Electronic Transactions on Numerical Analysis.
Volume 23, pp. 76-87, 2006.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

QUASI-NEWTON PRECONDITIONERS FOR THE INEXACT NEWTON METHOD*

L. BERGAMASCHI†, R. BRU‡, A. MARTÍNEZ§, AND M. PUTTI†

Abstract. In this paper preconditioners for solving the linear systems of the Newton method in each nonlinear iteration are studied. In particular, we define a sequence of preconditioners built by means of Broyden-type rank-one updates. Optimality conditions are derived which guarantee that the preconditioned matrices are not far from the identity in a matrix norm. Some notes on the implementation of the corresponding inexact Newton method are given and some numerical results on two model problems illustrate the application of the proposed preconditioners.

Key words. Quasi-Newton method, Krylov iterations, updating preconditioners, inexact Newton method

AMS subject classifications. 65F10, 65H10, 15A12

1. Introduction.

Newton's method for the solution of systems of nonlinear equations

$$\boldsymbol{F}(\boldsymbol{x}) = 0 \tag{1.1}$$

with $\boldsymbol{F} : \mathcal{R}^n \to \mathcal{R}^n$ can be described as follows. If each component of $\boldsymbol{F}$, f_i, is differentiable, the Jacobian matrix $J(\boldsymbol{x})$ is defined by:

$$J(\boldsymbol{x})_{ij} = \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}).$$

Assuming that J is available, nonsingular, and "easy" to compute, Newton's method can be defined by the iteration

$$\left\{ \begin{array}{rcl} J(\boldsymbol{x}_k)\boldsymbol{s}_k & = & -\boldsymbol{F}(\boldsymbol{x}_k), \\ \boldsymbol{x}_{k+1} & = & \boldsymbol{x}_k + \boldsymbol{s}_k. \end{array} \right. \tag{1.2}$$

When J is large and sparse, e.g., for problems arising from the discretization of a nonlinear PDE, preconditioned Krylov based iterative schemes can be employed for the solution of the linear system, so that two nested iterative procedures need to be implemented. To avoid oversolving, i.e., excessive and essentially useless iterations of the inner scheme, it is crucial to employ an "inexact" technique [5]. This approach tries to control the number of linear iterations by allowing the accuracy of the linear solver to vary across nonlinear iterations [7].

Another crucial issue for the reduction of total linear iterations is to use efficient preconditioning techniques. In general, ILU-type preconditioners [14], [17] can be employed and calculated at every nonlinear iteration. Techniques for selectively evaluating a preconditioner P may be developed to save on the cost of the calculation of the preconditioner. Note that the two phases where efficiency can be mostly improved are the cost of the linear system solution (thus including the number of iterations) and the cost of preconditioner evaluation.

*Received November 2, 2005. Accepted for publication January 24, 2006. Recommended by C. Brezinski.

†Department of Mathematical Methods and Models for Scientific Applications, University of Padova, Italy, ({berga,putti}@dmsa.unipd.it). Work partially supported by the Italian MIUR project "Numerical Models for Multiphase Flow and Deformation in Porous Media".

‡ Instituto de Matemática Multidisciplinar, Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, Spain, (rbru@mat.upv.es). Work supported by the Spanish DGI (FEDER) grant MTM2004-02998, Generalitat Valenciana project GRUPOS03/062 and the research programme of the Univ. Politécnica of Valencia. Part of the work of this author was carried out during his visit to the Department of Mathematical Methods and Models for Scientific Applications of the Univ. of Padova.

§Department of Pure and Applied Mathematics, University of Padova, Italy, (acalomar@math.unipd.it). Work supported by the research fellowship "Parallel implementations of exponential integrators forODEs/PDEs".

In this paper we are mainly concerned with the efficient preconditioning of the linear system. The "optimal" preconditioner P would minimize the constant C of:

$$\|I - PJ(\boldsymbol{x}_k)\| \leq C. \tag{1.3}$$

Note that, ideally one would like C to be as small as possible or even to tend to zero as $k \to \infty$. This requires that information from the nonlinear iterative scheme be taken into account in the evaluation of P.

Standard preconditioners, such as ILU, are completely unaware of the underlying linearization process. Actually, often the Jacobian matrix tends to become more ill-conditioned as k grows, giving rise to increased numerical inaccuracies in the calculation of P.

The approach proposed in this paper is to solve system (1.2) with an iterative Krylov subspace method, starting with either ILU(0) [14] or AINV [2] preconditioners, computed from the initial Jacobian, and to update this preconditioner using a rank one sum. A sequence of preconditioners P_k can thus be defined by imposing the secant condition, as used in the implementation of quasi-Newton methods [6]. We choose to work with the Broyden update as described for instance in [9], and analyze the theoretical properties of the preconditioner and the numerical behavior of the resulting scheme.

Our approach is in the framework of the work of J. M. Martínez in [12], [13], where computing the preconditioner of choice (possibly even the null matrix) at every Newton step and then enriching it with a low-rank update is suggested. In these papers the author is mainly concerned with convergence of the Inexact Newton method while we focus on the optimality of the preconditioner at each nonlinear iteration. In addition we solve the Newtonian linear systems using an arbitrary iterative method with the stopping criterion of [5], while in Martinez's procedure this criterion is not used and the first iterate of the iterative linear solver should be constructed using the preconditioner. Morales and Nocedal in [15] proposed a rank-two update of the previously computed preconditioner in the acceleration of the Conjugate Gradient (CG) method. The vectors involved in this update do not come from the Newton iteration but they are generated using information from the CG iteration. Another study on updating factorized preconditioners can be found in [19].

