Numerical Approximation of the Product of the Square Root of a Matrix With a Vector

E. J. Allen[*], J. Baglama[**], and S. K. Boyd[*]

Department of Mathematics and Statistics[*]
Texas Tech University
Lubbock, TX 79409-1042

Department of Mathematical Sciences[**]
Ball State University
Muncie, IN 47306

Abstract

Given an $n \times n$ symmetric positive definite matrix A and a vector $\vec{c}$, two numerical methods for approximating $A^{1/2}\vec{c}$ are developed, analyzed, and computationally tested. The first method applies a Newton iteration to a specific nonlinear system to approximate $A^{1/2}\vec{c}$ while the second method applies a step-control method to numerically solve a specific initial-value problem to approximate $A^{1/2}\vec{c}$. Assuming that A is first reduced to tridiagonal form, the first method requires $O(n^2)$ operations per step while the second method requires $O(n)$ operations per step. In contrast, numerical methods that first approximate $A^{1/2}$ and then compute $A^{1/2}\vec{c}$ generally require $O(n^3)$ operations per step.

Keywords: Matrix square root, numerical method, nonlinear system, initial-value problem, Lanczos method

Mathematics subject classification: AMS(MOS) 65F30

1 Introduction

Let A be a given $n \times n$ symmetric positive definite matrix and $\vec{c}$ a given vector of length n. In this investigation, numerical methods for directly approximating $A^{1/2}\vec{c}$ are described, analyzed, and computationally compared. These methods differ from the approach of first approximating $A^{1/2}$ and then calculating $A^{1/2}\vec{c}$ which generally require $O(n^3)$ operations per step. The numerical methods described here for directly approximating $A^{1/2}\vec{c}$ can be achieved in $O(n^2)$ or in $O(n)$ operations per step assuming that A is first reduced to tridiagonal form. The initial reduction of A to tridiagonal form requires about $4n^3/3$ operations.

These numerical methods can be applied, for example, in numerical solution of stochastic differential equations. Two specific such applications occur in population dynamics [1] and in neutron transport [2]. In these problems, systems of Ito stochastic differential equations arise of the form

$$\begin{cases} d\vec{y}/dt = f(\vec{y},t) + A^{1/2}(\vec{y},t)d\vec{W}(t)/dt \\ \vec{y}(0) = \vec{y}(0) \end{cases} \tag{1.1}$$

where $A(\vec{y},t)$ is a known $n \times n$ symmetric positive definite matrix, vector $\vec{y}$ is a function of time t, and $\vec{W}(t)$ is the n-dimensional Wiener process. (Any symmetric square root of A can be substituted into (1.1). This follows from the forward Kolmogorov equation [3] which describes the probability distribution of $\vec{y}(t)$. For $A^{1/2}$ symmetric, the forward Kolmogorov equation depends on the elements of A and not of $A^{1/2}$.) To solve (1.1) numerically using Euler's method, for example, results in the iteration:

$$\vec{y}_{m+1} = \vec{y}_m + f(\vec{y}_m,t_m)\Delta t + A^{1/2}(\vec{y}_m,t_m)\vec{\eta}_m\sqrt{\Delta t} \tag{1.2}$$

for $m = 0,1,2\ldots$ where $t_m = m\Delta t$, $\vec{y}_m \approx \vec{y}(t_m)$, and $(\vec{\eta}_m)_i \in N(0,1)$ for $i = 1,2,\ldots,n$. To compute (1.2) at each time step, it is necessary to approximate the product $A^{1/2}(\vec{y}_m,t_m)\vec{\eta}_m$ given $A(\vec{y}_m,t_m)$ and $\vec{\eta}_m$. (It does not appear possible to reformulate (1.1) or to devise a numerical method where the square root of A does not appear in the iteration.)

Before describing the numerical methods for directly approximating $A^{1/2}\vec{c}$, it is important to review numerical methods for approximating $A^{1/2}$. There has been much recent interest in developing numerical methods for calculating $A^{1/2}$ [4-16]. In particular, Lakic and Petkovic[13] derive third-order methods, based on the Euler-Chebyshev method, for $n \times n$ matrices with real nonnegative eigenvalues. In their second method, $T_m \to A^{1/2}$ as $m \to \infty$ where $B = A/\|A\|, R_0 = I, S_0 = B$, and

$$\begin{cases} R_{m+1} = R_m(\frac{3}{8}I + \frac{3}{4}S_m(I - \frac{1}{6}S_m)) \\ S_{m+1} = S_m(\frac{3}{8}I + \frac{3}{4}S_m(I - \frac{1}{6}S_m))^{-2} \\ T_m = \sqrt{\|A\|}R_m. \end{cases} \tag{1.3}$$

Higham[11] recommends for A symmetric positive definite the following second-order Newton iteration in which $X_m \to A^{1/2}$ as $m \to \infty$ where $A = R^TR$ (Cholesky factorization), $Y_0 = R$, and

$$\begin{cases} Y_{m+1} = \frac{1}{2}(Y_m + Y_m^{-T}) \\ X_m = Y_m^T R. \end{cases} \tag{1.4}$$

In addition, Higham recommends using the numerically stable Schur method of Björck and Hammarling [5] for a general $n \times n$ matrix A. In this method, first the Schur form of A is computed, i.e. $A = QTQ^*$. Then $A^{1/2} = QRQ^*$ where $R = T^{1/2}$ and T is upper triangular. The matrix R can be computed using the algorithm:

$$\begin{cases} \text{for } \ j = 1 : n \\ r_{jj} = t_{jj}^{1/2} \\ \text{for } \ i = j-1 : -1 : 1 \\ r_{ij} = (t_{ij} - \sum\limits_{k=i+1}^{j-1} r_{ik} r_{kj})/(r_{ii} + r_{jj}) \\ \text{end} \\ \text{end.} \end{cases} \tag{1.5}$$

