_{seed}\|}\,\|J_k - P_{seed}^{-1}\|, \tag{4.18}$$

where ν is given in (4.16).

Proof. By (4.2), (4.5) and (4.15) we have

$$\begin{aligned}J_k - P_k^{-1} &= (J_k - J_{seed}) + (J_{seed} - P_{seed}^{-1}) - (P_k^{-1} - P_{seed}^{-1})\\ &= \Delta_k + (J_{seed} - P_{seed}^{-1}) - (\tilde{Z}^{-T}(\tilde{D} + \tilde{E}_k)\tilde{W}^{-1} - \tilde{Z}^{-T}\tilde{D}\tilde{W}^{-1})\\ &= \Delta_k + (J_{seed} - P_{seed}^{-1}) - \tilde{Z}^{-T}\tilde{E}_k\tilde{W}^{-1}.\\ &= \Delta_k + (J_{seed} - P_{seed}^{-1}) - \Delta_k - \tilde{Z}^{-T}(\Theta_1 + \Theta_2)\tilde{W}^{-1}. \qquad (4.19)\end{aligned}$$

Thus, (4.17) gives

$$\|J_k - P_k^{-1}\| \leq \epsilon\|J_{seed}\| + \nu(\|\Theta_1\| + \|\Theta_2\|). \tag{4.20}$$

Condition (4.17) provides also the following bound

$$\begin{aligned}\|J_{seed} - J_k\| - \epsilon\|J_{seed}\| &= \|J_{seed} - J_k\| - \|J_{seed} - P_{seed}^{-1}\|\\ &\leq \|J_k - P_{seed}^{-1}\|.\end{aligned}$$

This fact along with (4.20) yields

$$\begin{aligned}\|J_k - P_k^{-1}\| &\leq \frac{\epsilon\|J_{seed}\| + \nu(\|\Theta_1\| + \|\Theta_2\|)}{\|J_k - J_{seed}\| - \epsilon\|J_{seed}\|}\,(\|J_k - J_{seed}\| - \epsilon\|J_{seed}\|)\\ &\leq \frac{\epsilon\|J_{seed}\| + \nu(\|\Theta_1\| + \|\Theta_2\|)}{\|J_k - J_{seed}\| - \epsilon\|J_{seed}\|}\,\|J_k - P_{seed}^{-1}\|,\end{aligned}$$

which completes the proof. □

The above theorem shows that the distance of P_k^{-1} from J_k depends from the quality of P_{seed}^{-1} as an approximation to J_{seed} and from the discarded quantities Θ_1 and Θ_2 in forming $\tilde{E}_k$. In particular, small norms $\|\Theta_1\|$, $\|\Theta_2\|$ may yield $\|J_k - P_k^{-1}\| < \|J_k - P_{seed}^{-1}\|$, i.e. an improvement of the updated preconditioner upon P_{seed}. This case can occur if the approximations (4.6) tend to contain most of the significant entries of the matrices involved, e.g. when the Jacobian varies slowly and the entries of $\tilde{Z}$, $\tilde{W}$ decay fast away from the main diagonal, [9, 32].

In Section 5 we will show that updating the preconditioner compares favorably with using the frozen preconditioner J_{seed}. On the other hand, in case the quality of P_k deteriorates and the convergence of the linear solver becomes too slow, a poor descent direction for $\|F\|_2$ at x_k is computed and the backtracking strategy is likely to fail. One possibility to cope with this situation is to initialize a new reference matrix J_{seed} and the related preconditioner P_{seed}.

In the following theorems, assuming that J is Lipschitz continuous in a convex set containing a solution x^* of $F(x) = 0$, we will focus on the occurrence where x_{seed} is close enough to x^* to guarantee local convergence of the Inexact Newton method. In this case, the iterates x_k subsequent to x_{seed} belong to the ball of radius $\|x^* - x_{seed}\|$, and $\|\Delta_k\| = O(\|x_k - x_{seed}\|)$ from the Lipschitz continuity of J. To analyze the spectral properties of matrix $J_k P_k$ we first state the local convergence properties of Algorithm 2.1 which can be shown by an easy modification of arguments used in Theorem 11.3 in [33] and Theorem 6.1 in [25].

THEOREM 4.2. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable and x^* such that $F(x^*) = 0$. Using the vector norm $\|\cdot\|$ and the induced matrix operator norm, let J be Lipschitz continuous with constant $\Lambda/2$ in a ball $B(x^*, r)$ centered at x^* with radius $r > 0$, and such that $J(x^*)$ is nonsingular. Assume that $\eta_k \to 0$. Then, there exists $\delta > 0$ such that if $||x_0 - x^*|| \leq \delta$, then the sequence $\{x_k\}$ generated by Algorithm 2.1 is well defined, i.e. J_k is nonsingular for $k \geq 0$, $x_k \in B(x^*, \delta)$, s_k and η_k are accepted without modification in the while-loop and $x_k \to x^*$ superlinearly.*

We now state a formal result that holds under the assumption that our updating strategy does not break down. Hence, we assume that the updated preconditioners (4.5) are well defined (nonsingular). We stress that in case a diagonal approximation is used for $\tilde{E}_k$, this assumption is automatically satisfied by enforcing (4.13).

THEOREM 4.3. *Let the assumptions of Theorem 4.2 hold, $\{x_k\}$ be the sequence generated by the Newton-Krylov Algorithm 2.1, $x_{seed} = x_0$. Assume that the preconditioner P_{seed} in (4.4) is computed using a threshold τ,*

$$\|J_{seed} - P_{seed}^{-1}\| \leq h\tau, \tag{4.21}$$

for some positive h independent of τ and $||\hat{Z}|| \leq \zeta$, $||\hat{W}|| \leq \zeta$. Then the right preconditioned Jacobian matrix $J_k P_k$ can be written as

$$J_k P_k = I + R_k P_k, \quad R_k = J_k - P_k^{-1},$$

where, for the 1 and infinity norms

$$||R_k|| \leq h\tau + (1 + c)\Lambda\delta, \tag{4.22}$$

for some $c \geq 0$, and for the Euclidean norm

$$||R_k||_2 \leq h\tau + (1+c)\sqrt{n}\Lambda\delta, \tag{4.23}$$

for some $c \geq 0$. Moreover, for the 1 and infinity norms

$$||R_k\, P_k|| \leq \zeta^2\ (h\tau + (1+c)\Lambda\delta)\ \|(\tilde{D} + \tilde{E}_k)^{-1}\|, \tag{4.24}$$

while for the Euclidean norm

$$||R_k\, P_k||_2 \leq \frac{\zeta^2}{\sigma_{\min}(\tilde{D} + \tilde{E}_k)} \left(h\tau + (1+c)\sqrt{n}\Lambda\delta\right). \tag{4.25}$$