The paper is organized as follows: Section 2 defines the preconditioner P_k. In Section 3, we prove that we can make the quantity $\|I - P_k J(\boldsymbol{x}_k)\|$ as small as possible. Section 4 gives the main lines of the implementation of the preconditioner application. Section 5 reports some numerical results on two model problems. Finally, Section 6 gives some concluding remarks.

2. Preconditioning by Broyden update. The idea is to start with a preconditioner $P = B_0^{-1}$ for J_0. If the preconditioner is in the form of a sparse approximate inverse, then B_0^{-1} is known explicitly. Otherwise, if $B_0 = \tilde{L}\tilde{U}$ is an incomplete LU factorization, we only can compute B_0^{-1} times a vector by solving two triangular sparse linear systems.

The proposed approach tries to evaluate $P_{k+1} = B_{k+1}^{-1}$ at every Newton iteration by adding to the previous preconditioner a rank one update. Let us write

$$B_{k+1} = B_k + \boldsymbol{u}\boldsymbol{v}^T. \tag{2.1}$$

As in the Broyden scheme, B_{k+1} must satisfy the secant condition

$$B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k, \tag{2.2}$$

where $\boldsymbol{y}_k = \boldsymbol{F}_{k+1} - \boldsymbol{F}_k$. To uniquely determine $\boldsymbol{u}, \boldsymbol{v}$, we also impose that B_{k+1} be the closest matrix to B_k in the Frobenius norm among all the matrices satisfying (2.2),

$$B_{k+1} = \underset{B: B\boldsymbol{s}_k = \boldsymbol{y}_k}{\text{argmin}} \|B - B_k\|. \tag{2.3}$$

Combining the secant condition (2.2) with (2.1), since $B_k\boldsymbol{s}_k + \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{s}_k = \boldsymbol{y}_k$, we readily obtain:

$$\boldsymbol{u} = \frac{\boldsymbol{y}_k - B_k\boldsymbol{s}_k}{\boldsymbol{v}^T\boldsymbol{s}_k}.$$

Condition (2.3) is satisfied by choosing

$$\boldsymbol{v} = \frac{\boldsymbol{s}_k}{\|\boldsymbol{s}_k\|}.$$

Using $\boldsymbol{u}$ and $\boldsymbol{v}$ thus defined, for every matrix B satisfying the secant equation we can write

$$B_{k+1} = B_k + \frac{B\boldsymbol{s}_k - B_k\boldsymbol{s}_k}{\boldsymbol{s}_k^T\boldsymbol{s}_k}\boldsymbol{s}_k^T$$

from which, taking Frobenius norms, we have that

$$\|B_{k+1} - B_k\| = \left\|\frac{B\boldsymbol{s}_k - B_k\boldsymbol{s}_k}{\boldsymbol{s}_k^T\boldsymbol{s}_k}\boldsymbol{s}_k^T\right\| \le \|B - B_k\| \cdot \left\|\frac{\boldsymbol{s}_k\boldsymbol{s}_k^T}{\boldsymbol{s}_k^T\boldsymbol{s}_k}\right\| = \|B - B_k\|,$$

and thus (2.3) holds. The expression for B_{k+1} is then written as

$$B_{k+1} = B_k + \frac{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)\,\boldsymbol{s}_k^T}{\boldsymbol{s}_k^T\boldsymbol{s}_k}. \tag{2.4}$$

The final preconditioner $P_{k+1} = B_{k+1}^{-1}$ can be obtained by inversion using the Sherman-Morrison formula to yield:

$$B_{k+1}^{-1} = B_k^{-1} - \frac{\left(B_k^{-1}\boldsymbol{y}_k - \boldsymbol{s}_k\right)\boldsymbol{s}_k^T B_k^{-1}}{\boldsymbol{s}_k^T B_k^{-1}\boldsymbol{y}_k}. \tag{2.5}$$

In the next section we will prove that $\|I - B_k^{-1}J(\boldsymbol{x}_k)\|$ can be made as small as possible by suitable choices of the initial guess $\boldsymbol{x}_0$ and the initial preconditioner B_0. Note that this makes our preconditioner almost ideal in the sense of (1.3).

REMARK 2.1. *The definition of our preconditioner and the convergence analysis that follows is similar to the corresponding developments of Broyden's method, as reported in [9]. The main difference is that in our case the new update is calculated by Newton iteration using the real and not the approximate Jacobian matrix. Thus in all the following steps we need to take into account that* $\boldsymbol{y}_k - B_k\boldsymbol{s}_k \ne \boldsymbol{F}_{k+1}$.

3. Convergence analysis. We will make the *standard assumptions* on $\boldsymbol{F}$.

1. Equation (1.1) has a solution $\boldsymbol{x}^*$.
2. $J(\boldsymbol{x}) : \Omega \to \mathcal{R}^{n\times n}$ is Lipschitz continuous with Lipschitz constant γ.
3. $J(\boldsymbol{x}^*)$ is nonsingular.