For A symmetric positive definite, the Schur form of A is QDQ^* where D is diagonal so $A^{1/2} = QD^{1/2}Q^*$. In references [5-16], many interesting methods for numerically computing $A^{1/2}$ are described. The above three methods are representative and the methods involve at least $O(n^2)$ floating point operations per step. (For example, Lu's method [14] requires about $10n^3/3 + 5n^2m/2$ operations where m is an integer that depends on the desired accuracy of the approximation.) For large systems, these methods are computationally time consuming. The only other numerical method available for estimating $A^{1/2}\vec{c}$ employs a Krylov procedure for approximating functions [17]. When the $n \times n$ symmetric matrix A is very large, then $A^{1/2}\vec{c}$ can be approximated using the Krylov subspace

$$\mathbf{K}_m(A, \vec{c}) = \text{span}\{\vec{c}, A\vec{c}, A^2\vec{c}, \ldots, A^{m-1}\vec{c}\} \tag{1.6}$$

where $m \le n$. Let $V_m = [\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_m]$ be an orthonormal basis for (1.6) where $\vec{v}_1 = \frac{\vec{c}}{\|\vec{c}\|_2}$. The best approximation to $A^{1/2}\vec{c}$ from (1.6) is given by

$$A^{1/2}\vec{c} \approx \|\vec{c}\|_2 V_m V_m^T A^{1/2} V_m \vec{e}_1 \tag{1.7}$$

where $\vec{e}_1 = (1, 0, \ldots, 0)^T$. However, equation (1.7) requires $A^{1/2} V_m \vec{e}_1$. If the Lanczos method is used to create the orthonormal basis V_m then we have

$$\begin{aligned} AV_m &= V_m T_m + \vec{f}_m \vec{e}_m^{\,T} \\ T_m &= V_m^T A V_m \end{aligned} \tag{1.8}$$

where V_m is a $n \times m$ matrix, $V_m \vec{e}_1 = \frac{\vec{c}}{\|\vec{c}\|_2}$, $V_m^T V_m = I$, T_m is a $m \times m$ symmetric tridiagonal matrix, and $\vec{f}_m$ satisfies $V_m^T \vec{f}_m = 0$. Using $T_m = V_m^T A V_m$, $T_m^{1/2}$ approximates $V_m^T A^{1/2} V_m$. This gives the following approximation to $A^{1/2}\vec{c}$:

$$A^{1/2}\vec{c} \approx \|\vec{c}\|_2 V_m T_m^{1/2} \vec{e}_1. \tag{1.9}$$

However, application of this procedure still requires computing the square root of the tridiagonal matrix T_m by some numerical method such as by one of the methods described above.

In the present investigation, two numerical methods for directly approximating $A^{1/2}\vec{c}$, without explicitly approximating $A^{1/2}$, are described, analyzed, and computationally compared. The first method involves applying a modified Newton procedure to solve a specific nonlinear system. The second method involves applying a step-control method to solve a specific initial-value problem. If matrix A is initially reduced to Hessenberg form, only $O(n^2)$ arithmetic operations per step are required in the first method to compute $A^{1/2}\vec{c}$ and only $O(n)$ operations per step are required in second method to compute $A^{1/2}\vec{c}$. Reduction of A to Hessenberg form, i.e. tridiagonal form as A is symmetric, requires about $\frac{4}{3}n^3$ operations but is only performed once. Specifically, A is reduced to tridiagonal form using Householder similarity transformations to obtain

$$A = Q^T T Q \tag{1.10}$$

where T is tridiagonal and $Q^T Q = I$. Then, either the Newton or the step-control numerical method is used to calculate $T^{1/2}\hat{\vec{c}}$ where $\hat{\vec{c}} = Q\vec{c}$. Finally, $A^{1/2}\vec{c}$ is given by

$$A^{1/2}\vec{c} = Q^T T^{1/2}\hat{\vec{c}} = Q^T T^{1/2} Q\vec{c}. \tag{1.11}$$

Only once, initially, are $O(n^3)$ operations required and that is to compute Q and T. (Notice that, alternatively, A can be reduced to tridiagonal form using the Lanczos method and if $m < n$ approximation (1.9) can be applied. However, in the present investigation, Householder similarity tranformations appeared to provide greater accuracy and were used to reduce A to tridiagonal form in all the numerical examples.)

Before describing these two numerical methods, it is useful to state some important results about square roots of matrices. A nonsingular matrix always has at least one square root [18]. A singular matrix may not have a square root. Consider, for example, $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. A nonsingular matrix may have an infinite number of square roots. For example, $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}^2$ for any value of θ. An $n \times n$ matrix with n distinct nonzero eigenvalues has 2^n square roots [18]. Finally, a positive (semi)definite matrix has a unique positive (semi)definite square root [19].

In the next section, a numerical method for approximating $A^{1/2}\vec{c}$ is described that is based on solving a nonlinear system using a modified Newton's method. In the third section, it is shown that numerical solution of a certain initial-value problem yields an approximation to $A^{1/2}\vec{c}$. A step-control procedure based on the Runge-Kutta-Fehlberg method is used to approximately solve the initial-value problem to estimate $A^{1/2}\vec{c}$. These methods are computationally compared in the fourth section. The two numerical methods are completely different, the methods are useful for comparison purposes, and extensions of the methods may result in additional applications.

2 A Modified Newton Method

Let the function $F(\vec{x})$ be defined as

$$F(\vec{x}) = \begin{bmatrix} \vec{x}^T\vec{x} - \vec{c}^TA\vec{c} \\ \vec{x}^TA\vec{x} - \vec{c}^TA^2\vec{c} \\ \cdot \\ \cdot \\ \cdot \\ \vec{x}^TA^{n-1}\vec{x} - \vec{c}^TA^n\vec{c} \end{bmatrix}, \tag{2.1}$$

i.e. $(F(\vec{x}))_i = \vec{x}^TA^{i-1}\vec{x} - \vec{c}^TA^i\vec{c}$. Clearly, a solution of $F(\vec{x}) = \vec{0}$ is $\vec{x} = A^{1/2}\vec{c}$ where $A^{1/2}$ is any symmetric square root of A.

In this section, a modified Newton's method is described and analyzed for computing the solution of $F(\vec{x}) = \vec{0}$. Consider first Newton's method for solution of $F(\vec{x}) = \vec{0}$:

$$\vec{x}_{k+1} = \vec{x}_k - (F'(\vec{x}_k))^{-1}F(\vec{x}_k), \text{ for } k = 0, 1, 2, \cdots, \tag{2.2}$$

with $\vec{x}_0$ given and where

$$F'(\vec{x}) = 2\begin{bmatrix} \vec{x}^T \\ \vec{x}^TA \\ \cdot \\ \cdot \\ \cdot \\ \vec{x}^TA^{n-1} \end{bmatrix}. \tag{2.3}$$

Applying a Newton attraction theorem [20], the following result is obtained.

