ISSN 1360-1725

A New `sqrtm` for Matlab

N. J. Higham

Numerical Analysis Report No. 336

January 1999

A New `sqrtm` for MATLAB

Nicholas J. Higham*

January 4, 1999

Abstract

MATLAB's function `sqrtm` computes a square root of a matrix. We propose a replacement for the `sqrtm` in MATLAB 5.2 that is more accurate and returns useful information about the stability and conditioning of the problem.

Key words. matrix square root, MATLAB, Schur decomposition, condition number, stability

AMS subject classifications. 65F30

1 Introduction

MATLAB has included since at least version 3 a function `sqrtm` for computing a square root of a matrix. The function works by reducing the matrix to Schur form and then applying a recurrence of Parlett for computing a general function of a triangular matrix. An error estimate is computed and if it is too large then an attempt is made to improve the accuracy of the computed square root. The function `sqrtm` in versions of MATLAB up to version 5.2 can be much less accurate than is warranted by the condition of the problem. We propose a replacement for `sqrtm` that is more accurate and returns useful information about the stability and conditioning of the problem.

In Sections 2 and 3 we present some background theory on the existence of matrix square roots and their stability and conditioning. In Section 4 we describe the existing function `sqrtm` and then in Section 5 the proposed replacement. Numerical experiments in Section 6 compare the new and the old routines and conclusions are given in Section 7.

2 Matrix Square Roots

We begin by summarizing some salient features of the theory of matrix square roots. Further details can be found in [4], [6, Sec. 6.4]. The matrix $X \in \mathbb{C}^{n\times n}$ is a square root of $A \in \mathbb{C}^{n\times n}$ if $X^2 = A$. Any nonsingular matrix has a square root, but whether a singular matrix has a square root depends on the Jordan structure of the zero eigenvalues. If A

*Department of Mathematics, University of Manchester, Manchester, M13 9PL, England (`higham@ma.man.ac.uk`, `http://www.ma.man.ac.uk/~higham/`). This work was supported by Engineering and Physical Sciences Research Council grant GR/L76532.

is nonsingular and has s distinct eigenvalues then it has precisely 2^s square roots that are expressible as polynomials in the matrix A; if some eigenvalue appears in more than one Jordan block then there are infinitely many additional square roots, none of which is expressible as a polynomial in A. To illustrate, the singular matrix

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{2.1}$$

has no square root, a diagonal matrix $\mathrm{diag}(d_i) \in \mathbb{C}^{n\times n}$ with distinct diagonal entries has exactly 2^n square roots, $\mathrm{diag}(\pm\sqrt{d_i})$, and the $n \times n$ identity matrix has infinitely many square roots for $n > 2$, including any Householder matrix.

Although a square root is never unique when it exists, there is a distinguished square root of particular interest: the one all of whose eigenvalues lie in the right half-plane. To make this square root uniquely defined, we (arbitrarily) map any eigenvalues on the negative real axis to the positive imaginary axis. This square root is called the *principal square root* and it is a polynomial in the original matrix. When A is symmetric positive definite the principal square root is the unique symmetric positive definite square root, and when A is real and has a real square root that is a polynomial in A, the principal square root is real.

3 Stability and Conditioning of Square Roots

Two measures of the accuracy of a computed square root $\widehat{X}$ of A are the relative residual $\|A-\widehat{X}^2\|_F/\|A\|_F$ and the relative error $\|X-\widehat{X}\|_F/\|X\|_F$, where X is the square root of interest. Here, we are using the Frobenius norm, $\|A\|_F = (\sum_{i,j}|a_{ij}|^2)^{1/2}$. To know how small the relative residual can be expected to be we consider the correctly rounded exact square root: $\widetilde{X} = fl(X) = X + E$, where $\|E\|_F \le u\|X\|_F$, with u the unit roundoff. We have

$$\begin{aligned} \|A-\widetilde{X}^2\|_F &= \|XE + EX + E^2\|_2 \le 2\|X\|_F\|E\|_F + \|E\|_F^2 \\ &\le (2u+u^2)\|X\|_F^2 \\ &\le (2u+u^2)\alpha(X)\|A\|_F, \end{aligned} \tag{3.1}$$

where the stability factor

$$\alpha(X) = \frac{\|X\|_F^2}{\|A\|_F} = \frac{\|X\|_F^2}{\|X^2\|_F} \ge 1. \tag{3.2}$$

