Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Residual methods for the large-scale matrix p-th root and some related problems

Braulio De Abreu y Marcos Raydan

RT 2009-06

Centro de Cálculo Científico y Tecnológico (CCCT)

Caracas, Marzo, 2009

Residual methods for the large-scale matrix p-th root and some related problems

Braulio De Abreu [*] Marcos Raydan [†]

March 22, 2009

Abstract

The problem of finding the p-th root of a matrix has received special attention in the last few years. Standard approaches for this problem include and combine some variations of Newton's method, which in turn involve matrix factorizations that, in general, are not suitable for large-scale problems. Motivated by some recently developed low-cost iterative schemes for nonlinear problems, we consider and analyze specialized residual methods that only require a few matrix-matrix products per iteration, and hence are suitable for the large-scale case. As a by-product we also discuss the advantages of residual methods for general nonlinear problems whose variables separate. Preliminary and encouraging numerical results are presented for computing p-th roots of large-scale symmetric and positive definite matrices, for different values of p.

Key words: Nonlinear matrix equations, p-th roots of matrices, residual methods.

1 Introduction

Consider the nonlinear matrix equation

$$F(X) = X^p - A, \tag{1}$$

where $A \in I\!R^{n\times n}$ is symmetric and positive definite (SPD) and $p \geq 2$ is a positive integer. A solution $X \in I\!R^{n\times n}$ of (1) is called a matrix p-th root of A. In this work we are interested in the SPD solution of (1). This problem appears for instance as a useful tool in the calculation of matrix logarithms [6, 14], and also for computing the matrix sector function [22, 26]. It also appears in the solution of Markov processes associated

[*]Departamento de Matemáticas, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela. Supported by the Alma-Mater OPSU program. (`bdeabreu@uc.edu.ve`)

[†]Departamento de Cómputo Científico y Estadística, Universidad Simón Bolívar, Ap. 89000, Caracas 1080-A, Venezuela. Partially supported by the Scientific Computing Center at UCV. (`marcos.raydan@ciens.ucv.ve`)

with financial mathematics [9]. The problem of computing the square root ($p = 2$) has received significant attention [4, 5, 10, 11, 12, 17, 18, 23], and the general case has also been considered [2, 13, 24, 25, 26, 27]. Motivated by the recently developed low-cost residual schemes [15, 16] for nonlinear problems, and by the preliminary numerical results presented in [21], we present and analyze specialized residual methods for solving (1).

The general residual algorithm (see [15, 16, 21]) for solving $F(X) = 0$, where F is any nonlinear map from the space of real square matrices into the space of real square matrices, can be written as:

Algorithm 1 General residual algorithm

```
1: Let $X_0 \in I\!R^{n\times n}, \alpha_0 \in I\!R$, $\alpha_0 \neq 0$
2: for $k = 0, 1, \cdots$ do
3:     $X_{k+1} = X_k - (1/\alpha_k)F(X_k)$
4:     $S_k = X_{k+1} - X_k$
5:     $Y_k = F(X_{k+1}) - F(X_k)$
6:     $\alpha_{k+1} = \langle S_k, Y_k\rangle / \langle S_k, S_k\rangle$        $\triangleright$ $\langle A, B\rangle = trace(A^T B)$
7: end for
```

Notice that the search direction $-F(X_k)$, in Algorithm 1, makes it a derivative free method. Notice also that the step length $\lambda_k = (1/\alpha_k)$ is based on the nonmonotone spectral step length [1, 7, 19] (also known as the Barzilai-Borwein step length), that requires inexpensive calculations. For a complete review on the spectral step length see [3] and references therein. To study the convergence of Algorithm 1, for solving (1), we first present in Section 2 a more general analysis for problems whose variables separate. This approach will allow us to study some other related problems in Section 4. In Section 3 we present a convergence analysis of our specialized residual scheme for computing the matrix p-th root. In Section 5 we describe some preliminary numerical experiments, and in Section 6 we close with some final remarks.

2 A general analysis for problems whose variables separate

Consider the nonlinear system of equations whose variables separate

$$F(x_1, x_2, \ldots, x_n) \equiv (f_1(x_1), f_2(x_2), \ldots, f_n(x_n))^T = 0, \tag{2}$$

where $F : I\!R^n \to I\!R^n$ and $f_i : I\!R \to I\!R$ for $i = 1, \ldots, n$. Let us assume that F is continuously differentiable in a neighborhood of $r \in I\!R^n$ for which $F(r) = 0$, i.e., r solves (2). Let us also assume that $f_i'(r_i) > 0$ for all i. In Section 3 we will see that this last assumption holds when solving (1).

If we apply Algorithm 1 for solving problem (2), where vectors in $I\!R^n$ are interpreted as $n \times 1$ matrices, then the sequence of iterates can be viewed as n independent sequences where each sequence is related to one and only one variable, and all of them use the same step length, as follows:

$$\begin{array}{c} x_{1(k+1)} = x_{1(k)} - \frac{1}{\alpha_k} f_1(x_{1(k)}) \\ x_{2(k+1)} = x_{2(k)} - \frac{1}{\alpha_k} f_2(x_{2(k)}) \\ \vdots \\ x_{n(k+1)} = x_{n(k)} - \frac{1}{\alpha_k} f_n(x_{n(k)}). \end{array} \tag{3}$$