Theorem 2.1 *Let $A^{1/2}$ be a square root of A and let $\vec{z} = A^{1/2}\vec{c}$. Assume that $F'(\vec{z})$ is nonsingular. Then there exists a $\delta > 0$ and an open ball S about $\vec{z}$, i.e. $S = \{\vec{x} \in R^n : \|\vec{x} - \vec{z}\|_2 < \delta\}$, such that $F(\vec{x}) = \vec{0}$ has the unique solution $\vec{z}$ in S and the sequence $\{\vec{x}_k\}_{k=0}^{\infty}$ defined by (2.2) converges to $\vec{z}$ provided that $\vec{x}_0 \in S$.*

Proof: See [20], Theorem 10.2.2. □

Equation (2.2) can be written in the convenient form:

$$\begin{cases} \vec{x}_{k+1} = \frac{1}{2}\vec{x}_k + \vec{b}_k \\ F'(\vec{x}_k)\vec{b}_k = \vec{r} \end{cases} \tag{2.4}$$

where $\vec{r} = [\vec{c}^TA\vec{c} \;\; \vec{c}^TA^2\vec{c} \;\cdots\; \vec{c}^TA^{n-1}\vec{c}]^T$. Equation (2.4) is now modified to ensure that $\vec{x}_k^T\vec{x}_k = \vec{c}^TA\vec{c} = w^2$ for each k. Specifically,

$$\begin{cases} \vec{x}_{k+1} = w(\frac{1}{2}\vec{x}_k + \vec{b}_k)/w_k \\ F'(\vec{x}_k)\vec{b}_k = \vec{r} \\ w_k = ((\frac{1}{2}\vec{x}_k + \vec{b}_k)^T(\frac{1}{2}\vec{x}_k + \vec{b}_k))^{1/2} \end{cases} \tag{2.5}$$

where $w = (\vec{c}^T A\vec{c})^{1/2}$. Equation (2.5) is the modified form of Newton's method that is studied in the present investigation. Before presenting an error analysis of this method, it is worthwhile to note that often a good choice for $\vec{x}_0$ is given by

$$\vec{x}_0 = w\vec{y}/\|\vec{y}\|_2 \tag{2.6}$$

where $\vec{y}$ satisfies

$$\vec{y} = (D^{1/2} + B)\vec{c} \tag{2.7}$$

with $BD^{1/2} + D^{1/2}B = A - D$, and D is the diagonal matrix associated with A, that is $D_{ii} = A_{ii}$ for $1 \le i \le n$. This approximation results by letting $A^{1/2} = D^{1/2} + B$, squaring this expression, and neglecting the B^2 term. Matrix B can be rapidly calculated as $B_{ij} = (A-D)_{ij}/(D_{ii}^{1/2} + D_{jj}^{1/2})$ for $1 \le i, j \le n$. In addition, if the conditions of Theorem 2.1 are satisfied and $\vec{x}_0 = C_0\vec{c}$ where C_0 is symmetric, then it can be shown that $\vec{x}_k = C_k\vec{c}$ for $k = 0, 1, 2, \cdots$ where each C_k is symmetric. In this case, the sequence $\{\vec{x}_k\}_{k=0}^{\infty}$ converges to $A^{1/2}\vec{c}$ where $A^{1/2}$ is symmetric.

For method (2.5), the Newton-Kantorovich theorem [20] can be applied to prove that the method converges quadratically provided that $\|\vec{x}_0 - A^{1/2}\vec{c}\|_2$ is sufficiently small. However, for this specific problem, it is simpler to directly prove a quadratic convergence result than to show that the conditions of the Newton-Kantorovich theorem are satisfied. Therefore, the following theorem and proof are presented in this paper.

Theorem 2.2 *Let $A^{1/2}$ be a square root of symmetric positive definite $n \times n$ matrix A and define $R = F'(A^{1/2}\vec{c})$. Assume that R is nonsingular. Let $\vec{\varepsilon}_k = \vec{x}_k - A^{1/2}\vec{c}$ where $\vec{x}_k$ is defined by (2.5). Let $p = (\sum_{m=0}^{n-1} \|A\|_2^{2m})^{1/2}$. If $\|\vec{\varepsilon}_0\|_2 \le q = min(w, \frac{1}{9}p\|R^{-1}\|_2)$ where $w^2 = \vec{c}^T A\vec{c}$, then $\|\vec{\varepsilon}_{k+1}\|_2 \le 8p\|R^{-1}\|_2\|\vec{\varepsilon}_k\|_2^2$ for $k = 0, 1, 2, \ldots$ and $\vec{x}_k \to A^{1/2}\vec{c}$ as $k \to \infty$.*

Proof: The proof uses an inductive argument. Assume that $\|\vec{\varepsilon}_k\|_2 \le q$. It will be shown that this implies that $\|\vec{\varepsilon}_{k+1}\|_2 \le q$ and that $\|\vec{\varepsilon}_{k+1}\|_2 \le 8p\|R^{-1}\|\|\vec{\varepsilon}_k\|_2^2$. These inequalities imply that $\vec{\varepsilon}_k \to \vec{0}$ as $k \to \infty$.

