COMPUTATIONAL & APPLIED MATHEMATICS

Volume 27, N. 2, pp. 151–173, 2008

ISSN 0101-8205
www.scielo.br/cam

Residual iterative schemes for large-scale nonsymmetric positive definite linear systems

WILLIAM LA CRUZ[1]* and MARCOS RAYDAN[2]**

[1]Dpto. de Electrónica, Computación y Control, Escuela de Ingeniería Eléctrica
Facultad de Ingeniería, Universidad Central de Venezuela
Caracas 1051-DF, Venezuela
[2]Dpto. de Computación, Facultad de Ciencias, Universidad Central de Venezuela
Ap. 47002, Caracas 1041-A, Venezuela

E-mails: william.lacruz@ucv.ve / mraydan@kuaimare.ciens.ucv.ve

Abstract. A new iterative scheme that uses the residual vector as search direction is proposed and analyzed for solving large-scale nonsymmetric linear systems, whose matrix has a positive (or negative) definite symmetric part. It is closely related to Richardson's method, although the stepsize and some other new features are inspired by the success of recently proposed residual methods for nonlinear systems. Numerical experiments are included to show that, without preconditioning, the proposed scheme outperforms some recently proposed variations on Richardson's method, and competes with well-known and well-established Krylov subspace methods: GMRES and BiCGSTAB. Our computational experiments also show that, in the presence of suitable preconditioning strategies, residual iterative methods can be competitive, and sometimes advantageous, when compared with Krylov subspace methods.

Mathematical subject classification: 65F10, 65F25, 49M07.

Key words: linear systems, Richardson's method, Krylov subspace methods, spectral gradient method.

#722/07. Received: 08/V/07. Accepted: 28/XI/07.

*This author was supported by CDCH project PI-08-14-5463-2004 and Fonacit UCV-PROJECT 97-003769.

**This author was supported by the Graduate Program in Computer Science at UCV and Fonacit UCV-PROJECT 97-003769.

1 Introduction

We are interested in solving linear systems of equations,

$$Ax = b, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is not symmetric, $(A + A^T)$ is positive (or negative) definite, $b \in \mathbb{R}^n$, and n is large. This kind of linear systems arise in many areas of scientific computing. For example, when discretizing the two point boundary value problems or the partial differential equations that frequently appear in oil-reservoir engineering, in weather forecasting, or in electronic device modelling among others, linear systems like (1) need to be solved (see, e.g., [1, 16]).

The well-known Richardson's method (also known as Chebyshev method) and its variations are characterized by using the residual vector, $r(x) = b - Ax$, as search direction to solve linear systems iteratively (see, e.g., [7, 11, 12, 24, 26]). In general, these variations of Richardson's method have not been considered to be competitive with Krylov subspace methods, which represent nowadays the best-known options for solving (1), specially when combined with suitable preconditioning strategies. For a review on Richardson's method and variations see [8, 29, 30].

Nevertheless, from a different perspective, solving linear systems of equations can be seen as a particular (although very special) case of solving nonlinear systems of equations. For nonlinear systems, some new iterative schemes have recently been presented that use in a systematic way the residual vectors as search directions [19, 20]. These low-memory ideas become effective, and competitive with Newton-Krylov ([3, 9, 10, 18]) schemes for large-scale nonlinear systems, when the step lengths are chosen in a suitable way.

In this work we combine and adapt, for linear systems, the ideas introduced in [19, 20] for the nonlinear case. To be precise, we present in Section 2 a scheme that takes advantage of the method presented in [19] for choosing the direction, plus or minus the residual, depending on the sign of a Rayleigh quotient closely related to the step length. It also takes advantage of the new globalization strategy proposed and analyzed in [20] that allows the norm of the residual to decrease non monotonically and still guarantees convergence.

It is worth noticing that since our proposal uses plus or minus the residual

as search direction, then it can be viewed as a new variant of the well-known Richardson's method. However, there are significant new features. The most important is the use of the spectral steplength choice, also known as the Barzilai-Borwein choice, ([2, 13, 17, 22]) that has proved to yield fast local convergence for the solution of nonlinear optimization problems ([4, 5, 6, 15, 23]). However, this special choice of step size cannot guarantee global convergence by itself, as it usually happens with other variations (see, e.g., [7, 11, 12, 24, 26]). For that, we combine its use with a tolerant globalization strategy, that represents the second new feature. In section 3 we present a preliminary numerical experimentation to compare the proposed scheme with some variations on Richardson's method, and also with well-known and well-established Krylov subspace methods: GMRES and BiCGSTAB, with and without preconditioning. In section 4 we present some concluding remarks.

2 General algorithm and convergence

We now present our general algorithm for solving the minimization problem

$$\min_{x\in\mathbb{R}^n} f(x) \equiv \|g(x)\|^2,$$

where

$$g(x) = Ax - b, \tag{2}$$

and $\|\cdot\|$ denotes, throughout this work, the Euclidian norm. The new algorithm generates the iterates using plus or minus the residual vector as search direction, and a spectral steplength closely related to the Barzilai-Borwein choice of steplength [2], as follows

$$x_{k+1} = x_k + \mathrm{sgn}(\beta_k)(1/\beta_{k-1})r_k,$$

where $\beta_k = (r_k^t A r_k)/(r_k^t r_k)$, and $r_k = b - Ax_k = -g(x_k)$ is the residual vector at x_k. The use of the this steplength is inspired by the success obtained recently for solving nonlinear systems of equations [19, 20]. Properties of the spectral step length, $\alpha_{k+1} = (r_k^t r_k)/(r_k^t A r_k)$, for the minimization of convex quadratic functions were established in [22], and further analyzed in [13]. For a review containing the more recent advances on spectral choices of the steplength, see [17].

To guarantee convergence, from any initial guess x_0 and any positive initial steplength $1/\beta_0$, the nonmonotone line search condition

$$f(x_{k+1}) \leq f(x_k) + \eta_k - \gamma\lambda_k^2\|g(x_k)\|^2, \tag{3}$$

needs to be enforced at every k, for a $0 < \lambda_k \leq 1$. Obtaining λ_k via a backtracking process to force (3) represents a globalization strategy that is inspired by the proposition presented in [20], and requires some given parameters: $\{\eta_k\}$, γ, and $\sigma_{\min} < \sigma_{\max}$. Let us assume that $\{\eta_k\}$ is a given sequence such that $\eta_k > 0$ for all $k \in \mathbb{N}$ (the set of natural numbers) and

$$\sum_{k=0}^{\infty} \eta_k = \eta < \infty. \tag{4}$$

Let us also assume that $\gamma \in (0, 1)$ and $0 < \sigma_{\min} < \sigma_{\max} < 1$.