Proof. The inverse of the updated preconditioner is given by

$$P_k^{-1} = \tilde{Z}^{-T} \left(\tilde{D} + g\left(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}\right)\right) \tilde{W}^{-1},$$

where $\tilde{\Delta}_k$ and g are defined in (4.6). We observe that

$$R_k = J_k - P_k^{-1} = J_{seed} + \Delta_k - \tilde{Z}^{-T} \left(\tilde{D} + g\left(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}\right)\right) \tilde{W}^{-1},$$

can be split as

$$R_k = \left(J_{seed} - P_{seed}^{-1}\right) + \left(\Delta_k - \tilde{Z}^{-T} g\left(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}\right) \tilde{W}^{-1}\right). \tag{4.26}$$

Let us consider the following upper bound for $||R_k||$

$$||R_k|| \leq \|J_{seed} - P_{seed}^{-1}\| + \underbrace{||\Delta_k - \tilde{Z}^{-T} g\left(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}\right) \tilde{W}^{-1}||}_{b}. \tag{4.27}$$

The term $\|J_{seed} - P_{seed}^{-1}\|$ is bounded as in (4.21). Regarding the term b, first suppose to use either the 1 or infinity norm. If the operator g is such that

$$g\left(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}\right) = \tilde{Z}^T \tilde{\Delta}_k \tilde{W}, \tag{4.28}$$

then

$$b = ||\Delta_k - \tilde{\Delta}_k|| = \|o_f(\Delta_k)\| \leq \|\Delta_k|| = ||J_k - J_{seed}||.$$

More generally, in case (4.28) does not hold, it is easy to see that

$$b \leq (1+c)\, ||\Delta_k||,$$

for some nonnegative scalar c.

Now, the bound (4.22) can be obtained using the Lipschitz continuity of the Jacobian. In fact,

$$\|\Delta_k\| \leq \frac{\Lambda}{2}||x_k - x_{seed}|| \leq \frac{\Lambda}{2}(||x^* - x_{seed}|| + ||x_k - x^*||) \leq \Lambda\delta, \tag{4.29}$$

where the last inequality holds as all iterates x_k are contained in the ball $B(x^*, \delta)$. Finally, by

$$||P_k|| \leq ||\tilde{W}||\, ||\tilde{Z}||\, ||(\tilde{D} + \tilde{E}_k)^{-1}||,$$

and (4.22) we obtain (4.24).

Let now consider the use of the Euclidean norm. The inequalities (4.23) and (4.25) are obtained proceeding as above noting that

$$b = \|o_f(\Delta_k)\|_2 \leq \sqrt{\|o_f(\Delta_k)\|_1 \|o_f(\Delta_k)\|_\infty} \leq \sqrt{\|\Delta_k\|_1 \|\Delta_k\|_\infty} \leq \sqrt{n}\|\Delta_k\|_2.$$

□

The above theorem shows that the eigenvalue cluster's radius of $J_k P_k$ depends on the magnitude of the norms $\|\tilde{Z}\|$, $\|\tilde{W}\|$, $\|x_{seed} - x_*\|$ and the accuracy of the seed preconditioner. In fact, the update technique solves a sequence of systems through subsequent exploitation of information from the current matrix J_k and Theorem 4.3 suggests that the updated preconditioner can give a clustered spectrum as if it would have been recomputed. We believe that this result supports the updating strategy as an improvement upon reusing the seed preconditioner which may give poor results even for small variations of the matrices in the sequence, [5, 9].

As a consequence of Theorem 4.3, we have the following results.

COROLLARY 4.4. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ satisfies the hypotheses of Theorem 4.3. Then there exist δ^* and τ^* such that, for any $0 < \delta \leq \delta^*$ and $0 < \tau \leq \tau^*$, the eigenvalues of the preconditioned matrices $J_k P_k$ are clustered at 1 in the right half complex plane for all k.*

Proof. Consider the bounds (4.24), (4.25) and let δ^* and τ^* be a couple of real positive numbers such that for any $0 < \delta \leq \delta^*$ and $0 < \tau \leq \tau^*$ either

$$\rho_1 = \zeta^2 \ (h\tau + (1+c)\Lambda\delta) \ \|(\tilde{D} + \tilde{E}_k)^{-1}\| < 1,$$

or

$$\rho_2 = \frac{\zeta^2}{\sigma_{\min}(\tilde{D} + \tilde{E}_k)} \left(h\tau + (1+c)\sqrt{n}\Lambda\delta\right) < 1.$$

Then we get a cluster at 1 in the right half complex plane with radius equal to ρ_1 or ρ_2, respectively. □

We note that, with a clustered spectrum, we can expect a fast convergence of preconditioned iterations. Since the matrices of the algebraic linear systems are non-symmetric, convergence behavior of the Krylov subspace method used to solve these linear systems depends on further characteristics such as departure from symmetry, condition number of the matrix of eigenvectors, pseudospectra; see, e.g., the convergence analysis of GMRES in [37]. On the other hand, in our numerical tests, we use BiCGSTAB that does not enforce an optimality condition, [37, Chapter 3] and we found that very often eigenvalues can give useful insights for many nonsymmetric problems when the Jacobian matrix J is diagonalizable; see, e.g., [11, 12]. Therefore, we will not consider this issue any further.

