Blocked Algorithms for the Matrix Square Root

Edvin Deadman \(^1\) Nick Higham \(^2\) Rui Ralha \(^3\)

\(^1\)University of Manchester / Numerical Algorithms Group
\(^2\)University of Manchester
\(^3\)University of Minho, Portugal

May 18, 2012
Outline

1. Numerical Libraries and NAG
2. The Matrix Square Root:
   - Definition and Uses
   - The Schur Method
   - Blocking
   - Parallelism
Why use numerical libraries?

- Numerical computation is difficult to do accurately.
- Writing routines from scratch is time consuming.
- Commonly encountered problems:
  - Overflow/underflow: how does the computer deal with large / small numbers?
  - Condition: how sensitive is the solution to small changes in the input?
  - Stability: how sensitive is the computation to rounding errors?
- Important considerations:
  - Error analysis.
  - Information about error bounds.
  - Parallelism.
How a numerical library is used
The NAG Library

- Root Finding
- Summation of Series
- Quadrature
- Ordinary Differential Equations
- Partial Differential Equations
- Numerical Differentiation
- Integral Equations
- Mesh Generation
- Interpolation
- Curve and Surface Fitting
- Optimization
- Approximations of Special Functions
- Dense Linear Algebra
- Sparse Linear Algebra
- Correlation & Regression Analysis
- Multivariate Methods
- Analysis of Variance
- Random Number Generators
- Univariate Estimation
- Nonparametric Statistics
- Smoothing in Statistics
- Contingency Table Analysis
- Survival Analysis
- Time Series Analysis
- Operations Research
A square root of $A$ is a solution of $X^2 = A$. 
A square root of $A$ is a solution of $X^2 = A$.

There can be infinitely many square roots. They are never unique:

\[
\begin{pmatrix}
\cos \theta & \sin \theta \\
\sin \theta & -\cos \theta
\end{pmatrix}^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.
\]
Defining the Matrix Square Root

- A square root of $A$ is a solution of $X^2 = A$.
- There can be infinitely many square roots. They are never unique:
  \[
  \begin{pmatrix}
  \cos \theta & \sin \theta \\
  \sin \theta & -\cos \theta
  \end{pmatrix}^2 = \begin{pmatrix}
  1 & 0 \\
  0 & 1
  \end{pmatrix}.
  \]
- If $A$ has $m$ Jordan blocks and $s \leq m$ distinct eigenvalues then there are $2^s$ square roots that are primary functions of $A$. 
Defining the Matrix Square Root

- A square root of \( A \) is a solution of \( X^2 = A \).
- There can be infinitely many square roots. They are never unique:
  \[
  \begin{pmatrix}
  \cos \theta & \sin \theta \\
  \sin \theta & -\cos \theta
  \end{pmatrix}^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.
  \]
- If \( A \) has \( m \) Jordan blocks and \( s \leq m \) distinct eigenvalues then there are \( 2^s \) square roots that are primary functions of \( A \).
- If \( A \) has no eigenvalues on the negative real line, then there is a unique *principal square root* \( A^{1/2} \) with eigenvalues in the right half plane.
Defining the Matrix Square Root: Example

\[ A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \]
Defining the Matrix Square Root: Example

\[ A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \]

Four primary square roots:

\[ \begin{pmatrix} 1 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{2} & \frac{1}{2\sqrt{2}} \\ 0 & 0 & 0 & \sqrt{2} \end{pmatrix}, \begin{pmatrix} -1 & -\frac{1}{2} & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -\sqrt{2} & -\frac{1}{2\sqrt{2}} \\ 0 & 0 & 0 & -\sqrt{2} \end{pmatrix}, \begin{pmatrix} 1 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\sqrt{2} & -\frac{1}{2\sqrt{2}} \\ 0 & 0 & 0 & -\sqrt{2} \end{pmatrix}, \begin{pmatrix} -1 & -\frac{1}{2} & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & \sqrt{2} & \frac{1}{2\sqrt{2}} \\ 0 & 0 & 0 & \sqrt{2} \end{pmatrix}. \]
Defining the Matrix Square Root: Example

\[ A = \begin{pmatrix}
1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 2 & 1 \\
0 & 0 & 0 & 2
\end{pmatrix} \]

Four primary square roots:

\[ \begin{pmatrix}
1 & \frac{1}{2} & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \sqrt{2} & \frac{1}{2\sqrt{2}} \\
0 & 0 & 0 & \sqrt{2}
\end{pmatrix}, \quad \begin{pmatrix}
-1 & -\frac{1}{2} & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & -\sqrt{2} & -\frac{1}{2\sqrt{2}} \\
0 & 0 & 0 & -\sqrt{2}
\end{pmatrix} \]

\[ \begin{pmatrix}
1 & \frac{1}{2} & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -\sqrt{2} & -\frac{1}{2\sqrt{2}} \\
0 & 0 & 0 & -\sqrt{2}
\end{pmatrix}, \quad \begin{pmatrix}
-1 & -\frac{1}{2} & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & \sqrt{2} & \frac{1}{2\sqrt{2}} \\
0 & 0 & 0 & \sqrt{2}
\end{pmatrix} \]
Uses of the Matrix Square Root

