LOW-RANK UPDATE OF PRECONDITIONERS FOR THE INEXACT NEWTON METHOD WITH SPD JACOBIAN

L. BERGAMASCHI† R. BRU,*‡ AND A. MARTíNEZ†

Abstract. In this note preconditioners for the Conjugate Gradient method are studied to solve the Newton system with symmetric positive definite Jacobian. In particular, we define a sequence of preconditioners built by means of BFGS rank-two updates. Reasonable 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 a number of model problems illustrate the efficiency of the proposed preconditioners.

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

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

1. Introduction. In this note we are mainly concerned with the efficient preconditioning of the linear system arising in the Newton iteration for the solution of a general system of nonlinear equations: $\boldsymbol{F}(\boldsymbol{x}) = 0$, which is usually written as

$$\left\{ \begin{array}{lcl} J(\boldsymbol{x}_k)\boldsymbol{s}_k & = & -\boldsymbol{F}(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1} & = & \boldsymbol{x}_k + \boldsymbol{s}_k \end{array} \right. . \tag{1.1}$$

In a number of applications such as e. g., unconstrained optimization, or discretization of nonlinear PDEs with Picard linearization, J is symmetric positive definite (SPD). When in addition J is large and sparse, the preconditioned Conjugate Gradient (PCG) method can be employed for the solution of the linear system, so that two nested iterative procedures need to be implemented, the outer iteration formed by the Newton steps and the inner iterations of the PCG method.

Since in each Newton step a new system has solved, we are dealing with the construction of a sequence of preconditioners P_k which are "optimal" in the sense that they would minimize the constant C of:

$$\|I - P_k J(\boldsymbol{x}_k)\| \leq C. \tag{1.2}$$

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_k. In particular, the secant condition should be satisfied by those preconditioners.

Then, instead to construct a new preconditioner of the corresponding Jacobian in each outer iteration we propose to update an initial preconditioner using information of the Newton method. The approach proposed in this note follows the line of a previous work [3] where the authors have shown to accelerate the Inexact Newton method by a Broyden-type rank-one of a preconditioner of choice. It is worth to mention that this technique can be considered in the framework of the papers [14, 15] by J. M. Martínez. A related preconditioner has been used by Morales and Nocedal in [17].

In this note our aim is to solve the SPD system (1.1) with the PCG method, starting with an initial preconditioner, in our case either IC(0) [16] or AINV [1] preconditioners, computed from the initial Jacobian, and to update this preconditioner using low-rank matrices. A sequence of SPD 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 BFGS update as described for instance in [11], and analyze the theoretical properties of the preconditioner and

*Corresponding author.

†Department of Mathematical Methods and Models for Scientific Applications, University of Padova, Italy, ({berga,acalomar}@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 MTM2007-64477.

the numerical behavior of the resulting scheme. This rank-two update has been used recently in tuning eigensolvers (see [21]), in addition to a rank-one modification (see [8] and the references therein).

The paper is organized as follows: §2 defines the preconditioner P_k, in §3 we prove that we can make the quantity $\|I-P_kJ(\boldsymbol{x}_k)\|$ as small as possible, §4 gives the main lines of the implementation of the preconditioner application. §5 reports some numerical results on a model problem and onto unconstrained optimization problems, and finally §6 gives some concluding remarks.

2. Updating the preconditioners by a low-rank updates. The idea is to start with a preconditioner $P_0 = B_0^{-1} \approx J_0^{-1}$. If the preconditioner is under the form of a sparse approximate inverse P_0 is known explicitly, otherwise, if $B_0 = \tilde{L}\tilde{U}$ is an incomplete LU factorization, we only can compute P_0 times a vector by solving two triangular sparse linear systems.

Let us define $\boldsymbol{y}_k = \boldsymbol{F}(\boldsymbol{x}_{k+1}) - \boldsymbol{F}(\boldsymbol{x}_k)$ and recall that $\boldsymbol{s}_k$ is the solution of the kth Newton system. Following [12] we can develop a similar recurrence formula for the preconditioner as

$$P_{k+1} = \left(I - \frac{\boldsymbol{s}_k\boldsymbol{y}_k^T}{\boldsymbol{s}_k^T\boldsymbol{y}_k}\right) P_k \left(I - \frac{\boldsymbol{y}_k\boldsymbol{s}_k^T}{\boldsymbol{s}_k^T\boldsymbol{y}_k}\right) + \frac{\boldsymbol{s}_k\boldsymbol{s}_k^T}{\boldsymbol{s}_k^T\boldsymbol{y}_k}. \tag{2.1}$$

If the Jacobian matrices are SPD and P_0 is so, then also P_k is SPD under the condition $\boldsymbol{s}_k^T\boldsymbol{y}_k > 0$ (see Lemma 4.1.1 in [12]).

It can also be easily proved that the sequence of matrices just defined satisfy the secant condition

$$P_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k$$

In the next section we will prove that $\|I - P_kJ(\boldsymbol{x}_k)\|$ can be made as small as possible by suitable choices of the initial guess $\boldsymbol{x}_0$ and the initial preconditioner P_0. Note that this makes our preconditioner almost ideal in the sense of (1.2).

3. Convergence analysis. From now on we will also denote the preconditioner matrix and its inverse by B_k^{-1} and B_k, respectively. Let us denote with Ω an open subset of $\mathcal{R}^n$, we will make the following *standard assumptions* on $\boldsymbol{F}$.