The inverse of the common step length, α_k at Step 6 in Algorithm 1, that will be also denoted as *the slope* of the iteration, can be obtained as

$$\alpha_{k+1} = \frac{\sum_{i=1}^{n}(x_{i(k+1)} - x_{i(k)})(f_i(x_{i(k+1)}) - f_i(x_{i(k)}))}{\sum_{i=1}^{n}(x_{i(k+1)} - x_{i(k)})^2}. \tag{4}$$

Multiplying and dividing each term in the numerator by $x_{i(k+1)} - x_{i(k)}$, we have that

$$\alpha_{k+1} = \frac{\sum_{i=1}^{n}(x_{i(k+1)} - x_{i(k)})^2 \frac{f_i(x_{i(k+1)}) - f_i(x_{i(k)})}{x_{i(k+1)} - x_{i(k)}}}{\sum_{i=1}^{n}(x_{i(k+1)} - x_{i(k)})^2}. \tag{5}$$

Let us define

$$\sigma_{i(k+1)} = \frac{f_i(x_{i(k+1)}) - f_i(x_{i(k)})}{x_{i(k+1)} - x_{i(k)}}. \tag{6}$$

It is worth noticing that $\sigma_{i(k+1)}$ coincides with the slope used by the classical secant method for one variable when solving each one of the $f_i(x_i) = 0$ equations separately. Therefore, if we start sufficiently close to the solution vector r, then for each i

$$\lim_{k\to\infty} \sigma_{i(k)} = f_i^{'}(r_i) > 0. \tag{7}$$

From the formulations in Algorithm 1 we have

$$x_{i(k+1)} - x_{i(k)} = -\frac{1}{\alpha_k} f_i(x_{i(k)}). \tag{8}$$

Using now (8) and substituting $\frac{f_i(x_{i(k+1)}) - f_i(x_{i(k)})}{x_{i(k+1)} - x_{i(k)}}$ by $\sigma_{i(k+1)}$ in (5) we obtain

$$\alpha_{k+1} = \sigma_{1(k+1)} \frac{f_1^2(x_{1(k)})}{\sum_{i=1}^{n} f_i^2(x_{i(k)})} + \sigma_{2(k+1)} \frac{f_2^2(x_{2(k)})}{\sum_{i=1}^{n} f_i^2(x_{i(k)})} + \cdots + \sigma_{n(k+1)} \frac{f_n^2(x_{n(k)})}{\sum_{i=1}^{n} f_i^2(x_{i(k)})}. \tag{9}$$

The following interpretation of (9) is of value: The inverse of the common step length, α_{k+1}, is the weighted average of the individual slopes (secant type) of each one of the one-variable equations, and the weights of each slope is given by the square of the associated residual, $f_i^2(x_{i(k)})$. Hence, at each iteration of Algorithm 1, the common slope approximates the individual slope associated with the equation with the largest residual. Moreover, from (9) the following useful inequality is also obtained

$$min(\sigma_{i(k)}) < \alpha_k < max(\sigma_{i(k)}). \tag{10}$$

Another useful result, that will be needed for our analysis, relates two consecutive residuals for each independent equation. First, let us denote

$$R_{i,k} = f_i(x_{i,k}). \tag{11}$$

We can now write (6) as follows

$$R_{i,k+1} - R_{i,k} = \sigma_{i(k+1)}(x_{i(k+1)} - x_{i(k)}).$$

Using (8) and (11) we now substitute $x_{i(k+1)} - x_{i(k)}$ by $-\frac{1}{\alpha_k}R_{i,k}$ and we obtain

$$R_{i,k+1} - R_{i,k} = -\frac{\sigma_{i(k+1)}}{\alpha_k}R_{i,k},$$

that in turn implies

$$R_{i,k+1} = (1 - \frac{\sigma_{i(k+1)}}{\alpha_k})R_{i,k}. \tag{12}$$

We are now ready to present our convergence analysis. The following line of arguments resembles the one presented in [19] for minimizing convex quadratics. However, since we are solving a different problem, the differences are fundamental. Let us recall that we are assuming that $f_i^{'}(r_i) > 0$ for all i. Hence, by continuity the slope of each equation is also positive near the solution vector $r \in I\!\!R^n$. Let us also assume, without any loss of generality, that the slopes are ordered as follows:

$$0 < \sigma_{1(k)} < \sigma_{2(k)} < \cdots < \sigma_{n(k)}.$$

The analysis for problems in which all the slopes are negative is identical. Our first result is concerned with the convergence behavior of the residual associated with the equation with the smallest slope.

Lemma 1 *If Algorithm 1 is applied for solving (2), and the initial guess is sufficiently close to the solution vector r, then the sequence $\{f_1(x_{1(k)})\}$ converges to zero when k tends to infinity.*

Proof. From (12) it follows for the first one-variable equation that

$$R_{1,k+1} = (1 - \frac{\sigma_{1(k+1)}}{\alpha_k})R_{1,k}. \tag{13}$$

Using (7) and the well-known local properties of the secant method for one variable, there exists ξ, $0 < \xi < 1$, such that

$$\sigma_{1,k+1} = b_k\sigma_{1,k}, \quad 1 - \xi < b_k < 1 + \xi,$$

and using (10) there exists $\zeta > 0$ such that

$$0 < \zeta < \frac{\sigma_{1,k}}{\sigma_{n,k}} < 1.$$

Combining all these previous inequalities we obtain

$$|1-\frac{\sigma_{1(k+1)}}{\alpha_k}| < max(\xi, |1-(1-\xi)\zeta|, |1-(1+\xi)\zeta|) < 1, \quad \forall k,$$