Algorithm 2.1. *Residual Algorithm* 1 *(RA1)*

Given: $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\gamma \in (0, 1)$, $0 < \sigma_{\min} < \sigma_{\max} < 1$, $\{\eta_k\}_{k\in\mathbb{N}}$ such that (4) holds. Set $r_0 = b - Ax_0$, and $k = 0$;

Step 1. If $r_k = 0$, stop the process (successfully);

Step 2. Set $\beta_k = (r_k^t A r_k)/(r_k^t r_k)$;

Step 3. If $\beta_k = 0$, stop the process (unsuccessfully);

Step 4. (Backtracking process) Set $\lambda \leftarrow 1$;

Step 5. If $\|r_k - \text{sgn}(\beta_k)(\lambda/\alpha_k)Ar_k\|^2 \leq \|r_k\|^2 + \eta_k - \gamma\lambda^2\|r_k\|^2$ go to Step 7;

Step 6. Choose $\sigma \in [\sigma_{\min}, \sigma_{\max}]$, set $\lambda \leftarrow \sigma\lambda$, and go to Step 5;

Step 7. Set $\lambda_k = \lambda$, $x_{k+1} = x_k + \text{sgn}(\beta_k)(\lambda/\alpha_k)r_k$, and $r_{k+1} = r_k - \text{sgn}(\beta_k)(\lambda/\alpha_k)Ar_k$;

Step 8. Set $\alpha_{k+1} = |\beta_k|$, $k = k + 1$ and go to Step 1.

Remark 2.1. In practice, the parameters associated with the line search strategy are chosen to reduce the number of backtrackings as much as possible while keeping the convergence properties of the method. For example, the parameter $\gamma > 0$ is chosen as a very small number ($\gamma \approx 1.D - 4$), and η_k is chosen as a

large number when $k = 0$ and then it is reduced as slow as possible making sure that (4) holds (e.g., $\eta_k = 10^4(1 - 10^{-6})^k$). Finally $\sigma_{\min} < \sigma_{\max}$ are chosen as classical safeguard parameters in the line search process (e.g., $\sigma_{\min} = 0.1$ and $\sigma_{\max} = 0.5$).

Remark 2.2. For all $k \in \mathbb{N}$, the following properties can be easily established:

(i) $g(x_{k+1}) = g(x_k) - \operatorname{sgn}(\beta_k)(\lambda_k/\alpha_k)Ag(x_k)$.

(ii) $d_k = \operatorname{sgn}(\beta_k)(1/\alpha_k)r_k = -\operatorname{sgn}(\beta_k)(1/\alpha_k)g(x_k)$.

(iii) $f(x_k + \lambda d_k) = \|r_k - \operatorname{sgn}(\beta_k)(\lambda/\alpha_k)Ar_k\|^2$.

(iv) $\|r_k - (\lambda/\alpha_k)\operatorname{sgn}(\beta_k)Ar_k\|^2 \leq \|r_k\|^2 + \eta_k - \gamma\lambda^2\|r_k\|^2$.

In order to present some convergence analysis for Algorithm 2.1, we first need some technical results.

Proposition 2.1. *Let $\{x_k\}_{k\in\mathbb{N}}$ be the sequence generated by Algorithm* 2.1. *Then, d_k is a descent direction for the function f, for all $k \in \mathbb{N}$.*

Proof. Since $\nabla f(x) = 2A^t g(x)$, then we can write

$$\begin{aligned}
\nabla f(x_k)^t d_k &= 2(g(x_k)^t A)(-\operatorname{sgn}(\beta_k)(1/\alpha_k)g(x_k)) \\
&= -2(g(x_k)^t Ag(x_k))\operatorname{sgn}(\beta_k)(1/\alpha_k) \\
&= -2(\beta_k(g(x_k)^t g(x_k)))\operatorname{sgn}(\beta_k)(1/\alpha_k) \\
&= -2(|\beta_k|/\alpha_k)\|g(x_k)\|^2 < 0, \text{ for } k \in \mathbb{N}.
\end{aligned}$$

This completes the proof. □

By Proposition 2.1 it is clear that Algorithm 2.1 can be viewed as an iterative process for finding stationary points of f. In that sense, the convergence analysis for Algorithm 2.1 consists in proving that the sequence of iterates $\{x_k\}_{k\in\mathbb{N}}$ is such that $\lim_{k\to\infty} \nabla f(x_k) = 0$.

First we need to establish that Algorithm 2.1 is well defined.

Proposition 2.2. *Algorithm* 2.1 *is well defined.*

Proof. Since $\eta_k > 0$, then by the continuity of $f(x)$, the condition

$$f(x_k + \lambda d_k) \leq f(x_k) + \eta_k - \gamma\lambda^2 \|g(x_k)\|^2,$$

is equivalent to

$$\|r_k - (\lambda/\alpha_k)\,\mathrm{sgn}(\beta_k) A r_k\|^2 \leq \|r_k\|^2 + \eta_k - \gamma\lambda^2 \|r_k\|^2,$$

that holds for $\lambda > 0$ sufficiently small. □

Our next result guarantees that the whole sequence of iterates generated by Algorithm 2.1 is contained in a subset of $\mathbb{R}^n$.

Proposition 2.3. *The sequence* $\{x_k\}_{k\in\mathbb{N}}$ *generated by Algorithm* 2.1 *is contained in the set*

$$\Phi_0 = \left\{x \in \mathbb{R}^n : 0 \leq f(x) \leq f(x_0) + \eta\right\}. \tag{5}$$

Proof. Clearly $f(x_k) \geq 0$ for $k \in \mathbb{N}$. Hence, it suffices to prove that $f(x_k) \leq f(x_0) + \eta$ for $k \in \mathbb{N}$. For that we first prove by induction that

$$f(x_k) \leq f(x_0) + \sum_{i=0}^{k-1} \eta_i. \tag{6}$$

Equation (6) holds for $k = 1$. Indeed, since λ_1 satisfies (3) then

$$f(x_1) \leq f(x_0) + \eta_0.$$

Let us suppose that (6) holds for $k-1$ where $k \geq 2$. We will show that (6) holds for k. Using (3) and (6) we obtain

$$f(x_k) \leq f(x_{k-1}) + \eta_{k-1} \leq f(x_0) + \sum_{i=0}^{k-2} \eta_i + \eta_{k-1} = f(x_0) + \sum_{i=0}^{k-1} \eta_i,$$

which proves that (6) holds for $k \geq 2$. Finally, using (4) and (6) it follows that, for $k \geq 0$:

$$f(x_{k+1}) \leq f(x_0) + \sum_{i=0}^{k} \eta_i \leq f(x_0) + \eta,$$

and the result is established. □

For our convergence results, we need the following two technical propositions. The next one is presented and established in [14] as Lemma 3.3. We include it here for the sake of completeness.

Proposition 2.4. *Let $\{a_k\}_{k\in\mathbb{N}}$ and $\{b_k\}_{k\in\mathbb{N}}$ be sequences of positive numbers satisfying*

$$a_{k+1} \leq (1+b_k)a_k + b_k \ \text{ and } \ \sum_{k=0}^{\infty} b_k < \infty.$$

Then, $\{a_k\}_{k\in\mathbb{N}}$ converges.