- Markov models of finance and population dynamics:
  - If $P(t)$ is the transition matrix for a time step $t$, then $\sqrt{P}$ could be used for $\frac{t}{2}$.
Markov models of finance and population dynamics:
- If $P(t)$ is the transition matrix for a time step $t$, then $\sqrt{P}$ could be used for $\frac{t}{2}$.

Solution of differential equations:
- $\ddot{y} + Ay = 0$ has solution $y = \cos(\sqrt{A}t)y_0 + \sin(\sqrt{A}t)y_1$. 
Uses of the Matrix Square Root

- Markov models of finance and population dynamics:
  - If $P(t)$ is the transition matrix for a time step $t$, then $\sqrt{P}$ could be used for $\frac{t}{2}$.
- Solution of differential equations:
  - $\ddot{y} + Ay = 0$ has solution $y = \cos(\sqrt{A}t)y_0 + \sin(\sqrt{A}t)y_1$.
- Polar decomposition / matrix sign decomposition.
Markov models of finance and population dynamics:
- If $P(t)$ is the transition matrix for a time step $t$, then $\sqrt{P}$ could be used for $\frac{t}{2}$.

Solution of differential equations:
- $\ddot{y} + Ay = 0$ has solution $y = \cos(\sqrt{A}t)y_0 + \sin(\sqrt{A}t)y_1$.

Polar decomposition / matrix sign decomposition.

Important kernel routine for computing matrix logarithm, $p$th roots and powers and trigonometric matrix functions.
The Schur Method

1. Compute a Schur decomposition: \( A = QTQ^* \) with \( T \) upper triangular.

2. Expand \( U^2 = T \) elementwise. For a primary square root, \( U \) is also upper triangular:

\[
U^2_{ii} = T_{ii},
\]

\[
U_{ii}U_{ij} + U_{ij}U_{jj} = T_{ij} - \sum_{k=i+1}^{j-1} U_{ik}U_{kj}.
\]

\( U \) is found either a column or a superdiagonal at a time.

3. \( \sqrt{A} = QUQ^* \)
The Schur Method: Properties

- Cost: \(28\frac{1}{3} n^3\) flops.
- Real arithmetic version uses quasi upper triangular Schur decomposition and \(2 \times 2\) diagonal block structure.
- Computed square root of the triangular matrix \(\hat{U}\) satisfies \(\hat{U}^2 = T + \Delta T\), where

\[
|\Delta T| \leq \tilde{\gamma}_n |\hat{U}|^2
\]

(this only holds normwise in the real arithmetic version).
The Schur Method: Properties

- Cost: $28 \frac{1}{3} n^3$ flops.
- Real arithmetic version uses quasi upper triangular Schur decomposition and $2 \times 2$ diagonal block structure.
- Computed square root of the triangular matrix $\hat{U}$ satisfies $\hat{U}^2 = T + \Delta T$, where

  $$|\Delta T| \leq \tilde{\gamma}_n |\hat{U}|^2$$

  (this only holds normwise in the real arithmetic version).
- No use of level 3 BLAS - slow!
- Now focus on the triangular phase of the algorithm.
Run times for random complex triangular matrices

Edvin Deadman, Nick Higham, Rui Ralha, University of Manchester / Numerical Algorithms Group, University of Manchester, University of Minho, Portugal

Blocked Algorithms for the Matrix Square Root
The Blocked Schur Method

- The $U_{ij}$ and $T_{ij}$ are now taken to be blocks:

$$U_{ii}^2 = T_{ii}, \quad (1)$$

$$U_{ii}U_{ij} + U_{ij}U_{jj} = T_{ij} - \sum_{k=i+1}^{j-1} U_{ik}U_{kj}. \quad (2)$$

- Solve (1) using the point method and (2) by solving the Sylvester equation (e.g. xTRSYL in LAPACK).
The Blocked Schur Method

- The $U_{ij}$ and $T_{ij}$ are now taken to be blocks:

  $$U_{ii}^2 = T_{ii}, \quad (1)$$

  $$U_{ii}U_{ij} + U_{ij}U_{jj} = T_{ij} - \sum_{k=i+1}^{j-1} U_{ik}U_{kj}. \quad (2)$$

- Solve (1) using the point method and (2) by solving the Sylvester equation (e.g. xTRSYL in LAPACK).