Standard Assumptions:

1. Equation $\boldsymbol{F}(\boldsymbol{x}) = 0$ 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.

Moreover, we assume in the sequel that our preconditioner accelerates the Newton systems in the framework of the Inexact Newton methods [5]. We therefore stop the linear iteration as soon as the following test is satisfied

$$\|J(\boldsymbol{x}_k)\boldsymbol{s}_k + \boldsymbol{F}(\boldsymbol{x}_k)\| \le \eta_k\|\boldsymbol{F}(\boldsymbol{x}_k)\|. \tag{3.1}$$

The sequence $\{\eta\}$ will be chosen such that $\eta_k = O(\|\boldsymbol{F}(\boldsymbol{x}_k)\|)$. This will guarantee quadratic convergence of the Inexact Newton method and as a consequence,the following result.

PROPOSITION 3.1. *Let the standard assumptions hold, and define $\boldsymbol{e}_k = \boldsymbol{x}^* - \boldsymbol{x}_k$. There exist $\delta > 0$ and $r < 1$ such that if $\|\boldsymbol{e}_0\| < \delta$ then $\|\boldsymbol{e}_{k+1}\| \le r\|\boldsymbol{e}_k\|$ for every k.*

As in observed in Lemma 4.1.4 of [12], we may assume $J(\boldsymbol{x}^*) = I$ for the convergence analysis.

It is now convenient to indicate with the subscript c every vector or matrix referring to the current iterate and with the subscript $+$ every quantity referring to the new iterate $k+1$. Following this notation Newton's method can be stated as

$$\begin{aligned} J_c\boldsymbol{s} &= -\boldsymbol{F}_c \\ \boldsymbol{x}_+ &= \boldsymbol{x}_c + \boldsymbol{s} \end{aligned} \tag{3.2}$$

We define the error vectors

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

and the error matrices

$$E_+ = B_+^{-1} - J(\boldsymbol{x}^*)^{-1} = B_+^{-1} - I, \qquad E_c = B_c^{-1} - J(\boldsymbol{x}^*)^{-1} = B_c^{-1} - I$$

where $B_+^{-1} = P_+$ and $B_c^{-1} = P_c$.

LEMMA 3.2. *Let the standard assumptions hold. Let* $\|\boldsymbol{e}_c\| \leq \delta_0 < \dfrac{1}{\gamma}$. *Then*

$$\|J_c^{-1} - I\| \leq \frac{\gamma \|\boldsymbol{e}_c\|}{1 - \gamma \delta_0}$$

Proof. Follows from Theorem 1.2.1 in [11], also known as Banach lemma. ∎

We now prove the following important Lemma which bounds the norm of E_+ in terms of the norm of E_c and $\boldsymbol{e}_c$. In addition, we prove that the sequence of preconditioners constructed with the equation (2.1) is SPD provided that P_0 is SPD. Some parts of the proof of this Lemma follow the lines of that of Lemma 4.1.5 in [12]. We report also the common parts for sake of completeness.

LEMMA 3.3. *Let the standard assumptions hold. If* B_c *is SPD and*

$$x_+ = x_c - J_c^{-1} \boldsymbol{F}_c$$

then there is δ_0 *such that if*

$$0 < \|\boldsymbol{x}_c - \boldsymbol{x}^*\| \leq \delta_0 \quad \textit{and} \qquad \|E_c\| \leq \delta_0$$

then $\boldsymbol{y}^T \boldsymbol{s} > 0$ *showing that* B_+ *is SPD. Moreover,*

$$E_+ = P_s E_c P_s + \Delta \tag{3.3}$$

with $P_s = I - \boldsymbol{w}\boldsymbol{w}^T$, $\boldsymbol{w} = \dfrac{\boldsymbol{s}}{\|\boldsymbol{s}\|}$ *and for some* $K > 0$, $\|\Delta\| \leq K\|\boldsymbol{e}_c\|$ *and*

$$\|E_+\| \leq \|E_c\| + K\|\boldsymbol{e}_c\|.$$

Proof. Let δ_0 be small enough so that $\boldsymbol{F}_c \neq 0$ if $\boldsymbol{x}_c \neq \boldsymbol{x}^*$. Then by the fundamental theorem of calculus

$$\boldsymbol{F}_c = \int_0^1 J(x^* + t\boldsymbol{e}_c)\boldsymbol{e}_c dt \; = \boldsymbol{e}_c + \Delta_1 \boldsymbol{e}_c = (I + \Delta_1)\boldsymbol{e}_c.$$

where

$$\Delta_1 = \int_0^1 \left(J(x^* + t\boldsymbol{e}_c) - I\right) dt \; .$$

From the Lipschitz condition, $\|\Delta_1\| \leq \gamma\|\boldsymbol{e}_c\|/2$. If we write

$$\boldsymbol{s} = -J_c^{-1}\boldsymbol{F}_c = (-I + I - J_c^{-1})(I + \Delta_1)\boldsymbol{e}_c$$

then, by Lemma 3.2,

$$\|\boldsymbol{e}_c\| \left(1 - \frac{\gamma\delta_0}{1 - \gamma\delta_0}\right)(1 - \gamma\delta_0/2) \leq \|\boldsymbol{s}\| \leq \|\boldsymbol{e}_c\| \left(1 + \frac{\gamma\delta_0}{1 - \gamma\delta_0}\right)(1 + \gamma\delta_0/2)$$

so that, if

$$\delta_0 < \frac{2-\sqrt{2}}{2\gamma} \tag{3.4}$$

then

$$1/2\|\boldsymbol{e}_c\| \leq \|\boldsymbol{s}\| \leq 2\|\boldsymbol{e}_c\|. \tag{3.5}$$