COROLLARY 4.5. *Under the hypotheses of Theorem 4.3, if $\tilde{E}_k$ is a diagonal approximation for E_k and $[\tilde{D} + \tilde{E}_k]_{i,i}$ is the ith diagonal entry of the diagonal matrix $\tilde{D} + \tilde{E}_k$ we have*

$$J_k P_k = I + R_k P_k,$$

where

$$||R_k P_k|| \leq \frac{h\tau + (1+c)\Lambda\delta}{\min_i \left|[\tilde{D} + \tilde{E}_k]_{i,i}\right|},$$

for 1 and infinity norm and

$$||R_k P_k||_2 \leq \frac{h\tau + (1+c)\Lambda\sqrt{n}\delta}{\min_i \left|[\tilde{D} + \tilde{E}_k]_{i,i}\right|},$$

Proof. The thesis follows from (4.24) and (4.25) by observing that

$$\|(\tilde{D} + \tilde{E}_k)^{-1}\|_1 = \|(\tilde{D} + \tilde{E}_k)^{-1}\|_\infty = \Big(\sigma_{min}(\tilde{D} + \tilde{E}_k)\Big)^{-1} = \frac{1}{\min_i \left|[\tilde{D} + \tilde{E}_k]_{i,i}\right|}.$$

□

We conclude our analysis providing a relation between the inverse of the update preconditioner P_k and a preconditioner P_R recomputed from scratch for J_k and such that

$$\|J_k - P_R\| = \epsilon\|J_k\|. \tag{4.30}$$

The matrix P_R can be computed in the form of an incomplete LDU factorization of J_k or can be regarded as the inverse of an approximate inverse preconditioner for J_k. Without loss of generality, we assume that the scalar ϵ used above is the same of (4.17).

Corollary 4.6. *Let P_k be given in (4.5) and P_R be a recomputed preconditioner for J_k. If P_{seed} satisfies (4.17) and P_R (4.30), then*

$$\|P_k^{-1} - P_R\| \leq \epsilon(\|J_{seed}\| + \|J_k\|) + \nu(\|\Theta_1\| + \|\Theta_2\|), \tag{4.31}$$

where ν is defined as in (4.16). Moreover, under the assumptions of Theorem 4.3 and using norm 1 or infinity, it follows

$$\|P_k^{-1} - P_R\| \leq \epsilon(\|J_{seed}\| + \|J_k\|) + 2\Lambda\nu\zeta^2\delta. \tag{4.32}$$

Proof. By (4.19) we have

$$\begin{aligned} P_k^{-1} - P_R &= (P_k^{-1} - J_k) + (J_k - P_R) \\ &= -(J_{seed} - P_{seed}^{-1}) + \tilde{Z}^{-T}(\Theta_1 + \Theta_2)\tilde{W}^{-1} + (J_k - P_R). \end{aligned} \tag{4.33}$$

Thus, (4.17) and (4.30) give (4.31).

Let us now assume the hypotheses of Theorem 4.3. By (4.14), it easily follows

$$\|\Theta_1\| \leq \|\tilde{Z}^T\, \Delta_k\, \tilde{W}\| \leq \zeta^2\|\Delta_k\|,$$

and

$$\|\Theta_2\| \leq \|\tilde{Z}^T\, o_f(\Delta_k)\, \tilde{W}\| \leq \zeta^2\|o_f(\Delta_k)\|.$$

Then, using (4.29), we get (4.32). □

Inequality (4.31) can be viewed as a condition of bounded deterioration of the updated preconditioner with respect to a recomputed and accurate preconditioner P_R. In fact, P_k^{-1} is near to P_R whenever the quantities ϵ, $\|\Theta_1\|$, $\|\Theta_2\|$ are sufficiently small. Note that $\|\Theta_1\|$, $\|\Theta_2\|$ are small whenever x_{seed} and x_k are close to a solution x^*.

5. Numerical Results. In this section we give some details about the procedures that were actually implemented. Then, we present some numerical experiments which illustrate the efficiency and reliability of our preconditioning technique.

5.1. Implementation details. The Krylov solver used in our implementation is BiCGSTAB, [37]. The level of inexactness in the solution of the Newton equations is fixed in order to avoid *oversolving* and to guarantee quadratic convergence of the Newton-Krylov procedure. Following the results by Eisenstat and Walker [26, 36], we set $\eta_0 = \eta_{max} = 0.5$ and used the so-called $Choice\ 2$

$$\eta_k = \gamma \left(\frac{\|F(x_{k+1})\|_2}{\|F(x_k)\|_2} \right)^2, \ k \geq 1,$$

with $\gamma = 0.9$ and safeguard

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

if $\gamma \bar{\eta}_{k-1}^2 > 0.1$. Then, the additional safeguard $\eta_k = \min\{\eta_k, \eta_{max}\}$ is imposed.

Starting from the null initial guess, a maximum of $\texttt{LI}_{\texttt{max}} = 400$ linear iterations is allowed. If within $\texttt{LI}_{\texttt{max}}$ iterations, BiCGSTAB does not provide a step satisfying (2.3), we proceed as described in §3. Specifically, we check if the last computed step is an inexact Newton step and in the affirmative case we apply the backtracking strategy along it.

In the backtracking strategy, the scalar α is set to 10^{-4} and σ is computed by the three-point parabolic rule. A maximum of $B_{max} = 20$ backtracks is allowed.

In our setting, right preconditioners are applied and the reference matrices J_{seed} are computed by finite differences avoiding the approximation of known zero entries. The matrix $J_{seed} = J_0$ is used as the first reference matrix and an approximate sparse inverse preconditioner $P_{seed} = P_0$ is constructed using two thresholds. Specifically the incomplete LDU factorization of J_{seed} is computed by the Matlab function `luinc` with a specified droptol `drop_ILU` and the approximate inversion of the factors L and U is obtained using a drop tolerance `drop_AI`. Then, the update of the preconditioner is performed allowing for a diagonal or tridiagonal banded approximation in (4.6).

In case a diagonal approximation is used, the update of the preconditioner is performed monitoring the magnitude of the entries of $\tilde{D} + \tilde{E}_k$. In particular, let $P_k = \tilde{W}(\tilde{D} + \tilde{E}_k)^{-1}\tilde{Z}^T$ in (4.5) be the candidate preconditioner for matrix J_k and denote by $[\tilde{D} + \tilde{E}_k]_{i,i}$ the ith diagonal elements of matrix $\tilde{D} + \tilde{E}_k$. If

$$\min_{i=1,\ldots,n} \left| [\tilde{D} + \tilde{E}_k]_{i,i} \right| \leq 10^{-4} \|J_{seed}\|_1, \tag{5.1}$$

then the candidate preconditioner P_k is abandoned and replaced by the one used in the previous Newton equation. In fact, the preconditioner is frozen.