Proposition 2.5. *If $\{x_k\}_{k\in\mathbb{N}}$ is the sequence generated by Algorithm* 2.1*, then*

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\|^2 < \infty, \tag{7}$$

and

$$\lim_{k\to\infty} \lambda_k \|g(x_k)\| = 0. \tag{8}$$

Proof. Using Proposition 2.3 and the fact that Φ_0 is clearly bounded, $\{\|g(x_k)\|\}_{k\in\mathbb{N}}$ is also bounded. Since $\|x_{k+1} - x_k\| = \lambda_k \|g(x_k)\|$, then using (3) we have that

$$\|x_{k+1} - x_k\|^2 = \lambda_k^2 \|g(x_k)\|^2 \leq \frac{\eta_k}{\gamma} + \frac{1}{\gamma}\big(f(x_k) - f(x_{k+1})\big). \tag{9}$$

Since η_k satisfies (4), adding in both sides of (9) it follows that

$$\begin{aligned} \sum_{k=0}^{\infty} \|x_{k+1} - x_k\|^2 &\leq \frac{1}{\gamma}\sum_{k=0}^{\infty} \eta_k + \frac{1}{\gamma}\sum_{k=0}^{\infty} \big(f(x_k) - f(x_{k+1})\big) \\ &\leq \frac{\eta + f(x_0)}{\gamma} < \infty, \end{aligned}$$

which implies that

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0,$$

and so

$$\lim_{k\to\infty} \lambda_k \|g(x_k)\| = 0.$$

Hence, the proof is complete. □

Proposition 2.6. *If $\{x_k\}_{k\in\mathbb{N}}$ is the sequence generated by Algorithm* 2.1*, then the sequence $\{\|g(x_k)\|\}_{k\in\mathbb{N}}$ converges.*

Proof. Since $f(x_k) \geq 0$ and $(1 + \eta_k) \geq 1$, for all $k \in \mathbb{N}$, then using (3) we have that

$$f(x_{k+1}) \leq f(x_k) + \eta_k \leq (1 + \eta_k) f(x_k) + \eta_k.$$

Setting $a_k = f(x_k)$ and $b_k = \eta_k$, then it can also be written as

$$a_{k+1} \leq (1 + b_k)a_k + b_k,$$

and $\sum_{k=0}^{\infty} b_k < \eta < \infty$. Therefore, by Proposition 2.4, the sequence $\{a_k\}_{k\in\mathbb{N}}$ converges, i.e., the sequence $\{f(x_k)\}_{k\in\mathbb{N}}$ converges. Finally, since $f(x) = \|g(x)\|^2$, then the sequence $\{\|g(x_k)\|\}_{k\in\mathbb{N}}$ converges. □

We now present the main convergence result of this section. Theorem 2.1 shows that either the process terminates at a solution or it produces a sequence $\{r_k\}_{k\in\mathbb{N}}$ for which $\lim_{k\to\infty} r_k^t A r_k = 0$.

Theorem 2.1. *Algorithm* 2.1 *terminates at a finite iteration i where* $r_i = 0$*, or it generates a sequence* $\{r_k\}_{k\in\mathbb{N}}$ *such that*

$$\lim_{k\to\infty} r_k^t A r_k = 0.$$

Proof. Let us assume that Algorithm 2.1 does not terminate at a finite iteration. By continuity, it suffices to show that any accumulation point $\bar{x}$ of the sequence $\{x_k\}_{k\in\mathbb{N}}$ satisfies $g(\bar{x})^t A g(\bar{x}) = 0$. Let $\bar{x}$ be a accumulation point of $\{x_k\}_{k\in\mathbb{N}}$. Then, there exists an infinite set of indices $R \subset \mathbb{N}$ such that $\lim_{k\to\infty, k\in R} x_k = \bar{x}$.

From Proposition 2.5 we have that

$$\lim_{k\to\infty} \lambda_k \|g(x_k)\| = 0$$

that holds if

$$\lim_{k\to\infty} \|g(x_k)\| = 0, \tag{10}$$

or if

$$\liminf_{k\to\infty} \lambda_k = 0. \tag{11}$$

If (10) holds, the result follows immediately.

Let us assume that (11) holds. Then, there exists an infinite set of indices $K = \{k_1, k_2, k_3, \dots\} \subseteq \mathbb{N}$ such that

$$\lim_{j\to\infty} \lambda_{k_j} = 0.$$

If $R \cap K = \emptyset$, then by Proposition 2.5

$$\lim_{k\to\infty, k\in R} \|g(x_k)\| = 0.$$

Therefore, the thesis of the theorem is established.

Without loss of generality, we can assume that $K \subseteq R$. By the way λ_{k_j} is chosen in Algorithm 2.1, there exists an index $\bar{j}$ sufficiently large such that for all $j \geq \bar{j}$, there exists ρ_{k_j} ($0 < \sigma_{\min} \leq \rho_{k_j} \leq \sigma_{\max}$) for which $\lambda = \lambda_{k_j}/\rho_{k_j}$ does not satisfy condition (3), i.e.,

$$\begin{aligned} f\left(x_{k_j} + \frac{\lambda_{k_j}}{\rho_{k_j}} d_{k_j}\right) &> f(x_{k_j}) + \eta_{k_j} - \gamma \frac{\lambda_{k_j}^2}{\rho_{k_j}^2} \|g(x_{k_j})\|^2 \\ &\geq f(x_{k_j}) - \gamma \frac{\lambda_{k_j}^2}{\rho_{k_j}^2} \|g(x_{k_j})\|^2. \end{aligned}$$

Hence,

$$\frac{f\left(x_{k_j} + \frac{\lambda_{k_j}}{\rho_{k_j}} d_{k_j}\right) - f\left(x_{k_j}\right)}{\lambda_{k_j}/\rho_{k_j}} > -\gamma \frac{\lambda_{k_j}}{\rho_{k_j}} \|g(x_{k_j})\|^2 \geq -\gamma \frac{\lambda_{k_j}}{\sigma_{\min}} \|g(x_{k_j})\|^2.$$

By the Mean Value Theorem it follows that

$$\nabla f\left(x_{k_j} + t_{k_j} d_{k_j}\right)^t d_{k_j} > -\gamma \frac{\lambda_{k_j}}{\sigma_{\min}} \|g(x_{k_j})\|^2, \quad \text{for } j \geq \bar{j}, \tag{12}$$

where $t_{k_j} \in [0, \lambda_{k_j}/\rho_{k_j}]$ tends to zero when $j \to \infty$. By continuity and the definitions of β_k and d_k, we obtain

$$\lim_{j\to\infty} d_{k_j} = -\operatorname{sgn}\left(\frac{g(\bar{x})^t A g(\bar{x})}{g(\bar{x})^t g(\bar{x})}\right)(1/\bar{\alpha}) g(\bar{x}), \tag{13}$$

where $\bar{\alpha} = \lim_{k\to\infty, k\in K} \alpha_k$. We can assume that $\bar{\alpha} > 0$. If $\bar{\alpha} = 0$, then by the definition of the α_k, the thesis of the theorem is established. Setting $\bar{d} = \lim_{j\to\infty} d_{k_j}$ and noticing that $(x_{k_j} + t_{k_j} d_{k_j}) \to \bar{x}$ when $j \to \infty$, then taking limits in (12) we have that

$$\nabla f(\bar{x})^t \bar{d} \geq 0. \tag{14}$$

Since $\nabla f(\bar{x})^t \bar{d} = 2 g(\bar{x})^t A \bar{d}$, $\bar{\alpha} > 0$, and $\nabla f(\bar{x})^t \bar{d} < 0$, then by (13) and (14) we obtain

$$g(\bar{x})^t A g(\bar{x}) = 0.$$

This completes the proof. □

Theorem 2.1 guarantees that Algorithm 2.1 converges to a solution of (1) whenever the Rayleigh quotient of A,

$$c(x) = \frac{x^t Ax}{x^t x}, \quad x \neq 0, \tag{15}$$

satisfies that $|c(r_k)| > 0$, for $k \geq 1$. If the matrix A is indefinite, then it could happen that Algorithm 2.1 generates a sequence $\{r_k\}_{k\in\mathbb{N}}$ that converges to the residual $\bar{r}$ such that $\bar{r}^t A\bar{r} = 0$ and $\bar{r} \neq 0$.