We will now bound the norm of the difference between B_{k+1} and $J(\boldsymbol{x}_{k+1})$ in terms of the errors in the current and new iterates. It is convenient to indicate with the subscript c every vector or matrix referring to the current iterate and with $+$ every quantity referring to the new iterate $k+1$. Following this notation Newton's method can be stated as

$$\begin{aligned} J_c\boldsymbol{s} &= -\boldsymbol{F}_c, \\ \boldsymbol{x}_+ &= \boldsymbol{x}_c + \boldsymbol{s}. \end{aligned} \tag{3.1}$$

ETNA
Kent State University
etna@mcs.kent.edu

We define the error vectors

$$\boldsymbol{e}_+ = x^* - x_+, \qquad \boldsymbol{e}_c = x^* - x_c$$

and the error matrices

$$E_+ = B_+ - J(\boldsymbol{x}^*), \quad E_c = B_c - J(\boldsymbol{x}^*).$$

The next lemma is the analog of Theorem 7.2.2 in [9]. It states that the sequence $\{B_k\}$ remains close to $J(\boldsymbol{x}^*)$ as $\boldsymbol{x}_k$ approaches $\boldsymbol{x}^*$.

LEMMA 3.1. *Let the standard assumptions hold. Then*

$$\|E_+\| \leq \|E_c\| + \gamma \frac{(\|\boldsymbol{e}_c\| + \|\boldsymbol{e}_+\|)}{2}.$$

Proof. Since

$$(B_c - J_c)\boldsymbol{s} - \boldsymbol{F}_c = B_c\boldsymbol{s} = J(\boldsymbol{x}^*)\boldsymbol{s} + E_c\boldsymbol{s},$$

we have

$$\begin{aligned} E_c\boldsymbol{s} &= (B_c - J_c)\boldsymbol{s} - J(\boldsymbol{x}^*)\boldsymbol{s} - \boldsymbol{F}_c \\ &= (B_c - J_c)\boldsymbol{s} - \boldsymbol{F}_+ + (\boldsymbol{F}_+ - \boldsymbol{F}_c - J(\boldsymbol{x}^*)\boldsymbol{s}) \\ &= (B_c - J_c)\boldsymbol{s} - \boldsymbol{F}_+ + \int_0^1 (J(x_c + t\boldsymbol{s}) - J(\boldsymbol{x}^*))\, dt \\ &= (B_c - J_c)\boldsymbol{s} - \boldsymbol{F}_+ + \Delta_c\boldsymbol{s}, \end{aligned} \tag{3.2}$$

where $\Delta_c = \int_0^1 (J(x_c + t\boldsymbol{s}) - J(\boldsymbol{x}^*))\, dt$. From (2.4) we can write

$$E_+ = B_+ - J(\boldsymbol{x}^*) = B_c - J(\boldsymbol{x}^*) + \frac{(\boldsymbol{F}_+ - \boldsymbol{F}_c - B_c\boldsymbol{s})\, \boldsymbol{s}^T}{\boldsymbol{s}^T\boldsymbol{s}}. \tag{3.3}$$

Then, setting $P_s = \dfrac{\boldsymbol{s}\boldsymbol{s}^T}{\boldsymbol{s}^T\boldsymbol{s}}$ and substituting (3.2) into (3.3) we obtain

$$\begin{aligned} E_+ &= E_c + (B_c - J_c)P_s - E_cP_s + \Delta_cP_s - (\boldsymbol{F}_c + B_c\boldsymbol{s})\, \frac{\boldsymbol{s}^T}{\boldsymbol{s}^T\boldsymbol{s}} \\ &= E_c\,(I - P_s) + (B_c - J_c)P_s + \Delta_cP_s - (\boldsymbol{F}_c + B_c\boldsymbol{s})\, \frac{\boldsymbol{s}^T}{\boldsymbol{s}^T\boldsymbol{s}} \\ &= E_c\,(I - P_s) + \Delta_cP_s. \end{aligned} \tag{3.4}$$

Using the standard assumptions and taking norms, since $\|P_s\| = 1$ we have

$$\begin{aligned} \|E_+\| &\leq \|E_c\| + \|\Delta_c\| \\ &\leq \|E_c\| + \gamma \frac{(\|\boldsymbol{e}_c\| + \|\boldsymbol{e}_+\|)}{2}. \qquad \square \end{aligned} \tag{3.5}$$

The next Lemma 3.2 and Theorem 3.3 prove that if the initial Newton point $\boldsymbol{x}_0$ is sufficiently near the solution, and B_0 sufficiently near $J(\boldsymbol{x}^*)$, then the sequence $\{B_k^{-1}\}$ is well defined. The proof of Lemma 3.2 is not given since it is analogous to that of Theorem 7.2.3 in [9].

ETNA
Kent State University
etna@mcs.kent.edu

LEMMA 3.2. *Let the standard assumptions hold. Fix* $\delta_1 > 0$. *Then there are* δ, δ_B *such that if* $\|\boldsymbol{e}_0\| < \delta$ *and* $\|E_0\| < \delta_B$ *then* $\|E_k\| < \delta_1$, $\forall k > 0$.

THEOREM 3.3. *Let the standard assumptions hold. Define* $\alpha = \|J(\boldsymbol{x}^*)^{-1}\|$. *Fix* $0 < \delta_1 < \dfrac{1}{\alpha}$. *Then there exist* δ *and* δ_B *such that if* $\|\boldsymbol{e}_0\| < \delta$ *and* $\|E_0\| < \delta_B$ *then*

$$\|B_+^{-1}\| < \frac{\alpha}{1-\delta_1\alpha}. \tag{3.6}$$

Proof. From Lemma 3.2

$$\begin{aligned}
\|B_+^{-1}\| &= \|B_+^{-1} - J(\boldsymbol{x}^*)^{-1} + J(\boldsymbol{x}^*)^{-1}\| \\
&\le \|B_+^{-1} - J(\boldsymbol{x}^*)^{-1}\| + \|J(\boldsymbol{x}^*)^{-1}\| \\
&= \|B_+^{-1}\left(B_+ - J(\boldsymbol{x}^*)\right) J(\boldsymbol{x}^*)^{-1}\| + \|J(\boldsymbol{x}^*)^{-1}\| \\
&= \|B_+^{-1} E_+ J(\boldsymbol{x}^*)^{-1}\| + \|J(\boldsymbol{x}^*)^{-1}\| \\
&< \|B_+^{-1}\|\delta_1\alpha + \alpha. \qquad (3.7)
\end{aligned}$$