First, since $\vec{x}_k = A^{1/2}\vec{c} + \vec{\varepsilon}_k$, by (2.5),

$$A^{1/2}\vec{c} + \vec{\varepsilon}_{k+1} = w(\frac{1}{2}A^{1/2}\vec{c} + \frac{1}{2}\vec{\varepsilon}_k + \vec{b}_k)/w_k. \tag{2.8}$$

Define $\vec{z}_k$ by

$$\vec{z}_k = \vec{b}_k - \frac{1}{2}A^{1/2}\vec{c} + \frac{1}{2}\vec{\varepsilon}_k. \tag{2.9}$$

Then (2.8) becomes:

$$\vec{\varepsilon}_{k+1} = (A^{1/2}\vec{c} + \vec{z}_k)/(1 + r_k)^{1/2} - A^{1/2}\vec{c} \tag{2.10}$$

where

$$r_k = (2\vec{z}_k^T A^{1/2}\vec{c} + \vec{z}_k^T \vec{z}_k)/w^2. \tag{2.11}$$

Taking the product $\vec{\varepsilon}_{k+1}^T \vec{\varepsilon}_{k+1}$ gives

$$\|\vec{\varepsilon}_{k+1}\|_2^2 = 2w^2 - 2(A^{1/2}\vec{c})^T(A^{1/2}\vec{c} + \vec{z}_k)/(1+r_k)^{1/2}. \tag{2.12}$$

Consider $\vec{z}_k$. As $F'(\vec{\varepsilon}_k + A^{1/2}\vec{c})\vec{b}_k = \vec{r}$, then

$$F'(\vec{\varepsilon}_k + A^{1/2}\vec{c})\vec{z}_k = \frac{1}{2}F'(\vec{\varepsilon}_k)\vec{\varepsilon}_k. \tag{2.13}$$

Letting $C_k = F'(\vec{\varepsilon}_k)$, (2.13) becomes

$$(R + C_k)\vec{z}_k = \frac{1}{2}C_k\vec{\varepsilon}_k. \tag{2.14}$$

But $\|C_k\|_2 \leq 2p\|\vec{\varepsilon}_k\|_2$ and by hypothesis, $\|\vec{\varepsilon}_k\|_2 \leq q$. Hence $R + C_k$ is nonsingular as $R + C_k = R(I + R^{-1}C_k)$ and $\|R^{-1}C_k\|_2 \leq 2\|R^{-1}\|pq \leq \frac{2}{9}$. Therefore, $\vec{z}_k = \frac{1}{2}(R+C_k)^{-1}C_k\vec{\varepsilon}_k$ and it follows that

$$\begin{aligned}\|\vec{z}_k\|_2 &\leq \frac{1}{2}\|R^{-1}\|_2\|C_k\|_2\|\vec{\varepsilon}_k\|_2/(1-\frac{2}{9})\\ &\leq \frac{9}{7}p\|R^{-1}\|_2\|\vec{\varepsilon}\|_2^2 \leq \frac{1}{7}w.\end{aligned} \tag{2.15}$$

Returning to r_k, the above inequalities imply that

$$\begin{aligned}|r_k| &\leq \frac{2}{w}\|\vec{z}_k\|_2 + \frac{1}{w^2}\|\vec{z}_k\|_2^2\\ &\leq \frac{15}{7w}\|\vec{z}_k\|_2 \leq \frac{135}{49w}p\|R^{-1}\|_2\|\vec{\varepsilon}_k\|_2^2 \leq \frac{15}{49}.\end{aligned} \tag{2.16}$$

Considering (2.12), but with $(1+r_k)^{-1/2}$ expanded three terms in a Taylor series, gives

$$\|\vec{\varepsilon}_{k+1}\|_2^2 = 2w^2 - 2(A^{1/2}\vec{c})^T(A^{1/2}\vec{c} + \vec{z}_k)(1 - \frac{1}{2}r_k + \frac{3}{8}(1+\gamma_k)^{-5/2}r_k^2) \tag{2.17}$$

for some γ_k with $0 \leq |\gamma_k| \leq |r_k| \leq \frac{15}{49}$. The above expression can be simplified to

$$\begin{aligned}\|\vec{\varepsilon}_{k+1}\|_2^2 &= 2(\vec{z}_k^T A^{1/2}\vec{c})^2/w^2 + \vec{z}_k^T\vec{z}_k + (A^{1/2}\vec{c})^T\vec{z}_k\vec{z}_k^T\vec{z}_k/w^2\\ &-2(A^{1/2}\vec{c})^T(A^{1/2}\vec{c} + \vec{z}_k)\frac{3}{8}(1+\gamma_k)^{-5/2}r_k^2.\end{aligned} \tag{2.18}$$

Applying the Cauchy-Scharwz inequality and substituting in the lower bound on γ_k yields

$$\|\vec{\varepsilon}_{k+1}\|_2^2 \leq \frac{22}{7}\|\vec{z}_k\|_2^2 + \frac{3}{4}(\frac{49}{34})^{5/2}w(w + \|\vec{z}_k\|_2)r_k^2. \tag{2.19}$$

Finally, applying inequalities (2.15) and (2.16),

$$\|\vec{\varepsilon}_{k+1}\|_2 \leq (13)^{1/2}\|\vec{z}_k\|_2 \leq 8p\|R^{-1}\|_2\|\vec{\varepsilon}_k\|_2^2. \tag{2.20}$$

Thus, as $\|\vec{\varepsilon}_k\|_2 \leq q$, then $\|\vec{\varepsilon}_{k+1}\|_2 \leq \|\vec{\varepsilon}_k\|_2 \leq q$ and the inductive proof is complete. □

There are two computational difficulties associated with applying method (2.5). First, even with A reduced to tridiagonal form, the number of operations per step is $O(n^3)$. However, the quasi-Newton procedure, Broyden's method [20,21] can be applied in $O(n^2)$ operations per step for this problem. A modified Broyden's method, analogous to method (2.5), has the form for $k = 0, 1, 2, \ldots,$

$$\begin{cases} \vec{v}_{k+1} = \vec{x}_k - H_k F(\vec{x}_k) \\ \vec{x}_{k+1} = w\vec{v}_{k+1}/(\vec{v}_{k+1}^T\vec{v}_{k+1})^{1/2} \\ H_{k+1} = H_k + (\vec{s}_k - H_k\vec{y}_k)\vec{s}_k^T H_k/\vec{s}_k^T H_k\vec{y}_k \\ \vec{y}_k = F(\vec{x}_{k+1}) - F(\vec{x}_k) \\ \vec{s}_k = \vec{x}_{k+1} - \vec{x}_k \end{cases} \tag{2.21}$$

where $H_0 = (F'(\vec{x}_0))^{-1}$. With A tridiagonal, method (2.21) requires only $O(n^2))$ operations per iteration and compares very well in accuracy with method (2.5). The second difficulty associated with this method is that $F'(\vec{x}_k)$ is ill-conditioned for n large. This leads to computational problems using either method (2.5) or (2.21). To alleviate this problem, a special Krylov subspace procedure based on the Lanczos method [22,23,24] was developed in the present investigation and is applied with method (2.5) to decompose A into a product of orthogonal and tridiagonal matrices at each iteration. When applied with A reduced to tridiagonal form, this procedure also only requires $O(n^2)$ operations per step and significantly reduces the ill-conditioning problem associated with $F'(\vec{x}_k)$ for n large.