In our next result, we show the convergence of the algorithm when the symmetric part of A, $A_s = (A^t + A)/2$, is positive definite, that appears in many different applications. Of course, similar properties will hold when A_s is negative definite.

Theorem 2.2. *If the matrix A_s is positive definite, then Algorithm* 2.1 *terminates at a finite iteration i where $r_i = 0$, or it generates a sequence $\{r_k\}_{k\in\mathbb{N}}$ such that*

$$\lim_{k\to\infty} r_k = 0.$$

Proof. Since A_s is positive definite, $r_k^t Ar_k = r_k^t A_S r_k > 0$, $r_k \neq 0$, for all $k \in \mathbb{N}$. Then, by the Theorem 2.1 the Algorithm 2.1 terminates at a finite iteration i where $r_i = 0$, or it generates a sequence $\{r_k\}_{k\in\mathbb{N}}$ such that $\lim_{k\to\infty} r_k = 0$. □

To be precise, the next proposition shows that if A_s is positive definite, then in Algorithm 2.1 it holds that $\beta_k > 0$ and $d_k = (1/\alpha_k)r_k$, for all $k \in \mathbb{N}$.

Proposition 2.7. *Let the matrix A_s be positive definite, and let $\alpha_{\min}$ and $\alpha_{\max}$ be the smallest and the largest eigenvalues of A_s, respectively. Then the sequences $\{\beta_k\}_{k\in\mathbb{N}}$ and $\{d_k\}_{k\in\mathbb{N}}$, generated by Algorithm* 2.1 *satisfy that $d_k = (1/\alpha_k)r_k$, for $k \in \mathbb{N}$.*

Proof. It is well-known that the Rayleigh quotient of A satisfies, for any $x \neq 0$,

$$0 < \alpha_{\min} \leq c(x) \leq \alpha_{\max}. \tag{16}$$

By the definition of β_k we have that

$$\beta_k = c(r_k) \geq \alpha_{\min} > 0, \quad \text{for } k \geq 0.$$

Moreover, since $\beta_k > 0$ for $k \geq 0$, then

$$d_k = \operatorname{sgn}(\beta_k)(1/\alpha_k)r_k = (1/\alpha_k)r_k, \quad \text{for } k \geq 0. \qquad \square$$

Proposition 2.7 guarantees that the choice $d_k = (1/\alpha_k)r_k$ is a descent direction, when A_s is positive definite. This yields a simplified version of the algorithm for solving linear systems when the matrix has positive (or negative) definite symmetric part. This simplified version of the algorithm will be referred, throughout the rest of this document, as the Residual Algorithm 2 (*RA2*), for which $\beta_k = \alpha_{k+1} = (r_k^t A r_k)/(r_k^t r_k)$ and $\operatorname{sgn}(\beta_k) = 1$ for all k.

Remark 2.3.

(i) Algorithm *RA2* is well defined.

(ii) The sequence $\{x_k\}_{k\in\mathbb{N}}$ generated by Algorithm *RA2* is contained in Φ_0.

(iii) Since $\alpha_{k+1} = c(r_k)$ and $c(x)$ satisfies (16), then

$$0 < \alpha_{\min} \leq \alpha_k \leq \alpha_{\max}, \quad \text{for } k \geq 1. \tag{17}$$

Moreover,

$$0 < (\lambda_k/\alpha_k) \leq \sigma_{\max}\alpha_{\min}^{-1}, \quad \text{for } k \geq 1.$$

(iv) Since Algorithm *RA2* is a simplified version of Algorithm 2.1 when A_s is positive definite, then its convergence is established by Theorem 2.2.

3 Numerical experiments

We report on some numerical experiments that illustrate the performance of algorithm *RA2*, presented and analyzed previously, for solving nonsymmetric and positive (or negative) definite linear systems. In all experiments, computing was done on a Pentium IV at 3.0 GHz with MATLAB 6.0, and we stop the iterations when

$$\frac{\|r_k\|}{\|b\|} \leq \varepsilon, \tag{18}$$

where $0 < \varepsilon \ll 1$.

As we mentioned in the introduction, our proposal can be viewed as a new variant of the well-known Richardson's method, with some new features. First, we will compare the behavior of algorithm *RA2* with two different variations of Richardson's method, whose general iterative step from a given x_0 is given by

$$x_{k+1} = x_k + \lambda_k r_k,$$

where the residual vectors can be obtained recursively as

$$r_{k+1} = r_k - \lambda A r_k.$$

It is well-known that if we choose the steplengths as the inverse of the eigenvalues of A (i.e., $\lambda_k = \lambda_i^{-1}$ for $1 \leq i \leq n$), then in exact arithmetic the process terminates at the solution in at most n iterations (see [8] and references therein). Due to roundoff errors, in practice, this process is repeated cyclically (i.e., $\lambda_k = \lambda_i^{-1}{}_{\bmod n}$ for all $k \geq n$). In our results, this cyclic scheme will be reported as the *ideal Richardson's method*.

A recent variation on Richardson's method that has optimal properties concerning the norm of the residual, is discussed by Brezinski [7] and chooses the steplength as follows:

$$\lambda_k = \frac{r_k^T w_k}{w_k^T w_k}, \tag{19}$$

where $w_k = Ar_k$. This option will be referred in our results as the *Optimal Richardson's Method* (ORM).

For our first experiment, we set $n = 100$, $b = \text{rand}(n, 1)$, $\varepsilon = 10^{-16}$, and the matrix

$$A = -Gallery\,('lesp', n).$$

The results for this experiment are shown in Figure 1. We observe that the ideal Richardson's method is numerically unstable and requires several cycles to terminate the process. The ORM has a monotone behavior and it is numerically stable, but requires more iterations than the *RA2* algorithm to reach the same accuracy. It is also worth noticing the nonmonotone behavior of the *RA2* algorithm that accounts for the fast convergence.