which implies that the sequence $\{f_1(x_{1(k)})\}$ converges to zero. $\square$

Lemma 2 *If* $\lim_{k\to\infty} R_{1,k} = 0,\ \lim_{k\to\infty} R_{2,k} = 0,\ \cdots,$ *and* $\lim_{k\to\infty} R_{l,k} = 0,$ *then*

$$\liminf_{k\to\infty} R_{l+1,k} = 0.$$

Proof. Let us suppose by way of contradiction that there exists $\epsilon > 0$ such that, for all k,

$$R_{l+1,k}^2 > \epsilon. \tag{14}$$

Since we are sufficiently close to the solution, using (7), there exists ξ such that

$$\sigma_{i,k+1} = b_k \sigma_{i,k}, \qquad 1-\xi < b_k < 1+\xi, \qquad 0 < \xi < \frac{2}{5}. \tag{15}$$

From (9) and (11) the common slope can be written as

$$\alpha_{k+1} = \frac{\sigma_{1(k+1)} R_{1,k}^2 + \sigma_{2(k+1)} R_{2,k}^2 + \cdots + \sigma_{n(k+1)} R_{n,k}^2}{\sum_{i=1}^n R_{i,k}^2}. \tag{16}$$

Since the sequences $R_{1,k}^2, \ldots, R_{l,k}^2$ converge to zero, then there exists $\hat{k}$ sufficiently large such that

$$\sum_{i=1}^{l} R_{i,k}^2 \le \frac{1}{4}\epsilon, \qquad \text{for all} \qquad k \ge \hat{k}. \tag{17}$$

Hence, from (9) and (17), it follows that for all $k \ge \hat{k}$

$$\frac{\sigma_{l+1(k+1)} \sum_{i=l+1}^n R_{i,k}^2}{\frac{1}{4}\epsilon + \sum_{i=l+1}^n R_{i,k}^2} \le \alpha_{1(k+1)} \le \sigma_{n,k+1}. \tag{18}$$

Since

$$\sum_{i=l+1}^{n} R_{i,k}^2 \ge R_{l+1,k}^2 \ge \epsilon,$$

and using (18), we have that

$$\frac{4}{5}\sigma_{l+1(k)} \le \alpha_{k+1} \le \sigma_{n,k+1} \quad \text{for all} \quad k \ge \hat{k}. \tag{19}$$

Combining now (19) and (15) it follows that

$$|1-\frac{\sigma_{l+1,k+1}}{\alpha_k}| = |1-\frac{b_k \sigma_{l+1,k}}{\alpha_k}| \le max(\frac{3}{4}, 1-b_k\frac{\sigma_{l+1,k}}{\sigma_{n,k}}) \quad \text{for all} \quad k \ge \hat{k}+1.$$

Sufficiently close to the solution it holds that $0 < \zeta < \frac{\sigma_{l+1,k}}{\sigma_{n,k}} < 1$, and so

$$|1 - \frac{\sigma_{l+1,k+1}}{\alpha_k}| \leq max(\frac{3}{4}, 1 - \frac{2}{5}\zeta) \quad \text{for all} \quad k \geq \hat{k} + 1.$$

Finally using (12) we obtain

$$|R_{l+1,k+1}| = |(1 - \frac{\sigma_{l(k+1)}}{\alpha_k})||R_{l+1,k}| \leq \hat{c}|R_{l+1,k}|,$$

where

$$\hat{c} = max(\frac{3}{4}, 1 - \frac{2}{5}\zeta) < 1, \tag{20}$$

which contradicts (14), and hence the result holds. $\square$

Theorem 1 *Consider Algorithm 1 for solving problem (2). Let us assume that F is continuously differentiable in a neighborhood of $r \in I\!R^n$ for which $F(r) = 0$ and $f_i'(r_i) > 0$, for $1 \leq i \leq n$. If we start sufficiently close to r then the sequence $\{x_{i(k)}\}$ converges to r.*

Proof. Let us assume, without any loss of generality, that $\alpha_{1,k} \leq \alpha_{2,k} \leq \cdots \leq \alpha_{n,k}$. We prove that $R_{p,k}$ converges to zero for $1 \leq p \leq n$ by induction on p. From Lemma 1 we have that $\lim_{k\to\infty} R_{1,k} = 0$. Consider an integer p from the interval $2 \leq p \leq n$, and let us assume that $R_{1,k}, \cdots, R_{p-1,k}$ converge to zero. Therefore, for any given $\epsilon > 0$ there exists $\hat{k}$ sufficiently large such that

$$\sum_{i=1}^{p-1} R_{i,k}^2 \leq \frac{1}{4}\epsilon, \quad \text{for all} \quad k \geq \hat{k}. \tag{21}$$

From (16) and (21) we obtain

$$\frac{\sigma_{p,k+1} \sum_{i=p}^{n} R_{i,k}^2}{\frac{1}{4}\epsilon + \sum_{i=p}^{n} R_{i,k}^2} \leq \alpha_{k+1} \leq \sigma_{n,k+1}, \tag{22}$$

for all $k \geq \hat{k}$. Using now Lemma 2 there exists $k_p \geq \hat{k}$ such that

$$R_{p,k_p}^2 < \epsilon.$$

Now, let us say that $k_0 > k_p$ is any integer for which $R_{p,k_0-1}^2 < \epsilon$ and $R_{p,k_0}^2 \geq \epsilon$. Clearly,

$$\sum_{i=p}^{n} R_{i,k}^2 \geq R_{p,k}^2 \geq \epsilon \quad \text{for} \quad k_0 \leq k \leq j - 1, \tag{23}$$

where j is the first integer greater than k_0 for which $R_{p,j}^2 < \epsilon$. Hence, from (22) and (23), we have that

$$\frac{4}{5}\sigma_{p,k+1} \leq \alpha_{k+1} \leq \sigma_{n,k} \quad \text{for} \quad k_0 \leq k \leq j - 1. \tag{24}$$

Using now (24) and equation (12) applied to p, it follows that

$$|R_{p,k+2}| \leq \hat{c}|R_{p,k+1}| \quad \text{for} \quad k_0 \leq k \leq j-1,$$

where $\hat{c}$ is the constant given by (20) that satisfies $\hat{c} < 1$.

