Research Article

An Algorithm for Computing Geometric Mean of Two Hermitian Positive Definite Matrices via Matrix Sign

F. Soleymani, 1 M. Sharifi, 1 S. Shateyi, 2 and F. Khaksar Haghani 1

1 Department of Mathematics, Islamic Azad University, Shahrekord Branch, Shahrekord, Iran
2 Department of Mathematics and Applied Mathematics, School of Mathematical and Natural Sciences, University of Venda, Private Bag X5050, Thohoyandou 0950, South Africa

Correspondence should be addressed to S. Shateyi; stanford.shateyi@univen.ac.za

Received 7 May 2014; Revised 12 July 2014; Accepted 23 July 2014; Published 6 August 2014

Academic Editor: Alicia Cordero

Copyright © 2014 F. Soleymani et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Using the relation between a principal matrix square root and its inverse with the geometric mean, we present a fast algorithm for computing the geometric mean of two Hermitian positive definite matrices. The algorithm is stable and possesses a high convergence order. Some experiments are included to support the proposed computational algorithm.

1. Introduction

This paper tries to provide a fast algorithm for finding the geometric mean of two Hermitian positive definite matrices via the application of matrix sign function. It is known that the scalar arithmetic-geometric mean \( \text{agm}(a, b) \) of two (nonnegative) numbers \( a \) and \( b \) is defined by starting with \( a_0 = a \) and \( b_0 = b \) and then iterating

\[
    a_{k+1} = \frac{1}{2} \left[ a_k + b_k \right], \\
    b_{k+1} = \sqrt{a_k b_k}, \quad k = 0, 1, 2, \ldots,
\]

until \( a_k = b_k \) to the desired precision. The scalar sequences of \( \{a_k\} \), \( \{b_k\} \) converge to each other. Note that in (1), \( \sqrt{a_k b_k} \) is the geometric mean of two positive numbers \( a_k \) and \( b_k \) per computing cycle. Although this iterative formula is quite easy and reliable for two scalars, its extension for general square and nonsingular matrices is not an easy task. In this work, we are concerned with the matrix case of two square Hermitian positive definite matrices.

A right definition of the matrix geometric mean \( \text{GM}(A, B) \) of two positive definite matrices \( A \) and \( B \) can be expressed by

\[
    \text{GM}(A, B) := A (A^{-1} B)^{1/2},
\]

where given a square matrix \( M \) having no nonpositive real eigenvalues, \( M^{1/2} \) denotes the unique solution of the quadratic matrix equation

\[
    X^2 = M,
\]

whose eigenvalues lie in the right half plane. The definition (2) was given in the seventies by Pusz and Woronowicz [1]. There are some other important definitions for computing the matrix geometric mean of two matrices; see for example, the work of Lawson-Lim [2]. Note that when just two matrices are involved the theory is well developed, but in case of finding the matrix geometric mean of more than two matrices, the formulation is kind of hard; see for more [3].

A variant formulation for the scalar case of geometric mean could be expressed by

\[
    \text{gm}(a, b) = \sqrt{ab} = \left( \frac{ab}{a} \right)^{1/2} = \sqrt{a^2 b} = \sqrt{a^2} \sqrt{b},
\]

where \( a \) and \( b \) are positive.

Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 978629, 6 pages
http://dx.doi.org/10.1155/2014/978629
As could be seen in (4), the computation of scalar geometric mean is fully related to the square root of the positive scalar $a$ and its inverse. A similar and significant definition for a matrix geometric mean has been developed and suggested by Bhatia in [4] as follows:

$$A\#B = A^{1/2}(A^{-1/2}BA^{-1/2})^{1/2}A^{1/2},$$

for two Hermitian positive definite (HPD) matrices $A$ and $B$. We use the notation $\#$ to show the geometric mean of two HPD matrices.

Note that the formulas (2) and (5) coincide and are equal. For more, refer to [5].