We now present a comparison on several problems with well-known Krylov subspace methods. GMRES [25] and BiCGSTAB [28] are among the best-known

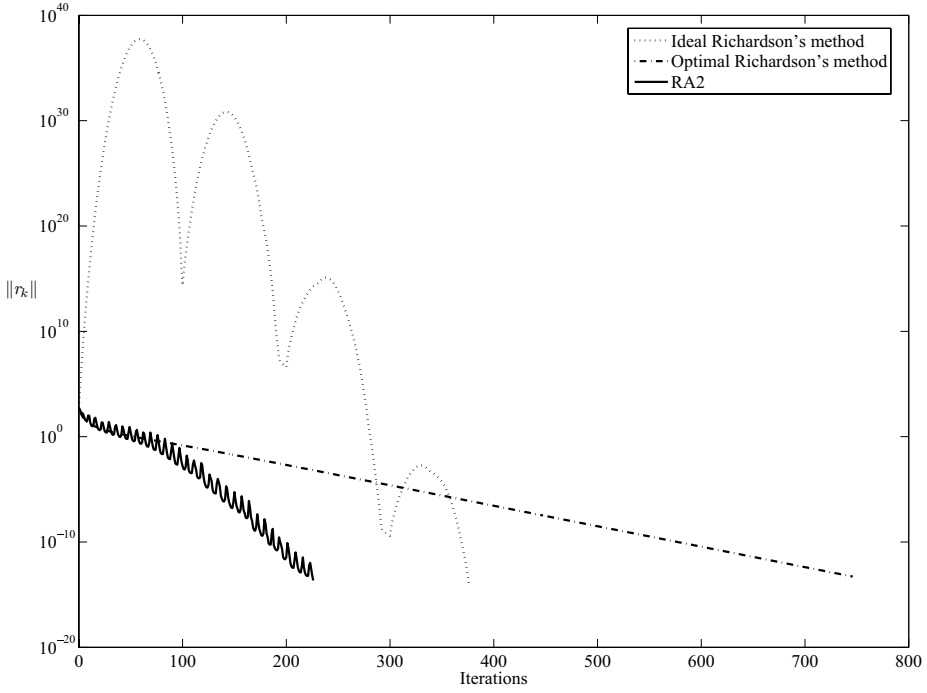

Figure 1 – Behavior of the ideal and the optimal Richadson's method when compared with the *RA2* algorithm for $A = -Gallery\,('lesp', 100)$.

Krylov iterative methods for solving large-scale non symmetric linear systems (see, e.g., [27, 29]). Therefore, we compare the performance of algorithm *RA2* with these two methods, and also with ORM, without preconditioning and also taking advantage of two classical preconditioning strategies of general use: Incomplete LU (ILU) and SSOR. For the preconditioned version of ORM [7], the search direction is given by $z_k = Cr_k$, where C is the preconditioning matrix; and the steplength is given by (19) where now $w_k = Az_k$. In the preconditioned version of *RA2* the residual is redefined as $r_k = C(b - Ax_k)$ for all $k \geq 0$, where once again C is the preconditioning matrix.

In all the experiments described here we use the Householder MATLAB implementation of GMRES, based on [27, Alg. 6.11, page 172], with the restart parameters $m = 20$ (GMRES(20)) and $m = 40$ (GMRES(40)), and the MATLAB implementation of BiCGSTAB, based on [27, Alg. 7.7, page 234]. For

all considered methods we use the vector $x_0 = 0$ as the initial guess. For algorithm *RA2* we use the following parameters: $\alpha_0 = \|b\|$, $\gamma = 10^{-4}$, $\sigma_{\min} = 0.1$, $\sigma_{\max} = 0.5$, and $\eta_k = 10^4 \left(1 - 10^{-6}\right)^k$. For choosing a new λ at Step 4, we use the following procedure, described in [20]: given the current $\lambda_c > 0$, we set the new $\lambda > 0$ as

$$\lambda = \begin{cases} \sigma_{\min}\lambda_c, & \text{if } \lambda_t < \sigma_{\min}\lambda_c; \\ \sigma_{\max}\lambda_c, & \text{if } \lambda_t > \sigma_{\max}\lambda_c; \\ \lambda_t, & \text{otherwise;} \end{cases}$$

where

$$\lambda_t = \frac{\lambda_c^2 f(x_k)}{f(x_k + \lambda_c d) + (2\lambda_c - 1) f(x_k)}.$$

In the following tables, the process is stopped when (18) is attained, but it can also be stopped prematurely for different reasons. We report the different possible failures observed with different symbols as follows:

* : The method reaches the maximum (20000) number of iterations.

** : The method stagnates (three consecutive iterates are exactly the same).

*** : Overflow is observed while computing one of the internal scalars.

For our second experiment we consider a set of 10 test matrices, described in Table 1, and we set the right hand side vector $b = (1, 1, \ldots, 1)^t \in \mathbb{R}^n$. In Table 1 we report the problem number (M), a brief description of the matrix, and the MATLAB commands to generate it.

We summarize on Table 2 the behavior of GMRES(20), GMRES(40), BICGSTAB, ORM, and *RA2* without preconditioning. We have chosen $\varepsilon = 10^{-10}$ in (18) for stopping the iterations. We report the matrix number (M) from Table 1, the dimension of the problem (n); the number of iterations (Iter); and the CPU time in seconds until convergence (T). GMRES(m) requires per iteration one matrix-by-vector multiplication, solving one preconditioned system, and computing 2(Iter mod m) inner products. BICGSTAB requires per iteration two matrix-by-vector multiplications, solving one preconditioned system, and computing 4 inner products. Finally, ORM and *RA2* require per iteration one matrix-by-vector multiplication, solving one preconditioned system, and computing 2 inner products.

M	Description	MATLAB Commands
1	Sparse adjacency matrix from NASA airfoil.	MATLAB demo: airfoil
2	Singular Toeplitz lower Hessenberg matrix	A=gallery('chow',n,1,1)
3	Circulant matrix	A=gallery('circul',v), where $v \in \mathbb{R}^n$ is such that $v_i = \begin{cases} 10^{-6}, & i = 1, \\ 1, & i = n/2, \\ -1, & i = n, \\ 0, & \text{otherwise.} \end{cases}$
4	Diagonally dominant, ill conditioned, tridiagonal matrix	A=gallery('dorr',n,1)
5	Perturbed Jordan block	A=gallery('forsythe',n,-1,2)
6	Matrix whose eigenvalues lie on a vertical line in the complex plane	A=gallery('hanowa',n,n)
7	Jordan block	A=gallery('jordbloc',n,2)
8	Tridiagonal matrix with real sensitive eigenvalues	A = -gallery('lesp',n)
9	Pentadiagonal Toeplitz matrix	A=gallery('toeppen',n,1,10,n,-10,-1)
10	Upper triangular matrix discussed by Wilkinson and others	A=gallery('triw',n,-0.5,2)

Table 1 – First set of test matrices.

In Tables 3 and 4 we report the results for the matrices 4, 5, 6, 7, 8 and 9, when we use the following two preconditioning strategies.

(A) *Incomplete LU factorization with drop tolerance*
The preconditioning matrix is obtained, in MATLAB, with the command $[L, U] = \text{luinc}(A, 0.5)$.

(B) *The SSOR preconditioning strategy*
The preconditioning matrix is given by

$$(D - \omega E)D^{-1}(D - \omega F),$$

where $-E$ is the strict lower triangular part of A, $-F$ is the strict upper triangular part of A, and D is the diagonal part of A. We take $\omega = 1$.

In this case, we set $\varepsilon = 5 \times 10^{-15}$ in (18) for stopping the iterations. In Figure 2 we show the behavior of all considered methods when using preconditioning strategy (A) for problems 5, 7 and 8; and in Figure 3 for problems 4, 5, 7 and 8 when using preconditioning strategy (B).