Finally using (10), (12), and the fact that $0 < \zeta < \frac{\sigma_{1,k}}{\sigma_{n,k}} < 1$, the following bound can be obtained

$$|R_{p,k_0+1}| \leq (\frac{1}{\zeta} - 1)^2 |R_{p,k_0-1}|,$$

which in turn implies

$$(R_{p,k_0+1})^2 \leq (\frac{1}{\zeta} - 1)^4 (R_{p,k_0-1})^2 \leq (\frac{1}{\zeta} - 1)^4 \epsilon,$$

for all $k_0 + 1 \leq k \leq j+1$. From the conditions on k_0 and j, it follows that $R^2_{p,k}$ is bounded from above by a multiple of ϵ. Since $\epsilon > 0$ can be chosen arbitrarily small, we conclude that $\lim_{k\to\infty} R_{p,k} = 0$, which completes the proof. □

3 Analysis for computing the matrix p-th root

We will present an analysis of the residual algorithm for computing the SPD matrix root of a given SPD matrix A. First, we will establish that Algorithm 1 applied to $X^p - A = 0$, from an initial guess X_0 that commutes with A, generates the same sequences $\{\alpha_k\}$ and $\{R_k\}$ that it generates from $Z_0 = Q^T X_0 Q$ when applied to the simplified problem $Z^p - \Lambda = 0$, where Z and $\Lambda = Q^T A Q$ are both diagonal matrices. In here, Q is a suitable orthogonal matrix. For the sake of clarity and completeness let us now present both algorithms.

Algorithm 2 Residual algorithm for $X^p - A = 0$

```
1: Let $X_0 \in I\!R^{n\times n}$ commuting with $A, \alpha_0 \in I\!R$, $\alpha_0 \neq 0$
2: for $k = 0, 1, \cdots$ do
3:     $X_{k+1} = X_k - (1/\alpha_k)(X_k^p - A)$
4:     $S_k = X_{k+1} - X_k$
5:     $Y_k = X_{k+1}^p - X_k^p$
6:     $\alpha_{k+1} = \langle S_k, Y_k\rangle / \langle S_k, S_k\rangle$
7: end for
```

Algorithm 3 Residual algorithm for $Z^p - \Lambda = 0$

```
1: Let $Z_0 = Q^T X_0 Q \in I\!R^{n\times n}, \sigma_0 \in I\!R$, $\sigma_0 \neq 0$
2: for $k = 0, 1, \cdots$ do
3:     $Z_{k+1} = Z_k - (1/\sigma_k)(Z_k^p - \Lambda)$
4:     $\widehat{S}_k = Z_{k+1} - Z_k$
5:     $\widehat{Y}_k = Z_{k+1}^p - Z_k^p$
6:     $\sigma_{k+1} = \langle \widehat{S}_k, \widehat{Y}_k\rangle / \langle \widehat{S}_k, \widehat{S}_k\rangle$
7: end for
```

We are now ready to establish our main convergence result.

Theorem 2 *If $\alpha_0 = \sigma_0$, and $X_0 = QZ_0Q^T$ where Z_0 is a diagonal matrix, then $\alpha_k = \sigma_k$ and $X_k = QZ_kQ^T$, for all k, and hence the sequence $\{X_k\}$ generated by Algorithm 2 converges to X_r if and only if the sequence $\{Z_k\}$ generated by Algorithm 3 converges to Z_r, where $Z_r = Q^T X_r Q$.*

Proof. Let us show by induction on k that $\alpha_k = \sigma_k$ and $X_k = QZ_kQ^T$ for all k. It clearly holds for $k = 0$. Let us assume that $\alpha_k = \sigma_k$ and $X_k = QZ_kQ^T$. Therefore,

$$X_{k+1} = QZ_kQ^T - \frac{1}{\sigma_k}(QZ_k^pQ^T - Q\Lambda Q^T) = Q(Z_k - \frac{1}{\sigma_k}(Z_k^p - \Lambda))Q^T = QZ_{k+1}Q^T.$$

Since $trace(AB) = trace(BA)$, $Q^TQ = I$, $S_k = Q\widehat{S}_kQ^T$ and $Y_k = Q\widehat{Y}_kQ^T$, then

$$\alpha_{k+1} = \frac{trace(S_k^TY_k)}{trace(S_k^TS_k)} = \frac{trace(Q\widehat{S}_k^TQ^TQ\widehat{Y}_kQ^T)}{trace(Q\widehat{S}_k^TQ^TQ\widehat{S}_kQ^T)} = \frac{trace(\widehat{S}_k^T\widehat{Y}_k)}{trace(\widehat{S}_k^T\widehat{S}_k)} = \sigma_{k+1}.$$