In this procedure, the primary iteration is (2.5). However, to compute $\vec{b}_k$ from the linear system $F'(\vec{x}_k)\vec{b}_k = \vec{r}$ at each step, a special technique explained below is used. First, using the Lanczos method with initial vector $\vec{c}$, matrix A is decomposed in the form

$$AV_c = V_cT_c \tag{2.22}$$

where $V_c^TV_c = I, T_c$ is tridiagonal, $K_c = V_cR_c, K_c = [\vec{c} \quad A\vec{c} \quad \cdots \quad A^{n-1}\vec{c}]$, and $R_c = [\vec{e}_1 \quad T_c\vec{e}_1 \quad \cdots \quad T_c^{n-1}\vec{e}_1]$. For A tridiagonal, this decomposition can be performed in $O(n^2)$ operations. (Notice that this step can be skipped if matrix A is initially made tridiagonal using the Lanczos method with vector $\vec{c}$.) Next, it is noticed that

$$F'(\vec{x}_k)\vec{b}_k = \vec{r} \tag{2.23}$$

has the form

$$2K_x^T\vec{b}_k = K_c^TA\vec{c} \tag{2.24}$$

where $K_x = [\vec{x}_k \quad A\vec{x}_k \quad \cdots \quad A^{n-1}\vec{x}_k]$. Thus,

$$\vec{b}_k = \frac{1}{2}(K_x^T)^{-1} K_c^T A\vec{c} \tag{2.25}$$

and

$$\vec{b}_k^T = \frac{1}{2}\vec{c}^T A K_c K_x^{-1}. \tag{2.26}$$

The full nonsymmetric matrix K_x in (2.26) can be simplified by applying the Lanczos method to matrix A with initial vector $\vec{x}_k$. Then,

$$AV_x = V_x T_x \tag{2.27}$$

where T_x is tridiagonal, $V_x^T V_x = I$, and $K_x = V_x R_x$ with $R_x = [\vec{e}_1 \;\; T_x\vec{e}_1 \;\; \cdots \;\; T_x^{n-1}\vec{e}_1]$. In the above, however, R_x and R_c are ill-conditioned so $\vec{b}_k$ must be expressed without using R_x and R_c. To accomplish this, matrix M is defined as

$$M = R_c R_x^{-1}. \tag{2.28}$$

Then, $\vec{b}_k^T$ becomes $\vec{b}_k^T = \frac{1}{2}[\vec{c}^T A V_c R_c R_x^{-1} V_x^{-1}] = \frac{1}{2}[\vec{c}^T V_c T_c M V_x^T]$ so

$$\vec{b}_k = \frac{\|\vec{c}\|_2}{2} V_x M^T T_c \vec{e}_1 \tag{2.29}$$

as $V_c^T \vec{c} = \|\vec{c}\|_2 \vec{e}_1$. Assuming that A is tridiagonal, V_x can be found by using the Lanczos method in $O(n^2)$ operations. (This step is required for each iteration k and if A is not initially reduced to tridiagonal form, this step would require $O(n^3)$ operations rather than $O(n^2)$ operations.) Finally, by equating columns of the equation $MR_x = R_c$, upper triangular matrix M can be shown to satisfy

$$\begin{cases} \vec{e}_1 = M\vec{e}_1 \\ T_c M = M T_x. \end{cases} \tag{2.30}$$

As T_c and T_x are tridiagonal, M can be efficiently calculated column by column using (2.30) in $O(n^2)$ operations. The following MATLAB procedure illustrates one way M can be calculated.

```
$M = \text{sparse}(n, n);$
$M(1, 1) = 1;$
for $i = 2 : n$
$v1 = M(1 : i, 1 : i - 1) * T_x(1 : i - 1, i - 1);$
$v2 = T_c(1 : i, 1 : i - 1) * M(1 : i - 1, i - 1);$
$v1(i) = 0;$
$M(1 : i, i) = (v2 - v1)/T_x(i, i - 1);$
end
```

The vector $\vec{b}_k$ is now calculated using (2.29).

To summarize this procedure, method (2.5) is modified to the form:

$$\begin{cases} \vec{x}_{k+1} = w(\frac{1}{2}\vec{x}_k + \vec{b}_k)/w_k \\ \vec{b}_k = \frac{\|\vec{c}\|_2}{2} V_x M^T T_c \vec{e}_1 \\ w_k = ((\frac{1}{2}\vec{x}_k + \vec{b}_k)^T(\frac{1}{2}\vec{x}_k + \vec{b}_k))^{1/2} \end{cases} \tag{2.31}$$

where the Lanczos method is used to find T_c, T_x, and V_x, and M is computed from (2.30). Notice that this method avoids the ill-conditioned matrices $F'(\vec{x}_k), R_x$, and R_c and each step can be performed in $O(n^2)$ arithmetical operations. As shown in the fourth section, computational results obtained using method (2.31) are more stable than those obtained using methods (2.5) or (2.21).

3 An Initial-Value Problem Method

In this section, it is assumed that the $n \times n$ symmetric positive definite matrix A satisfies $\|A\|_\infty < 1$. This assumption entails no loss of generality as $A^{1/2}\vec{c} = \beta^{1/2}(A/\beta)^{1/2}\vec{c}$ for β a scalar and $\|A/\beta\|_\infty$ can be made less than unity by selecting β sufficiently large.

