Fixed Point Theory, 24(2023), No. 1, 295-308
DOI: 10.24193/fpt-ro.2023.1.16
http://www.math.ubbcluj.ro/~nodeacj/sfptcj.html

SCALED FIXED POINT ALGORITHM FOR COMPUTING THE MATRIX SQUARE ROOT

HARRY OVIEDO*, HUGO LARA** AND OSCAR DALMAU***

*Facultad de Ingenieria y Ciencias, Universidad Adolfo Ibáñez (UAI)
Santiago de Chile, Chile
E-mail: harry.oviedo@uai.cl

**Universidade Federal de Santa Catarina, Blumenau, SC. Brazil
E-mail: hugo.lara.urdaneta@ufsc.br

***Centro de Investigación en Matemáticas, CIMAT A.C. Guanajuato, Mexico
E-mail: dalmau@cimat.mx

Abstract. This paper addresses the numerical solution of the matrix square root problem. Two fixed point iterations are proposed by rearranging the nonlinear matrix equation $A - X^2 = 0$ and incorporating a positive scaling parameter. The proposals only need to compute one matrix inverse and at most two matrix multiplications per iteration. A global convergence result is established. The numerical comparisons versus some existing methods from the literature, on several test problems, demonstrate the efficiency and effectiveness of our proposals.
Key Words and Phrases: Matrix square root, fixed point algorithm, matrix iteration and geometric optimization.
2020 Mathematics Subject Classification: 65J15, 65F45, 65H10.

1. Introduction

In this paper, we derive novel fixed point algorithms for numerically approximating the solution of the matrix square root problem. Given a square matrix $A \in \mathbb{R}^{n\times n}$, we address the problem of finding a matrix $X \in \mathbb{R}^{n\times n}$ such that it satisfies the following quadratic system of equations

$$X^2 = A. \tag{1.1}$$

Our approach lies on the special case of (1.1) when A is a symmetric positive semi–definite (PSD) matrix with real entries. The matrix square root problem plays an important role in many applications, and arises in several contexts such as: computation of the matrix sign function [8], signal processing applications [16, 21], parallel translation and polar retractions for optimization on Riemannian manifolds [9, 15, 22, 23], the Karcher mean computation [9], among others.

It is well–known that the system (1.1) does not have a unique solution (if one exists). However, if A is positive semi–definite then problem (1.1) has exactly a

unique positive semi–definite solution, denoted by $A^{1/2}$, which is called *the principal square root of* A, see [2]. This seems to be the most frequent case in practice.

The most numerically stable way to solve problem (1.1) is via the Schur decomposition. This strategy reduces the problem (1.1) into the computation of the matrix square root of an upper triangular matrix. Specifically, let $A = UTU^*$ a Schur decomposition of A, where T is upper triangular and U is unitary. Then, observe that $A^{1/2} = UT^{1/2}U^*$. A blocked Schur procedure for solving (1.1) is presented in [3]. When $A \in \mathbb{R}^{n\times n}$ is symmetric, this method is reduced to compute an eigenvalue decomposition of A. However, this strategy is impractical for n large. Therefore, there is not other option than to resort to iterative methods.

Several types of iterative algorithms have been introduced to address Problem (1.1). Possibly the first one is the Newton method, developed by Higham in [6], which constructs a sequence of iterates by the following recurrence

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A), \quad \text{starting at} \quad X_0 = A. \tag{1.2}$$

This method enjoys a quadratic convergence rate to $A^{1/2}$ under some assumptions, see [6]. However, Newton iteration suffers from instability near the solution, and absence of global convergence. In an attempt to overcome these drawbacks, in [8] is introduced a scaled Newton method to approximate the solution of (1.1) via polar decomposition. First, it is computed the Cholesky factorization $A = LL^\top$ to obtain the square root as $A^{1/2} = UL^\top$, where U is the limit of the sequence $\{U_k\}$ generated by

$$U_{k+1} = \frac{1}{2}\left(\mu_k U_k + \frac{1}{\mu_k}U_k^{-\top}\right), \quad \text{starting at } U_0 = L, \tag{1.3}$$

where $\mu_k > 0$ is the scaling parameter.

In [19], Sra developed a fixed point iteration, which is related to a non–convex optimization problem. Starting at $X_0 = \frac{1}{2}(A + I)$, Sra's iteration solve (1.1) by running

$$X_{k+1} = [(X_k + A)^{-1} + (X_k + I)^{-1}]^{-1}. \tag{1.4}$$

This iteration was also considered in [1], in the context of geometric mean computation of positive operators, motivated by electrical resistance networks. In [19], Sra established the linear convergence of (1.4) to $A^{1/2}$ based on a geometric optimization approach. Specifically, Sra cast Problem (1.1) as the following non–convex optimization model

$$\min_{X\succ 0} \mathcal{F}(X) = \delta_S^2(X, A) + \delta_S^2(X, I), \tag{1.5}$$

whose unique solution is the desired square root $X^* = A^{1/2}$. Here, $\delta_S^2(\cdot,\cdot)$ denotes the S–divergence (see [20]) defined by

$$\delta_S^2(X, Y) = \log\det\left(\frac{X+Y}{2}\right) - \frac{1}{2}\log\det\left(\frac{XY}{2}\right). \tag{1.6}$$

The first–order optimality conditions associated with (1.5), implies the following matrix equation,

$$\frac{1}{2}\left(\frac{X+A}{2}\right)^{-1} + \frac{1}{2}\left(\frac{X+I}{2}\right)^{-1} - X^{-1} = 0. \tag{1.7}$$

Direct manipulation of this Riccati equation leads to the Sra's iteration (1.4).