The rest of this paper is organized as follows. In Section 2 we combine the root-finding Newton's method and a Chebyshev-Halley-type method to devise a fast iterative formula for solving nonlinear equations. In Section 3 we provide the link between nonlinear equation solvers and the calculation of some special matrix functions. This will illustrate how the new algorithms could be constructed and implemented. An implementation of the proposed iterative formula in symbolic software Mathematica [6] along a discussion about the stability of the scheme will be provided therein. Note that the idea of computing the geometric mean using the sign function can also be found in [7]. In Section 4 we show the numerical results and highlight the benefit of the proposed technique. Conclusions are drawn in Section 5.

2. Construction of a New Method

It is known that a common way for improving the convergence speed of iterative methods for solving nonlinear scalar smooth equations is to combine the already developed schemes. Hence, let us first combine the well-known method of Newton into a special method of Chebyshev-Halley-type schemes. Hence, let us first combine the well-known method of Newton into a special method of Chebyshev-Halley-type

$$x_{k+1} = y_k - \left(1 + \frac{1}{2} \left(\frac{L_k}{1 - ((4/5) L_k)}\right)\right) \frac{f(y_k)}{f'(y_k)},$$

wherein $L_k = f''(y_k) f(y_k) / f'(y_k)^2$. For obtaining a background about nonlinear equations, one may consult [9, 10].

**Theorem 1.** Let $\alpha \in D$ be a simple zero of a sufficiently differentiable function $f: D \subset \mathbb{R} \rightarrow \mathbb{R}$ for an open interval $D$, which contains $x_0$ as an initial approximation of $\alpha$. Then, the iterative expression (6) without memory satisfies the error equation below

$$e_{k+1} = \left( \frac{2c_3}{5} - c_0 c_5 \right) e_k^5 + O(e_k^6),$$

wherein $c_j = f^{(j)}(\alpha) / (j! f'(\alpha))$, $e_k = x_k - \alpha$.

**Proof.** The proof of this theorem is based on Taylor's series expansion of the iterative method (6) around the solution in the $k$th iterate. To save the space and not to be distracted from the main topic, we here exclude the proof. The steps of the proof are similar to those taken in [11].

The iterative method (6) reaches sixth-order convergence using five functional evaluations and thus achieves the efficiency index $6^{1/6} = 1.430$, which is higher than that of Newton; that is, $2^{1/2} = 1.414$. Furthermore, applying (6) for solving the polynomial equation $g(x) \equiv x^2 - 1 = 0$ has global convergence (except the points lying on the imaginary axis). This global behavior which could easily be shown by drawing the basins of attraction (6) for $g(x)$ is useful in practical matrix problems so as to allow us deal with all kinds of HPD matrices with any spectra.

**Remark 2.** Note that since the global behavior can be observed from the basins of attraction, then we do not pursue this fact by theoretical analysis for (6).

A recent discussion about the relation between matrix sign and nonlinear equation solvers is given in [12, 13]. We state that all of such extensions in the scalar case and studying their orders are of symbolic computational nature. Such an extension to the matrix environment will also be done in the next section using symbolic computations.

3. Construction of an Algorithm for Geometric Mean

The relation between Section 2 and our main aim for computing matrix geometric mean is not clear at the first sight. To illustrate this, we recall that many of the important matrix functions such as matrix square root, that is, the solution to the matrix equation (3), and matrix sign function which is the solution to the following matrix equation:

$$X^2 = I,$$

can be calculated approximately by matrix iteration functions. Such fixed-point type methods are convergent under special conditions to the aimed matrix function and are basically originated from root-finding methods [7].

On the other hand, the important definition of the matrix geometric mean (5) requires the computation of the matrix square root of $A$ and its inverse. Hence, in order to design an efficient algorithm for (5), we wish to construct an iterative expression in which we compute both $A^{1/2}$ and $A^{-1/2}$ at the same time.

