Hindawi Publishing Corporation
International Journal of Mathematics and Mathematical Sciences
Volume 2014, Article ID 613840, 8 pages
http://dx.doi.org/10.1155/2014/613840

Research Article

Inversion Free Algorithms for Computing the Principal Square Root of a Matrix

Nicholas Assimakis[1] and Maria Adam[2]

[1] *Department of Electronic Engineering, Technological Educational Institute of Central Greece, 3rd km Old National Road Lamia-Athens, 35100 Lamia, Greece*
[2] *Department of Computer Science and Biomedical Informatics, University of Thessaly, 2-4 Papasiopoulou Street, 35100 Lamia, Greece*

Correspondence should be addressed to Maria Adam; madam@dib.uth.gr

Received 13 February 2014; Revised 13 May 2014; Accepted 13 May 2014; Published 18 June 2014

Academic Editor: Ram U. Verma

New algorithms are presented about the principal square root of an $n \times n$ matrix A. In particular, all the classical iterative algorithms require matrix inversion at every iteration. The proposed inversion free iterative algorithms are based on the Schulz iteration or the Bernoulli substitution as a special case of the continuous time Riccati equation. It is certified that the proposed algorithms are equivalent to the classical Newton method. An inversion free algebraic method, which is based on applying the Bernoulli substitution to a special case of the continuous time Riccati equation, is also proposed.

1. Introduction

Let A be a real or complex $n \times n$ matrix with no eigenvalues on $\mathbb{R}^-$ (the closed negative real axis). Then there exists a unique $n \times n$ matrix X such that $X^2 = A$ and the eigenvalues of X lie in the segment $\{z : -\pi/2 < \arg(z) < \pi/2\}$. We refer to X as the *principle square root* of A.

The computation of the principal square root of a matrix is a problem of great interest and practical importance with numerous applications in mathematical and engineering problems. Due to the importance of the problem, many iterative algorithms have been proposed and successfully employed for calculating the principal square root of a matrix [1–7] without seeking the eigenvalues and eigenvectors of the matrix; these algorithms require matrix inversion at every iteration. Blocked Schur Algorithms for Computing the Matrix Square Root are proposed in [8] where the matrix is reduced to upper triangular form and a recurrence relation enables the square root of the triangular matrix to be computed a column or superdiagonal at a time. In this paper new inversion free iterative algorithms for computing the principal square root of a matrix are proposed.

The paper is organized as follows: in Section 2 a survey of classical iterative algorithms is presented; all the algorithms use matrix inversion in every iteration. In Section 3, the inversion free iterative algorithms are developed; the algorithms are derived by the Schulz iteration for computing the inversion of a matrix or by eliminating matrix inversion from the convergence criteria of the algorithms. In Section 4, an algebraic method for computing the principal square root of a matrix is presented; the method is associated with the solution of a special case of the continuous time Riccati equation, which arises in linear estimation [9] and requires only one matrix inversion (this is not an iterative method). Simulation results are presented in Section 5. Section 6 summarizes the conclusions.

2. Inversion Use Iterative Algorithms

In this section, a survey of classical iterative algorithms, which use matrix inversion in every iteration is presented. Among the algorithms, which have been proposed for computing the principle square root of a matrix, the Newton methods have been studied for many years. Since the

convergence is proved but numerical instability has been observed [4], several Newton variants have been derived. Newton methods have also been used for computing the pth root a matrix [1, 2]. The conclusion that a nonsingular and diagonalizable matrix for any positive integer n has an nth root and is stated in [10].

Algorithm 1(a). This algorithm is the classical Newton method [6]

$$Y_{n+1} = \frac{1}{2}\left(Y_n + Y_n^{-1}A\right), \quad Y_0 = A,\ Y_n \longrightarrow A^{1/2}. \tag{1}$$

Note that we are also able to use the initial condition $Y_0 = I$.

Note that all the algorithms are applicable for $n = 0, 1, \ldots$ and that the convergence is achieved, when $\|Q_{n+1} - Q_n\| < \varepsilon$, where Q_n is the sequence that converges to the principle square root of A and ε is a small positive number and $\|M\|$ denotes the spectral norm of the matrix M (i.e., the largest singular value of M).

Algorithm 2(a). This algorithm is a variant of the classical Newton method [7]

$$Z_{n+1} = \frac{1}{2}\left(Z_n + AZ_n^{-1}\right), \quad Z_0 = A,\ Z_n \longrightarrow A^{1/2}. \tag{2}$$

Note that we are also able to use the initial condition $Z_0 = I$.