Now from the standard assumptions, and again from the fundamental theorem of calculus we have that

$$\boldsymbol{y} = \boldsymbol{F}_+ - \boldsymbol{F}_c = \int_0^1 J(x_c + t\boldsymbol{s})\boldsymbol{s}dt = \boldsymbol{s} + \int_0^1 (J(x_c + t\boldsymbol{s}) - I)\boldsymbol{s}dt = \boldsymbol{s} + \Delta_2\boldsymbol{s}, \tag{3.6}$$

where $\Delta_2 = \displaystyle\int_0^1 (J(x_c + t\boldsymbol{s})\boldsymbol{s} - I)\, dt$. From the standard assumptions and Proposition 3.1, $\|\Delta_2\| \leq \gamma\|\boldsymbol{e}_c\|$. Therefore, from (3.5)

$$\boldsymbol{y}^T\boldsymbol{s} = \boldsymbol{s}^T\boldsymbol{s} + (\Delta_2\boldsymbol{s})^T\boldsymbol{s} \geq \|\boldsymbol{s}\|^2(1 - \gamma\delta_0) > 0 \tag{3.7}$$

hence B_+^{-1} is SPD by (2.1). Let us write

$$\begin{aligned} \frac{\boldsymbol{s}\boldsymbol{y}^T}{\boldsymbol{y}^T\boldsymbol{s}} &= \frac{\boldsymbol{s}\boldsymbol{s}^T + \boldsymbol{s}(\Delta_2\boldsymbol{s})^T}{\boldsymbol{s}^T\boldsymbol{s} + (\Delta_2\boldsymbol{s})^T\boldsymbol{s}} = \frac{\boldsymbol{s}\boldsymbol{s}^T}{\boldsymbol{s}^T\boldsymbol{s}} - \Delta_3 \\ \frac{\boldsymbol{s}\boldsymbol{s}^T}{\boldsymbol{y}^T\boldsymbol{s}} &= \frac{\boldsymbol{s}\boldsymbol{s}^T}{\boldsymbol{s}^T\boldsymbol{s} + (\Delta_2\boldsymbol{s})^T\boldsymbol{s}} = \frac{\boldsymbol{s}\boldsymbol{s}^T}{\boldsymbol{s}^T\boldsymbol{s}} + \Delta_4 \end{aligned} \tag{3.8}$$

with $\|\Delta_3\| \leq \dfrac{4\gamma}{1-\delta_0\gamma}\|\boldsymbol{s}\| = C\|\boldsymbol{s}\|$ and $\|\Delta_4\| \leq \dfrac{2\gamma}{1-\delta_0\gamma}\|\boldsymbol{s}\| = \frac{C}{2}\|\boldsymbol{s}\|$. Then, from (2.1) we have

$$\begin{aligned} E_+ = B_+^{-1} - I &= (I - \boldsymbol{w}\boldsymbol{w}^T + \Delta_3)B_c^{-1}(I - \boldsymbol{w}\boldsymbol{w}^T + \Delta_3^T) + \boldsymbol{w}\boldsymbol{w}^T - I + \Delta_4 \\ &= (P_s + \Delta_3)B_c^{-1}(P_s + \Delta_3^T) - P_s + \Delta_4 \\ &= P_sB_c^{-1}P_s + \Delta_3B_c^{-1}(P_s + \Delta_3^T) + P_sB_c^{-1}\Delta_3^T + \Delta_4 - P_s \\ &= P_sE_cP_s + P_s^2 - P_s + \Delta \\ &= P_sE_cP_s + \Delta \end{aligned} \tag{3.9}$$

where we used the fact that P_s is a projector and set

$$\Delta = \Delta_3B_c^{-1}(P_s + \Delta_3^T) + P_sB_c^{-1}\Delta_3^T + \Delta_4 \ .$$

If (3.4) holds then $\delta_0 < \frac{1}{2\gamma} \leq 1/2$ and therefore

$$\begin{aligned} \|\Delta\| &\leq (1+\delta_0)\|\Delta_3\|(2 + C\|\boldsymbol{s}\|) + \frac{C}{2}\|\boldsymbol{s}\| \\ &\leq 3/2C\|\boldsymbol{s}\|(2 + C\|\boldsymbol{s}\|) + \frac{C}{2}\|\boldsymbol{s}\| \\ &= \frac{C}{2}\|\boldsymbol{s}\|\,(7 + 6C\delta_0) \\ &\leq C\|\boldsymbol{e}_c\|\,(7 + 6C\delta_0) \end{aligned} \tag{3.10}$$

Choosing δ_0 s. t. $6C\delta_0 \leq 1$ and setting $K = 8C$, the proof follows by the standard assumptions. □

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^{-1} and $J(\boldsymbol{x}_k)^{-1}$.

LEMMA 3.4. *Let the standard assumptions hold and* $\|\boldsymbol{e}_c\| \leq \delta$. *Then setting*

$$E'_+ = {B_+}^{-1} - J_+^{-1}, \quad \textit{and} \quad E'_c = {B_c}^{-1} - J_c^{-1}$$

we have, for some $K_1 > 0$,

$$\|E'_+\| \leq \|E'_c\| + K_1\|\boldsymbol{e}_c\|$$

Proof. Since $E'_+ = E_+ + I - J_+^{-1} = E_+ - \Delta J_+^{-1}$ and $E'_c = E_c + I - J_c^{-1} = E_c - \Delta J_c^{-1}$, and using equation (3.3) we have

$$\begin{aligned} E'_+ &= \Delta J_+^{-1} + P_s(E'_c + \Delta J_c^{-1})P_s + \Delta \qquad (3.11)\\ &= \Delta J_+^{-1} + P_s E'_c P_s + P_s \Delta J_c^{-1} P_s + \Delta. \end{aligned}$$