Rewriting (3.7) we have

$$\|B_+^{-1}\|\left(1-\delta_1\alpha\right) < \alpha$$

from which the thesis, by the hypothesis $\delta_1\alpha < 1$. □

We now prove that we can make $\|I - B_k^{-1}J(\boldsymbol{x}_k)\|$ as small as possible. To this end we need the following two Lemmas which bound the difference between B_k and $J(\boldsymbol{x}_k)$.

LEMMA 3.4. *Let the standard assumptions hold. Then setting*

$$E'_+ = B_+ - J_+, \quad \textit{and} \quad E'_c = B_c - J_c,$$

we have

$$\|E'_+\| \le \|E'_c\| + \gamma\frac{3}{2}\left(\|\boldsymbol{e}_c\| + \|\boldsymbol{e}_+\|\right).$$

Proof. Since $E'_+ = E_+ + J(\boldsymbol{x}^*) - J_+ = E_+ + \Delta J_+$ and $E'_c = E_c + J(\boldsymbol{x}^*) - J_c = E_c + \Delta J_c$, eq. (3.4) becomes

$$\begin{aligned}
E'_+ &= \Delta J_+ + (E'_c - \Delta J_c)(I - P_s) + \Delta_c P_s \\
&= E'_c(I - P_s) - \Delta J_c(I - P_s) + \Delta J_+ + \Delta_c P_s. \qquad (3.8)
\end{aligned}$$

Now taking norms, and using the standard assumptions,

$$\|E'_+\| \le \|E'_c\| + \gamma\|\boldsymbol{e}_c\| + \gamma\|\boldsymbol{e}_+\| + \gamma\frac{\|\boldsymbol{e}_c\| + \|\boldsymbol{e}_+\|}{2}$$

and the thesis holds. □

LEMMA 3.5. *Let the standard assumptions hold. Fix* $\delta'_1 > 0$. *Then there are* δ, δ_B *such that if* $\|\boldsymbol{e}_0\| < \delta$ *and* $\|E'_0\| < \delta'_B$ *then* $\|E'_k\| < \delta'_1$, $\forall k > 0$.

Proof. Analogous to that of Lemma 3.2. □

The next theorem will establish a bound of $\|I - B_+^{-1}J_+\|$ in terms of $\|J(\boldsymbol{x}^*)^{-1}\|$.

ETNA
Kent State University
etna@mcs.kent.edu

THEOREM 3.6. *Let the standard assumptions hold. Fix* $0 < \delta_1 < \dfrac{1}{\alpha}$, $\delta_1' > 0$. *Then there are* δ, δ_B, δ_B' *such that if* $\|\boldsymbol{e}_0\| < \delta$, $\|E_0\| < \delta_B$ *and* $\|E_0'\| < \delta_B'$ *then*

$$\|I - B_+^{-1} J_+\| < \frac{\delta_1' \alpha}{1 - \delta_1 \alpha}.$$

Proof. Let δ, δ_B, δ_B' be as in Lemmas 3.2 and 3.5. From Lemma 3.5 and Theorem 3.3 we have

$$\|I - B_+^{-1} J_+\| = \|B_+^{-1} (B_+ - J_+)\| \le \|B_+^{-1}\| \|E_+'\| < \frac{\delta_1' \alpha}{1 - \delta_1 \alpha}. \qquad \square \tag{3.9}$$

4. Notes on implementation. In this section we give the main lines of the implementation of the product of our preconditioner times a vector, which is needed when using a preconditioned Krylov method.

At a certain nonlinear iteration level, k, and given a vector $\boldsymbol{z}_k^{(l)}$, we want to compute

$$\boldsymbol{c} = B_k^{-1} \boldsymbol{z}_k^{(l)}. \tag{4.1}$$

Here with superscript l we indicate the linear iteration index. For $k \ge 0$, B_{k+1} is given inductively by (2.5):

$$B_{k+1}^{-1} = B_k^{-1} - \frac{\left(B_k^{-1} \boldsymbol{y}_k - \boldsymbol{s}_k\right) \boldsymbol{s}_k^T B_k^{-1}}{\boldsymbol{s}_k^T B_k^{-1} \boldsymbol{y}_k}. \tag{4.2}$$

If we set $\boldsymbol{v}_k = \dfrac{\boldsymbol{s}_k}{\|\boldsymbol{s}_k\|}$, $\boldsymbol{u}_k = \dfrac{\boldsymbol{y}_k - B_k \boldsymbol{s}_k}{\|\boldsymbol{s}_k\|}$ and $\boldsymbol{w}_k = \dfrac{B_k^{-1} \boldsymbol{u}_k}{1 + \boldsymbol{v}_k^T B_k^{-1} \boldsymbol{u}_k}$, then

$$\begin{aligned} B_k^{-1} &= \left(I - \boldsymbol{w}_{k-1} \boldsymbol{v}_{k-1}^T\right) B_{k-1}^{-1} \\ &= \left(I - \boldsymbol{w}_{k-1} \boldsymbol{v}_{k-1}^T\right) \left(I - \boldsymbol{w}_{k-2} \boldsymbol{v}_{k-2}^T\right) \cdots \left(I - \boldsymbol{w}_0 \boldsymbol{v}_0^T\right) B_0^{-1}. \end{aligned} \tag{4.3}$$