Consequently for all k

$$\|X_k - X_r\|_F = \|Z_k - Z_r\|_F, \tag{25}$$

where X_r is the limit of the sequence generated by Algorithm 2, and Z_r s the limit of the sequence generated by Algorithm 3. In here, for any square matrix A, $\|A\|_F^2 = trace(A^TA)$. Finally, from (25), the result holds. □

Another consequence of the similarity between the matrices X_k and Z_k is the following: If the solution of the original problem, X_r, is SPD then the solution of the simplified problem, Z_r, is also SPD. Hence, $Z_r(i,i) > 0$ for all $i = 1\cdots n$, and so the derivative of each independent simplified equation is also positive, i. e., $pZ_r(i,i)^{p-1} > 0$ for $i = 1\cdots n$. Our next result establishes the local convergence of Algorithm 2 for finding the matrix p-th root.

Theorem 3 *Let A be a given SPD matrix. If X_0 is chosen sufficiently close to X_r, the SPD solution of $X^p - A = 0$, then the sequence $\{X_k\}$ generated by Algorithm 2 converges to X_r.*

Proof. The assumptions of Theorem 1 are satisfied for solving the simplified problem $Z^p - \Lambda = 0$, whose variables separate, since $pZ_r(i,i)^{p-1} > 0$ for $i = 1 \cdots n$. Therefore, the sequence $\{Z_k\}$ generated by Algorithm 3 converges to Z_r. Hence, from Theorem 2 the sequence $\{X_k\}$ generated by Algorithm 2 converges to $X_r = QZ_rQ^T$, the SPD solution of $X^p - A = 0$. $\square$

We would like to close this section discussing the practical choice of the initial data in Algorithm 2. If the general Algorithm 1 is applied to a linear problem whose variables separate, then the global convergence is guaranteed from any initial X_0 and any initial $\alpha_0 > 0$, for two reasons: The solution is unique and the slope for each independent equation is constant. Unfortunately, this is not the case when solving nonlinear problems whose variables separate. In general, the set of assumptions considered for Theorem 1 hold only in a neighborhood of the solution, and the size of that neighborhood is problem dependent. For the problem of computing the matrix p-th root, the size of the neighborhood of global convergence decreases when p increases. Indeed, the size of the neighborhood for which the sign of the independent first derivatives remain unchanged decreases when p increases. In addition to that, we have another difficulty: The real p-th root of a given SPD matrix is not unique when p is even, and so, if the initial data is not chosen properly, the method could converge to a solution which is not SPD. In this work, for choosing the initial data we have used the following effective procedures.

We choose $X_0 = \kappa_1 I + \kappa_2 A$, where κ_1 and κ_2 are obtained such that the smallest and largest eigenvalues of X_0 coincide with the smallest and largest eigenvalues of $\sqrt[p]{A}$, which clearly guarantees that X_0 is SPD and commutes with A. For that we need to obtain the smallest and the largest eigenvalues of A, which can be accomplished using the ARPACK library, or the `eigs` function from Matlab. To be precise, if λ_{min} and λ_{max} are the extreme eigenvalues of A, then κ_1 and κ_2 should satisfy $\sqrt[p]{\lambda_{min}} = \kappa_1 + \kappa_2\lambda_{min}$ and $\sqrt[p]{\lambda_{max}} = \kappa_1 + \kappa_2\lambda_{max}$. After that, the initial α_0 is chosen as

$$\alpha_0 = 0.8 * p\sqrt[p]{(\lambda_{max})^{p-1}},$$

that represents 80% of the largest slope at the solution out of all the independent equations of the simplified problem.

Using these specialized procedures for choosing X_0 and α_0 we have observed global convergence of Algorithm 2 in our numerical experiments. Nevertheless, for the sake of robustness, we recommend to incorporate the Nonmonotone Derivative-Free (NDF) line search, proposed and analyzed in [16], for the residual function ($\|F(X_k)\|_F$) with the parameters $M = 10$, $\gamma = 1.D-4$, and $\eta = 0$. Roughly speaking the globalization strategy should reject X_{k+1}, and backtracks in the direction of $-F(X_k)$, when the current residual is greater than the largest of the previous M residuals. The advantages of this globalization strategy are that it rejects the current point very seldom and guarantees global convergence. Therefore, in practice very few backtrackings will be needed. The globalized version of Algorithm 2 is presented in Algorithm 4.

Algorithm 4 Globalized Residual algorithm for $X^p - A = 0$

```
1: Let $X_0 \in I\!R^{n\times n}$ commuting with $A, \alpha_0 \in I\!R$, $\alpha_0 \neq 0$
2: Let $\gamma = 1.D-04,\ M = 10$
3: for $k = 0, 1, \cdots$ do
4:     $R_k = X_k^p - A$
5:     $f_k = \|R_k\|_F$
6:     $\overline{f}_k = \max_{0\leq j\leq min(k,M)} f_{k-j}$
7:     $X_+ = X_k - (1/\alpha_k)R_k$
8:     while $\|X_+^p - A\|_F > (\overline{f}_k - \gamma * (\alpha_k^2) * f_k)$ do
9:         $\alpha_k = 2\alpha_k$                                  ▷ Backtracking
10:        $X_+ = X_k - (1/\alpha_k)R_k$
11:    end while
12:    $X_{k+1} = X_+$
13:    $S_k = X_{k+1} - X_k$
14:    $Y_k = X_{k+1}^p - X_k^p$
15:    $\alpha_{k+1} = \langle S_k, Y_k\rangle / \langle S_k, S_k\rangle$
16: end for
```

As mentioned before, the step length α_k obtained at Step 15 will be accepted most of the time (without backtracking) at Step 12. The computational cost of Algorithm 4 is dominated by the residual evaluation. If we use the binary powering method to evaluate X^p the computational cost is dominated by $[\log_2 p] + \mu - 1$ matrix-matrix symmetric multiplications per iteration, where μ is the number of 1's in the binary representation of p (see [9] page 72); i.e., the cost is $([\log_2 p] + \mu - 1)n^3)$. However, it is worth mentioning that using the Strassen fast multiplication algorithm [28] it can be reduced to $O(([\log_2 p] + \mu - 1)n^{\log_2 7})$.