Thus the best we can expect is that the relative residual of a computed $\widehat{X}$ is of order $\alpha(\widehat{X})u$. The stability factor $\alpha(X)$ can be arbitrarily large and it is related to the condition number with respect to inversion, $\kappa(X) = \|X\|_F\|X^{-1}\|_F$, by

$$\frac{\kappa(X)}{\kappa(A)} \le \alpha(X) \le \kappa(X).$$

The relative error of $\widehat{X}$ depends on the conditioning of X, so we now derive a condition number, following [4]. If $A = X^2$ and $A + \mathit{\Delta}A = (X + \mathit{\Delta}X)^2$ then $X\mathit{\Delta}X + \mathit{\Delta}XX = \mathit{\Delta}A - \mathit{\Delta}X^2$, which can be written

$$F'(X)\mathit{\Delta}X = \mathit{\Delta}A - \mathit{\Delta}X^2,$$

where $F'(X) : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ is the Fréchet derivative of $F(X) = X^2 - A$ at X. Hence $\Delta X = F'(X)^{-1}(\Delta A - \Delta X^2)$ and taking norms and solving the resulting quadratic inequality in $\|\Delta X\|_F$ gives the sharp inequality

$$\frac{\|\Delta X\|_F}{\|X\|_F} \le \left(\|F'(X)^{-1}\|_F \frac{\|A\|_F}{\|X\|_F}\right) \frac{\|\Delta A\|_F}{\|A\|_F} + O(\|\Delta A\|_F^2). \tag{3.3}$$

This leads to the definition of the matrix square root condition number

$$\chi(X) = \|F'(X)^{-1}\|_F \frac{\|A\|_F}{\|X\|_F}.$$

For the Frobenius norm it can be shown that

$$\|F'(X)^{-1}\|_F = \|(I \otimes X + X^T \otimes I)^{-1}\|_2,$$

where $\otimes$ denotes the Kronecker product [6, Chap. 4]. Let X have the Schur decomposition $X = QRQ^*$, where Q is unitary and R is upper triangular. Then

$$I \otimes X + X^T \otimes I = (\overline{Q} \otimes Q)\,(I \otimes R + R^T \otimes I)\,(Q^T \otimes Q^*),$$

from which it follows that

$$\|(I \otimes X + X^T \otimes I)^{-1}\|_2 = \|(I \otimes R + R^T \otimes I)^{-1}\|_2.$$

The matrix $W = I \otimes R + R^T \otimes I \in \mathbb{C}^{n^2\times n^2}$ has the block lower triangular form illustrated for $n = 3$ by

$$W = \begin{bmatrix} R + r_{11}I & & \\ r_{12}I & R + r_{22}I & \\ r_{13}I & r_{23}I & R + r_{33}I \end{bmatrix}.$$

We estimate $\|W^{-1}\|_2$ by applying a few steps of the power method on $(W^*W)^{-1}$ with starting vector $[1, 1, \ldots, 1]^T$. The systems $Wx = b$ and $W^Ty = c$ can both be solved in n^3 flops by block substitution; this is of the same order as the cost of computing X by the methods described in the next two sections, but is considerably less than the $O(n^5)$ flops required to compute W^{-1} explicitly.

4 Matlab 5.2's `sqrtm`

Matlab 5.2's `sqrtm` computes the principal square root of $A \in \mathbb{C}^{n\times n}$. It first computes the Schur decomposition $A = QTQ^*$. Then it applies a recurrence of Parlett for computing $f(T)$, where f is an arbitrary function [2, Sec. 11.1.3], [7]. Parlett's recurrence is obtained by solving for $F = f(T)$ in the commutativity relation $FT = TF$, which involves dividing by $t_{ii} - t_{jj}$ for all $i \ne j$. The recurrence therefore breaks down when T has repeated diagonal elements, that is, when A has multiple eigenvalues. However, the principal square root is still well defined when there are multiple eigenvalues (assuming A has a square root). The practical upshot of using Parlett's recurrence is that `sqrtm` can produce inaccurate results in situations where the principal square root can be obtained accurately by other means.

The function sqrtm in MATLAB 5.2 is listed in Appendix A.1. It calls funm, which implements Parlett's recurrence, and which returns an error estimate, based on the reciprocal of $\min_{i\neq j}|t_{ii}-t_{jj}|$. If this error estimate exceeds a tolerance tol, set to be a multiple of the unit roundoff, sqrtm enters a phase in which it tries to improve the accuracy of the computed square root $\widehat{X}$. First, it checks to see if the relative residual $\|A-\widehat{X}^2\|_1/\|A\|_1$ exceeds the tolerance. This test is of dubious validity because, as we saw in Section 3, even the correctly rounded exact square root may have a large relative residual. If the test is failed then an orthogonal similarity transformation is applied to A, the whole computation is repeated, and the inverse transformation is performed. Then one step of Newton's method [3] is applied and the relative residual computed once more. The logic behind the similarity transformation and the Newton step is not clear, since the similarity does not change the eigenvalues and so should make little difference to the accuracy of funm, and Newton's method can increase the error because of its numerical instability [3]. ("Newton's method" here refers to a method obtained from the true Newton method by making commutativity assumptions, and these assumptions destroy the self-correcting nature of Newton's method.)