At the initial nonlinear iteration $k = 0$, under the hypothesis that B_0^{-1} (or B_0) is explicitly known, we simply have

$$\boldsymbol{c} = B_0^{-1} \boldsymbol{z}_0^{(l)}.$$

If $k = 1$,

$$\boldsymbol{c} = B_1^{-1} \boldsymbol{z}_1^{(l)} = \left(I - \boldsymbol{w}_0 \boldsymbol{v}_0^T\right) B_0^{-1} \boldsymbol{z}_1^{(l)}.$$

Before doing this we have to compute $\boldsymbol{w}_0 = \dfrac{B_0^{-1} \boldsymbol{u}_0}{1 + \boldsymbol{v}_0^T B_0^{-1} \boldsymbol{u}_0}$.

If $k > 1$, let us suppose that $\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{w}_i$ are known, $i = 0, \ldots, k-2$. Then to be able to compute $\boldsymbol{c} = B_k^{-1} \boldsymbol{z}_k^{(l)}$, before starting the linear system solution we have to evaluate $\boldsymbol{v}_{k-1}, \boldsymbol{u}_{k-1}$ and $\boldsymbol{w}_{k-1} = \dfrac{\boldsymbol{u}_{k-1}'}{1 + \boldsymbol{v}_{k-1}^T \boldsymbol{u}_{k-1}'}$ where $\boldsymbol{u}_{k-1}' = B_{k-1}^{-1} \boldsymbol{u}_{k-1}$ can be evaluated using (4.3) at the price of an application of B_0^{-1}, $k-1$ dot products and $k-1$ daxpy operations. Now we are ready to compute

$$\boldsymbol{c} = \left(I - \boldsymbol{w}_{k-1} \boldsymbol{v}_{k-1}^T\right) \left(I - \boldsymbol{w}_{k-2} \boldsymbol{v}_{k-2}^T\right) \cdots \left(I - \boldsymbol{w}_0 \boldsymbol{v}_0^T\right) B_0^{-1} \boldsymbol{z}_k^{(l)}$$

at the price of an application of B_0^{-1}, k dot products and k daxpy operations.

Note that the updating of the preconditioner just described, being based on scalar products and daxpy operations, is well suited to parallelization.

ETNA
Kent State University
etna@mcs.kent.edu

4.1. The Newton-Broyden algorithm. The implementation of the Inexact Newton method requires the definition of a stopping criterion for the linear solver (1.2) based on the nonlinear residual. Following [5], we stop the linear iteration as soon as the following test is satisfied

$$\|J(\boldsymbol{x}_k)\boldsymbol{s}_k + \boldsymbol{F}(\boldsymbol{x}_k)\| \leq \eta_k \|\boldsymbol{F}(\boldsymbol{x}_k)\|. \tag{4.4}$$

Superlinear or even quadratic convergence of the Inexact Newton method can be achieved by properly setting the sequence $\{\eta_k\}$.

We can now write the Newton-Broyden Algorithm as follows:

NEWTON-BROYDEN (NB) ALGORITHM
Input: $\boldsymbol{x}_0, \boldsymbol{F}, nlmax$, tol

- WHILE $\|\boldsymbol{F}(\boldsymbol{x}_k)\| >$ tol AND $k < nlmax$ DO
    1. Compute $B_0(B_0^{-1})$ approximating $J_0(J_0^{-1})$; $k = 0$
    2. IF $k > 0$ THEN update B_k^{-1} from B_{k-1}^{-1}.
    3. Solve $J(\boldsymbol{x}_k)\boldsymbol{s}_k = -\boldsymbol{F}(\boldsymbol{x}_k)$ by a Krylov method with preconditioner B_k^{-1} and tolerance η_k.
    4. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$
    5. $k = k + 1$
- END WHILE

Since the actual computation of the preconditioner is never performed explicitly, the stated algorithm suffers from two main drawbacks, namely increasing costs of memory for w_k and v_k and of preconditioner application. These drawbacks are common to many iterative schemes, such as for example sparse (Limited Memory) Broyden implementations [16], GMRES [18] and Arnoldi method for eigenvalue problems [11]. Several options can be devised to alleviate these difficulties, all based on variations of a restart procedure, by which the scheme is reset after a chosen number of iterations. Our implementation is described in the following section.

4.2. Restart. If the number of nonlinear iterations is high (e.g. more than ten iterations), the application of Broyden preconditioner may be too expensive to be counterbalanced by a reduction in the iteration number. To this aim we define $k_{\max}$ the maximum number of rank one corrections we allow. After $k_{\max}$ nonlinear iterations the previous computed $\boldsymbol{w}_i, \boldsymbol{v}_i, i = 1, \ldots, k_{\max}$ are discarded, and a new preconditioner B_0^{-1} is computed.