4 Some related problems

The general residual method (Algorithm 1), and Theorem 1, can be applied to some other interesting problems that can be written equivalently as a nonlinear problem whose variables separate, and for which the first derivatives of each independent equation have the same sign at the solution.

For example, consider the following unconstrained optimization problem

$$\min \sum_{i=1}^{n} (e^{x_i} - x_i), \tag{26}$$

that is equivalent to solving the following n independent nonlinear equations,

$$e^{x_i} = 1,$$

for $1 \leq i \leq n$. The function in (26) is strictly convex, the Hessian matrix is diagonal for all x, and indeed the Barzilai-Borwein method converges globally in its pure form [20], i. e., without any globalization strategy. Theorem 1 explains this behavior.

Another interesting problem is to approximate the inverse of an SPD matrix A, that can be formulated as solving the linear map

$$AX - I = 0.$$

The iterations of the general residual algorithm are given by

$$X_{k+1} = X_k - \frac{1}{\alpha_k}(AX_k - I).$$

If we perform the change of variables $Q^T X_k Q = Y_k$, where Q is an orthogonal matrix such that $Q^T AQ = \Lambda$ where Λ is a diagonal matrix, then the iterations are written as

$$Y_{k+1} = Y_k - \frac{1}{\alpha_k}(\Lambda Y_k - I),$$

which is equivalent to solving $\Lambda Y - I = 0$ whose solution is clearly diagonal. Therefore, Algorithm 1 has global convergence from any initial guess. This result holds, in general, when solving the linear system $Ax = b$ for any SPD matrix A, which has been established in [19].

The same approach can also be applied when solving the quadratic equation

$$X^2 + BX + C = 0,$$

where B and C are symmetric and commute; and where the matrix $B^2 - 4C$ is symmetric and positive definite. In this case, the equation has an explicit solution given by

$$X_r = (-B + (B^2 - 4C)^{1/2})/2.$$

This specific quadratic equation can be solved using Algorithm 1 since the set of assumptions guarantees the existence of an orthogonal matrix Q that simultaneously diagonalizes B and C. If we apply Q on the left and also on the right we obtain an equivalent quadratic equation for which all the involved matrices are diagonal. Therefore the problem is equivalent to a problem whose variables separate. If we consider the SPD square root of $B^2 - 4C$ then the derivatives of each independent equation are positive, and so Theorem 1 can be applied.

5 Numerical experiments

We now present some numerical experiments that illustrate the behavior of Algorithm 4 for some values of p. In all our experiments X_0 and α_0 are chosen using the specialized procedures described in Section 4. The tests were carried out in MATLAB 6.5 on a 1.6 GHz Pentium IV.

First we show the behavior of the norm of the residual vectors $\|R_k\|_F = \|X_k^p - A\|_F$ (Figure 1), and the inverse of the common step lengths α_k (Figure 2), when computing the square root of $A = gallery('moler', 16)$, which is a 16×16 SPD matrix, whose condition number is $\kappa_A = 4.1723e + 010$. We can observe the nonmonotone behavior of the algorithm that accounts for the fast convergence. We also observe that the sequence $\{1/\alpha_k\}$ oscillates between two values that correspond to the smallest and the largest slopes of the individual equations.

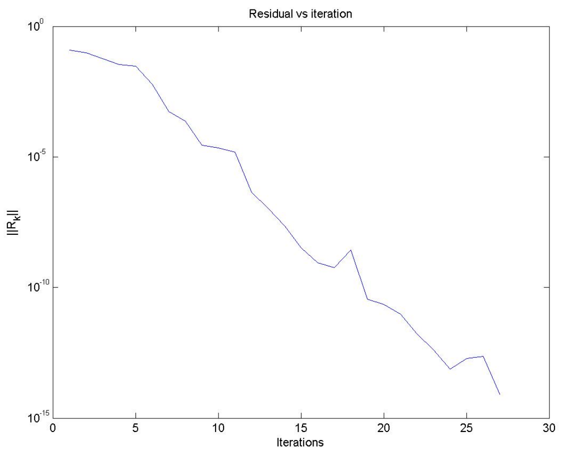

Figure 1: Residual behavior for $A = gallery('moler', 16)$ when $p = 2$

In Figure 3 we show the behavior of the norm of the residual when computing the cubic root of $A = gallery('moler', 16)$. Once again we observe the nonmonotone behavior of Algorithm 4.

