Results Matter. Trust NAG.

Matrix Functions and the NAG Library

The University of Manchester

1. Introduction

Matrix functions are a generalization of scalar functions to matrices.

- Examples: $\exp(A)$, $\sin(A)$, $\sqrt{A}$, where A is square.
- Applications: Markov models in finance, differential equations, networks, computer animation software.

The Numerical Algorithms Group (NAG) has collaborated with the University of Manchester on matrix function algorithms.

This poster discusses the techniques developed at the University of Manchester to compute functions of matrices and the challenges encountered in implementing these algorithms in the NAG Library.

2. Taylor Series and Padé Approximants

Taylor series provide a natural way of evaluating matrix functions:

$$\exp(A) = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots$$

For a general function f, convergence is guaranteed provided the eigenvalues lie within the radius of convergence of the scalar Taylor series.

We can also use Padé approximants, the ratio of two rational functions:

- may produce approximations of a given accuracy at a lower cost.
- can be evaluated by explicity computing the rational functions, or by using continued fraction or partial fraction representations.

3. Scaling and Squaring

For $\exp(A)$, Taylor series or Padé approximants are more reliable when $\|A\|$ is small. For $\log(A)$ or A^p ($p \in \mathbb{R}$), they are more reliable when $\|A - I\|$ is small. We can exploit the identities

$$\exp(A) = \exp(A/2^s)^{2^s},$$
$$\log(A) = 2^s \log(A^{1/2^s}),$$
$$A^p = (A^{1/2^s})^{2^s p}, \quad s \in \mathbb{N},$$

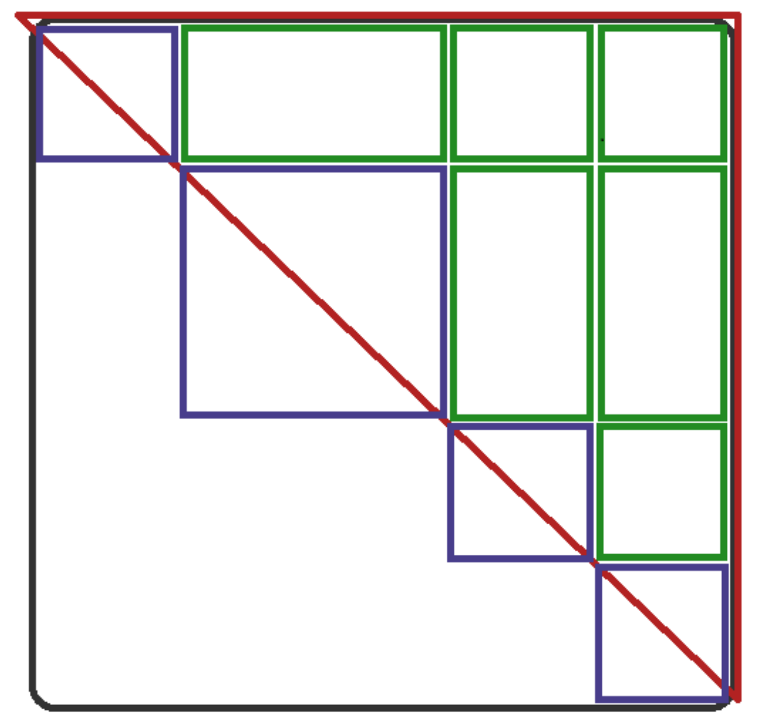
to decrease the relevant norms.

Scaling and squaring can produce accurate, efficient algorithms.

Aim: choose s and the Taylor or Padé degree m to compute $f(A)$ with backward error bounded by unit roundoff with minimal cost. Sharper bounds can be obtained using $\|A^t\|^{1/t}$, $t \in \mathbb{N}$. This prevents overscaling.

4. Parallelizing the Schur–Parlett Algorithm

The Schur–Parlett algorithm evaluates a general matrix function $f(A)$ using Taylor series on various specially chosen submatrices.


Stages of the Schur–Parlett algorithm

1. Compute a Schur decomposition $A = QTQ^*$, where $A \in \mathbb{C}^{n\times n}$, T is triangular and Q is unitary (this reduces the cost of subsequent matrix multiplications).
2. Reorder Schur form, grouping similar eigenvalues together in diagonal blocks. Use Taylor series for the diagonal blocks (closely grouped eigenvalues improves convergence).
3. Compute off-diagonal blocks by repeated solution of Sylvester equations.
4. Multiply by Schur vectors: $f(A) = Qf(T)Q^*$.