Algorithm 3(a). This algorithm is a variant of the classical Newton method, where the principle square root of A^{-1} is computed as well [6]

$$\begin{aligned} M_{n+1} &= \frac{1}{2}\left(M_n + N_n^{-1}\right), \quad M_0 = A,\ M_n \longrightarrow A^{1/2}, \\ N_{n+1} &= \frac{1}{2}\left(N_n + M_n^{-1}\right), \quad N_0 = I,\ N_n \longrightarrow A^{-1/2}. \end{aligned} \tag{3}$$

Algorithm 4(a). This algorithm is proposed in [7]

$$\begin{aligned} G_{n+1} &= 4G_n\left(I + G_n\right)^{-2}, \quad G_0 = A,\ G_n \longrightarrow I, \\ R_{n+1} &= \frac{1}{2}R_n\left(I + G_n\right), \quad R_0 = I,\ R_n \longrightarrow A^{1/2}. \end{aligned} \tag{4}$$

Algorithm 5(a). This algorithm is proposed in [5]

$$\begin{aligned} &X_{n+1} = \frac{1}{2}I + X_n^{-1} - \frac{1}{2}X_n^{-2}, \quad X_0 = \frac{1}{2}\left(I + A\right),\ X_n \longrightarrow I, \\ &W_{n+1} = W_n X_n, \quad W_0 = I,\ W_n \longrightarrow A^{1/2}. \end{aligned} \tag{5}$$

Algorithm 6(a). It will be shown that the problem of computing the principal square root of a matrix is equivalent to the problem of solving a related Riccati equation. This algorithm takes advantage of this relation and is derived via the solution of the related Riccati equation.

The problem of computing the principal square root of a matrix is associated with a well-known problem of estimation theory, namely the problem of solving a continuous time Riccati equation. The Riccati equation arises in linear estimation [3], namely in the implementation of the Kalman-Bucy filter and it is formulated as

$$\begin{aligned} \frac{dP(t)}{dt} &= F(t)P(t) + P(t)F^T(t) + Q(t) \\ &\quad - P(t)H^T(t)R^{-1}(t)H(t)P(t), \quad P(0) = P_0, \end{aligned} \tag{6}$$

where $P(t)$ is the filtering error-covariance matrix, $F(t)$ and $H(t)$ are the system dynamic and output matrices, respectively. $Q(t)$ and $R(t)$ are the plant and measurement noise covariance matrices, respectively.

In the special case of the time invariant model where $F(t) = F = 0$, $H(t) = H$, $Q(t) = Q = A$, $R(t) = R$ and $H^T(t)R^{-1}(t)H = I$, and using the initial condition $P_0 = 0$, the Riccati equation of interest takes the following form

$$\frac{dP(t)}{dt} = A - P(t)P(t), \quad P(0) = 0. \tag{7}$$

It is well known [11, 12] that there exists a steady state solution of the Riccati equation.

The following statement is now obviously true: "The problem of computing the principal square root of matrix A is equivalent to the problem of solving the Riccati equation (7), the steady state solution of which is equal to the principal square root of matrix A".

The idea is to apply the Bernoulli substitution [11] in the special case of the continuous time Riccati equation (7): an integration free solution of (7) can be obtained using the Bernoulli substitution [11]

$$P(t) = V(t)U^{-1}(t). \tag{8}$$

Substituting (8) in (7), we have

$$\begin{aligned} \frac{dU(t)}{dt} &= V(t), \quad U(0) = I, \\ \frac{dV(t)}{dt} &= AU(t), \quad V(0) = 0. \end{aligned} \tag{9}$$

Then, by discretising the above equations, we have:

$$U_{k+1} = U_k + V_k, \quad U_0 = I, \tag{10}$$

$$V_{k+1} = AU_k + V_k, \quad V_0 = 0, \tag{11}$$

$$P_{k+1} = V_{k+1} + U_{k+1}^{-1}, \quad P_0 = 0. \tag{12}$$

Thus, by applying the Bernoulli substitution [11] in a special case of the continuous time Riccati equation, the following algorithm is derived

$$\begin{aligned} &U_{n+1} = U_n + V_n, \quad U_0 = I, \\ &V_{n+1} = AU_n + V_n, \quad V_0 = 0, \\ &P_n = V_n U_n^{-1}, \\ &P_n \longrightarrow A^{1/2}. \end{aligned} \tag{13}$$

Note that the following variant uses iterations only for the first quantity involved

$$\begin{aligned} U_{n+1} &= 2U_n + (A-I)\,U_{n-1}, \quad U_1 = I,\ U_0 = I, \\ V_{n+1} &= U_{n+1} + (A-I)\,U_n, \\ P_n &= V_n U_n^{-1}, \\ P_n &\longrightarrow A^{1/2}. \end{aligned} \tag{14}$$

Algorithm 7(a). This algorithm is derived via the solution of the related Riccati equation using the doubling principle [12].