Toward this aim, we apply the new nonlinear equation solver (6) for solving the matrix equation (8). This application would yield the following matrix iteration in its reciprocal form:

$$X_{k+1} = \left(48X_k + 272X_k^3 + 272X_k^5 + 48X_k^7\right)
\times \left[7I + 148X_k^4 + 330X_k^4 + 148X_k^6 + 7X_k^8\right]^{-1},$$

with a proper $X_0$ for finding the sign matrix.
Remark 3. The iteration (9) is not a member of the Padé family of iterations introduced in [14] for matrix sign.

In order to connect (9) with our aim, we remind an important identity as follows [7]:

$$\text{sign} \begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix} = \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix},$$  \hspace{1cm} (10)

which indicates an important relationship between principal matrix square root $A^{1/2}$ and the matrix sign function.

The identity (10) has an advantage which is the computation of the principal inverse matrix square root along with the principal matrix square root at the same time. Let us in what follows first study the stability behavior of (9).

Lemma 4. Let $A = [a_{i,j}]_{n \times n}$ have no eigenvalues on $\Re^-$. Then, the sequence $\{X_k\}_{k=0}^{\infty}$ generated by (9) using $X_0 = \begin{bmatrix} I & A \\ 0 & 0 \end{bmatrix}$ is asymptotically stable.

Proof. Let $\Delta_k$ be a numerical perturbation introduced at the $k$th iterate of (9). We obtain

$$\widetilde{X}_k = X_k + \Delta_k.$$  \hspace{1cm} (11)

All terms that are quadratic in their errors are removed from our analysis. This formal manipulation is meaningful if $\Delta_k$ is sufficiently small. We have

$$\widetilde{X}_{k+1} = \left(48\widetilde{X}_k + 272\widetilde{X}_k^2 + 272\widetilde{X}_k^3 + 48\widetilde{X}_k^4\right)$$
$$\times \left[7I + 148\widetilde{X}_k^2 + 330\widetilde{X}_k^4 + 148\widetilde{X}_k^6 + 7\widetilde{X}_k^8\right]^{-1}$$
$$= \left(48(X_k + \Delta_k) + 272(X_k + \Delta_k)^3 + 272(X_k + \Delta_k)^5 + 48(X_k + \Delta_k)^7\right)$$
$$\times \left[7I + 148(X_k + \Delta_k)^2 + 330(X_k + \Delta_k)^4 + 148(X_k + \Delta_k)^6 + 7(X_k + \Delta_k)^8\right]^{-1}. \hspace{1cm} (12)

Note that the commutativity between $X_k$ and $\Delta_k$ is not used throughout this lemma. To simplify this, the following identity will be used (for any nonsingular matrix $B$ and the matrix $C$):

$$(B + C)^{-1} \approx B^{-1} - B^{-1}CB^{-1}. \hspace{1cm} (13)$$

Simplifying yields

$$\widetilde{X}_{k+1} = \left(S + \frac{5}{2}\Delta_k + \frac{3}{2}S\Delta_k S\right)\left(I + 2S\Delta_k + 2\Delta_k S\right)^{-1}$$
$$\approx \left(S + \frac{5}{2}\Delta_k + \frac{3}{2}S\Delta_k S\right)\left(I - 2S\Delta_k - 2\Delta_k S\right), \hspace{1cm} (14)$$

where $S^2 = I$, $S^{-1} = S$, and for enough large $k$, we assumed $X_k \approx S$. After some algebraic manipulations and using $\Delta_{k+1} = \widetilde{X}_{k+1} - X_{k+1} \approx \widetilde{X}_{k+1} - S$, we conclude that

$$\Delta_{k+1} \approx \frac{1}{2}\Delta_k - \frac{1}{2}S\Delta_k S. \hspace{1cm} (15)$$