Another first–order method was proposed in [17]. Namely, the classical steepest descent method (SD) for minimizing the least–square problem

$$\min ||X^2 - A||_F^2, \quad s.t. \quad X \succeq 0, \tag{1.8}$$

related to Problem (1.1). The main advantage of the steepest descent method to solve (1.8) over the Newton and Sra methods, is that the SD method does not require computing a matrix inverse per iteration, which makes the SD method an attractive procedure. However, the optimal step–size depends on a certain constant $c > 0$, whose existence is theoretically guaranteed (see Theorem 3.2 in [17]), leading to the absence of a closed formula for the step–size in practice. Although strategies of sufficient descent as the standard Armijo–rule with a backtracking strategy can be used, this could cause the SD algorithm to perform many matrix multiplications per iteration, which is not desired to design an efficient method.

Recently, Gawlik in [5] introduced the Zolotarev iterations for finding the matrix square root, this approach is based on a recursion for rational approximations of $\sqrt{a}$, which is extended to the matrix case. This procedure is similar to the Padè's iterations [7, 11], but converges more rapidly to the solution for matrices that have eigenvalues with widely varying magnitudes.

In this paper, we introduce two new first–order fixed point methods to compute a numerical solution of (1.1). Based on the Sra' iteration (1.4) and keeping in mind the high computational cost per iteration of (1.4), we propose some fixed point methods equipped with a scaling parameter, which only need to compute one inverse matrix and at most two matrix multiplications per iteration. In addition, we establish a global convergence result under the Thompson metric, following the idea of the Sra' demonstration in [19]. Furthermore, we perform some numerical comparisons between our proposals and other state–of–the–art methods, in order to demonstrate the effectiveness of our procedures. Several numerical experiments show that our proposals are more efficient than the Sra iteration and also converge faster to the solution of (1.1).

The rest of this paper is organized as follows. In section 2, we introduce our two fixed point iterations for solving the square root problem (1.1). A convergence analysis is given in section 3. Some numerical tests on several experimental problems are presented in section 4. Finally, conclusions are drawn in section 5.

2. Two fixed point methods

In this section, we introduce two fixed point methods to deal with the numerical solution of the matrix square root problem. Motivated by the Sra' iteration and looking for numerical efficiency, we construct new iterative schemes from the matrix equation (1.1). Let μ be a positive parameter (conveniently chosen). Adding the term μX on both sides of the equation (1.1), and then post–multiplying by the inverse matrix $(X + \mu I)^{-1}$, we arrive at

$$X = (A + \mu X)(X + \mu I)^{-1}, \tag{2.1}$$

which leads us to our first fixed point iteration, starting at an initial symmetric positive semi–definite matrix $X_0 \in \mathbb{R}^{n\times n}$,

$$X_{k+1} = (A + \mu X_k)(X_k + \mu I)^{-1}, \quad k = 0, 1, 2, \dots \tag{2.2}$$

Following a similar reasoning, we can build another fixed point iteration to address problem (1.1). From $A = XX$, multiplying this equation by $X^\top$, adding μX and then multiplying by $(X^\top X + \mu I)^{-1}$ we obtain,

$$X = (X^\top X + \mu I)^{-1}(X^\top A + \mu X), \tag{2.3}$$

which suggests the following fixed point iterative process,

$$X_{k+1} = (X_k^\top X_k + \mu I)^{-1}(X_k^\top A + \mu X_k), \quad k = 0, 1, 2, \dots \tag{2.4}$$

From equations (2.2) and (2.4), we note that our approaches are computationally less costly than the Sra' iteration due to our iterative procedures only require to compute a matrix inverse per iteration, while the algorithm (1.4) needs three. In addition, our proposals incorporate a scale parameter that, if its properly selected, can speed up the convergence. Furthermore, comparing both iterative processes (2.2) and (2.4), it is clear that each iteration of our first method is computationally more efficient than our scheme (2.4), since it requires fewer matrix multiplications. Now we describe our efficient fixed point iterative algorithm.

Algorithm 1 Fixed Point Method (FPM)

Require: $A \in \mathbb{R}^{n\times n}$ a given positive definite matrix, $X_0 \in \mathbb{R}^{n\times n}$, $\mu > 0$, $\epsilon \in (0, 1)$, $k = 0$.

Ensure: An ϵ–approximate solution of the system of equations (1.1)

1: **while** $||A - X_k^2||_F/||A||_F > \epsilon$ **do**
2: $\quad X_{k+1} = (A + \mu X_k)(X_k + \mu I)^{-1}$;
3: $\quad k = k + 1$;
4: **end while**
5: $X^* = X_k$.