- Same error bounds hold true.

- Over 90% of run time spent in GEMM calls and 8% in Sylvester equation solution.

- Not very sensitive to block size or choice of kernel routines.
Run times for random complex triangular matrices

Edvin Deadman, Nick Higham, Rui Ralha, University of Manchester / Numerical Algorithms Group, University of Manchester, University of Minho, Portugal
Blocked Algorithms for the Matrix Square Root
Recursive solution of the triangular phase:

\[
\begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}^2 = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}.
\]

\( U_{11}^2 = T_{11} \) and \( U_{22}^2 = T_{22} \) are solved recursively.

Solve \( U_{11} U_{12} + U_{12} U_{22} = T_{12} \) using the recursive method of Jonsson & Kågström.

‘Point’ algorithms are used when the recursion has reached a certain size (e.g. \( n = 64 \)).

Same error bound holds as point algorithm (proof by induction).
Run times for random complex triangular matrices
Aside: multiplying two triangular matrices
Full complex matrices (called from MATLAB)
Full real matrices (called from MATLAB)
Parallelism

Which approach is best in parallel?
Parallelism

Which approach is best in parallel?

Options for parallelism:

- Threaded BLAS.
- Explicit loop-based parallelism of the triangular phase.
- OpenMP tasks.
Parallel Point and Block Methods

Blocks below and left of \((i, j)\) block must be computed first:
Synchronisation required after each superdiagonal:
Synchronisation required after each superdiagonal:
Parallel Point and Block Methods

Synchronisation required after each superdiagonal:
Synchronisation required after each superdiagonal:
Recursive Blocking in Parallel

Task based approach - each recursive call generates new tasks. Synchronization points required:
Recursive Blocking in Parallel

Task based approach - each recursive call generates new tasks. Synchronization points required:
Recursive Blocking in Parallel

Task based approach - each recursive call generates new tasks. Synchronization points required:
Recursive Blocking in Parallel

Task based approach - each recursive call generates new tasks. Synchronization points required:
Recursive Blocking in Parallel

Task based approach - each recursive call generates new tasks. Synchronization points required:
Parallel Results for $4000 \times 4000$ Triangular Matrices

Edvin Deadman, Nick Higham, Rui Ralha, University of Manchester / Numerical Algorithms Group, University of Manchester, University of Minho, Portugal

Blocked Algorithms for the Matrix Square Root
Parallel Results for $4000 \times 4000$ Full Matrices
Implementation for the NAG Library

- Key: robustness and error handling.
Key: robustness and error handling.

Extension to singular matrices:
- Eigenvalue considered to vanish if $\lambda < \epsilon \|A\|$.
- Reorder and check if the vanishing eigenvalue is semisimple.

- Negative eigenvalues: Eigenvalue considered to lie on $\mathbb{R}^-$ if $\text{Re}(\lambda) < 0$ and $|\text{Im}(\lambda)| < \epsilon |\text{Re}(\lambda)|$. A non-principal square root can be returned in this case.

Condition estimation: Condition number estimates and residual bounds are available for the matrix square root.
Condition number is expensive to compute.
Implementation for the NAG Library

- Key: robustness and error handling.
- Extension to singular matrices:
  - Eigenvalue considered to vanish if $\lambda < \epsilon \|A\|
  - Reorder and check if the vanishing eigenvalue is semisimple.
- Negative eigenvalues:
  - Eigenvalue considered to lie on $\mathbb{R}^-$ if $Re(\lambda) < 0$ and $|Im(\lambda)| < \epsilon |Re(\lambda)|$
  - A non-principal square root can be returned in this case.
Key: robustness and error handling.

Extension to singular matrices:
- Eigenvalue considered to vanish if $\lambda < \epsilon \|A\|$
- Reorder and check if the vanishing eigenvalue is semisimple.

Negative eigenvalues:
- Eigenvalue considered to lie on $\mathbb{R}^-$ if $Re(\lambda) < 0$ and $|Im(\lambda)| < \epsilon |Re(\lambda)|$.
- A non-principal square root can be returned in this case.

Condition estimation:
- Condition number estimates and residual bounds are available for the matrix square root.
- Condition number is expensive to compute.
Real and complex routines, with or without condition estimation, and extra routines specifically for triangular matrices.

Test programs:
- Check the residual \( (\sqrt{A})^2 - A \) against the theoretical residual bound.
- Test against computation in VPA using condition estimate.
- Tricky test cases e.g. almost singular matrices, nearly negative eigenvalues.
- Error exits and illegal inputs.
Recursive blocking can be fast, when the algorithm allows.
NAG implementation uses OpenMP, so will use standard blocking.
Matrix square root is a key computational kernel - BLAS?
The fastest method in serial is not necessarily the fastest method in parallel.
References