Applying (15) recursively, we have

$$\|\Delta_{k+1}\| \leq \frac{1}{2^{k+1}}\|\Delta_0 - S\Delta_0 S\|. \hspace{1cm} (16)$$

From (16), we can conclude that the perturbation at the iterate $k+1$ is bounded. This allows us to conclude that the perturbation at a given step is bounded at the subsequent steps. Therefore, the sequence $\{X_k\}$ generated by (9) is asymptotically stable. \hfill $\square$

Note that Lemma 4 just shows the asymptotical stability. In general, the fixed-point iteration $X_{k+1} = g(X_k)$ is asymptotically stable in a neighborhood of a fixed point $X$ if Fréchet derivative $dg_X$ has bounded powers [7]. Furthermore, if $X_{k+1} = g(X_k)$ is any super-linearly convergent iteration for $S = \text{sign}(X_0)$, then $dg_X(E) = L_{k}(E) = (1/2)(E - SES)$, where $L_k$ is the Fréchet derivative of the matrix sign function at $S$. Hence $dg_X$ is idempotent ($dg_X^2 = dg_X$) and the iteration is stable, thus all sign iterations, such as (9), are automatically stable. It is now easy to deduce the following convergence theorem for (9).

Theorem 5. Let $A = [a_{i,j}]_{n \times n}$ have no eigenvalues on $\Re^-$. If $X_0 = \begin{bmatrix} I & A \\ 0 & 0 \end{bmatrix}$, then the iterative method (9) converges to $S = \text{sign}(\begin{bmatrix} I & A \\ 0 & 0 \end{bmatrix})$.

Proof. The convergence of rational iterations can be analyzed in terms of the convergence of the eigenvalues of the matrices $X_k$. The reason for this is that if $X$ has a Jordan decomposition $X = ZJZ^{-1}$, then $R(X) = ZR(J)Z^{-1}$. Let $A$ have a Jordan canonical form arranged as $Z^{-1}AZ = \Lambda$, where $Z$ is a nonsingular matrix. It is also known that [7]

$$\text{sign}(\Lambda) = \text{sign}(Z^{-1}AZ) = Z^{-1}\text{sign}(A)Z. \hspace{1cm} (17)$$

If we define $D_k = Z^{-1}X_kZ$, then from the method (9), we obtain

$$D_{k+1} = \left(48D_k + 272D_k^3 + 272D_k^5 + 48D_k^7\right)$$
$$\times \left[7I + 148D_k^2 + 330D_k^4 + 148D_k^6 + 7D_k^8\right]^{-1}. \hspace{1cm} (18)$$

Notice that if $D_0$ is a diagonal matrix then, based on an inductive proof, all successive $D_k$ are diagonal too. From (18), it is enough to prove that $\{D_k\}$ converges to $\text{sign}(\Lambda)$, in order to ensure the convergence of the sequence generated by (9).

We can write (18) as $n$ uncoupled scalar iterations to solve $g(x) = x^3 - 1 = 0$, given by

$$d_{k+1}^i = \frac{48d_k^i + 272d_k^{3i} + 272d_k^5 + 48d_k^7}{7 + 148d_k^2 + 330d_k^4 + 148d_k^6 + 7d_k^8}, \hspace{1cm} (19)$$

where $d_k^i = (D_k)_{ij}$ and $1 \leq i \leq n$. From (18) and (19), it is enough to study the convergence of $\{d_k^i\}$ to $\text{sign}(\lambda_i)$, for all $1 \leq i \leq n$.