Remark 2.1 By changing line 2 of Algorithm 1 by the update formula (2.4), we get our second fixed point iterative method. In this work, we will analyze this second variant only from a numerical point of view.

3. Convergence analysis

We now analyze Algorithm 1 by revealing the behaviour of the residual $\delta_T(X_k, X_*)$, where δ_T denotes the Thompson metric and X^* is the solution of (1.1). The theoretical results provided here use similar tools than [19]. Specifically, at the end of this section, we prove that our scheme (2.2) is a fixed–point iteration under the Thompson part metric defined by

$$\delta_T(X, Y) = ||\log(X^{-\frac{1}{2}} Y X^{-\frac{1}{2}})||_2, \tag{3.1}$$

where $||\cdot||_2$ is the usual matrix norm and "log" denotes the matrix logarithm. In the rest of this article, we will denote by $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ the smallest and largest real part of the eigenvalues of the square matrix $M \in \mathbb{R}^{n\times n}$, respectively.

The following lemma provides us some remarkable properties associated to the Thompson metric. For details about Lemma 3.1 please see [12, 13, 14, 18].

Lemma 3.1 [Proposition 4.2 in [18]]. Consider the Thompson metric defined in (3.1). Let $A, B, X, Y \in \mathbb{R}^{n\times n}$ be symmetric positive definite matrices, then the following properties hold

$$\delta_T(X^{-1}, Y^{-1}) = \delta_T(X, Y), \tag{3.2}$$

$$\delta_T(X + A, Y + B) \leq \max\{\delta_T(X, Y), \delta_T(A, B)\}, \tag{3.3}$$

and

$$\delta_T(X + A, Y + A) \leq \left(\frac{\alpha}{\alpha + \lambda_{\min}(A)}\right) \delta_T(X, Y), \tag{3.4}$$

where $\alpha = \max\{||X||_2, ||Y||_2\}$.

Proposition 3.2 establishes another property of the Thompson metric, which is fundamental to demonstrate the global convergence of our Algorithm 1.

Proposition 3.2 Consider the Thompson metric defined in (3.1). Let $A, B, X, Y \in \mathbb{R}^{n\times n}$ be symmetric positive definite matrices, then

$$\delta_T(XA^{-1}, YA^{-1}) = \delta_T(X, Y). \tag{3.5}$$

Proof. To show this property, firstly observe that

$$\lambda_{\max}(AB^{-1}) = \lambda_{\max}(B^{-1}A) = \lambda_{\max}(B^{-\frac{1}{2}}AB^{-\frac{1}{2}}). \tag{3.6}$$

In fact, let (λ, x) be an eigenpair of AB^{-1}, and (γ, w) an eigenpair of $B^{-1}A$. Then, for $v = B^{-\frac{1}{2}}x$, we have

$$AB^{-1}x = \lambda x \Leftrightarrow AB^{-\frac{1}{2}}v = \lambda B^{\frac{1}{2}}v \Leftrightarrow B^{-\frac{1}{2}}AB^{-\frac{1}{2}}v = \lambda v,$$

which proves that λ is an eigenvalue of $B^{-\frac{1}{2}}AB^{-\frac{1}{2}}$.

Similarly, for $y = B^{\frac{1}{2}}w$, we have

$$B^{-1}Aw = \gamma w \Leftrightarrow B^{-\frac{1}{2}}AB^{-\frac{1}{2}}y = \gamma y,$$

obtaining that γ is an eigenvalue for $B^{-\frac{1}{2}}AB^{-\frac{1}{2}}$.

Taken maximum on the Rayleigh quotient we obtain our claim.

On the other hand, to prove (3.5) observe that

$$\lambda_{\max}((XA^{-1})^{-1}YA^{-1}) = \lambda_{\max}(AX^{-1}YA^{-1}) = \lambda_{\max}(A^{-1}AX^{-1}Y) = \lambda_{\max}(X^{-1}Y).$$

In the second equality above, we used the relation (3.6). Analogously, we can prove

$$\lambda_{\max}((YA^{-1})^{-1}XA^{-1}) = \lambda_{\max}(Y^{-1}X).$$

Then,

$$\begin{aligned}
\delta_T(XA^{-1}, YA^{-1}) &= \max\{\log\lambda_{\max}((XA^{-1})^{-1}YA^{-1}), \log\lambda_{\max}((YA^{-1})^{-1}XA^{-1})\} \\
&= \max\{\log\lambda_{\max}(X^{-1}Y), \log\lambda_{\max}(Y^{-1}X)\} \\
&= \delta_T(X, Y),
\end{aligned}$$

which completes the proof. □

Now consider the positive semi–definite matrix interval $\mathcal{I} = [2A(A+I)^{-1}, \frac{1}{2}(A+I)]$ and the mapping $\mathcal{G} \equiv X \to (\mu X + A)(X + \mu I)^{-1}$. We claim that $\mathcal{G}$ maps the interval $\mathcal{I}$ to itself. That is, if $X \in \mathcal{I}$ we have

$$0 \prec 2A(A+\sigma_1 I)(A+I)^{-1}(A+\sigma_1 I)^{-1} \preceq \mathcal{G}(X) \preceq \frac{1}{2}(\sigma_2 A+I)(A+I)(\sigma_2 A+I)^{-1}, \quad (3.7)$$