From the analysis performed in the previous section, we know that if the difference between J_k and J_{seed} becomes large then the quality of the updated preconditioner P_k can deteriorate and the convergence of the linear solver can slow down. This may yield a poor descent direction for $\|F\|_2$ at x_k and the failure of the backtracking strategy. Therefore, a new reference matrix and preconditioner are initialized after $\texttt{LI}_{\texttt{max}}$ linear iterations to solve the last Newton iteration. As a consequence, an LDU factorization of J_{seed} and sparsified factors $\tilde{Z}$ and $\tilde{W}$ in (4.8) are updated using the procedure described above.

Our preconditioning technique (`Update`) is compared here with three other strategies. The first strategy (`Freeze`) consists in computing an ILU preconditioner only for the first reference matrix J_{seed} and reusing it in all the subsequent nonlinear iterations. The second strategy (`Refresh`) refreshes the preconditioner occasionally; in fact the preconditioner P_{seed} is frozen and, in case the limit of $\mathtt{LI}_{\max}$ linear iterations is reached, it is computed from scratch. The third strategy (`Recomp`) computes an ILU preconditioner for each Newton equation. In all cases, the Matlab function `luinc` and droptol `droptol_ILU` are employed.

The algorithms sketched above were written in Matlab 7.6 and run on a Intel Xeon (TM) 3.4 Ghz desktop with 1GB RAM. The machine precision is $\epsilon_m \simeq 2 \cdot 10^{-16}$.

5.2. Test problems, notations and parameters. We report here runs with some classical benchmark problems: the *nonlinear convection-diffusion problem*, the *flow in a porous medium problem*, the *countcurrent reactor problem* and the *2D driven cavity problem*. To describe the results of these experiments we give the dimension n of the discretized problem, the numbers "`NI`" and "`LI`" of nonlinear and linear iterations performed, the number "`NJ`" of reference matrices J_{seed} needed for the solve. Trivially, `NJ=1` for the `Freeze` strategy and `NJ=NI` for the `Recomp` strategy.

We monitor the seconds of CPU "`Time_P`" needed, on average, for computing one reference Jacobian matrices and its ILU decomposition and the overall execution time "`Time`" in seconds taken to perform the Newton-Krylov procedure. The CPU time `Time` measures the execution time needed for the overall procedure; hence it includes the cost of triangular solves in the `Freeze` and `Recomp` strategies and the approximate inversion of the factors L and U in the `Update` strategy.

Finally, we examine the fill-in occurred for computing the preconditioner. The data monitored in the `Freeze` and `Recomp` strategies are

$$\mathtt{FL\_LU} = \frac{nnz(L) + nnz(U) - n}{n^2}, \tag{5.2}$$

where L and U are the matrices in (4.7) and $nnz(\cdot)$ is the number of nonzero entries. In the `Update` strategy we examine the matrices $\tilde{Z}$, $\tilde{W}$ in (4.8) and compute

$$\mathtt{FL\_ZW} = \frac{nnz(\tilde{Z}) + nnz(\tilde{W}) - n}{n^2}. \tag{5.3}$$

We declare a successful termination of the Linesearch Newton-Krylov method whenever

$$\|F_k\|_2 < 10^{-8}. \tag{5.4}$$

A failure is declared when the previous condition is not satisfied within 100 nonlinear iterations or a sufficient decrease on $\|F\|_2$ is not obtained within $\mathtt{B}_{\max} = 20$ backtracks. The latter failure will be denoted as "`F_B`".

The nonlinear convection-diffusion problem. The two-dimensional nonlinear convection-diffusion model problem has the form, see e.g. [27]

$$\begin{aligned} -\Delta u + Re\, u(u_x + u_y) &= f(x,y) \quad && \text{in } \Omega = [0,1] \times [0,1], \\ u &= 0 && \text{in } \partial\Omega, \end{aligned}$$

where $f(x,y) = 2000x(1-x)y(1-y)$, and Re is the Reynolds number. We discretized this problem using second order centered finite differences on a uniform $m \times m$ grid and performed runs for various uniform meshes and Reynolds numbers.

		Freeze		Recomp		Refresh			Update		
Re	n	NI	LI	NI	LI	NI	LI	NJ	NI	LI	NJ
250	22500	19	3036	14	37	14	507	2	16	405	1
	40000	13	1250	15	55	13	1250	1	15	359	1
	62500	14	1154	15	78	14	1154	1	14	520	1
500	22500	F_B	–	15	36	16	788	2	18	781	2
	40000	26	4292	14	42	16	755	2	17	647	1
	62500	15	2119	15	58	14	1670	2	16	703	1
1000	22500	F_B	–	18	38	16	638	2	19	934	2
	40000	F_B	–	17	46	19	634	2	20	959	1
	62500	F_B	–	17	62	18	2147	2	19	977	1

TABLE 5.1

Nonlinear convection-diffusion problem. Test results: number of nonlinear and linear iterations, number of Jacobian evaluations.

The initial guess for the discretized unknown is the null vector. In the preconditioning strategies, the drop tolerances used are $\texttt{drop\_ILU} = 10^{-2}$, $\texttt{drop\_AI} = 10^{-1}$. The updates (4.5) of the preconditioner are performed using tridiagonal matrices in (4.6), i.e.

$$\tilde{\Delta}_k = \text{band}(\Delta_k, 1, 1), \quad \tilde{E}_k = \text{band}(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}, 1, 1).$$

In Tables 5.1, 5.2 we report the results obtained for the values $m = 150$, 200, 250, and $Re = 250$, 500, 1000. In Table 5.2 the time Time_P needed for computing the reference Jacobians and their ILU decomposition is mostly ascribed to the construction of the Jacobian matrices. The values of the fill-in parameters FL_LU and FL_ZW given in (5.2), (5.3) vary in the ranges $[2 \cdot 10^{-4}, 8 \cdot 10^{-4}]$ and $[4 \cdot 10^{-4}, 7 \cdot 10^{-2}]$, respectively.

The Freeze strategy lacked robustness and required a large number of linear iterations. Refreshing the preconditioner was beneficial and gave considerable gains in robustness and efficiency. All failures of the Freeze strategy were recovered by the Refresh strategy and in the tests where the two strategies did not coincide, two Jacobians and preconditioners were computed.