		GMRES(20)		GMRES(40)		BICGSTAB		*RA2*		ORM	
M	n	Iter	T	Iter	T	Iter	T	Iter	T	Iter	T
1	4253	60	0.406	60	0.425	**	**	64	0.047	75	0.094
2	1000	229	3.047	229	3.100	423	9.266	538	5.594	1044	20.703
3	5000	*	*	*	*	1	0.00	2	0.00	1	0.00
4	500	20674	32.375	20674	32.594	549	0.188	19449	4.969	*	*
5	5000	28	0.313	28	0.334	77	0.141	29	0.016	28	0.015
6	5000	17	0.188	17	0.189	21	0.047	31	0.021	27	0.016
7	5000	27	0.297	27	0.288	62	0.125	28	0.047	27	0.047
8	5000	2562	22.203	2562	21.266	4740	13.688	10943	10.297	*	*
9	5000	4	0.031	4	0.031	4	0.031	4	0.031	4	0.031
10	5000	3067	26.172	3067	26.313	***	***	3408	3.203	3151	3.625

Table 2 – GMRES(20), GMRES(40), BICGSTAB, *RA2*, and ORM without preconditioning.

		GMRES(20)		GMRES(40)		BICGSTAB		*RA2*		ORM	
M	n	Iter	T	Iter	T	Iter	T	Iter	T	Iter	T
4	50000	*	*	*	*	*	*	3	0.088	2	0.094
5	500000	38	59.500	48	97.859	81	37.781	20	4.625	20	5.469
6	500000	1	0.715	1	0.715	1	0.715	2	0.547	1	0.438
7	500000	37	57.500	38	84.375	72	33.469	20	4.641	19	5.453
8	500000	21	34.359	21	35.438	46	22.969	10	2.516	11	3.391
9	500000	2	2.341	2	2.859	3	2.250	2	0.719	2	0.922

Table 3 – GMRES(20), GMRES(40), BICGSTAB, *RA2*, and ORM with preconditioning (A).

For our third experiment we explore the effect, on the convergence behavior of *RA2*, produced by clustering the eigenvalues. For that we consider the right

		GMRES(20)		GMRES(40)		BICGSTAB		*RA2*		ORM	
M	n	Iter	T	Iter	T	Iter	T	Iter	T	Iter	T
4	50000	*	*	*	*	*	*	3	0.078	2	0.078
5	500000	25	286.641	25	288.563	46	266.906	11	61.844	11	62.328
6	500000	1	0.425	1	0.425	1	0.425	2	0.391	1	0.266
7	500000	21	36.641	21	37.828	26	16.469	11	4.703	10	4.828
8	500000	11	14.500	11	15.000	23	11.703	7	2.063	8	2.766
9	500000	2	2.391	2	2.953	3	2.859	2	1.922	2	2.063

Table 4 – GMRES(20), GMRES(40), BICGSTAB, *RA2*, and ORM with preconditioning (B).

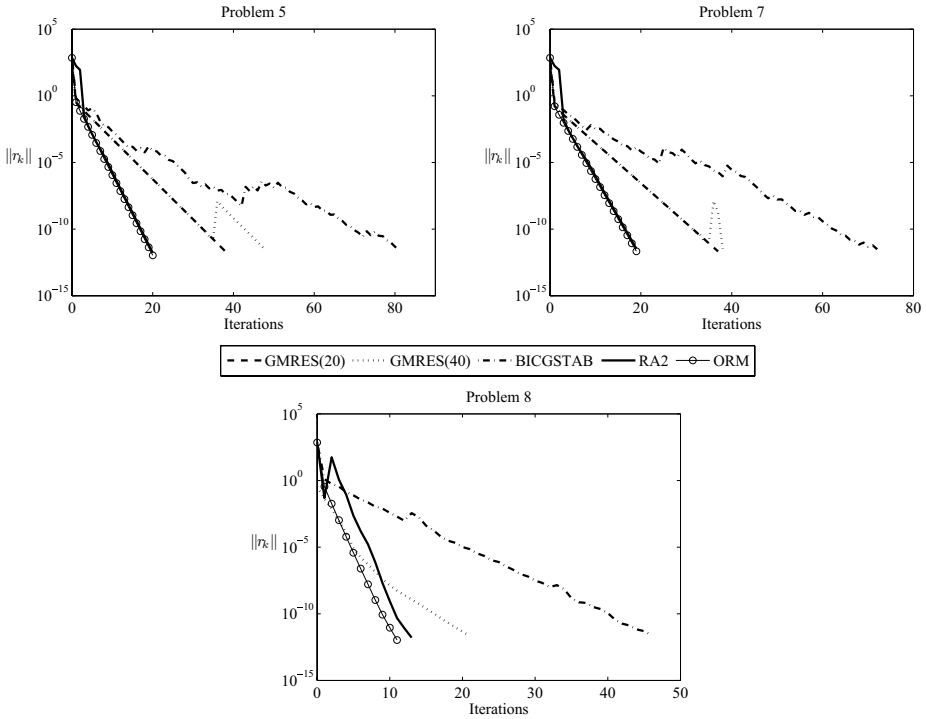

Figure 2 – Behavior of all methods when using preconditioning techniques (A).

hand side vector $b = (1, 1, \ldots, 1)^t$ and the matrix

$$A = \begin{pmatrix} a_{11} & -1 & & & \\ 1 & a_{22} & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & a_{n-1,n-1} & -1 \\ & & & 1 & a_{nn} \end{pmatrix},$$

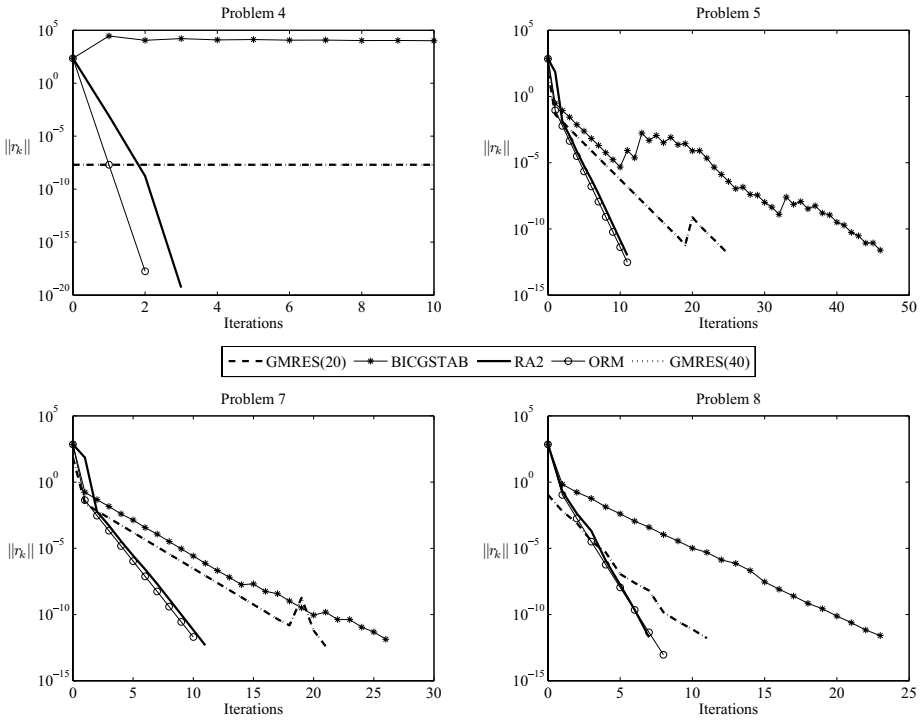

Figure 3 – Behavior of all methods when using preconditioning techniques (B).