where $\sigma_1 = 1 + 2\mu$ and $\sigma_2 = 2 + \mu$. In fact, if $X \in \mathcal{I}$ then we have on one hand

$$[(2+\mu)I + A](I + A^{-1})^{-1} \preceq \mu X + A \preceq \frac{1}{2}[\mu I + (2+\mu)A], \quad (3.8)$$

and on the other hand, $X \in \mathcal{I}$ also implies

$$((2+\mu)I + \mu A^{-1})(I + A^{-1})^{-1} \preceq X + \mu I \preceq \frac{1}{2}[(1+2\mu)I + A], \quad (3.9)$$

so, (3.9) and the positive definition of the involved matrices lead to

$$2((1+2\mu)I + A)^{-1} \preceq (X + \mu I)^{-1} \preceq (A + I)[(2+\mu)A + \mu I]^{-1}. \quad (3.10)$$

Now, multiplying the respective matrices in (3.8) and (3.10), we obtain our claim. Now, we are ready to show the global convergence result for our Algorithm 1.

Theorem 3.3 Let $\{X_k\}_{k\geq 0}$ be any infinite sequence generated by Algorithm 1 with $X_0 \in \mathcal{I}$ be a symmetric and positive semi–definite matrix, $\mu > 0$ and $X^* = A^{1/2}$ be the exact solution of (1.1). Then, there exists a positive constant $\gamma \in (0, 1)$ such that

$$\delta_T(X_k, X^*) \leq \gamma^k \delta_T(X_0, X^*).$$

Moreover,

$$\lim_{k\to\infty} X_k = X^*.$$

Proof. Consider the nonlinear map $\mathcal{G} : \mathcal{I} \to \mathcal{I}$ previously defined and take arbitrary pair $X, Y \in \mathcal{I}$. Then using property (3.3), we obtain

$$\begin{aligned}
&\delta_T(\mathcal{G}(X), \mathcal{G}(Y)) \\
&= \delta_T[(\mu X + A)(X + \mu I)^{-1}, (\mu Y + A)(Y + \mu I)^{-1}] \\
&= \delta_T[\mu X(X + \mu I)^{-1} + A(X + \mu I)^{-1}, \mu Y(Y + \mu I)^{-1} + A(Y + \mu I)^{-1}] \\
&\leq \max\{\delta_T(\mu X(X + \mu I)^{-1}, \mu Y(Y + \mu I)^{-1}), \delta_T(A(X + \mu I)^{-1}, A(Y + \mu I)^{-1})\}.
\end{aligned}$$

Now let us establish bounds on each argument of the maximum: for the first one, we use properties (3.2) (twice) and (3.4):

$$\begin{aligned}
\delta_T(\mu X(X + \mu I)^{-1}, \mu Y(Y + \mu I)^{-1}) &= \delta_T\left(\frac{1}{\mu}(X + \mu I)X^{-1}, \frac{1}{\mu}(Y + \mu I)Y^{-1}\right) \\
&= \delta_T\left(\frac{1}{\mu}I + X^{-1}, \frac{1}{\mu}I + Y^{-1}\right) \\
&\leq \left(\frac{\alpha_1}{\alpha_1 + \mu^{-1}}\right)\delta_T(X^{-1}, Y^{-1}) \\
&= \left(\frac{\alpha_1}{\alpha_1 + \mu^{-1}}\right)\delta_T(X, Y),
\end{aligned}$$

where $\alpha_1 = \max\{||X^{-1}||_2, ||Y^{-1}||_2\}$.

Similarly, by using properties (3.2), (3.4) and (3.5), the second argument becomes

$$\begin{aligned}\delta_T(A(X+\mu I)^{-1}, A(Y+\mu I)^{-1}) &= \delta_T((X+\mu I)A^{-1}, (Y+\mu I)A^{-1}) \\ &= \delta_T(XA^{-1}+\mu A^{-1}, YA^{-1}+\mu A^{-1}) \\ &\leq \left(\frac{\bar{\alpha}_2}{\bar{\alpha}_2+\mu\lambda_{\min}(A^{-1})}\right)\delta_T(XA^{-1}, YA^{-1}) \\ &= \left(\frac{\bar{\alpha}_2}{\bar{\alpha}_2+\frac{\mu}{\lambda_{\max}(A)}}\right)\delta_T(X,Y),\end{aligned}$$

where $\bar{\alpha}_2 = \max\{||XA^{-1}||_2, ||YA^{-1}||_2\}$. Let us denote $\alpha_2 = \max\{||X||_2, ||Y||_2\}$. Then $\bar{\alpha}_2 \leq \frac{\alpha_2}{\lambda_{\min}(A)}$. Since the function $h(\alpha) = \alpha/(\alpha+c)$ is increasing, we obtain

$$\delta_T(A(X+\mu I)^{-1}, A(Y+\mu I)^{-1}) \leq \left(\frac{\alpha_2}{\alpha_2+\frac{\mu}{\kappa(A)}}\right)\delta_T(X,Y),$$

where $\kappa(A)$ denotes the condition number of A, i.e. $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$. Merging these two expressions in the above maximum, we arrive at

$$\delta_T(\mathcal{G}(X), \mathcal{G}(Y)) \leq \gamma\delta_T(X,Y),$$

where

$$\gamma = \max\{\frac{\alpha_1}{\alpha_1+\mu^{-1}}, \frac{\alpha_2}{\alpha_2+\mu\kappa(A)}\} < 1.$$