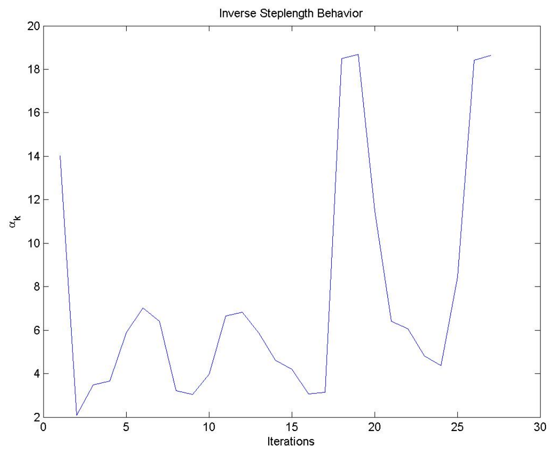

Figure 2: Behavior of the inverse of the step length for $A = gallery('moler', 16)$ when $p = 2$

For our next experiment we generate several diagonal matrices using the MATLAB

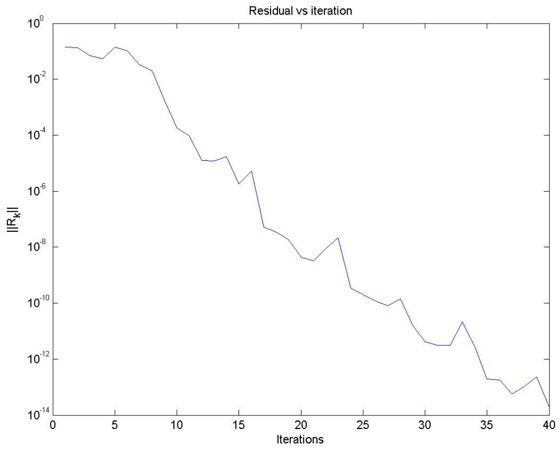

Figure 3: Residual behavior for $A = gallery('moler', 16)$ when $p = 3$

function $lineal(n, \kappa_A)$ with eigenvalues uniformly distributed between 1 and κ_A, where κ_A varies from $1.e+3$ to $1.e+9$. In tables 1 ($p = 2$) and 2 ($p = 3$) we report the matrix A, the size n, the condition number κ_A, the number of iterations, the computational cost (Cost) reported in number of matrix-matrix products (M'), and the relative residual $Rr = \frac{\|X^p - A\|_\infty}{\|A\|_\infty}$. For these experiments the stopping criterion is $\frac{\|X_{k+1}-X_k\|}{\|X_k\|} < 1e-14$.

A	n	κ_A	Iterations	Cost	Rr
gallery(morel',16)	16	4.1723e+010	28	$28M'$	7.1804e-015
lineal(100,1E03)	100	1E03	51	$51M'$	3.5202e-014
lineal(100,1E06)	100	1E06	58	$58M'$	7.5670e-015
lineal(100,1E09)	100	1E09	51	$51M'$	9.3913e-014
lineal(500,1E03)	500	1E03	78	$78M'$	1.1966e-014
lineal(500,1E06)	500	1E06	95	$95M'$	2.7181e-014
lineal(500,1E09)	500	1E09	92	$92M'$	2.5799e-014

Table 1: Behavior of Algorithm 4 for computing the square root

From Tables 1 and 2 we observe that the number of required iterations increases when p increases from 2 to 3. This result should not be a surprise since the condition of the derivative of $F(X) = X^p - A$ at the solution $\sqrt[p]{A}$ is given by

$$\kappa(F'(\sqrt[p]{A})) = \kappa(A)^{\frac{p-1}{p}},$$

which indeed increases with p. We can also observe that the computational cost increases when the dimension and the condition number of A increase. Concerning the globalization strategy, we have noticed in all our experiments that, indeed, very few backtrackings are required during the convergence process (none in most experiments, and very seldom 1 or 2 at most).

A	n	κ_A	Iterations	Cost	Rr
gallery(morel',16)	16	4.1723e+010	42	$84M'$	1.4204e-015
lineal(100,1E03)	100	1E03	81	$162M'$	1.1781e-013
lineal(100,1E06)	100	1E06	82	$164M'$	9.7789e-015
lineal(100,1E09)	100	1E09	85	$170M'$	3.5282e-014
lineal(500,1E03)	500	1E03	114	$228M'$	9.6582e-014
lineal(500,1E06)	500	1E06	136	$272M'$	1.2022e-014
lineal(500,1E09)	500	1E09	148	$296M'$	7.6175e-014

Table 2: Behavior of Algorithm 4 for computing the cubic root

We close this section by mentioning Smith's method which is the method of choice for computing pth roots of matrices. Smith's algorithm (Algorithm 5) is based on the Schur's factorization. Clearly, for SPD matrices it is significantly simplified since the triangular matrix becomes diagonal.

Algorithm 5 Smith's Algorithm for SPD matrices

```
1: $[Q, D] = schur(A)$
2: $\Lambda = D^{\frac{1}{p}}$
3: $R = Q\Lambda Q^t$
```

Smith's method achieves a high precision in the relative residual of the order of $1e-15$ [9]. From Tables 1 and 2 we observe that the same precision can be obtained by Algorithm 4. However, the computational cost of Algorithm 5 is $28n^3$ [9], which is equivalent to the cost of 28 matrix-matrix products ($28M'$), i.e., the cost of 28 iterations of the new method. It is worth mentioning that Algorithm 5 cannot find an approximate solution with a lower precision. In contrast, if only a relative residual of order $1e-6$ is required, then Algorithm 4 can be stopped prematurely to accomplish it. In Table 3, for example, we show the behavior of Algorithm 4 after 20 iterations for computing the square root of several matrices. In those cases the computational cost is fixed at $20M'$ and the required low precision is obtained.