It is obvious that (10) and (11) can be easily written in a state space form as follows

$$\begin{bmatrix} U_{k+1} \\ V_{k+1} \end{bmatrix} = \Phi \begin{bmatrix} U_k \\ V_k \end{bmatrix}, \quad \Phi = \begin{bmatrix} I & I \\ A & I \end{bmatrix}. \tag{15}$$

Let us denote:

$$S(0)^{2^k} = S(k) = \begin{bmatrix} a_k & b_k \\ Ab_k & a_k \end{bmatrix}, \quad S(0) = \begin{bmatrix} I & I \\ A & I \end{bmatrix}. \tag{16}$$

Then, using the doubling principle [12], that is, calculating the power

$$S(k+1) = S^2(k) = S(k)\,S(k), \tag{17}$$

it is easy to prove that the following equations hold

$$\begin{aligned} a_{k+1} &= a_k a_k + b_k A b_k, \\ b_{k+1} &= a_k b_k + b_k a_k, \\ Ab_{k+1} &= Ab_k a_k + a_k A b_k, \\ a_{k+1} &= Ab_k b_k + a_k a_k. \end{aligned} \tag{18}$$

Due to the form of matrix $S(0)$ given in (16), it is obvious that the matrices a_k and b_k are polynomials in A of the form

$$\begin{aligned} a_k &= \sum_{i=0}^{\ell(k)} x_i A^i, \\ b_k &= \sum_{j=0}^{m(k)} y_j A^j, \end{aligned} \tag{19}$$

where x_i and y_j are scalar coefficients and $\ell(k)$ and $m(k)$ are functions of the number of iterations, which need not be specified. Then, directly from the above equation we derive

$$a_k A = A a_k, \tag{20}$$

$$b_k A = A b_k, \tag{21}$$

$$a_k b_k = b_k a_k, \tag{22}$$

since

$$a_k b_k = \sum_{i=0}^{\ell(k)} x_i A^i \sum_{j=0}^{m(k)} y_j A^j = \sum_{j=0}^{m(k)} y_j A^j \sum_{i=0}^{\ell(k)} x_i A^i = b_k a_k. \tag{23}$$

Furthermore, it is easily seen from (15) and (16) that the following state space equation holds

$$\begin{bmatrix} U_{2^k} \\ V_{2^k} \end{bmatrix} = S(0)^{2^k} \begin{bmatrix} U_0 \\ V_0 \end{bmatrix} = \begin{bmatrix} a_k & b_k \\ Ab_k & a_k \end{bmatrix} \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{24}$$

Then, it is obvious that:

$$\begin{aligned} U_{2^k} &= a_k, \\ V_{2^k} &= Ab_k, \\ P_{2^k} &= P_k = V_{2^k} U_{2^k}^{-1}. \end{aligned} \tag{25}$$

Then, using (18), (20)–(22) and (24)-(25), we have

$$a_{k+1} = a_k^2 + Ab_k^2, \quad a_0 = I, \tag{26}$$

$$b_{k+1} = 2a_k b_k, \quad b_0 = I, \tag{27}$$

$$P_{k+1} = Ab_{k+1} a_{k+1}^{-1}, \quad P_0 = 0. \tag{28}$$

Thus, by using the doubling principle [12], the following algorithm is derived

$$\begin{aligned} a_{n+1} &= a_n^2 + Ab_n^2, \quad a_0 = I, \\ b_{n+1} &= 2a_n b_n, \quad b_0 = I, \\ P_n &= Ab_n a_n^{-1}, \\ P_n &\longrightarrow A^{1/2}. \end{aligned} \tag{29}$$

Algorithm 8(a). In the following, using (22) and (26)–(28), we derive

$$P_{k+1} = 2\left(P_k^{-1} + P_k A^{-1}\right)^{-1}, \tag{30}$$

with initial condition

$$P_1 = 2A(I+A)^{-1} = 2\left(A^{-1} + I\right)^{-1}. \tag{31}$$

Note that, where we are obliged to use the initial condition P_1 by (31), which is derived from (26)–(28) instead of using the initial condition P_0 in (28), due to the fact that (30) requires inversion of matrix P_k and $P_0 = 0$ is a singular matrix.

At this point, we are able to use (30) with initial condition $P_0 = A$, resulting to the same sequence of P_k as in (26)–(28).

Then, it is obvious that the algorithm in (26)–(28) can be written in the following equivalent form:

$$P_{n+1} = 2\left(P_n^{-1} + P_n A^{-1}\right)^{-1}, \quad P_0 = A,\ P_n \longrightarrow A^{1/2}. \tag{32}$$

Remark 1. (1) All the above algorithms are equivalent to each other. In fact, the relations between the quantities involved in the above algorithms hold:

$$Y_n = Z_n = M_n = AN_n = G_n = W_n = a_n b_n^{-1} = P_n^{-1} A. \tag{33}$$

The proof is derived by induction and is trivial.

(2) The relationship between the Riccati equation and the principal square root algorithms is certified: it is shown that the algorithm derived by solving a special case of the continuous time Riccati equation is equivalent to the classical Newton method.

3. Inversion Free Iterative Algorithms

In this section, the inversion free iterative algorithms are developed by using the Schulz iteration for computing the inversion of a matrix or by eliminating matrix inversion from the convergence criteria of the algorithms.

Algorithm 1(b). This algorithm is an inversion free variant of the Newton method in Algorithm 1(a); the basic idea is to replace the computation of Y_n^{-1} in Algorithm 1 by the Schulz iteration for Y_n^{-1} as described in [13]

$$\begin{aligned} Y_{n+1} &= \frac{1}{2}\left(Y_n + X_n A\right), \quad Y_0 = A,\ Y_n \longrightarrow A^{1/2}, \\ X_{k+1} &= X_k\left(2I - Y_n X_k\right), \quad X_0 = \frac{I}{\|Y_n\|_\infty},\ X_k \longrightarrow X_n, \end{aligned} \tag{34}$$

where $\|Y_n\|_\infty$ denotes the infinity norm of the matrix Y_n.