Consider the initial-value problem

$$\begin{cases} d\vec{x}(t)/dt = -\frac{1}{2}(At + (1-t)I)^{-1}(I - A)\vec{x}(t) \\ \vec{x}(0) = \vec{c}. \end{cases} \tag{3.1}$$

As $(I-A)$ is nonsingular and commutes with $(At+(1-t)I)^{-1}$, the solution of this initial-value problem is

$$\vec{x}(t) = (At + (1-t)I)^{1/2}\vec{c}. \tag{3.2}$$

Hence, $\vec{x}(1) = A^{1/2}\vec{c}$. Furthermore, as shown in the following theorem, $\vec{x}(1) = A^{1/2}\vec{c}$ where $A^{1/2}$ is the positive definite square root of A.

Theorem 3.1 *The solution $\vec{x}(t)$ of (3.1) satisfies $\vec{x}(1) = A^{1/2}\vec{c}$ where $A^{1/2}$ is the positive definite square root of A.*

Proof: First, as $(At + (1-t)I)^{-1}(I - A)$ is continuous on the interval $0 \le t \le 1$, system (3.1) has a unique solution on this interval [25]. Let $\vec{x}(t) = \sum_{i=1}^{n} b_i(t)\vec{z}_i$ where $\vec{z}_i, 1 \le i \le n$, are orthogonal eigenvectors of A with respective eigenvalues $\lambda_i, 1 \le i \le n$. Substituting this expression into (3.1) results in

$$\begin{cases} db_i(t)/dt = -\frac{1}{2}(1-\lambda_i)(1 + (\lambda_i - 1)t)^{-1}b_i(t) \\ b_i(0) = \vec{c}^T\vec{z}_i/\vec{z}_i^T\vec{z}_i. \end{cases} \tag{3.3}$$

The solution of (3.3) is:

$$b_i(t) = (1 + (\lambda_i - 1)t)^{1/2}\vec{c}^T\vec{z}_i/\vec{z}_i^T\vec{z}_i \text{ for } i = 1, 2, \ldots, n. \tag{3.4}$$

It follows that the solution $\vec{x}(t)$ at $t = 1$ is

$$\vec{x}(1) = \sum_{i=1}^{n} \lambda_i^{1/2} (\vec{c}^T \vec{z}_i)\vec{z}_i / \vec{z}_i^T \vec{z}_i = A^{1/2}\vec{c} \tag{3.5}$$

where $A^{1/2}$ is the positive definite square root of A. $\square$

Before continuing, it is important to note that there are an infinite number of initial-value problems whose solution at $t = 1$ is $A^{1/2}\vec{c}$. However, it is difficult to find an initial-value problem as simple as (3.1) that has two important features. The first feature is that the right-hand side of (3.1) contains no square roots and is continuous for $0 \leq t \leq 1$. The second feature, which will be shown below, is that (3.1) can be solved numerically using $O(n)$ operations per iteration if A is tridiagonal.

To estimate $A^{1/2}\vec{c}$, (3.1) is solved numerically from $t = 0$ to $t = 1$. The numerical approximation obtained for $\vec{x}(1)$ provides an estimate of $A^{1/2}\vec{c}$. There are, of course, many accurate numerical schemes for solving initial-value system (3.1). A step-control method is applied in the present investigation. (A step-control method is applied here to reduce the total number of calculations required to achieve a specified error. This allows a fair computational comparison of this method with the other methods.) The step-control procedure applied here is based on the popular Runge-Kutta-Fehlberg procedure [21,26] in which the results of two different Runge-Kutta methods (of orders 4 and 5) are combined to estimate the error at each step and control the step size.

Although not applied computationally in the present investigation, it is worthwhile to consider the simple Euler's method for this problem. Euler's method for system (3.1) has the form:

$$\begin{cases} \vec{x}_{k+1} = \vec{x}_k + \Delta t \vec{r}_k \\ (I - (I - A)t_k)\vec{r}_k = \frac{1}{2}(I - A)\vec{x}_k \end{cases} \tag{3.6}$$

for $k = 0, 1, 2, \ldots, N$ where $\vec{x}_0 = \vec{c}$, $t_k = k\Delta t$, and $\Delta t = 1/N$. Then, $\vec{x}_N \approx \vec{x}(1) = A^{1/2}\vec{c}$. In addition, if A is tridiagonal and positive definite (reduced initially to Hessenburg form), then only $O(n)$ operations are required per time step in Euler's method (3.6) to calculate $\vec{x}_{k+1}$ from $\vec{x}_k$. It is easy to see that this is likewise the case for the Runge-Kutta-Fehlberg procedure. That is, numerical solution of (3.1) requires only $O(n)$ operations per iteration after initial reduction of A to tridiagonal form.

The following proposition about iteration (3.6) is conceptually useful and supports the conclusion of Theorem 3.1. First, note that (3.6) can be put in the form:

$$\vec{x}_N = C_{N-1}C_{N-2}\ldots C_1 C_0 \vec{c} \tag{3.7}$$

where $C_k = I - \frac{\Delta t}{2}(I - (I - A)t_k)^{-1}(I - A)$ for $k = 0, 1, 2, \ldots, N - 1$. Equation (3.7) and Proposition 3.1 imply that $C_{N-1}C_{N-2}\ldots C_1 C_0$ is an approximation to the positive definite square root of A. (It is interesting to note that $C_{N-1}C_{N-2}\ldots C_1 C_0$ is similar in form to an approximation to $A^{1/2}$ developed by Lu [14] using an entirely different approach, specifically, a Pade′ approximation method.)

Proposition 3.1: *Assume that the symmetric positive definite $n \times n$ matrix A satisfies $\|A\|_\infty < 1$ and that $\Delta t < \gamma = 2(1 - \|I - A\|_2)/\|I - A\|_2$. Then $C_{N-1}C_{n-2}\dots C_1C_0$ is symmetric positive definite.*

Proof: First note that iteration (3.6) can be written as:

$$\begin{cases} \vec{x}_{k+1} = (I - \Delta t B_k)\vec{x}_k \ \text{ for } \ k = 0, 1, \dots, N-1 \\ B_k = \frac{1}{2}(I - (I - A)t_k)^{-1}(I - A) = \frac{1}{2}(I - A)(I - (I - A)t_k)^{-1}. \end{cases} \tag{3.8}$$