Now taking norms, and using Proposition 3.1 and Lemma 3.2,

$$\begin{aligned} \|E'_+\| &\leq \|E'_c\| + \|\Delta J_+\| + \|\Delta J_c\| + \|\Delta\| \qquad (3.12)\\ &\leq \|E'_c\| + \frac{\gamma\|\boldsymbol{e}_+\|}{1-\gamma\|\boldsymbol{e}_+\|} + \frac{\gamma\|\boldsymbol{e}_c\|}{1-\gamma\|\boldsymbol{e}_c\|} + K\|\boldsymbol{e}_c\| \\ &\leq \|E'_c\| + 2\gamma\frac{\|\boldsymbol{e}_c\|}{1-\gamma\delta} + K\|\boldsymbol{e}_c\| \qquad (3.13)\\ &= \|E'_c\| + \left(K + \frac{2\gamma}{1-\gamma\delta}\right)\|\boldsymbol{e}_c\| \end{aligned}$$

and the thesis holds, provided that $\delta \leq \frac{1}{2\gamma}$, with $K_1 = K + 4\gamma$.
□

The next result proves that if the initial Newton point $\boldsymbol{x}_0$ is sufficiently near to the solution, and B_0 sufficiently near to J_0, then E'_k can be made as small as desired.

LEMMA 3.5. *Let the standard assumptions hold. Fixed* $\varepsilon > 0$, *there are* δ_0, δ_B *such that if* $\|\boldsymbol{e}_0\| < \delta_0$ *and* $\|E'_0\| < \delta_B$ *then* $\|E'_k\| < \varepsilon$, $\forall k > 0$.

Proof. We show that $\|E'_k\| < \varepsilon$ by induction. We use Lemma 3.4. It is clear that $\|E'_1\| \leq \|E'_0\| + K_1\|\boldsymbol{e}_0\| \leq \delta_B + K_1\delta_0$. If we choose δ_0 such that $K_1\delta_0 < \varepsilon$ and set $\delta_B = \varepsilon - K_1\delta_0$ we have $\|E'_1\| < \varepsilon$. Let us suppose that $\|E'_k\| < \varepsilon$. Then, using Proposition 3.1, we have

$$\begin{aligned} \|E'_{k+1}\| &\leq \|E'_k\| + K_1\|\boldsymbol{e}_k\| \\ &\leq \|E'_k\| + K_1(r\|\boldsymbol{e}_{k-1}\|) \leq \cdots \leq \|E'_k\| + K_1(r^k\|\boldsymbol{e}_0\|) \leq \|E'_k\| + K_1(r^k\delta_0). \qquad (3.14) \end{aligned}$$

Reasoning recursively, we obtain

$$\|E'_k\| \leq \|E'_{k-1}\| + K_1(r^{k-1}\delta_0)$$

Finally

$$\|E'_{k+1}\| \leq \|E'_0\| + K_1\delta_0\sum_{i=0}^{k} r^i \leq \varepsilon + K_1\delta_0\sum_{i=0}^{k} r^i \leq \varepsilon + K_1\delta_0\frac{1}{1-r}.$$

Now choose δ_0 such that $K_1\delta_0\frac{1}{1-r} < \varepsilon$ and set $\delta_B = \varepsilon - K_1\delta_0\frac{1}{1-r}$ and the proof follows. □

The following theorem establish that $\|I - B_+^{-1}J_+\|$ can be made arbitrarily small provided that $\|\boldsymbol{e}_0\|$ and $\|E'_0\|$ are sufficiently small.

THEOREM 3.6. *Let the standard assumptions hold. Fixed* $\varepsilon_1 > 0$, *then there are* δ_0, δ_B *such that if* $\|\boldsymbol{e}_0\| < \delta_0$ *and* $\|E'_0\| < \delta_B$ *then*

$$\|I - B_+^{-1}J_+\| < \varepsilon_1.$$

Proof. Using Lemma 3.5 and the Lipschitz continuity of $J(\boldsymbol{x})$ we have:

$$\|I - B_+^{-1}J_+\| = \| \left(J_+^{-1} - B_+^{-1}\right) J_+\| \leq \|J_+\|\|E'_+\| \leq (1+\gamma\|\boldsymbol{e}_c\|)\,\varepsilon \leq (1+\gamma\delta_0)\varepsilon. \tag{3.15}$$

If $\delta_0 < \frac{1}{2\gamma}$ we can choose $\varepsilon = \frac{2}{3}\varepsilon_1$, and δ_0, δ_B as in the previous Lemma 3.5 so that the thesis holds.
□

4. Implementation of the BFGS update. 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. Throughout the section we write $P_k = B_k^{-1}$.

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

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

Here with superscript l we indicate the linear iteration index. Let us suppose to compute an initial preconditioner P_0. Then, at the initial nonlinear iteration $k = 0$, we simply have $\boldsymbol{c} = P_0 \boldsymbol{z}_0^{(l)}$. Applying P_k with $k > 0$ requires ad hoc procedures depending on the update formula.