- The blocks in each superdiagonal can be computed in parallel.
- The performance gain depends on the eigenvalue distribution of A, since this determines the diagonal block structure.

Eigenvalue distribution	**Speedup 1 core → 8 cores**
One block of size 1000	1 (no speedup)
20 blocks of size 50	5
1000 blocks of size 1	5

The performance gain when moving from serial to parallel can be highly dependent on the input data.

5. Testing and Error Analysis

For many algorithms, error bounds are not available, owing to the algorithm's complexity. How do we know we're getting the 'right' answer?

(a) **High Precision**

- Matrix function can be computed in higher precision arithmetic.
- Condition number κ (a measure of the sensitivity of the problem to changes in input) can also be computed.
- Normwise relative error in double precision estimate should be at most of the order of $u \times \kappa$, where u is the machine unit roundoff.

(b) **Matrix Function Identities**

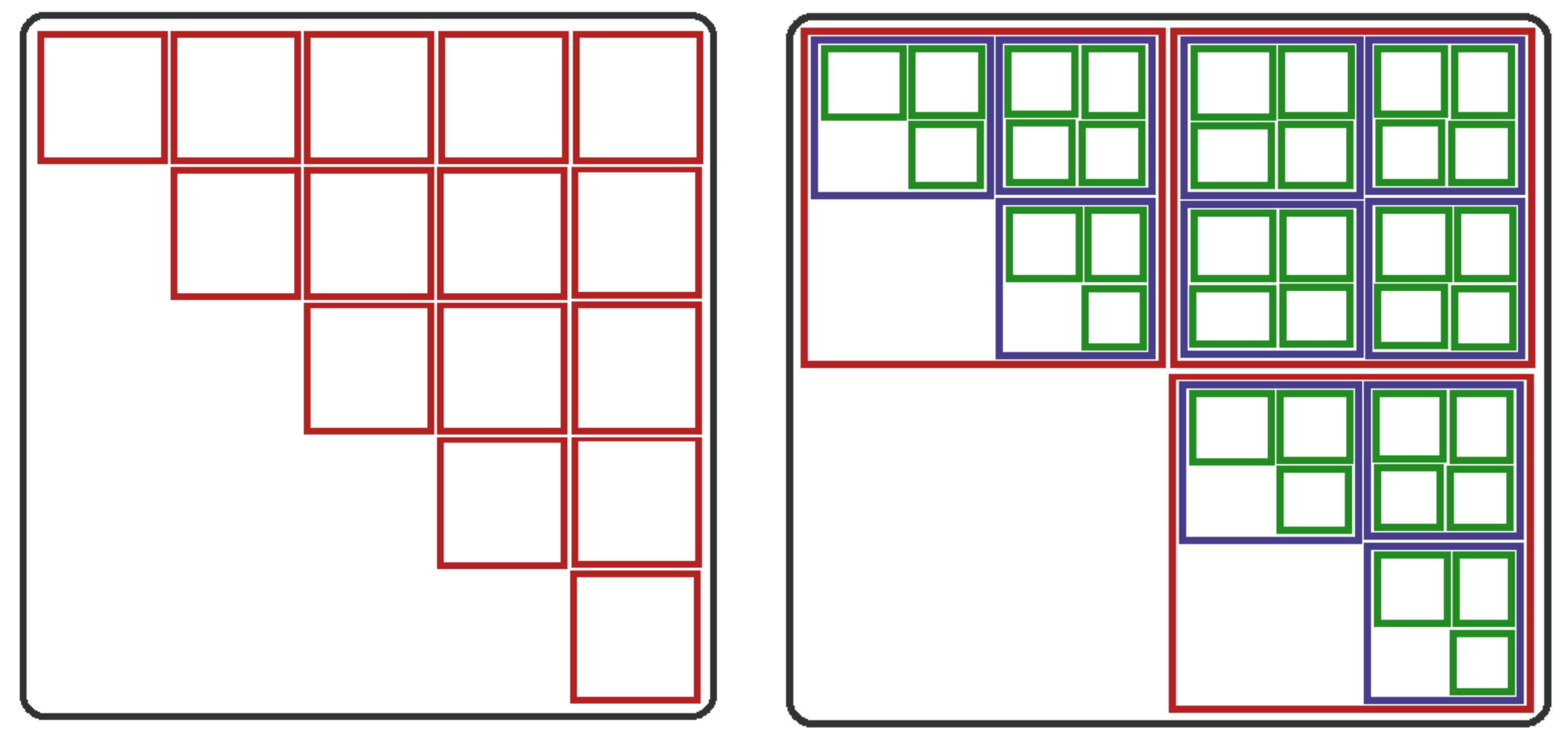
Certain scalar identities apply to matrices, e.g.

$$\exp(\log(A)) = A,$$
$$\sin^2 A + \cos^2 A = I.$$

We can bound the residuals consistent with stability using Fréchet derivatives.