A	n	κ_A	Cost	Rr
lineal(100,1E03)	100	1E03	$20M'$	3.5501e-006
lineal(100,1E06)	100	1E06	$20M'$	6.4818e-006
lineal(100,1E09)	100	1E09	$20M'$	6.4974e-006

Table 3: Behavior of Algorithm 4 for computing the square root after 20 iterations

6 Final remarks

We have presented and analyzed a low-cost residual algorithm for computing the p-th root of a given SPD matrix. It has local convergence properties in its pure form (Algorithm

2), and global convergence in its globalized form (Algorithm 4). The new method is more efficient and requires fewer iterations for convergence when p is small, since the condition number of the derivative of the residual map, $X^p - A$, increases with p. Our results indicate that the new scheme is robust in general, and suitable for large-scale problems for which high accuracy is not required.

The new residual method, as well as its convergence analysis, can be applied to any problem that can be written equivalently as a nonlinear problem whose variables separate. It is also worth mentioning that these type of residual methods are well-suited for parallel architectures, since only matrix-matrix products are required.

In this work, we have considered the SPD case, which is clearly the best possible scenario for Smith's method. In the near future we would like to study the behavior of these type of residual methods for computing the p-th root of nonsymmetric matrices.

References

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

[2] D. A. Bini, N. J. Higham and B. Meini (2005). Algorithms for the matrix p-th root. *Numerical Algorithms* 39, 349–378.

[3] E. Birgin, J. M. Martínez and M. Raydan (2009). Spectral Projected Gradient Methods. In *Encyclopedia of Optimization, Second Ed.*, Editors Editors: C. A. Floudas and P. M. Pardalos, Part 19, pp. 3652-3659.

[4] A. Bjorck and S. Hammarling (1983). A Schur methods for the square root of a matrix. *Linear Algebra and its Applications* 52/53, 127–140.

[5] J. P. Chehab and M. Raydan (2005). Implicit and adaptive inverse preconditioned gradient methods for nonlinear problems. *Applied Numerical Mathematics* 55, 32–47.

[6] S. H. Cheng, N. J. Higham, C. S. Kenney and A. J. Laub (2001). Approximating the logarithm of a matrix to specified accuracy. *SIAM J. Matrix Anal. Appl.* 22, 1112–1125.

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

[8] B. De Abreu, M. Monsalve and M. Raydan (2008). Specialized hybrid Newton schemes for matrix p-th roots. *Applied Mathematical Sciences*, 2, No. 49, 2401–2424.

[9] N. J. Higham (2008), *Functions of Matrices: Theory and Computation*, SIAM, USA.

[10] N. J. Higham (1986). Newton's methods for the matrix square root. *Mathematics of Computation* 46, 537–549.

[11] N. J. Higham (1997). Stable iterations for the matrix square root. *Numerical Algorithms* 15, 227–242.

[12] B. Iannazzo (2003). A note on computing the matrix square root. *Calcolo* 40, 273–283.

[13] B. Iannazzo (2007). On the Newton method for the matrix p-*th* root. *SIAM J. Matrix Anal. Appl.* 28, 503–523.

[14] C. S. Kenney and A. J. Laub (1989). Padé error estimates for the logarithm of a matrix. *Internat. J. Control* 50, 707–730.

[15] W. La Cruz and M. Raydan (2003). Nonmonotone Spectral Methods for Large-Scale Nonlinear Systems, *Optimization Methods and Software*, 18, 583–599.

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

[17] B. Meini (2004). The matrix square root from a new functional perspective: theoretical results and computational issues. *SIAM J. Matrix Anal. Appl.* 26, 362–376.

[18] M. Monsalve and M. Raydan (2008). A secant method for nonlinear matrix problems. In: P. Van Dooren and A. Routray (Eds.), Numerical Linear Algebra in Signals, Systems and Control, Springer Verlag, Amsterdam, Holland, to appear.

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

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

[21] M. Raydan (2004). Low-cost methods for nonlinear large-scale matrix equations. In: Y. Yuan(Ed.), Numerical linear algebra and optimization, Science Press, Beijing, China, pp. 79–87.

[22] L. S. Shieh, Y. T. Tsay and C. T. Wang (1984). Matrix sector functions and their application to system theory. *IEEE Proc.* 131, 171–181.

[23] L. S. Shieh, S. R. Lian and B. C. Mcinnis (1987). Fast and stable algorithms for computing the principal square root of a complex matrix. *IEEE Transactions on Automatic Control* AC-32, 820–822.

[24] L. S. Shieh, Y. T. Tsay and R. E. Yates (1985). Computation of the principal nth roots of complex matrices. *IEEE Transactions on Automatic Control* AC-30, 606–608.

[25] M. I. Smith (2003). A Schur algorithm for computing matrix pth roots, *SIAM J. Matrix Anal. Appl.* 24, 971–989.

[26] J. S. H. Tsai, L. S. Shieh, and R. E. Yates (1988). Fast and stable algorithms for computing the principal nth root of a complex matrix and the matrix sector function. *Comput. Math. Applic.* 15, 903–913.

[27] Y. T. Tsay, L. S. Shieh, and J. S. H. Tsai (1986). A fast method for computing the principal nth roots of complex matrices. *Linear Alg. Appl.* 76, 205–221.

[28] V. Strassen (1969). Gaussian Elimination is not Optimal. *Numer. Math.* 13, 354–356.