Note that $\|B_k\|_2 \le \frac{1}{2}\|I - A\|_2\|(I - (I - A)t_k)^{-1}\|_2 \le \frac{1}{2}\|I - A\|_2/(1 - t_k\|I - A\|_2) \le \frac{1}{2}\|I - A\|_2/(1 - \|I - A\|_2)$ since $\|I - A\|_2 < 1$. Thus, as $\Delta t < \gamma$, $\|B_k\|_2\Delta t < 1$ and $I - \Delta t B_k$ is symmetric positive definite for $k = 0, 1, \dots, N-1$. In addition, $C_k = I - \Delta t B_k$ and iteration (3.8) has the alternate form:

$$\begin{cases} \vec{x}_{k+1} = C_k\vec{x}_k \ \text{ for } \ k = 0, 1, \dots, N-1 \\ \vec{x}_0 = \vec{c}. \end{cases} \tag{3.9}$$

For $k = 1, \vec{x}_2 = C_1C_0\vec{x}_0 = C_1C_0\vec{c}$. But $(C_1C_0)^T = C_0^TC_1^T = C_0C_1 = C_1C_0$. To see the last equality, note that $C_0C_1 = (I - \Delta t B_0)(I - \Delta t B_1) = (I - \frac{\Delta t}{2}(I - A))(I - \frac{\Delta t}{2}(I - (I - A)\Delta t)^{-1})(I - A) = (I - \frac{\Delta t}{2}(I - (I - A)\Delta t)^{-1})(I - A)(I - \frac{\Delta t}{2}(I - A))$ as $(I - A)(I - (I - A)\Delta t)^{-1} = (I - (I - A)\Delta t)^{-1}(I - A)$. Thus, C_1C_0 is symmetric. Similarly, it can be shown that $C_2C_1C_0$ is symmetric, $C_3C_2C_1C_0$ is symmetric, $\cdots$, and $C_{N-1}C_{N-2}\cdots C_1C_0$ is symmetric. As seen earlier, C_k is symmetric positive definite for each k. In addition, C_1C_0 is positive definite. To see this, notice that if $C_1C_0\vec{x} = \lambda\vec{x}$, then $C_0\vec{x} = \lambda C_1^{-1}\vec{x}$ so that $\vec{x}^TC_0\vec{x} = \lambda\vec{x}^TC_1^{-1}\vec{x}$ or that $\lambda = \vec{x}^TC_0\vec{x}/\vec{x}^TC_1^{-1}\vec{x} > 0$. As all the eigenvalues of C_1C_0 are positive, C_1C_0 is positive definite. By an inductive argument, $C_{N-1}C_{N-2}\cdots C_1C_0$ is positive definite. $\Box$

In the next section, The two numerical methods (2.31) and (3.6) are computationally compared for several matrices A. Computational experiments based on the biological or physical examples described in [1] or [2] are not performed in the present investigation.

4 Computational Comparisons

In computational experiments, five different forms for the $n \times n$ symmetric positive definite matrix A were considered. These were the following:

A_1 = tridiagonal with diagonal elements 4 and off-diagonal elements -1,

$A_2 = \frac{1}{2}B^TDB$ where $B = \begin{bmatrix} I & -I \\ I & I \end{bmatrix}$ with I the $\frac{n}{2} \times \frac{n}{2}$ identity matrix and D a diagonal matrix with elements $d_{ii} = i$ for $i = 1, 2, \dots, n$,

$A_3 =$ tridiagonal with diagonal elements 2 and off-diagonal elements -1,

$A_4 = B^T B$ where $b_{ij} = 1$ for $j \leq i$ and $b_{ij} = 0$ for $j > i$, and

$A_5 = n \times n$ Hilbert matrix.

The size n of matrix A was selected to be n=4, 8, 16, 32, and 64. The condition numbers, $\|A\|_2 \|A^{-1}\|_2$, of the above matrices for these values of size n are tabulated in Table 4.1.

Table 4.1 Condition Numbers of Matrix A For Five Values of Size n

Matrix A	n=4	n=8	n=16	n=32	n=64
A_1	2.36	2.77	2.93	2.98	3.00
A_2	4.00	8.00	16.00	32.00	64.00
A_3	9.47	32.2	116.5	440.7	1711.7
A_4	29.3	113.5	437.7	1708.7	6740.7
A_5	1.55×10^4	1.53×10^{10}	2.02×10^{22}	4.75×10^{46}	3.48×10^{95}

In all the calculations, the vector $\vec{c}$ was assigned the entries $c_i = -1$ for i odd and $c_i = 3$ for i even. (In order to check and compare computational results, a specific vector was selected for $\vec{c}$ rather than, for example, assigning $\vec{c}$ as a random vector.) Five methods were computationally compared for calculating $A^{1/2}\vec{c}$. Methods (1.3) and (1.4), that involved first calculating $A^{1/2}$ and then computing $A^{1/2}\vec{c}$, were compared with Broyden method (2.21), the Newton-Lanczos procedure (2.31), and the Runge-Kutta-Fehlberg(RKF) method for (3.1). Iterations continued in the methods until the error $\|F(\vec{x}_k)\|_2$ satisfied $\|F(\vec{x}_k)\|_2 < 10^{-5}$ with F defined in (2.1). Iterations (or steps) are defined in the RKF procedure as the number of intervals in t required to achieve the desired accuracy. The results of the computations for these five methods for all the matrices studied are given in Table 4.2. A * in the table signifies that the method failed to converge in 1000 iterations for that matrix.

Based on the calculational results obtained, which are summarized in Table 4.2, the Newton-Lanczos method (2.31) and the RKF method (3.1) were the fastest methods computationally for the matrices studied. (Recall that the computational work per iteration is proportional to n^3 for methods (1.3) and (1.4), proportional to n^2 for methods (2.21) and (2.31), and proportional to n for method (3.1).) Method (2.31) is clearly superior to method (2.21) (as well as to (2.5)) as methods (2.21) and (2.5) suffer from ill-conditioning as matrix size n increases. Methods (3.1) and (1.3) converged for all matrices studied. However, method (3.1) is faster computationally than method (1.3) for large matrices requiring only $O(n)$ operations per step.