For $k \geq 0$, P_{k+1} is given inductively by (2.1):

$$\begin{aligned} P_{k+1} &= \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{y}_k} + \left(I - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\boldsymbol{s}_k^T \boldsymbol{y}_k}\right) P_k \left(I - \frac{\boldsymbol{y}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{y}_k}\right) \\ &= R_k P_k R_k^T + A_k \end{aligned} \tag{4.2}$$

where

$$R_k = I - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\alpha_k}, \quad A_k = \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\alpha_k}, \qquad \alpha_k = \boldsymbol{s}_k^T \boldsymbol{y}_k.$$

Application of preconditioner P_k to the vector $\boldsymbol{z}_k^{(l)}$ can be performed at the price of $2k$ dot products and $2k$ daxpys as follows. If $k = 1$ then

$$\boldsymbol{c} = P_1 \boldsymbol{z}_1^{(l)} = \frac{\boldsymbol{s}_0 \boldsymbol{s}_0^T}{\boldsymbol{s}_0^T \boldsymbol{y}_0} \boldsymbol{z}_1^{(l)} + \left(I - \frac{\boldsymbol{s}_0 \boldsymbol{y}_0^T}{\boldsymbol{s}_0^T \boldsymbol{y}_0}\right) P_0 \left(I - \frac{\boldsymbol{y}_0 \boldsymbol{s}_0^T}{\boldsymbol{s}_0^T \boldsymbol{y}_0}\right) \boldsymbol{z}_1^{(l)}. \tag{4.3}$$

Computation of (4.3) requires the steps sketched in Figure 4. By recursively exploiting (2.1) it is

$$\begin{aligned} &a = \frac{\boldsymbol{s}_0^T \boldsymbol{z}_1^{(l)}}{\alpha_0} \\ &\boldsymbol{w} = \boldsymbol{z}_1^{(l)} - a\boldsymbol{y}_0 \\ &\boldsymbol{w} := P_0 \boldsymbol{w} \\ &b = \frac{\boldsymbol{y}_0^T \boldsymbol{w}}{\alpha_0} \\ &\boldsymbol{c} = (a-b)\boldsymbol{s}_0 + \boldsymbol{w} \end{aligned}$$

FIG. 4.1. *Computation of* $\boldsymbol{c} = P_1 \boldsymbol{z}_1^{(l)}$ *for the BFGS preconditioner.*

```
w = z_r^(l)
FOR r := k − 1  TO 0
      a_r = s_r^T w/α_r
      w := w − a_r y_r
END FOR
c = P_0 w
FOR r := 0  TO k − 1
      b = y_r^T c/α_r
      c := c + (a_r − b)s_r
END FOR
```

FIG. 4.2. *Computation of* $\boldsymbol{c} = P_k \boldsymbol{z}_k^{(l)}$ *for the BFGS preconditioner.*

easy to show that computation of $\boldsymbol{c} = P_k \boldsymbol{z}_k^{(l)}$ requires the steps listed in Figure 4.

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

4.1. The Newton-BFGS Algorithm. Following [5], the implementation of the Inexact Newton method requires the definition of a stopping criterion for the linear solver (3.1) based on the nonlinear residual. More precisely, we stop the linear iteration using the following test

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

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-BFGS Algorithm as follows:

NEWTON-BFGS (NBFGS) ALGORITHM
Input: $\boldsymbol{x}_0, \boldsymbol{F}, nlmax$, toll

- WHILE $\|\boldsymbol{F}(\boldsymbol{x}_k)\| > \text{toll}\ \|\boldsymbol{F}(\boldsymbol{x}_0)\|$ 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 (CG) 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

This kind of algorithm suffers from two main drawbacks, namely increasing costs of memory for w_k and c_k and of preconditioner application. Note that these drawbacks are common to many iterative schemes, such as for example sparse (Limited Memory) Broyden implementations [18], GMRES [19] and Arnoldi method for eigenvalue problems [13]. There are different ways to decrease these difficulties, all based on variations of a restart procedure, that is, the iteration scheme is reset after a fixed number of iterations.

4.2. Restart. If the number of nonlinear iterations is high (e.g. more than ten iterations), the application of BFGS preconditioner may be too heavy to be counterbalanced by a reduction in the iteration number. To this aim we define $k_{\max}$ the maximum number of rank two corrections we allow. When the nonlinear iteration counter k is larger than $k_{\max}$, the vectors $\boldsymbol{s}_i, \boldsymbol{y}_i,\ i = k - k_{\max}$ are substituted with the last computed $\boldsymbol{s}_k, \boldsymbol{y}_k$ and a new preconditioner B_0^{-1} is computed. Vectors $\{\boldsymbol{s}_i, \boldsymbol{y}_i\}$ are stored in a matrix V with n rows and $2 \times k_{\max}$ columns.

RESTARTED NEWTON-BFGS (RNBFGS) ALGORITHM
Input: $\boldsymbol{x}_0, \boldsymbol{F}, k_{\max}, nlmax$, toll

- Compute $B_0(B_0^{-1})$ approximating $J_0(J_0^{-1})$; $k = 0$
- WHILE $\|\boldsymbol{F}(\boldsymbol{x}_k)\| > \text{toll}\ \|\boldsymbol{F}(\boldsymbol{x}_0)\|$ AND $k < nlmax$ DO
  1. IF $k > 0$ THEN update B_k^{-1} from B_{k-1}^{-1} using the columns of V
  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. $j = k$ MOD $k_{\max}$; $V(*, 2j+1) := \boldsymbol{s}_k,\ V(*, 2j+2) := \boldsymbol{y}_k$,
  5. $k := k + 1$
  6. IF k MOD $k_{\max} = 0$ THEN compute $B_k(B_k^{-1})$ approximating $J_k(J_k^{-1})$
- END WHILE

5. Numerical Results. Here we give the numerical performance of our sequence of preconditioners in solving a class of nonlinear test problems of large size. As initial preconditioners (P_0), we consider either the incomplete Cholesky factorization IC(0) [16] or the approximate inverse preconditioner AINV [1, 2]. 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 pro-

vided by AINV, as the product of two sparse triangular factors, and hence its application needs two matrix-vector products.