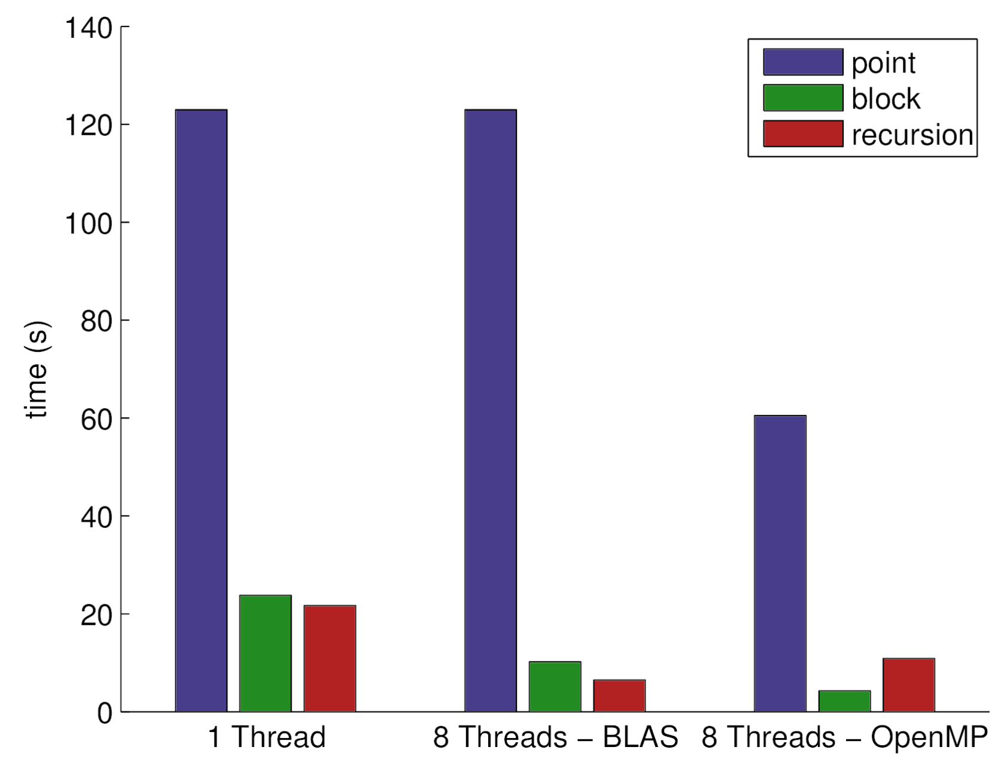
6. Parallelizing the Matrix Square Root

The square root of a matrix can be found by solving recursively on the upper triangular Schur form. Two blocking strategies are available, 'standard blocking' and 'recursive blocking'.


Standard and recursive blocking schemes for the matrix square root

The graph shows three sets of timings for the blocking schemes:

- serial implementation,
- using threaded BLAS,
- explicitly parallelized using OpenMP.



The fastest serial algorithm may not be the fastest in parallel.

7. Summary

- NAG is implementing new matrix function algorithms developed at the University of Manchester. New routines have appeared from Mark 23 of the NAG Library onwards.
- Parallelism needs to be built into new algorithms from the outset.
- Performance gain from parallelism is dependent on the input data.
- The best algorithm in serial may not be the best algorithm in parallel.

References

To find out more, see the following papers, and references therein:

N. Higham. *Functions of Matrices: Theory and Computation.* SIAM, Philadelphia, PA, USA, 2008.
A. H. Al-Mohy, N. J. Higham and S. Relton. Computing the Fréchet derivative of the matrix logarithm and estimating the condition number. SIAM J. Sci. Comput., 35(4), C394-C410
N. J. Higham and L. Lin. An improved Schur–Padé algorithm for fractional powers of a matrix and their Fréchet derivatives. SIAM. J. Matrix Anal. & Appl., 34(3), 1341-1360
E. Deadman, N. J. Higham and R. Ralha. Blocked Schur algorithms for computing the matrix square root. Proc of PARA2012, Springer–Verlag
E. Deadman and N.J. Higham. Testing matrix function algorithms using identities. ACM Trans. Math. Softw. 42(1) 2016.

Edvin Hopkins edvin.hopkins@nag.co.uk