Since the positive definite interval $\mathcal{I}$ is a compact set, we can choose γ independently from X and Y. Specifically, since

$$\alpha_1 \leq ||\frac{1}{2}(I+A^{-1})||_2 \text{ and } \alpha_2 \leq ||\frac{1}{2}(I+A)||_2$$

then

$$\gamma = \max\left\{\frac{1+||A^{-1}||_2}{1+||A^{-1}||_2+2\mu^{-1}}, \frac{1+||A||_2}{1+||A||_2+\mu\kappa(A)^{-1}}\right\} < 1,$$

which is strictly less than one for definite positive matrix A. Thus, the map $\mathcal{G}$ is a strict contraction. Hence, from Banach contraction theorem it follows that $\delta_T(X_k, X^*)$ converges at linear rate given by γ, and $X_k \to X^*$. □

Remark 3.4 If we choose the parameter as $\mu = \sqrt{\frac{(1+||A||_2)\kappa(A)}{(1+||A^{-1}||_2)}}$ to balance the arguments of the maximum defining γ then we construct a theoretically optimal convergence rate.

4. Numerical experiments

In this section, we report some numerical results associated with the two variants of our Algorithm 1 FPM1 and FPM2 (the iterative schemes (2.2)–(2.4) respectively) and compare with some existing methods on the literature such as the Sra's iteration (1.4), the Newton method given by (1.2), and the gradient method (GradM) proposed in [17] with Barzilai–Borwein [10] step–size corrected with Armijo line search, in order to demonstrate the effectiveness of our proposal on three different experiments. All

test problems were performed on a intel(R) CORE(TM) i7–4770, CPU 3.40 GHz with 500GB HD and 16GB RAM, and all methods were implemented in Matlab.

In all experiments, presented in this section, in addition to checking the residual norm $E_k = ||A - X_k^2||_F / ||A||_F$, we also compute the relative change of two consecutive iterates

$$reschg_k = \frac{||X_{k+1} - X_k||_F}{||X_k||_F}. \tag{4.1}$$

We let all algorithms run up to N iterations and stop them at iteration $k < N$ if $E_k < \epsilon$, or $reschg_k < \epsilon_X$. We use the default values $N = 1000$, $\epsilon =$ 1e-5 and $\epsilon_X =$ 1e-6. Furthermore, for our procedures FPM1 and FPM2, we set $\mu = \nu\sqrt{\frac{(1+||A||_2)\kappa(A)}{(1+||A^{-1}||_2)}}$ with $\nu \in (0, 1)$. In addition, for all experiments and for all methods, we use the starting point $X_0 = (1/2)(A + I)$.

4.1. **Randomly generated symmetric positive definite problems.** In this subsection, we test the performance of all the methods on problems of the form (1.1) with $A \in \mathbb{R}^{n\times n}$ generated as follow, $A = QDQ^\top$ where

$$Q = (I - 2w_1w_1^\top)(I - 2w_2w_2^\top)(I - 2w_3w_3^\top),$$

with w_1, w_2 and w_3 are three n-dimensional vector randomly generated in the unitary sphere, and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix $D = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$, whose i–th eigenvalue is defined by

$$\log(\lambda_i) = \left(\frac{i-n}{n-1}\right) ncond.$$

The parameter *ncond*, in the above equality, controls the condition number of A. Note that in such kind of problems the logarithms of the eigenvalues (and not the eigenvalues) are uniformly distributed, leading to problems which are typically harder to solve. In addition, observe that the optimal solution is $A^{1/2} = QD^{1/2}Q^\top$, due to Q is an orthogonal matrix.

To illustrate the behaviour of the five methods, we show, in Figure 1, the residual norm E_k along the iterations for a randomly generated problem with $n = 100$ and $ncond = 6$. In this Figure, we observe that the faster procedure is the Newton's method, which is expected due to its quadratic convergence behaviour. We also see that our FPM1 reduces the residual norm E_k more quickly than the Sra's iteration. In addition, we note that the gradient method produces a very slow decrease in the residual norm but the estimated solution is far from the $A^{1/2}$.

Table 1 contains the numerical results associated to this experiment for three different values of $n = 100, 500, 1000$ and varying $ncond = 1, 3, 5, 10$. For each pair $(n, ncond)$, we generate ten independent problems and then, we report the average number of iterations (Nitr), the mean CPU time in seconds and the average of the residual norm, that is, $Res = (1/10)\sum_{i=1}^{10} E(\hat{X}_i)$ where $E(\hat{X}_i) = ||A - \hat{X}_i^2||_F$ and $\hat{X}_i$ denotes the estimated solution obtained by the algorithm, solving the i–th problem.

In order to compare the efficiency of the algorithms, we adopt the performance profile [4] introduced by Dolan and More to illustrate the whole performance of all the methods for the 120 problems tested in this subsection.