RESTARTED NEWTON-BROYDEN (RNB) ALGORITHM
Input: $\boldsymbol{x}_0, \boldsymbol{F}, k_{\max}, nlmax$, tol

- Compute $B_0(B_0^{-1})$ approximating $J_0(J_0^{-1})$; $k = 0$, restart=TRUE
- WHILE $\|\boldsymbol{F}(\boldsymbol{x}_k)\| >$ tol AND $k < nlmax$ DO
    1. IF (NOT restart) THEN update B_k^{-1} from B_{k-1}^{-1}; restart = FALSE.
    2. Solve $J(\boldsymbol{x}_k)\boldsymbol{s}_k = -\boldsymbol{F}(\boldsymbol{x}_k)$ by a Krylov method with preconditioner B_k^{-1} and tolerance η_k.
    3. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$
    4. $k = k + 1$
    5. IF k MOD $k_{\max} = 0$ THEN
        - restart = TRUE; compute $B_k(B_k^{-1})$ approximating $J_k(J_k^{-1})$
- END WHILE

ETNA
Kent State University
etna@mcs.kent.edu

REMARK 4.1. *Our preconditioner can be viewed as a particular case of the one defined in [12] as $B_k = C(\boldsymbol{x}_k) + A_k$, where we set*

$$C(\boldsymbol{x}_k) = \begin{cases} ILU/AINV(J_k), & \textit{if } k \text{ MOD } k_{\max} = 0, \\ 0, & \textit{otherwise}. \end{cases}$$

5. Numerical results. In this section we report the numerical performance of our preconditioners in solving two nonlinear test problems of large size. As initial preconditioners (B_0), we consider either the incomplete Cholesky factorization ILU(0) [14] or the approximate inverse preconditioner AINV [2], [3]. Note that, in the first case, B_0 is known and the application of B_0^{-1} results in two triangular sparse linear system solutions. On the other hand, B_0^{-1} is explicitly provided by AINV, as the product of two sparse triangular factors, and hence its application needs two matrix-vector products. Our aim is to give experimental evidence that the Broyden correction, implemented as explained in the previous sections, produces an acceleration of the convergence of the iterative solver, independently of the initial preconditioner of choice.

All the numerical experiments were performed on a Compaq ES40 equipped with an alpha-processor "ev6.8" at 833Mhz, 4.5 GB of core memory, and 16 MB of secondary cache. The CPU times are measured in seconds. The solution of systems (2) is performed by the BiCGstab iterative method [20] because in our examples the Jacobian is symmetric but is not guaranteed to be positive definite. We stop the iteration whenever the exit test (4.4) with constant $\eta_k = 10^{-4}$ is fulfilled.

5.1. Example 1: the Bratu problem. We consider a generalization of the classical discrete Bratu problem [8]:

$$A\boldsymbol{u} = \lambda D(u), \qquad D = \text{diag}(\exp(u_1), \ldots, \exp(u_n))$$

where A is a symmetric positive definite matrix arising from a 2d or 3d discretization of the diffusion equation on the unitary domain:

$$-\nabla\left(K\nabla u\right) = f \tag{5.1}$$

with variable diffusion coefficient K, and λ is a real parameter. We used $\lambda = 1$ and $\boldsymbol{x}_0 = (0.1, \ldots, 0.1)^T$. We consider matrices arising from discretization employing Finite Elements (FE) and Mixed Finite Elements (MFE).

5.1.1. 2d MFE discretization. Matrix A has 28600 rows and 142204 nonzero elements. It arises from a Mixed Finite Element discretization of the diffusion operator (with constant $K \equiv 1$) in two spatial dimensions on triangles of uniform sizes.

Tables 5.1 and 5.2 report the results of the nonlinear convergence, giving the number of nonlinear iterations, the cumulative number of linear iterations, the total CPU time and the CPU time needed to evaluate the preconditioner (computation of B_0^{-1} when needed plus all the updates). When the Broyden acceleration is used, we also provide the values of $k_{\max}$. With ILU(0)(J_0) we mean that the preconditioner is computed once and for all at the beginning of the nonlinear iteration, while ILU(0)(J_k) indicates that the incomplete factorization is accomplished at each nonlinear iteration.

In both cases (B_0 = ILU(0)/AINV) the RNB algorithm produces an acceleration in terms of both number of iterations and CPU time. The simple NB algorithm (without restart) appears to be less efficient. This is not surprising since the results of Theorem 3 only hold when x_0 is near the solution. In the NB algorithm, B_0 is computed only once, when the iterate is still far from x^*, while in the RNB approach, the "initial" preconditioner is computed every $k_{\max}$ iterations, thus taking advance from the nonlinear convergence.

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 5.1
Results on MFE matrix with B_0 = ILU(0).

preconditioner	$k_{\max}$	nlit	iter	CPU	
				tot	precond
ILU(0)(J_0)	–	7	851	11.59	0.01
ILU(0)(J_k)	–	7	754	8.65	0.04
RNB-ILU(0)	1	7	442	6.51	0.08
RNB-ILU(0)	2	7	470	6.56	0.06
RNB-ILU(0)	3	7	501	6.93	0.06
RNB-ILU(0)	5	7	529	7.09	0.06
NB-ILU(0)	∞	6	515	9.67	0.06

TABLE 5.2
Results on MFE matrix with B_0 = AINV(0.1).

preconditioner	$k_{\max}$	nlit	iter	CPU	
				tot	precond
AINV(0.1) (J_0)	–	7	882	18.35	0.17
AINV(0.1) (J_k)	–	7	908	19.70	1.27
RNB-AINV(0.1)	1	8	574	14.62	1.57
RNB-AINV(0.1)	2	7	517	13.07	0.81
RNB-AINV(0.1)	3	7	647	16.10	0.63
RNB-AINV(0.1)	4	7	502	12.61	0.44
RNB-AINV(0.1)	5	7	561	14.08	0.44
NB-AINV(0.1)	∞	7	655	17.15	0.26

5.2. Example 2. We consider here a slight modification of the Bratu problem which leads to an increasingly ill-conditioned Jacobian matrix. This is easily accomplished by taking a negative value for the parameter λ. We choose to work with $\lambda = -1$ and the same initial vector as in the previous example.