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. In the solution of the systems 1.1 we employed the PCG iterative method [20] and stop the iteration whenever the exit test (4.4) with constant $\eta_k = 10^{-4}$ is fulfilled. The nonlinear iteration is stopped whenever $\|\boldsymbol{F}(x_k)\| \leq 10^{-8}\|\boldsymbol{F}(x_0)\|$ (i. e. toll $= 10^{-8}$). In the subsequent tables we always report the total number of linear iterations and CPU time (subdivided in total and that needed to compute and update the preconditioner). The number of nonlinear iterations is recalled in the caption since it is independent on the preconditioner selected.

5.1. Discrete Bratu problem. We consider the classical discrete Bratu problem [7]:

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

where A is an SPD matrix arising from a 2d or 3d discretization of the diffusion equation on a unitary domain, and λ is a real parameter. We used $\lambda = -1$ to ensure that the Jacobian is SPD too, and $\boldsymbol{x}_0 = (0.1, \ldots, 0.1)^T$. We consider matrices arising from Finite Difference (FD) discretization with different spatial dimensions.

2d FD discretization. In this test case the discretization step is $h = 1.25 \times 10^{-3}$ which leads to a problem of size $n = 640000$. As initial preconditioner we selected IC(0) and AINV with a very small drop tolerance (0.01) which produces a far more dense preconditioner. The results regarding IC(0), summarized in Table 5.1 and Figure 5.1 show that the RNBfgs approach produces an improvement in the number of iterations. The CPU time is also slightly reduced only in the $k_{\max} = 1$ case which turns out to be the optimal value. In Table 5.2 the results using AINV preconditioner are reported. Here the optimal $k_{\max}$ value is 3 with a significant improvement in terms of CPU time due to (a) the reduction of the iteration number and (b) the selective computation of the (costly) preconditioner. High values of $k_{\max}$ do not generally produce any improvement, in accordance with the findings of Theorem 3.5 which requires x_0 and P_0 to be sufficiently close to x^* and $J(x_0)^{-1}$, respectively. We also note that from Figure 5.1 it is evident that the benefit of the proposed approach grows as the Newton sequence approaches x^*.

TABLE 5.1
Results on 2dFD matrix with B_0 = IC(0). (nlit = 14)

preconditioner	$k_{\max}$	iter	CPU	
			tot	precond
IC(0) (J_k)	–	2388	1064	3.9
RNBfgs-IC(0)	1	1841	954	4.6
RNBfgs-IC(0)	3	1801	1134	2.2

TABLE 5.2
Results on 2dFD matrix with B_0 = AINV(0.01). (nlit = 14)

preconditioner	$k_{\max}$	iter	CPU	
			tot	precond
AINV(0.01) (J_k)	–	1387	2090	430
RNBfgs-AINV(0.01)	3	1037	1334	150
RNBfgs-AINV(0.01)	5	1111	1640	81
RNBfgs-AINV(0.01)	7	1241	1774	46
NBfgs-AINV(0.01)	∞	2441	2685	11

3d FD discretization. In this test case the discretization step is $h = 1.25 \times 10^{-2}$ which leads to a problem of size $n = 512000$. Summary results are provided in Tables 5.3 – 5.5 for different initial preconditioners, We also tried $B_0 = \mathrm{diag}(J(x_0))$ in order to show that even starting with

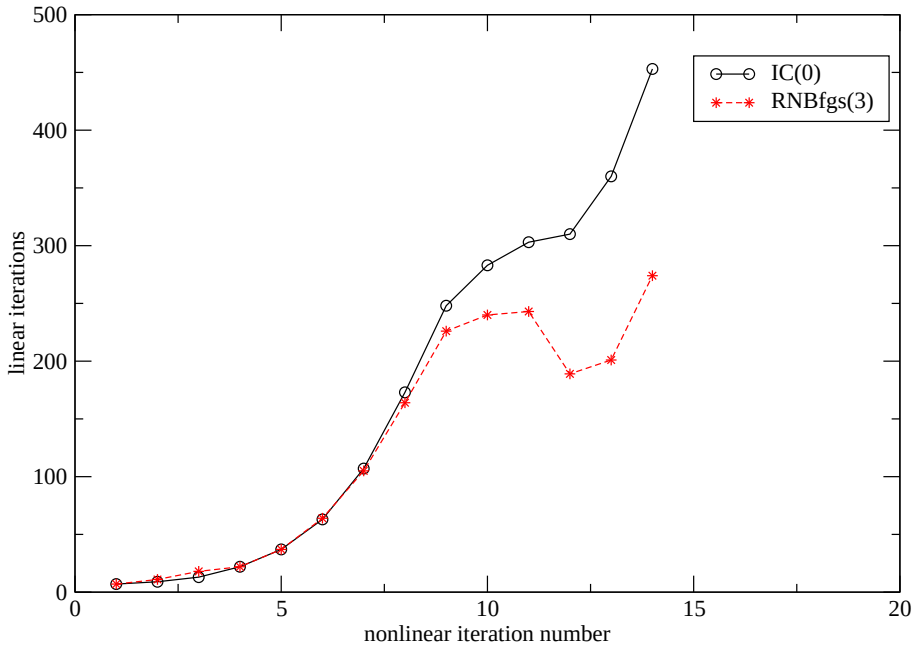

FIG. 5.1. *Number of linear iterations vs nonlinear iteration index in the solution of Example 2 discretized by 2D FD. The initial preconditioner is IC(0).*

a poor initial preconditioner, the rank-two update produces a reduction – though slight – of the iteration number (see Table 5.5, $k_{\max} = 1$).