where $n = 10000$, $a_{ii} = 3 + (i-1)\left(\frac{\alpha_{\max}-3}{n-1}\right)$, for $i = 1, \ldots, n$, and $\alpha_{\max} \geq 3$. It is clear that the symmetric part of A is a diagonal matrix whose eigenvalues are $\mu_i = a_{ii}$, for $i = 1, \ldots, n$. We consider the following three cases: (i) $\alpha_{\max} = 10$, (ii) $\alpha_{\max} = 1000$, and (iii) $\alpha_{\max} = 10000$. We have chosen $\varepsilon = 10^{-15}$ in (18) for stopping the iterations.

Figure 4 shows the behavior of *RA2*, with and without preconditioning, for the system $Ax = b$ and each one of the cases (i), (ii) and (iii). We clearly observe that clustering the eigenvalues of the symmetric part drastically reduce the number of required iterations. We can also observe that preconditioning (A) was more effective on cases (ii) and (iii), since the eigenvalues in case (i) are already clustered. A similar behavior is observed when the clustering effect is applied to ORM.

For our last test problem, we consider the second order centered-differences

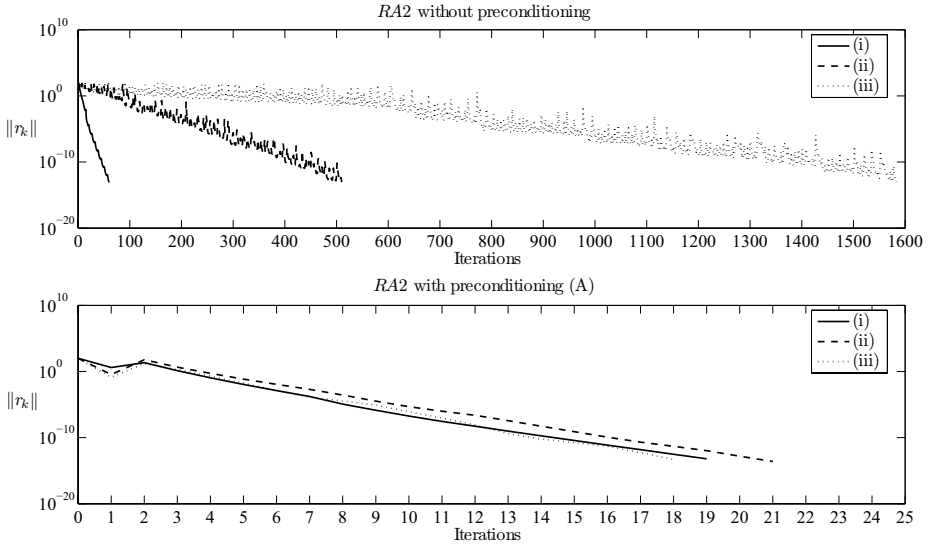

Figure 4 – Behavior of *RA2* for solving $Ax = b$ when clustering the eigenvalues for each one of the cases (i), (ii) and (iii).

discretization of

$$-\nabla^2 u + \gamma(xu_x + yu_y) + \beta u = f, \tag{20}$$

on the unit square, with homogeneous Dirichlet boundary conditions, $u = 0$, on the border of the region. We set the parameters $\gamma = 7100$ and $\beta = 100$ to guarantee that the symmetric part of the matrix is positive definite. The discretization grid has 71 internal nodes per axis producing an $n \times n$ matrix where $n = 5041$. The right hand side vector is chosen such that the solution vector is $x = (1, 1 \ldots, 1)^t$. Once again we compare GMRES(20), GMRES(40), BICGSTAB, ORM, and *RA2* with the preconditioning strategies (A) and (B).

In Table 5 we report the results obtained with GMRES, BICGSTAB, ORM, and *RA2* for solving problem (20), when using the preconditioning strategies described in (A) and (B). We set $\varepsilon = 10^{-13}$ in (18) for stopping the iterations. In Figure 5 we show the behavior of all methods when using preconditioning techniques (A) and (B) for solving (20).

We observe that, in general, *RA2* is a robust method for solving non symmetric linear systems whose symmetric part is positive (negative) definite. Moreover, it is competitive with the well-known GMRES and BICGSTAB in computational

	GMRES(20)		GMRES(40)		BICGSTAB		*RA2*		ORM	
Strategy	Iter	T	Iter	T	Iter	T	Iter	T	Iter	T
(A)	400	3.297	481	5.094	**	**	14	0.031	14	0.031
(B)	400	3.031	240	24.844	**	**	10	0.984	10	0.906

Table 5 – GMRES(20), GMRES(40), BICGSTAB, *RA2*, and ORM for solving (20).

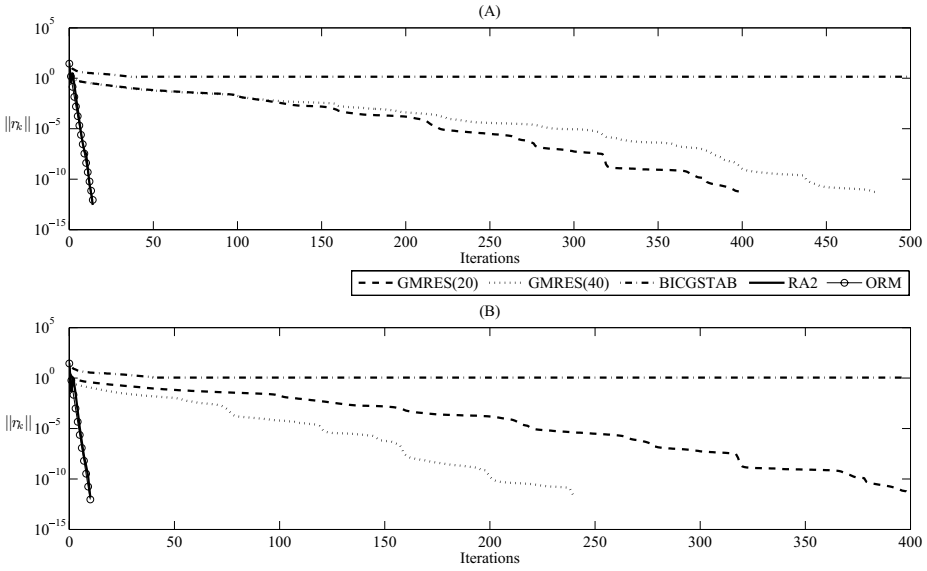

Figure 5 – Behavior of all methods when using preconditioning techniques (A) and (B) for solving (20).

cost and CPU time, without preconditioning. We also observe that *RA2* and ORM outperform GMRES and BICGSTAB when preconditioning strategies that reduce the number of cluster of eigenvalues are incorporated.