TABLE 5.3
Results on MFE matrix with B_0 = ILU(0).

preconditioner	$k_{\max}$	nlit	iter	CPU	
				tot	precond
ILU(0) (J_0)	–	14	849	15.61	0.01
ILU(0) (J_k)	–	14	597	7.04	0.08
RNB-ILU(0)	1	14	436	6.51	0.15
RNB-ILU(0)	2	14	415	6.56	0.12
RNB-ILU(0)	3	14	419	6.93	0.11
RNB-ILU(0)	4	14	407	7.09	0.11
NB-ILU(0)	∞	15	595	14.86	0.17

5.2.1. 2d MFE discretization. We report in Table 5.3 the results of the various algorithms described in Section 4, compared with the Inexact Newton method preconditioned with the simple ILU(0) computed at the initial iteration. Table 5.4 provides the same outcome as Table 5.3 but employing the AINV preconditioner. In the latter case we also provide the value of the drop tolerance used in the test.

Analyzing Table 5.3 we notice that the Broyden acceleration produces a mild reduction

ETNA
Kent State University
etna@mcs.kent.edu

of the CPU time as compared with the computation of ILU(0) at every iteration. The optimal $k_{\max}$ value is 1. The simple Newton-Broyden algorithm (i.e. with no restart) does not give any improvement in terms of CPU time and iteration number. In Table 5.4 the results relative to the AINV preconditioner with drop tolerance of 0.05 enhances the efficiency of the proposed preconditioner. Here the optimal $k_{\max}$ value is larger than 1. Using $k_{\max} > 1$ produces a saving in the CPU time needed for computing the costly AINV preconditioner.

TABLE 5.4
Results on MFE matrix with B_0 = AINV(0.05).

preconditioner	$k_{\max}$	nlit	iter	CPU tot	CPU precond
AINV(0.05) (J_0)	–	14	852	18.99	0.19
AINV(0.05) (J_k)	–	14	545	29.96	5.42
RNB-AINV(0.05)	1	14	332	18.34	5.06
RNB-AINV(0.05)	2	14	347	17.72	2.91
RNB-AINV(0.05)	3	14	348	16.13	2.06
RNB-AINV(0.05)	4	14	333	15.57	1.67
RNB-AINV(0.05)	5	14	353	16.27	1.26
NB-AINV(0.05)	∞	14	630	18.73	0.34

5.2.2. 3d FE discretization. Matrix A is the result of a 3D Finite Element discretization of (5.1) with coefficient K varying in space and using tetrahedral elements. It has 268 515 rows and 3 926 823 nonzero elements.

For this test case we also compare the Broyden acceleration with the result obtained by selectively computing the preconditioner (both ILU(0) and AINV) every $k_{\max}$ iterations; see Tables 5.5 and 5.6. In Figure 5.1 we report the linear iterations employed at each nonlinear

TABLE 5.5
Results on the 3DFE matrix with B_0 = ILU(0).

preconditioner	$k_{\max}$	nlit	iter	CPU tot	CPU precond
ILU(0) (J_0)	–	17	1880	1125.4	0.4
ILU(0) (J_k)	–	16	354	193.4	6.4
ILU(0) (J_k)	2	16	353	194.1	6.2
RNB-ILU(0)	1	16	230	137.7	9.4
RNB-ILU(0)	2	16	232	137.8	6.3
RNB-ILU(0)	3	16	250	149.7	5.5
RNB-ILU(0)	4	16	251	150.8	4.8
NB-ILU(0)	∞	17	1132	826.4	4.7

iteration for the ILU(0) and AINV(0.03) preconditioners computed at each nonlinear iteration and using the Broyden acceleration for a couple of $k_{\max}$ values. As is clear from figure 5.1, in this problem the Jacobian matrix becomes more ill-conditioned during the nonlinear iteration. For this reason, using the AINV preconditioner with a fixed drop tolerance value $(= 0.03)$, yields a sequence of preconditioners with increasing number of nonzero elements and consequent increasing evaluation cost. This explains the CPU times for preconditioner evaluation in Table 5.6. The RNB preconditioner provides a substantial reduction of the linear iterations especially near the solution of the nonlinear problem, as expected by the findings of the theory in Section 2.

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 5.6
Results on the 3DFE matrix with B_0 = AINV(0.03).

preconditioner	$k_{\max}$	nlit	iter	CPU	
				tot	precond
AINV(0.03)(J_0)	–	17	1991	755.3	3.5
AINV(0.03)(J_k)	–	16	682	470.4	151.9
AINV(0.03)(J_k)	4	16	705	355.6	34.0
RNB-AINV(0.03)	1	16	418	356.7	154.5
RNB-AINV(0.03)	4	16	448	262.1	37.4
RNB-AINV(0.03)	7	16	508	281.9	29.1
RNB-AINV(0.03)	10	16	556	289.9	17.7
NB-AINV(0.03)	∞	17	1186	591.0	5.2