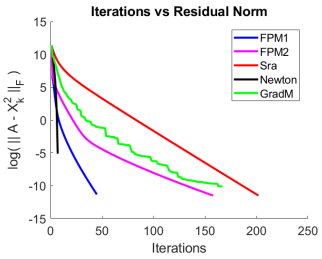


FIGURE 1. Behavior of the algorithms for $n = 100$, $ncond = 6$.

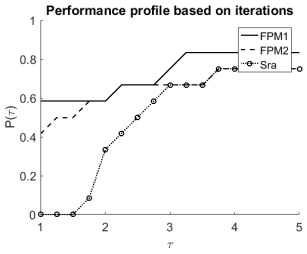
TABLE 1. Numerical results for symmetric positive definite problems.

	FPM1	**FPM2**	**Sra**	**Newton**	**GradM**	**FPM1**	**FPM2**	**Sra**	**Newton**	**GradM**
	n = 100, ncond = 1					n = 100, ncond = 3				
Nitr	7	10	26	5	28	23	25	45	5	72
Time	0.075	0.1264	0.53	0.0674	0.014	0.012	0.013	0.04	0.003	0.037
Res	9.05e-9	3.08e-9	7.67e-9	5.63e-16	6.45e-6	6.99e-6	7.40e-6	7.94e-6	8.28e-8	8.28e-6
	n = 100, ncond = 5					n = 100, ncond = 10				
Nitr	32	115	120	6	288	292	>2000	1716	7	>2000
Time	0.02	0.05	0.1	0.01	0.16	0.15	0.95	1.48	0.004	1.66
Res	8.70e-6	9.45e-6	9.03e-6	2.82e-10	9.83e-6	9.73e-6	0.0349	9.94e-6	1.1853e-15	0.1042
	n = 500, ncond = 1					n = 500, ncond = 3				
Nitr	53	8	21	5	31	51	25	47	5	52
Time	0.89	0.16	0.61	0.07	0.61	0.85	0.51	1.36	0.08	1.07
Res	8.15e-06	7.84e-6	7.66e-6	1.49e-12	5.35e-6	9.09e-6	8.42e-6	7.90e-6	1.62e-7	8.35e-6
	n = 500, ncond = 5					n = 500, ncond = 10				
Nitr	56	114	124	6	324	255	>2000	1747	Fail	>2000
Time	0.93	2.29	3.52	0.09	7.23	2.8098	40.59	51.31	Fail	69.28
Res	8.61e-6	9.82e-6	9.18e-6	4.9653e-16	9.79e-6	9.87e-9	2.69e+4	9.95e-6	Fail	22.2072
	n = 1000, ncond = 1					n = 1000, ncond = 3				
Nitr	76	9	22	5	31	74	26	48	6	78
Time	6.82	0.98	3.51	0.36	3.48	6.84	2.98	7.91	0.45	9.67
Res	9.84e-6	2.82e-6	5.68e-6	2.09e-12	7.56e-6	8.71e-6	6.80e-6	7.69e-6	2.26e-7	9.70e-6
	n = 1000, ncond = 5					n = 1000, ncond = 10				
Nitr	76	116	126	8	226	331	>2000	1768	Fail	>2000
Time	6.95	13.2	20.73	0.61	30.91	29.94	225.15	292.18	Fail	341.59
Res	9.91e-6	9.56e-6	9.31e-6	9.16e-11	9.47e-6	9.83e-6	4.51e+4	9.96e-6	Fail	70.8521

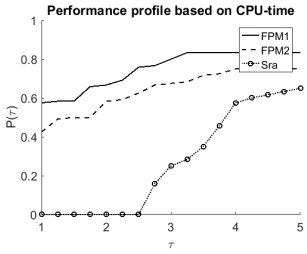
4.2. **Random correlation and low–rank matrices.** In this subsection, we test all the methods on random generated positive definite matrices built with the following matlab command:

- Example 1.1: random correlation matrices $A = \texttt{gallery}('\texttt{randcorr}', n))$.
- Example 1.2: $A = \texttt{eye}(n) + \beta UU^\top$, where U is generated by $U = \texttt{randn}(n, k)$ with $k = 10$, and a variable $\beta = \texttt{rand}$. Note that the product $UU^\top$ is a low–rank matrix.
- Example 1.3: the Hilbert matrix $A = \texttt{hilb}(n)$.

For examples 1.1 and 1.2, we vary n in $\{100, 250, 500, 1000\}$, and compare the mean number of iteration, the mean CPU time (in seconds) and the average error $E(\hat{X}_i) =$



(a) Performance profile based on the number of iterations



(b) Performance profile based on CPU–time

FIGURE 2. Performance profile based on the number of iterations and CPU–time, respectively.

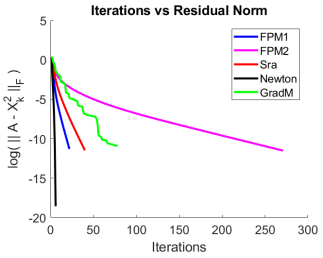
TABLE 2. Numerical results for Examples 1.1 and 1.2.