Remarkably, in most runs the Update procedure resulted to be faster than refreshing the Jacobian occasionally as only the reference Jacobian $J_{seed} = J_0$ was formed ($\texttt{NJ} = 1$). In the remaining two runs, it is outperformed by the Refresh strategy.

Computing the preconditioner for each system of the sequence is time-consuming and unnecessary. In the Recomp strategy, the number of linear iterations needed is very low, as expected, while the number of nonlinear iterations is comparable to that of the Refresh and Update strategies. Since the gain in the number of nonlinear iterations was marginal, computing the preconditioners from scratch resulted highly time-consuming and the execution time Time is mostly ascribed to such phase.

To give more insight into the behavior of our preconditioning strategy, we focused on the test where $Re = 250$ and $n = 40000$ and in Figures 5.1, 5.2, we plotted some results from the convergence history of the strategies considered. From the bottom part of Figure 5.1 showing the values $\|x_k - x_0\|_1 = \|x_k - x_{seed}\|_1$ over the nonlinear iterations, it is evident that the distance of x_k from x_{seed} is quite large. In particular, $\|x_k - x_0\|_1$ reaches the value of about $7 \cdot 10^3$ at the end of the iterative process. On the other hand, the Jacobian varies slowly; the value of $\|\Delta_k\|_1$ remains quite small starting from the value of about 0.6 at the first iteration and reaching the value 3.6

			Freeze	Recomp	Refresh		Update	
Re	n	Time_P	Time	Time	Time	NJ	Time	NJ
250	22500	46	102	666	105	2	64	1
	40000	191	226	2901	226	1	220	1
	62500	600	656	8013	656	1	657	1
500	22500	46	–	698	110	2	128	2
	40000	101	322	2630	397	2	234	1
	62500	600	674	7850	1271	2	671	1
1000	22500	46	–	865	108	2	147	2
	40000	191	–	3318	402	2	249	1
	62500	600	–	8691	1267	2	692	1

TABLE 5.2
Nonlinear convection-diffusion problem. Test results: execution times.

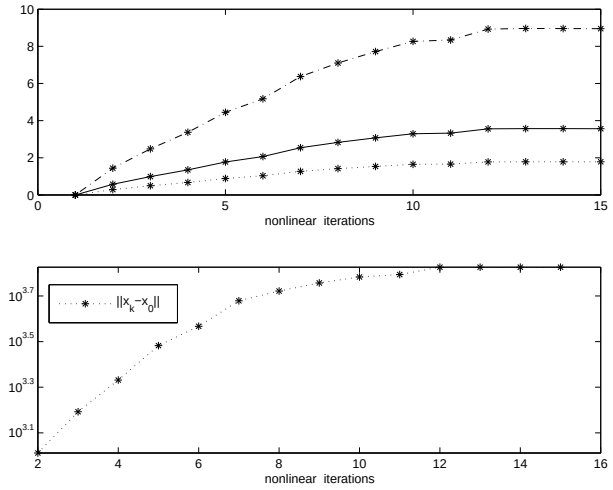

FIG. 5.1. *Convergence history of the nonlinear convection-diffusion problem with* $Re = 250$ *and* $n = 40000$ *solved by the* `Update` *strategy. The horizontal axis indicates the nonlinear iterations, asterisks indicate a new nonlinear iteration. In the upper part of the figure: the dotted curve represents* $\|o_f(\Delta_k)\|_1$*, the solid curve represents* $\|\Delta_k\|_1$*, the dashed curve is* $\|o_g(\tilde{Z}\tilde{\Delta}_k\tilde{W}^T)\|_1$*. In the bottom part the quantity* $\|x_k - x_0\|_1$ *is plotted in logarithmic scale.*

at the end of the iterative process. This is shown in the upper part of the figure, along with the values of $\|o_f(\Delta_k)\|_1$ and $\|o_g(\tilde{Z}\tilde{\Delta}_k\tilde{W}^T)\|_1$, i.e the quantities discarded by our preconditioning update. Obviously, all this quantities increase as long as the iterative process proceeds, but they remain quite small. This fact explains the good behavior of the updating strategy that does not require any refresh of J_{seed} even if the starting guess is far from the solution. Finally, in Figure 5.2 we show the number of linear iterations `LI` needed in the `Freeze` and `Update` strategies. It is interesting to note that, despite the slow variation of the Jacobians, the frozen preconditioner deteriorates in the progress of iterations. On the other hand, the `Update` strategy is very effective in the solution of most linear systems.

Countercurrent reactor problem. This problem is a system of equations resulting

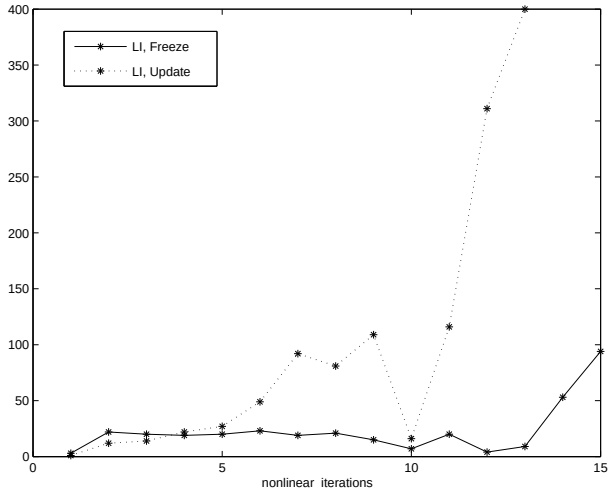

FIG. 5.2. *Convergence history of the nonlinear convection-diffusion problem with* $Re = 250$ *and* $n = 40000$. *The horizontal axis indicates the nonlinear iterations, asterisks indicate a new linear iteration. The dashed curve indicates the number of linear* `LI` *iteration in the* `Freeze` *strategy, the solid curve indicates the number of linear* `LI` *iteration in the* `Update` *strategy.*

from a set of countercurrent reactors and takes the form

$$\begin{aligned}
F_i(x) &= \beta - (1-\beta)x_{i+2} - x_i(1+4x_{i+1}), && i = 1,\\
F_i(x) &= -(2-\beta)x_{i+2} - x_i(1+4x_{i-1}), && i = 2,\\
F_i(x) &= \beta x_{i-2} - (1-\beta)x_{i+2} - x_i(1+4x_{i+1}), && \mathrm{mod}(i,2) = 1,\ 2 < i < n-1,\\
F_i(x) &= \beta x_{i-2} - (2-\beta)x_{i+2} - x_i(1+4x_{i-1}), && \mathrm{mod}(i,2) = 0,\ 2 < i < n-1,\\
F_i(x) &= \beta x_{i-2} - x_i(1+4x_{i+1}), && i = n-1,\\
F_i(x) &= \beta x_{i-2} - (2-\beta) - x_i(1+4_{i-1}), && i = n,
\end{aligned}$$

where $\beta = 0.5$, [15]. The preconditioners are constructed using the tolerances

$$\texttt{drop\_ILU} = 10^{-1}, \quad \texttt{drop\_AI} = 10^{-1}.$$

Then, our update (4.5) of the preconditioner was carried out in a matrix-free setting extracting the main diagonal from the matrices Δ_k and E_k in (4.2).