Algorithm 2(b). This algorithm is an inversion free variant of Algorithm 2(a), derived using the Schulz iteration idea

$$\begin{aligned} Z_{n+1} &= \frac{1}{2}\left(Z_n + A X_n\right), \quad Z_0 = A,\ Z_n \longrightarrow A^{1/2}, \\ X_{k+1} &= X_k\left(2I - Z_n X_k\right), \quad X_0 = \frac{I}{\|Z_n\|_\infty},\ X_k \longrightarrow X_n. \end{aligned} \tag{35}$$

Algorithm 3(b). This algorithm is an inversion free variant of Algorithm 3(a), derived using the Schulz iteration idea

$$\begin{aligned} M_{n+1} &= \frac{1}{2}\left(M_n + X_n\right), \quad M_0 = A,\ M_n \longrightarrow A^{1/2}, \\ X_{k+1} &= X_k\left(2I - N_n X_k\right), \quad X_0 = \frac{I}{\|N_n\|_\infty},\ X_k \longrightarrow X_n, \\ N_{n+1} &= \frac{1}{2}\left(N_n + Y_n\right), \quad N_0 = I,\ N_n \longrightarrow A^{-1/2}, \\ Y_{k+1} &= X_k\left(2I - M_n Y_k\right), \quad Y_0 = \frac{I}{\|M_n\|_\infty},\ Y_k \longrightarrow Y_n. \end{aligned} \tag{36}$$

Algorithm 4(b). This algorithm is an inversion free variant of Algorithm 4(a), derived using the Schulz iteration idea

$$\begin{aligned} G_{n+1} &= 4G_n X_n^2, \quad G_0 = A,\ G_n \longrightarrow I, \\ X_n &= \left(I + G_n\right), \\ Y_{k+1} &= Y_k\left(2I - X_n Y_k\right), \quad Y_0 = \frac{I}{\|X_n\|_\infty},\ Y_k \longrightarrow X_n, \\ R_{n+1} &= \frac{1}{2} R_n\left(I + G_n\right), \quad R_0 = I,\ R_n \longrightarrow A^{1/2}. \end{aligned} \tag{37}$$

Algorithm 5(b). This algorithm is an inversion free variant of Algorithm 5(a), derived using the Schulz iteration idea

$$\begin{aligned} X_{n+1} &= \frac{1}{2} I + Y_n - \frac{1}{2} Y_n^2, \quad X_0 = \frac{1}{2}\left(I + A\right),\ X_n \longrightarrow I, \\ Y_{k+1} &= Y_k\left(2I - X_n Y_k\right), \quad Y_0 = \frac{I}{\|X_n\|_\infty},\ Y_k \longrightarrow Y_n, \\ W_{n+1} &= W_n X_n, \quad W_0 = I,\ W_n \longrightarrow A^{1/2}. \end{aligned} \tag{38}$$

Algorithm 6(b). This algorithm is an inversion free variant of Algorithm 6(a), derived using another approach to eliminate the matrix inversion requirement

$$\begin{aligned} U_{n+1} &= U_n + V_n, \quad U_0 = I, \\ V_{n+1} &= A U_n + V_n, \quad V_0 = 0, \\ \delta_n &= V_{n+1} U_n - V_n U_{n+1} \longrightarrow 0, \\ V_n U_n^{-1} &\longrightarrow A^{1/2}. \end{aligned} \tag{39}$$

The convergence criterion depends on the quantity

$$\delta_n = V_{n+1} U_n - V_n U_{n+1}. \tag{40}$$

It is easy to prove that

$$\delta_{n+1} = (I - A)\,\delta_n, \quad \delta_0 = A. \tag{41}$$

Hence, if $\|I - A\| < 1$, then $\|\delta_{n+1}\| < \|\delta_n\| < \|A\|$; thus there exists $\ell : \|\delta_\ell\| < \varepsilon$ that can be calculated off-line.

Finally, we observe that

$$\delta_n = (I - A)^n A. \tag{42}$$

Hence, if all the eigenvalues of A lie inside the unit circle, then the quantity δ_n tends to zero as n tends to infinite. Furthermore, if the eigenvalues of A lie outside the unit circle, then we are able use the algorithm to calculate the principal square root of A^{-1}, from where we are able to find the principal square root of A (we need only two matrix inversions).