5 A New sqrtm

Our suggested replacement for sqrtm, which we refer to as sqrtm* to avoid confusion, is listed in Appendix A.2. Like sqrtm, it begins by reducing $A\in\mathbb{C}^{n\times n}$ to Schur form, $A=QTQ^*$. It then computes the principal square root R of T using a recurrence of Björck and Hammarling [1]. This recurrence is derived by considering the equation $R^2=T$. Equating (i,j) elements gives

$$t_{ij}=\sum_{k=i}^{j} r_{ik}r_{kj}, \qquad j\geq i.$$

It is easy to see that R can be computed a column at a time as follows:

```
for $j = 1\colon n$
    $r_{jj} = t_{jj}^{1/2}$
    for $i = j-1\colon -1\colon 1$
        $r_{ij} = \bigl(t_{ij} - \sum_{k=i+1}^{j-1} r_{ik}r_{kj}\bigr)/(r_{ii}+r_{jj})$
    end
end
```

The square root of A is then recovered as $X=QRQ^*$. The routine optionally computes $\alpha(X)=\alpha(R)=\|R\|_F^2/\|T\|_F$ and estimates the condition number $\chi(X)$ by the power method limited to 6 iterations, as described in Section 3.

The cost of the algorithm is $25n^3$ flops for the Schur decomposition [2, Sec. 7.5.6] plus $n^3/3$ flops for computing the square root of T, where a flop is a floating point operation. The estimation of $\chi(X)$ costs at most $12n^3$ flops, and so increases the operation count by at most 50%.

A straightforward rounding error analysis shows that the computed $\widehat{R}$ satisfies

$$|T-\widehat{R}^2|\leq\gamma_{n-1}|\widehat{R}|^2,$$

where $\gamma_k = ku/(1-ku)$. Hence

$$\|T - \widehat{R}^2\|_F \le \gamma_{n-1}\|\widehat{R}\|_F^2 \approx \gamma_{n-1}\alpha(R)\|T\|_F,$$

assuming that $\|\widehat{R}\|_F \approx \|R\|_F$. Taking account of the errors in the computation of the Schur decomposition and the transformation from $\widehat{R}$ to $\widehat{X}$, we obtain a bound of the form

$$\|A - \widehat{X}^2\|_F \le f(n)u\alpha(X)\|A\|_F, \tag{5.1}$$

where f is a cubic polynomial. This is comparable with the bound (3.1) for the correctly rounded square root and so is the best we can expect. For practical error estimation we can take $f(n) = n$, to obtain a more realistic bound.

In view of (3.3) and (5.1), the relative error can be bounded approximately by

$$\frac{\|\Delta X\|_F}{\|X\|_F} \le f(n)u\alpha(X)\chi(X). \tag{5.2}$$

Finally, we note that for real matrices an analogue of `sqrtm*` can be written that works with the real Schur decomposition, as explained in [4], so that only real arithmetic is used.

6 Numerical Experiments

We describe some numerical experiments that reveal the difference in reliability between `sqrtm` and the new routine, `sqrtm*`. We give results for three matrices. The machine precision $u = 2^{-53} \approx 1.1 \times 10^{-16}$.

The first matrix and its square root are

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ & \epsilon & 0 & 0 \\ & & \epsilon & 0 \\ & & & 1 \end{bmatrix}, \qquad X = \begin{bmatrix} 1 & 0 & 0 & 1/2 \\ & \sqrt{\epsilon} & 0 & 0 \\ & & \sqrt{\epsilon} & 0 \\ & & & 1 \end{bmatrix},$$

with $\epsilon = 2^{-24} \approx 5 \times 10^{-8}$. `Sqrtm` detects a poor initial square root and enters its improvement phase. It returns a computed square root with relative error, relative residual and `esterr` all of order 10^{-12}. `Sqrtm*` computes the exact square root. It computes the stability factor $\alpha(X) = 1.3$ and correctly obtains the condition number $\chi(X) = 4.8 \times 10^3$ after 3 power method iterations.

The second matrix is the 2×2 Jordan block (2.1), which has no square root. `Sqrtm` again invokes its improvement phase and returns a purported square root for which the relative residual and `esterr` are of order 10^{-1}. `Sqrtm*` issues a warning that the matrix is singular and may not have a square root and returns a matrix of Nans and Infs together with infinite values for α and the condition number estimate.