In Table 5.3 we show the results obtained using the starting guess $x_0 = (\beta, \ldots, \beta)^T$ and various problem dimensions. The `Freeze` and `Refresh` strategies failed to solve all tests and are not reported in the table.

In the `Recomp` strategy, the number of nonlinear and linear iterations slightly varies with the dimension n. On the other hand, the timings steadily increase with the dimension and are higher than in the `Update` strategy. In fact, the `Recomp` technique takes advantage from the computation of the preconditioner at each iteration from scratch, but savings in the linear iterations do not compensate the cost of evaluating the Jacobians and forming the preconditioners.

In the `Update` strategy, condition (5.1) was never met and the preconditioner was updated at each nonlinear iteration. Moreover, the reference matrix J_{seed} and the corresponding preconditioner P_{seed} were recomputed one time in the progress of the nonlinear iterations ($\texttt{NJ} = 2$). Concerning the number of linear iterations performed, since the reference preconditioner was refreshed, $\texttt{LI}_{\texttt{max}}$ iterations were performed in

		Recomp			Update			
n	Time_P	NI	LI	Time	NI	LI	NJ	Time
6400	3	10	15	32	11	524	2	22
8100	4	10	14	46	12	479	2	34
10000	6	10	14	63	18	630	2	38
12100	9	10	14	93	19	728	2	60
15625	14	11	15	157	17	931	2	76

TABLE 5.3

Countercurrent reactor problem. Test results: number of nonlinear and linear iterations, number of Jacobian evaluations, execution times

the solution of one linear systems. Excluding such system from the statistics, the average number of linear iteration per systems varied between 7 and 31.

Finally, we note that the maximum levels of fill-in FL_LU, FL_ZW in (5.2), (5.3) are of order 10^{-4} and 10^{-1} in the Recomp and Update strategies respectively.

Flow in a porous medium. The problem we consider is a steady state special case of a general equation that models the influence of capillary pressure and gravity on flow in a homogeneous medium [39]

$$\Delta(u^2) + d\frac{\partial}{\partial x}(u^3) + f = 0, \quad \text{in} \quad \Omega = [0,1] \times [0,1],$$
$$u = \begin{cases} 1, & \text{if } x = 0 \text{ or } y = 0, \\ 0, & \text{if } x = 1 \text{ or } y = 1, \end{cases} \quad \text{on} \quad \partial\Omega.$$

In our experiments we use the parameters as in [36]; d is set equal to 50 and f is a point source of magnitude 50 at the lower-left grid point. We discretized the problem using centered differences on a uniform grid and the initial guess for the discretized unknown is $u(x, y) = 1 - xy$ on the interior grid points.

To form the preconditioners we fix drop_ILU $= 10^{-1}$, drop_AI $= 10^{-1}$. Then, our updates are performed in a matrix-free setting by extracting the main diagonal from the matrices Δ_k and E_k in (4.2). Thus, from (4.5) we have

$$\tilde{\Delta}_k = \text{band}(\Delta_k, 0, 0), \quad \tilde{E}_k = \text{band}(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}, 0, 0).$$

Tables 5.4, 5.5 display the results obtained for $m \times m$ grids with $m = 100, 125, 150, 175$. The values FL_LU, FL_ZW of fill-in given in (5.2), (5.3) are small: FL_LU varies in the range $[2 \cdot 10^{-4}, 5 \cdot 10^{-4}]$ and FL_ZW varies in the range $[3 \cdot 10^{-4}, 9 \cdot 10^{-4}]$.

As usual, we inhibited the update of the preconditioner if the condition (5.1) holds; the updates of the preconditioner in the runs reported varied between five and six and in three cases out of four produced an improvement upon the other strategies. Specifically, the Freeze and Refresh strategies differed in the run corresponding to $n = 15625$ where the Refresh strategy outperformed the Update strategy in terms of linear iterations and execution time. In the remaining runs, the overall cost of the procedure was favorable to the Update strategy and major savings were obtained with $n = 22500$ and $n = 30625$.

In accordance to the previous problems, the margin of superiority of the Recomp strategy in the number of nonlinear iterations was slight and the gains in the values NI and LI did not compensate the overhead associated with initializing new Jacobians and preconditioners at each nonlinear iteration.

	Freeze		Recomp		Refresh			Update		
n	NI	LI	NI	LI	NI	LI	NJ	NI	LI	NJ
10000	14	1205	12	190	14	1205	1	15	1143	1
15625	17	1675	18	54	16	954	2	16	1256	1
22500	18	2560	13	288	18	2560	1	16	1284	1
30625	16	3345	13	346	16	3345	1	17	1598	1

TABLE 5.4

Flow in a porous medium. Test results: number of nonlinear and linear iterations, number of Jacobian evaluations.

		Freeze	Recomp	Refresh		Update	
n	Time_P	Time	Time	Time	NJ	Time	NJ
10000	45	285	526	285	1	280	1
15625	129	877	2224	675	2	734	1
22500	327	2900	4103	2900	1	1286	1
30625	524	6379	8304	6379	1	3290	1

TABLE 5.5

Flow in a porous medium. Test results: execution times.