Algorithm 7(b). This algorithm is an inversion free variant of Algorithm 7(a), derived using another approach to eliminate the matrix inversion requirement

$$\begin{aligned} a_{n+1} &= a_n^2 + A b_n^2, \quad a_0 = I, \\ b_{n+1} &= 2 a_n b_n, \quad b_0 = I, \\ b_{n+1} a_n - b_n a_{n+1} &\longrightarrow 0, \\ A b_n a_n^{-1} &\longrightarrow A^{1/2}. \end{aligned} \tag{43}$$

Note that the following variant uses another convergence criterion

$$\begin{aligned} a_{n+1} &= a_n^2 + Ab_n^2, \quad a_0 = I, \\ b_{n+1} &= 2a_n b_n, \quad b_0 = I, \\ e_{n+1} &= e_n^2, \quad e_0 = I - A, \\ e_n &= a_n^2 - Ab_n^2 \longrightarrow 0, \\ Ab_n a_n^{-1} &\longrightarrow A^{1/2}. \end{aligned} \tag{44}$$

The convergence criterion depends on the quantity

$$e_n = a_n^2 - Ab_n^2. \tag{45}$$

It is easy to prove that

$$e_{n+1} = e_n^2, \quad e_0 = I - A. \tag{46}$$

Hence, if $\|I - A\| < 1$, then $\|e_{n+1}\| < \|e_n\| < 1$; thus there exists $\ell : \|e_\ell\| < \varepsilon$ that can be calculated off-line.

Finally, we observe that

$$e_n = (I - A)^{2^n}. \tag{47}$$

Hence, if all the eigenvalues of A lie inside the unit circle, then the quantity e_n tends to zero as n tends to infinite. Furthermore, if the eigenvalues of A lie outside the unit circle, then we are able use the algorithm to calculate the principal square root of A^{-1}, from where we are able to find the principal square root of A (we need only two matrix inversions).

Algorithm 8(b). This algorithm is an inversion free variant of Algorithm 8(a), derived using the Schulz iteration idea:

$$\begin{aligned} &P_{n+1} = 2R_n, \quad P_0 = A, \; P_n \longrightarrow A^{1/2}, \\ &Z_n = \left(X_n + P_n A^{-1}\right), \\ &X_{k+1} = X_k\left(2I - Z_n X_k\right), \quad X_0 = \frac{I}{\|Z_n\|_\infty}, \; X_k \longrightarrow R_n, \\ &Y_{k+1} = X_k\left(2I - P_n Y_k\right), \quad Y_0 = \frac{I}{\|P_n\|_\infty}, \; Y_k \longrightarrow X_n. \end{aligned} \tag{48}$$

All the inversion use and inversion free iterative algorithms are summarized in Table 1.

Remark 2. (1) All the algorithms presented in Section 3 are inversion free algorithms; they do not require matrix inversion at every iteration. The elimination of the matrix inversion requirement can be achieved using the Schulz iteration.

(2) Algorithms 6(b) and 7(b) use another technique to eliminate the matrix inversion requirement. These two algorithms require only one matrix inversion, after the convergence of the algorithms in order to compute the final result. It is obvious that this matrix inversion can be substituted by the Schulz iteration.

(3) Algorithm 8(b) uses only one inversion (A^{-1}). It is obvious that this matrix inversion can be substituted by the Schulz iteration.

(4) Note that all the proposed inversion free iterative algorithms compute the principal square root of an $n \times n$ dimensional matrix with complexity of the order $O(n^3)$, due to the fact that the proposed inversion algorithms require matrix additions and matrix multiplications. The algorithms avoid possible problems in matrix inversion (and even at every iteration).

4. Algebraic Method

In this section an algebraic method for computing the principal square root of a matrix is presented. The method is associated with the solution of a special case of the continuous time Riccati equation which arises in linear estimation [9] and requires only one matrix inversion (this is not an iterative method).

We are able to solve the Riccati equation (7) using the algebraic method proposed in [14–17]. In fact the following matrix is formed:

$$\widetilde{\Phi} = \begin{bmatrix} 0 & I \\ A & 0 \end{bmatrix}. \tag{49}$$

Then, $\widetilde{\Phi}$ can be written as

$$\widetilde{\Phi} = WLW^{-1}, \tag{50}$$

where

$$L = \begin{bmatrix} \Lambda & 0 \\ 0 & -\Lambda \end{bmatrix} \tag{51}$$

is a block-diagonal matrix containing the eigenvalues of $\widetilde{\Phi}$, with Λ diagonal matrix with all the eigenvalues of $\widetilde{\Phi}$ lying the right half-plane eigenvalues of matrix $\widetilde{\Phi}$ and

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \tag{52}$$

is the matrix containing the corresponding eigenvectors of $\widetilde{\Phi}$.

Then, the solution of the Riccati equation is given in terms of the eigenvalues of matrix $\widetilde{\Phi}$ and formulated

$$X = W_{21} W_{11}^{-1}. \tag{53}$$

The method requires only one matrix inversion in order to compute the final result (this is not an iterative method). It is obvious that this matrix inversion can be substituted by the Schulz iteration.

We are also able to compute all the square roots (not only the principal square root) of a matrix, using the ideas in [14–16].

5. Simulation Results

Simulation results are given to illustrate the efficiency of the proposed methods. The proposed algorithms compute

Table 1: Principal square root iterative algorithms.