Table 4.2 Number of Iterations to Convergence for Approximating $A^{1/2}\vec{c}$

Matrix	Size n	Method (1.3)	Method (1.4)	Method (2.21)	Method (2.31)	Method (3.1)
A_1	4	3	4	230	4	2
A_2	4	2	3	126	4	3
A_3	4	4	5	30	5	4
A_4	4	4	6	987	12	7
A_5	4	7	10	*	14	55
A_1	8	3	4	*	3	3
A_2	8	2	3	*	6	6
A_3	8	4	6	*	6	10
A_4	8	5	7	*	65	14
A_5	8	9	13	*	*	78
A_1	16	3	4	*	4	4
A_2	16	3	4	*	8	8
A_3	16	5	7	*	6	15
A_4	16	6	8	*	*	18
A_5	16	10	*	*	*	92
A_1	32	3	4	*	4	4
A_2	32	3	4	*	7	11
A_3	32	5	8	*	7	20
A_4	32	6	9	*	*	24
A_5	32	10	*	*	*	105
A_1	64	3	4	*	4	5
A_2	64	3	4	*	*	10
A_3	64	6	8	*	7	25
A_4	64	7	10	*	*	30
A_5	64	10	*	*	*	118

5 Summary

Two numerical methods for calculating $A^{1/2}\vec{c}$, given an $n \times n$ symmetric positive definite matrix A and a vector $\vec{c}$, were derived, analyzed, and computationally tested. The two methods only require either $O(n^2)$ or $O(n)$ arithmetic operations per iteration assuming that A is initially reduced to tridiagonal form. (Reduction to tridiagonal form by Householder similarity transformations requires about $\frac{4}{3}n^3$ operations but is only performed once.) The first numerical method applies a Newton iteration to a certain nonlinear system. To reduce problems associated with the solution of an ill-conditioned linear system at each iteration, a special Krylov subspace procedure is applied in this numerical method. The second numerical method approximately solves a certain initial-value problem whose solution at $t = 1$ is $A^{1/2}\vec{c}$ where $A^{1/2}$ is the positive definite square root of A. A step-control procedure based on the Runge-Kutta-Fehlberg method was applied in the present investigation to numerically solve the initial-value problem. Both the Newton-Lanczos method and the step-control method rapidly converged for a variety of matrices studied. In particular, for large matrices, the two methods appear to be computationally superior to the procedure of first calculating $A^{1/2}$ and then computing the product $A^{1/2}\vec{c}$. Future work includes development and analysis of efficient numerical methods for approximating $A^{1/m}\vec{c}$ for general matrices A.

6 Acknowledgements

Partial support for this research was provided by National Science Foundation Grants DMS-9622690 and DMS-9626417. The authors are grateful to the anonymous referee whose insightful comments resulted in an improved paper.

References

[1] E. J. Allen, Stochastic Differential Equations and Persistence Time of Two Interacting Populations, *Dynamics of Continuous, Discrete, and Impulsive Systems*, 5, 271-281 (1999).

[2] W. D. Sharp and E. J. Allen, Stochastic Neutron Transport Equations for Rod and Plane Geometries, *Annals of Nuclear Energy*, 27, 99-116 (2000).

[3] T. Gard, *Introduction to Stochastic Differential Equations*, Springer-Verlag, New York (1987).

[4] S. K. Boyd, Numerical Methods for Approximation of Square Roots of Positive Definite Matrices in Matrix-Vector Products, Thesis in Mathematics, Texas Tech University, Lubbock (1999).

[5] A. Björck and S. Hammarling, A Schur Method for the Square Root of a Matrix, *Linear Algebra and Its Applications*, 52/53, 127-140 (1983).

[6] E. D. Denman, Roots of Real Matrices, *Linear Algebra and Its Applications*, 36, 133-139 (1981).

[7] V. Druskin and L.Knizhnerman, Extended Krylov Subspaces: Approximation of the Matrix Square Root and Related Functions, *SIAM J. Matrix Anal. Appl.*, 19, 755-771 (1998).

[8] M. A. Hasan, A Power Method for Computing Square Roots of Complex Matrices, *Journal of Mathematical Analysis and Applications*, 213, 393-405 (1997).

[9] N. J. Higham, Newton's Method for the Matrix Square Root, *Mathematics of Computation*, 46, 537-549 (1986).

[10] N. J. Higham, Computing Real Square Roots of a Real Matrix, *Linear Algebra and Its Applications*, 88/89, 405-430 (1987).

[11] N. J. Higham, Stable Iterations for the Matrix Square Root, *Numerical Algorithms*, 15, 227-242 (1997).

[12] W. D. Hoskins and D. J. Walton, A Faster, More Stable Method for Computing the pth Roots of Positive Definite Matrices, *Linear Algebra and Its Applications*, 26, 139-163 (1979).

[13] S. Lakic and M. S. Petkovic, On the Matrix Square Root, *ZAMM · Z. Angew. Math. Mech.*, 78, 173-182 (1998).

[14] Y. Y. Lu, A Pade$'$ Approximation Method for Square Roots of Symmetric Positive Definite Square Matrices, *SIAM J. Matrix. Anal. Appl.*, 19, 833-845 (1998).

[15] F. Stummel and K. Hainer, *Introduction to Numerical Analysis*, Scottish Academic Press, Edinburgh (1980).

[16] Y. T. Tsay and L. S. Shieh and J. S. H. Tsai, A Fast Method for Computing Principal nth Roots of a Complex Matrix, *Linear Algebra and Its Applications*, 76, 206-221 (1986).

[17] M. Hochbruck and C. Lubich, On Krylov Subspace Approximations to the Matrix Exponential Opperator, *SIAM J. Numer. Anal.*, 34, 1911-1925 (1997).

[18] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge (1991).

[19] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge (1985).

[20] J. M. Ortega and W. C. Reinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York (1970).

[21] R. L. Burden and J. D. Faires, *Numerical Analysis*, 5th edition, PWS-Kent Publishing Company, Boston (1993).

[22] C. Lanczos, An Iterative Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators, *Journal of Research of the National Bureau of Standards*, 45, 225-282 (1950).

[23] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston (1996).

[24] J. W. Demmel, *Applied Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, Philadelphia (1997).

[25] F. Brauer and J. A. Nohel, *The Qualitative Theory of Ordinary Differential Equations*, Dover Publications, Mineola, New York (1989).

[26] D. R. Kincaid and E. W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, Brooks/Cole, Pacific Grove, California (1991).