The third matrix is $A = (I + B)/2 \in \mathbb{R}^{4\times4}$, where B is the matrix `invol`[1] from the Test Matrix Toolbox [5]. Since B is involutary ($B^2 = I$), A is idempotent ($A^2 = A$), so A is its own principal square root. If it were formed exactly, A would have two zero eigenvalues and two eigenvalues 1, but because of rounding errors the computed matrix

[1]This matrix is also accessible as `gallery('invol',...)` in MATLAB 5.

has two eigenvalues of order 10^{-15}. `Sqrtm` makes use of its improvement phase and returns a computed square root with relative residual, relative error and `esterr` all of order 10^{-3}. `Sqrtm*` yields a relative residual of order 10^{-15} and a relative error of order 10^{-7}, and it returns $\alpha(X) = 1.6 \times 10^2$ and the (correct) condition estimate $\chi(X) = 1.31 \times 10^{10}$ after 3 power method iterations. The relative error is well within the bound (5.2).

7 Conclusions

The new routine, `sqrtm*`, has similar computational cost to `sqrtm` and produces a computed square root satisfying the best possible forms of normwise residual and forward error bounds. The experiments of Section 6 demonstrate the improved accuracy and stability over `sqrtm`, and confirm that the stability factor $\alpha(X)$ and condition number χ computed by the routine provide valuable information. `Sqrtm*` will replace `sqrtm` in version 5.3 of MATLAB (Cleve Moler, private communication).

A Appendix

A.1 Matlab's sqrtm.

```
function [S,esterr] = sqrtm(A)
%SQRTM  Matrix square root.
%   S = SQRTM(A) is the matrix square root of A.
%   Complex results are produced if A has negative eigenvalues.
%   A warning message is printed if the computed S*S is not close to A.
%
%   [S,esterr] = sqrtm(A) does not print any warning message, but
%   returns an estimate of the relative residual, norm(S*S-A)/norm(A).
%
%   If A is real, symmetric and positive definite, or complex, Hermitian
%   and positive definite, then so is the computed matrix square root.
%
%   Some matrices, like A = [0 1; 0 0], do not have any square roots,
%   real or complex, and SQRTM cannot be expected to produce one.
%
%   See also EXPM, LOGM, FUNM.

%   CBM, 7-12-92.
%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.5 $  $Date: 1997/12/05 03:31:41 $

% First try Parlett's method directly.
[S,esterr] = funm(A,'sqrt');
tol = 1000*eps;
% If funm's error estimate is small, accept the result.
if esterr >= tol
   % Use norm of residual instead of funm's crude error estimate.
   esterr = norm(S*S-A,1)/norm(A,1)
   if esterr >= tol | ~isfinite(esterr)
      [n,n] = size(A);
      I = eye(n,n);
      % Try again with a not-quite-random rotation.
      [J,K] = meshgrid(1:n);
      R = orth(I + J + K);
      [S,ignore] = funm(R*A*R','sqrt');
      S = R'*S*R;
      esterr = norm(S*S-A,1)/norm(A,1)
      if norm(imag(S),1) <= 1000*tol*norm(S,1), S = real(S); end
      % One step of improvement.
      S = (S + A/S)/2;
      esterr = norm(S*S-A,1)/norm(A,1)
   end
end
if (esterr >= tol | ~isfinite(esterr)) & (nargout < 2)
   warning(['SQRTM appears inaccurate.  esterr = ',num2str(esterr)])
end
```

A.2 The new routine: sqrtm*.