	Inversion use		Inversion free
1a	$Y_{n+1} = \frac{1}{2}\left(Y_n + Y_n^{-1}A\right), \quad Y_0 = A,\ Y_n \longrightarrow A^{1/2}$	1b	$Z_{n+1} = \frac{1}{2}\left(Z_n + AX_n\right), \quad Z_0 = A,\ Z_n \longrightarrow A^{1/2}$ $X_{k+1} = X_k\left(2I - Z_nX_k\right), \quad X_0 = \frac{I}{\|Z_n\|_\infty},\ X_k \longrightarrow X_n$
2a	$Z_{n+1} = \frac{1}{2}\left(Z_n + AZ_n^{-1}\right), \quad Z_0 = A,\ Z_n \longrightarrow A^{1/2}$	2b	$Z_{n+1} = \frac{1}{2}\left(Z_n + AX_n\right), \quad Z_0 = A,\ Z_n \longrightarrow A^{1/2}$ $X_{k+1} = X_k\left(2I - Z_nX_k\right), \quad X_0 = \frac{I}{\|Z_n\|_\infty},\ X_k \longrightarrow X_n$
3a	$M_{n+1} = \frac{1}{2}\left(M_n + N_n^{-1}\right), \quad M_0 = A,\ M_n \longrightarrow A^{1/2}$ $N_{n+1} = \frac{1}{2}\left(N_n + M_n^{-1}\right),\ N_0 = I,\ N_n \longrightarrow A^{-1/2}$	3b	$M_{n+1} = \frac{1}{2}\left(M_n + X_n\right), \quad M_0 = A,\ M_n \longrightarrow A^{1/2}$ $X_{k+1} = X_k\left(2I - N_nX_k\right), \quad X_0 = \frac{I}{\|N_n\|_\infty},\ X_k \longrightarrow X_n$ $N_{n+1} = \frac{1}{2}\left(N_n + Y_n\right), \quad N_0 = I,\ N_n \longrightarrow A^{-1/2}$ $Y_{k+1} = X_k\left(2I - M_nY_k\right), \quad Y_0 = \frac{I}{\|M_n\|_\infty},\ Y_k \longrightarrow Y_n$
4a	$G_{n+1} = 4G_n(I + G_n)^{-2}, \quad G_0 = A,\ G_n \longrightarrow I$ $R_{n+1} = \frac{1}{2}R_n\left(I + G_n\right), \quad R_0 = I,\ R_n \longrightarrow A^{1/2}$	4b	$G_{n+1} = 4G_nX_n^2, \quad G_0 = A,\ G_n \longrightarrow I$ $X_n = (I + G_n)$ $Y_{k+1} = Y_k\left(2I - X_nY_k\right), \quad Y_0 = \frac{I}{\|X_n\|_\infty},\ Y_k \longrightarrow X_n$ $R_{n+1} = \frac{1}{2}R_n\left(I + G_n\right), \quad R_0 = I,\ R_n \longrightarrow A^{1/2}$
5a	$X_{n+1} = \frac{1}{2}I + X_n^{-1} - \frac{1}{2}X_n^{-2}, \quad X_0 = \frac{1}{2}(I + A),\ X_n \longrightarrow I$ $W_{n+1} = W_nX_n, \quad W_0 = I,\ W_n \longrightarrow A^{1/2}$	5b	$X_{n+1} = \frac{1}{2}I + Y_n - \frac{1}{2}Y_n^2, \quad X_0 = \frac{1}{2}(I + A),\ X_n \longrightarrow I$ $Y_{k+1} = Y_k\left(2I - X_nY_k\right), \quad Y_0 = \frac{I}{\|X_n\|_\infty},\ Y_k \longrightarrow Y_n$ $W_{n+1} = W_nX_n, \quad W_0 = I,\ W_n \longrightarrow A^{1/2}$
6a	$U_{n+1} = U_n + V_n, \quad U_0 = I$ $V_{n+1} = AU_n + V_n, \quad V_0 = 0$ $P_n = V_nU_n^{-1}$ $P_n \longrightarrow A^{1/2}$	6b	$U_{n+1} = U_n + V_n, \quad U_0 = I$ $V_{n+1} = AU_n + V_n, \quad V_0 = 0$ $\delta_n = V_{n+1}U_n - V_nU_{n+1} \longrightarrow 0$ $V_nU_n^{-1} \longrightarrow A^{1/2}$
7a	$a_{n+1} = a_n^2 + Ab_n^2, \quad a_0 = I$ $b_{n+1} = 2a_nb_n, \quad b_0 = I$ $P_n = Ab_na_n^{-1}$ $P_n \longrightarrow A^{1/2}$	7b	$a_{n+1} = a_n^2 + Ab_n^2, \quad a_0 = I$ $b_{n+1} = 2a_nb_n, \quad b_0 = I$ $e_{n+1} = e_n^2, \quad e_0 = I - A$ $e_n = a_n^2 - Ab_n^2 \longrightarrow 0$ $Ab_na_n^{-1} \longrightarrow A^{1/2}$
8a	$P_{n+1} = 2\left(P_n^{-1} + P_nA^{-1}\right)^{-1}, \quad P_0 = A,\ P_n \longrightarrow A^{1/2}$	8b	$P_{n+1} = 2R_n, \quad P_0 = A,\ P_n \longrightarrow A^{1/2}$ $Z_n = \left(X_n + P_nA^{-1}\right)$ $X_{k+1} = X_k\left(2I - Z_nX_k\right), \quad X_0 = \frac{I}{\|Z_n\|_\infty},\ X_k \longrightarrow R_n$ $Y_{k+1} = X_k\left(2I - P_nY_k\right), \quad Y_0 = \frac{I}{\|P_n\|_\infty},\ Y_k \longrightarrow X_n$

accurate solutions as verified trough the following simulation examples.