	FPM1	**FPM2**	**Sra**	**Newton**	**GradM**	**FPM1**	**FPM2**	**Sra**	**Newton**	**GradM**
	Example 1.1, $n = 100$					Example 1.1, $n = 250$				
Nitr	26	212	49	5	100	30	210	56	5	123
Time	0.01	0.11	0.05	0.002	0.04	0.09	0.73	0.32	0.02	0.34
Residual	8.01e-6	9.45e-6	9.13e-6	1.99e-6	7.69e-6	8.59e-6	9.57e-6	9.20e-6	2.02e-6	7.41e-6
	Example 1.1, $n = 500$					Example 1.1, $n = 1000$				
Nitr	50	467	82	6	203	56	563	91	6	232
Time	0.84	9.37	2.30	0.08	3.18	5.15	61.54	14.74	0.47	22.08
Residual	9.24e-6	9.86e-6	9.46e-6	2.17e-6	8.21e-6	9.39e-6	9.92e-6	9.64e-6	1.54e-6	7.47e-6
	Example 1.2, $n = 100$					Example 1.2, $n = 250$				
Nitr	14	26	89	6	19	12	58	135	7	18
Time	0.01	0.01	0.08	0.003	0.01	0.04	0.19	0.76	0.02	0.05
Residual	5.30e-6	7.31e-6	9.25e-6	3.07e-7	4.64e-6	5.47e-6	8.58e-6	9.33e-6	7.72e-7	4.66e-6
	Example 1.2, $n = 500$					Example 1.2, $n = 1000$				
Nitr	13	111	178	7	18	12	698	230	7	17
Time	0.21	2.26	5.00	0.10	0.29	1.06	36.59	37.02	0.57	1.65
Residual	4.95e-6	9.34e-6	9.36e-6	5.48e-7	5.35e-6	4.50e-6	6.21e-6	9.54e-6	9.28e-7	3.51e-6

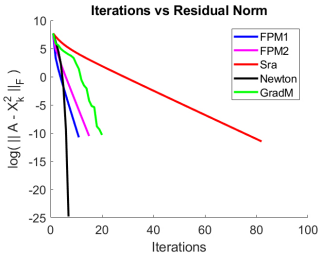
$||A - \hat{X}_i^2||_F$ obtained by the algorithms on a total of 30 independent instances, for each value of n. Note that the random correlation matrices are well–conditioned, the matrices given by Example 1.2 are moderately well–conditioned, while Hilbert's matrix is ill–conditioned. These test experiments were taken from [19].

Table 2 reports the numerical results associated to the test examples 1.1–1.2. From this table, we can see that the Newton's method obtains the best results both in CPU–time and in the number of iterations performed. Furthermore, we note that our FPM1 outperforms the other first–order approaches both in terms of iterations and CPU time. In fact, we observe that our FPM1 performs almost the same number of iterations as Newton's method for low–rank type problems. We also note that our second proposal converges very slowly for random correlation matrices, while for test Examples 1.2, this procedure is faster than the Sra's iteration.

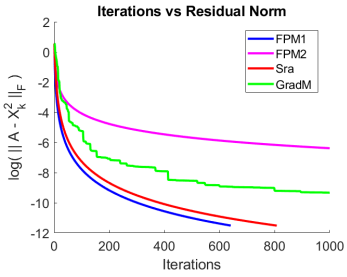
The residual norm of the iterates for the three different examples 1.1, 1.2 and 1.3 are shown in Figure 3. In the subfigure (c) we omit the curve associated with Newton's



(a) Residual for random correlation matrices with $n = 100$



(b) Residual for low–rank matrices with $n = 100$



(c) Residual for the Hilbert matrix with $n = 50$

FIGURE 3. Residual vs the iterations number for all methods.

method because this procedure fails in ill–conditioned problems. Figure 3 shows that our first proposal FPM1 converges faster than the rest of the first–order methods, while the Newton's method is superior to the rest of methods when the matrix A is well–conditioned or moderately well–conditioned. In addition, we notice that in the ill–conditioned situation, our FPM1 procedure has a very similar behavior to the Sra's iteration, however our proposal makes a smaller number of matrix inversions and therefore is the most efficient method in this case.

5. Conclusion and perspectives

The goal of this paper is to develop an effective algorithm which is able to compute the square root of a given symmetric positive semi–definite matrix. Our strategy is simply to rearrange the nonlinear equation $A - X^2 = 0$, in order to design a contractive mapping which is regulated for an exogenous (conveniently chosen) parameter, leading to effective scaled fixed point methods, which only require calculating one matrix inverse (numerically, solving a linear system of equations) and one matrix product per iteration. We further theoretically show the global convergence of one of our proposed numerical methods. In fact, we demonstrate that this first proposal converges q–linearly to $A^{1/2}$. The second one is only numerically studied. However,

our numerical experiments on randomly generated symmetric positive definite matrices, with different conditioning situations, show that our two procedures are effective to solve the matrix equation $A - X^2 = 0$. In addition, our numerical tests show that our first proposal (FPM1) outperforms some first–order methods existing in the literature.

Although in the literature there are second–order methods such as the scaled Newton method, which is one of the most effective algorithms at present, both the Sra and our theoretical analysis add understanding and a conceptual value about the convergence properties of first–order methods under the Thompson metric and using their properties. These ideas might be valuable to analyze other types of methods in the context of geometric optimization field.