TABLE 5.3
Results on 3dFD matrix with B_0 = IC(0). (nlit = 12)

preconditioner	$k_{\max}$	iter	CPU	
			tot	precond
IC(0) (J_k)	–	397	200	4.1
RNBfgs-IC(0)	1	281	162	4.8
MNBfgs-IC(0)	1	281	155	4.6
RNBfgs-IC(0)	2	284	175	2.8
RNBfgs-IC(0)	3	297	195	2.4
MNBfgs-IC(0)	3	287	181	2.6

TABLE 5.4
Results on 3dFD matrix with B_0 = AINV(0.01). (nlit = 12)

preconditioner	$k_{\max}$	iter	CPU	
			tot	precond
AINV(0.01) (J_k)	–	400	503	240
RNBfgs-AINV(0.01)	1	281	425	238
RNBfgs-AINV(0.01)	2	285	334	113
RNBfgs-AINV(0.01)	5	325	326	53
RNBfgs-AINV(0.01)	10	500	397	30

In view of the fact that the BFGS-based preconditioner is expected to work well as x_0 approaches the exact solution, we implement a Mixed strategy (MNBfgs) which consists in using the preconditioner of choice for the first iterations, namely until the initial residual is reduced by a fixed quantity (10^{-1} in our tests), and then switch to RNBfgs for the subsequent nonlinear

iterations. This variant is aimed at furtherly reducing the CPU time by avoiding the use of the more costly RNBfgs preconditioner when it is not expected to accelerate the PCG method. Some results with the MNBfgs are provided in Tables 5.3 and 5.5 for the IC(0) and diagonal initial preconditioners. MNBfgs, compared with RNBfgs with same $k_{\max}$, displays a slight CPU time reduction.

TABLE 5.5
Results on 3dFD matrix with B_0 = Jacobi. (nlit = 12)

preconditioner	$k_{\max}$	iter	CPU	
			tot	precond
Jacobi (J_k)	–	1021	235	0.8
RNBfgs-Jacobi	1	767	234	1.5
MNBfgs-Jacobi	1	767	220	1.2
RNBfgs-Jacobi	2	759	261	1.0
RNBfgs-Jacobi	3	773	302	0.8
RNBfgs-Jacobi	5	798	385	0.8

5.2. The modified PHI-2 problem. We now test our preconditioner correction in the solution of the following nonlinear problem

$$A\boldsymbol{u} = \lambda D(u), \qquad D = \mathrm{diag}(u_1^3, \ldots, u_n^3)$$

This test is a variant of the PHI-2 equation [10] to maintain the Jacobian SPD. Originally $D = \mathrm{diag}(1 + u_1^2, \ldots, 1 + u_n^2)$. Again A is the discretization matrix of the Laplacian operator. In table 5.6 we report the results of various choices of $k_{\max}$ and the initial preconditioner with A as in 2D discretization of the Bratu problem.

TABLE 5.6
Results on the PHI-2 problem. (nlit = 14)

initial precond.	$k_{\max}$	iter	CPU	
			tot	precond
AINV(0.01)	0	995	285.89	79.18
AINV(0.01)	1	793	256.04	82.46
AINV(0.01)	2	868	240.99	42.59
AINV(0.01)	5	965	255.99	22.83
AINV(0.1)	0	3004	212.74	7.56
AINV(0.1)	1	2420	195.29	7.57
IC(0)	0	1719	121.63	0.88
IC(0)	1	1382	114.26	1.05
IC(0)	2	1438	121.63	0.61

With every choice of the initial preconditioner, $k_{\max} > 0$ produces an improvement in both iteration number and CPU time with respect to the case $k_{\max} = 0$. In particular with the "dense" AINV(0.01), $k_{\max} = 2$ yields a 16% improving in the number of iteration and 13% in the CPU time. For the sparser AINV(0.1) and IC(0), $k_{\max} = 1$ reveals the optimal choice with a roughly 20% iteration spared.

5.3. Optimization problems. We now report some preliminary results of our preconditioner onto unconstrained optimization problems `SPARSINE` and `CRAGGLVY`, taken from CUTEr library [9] (available at `http://hsl.rl.ac.uk/cuter-www/problems.html`).

Problem: SPARSINE. The size of this problem is $n = 5000$ and 154108 is the number of nonzero elements. For this problem the initial IC(0) could not be computed so we report results of our update formula used to accelerate the Jacobi initial preconditioner. Table 5.7 gives evidence that the improvement in the number of iterations is important, irrespective of the $k_{\max}$ value.

TABLE 5.7
Results on SPARSINE problem with B_0 = Jacobi. (nlit = 8)

preconditioner	$k_{\max}$	iter	CPU	
			tot	precond
Jacobi (J_k)	–	73609	296	0.1
RNBfgs-Jacobi	2	22338	92	0.1
RNBfgs-Jacobi	5	22212	96	0.1
NBfgs-Jacobi	∞	22763	98	0.1

Problem: CRAGGLVY. For this problem, which has $n = 25000$ and 74998 nonzero elements, the conditioning of the Newton matrices is very good, so very few iterations proved sufficient to achieve convergence. In this case we can not expect big improvement from the BFGS correction. We selected AINV as initial preconditioner with varying (0.1 and 0.5) drop tolerance. and used the mixed approach. For the less sparse preconditioner (AINV(0.1)) the number of iterations is not reduced, however, for $k_{\max} = 2$ MNBFGS took 5.9 seconds vs the 7.0 seconds needed by simple AINV(0.1), as accounted by Table 5.8 This saving is again the result of the selective computation of the preconditioner. When the preconditioner is more sparse (AINV(0.5)) the MNBfgs approach yields also a reduction on the number of iterations.