All the algorithms are considered inversion free iterative algorithms, when Moore-Penrose matrix inversion is applied.

Example 3. Consider the 2×2 symmetric and positive definite matrix

$$A = \begin{bmatrix} 0.9 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}. \tag{54}$$

All the inversion use and inversion free iterative algorithms have been applied with convergence criterion $\varepsilon = 10^{-6}$ and compute the same principal square root of A:

$$A^{1/2} = \begin{bmatrix} 0.9126 & 0.2592 \\ 0.2592 & 1.0163 \end{bmatrix}. \tag{55}$$

The proposed algebraic method has also been applied and computes the accurate principal square root of A.

Example 4. Consider the 2×2 matrix

$$A = \begin{bmatrix} 1.0 & 0.5 \\ 0.2 & 4.0 \end{bmatrix}. \tag{56}$$

All the iterative algorithms have been applied and compute the same principal square root of A:

$$A^{1/2} = \begin{bmatrix} 0.9944 & 0.1671 \\ 0.0669 & 1.9972 \end{bmatrix}. \tag{57}$$

Algorithm 7(b) does not converge to the principal square root of A. But, the algorithm has been used to calculate the principal square root of A^{-1}, from where we are able to find the principal square root of A (we need only two matrix inversions).

The proposed algebraic method has also been applied and computes the accurate principal square root of A.

Example 5. This example is taken from [2]. Consider the 3×3 matrix

$$A = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.2 & 0.7 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}. \tag{58}$$

All the iterative algorithms have been applied and compute the same principal square root of A:

$$A^{1/2} = \begin{bmatrix} 0.7572 & 0.1883 & 0.0544 \\ 0.1248 & 0.8208 & 0.0544 \\ 0.0567 & 0.0522 & 0.8911 \end{bmatrix}. \tag{59}$$

Algorithm 6(b) is not stable: it converges to the right value of the principal square root of A after 7 iterations, but after 144 iterations it becomes to diverge and then it converges after 161 iterations to a value, which is not the principal square root of A.

The proposed algebraic method has also been applied and computes the accurate principal square root of A.

Example 6. Consider the 4×4 matrix

$$A = \begin{bmatrix} 1 & 0.990 & 0.981 & 0.947 \\ 0.890 & 1 & 0.980 & 0.765 \\ 0.981 & 0.980 & 1 & 0.395 \\ 0.942 & 0.961 & 0.945 & 1 \end{bmatrix}. \tag{60}$$

All the iterative algorithms have been applied and compute the same principal square root of A:

$$A^{1/2} = \begin{bmatrix} 0.5905 & 0.4175 & 0.4200 & 1.5953 \\ 0.2600 & 0.6228 & 0.5139 & 0.5807 \\ 0.7144 & 0.5838 & 0.6952 & -0.2191 \\ 0.4077 & 0.4002 & 0.3804 & 0.7892 \end{bmatrix}. \tag{61}$$

Algorithms 6(a), 6(b), 7(a), and 7(b) are not stable.

The proposed algebraic method has also been applied and computes the accurate principal square root of A.

Example 7. Consider the 6×6 matrix

$$A = \begin{bmatrix} 4 & 8 & 9 & 1 & 2 & 7 \\ 1 & 2 & 6 & 3 & 1 & 9 \\ 1 & 3 & 10 & 2 & 3 & 3 \\ 1 & 5 & 2 & 7 & 5 & 3 \\ 2 & 1 & 1 & 2 & 4 & 6 \\ 2 & 1 & 3 & 3 & 2 & 8 \end{bmatrix}. \tag{62}$$

All the iterative algorithms have been applied and compute the same principal square root of A:

$$A^{1/2} = \begin{bmatrix} 1.8975 & 2.4661 & 1.2370 & -0.1279 & 0.5220 & 0.0931 \\ 0.0018 & 1.1396 & 1.1793 & 0.4085 & -0.1924 & 2.2785 \\ 0.1371 & 0.5711 & 3.0104 & 0.2745 & 0.5581 & 0.1748 \\ 0.1380 & 1.2590 & 0.0904 & 2.499 & 1.2067 & -0.2751 \\ 0.3989 & -0.0282 & -0.0210 & 0.3148 & 1.7838 & 1.3116 \\ 0.3744 & -0.2178 & 0.4665 & 0.5487 & 0.1801 & 2.8800 \end{bmatrix}. \tag{63}$$

Algorithm 4(b) does not converge to the principal square root of A. But, it has been used to calculate the principal square root of A^{-1}, from where we are able to find the principal square root of A (we need only two matrix inversions).