The driven cavity problem. The two-dimensional driven cavity problem has the form

$$\begin{aligned} \frac{1}{Re}\Delta^2 u + (u_y \Delta u_x - u_x \Delta u_y) = 0 \quad & \text{in } \Omega = [0,1]\times[0,1], \\ u = 0 \quad & \text{on } \partial\Omega, \\ \frac{\partial u}{\partial n} = \begin{cases} 1 & \text{if } y = 1 \\ 0 & \text{otherwise} \end{cases} \quad & \text{on } \partial\Omega, \end{aligned}$$

where Re is the Reynolds number. We discretized by 13-point finite differences on a uniform $m \times m$ grid generating a system of $n = m^2$ nonlinear equations and started from the null initial guess. The code implementing this discretization is provided by P. Brown; see [36].

The sequence of Newton equations resulted hard to be solved by BiCGSTAB. Sparse preconditioners caused BiCGSTAB to stagnate at the first nonlinear iteration and for this reason we allowed more fill-in setting drop_ILU= 10^{-3} for the incomplete L, U factors and drop_AI= 10^{-2} in the approximate inversion of the L and U factors.

As stressed in [22], this problem represents the case where the refreshed and recomputed ILU preconditioners deteriorate in the progress of nonlinear iterations. When BiCGSTAB stagnates the computed inexact Newton step is not a good descent direction and causes failures in the backtracking strategy.

Since recomputing the preconditioner should be avoided, the discretized problem was hard to be solved with the Update strategy too and required the use of different options from those seen in the previous benchmarks. The updates (4.5) of the preconditioner were generated by tridiagonal approximations $\tilde{E}_k$ for E_k in (4.6), i.e.

$$\tilde{\Delta}_k = \text{band}(\Delta_k, 1, 1), \quad \tilde{E}_k = \text{band}(\tilde{Z}^T \tilde{\Delta}_k \tilde{W}, 1, 1).$$

Then, in order to avoid recomputing the preconditioner after LI_max= 400 linear iterations, we proceeded updating the preconditioner with our strategy but inhibited the computation of a new reference preconditioner P_{seed}.

		Freeze			Update			
Re	Time_P	NI	LI	Time	NI	LI	NJ	Time
200	288	23	3250	431	19	1677	1	725
250	288	21	3133	423	17	1691	1	711
300	288	–	–	–	32	5167	1	1465
350	288	–	–	–	35	5745	1	1524

TABLE 5.6

Driven cavity problem, $n = 22500$. Test results: number of nonlinear and linear iterations, number of Jacobian evaluations.

Several tests cases were considered varying the dimension m and the Reynolds number Re. The `Recomp` and `Refresh` strategies performed poorly, as expected, while both the `Freeze` and the `Update` strategies resulted competitive. The good performance of the `Freeze` strategy clearly depends on the large value η_{max} which avoids pointless accuracy in solving the linear systems, i.e. the oversolving phenomenon. When both the `Freeze` and the `Update` strategies succeed, the values `NI` and `LI` are typically in favour of the `Update` strategy while the lowest timings are achieved by the `Freeze` strategy. This fact depends on the computational overhead needed to form the tridiagonal approximations $\tilde{\Delta}_k$ and $\tilde{E}_k$ and factorize $\tilde{E}_k$ in the `Update` strategy.

In Table 5.6 we reports results obtained using $m = 150$ and Reynolds number $Re = 200, 250, 300, 350$. The `Recomp` and `Refresh` procedures are not displayed as they failed in solving all tests. The levels of fill-in `FL_LU`, `FL_ZW` are $2 \cdot 10^{-3}$ and $4 \cdot 10^{-2}$ in the `Freeze` and `Update` strategies respectively. Note that the linear systems become more difficult to be solved as the Reynolds number increases and the `Freeze` strategy fails in two tests. On the other hand, the `Update` strategy produces good preconditioners and recovers the failures of the `Freeze` strategy. Finally, the overall cost of the procedures confirm the above considerations: in runs successfully solved by both the `Freeze` and `Update` strategies, freezing gains on updating in terms of execution times.

6. Summary of the runs. To summarize the runs presented and visualize the overall performances of the `Freeze`, `Recomp`, `Refresh` and `Update` preconditioning strategies, we employ the *performance profile* approach; see [21]. In this approach, when m solvers are compared on a test set, the performance of each solver in the solution of a test is measured by the ratio of its computational effort and the best computational effort by any solver on this test. Specifically, for each test t solved by the solver s, let $Q_{s,t}$ denote the computational effort required by the solver s to solve the test t. Moreover, let $\underline{Q}_{s,t}$ be the computational effort of the solver that results to be the most efficient in the solution of test t. Then, the ratio

$$q_{s,t} = \frac{Q_{s,t}}{\underline{Q}_{s,t}},$$

measures the performance on test t by solver s with respect to the best performance among all the solvers on such test. Clearly, $q_{s,t} \geq 1$ and $q_{s,t} = 1$ means that the solver s is the most effective in solving the test t over all the solvers. Then, the performance profile of solver s is defined as

$$\pi_s(\tau) = \frac{no.\ of\ tests\ s.t.\ q_{s,t} \leq \tau}{total\ no.\ of\ tests}, \qquad \tau \geq 1.$$

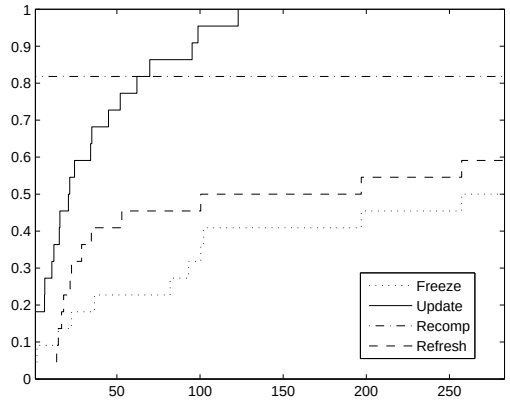

FIG. 6.1. *Performance profile in terms of BiCGSTAB iterations*

Assume that a parameter $q_M > q_{s,t}$ for all s, t is chosen. Then, if the solver s fails in solving test t, $q_{s,t}$ is set to q_M. Note that $q_{s,t} = q_M$ if and only if solver s does not solve test t. As a result of this convention $\pi_s(q_M) = 1$ and $\lim_{\tau \to q_M} \pi_s(\tau)$ is the probability that the solver solves a problem. Moreover, the performance profile flattens for $\tau \in [\bar{\tau}, q_M]$ for some $\bar{\tau} < q_M$. Then, if $\pi_s(\tau)$ is plotted in $[0, \bar{\tau}]$, the value of $\lim_{\tau \to q_M^-} \pi_s(\tau)$ can be readily seen in the performance profile's plot and the right side of the plot gives the percentage of the test problems that are successfully solved by the solver. On the other hand, the left side of the plot gives the percentage of test problems for which the solver is the fastest.