From (19) and since the eigenvalues of $A$ are not pure imaginary, we have that $\text{sign}(\lambda_i) = s_i = \pm 1$. Thus, we attain

$$\frac{d_{k+1}^i - 1}{d_{k+1}^i + 1} = \frac{-1 + d_k^i(6 + 7d_k^i)}{1 + d_k^i(6 + 7d_k^i)}. \hspace{1cm} (20)$$
Abstract and Applied Analysis

Since $|d^k_\alpha| = |\lambda| > 0$, we have
\[
\lim_{k \to \infty} \left| \frac{d^k_{k+1} - 1}{d^k_{k+1} + 1} \right| = 0, \tag{21}
\]
and $\lim_{k \to \infty} |d^k_\alpha| = 1 = |\text{sign}(\lambda)|$. This shows that $\{d^k_\alpha\}$ is convergent. Now, it could be easy to conclude that $\lim_{k \to \infty} D_k = \text{sign}(A)$. Finally, we have
\[
\lim_{k \to \infty} X_k = Z \left( \lim_{k \to \infty} D_k \right) Z^{-1} = Z \text{ sign } (A) \quad Z^{-1} = \text{ sign } (A). \tag{22}
\]
The proof is complete.

The iteration (9) requires one matrix inversion before computing step and obtains both $\sqrt[n]{A}$ and $A^{-1/2}$ which are of interest in (5). The implementation of (9) for computing principal square roots requires a sharp attention so as to save much effort. Since the intermediate matrices are all sparse (at least half of the entries are zero), then one could simply use sparse approximation techniques to save up the memory and time.

An implementation of (9) to compute $A \# B$ for two HPD matrices in the programming package Mathematica is brought forward as in (Algorithm 1).

In Algorithm 1, the four-argument function MGM computes $A \# B$ by entering the four arguments “matrix $A$,” “matrix $B$,” “the maximum number of iterations that (9) is allowed to take,” and the “tolerance” of the stopping termination in the infinity norm $\|X_{k+1} - X_k\|_\infty/\|X_k\|_\infty \leq \varepsilon$.

Note that for computing the principal matrix square root of $(A^{-1/2}BA^{-1/2})^{1/2}$, we have used the Jordan Canonical approach, which has been provided in the general form for computing matrix functions in the code FunM.

4. Experiments

We test the contributed method (9) denoted by PM using Mathematica 8 in machine precision. Apart from this scheme, another iterative method which is known as DB method [15] and given by
\[
Y_0 = A, \quad Z_0 = I, \quad k = 0, 1, \ldots, \nonumber
\]
\[
Y_{k+1} = \frac{1}{2} \left[ Y_k + Z_k^{-1} \right], \quad Z_{k+1} = \frac{1}{2} \left[ Z_k + Y_k^{-1} \right], \tag{23}
\]
is considered. This method generates the sequences $\{Y_k\}$ and $\{Z_k\}$ which converge to $A^{1/2}$ and $A^{-1/2}$, respectively.

We remark that the first and the second substeps of (6) result in the quadratic Newton’s method (NM) and cubic Chebyshev-Halley’s method (CHM) for matrix sign [7] in what follows:
\[
X_{k+1} = \frac{1}{2} \left( X_k + X_k^{-1} \right), \quad X_{k+1} = \left( 16X_k + 24X_k^3 \right) \left[ 3I + 30X_k^2 + 7X_k^4 \right]^{-1}. \tag{24}
\]

Example 6. As the first example, we consider the matrix $A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 3 & 1 \\ -1 & 2 \end{pmatrix}$, whereas its exact matrix geometric mean is given by [16]: $A \# B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$.

The proposed approach converges to the solution matrix in 2 iterations, which shows a completely fast convergence.

Example 7. We now consider the two HPD matrices as follows:
\[
A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 3 & 2 \\ 2 & 5 \end{pmatrix}, \tag{25}
\]
when $n = 250$ and compute $A \# B$.

We compare the behavior of different methods and report the numerical results using $\|X_{k+1} - X_k\|_\infty/\|X_k\|_\infty \leq \varepsilon = 10^{-5}$ in Figure 1. As could be seen, the numerical results are in harmony with the theoretical aspects of Section 3 and show a fast convergence for the proposed method (9) instead of DB, NM, and CHM.