The proposed algebraic method has also been applied and computes the accurate principal square root of$A$.

We are able to conclude that Algorithms 6 and 7 are not always stable. Note that if any algorithm is not stable we are able to try to compute the principal square root of A^{-1}, from where we are able to find the principal square root of A (we need only two matrix inversions). Finally, the proposed algebraic method has also been applied and computes the accurate principal square root of A.

6. Conclusions

In this paper, a survey of classical iterative algorithms for computing the principal square root of a matrix is presented; these algorithms require matrix inversion at every iteration. New algorithms for computing the principal square root of a matrix are proposed. The novelty of this work is the derivation of inversion free algorithms. The elimination of the matrix inversion requirement can be achieved by using the Schulz

iteration. The proposed algorithms are derived by using the Newton method or by applying the Bernoulli substitution (in a special case of the continuous time Riccati equation) or by using the doubling principle. An inversion free algebraic algorithm is also derived.

Note that, among the proposed algorithms, the one per step iterative algorithm, the one doubling iterative algorithm and the algebraic algorithm, are associated with the solution of a special case of the continuous time Riccati equation, which arises in linear estimation. The relations between these algorithms and the classical Newton method are established.

All the proposed inversion free iterative algorithms compute the principal square root of an $n \times n$ matrix achieving $O(n^3)$ complexity, due to the fact that the proposed inversion algorithms require matrix additions and matrix multiplications. The algorithms avoid matrix inversion. This means that they avoid possible problems in matrix inversion (at every iteration) and that they are easily programmable.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. A. Bini, N. J. Higham, and B. Meini, "Algorithms for the matrix pth root," *Numerical Algorithms*, vol. 39, no. 4, pp. 349–378, 2005.

[2] C.-H. Guo and N. J. Higham, "A Schur-Newton method for the matrix pth root and its inverse," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 788–804, 2006.

[3] N. J. Higham, "Newton's method for the matrix square root," *Mathematics of Computation*, vol. 46, no. 174, pp. 537–549, 1986.

[4] P. Laasonen, "On the iterative solution of the matrix equation $AX^2 - I = 0$," *Mathematical Tables and Other Aids to Computation*, vol. 12, pp. 109–116, 1958.

[5] D. G. Lainiotis, N. D. Assimakis, and S. K. Katsikas, "Fast and stable algorithm for computing the principal square root of a complex matrix," *Neural, Parallel & Scientific Computations*, vol. 1, no. 4, pp. 467–476, 1993.

[6] L. S. Shieh, S. R. Lian, and B. C. Mcinnis, "Fast and stable algorithms for computing the principal square root of a complex matrix," *IEEE Transactions on Automatic Control*, vol. 32, no. 9, pp. 820–822, 1987.

[7] J. S. H. Tsai, L. S. Shieh, and R. E. Yates, "Fast and stable algorithms for computing the principal nth root of a complex matrix and the matrix sector function," *Computers & Mathematics with Applications*, vol. 15, no. 11, pp. 903–913, 1988.

[8] E. Deadman, N. J. Higham, and R. Ralha, "Blocked schur algorithms for computing the matrix square root," in *Proceedings of the 11th International Conference on Applied Parallel and Scientific Computing (PARA '12)*, vol. 7782 of *Lecture Notes in Computer Science*, pp. 171–182, Springer, 2013.

[9] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering, Transactions of the ASME D*, vol. 83, pp. 95–107, 1961.

[10] B. Yuttanan and C. Nilrat, "Roots of Matrices," *Songklanakarin Journal of Science and Technology*, vol. 27, no. 3, pp. 659–665, 2005.

[11] C. S. Kenney and R. B. Leipnik, "Numerical integration of the differential matrix Riccati equation," *IEEE Transactions on Automatic Control*, vol. 30, no. 10, pp. 962–970, 1985.

[12] D. G. Lainiotis, "Partitioned Riccati solutions and integration-free doubling algorithms," *IEEE Transactions on Automatic Control*, vol. 21, no. 5, pp. 677–689, 1976.

[13] X. Zhan, "Computing the extremal positive definite solutions of a matrix equation," *SIAM Journal on Scientific Computing*, vol. 17, no. 5, pp. 1167–1174, 1996.

[14] M. Adam and N. Assimakis, *Matrix Equations Solutions Using Riccati Equation Theory and Applications*, LAMPERT Academic Publishing, 2012.

[15] M. Adam, N. Assimakis, G. Tziallas, and F. Sanida, "Riccati equation solution method for the computation of the solutions of $X + A^T X^{-1} A = Q$ and $X - A^T X^{-1} A = Q$," *The Open Applied Informatics Journal*, vol. 3, pp. 22–33, 2009.

[16] N. Assimakis and M. Adam, "Iterative and algebraic algorithms for the computation of the steady state Kalman filter gain," *ISRN Applied Mathematics*, vol. 2014, Article ID 417623, 10 pages, 2014.

[17] D. R. Vaughan, "A nonrecursive algebraic solution for the discrete time Riccati equation," *IEEE Transactions on Automatic Control*, vol. 15, no. 5, pp. 597–599, 1970.