Acknowledgements. The second author wants to thank the Federal University of Santa Catarina–Brazil and remarks that his contribution to the present article was predominantly carried out at this institution.

References

[1] T. Ando, *Fixed points of certain maps on positive semidefinite operators*, Functional Analysis and Approximation, Springer, (1981), 29-38. DOI: https://doi.org/10.1007/978-3-0348-9369-5_6

[2] S. Bernstein Dennis, *Matrix Mathematics: Theory, Facts, and Formulas*, Princeton University Press, 2009.

[3] E. Deadman, N.J. Higham, R. Ralha, *Blocked Schur algorithms for computing the matrix square root*, International Workshop on Applied Parallel Computing, **1**(2012), 171-182. DOI: https://doi.org/10.1007/978-3-642-36803-5_12

[4] E.D. Dolan, J.J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, **91**(2002), 201-213. DOI: https://doi.org/10.1007/s101070100263

[5] E.S. Gawlik, *Zolotarev iterations for the matrix square root*, SIAM Journal on Matrix Analysis and Applications, **40**(2019), 696-719.

[6] N.J. Higham, *Newton's method for the matrix square root*, Mathematics of Computation, **46**(1986), 537-549. DOI: https://doi.org/10.1090/S0025-5718-1986-0829624-5

[7] N.J. Higham, *Stable iterations for the matrix square root*, Numerical Algorithms. **15**(1997), 227-242. DOI: https://doi.org/10.1023/A:1019150005407

[8] N.J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematic, 2008.

[9] B. Iannazzo, M. Porcelli, *The Riemannian Barzilai-Borwein method with nonmonotone line search and the matrix geometric mean computation*, IMA Journal of Numerical Analysis, **38**(2017), no. 1, 495-517. DOI: https://doi.org/10.1093/imanum/drx015

[10] W. La Cruz, M. Raydan, *Nonmonotone spectral methods for large-scale nonlinear systems*, Optimization Methods and Software, **18**(2003), 583-599. DOI: https://doi.org/10.1080/10556780310001610493

[11] B. Laszkiewicz, K. Zietak, *A Padé family of iterations for the matrix sector function and the matrix pth root*, Numerical Linear Algebra with Applications, **16**(2009), 951-970. DOI: https://doi.org/10.1002/nla.656

[12] H. Lee, Y. Lim, *Invariant metrics, contractions and nonlinear matrix equations*, Nonlinearity, **21**(2008), no. 4, 857-878. DOI: https://doi.org/10.1088/0951-7715/21/4/011

[13] B. Lemmens, R. Nussbaum, *Nonlinear Perron-Frobenius Theory*, Cambridge University Press, **189**, 2012.

[14] Y. Lim, M. Pálfia, *Matrix power means and the Karcher mean*, Journal of Functional Analysis, **262**(2012), no. 4, 1498-1514. DOI: https://doi.org/10.1016/j.jfa.2011.11.012

[15] H. Oviedo, *Implicit steepest descent algorithm for optimization with orthogonality constraints*, Optimization Letters, (2021). DOI: https://doi.org/10.1007/s11590-021-01801-5

[16] V.Y. Pan, Z. Chen, A. Zheng and others, *The complexity of the algebraic eigenproblem*, Mathematical Sciences Research Institute Berkeley, (1998). https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.1565&rep=rep1&type=pdf

[17] J. Prateek, J. Chi, K. Sham, N. Praneeth, *Global convergence of non-convex gradient descent for computing matrix squareroot*, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, **54**(2017), 479-488.

[18] S. Sra, R. Hosseini, *Conic geometric optimization on the manifold of positive definite matrices*, SIAM Journal on Optimization, **25**(2015), no. 1, 713-739. DOI: https://doi.org/10.1137/140978168

[19] S. Sra, *On the matrix square root via geometric optimization*, The Electronic Journal of Linear Algebra, **3**(2016), 433-443. DOI: https://doi.org/10.48550/arXiv.1507.08366

[20] S. Sra, *Positive definite matrices and the S-divergence*, Proceedings of the American Mathematical Society, **144**(2016), 2787-2797. DOI: https://doi.org/10.1090/proc/12953

[21] R. Van Der Merwe, E.A. Wan, *The square-root unscented Kalman filter for state and parameter-estimation*, IEEE International Conference on Acoustics, Speech, and Signal Processing, Proceedings (Cat. No. 01CH37221), **6**(2001), 3461-3464. DOI: https://doi.org/10.1109/ICASSP.2001.940586

[22] X. Yuan, W. Huang, P.A. Absil, K.A. Gallivan, *A Riemannian limited-memory BFGS algorithm for computing the matrix geometric mean*, Procedia Computer Science, **80**(2016), 2147-2157. DOI: https://doi.org/10.1016/j.procs.2016.05.534

[23] X. Zhu, *A Riemannian conjugate gradient method for optimization on the Stiefel manifold*, Computational Optimization and Applications, **67**(2017), 73-110. DOI: https://doi.org/10.1007/s10589-016-9883-4

Received: October 7, 2020; Accepted: May 14, 2022.