4 Conclusions

We present a residual algorithm *RA2* for solving large-scale nonsymmetric linear systems when the symmetric part of the coefficient matrix is positive (or negative) definite. Due to its simplicity for building the search direction, the method is very easy to implement, memory requirements are minimal and, so, its use for solving large-scale problems is attractive. MATLAB codes written by the authors are available upon request.

We have compared the performance of the new residual method, as well as the Optimal Richardson Method (ORM), with the restarted GMRES and BICGSTAB, on some test problems, without preconditioning and also using two classical preconditioning strategies (ILU and SSOR). Our preliminary numerical results indicate that using the residual direction with a suitable step length can be competitive for solving large-scale problems, and preferable when the eigenvalues are clustered by a preconditioning strategy. Many new preconditioning techniques have been recently developed (see e.g., [27, 29] and references therein) that possess the clustering property when dealing with nonsymmetric matrices. In general, any preconditioning strategy that reduces the number of cluster of eigenvalues of the coefficient matrix (suitable for Krylov-subspace methods) should accelerate the convergence of the residual schemes considered in this work. In that sense, the *RA2* method can be viewed as an extension of the preconditioned residual method, based on the Barzilai-Borwein choice of step length, introduced in [21].

In order to explain the outstanding and perhaps unexpected good behavior of *RA2* and ORM when an effective preconditioner is applied, it is worth noticing that if the preconditioning matrix C approaches A^{-1} then the steplength chosen by *RA2*, as well as the one chosen by ORM, approaches 1. Consequently, both preconditioned residual iterations tend to recover Newton's method for finding the root of the linear map $CAx - Cb = 0$, which requires for $C = A^{-1}$ only one iteration to terminate at the solution.

For nonsymmetric systems with an indefinite symmetric part, the proposed general scheme *RA1* can not guarantee convergence to solutions of the linear system, and usually convergence to points that satisfy $\lim_{k\to\infty} r_k^t A r_k = 0$ but such that $\lim_{k\to\infty} r_k \neq 0$, as predicted by Theorem 2.1, is observed in practice. For this type of general linear systems, it would be interesting to analyze in the near future the possible advantages of using ORM with suitable preconditioning strategies.

Acknowledgement. We are indebted to an anonymous referee, whose insightful comments helped us to improve the quality of the present paper.

REFERENCES

[1] O. Axelsson and V.A. Barker, *Finite Element Solution of Boundary Value Problems, Theory and Computation*. Academic Press, New York (1984).

[2] J. Barzilai and J.M. Borwein. *Two-point step size gradient methods*. IMA Journal of Numerical Analysis, **8** (1988), 141–148.

[3] S. Bellavia and B. Morini, *A globally convergent newton-gmres subspace method for systems of nonlinear equations*. SIAM J. Sci. Comput., **23** (2001), 940–960.

[4] E.G. Birgin and J.M. Martínez, *A spectral conjugate gradient method for unconstrained optimization*. Applied Mathematics and Optimization, **43** (2001), 117–128.

[5] E.G. Birgin and Y.G. Evtushenko, *Automatic differentiation and spectral projected gradient methods for optimal control problems*. Optimization Methods and Software, **10** (1998), 125–146.

[6] E.G. Birgin, J.M. Martínez and M. Raydan, *Nonmonotone spectral projected gradient methods on convex sets*. SIAM Journal on Optimization, **10** (2000), 1196–1211.

[7] C. Brezinski, *Variations on Richardson's method and acceleration*. Bull. Soc. Math. Belg., (1996), 33–44.

[8] C. Brezinski, *Projection Methods for Systems of Equations*. North Holland, Amsterdam (1997).

[9] P. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*. SIAM J. Sci. Comp., **11** (1990), 450–481.

[10] P. Brown and Y. Saad, *Convergence theory of nonlinear Newton-Krylov algorithms*. SIAM Journal on Optimization, **4** (1994), 297–330.

[11] D. Calvetti and L. Reichel, *Adaptive Richardson iteration based on Leja points*. J. Comp. Appl. Math., **71** (1996), 267–286.

[12] D. Calvetti and L. Reichel, *An adaptive Richardson iteration method for indefinite linear systems*. Numer. Algorithms, **12** (1996), 125–149.

[13] Y.H. Dai and L.Z. Liao, *R-linear convergence of the Barzilai and Borwein gradient method*. IMA Journal on Numerical Analysis, **22** (2002), 1–10.

[14] J.E. Jr. Dennis and J.J. Moré, *A characterization of superlinear convergence and its applications to quasi-Newton methods*. Math. of Comp., **28** (1974), 549–560.

[15] M.A. Diniz-Ehrhardt, M.A. Gomes-Ruggiero, J.M. Martínez and S.A. Santos, *Augmented Lagrangian algorithms based on the spectral gradient for solving nonlinear programming problems*. Journal of Optimization Theory and Applications, **123** (2004), 497–517.

[16] H.C. Elman, D.J. Sylvester and A.J. Wathen, *Finite Elements and Fast Iterative Solvers*. Oxford University Press (2005).

[17] R. Fletcher, On the Barzilai-Borwein method. In: *Optimization and Control with Applications* (L.Qi, K.L. Teo, X.Q. Yang, eds.) Springer, (2005), 235–256.

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

[19] W. La Cruz and M. Raydan, *Nonmonotone spectral methods for large-scale nonlinear systems*. Optimization Methods and Software, **18** (2003), 583–599.

[20] W. La Cruz, J.M. Martínez and M. Raydan, *Spectral residual method without gradient information for solving large-scale nonlinear systems*. Math. of Comp., **75** (2006), 1449–1466.

[21] B. Molina and M. Raydan, *Preconditioned Barzilai-Borwein method for the numerical solution of partial differential equations*. Numerical Algorithms, **13** (1996), 45–60.

[22] M. Raydan, *On the Barzilai and Borwein choice of the steplength for the gradient method*. IMA Journal on Numerical Analysis **13** (1993), 321–326.

[23] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*. SIAM Journal on Optimization, **7** (1997), 26–33.

[24] L. Reichel, *The application of Leja points to Richardson iteration and polynomial preconditioning*. Linear Algebra Appl., **154-156** (1991), 389–414.

[25] Y. Saad and M.H. Shultz, *GMRES: generalized minimal residual algorithm for solving nonsymmetric linear systems*. SIAM J. Sci. Stat. Comput., **7** (1986), 856–869.

[26] D.C. Smolarski and P.E. Saylor, *An optimum semi-iterative method for solving any linear system with a square matrix*. BIT, **28** (1988), 163–178.

[27] Y. Saad, *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia (2003).

[28] H.A. van der Vorst, *Bi-CGSTAB: a fast and smoothly convergent variant Bi-CG for the solution of non-symmetric linear systems*. SIAM J. Sci. Stat. Comput., **13** (1992), 631–644.

[29] H.A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press (2003).

[30] D.M. Young, A historical review of iterative methods. In: *A History of Scientific Computing* (S.G. Nash, Eds.) Addison-Wesley Reading, MA, 180–194 (1990).