This routine has been processed by The MathWorks for inclusion in MATLAB 5.3, based on the original routine we supplied.

```
function [X, arg2, condest] = sqrtm(A)
%SQRTM     Matrix square root.
%   X = SQRTM(A) is the principal square root of the matrix A, i.e. X*X = A.
%
%   X is the unique square root for which every eigenvalue has nonnegative
%   real part.  If A has any eigenvalues with negative real parts then a
%   complex result is produced.  If A is singular then A may not have a
%   square root.  A warning is printed if exactly singularity is detected.
%
%   With two output arguments, [X, RESNORM] = SQRTM(A) does not print any
%   warning, and returns the residual, norm(A-X^2,'fro')/norm(A,'fro').
%
%   With three output arguments, [X, ALPHA, CONDEST] = SQRTM(A) returns a
%   stability factor ALPHA and an estimate CONDEST of the matrix square root
%   condition number of X.  The residual NORM(A-X^2,'fro')/NORM(A,'fro') is
%   bounded by (n+1)*ALPHA*EPS and the Frobenius norm relative error in X is
%   bounded approximately by N*ALPHA*CONDEST*EPS, where N = MAX(SIZE(A)).
%
%   See also EXPM, SQRTM, FUNM.

%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.6 $  $Date: 1998/10/15 15:15:57 $
%
%   References:
%   N. J. Higham, Computing real square roots of a real
%       matrix, Linear Algebra and Appl., 88/89 (1987), pp. 405-430.
%   A. Bjorck and S. Hammarling, A Schur method for the square root of a
%       matrix, Linear Algebra and Appl., 52/53 (1983), pp. 127-140.

n = max(size(A));
[Q, T] = schur(A);        % T is real/complex according to A.
[Q, T] = rsf2csf(Q, T);   % T is now complex Schur form.

% Compute upper triangular square root R of T, a column at a time.

warns = warning;
warning('off');

R = zeros(n);
for j=1:n
    R(j,j) = sqrt(T(j,j));
    for i=j-1:-1:1
        s = 0;
        for k=i+1:j-1
            s = s + R(i,k)*R(k,j);
        end
        R(i,j) = (T(i,j) - s)/(R(i,i) + R(j,j));
    end
end

warning(warns);

if nargout > 2
   arg2 = norm(R,'fro')^2 / norm(T,'fro');
end

X = Q*R*Q';

nzeig = any(diag(T)==0);

if nzeig & (nargout ~= 2)
   warning('Matrix is singular and may not have a square root.')
end

if nargout == 2
   arg2 = norm(X*X-A,'fro')/norm(A,'fro');
end

if nargout > 2

   if nzeig
      condest = inf;
   else

      % Power method to get condition number estimate.

      tol = 1e-2;
      x = ones(n^2,1);    % Starting vector.
      cnt = 1;
      e = 1;
      e0 = 0;
      while abs(e-e0) > tol*e & cnt <= 6
         x = x/norm(x);
         x0 = x;
         e0 = e;
         Sx = tksolve(R, x);
         x = tksolve(R, Sx, 'T');
         e = sqrt(real(x0'*x));  % sqrt of Rayleigh quotient.
         % fprintf('cnt = %2.0f, e = %9.4e\n', cnt, e)
         cnt = cnt+1;
      end

      condest = e*norm(A,'fro')/norm(X,'fro');

   end
```

```
end

%====================================

function x = tksolve(R, b, tran)
%TKSOLVE     Solves triangular Kronecker system.
%            x = TKSOLVE(R, b, TRAN) solves
%                  A*x = b  if TRAN = '',
%                 A'*x = b  if TRAN = 'T',
%            where A = KRON(EYE,R) + KRON(TRANSPOSE(R),EYE).
%            Default: TRAN = ''.

if nargin < 3, tran = ''; end

n = max(size(R));
x = zeros(n^2,1);

I = eye(n);

if isempty(tran)

   % Forward substitution.
   for i = 1:n
       temp = b(n*(i-1)+1:n*i);
       for j = 1:i-1
           temp = temp - R(j,i)*x(n*(j-1)+1:n*j);
       end
       x(n*(i-1)+1:n*i) = (R + R(i,i)*I) \ temp;
   end

elseif strcmp(tran,'T')

   % Back substitution.
   for i = n:-1:1
       temp = b(n*(i-1)+1:n*i);
       for j = i+1:n
           temp = temp - conj(R(i,j))*x(n*(j-1)+1:n*j);
       end
       x(n*(i-1)+1:n*i) = (R' + conj(R(i,i))*I) \ temp;
   end

end

return
```

References

[1] Åke Björck and Sven Hammarling. A Schur method for the square root of a matrix. *Linear Algebra and Appl.*, 52/53:127–140, 1983.

[2] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Third edition, Johns Hopkins University Press, Baltimore, MD, USA, 1996. xxvii+694 pp. ISBN 0-8018-5413-X (hardback), 0-8018-5414-8 (paperback).

[3] Nicholas J. Higham. Newton's method for the matrix square root. *Math. Comp.*, 46 (174):537–549, April 1986.

[4] Nicholas J. Higham. Computing real square roots of a real matrix. *Linear Algebra and Appl.*, 88/89:405–430, 1987.

[5] Nicholas J. Higham. The Test Matrix Toolbox for MATLAB (version 3.0). Numerical Analysis Report No. 276, Manchester Centre for Computational Mathematics, Manchester, England, September 1995. 70 pp.

[6] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991. viii+607 pp. ISBN 0-521-30587-X.

[7] B. N. Parlett. A recurrence among the elements of functions of triangular matrices. *Linear Algebra and Appl.*, 14:117–121, 1976.