5. Conclusions

Based on the quadratical Newton’s scheme and the cubical method of Chebyshev-Halley, we have developed an iterative method with sixth order of convergence for solving nonlinear scalar smooth equations. The computational efficiency showed its superiority in contrast to NM. The method with global behavior for finding matrix sign function has been extended for computing the principal matrix square root and its inverse. This procedure was followed so as to build a fast algorithm for finding the matrix geometric mean of two HPD matrices. We also have studied the asymptotical stability for the proposed technique.

To illustrate the new technique some numerical examples were presented. Computational results have justified robust and efficient convergence behavior of the proposed method. Similar numerical experimentations, carried out for a number of problems of different types, confirmed the above conclusions to a great extent.

We conclude the paper with the remark that in many numerical applications high precision in computations is required. The results of numerical experiments show that the high order efficient methods such as (9) associated with a multiple-precision arithmetic are very useful, because they yield a clear reduction in the number of iterates.
Abstract and Applied Analysis

ClearAll["Global"]
FunM[fun_, X_] := Module[{faux, dim, mataux, JordanD, sim, JordanF, eps, fdiag, diagQ, fauxD},
dim = Length@X;
faux[xx_, i_, j_] := Which[i <= j, 1/Abs[i - j]! (D[fun, {x, Abs[i - j]}]) /. x -> xx, True, 0];
mataux[Y_] := Table[faux[Y[[i, j]]], {i, 1, dim}, {j, 1, dim}];
JordanD = JordanDecomposition[X] // N; sim = JordanD[[1]]; JordanF = JordanD[[2]]; eps = 1*10^-10;
diagQ = Norm[JordanF - DiagonalMatrix[Diagonal[JordanF]]]; fauxD[xx_] := (fun) /. x -> xx;
fdiag = DiagonalMatrix[Map[fauxD, Diagonal[JordanF]]];
Which[diagQ < eps, sim.fdiag.Inverse[sim], True, sim.mataux[JordanF].Inverse[sim]]]

GM[A_, B_, maxIterations_, tolerance_] := Module[{k = 0},
{n, n} = Dimensions[A]; Id = SparseArray[{i, i} -> 1., {n, n}];
A1 = SparseArray@ArrayFlatten[{{0, N@A}, {Id, 0}}]; Y[0] = A1;
R[0] = 1; Id2 = SparseArray[{i, i} -> 1., {2 n, 2 n}];
Quiet@While[k < maxIterations && R[k] > tolerance,
Y2 = Y[k].Y[k]; Y3 = Y2.Y[k]; Y4 = Y3.Y[k];
Y5 = Y4.Y[k]; Y6 = Y5.Y[k]; Y7 = Y6.Y[k]; Y8 = Y7.Y[k];
11 = SparseArray[7 Id2 + 148 Y2 + 330 Y4 + 148 Y6 + 7 Y8];
12 = SparseArray@ArrayFlatten[{{Inverse@11[[1 ;; n, 1 ;; n]], 0, 0, Inverse@11[[n + 1 ;; 2 n, n + 1 ;; 2 n]]}}];
Y[k + 1] = SparseArray[48 Y[k] + 272 Y3 + 272 Y5 + 48 Y7].12;
R[k + 1] = Norm[Y[k + 1] - Y[k], Infinity]/Norm[Y[k + 1], Infinity]; k++;
AS = Y[k][{1;; n, n + 1;; 2 n}]; IAS = Y[k][{n + 1;; 2 n, 1;; n}];
Mat = (IAS.B.IAS); AS.FunM[Sqrt[x], Mat].AS]

Algorithm 1

![Log plot comparison of number of iterations in Example 7](image)

Figure 1: The log plot for the comparison of number of iterations in Example 7.

Conflict of Interests

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

Acknowledgment

The authors are grateful to anonymous reviewers for carefully reading the paper and helping to improve the presentation.

References


Submit your manuscripts at http://www.hindawi.com