Here we use two different quantities to measure the computational effort of each strategy: the number `LI` of BiCGSTAB iterations performed and the execution time `Time` needed to solve each test. In Figures 6.1 and 6.2 we plot the profiles of the codes according to the two performance measures adopted. From these figures, it can be readily seen that the `Update` approach results to be the more robust as it solved all the tests. Considering Figure 6.1, as expected the `Recomp` strategy results to be the most effective in terms of linear iterations while our `Update` strategy outperforms the `Freeze` and `Refresh` approaches. Focusing on the overall computational time, Figure 6.2 shows that the proposed `Update` strategy is the most efficient in the solution of 73% of the tests. The remaining tests succesfully solved by the `Update` strategy require a computational effort that is at most two times the effort required by the best solver.

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.

[2] S. Bellavia, C. Cartis, N. I. M. Gould, B. Morini, Ph. L. Toint, *Convergence of a Regularized Euclidean Residual Algorithm for Nonlinear Least-Squares*, SIAM J. Numer. Anal., 48 (2010), pp. 1-29.

[3] 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.

[4] S. Bellavia, B. Morini, *Subspace trust-region methods for large bound-constrained nonlinear*

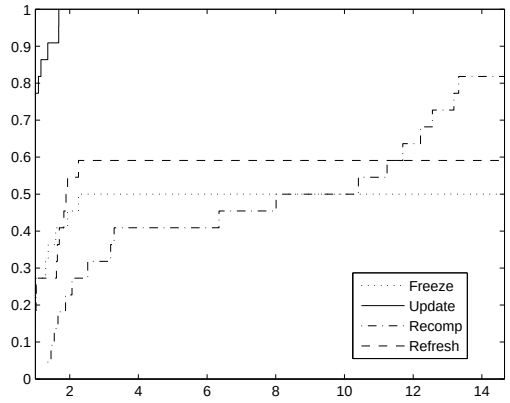

FIG. 6.2. *Performance profile in terms of execution time*

equations, SIAM J. Numer. Anal., 44 (2006), pp. 1535-1555.

[5] M. Benzi, D. Bertaccini, *Approximate inverse preconditioning for shifted linear systems*, BIT, 43 (2003), pp. 231–244.

[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] D. Bertaccini, *Efficient preconditioning for sequences of parametric complex symmetric linear systems*, ETNA, 18 (2004), pp. 49–64.

[10] D. Bertaccini, F. Sgallari, *Updating preconditioners for nonlinear deblurring and denoising image restoration*, Appl. Numer. Math., 60 (2010), pp. 994-1006.

[11] D. Bertaccini, G. H. Golub, S. Serra-Capizzano, *Spectral analysis of a preconditioned iterative method for the convection-diffusion equation*, SIAM J. Matr. Anal. Appl., 29 (2007), pp. 260–278.

[12] D. Bertaccini, M. K. Ng, *Band-Toeplitz preconditioned GMRES iterations for time-dependent PDEs*, BIT, 43 (2003), pp. 901-914.

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

[14] A. Björck, *Numerical methods for least squares problems*, SIAM, 1996.

[15] I.D.L. Bogles, J.D. Perkins, *A new sparsity preserving Quasi-Newton update for solving nonlinear equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 621-630.

[16] P. N. Brown, *A local convergence theory for combined inexact-Newton/finite-difference projection methods*, SIAM J. Numer. Anal., 24 (1987), pp. 407–434.

[17] P.N. Brown, Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.

[18] P.N. Brown, H. F. Walker, R. Wasyk, C. S. Woodward, *On using approximate finite-differences in matrix-free Newton-Krylov methods*, SIAM J. Numer. Anal., 46 (2008), pp. 1892-1911.

[19] J.K. Cullum, M. Tůma, *Matrix-free preconditioning using partial matrix estimation*, BIT, 46 (2006), pp. 711–729.

[20] R.S. Dembo, S.C. Eisenstat, T. Steihaug,*Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[21] E.D. Dolan, J.J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming 91 (2002), 201-213.

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

[23] J. Duintjer Tebbens, M. Tůma, *Preconditioner Updates for Solving Sequences of Linear Systems in Matrix-free Environment*, Numer. Linear Algebra Appl., DOI 10.1002/nla.695.

[24] A.R. Curtis, M.J.D. Powell, J.K. Reid, *On the estimation of sparse Jacobian matrices*, J. Inst.

Maths. Applics., 13 (1974), pp. 117-119.
[25] S.C. Eisenstat, H.F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.
[26] S.C. Eisenstat, H.F. Walker, *Choosing the forcing term in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.
[27] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers in Applied Mathematics, SIAM, 1995.
[28] 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.
[29] L. Luksan, *Inexact trust region method for large sparse systems of nonlinear equations*, J. Optim. Theory Appl., 81 (1994), pp. 569–591.
[30] G. Meurant, *On the incomplete Cholesky decomposition of a class of perturbed matrices*, SIAM J. Sci. Comput., 23, 2001, pp. 419–429.
[31] J. L. Morales, J. Nocedal, *Automatic preconditioning by limited memory Quasi-Newton updating*, SIAM J. Opt., 10, 2000, pp. 1079–1096.
[32] R. Nabben, *Decay rates of the inverse of nonsymmetric tridiagonal and band matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 820837.
[33] J. Nocedal, S.J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.
[34] 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.
[35] 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.
[36] M. Pernice, H.F. Walker, *NITSOL: a new iterative solver for nonlinear systems*, SIAM Journal Sci Comput., 19 (1998), pp. 302-318.
[37] Y. Saad, *Iterative methods for sparse linear systems*, 2nd ed., SIAM (2003).
[38] A.C.N. van Duin, *Scalable parallel preconditioning with the sparse approximate inverse of triangular systems*, SIAM J. Matr. Anal. Appl., 20 (1999), pp. 987-1006.
[39] C.J. van Duijn, J.M. de Graaf, *Large time behaviour of solutions of the porous medium equation with convection*, J. Differential Equations, 84 (1990), 183-203.