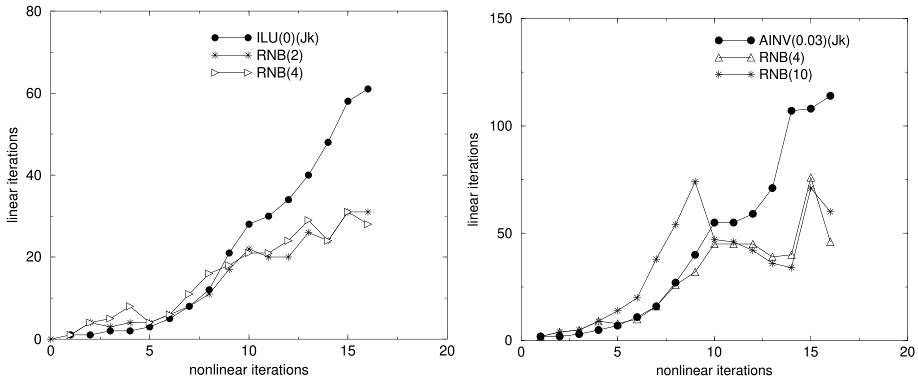

FIG. 5.1. *Number of linear iterations vs nonlinear iteration index in the solution of Example 2 discretized by 3D FE. The initial preconditioner is ILU(0) (left) and AINV(0.03) (right).*

The initial preconditioner does not remain a good preconditioner for the subsequent iterations as confirmed by the first row in Tables 5.5 and 5.6. The Broyden correction produces a reduction in the number of linear iterations and CPU time with respect to the ILU (AINV) preconditioner for small values of $k_{\max}$. Also, the proposed preconditioner is more efficient than simply computing selectively ILU or AINV. Comparing for example the results in the third and fifth rows of Table 5.6 we see that the Broyden acceleration reduces the linear iteration from 705 to 448 (35% reduction) and the CPU time from 355.6 to 262.1 seconds (26% reduction). Comparable reductions hold for the ILU(0) preconditioner.

6. Conclusions. We have proposed a family of preconditioners based on the Broyden secant update formula, with the aim of accelerating the convergence properties of a given preconditioner during the nonlinear process. During the Newton iteration, starting from a preconditioner B_0^{-1} approximating the inverse of the initial Jacobian matrix, a sequence of preconditioners B_k^{-1} is defined, by taking into account information of the previous nonlinear iterations. The developed theoretical analysis proves that the sequence satisfies $||I - B_k^{-1} J_k(x_k)|| \leq C$, with C that can be made arbitrarily small, depending on $\boldsymbol{x}_0$ and B_0.

The application of the proposed preconditioner may be memory and time consuming, especially when k is large. However, compared with the standard procedure of computing the preconditioner of choice at every Newton iteration, a limited memory variant of the proposed approach provides an improvement of the performance. Experimental results on a number of large test problems, show substantial savings in terms of both iteration number and computing time.

The algorithm described in this paper seems to be particularly appropriate for parallel implementation. First, the improvement of the proposed preconditioner is important especially when the computation of the initial preconditioner is costly, which is a common situation when solving systems in a parallel framework. Second, the implementation of the Quasi-Newton corrections as described in Section 4, being made up of a number of `daxpys` and scalar products, can be parallelized in a straightforward manner. If the computation and application of the initial preconditioner can be efficiently parallelized (which is the case for approximate inverse preconditioners; see [1] for AINV or [4], [10] for other approaches), the overall algorithm is completely parallel.

Acknowledgements. We would like to thank the referees and J. M. Martínez for their suggestions that have improved the presentation of this paper.

REFERENCES

[1] M. BENZI, J. MARÍN, AND M. TŮMA, *A two-level parallel preconditioner based on sparse approximate inverses*, in D. R. Kincaid and A. C. Elster, eds., *Iterative Methods in Scientific Computation IV*, Vol. 5 of IMACS Series in Computational and Applied Mathematics, New Brunswick, New Jersey, USA, 1999, International Assoc. for Mathematics and Computers in Simulation, pp. 167–178.

[2] M. BENZI AND M. TŮMA, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[3] ———, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.

[4] L. BERGAMASCHI AND A. MARTÍNEZ, *Parallel acceleration of Krylov solvers by factorized approximate inverse preconditioners*, in M. Daydè et al., eds., VECPAR 2004, Lecture Notes in Comput. Sci., No. 3402, Springer, Heidelberg, 2005, pp. 623–636.

[5] R. S. DEMBO, S. C. EISENSTAT AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[6] J. E. DENNIS, JR. AND J. J. MORÉ, *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19 (1977), pp. 46–89.

[7] S. C. EISENSTAT AND H. F. WALKER, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[8] D. R. FOKKEMA, G. L. G. SLEJIPEN AND H. A. VAN DER VORST, *Accelerated inexact Newton schemes for large systems of nolinear equations*, SIAM J. Sci. Comput., 19 (1997), pp. 657–674.

[9] C. T. KELLEY, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[10] L. YU. KOLOTILINA AND A. YU. YEREMIN, *Factorized sparse approximate inverse preconditionings I. Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[11] R. B. LEHOUCQ AND D. C. SORENSEN, *Deflation techniques for an implicit restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.

[12] J. M. MARTÍNEZ, *A theory of secant preconditioners*, Math. Comp., 60 (1993), pp. 681–698.

[13] ———, *An extension of the theory of secant preconditioners*, J. Comput. Appl. Math., 60 (1995), pp. 115–125.

[14] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[15] J. L. MORALES AND J. NOCEDAL, *Automatic preconditioning by limited memory Quasi-Newton updating*, SIAM J. Optim., 10 (2000), pp. 1079–1096.

[16] S. G. NASH AND J. NOCEDAL, *A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization*, SIAM J. Optim., 1 (1991), pp. 358–372.

[17] Y. SAAD, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[18] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[19] J. D. TEBBENS AND M. TŮMA, *Preconditioner updates for solving sequences of large and sparse nonsymmetric linear systems*, Technical Report, Institute of Computer Science, Academy of Sciences of the Czech Republic, V-940, 2005.

[20] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.