TABLE 5.8
Results on CRAGGLVY3 problem with B_0 = AINV. (nlit = 15)

preconditioner	$k_{\max}$	iter	CPU	
			tot	precond
AINV(0.1) (J_k)	–	90	7.0	3.8
MNBfgs-AINV(0.1)	1	88	7.2	2.3
MNBfgs-AINV(0.1)	2	96	5.9	1.7
MNBfgs-AINV(0.1)	5	119	6.1	1.3
AINV(0.5) (J_k)	–	198	7.3	3.4
MNBfgs-AINV(0.5)	1	167	7.6	3.6
MNBfgs-AINV(0.5)	2	175	6.6	2.2
MNBfgs-AINV(0.5)	5	188	6.8	1.2

6. Parameter tuning. The sequence of preconditioners just developed is based on a proper selection of a preconditioner for the initial Jacobian and on the tuning of the parameter $k_{\max}$. In this note we are not concerned with the first task which is very much problem dependent. The aim of this work is to theoretically prove and experimentally give evidence that, independently of the efficiency of the initial P_0, the BFGS correction leads to an improvement, for a limited value of $k_{\max}$, of the iteration number and CPU time. An analysis of the results in the previous section show that:

1. The value of $k_{\max}$ should not be high compared to the expected number of nonlinear iterations. Large $k_{\max}$ values produce an important overhead of each Conjugate Gradient iteration, and at the same make the P_k sequence rely on an inaccurate initial preconditioner. The suggested interval in which to choose our parameter is then $[1, 5]$.
2. The optimal value of $k_{\max}$ may be furtherly refined taking into account the following characteristic of the problem at hand.
   (a) **Sparsity**. If the linearized matrix is very sparse, then the PCG iteration time is highly increased by moderate value of $k_{\max}$, so sparsity suggests to keep this value as low as possible.
   (b) **Complexity of initial preconditioner evaluation.** If the initial preconditioner is costly, it is convenient to compute it selectively, so to suggest larger values of $k_{\max}$.
   (c) **Distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}^*$.** If $\|\boldsymbol{e}_0\|$ is large, Theorem 3.6 no longer guarantees

an improvement of the preconditioner quality so that again small $k_{\max}$ values are to be preferred.

7. Conclusions. A rank-two update sequence of preconditioners has been proposed for accelerating the PCG method when it is used in the solution of the inner linear systems of the inexact Newton method. This approach follows the previous work [3] where the initial preconditioner was enriched by a Broyden type rank-one updated. In this work a sequence of preconditioner is defined, based on the BFGS rank-two update formula. It has been shown that the sequence of the preconditioners is SPD and that $||I - P_k J(x_k)||$ remains bounded if the initial vector guess x_0 and P_0 are close enough to the exact solution and to the inverse of the Jacobian $J^{-1}(x_0)$, respectively. Since the construction of the preconditioner sequence is memory and time consuming, a restarted algorithm (RNBfgs) has been implemented.

Our first numerical experiments onto unconstrained minimization problems show that this algorithm provides an improvement of the performance compared with the IC or AINV or even the diagonal as initial preconditioners, and this is particularly efficient for small values of the restart parameter $k_{\max}$. The proposed technique has a number of advantage on simply computing a preconditioner of $J(\boldsymbol{x}_k)$: (a) it reduces the number of iterations at least for small $k_{\max}$ values; and (b) it reduces the cost of forming the preconditioner.

As a consequence, we expect that this technique, together with the one developed in [3], could be particularly effective e. g., in the interior point (IP) solution of constrained optimization problems, where linearized saddle-point Newton systems are very ill-conditioned toward the solution and the cost of the preconditioner computation may be prohibitive (see [4]).

Finally, the update procedure is well suited to parallelization since it is based on scalar products and daxpy operations.

REFERENCES

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

[2] ———, *A comparative study of sparse approximate inverse preconditioners*, Applied Numerical Mathematics, 30 (1999), pp. 305–340.

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

[4] L. Bergamaschi, J. Gondzio, and G. Zilli. *Preconditioning indefinite systems in interior point methods for optimization.* Computational Opt. & Appls, 28(2):149–171, 2004.

[5] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Num. 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] D. R. Fokkema, G. L. G. Slejipen, and H. A. van der Vorst, *Accelerated Inexact Newton schemes for large systems of nonlinear equations*, SIAM J. Sci. Comput., 19 (1997), pp. 657–674.

[8] M. A. Freitag, and A. Spence, *Rayleigh Quotient iteration and simplified Jacobi-Davidson method with preconditioned iterative solves*, Tenth Copper Mountain Conference on Iterative Methods, http://grandmaster.colorado.edu/ copper/2008/SCWinners/Freitag.pdf, 2008.

[9] N. I. M. Gould, D. Orban, and P. L. Toint, *Cuter and sifdec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Softw., 29 (2003), pp. 373–394.

[10] P. R. Graves-Morris, Personal communication.

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

[12] C. T. Kelley, *Iterative methods for optimization*, vol. 18 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[13] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicit restarted Arnoldi iteration*, SIAM J. Matrix Anal., 17 (1996), pp. 789–821.

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

[15] ———, *An extension of the theory of secant preconditioners*, J. Comput. Appl. Math., 60 (1995), pp. 115–125. Linear/nonlinear iterative methods and verification of solution (Matsuyama, 1993).

[16] 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.

[17] J. L. Morales and J. Nocedal, *Automatic preconditioning by limited memory Quasi-Newton updating*, SIAM J. on Optimization, 10 (2000), pp. 1079–1096.

[18] 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.

[19] 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.

[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.

[21] F. Xue, and H. C. Elman, *Convergence analysis of iterative solvers in inexact Rayleigh Quotient iteration*, Tenth Copper Mountain Conference on Iterative Methods, http://grandmaster.colorado.edu/ copper/2008/SCWinners/Xue.pdf, 2008.