A Catalogue of Software for Matrix Functions. Version 2.0

Higham, Nicholas J. and Deadman, Edvin

2016

MIMS EPrint: **2016.3**

ISSN 1749-9097

A Catalogue of Software for Matrix Functions. Version 2.0

Nicholas J. Higham* Edvin Deadman†

January 25, 2016; updated March 12, 2016.

Abstract

A catalogue of software for computing matrix functions and their Fréchet derivatives is presented. For a wide variety of languages and for software ranging from commercial products to open source packages we describe what matrix function codes are available and which algorithms they implement.

Contents

*School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (nick.higham@manchester.ac.uk, http://www.maths.manchester.ac.uk/~higham).

†Numerical Algorithms Group Ltd., Peter House, Oxford Street, Manchester, M1 5AN (edvin.deadman@nag.co.uk, http://www.maths.manchester.ac.uk/~edeadman).

1 Introduction

The earliest widely available software for computing functions of matrices is probably the function named `fun` in the original 1984 Fortran version of MATLAB:

```
      < M A T L A B >
    Version of 01/10/84

<>
help fun

FUN For matrix arguments  X , the  functions  SIN,  COS,  ATAN,
    SQRT,  LOG,  EXP and X**p are computed using eigenvalues  D
    and eigenvectors  V .  If  <V,D> =  EIG(X)   then   f(X)  =
    V*f(D)/V  .   This method may give inaccurate results if  V
    is badly conditioned.  Some idea of  the  accuracy  can  be
    obtained by comparing  X**1  with  X .
    For vector arguments,  the  function  is  applied  to  each
    component.
```

Since then, and especially in the last decade or so, the quantity of software for matrix functions has grown tremendously—to such an extent that it is hard to keep track of what is available. This document is an attempt to produce a catalogue of software for matrix functions available in different languages and packages.

The document lists what is available with a brief description of and reference to the algorithms that are used (where known). We make no attempt to judge the quality of the software. We also do not specify version numbers; for software still under development we are referring to the version current at the time of writing. This document is not intended to be exhaustive. For example, if a code or package has been superseded or is a translation of an existing code to another language we will usually omit it.

This catalogue is a revised an updated version of [51]. We welcome notification of errors and omissions, which will be incorporated into future versions.

For background on functions of matrices see [47], [49], or [55].

2 Applications of Matrix Functions

Matrix functions have applications in a diverse and growing range of areas of science, engineering, and the social sciences. We list a selection of areas in which we are aware of the use of matrix function software.

- Multizone models of pollutant transport in buildings take the form of linear systems of ordinary differential equations, which can be effectively solved using the matrix exponential [72].
- In Markov models in finance, statistics and social science [47, Sec. 2.3] transition probability matrices are related to the transition intensity matrix via the matrix exponential. Transition matrices for shorter time scales can be generated by taking matrix roots, but there are open questions about the existence and uniqueness of stochastic roots [52], [59].

- NMR spectroscopy involves evaluating the exponential of a symmetric diagonally dominant relaxation matrix [43], [68]. The package SIMPSON (`http://nmr.au.dk/software/simpson`) for numerical simulations of NMR experiments includes several methods for evaluating the matrix exponential.
- In control theory, linear dynamical systems can be expressed as continuous-time systems or as discrete-time state-space systems. The matrix exponential and logarithm can be used to convert between the two forms [47, Sec. 2.4]. In the Control System Toolbox for MATLAB, functions `c2d` and `d2c` carry out these conversions.
- In nuclear engineering the burnup equations are a first-order system of linear ordinary differential equations that are usually solved by time-stepping with the matrix exponential [74]. The Python Nuclear Engineering Toolkit (`http://pynesim.org`) uses the SciPy function `linalg.expm` (see Section 19.1).
- In social and information networks the elements of either the exponential or the resolvent of the adjacency matrix of the network can be used to quantify the importance of nodes within the network [30]. Recent research and software development has focused on computing these elements, including in cases with special structure; see [15] and the references therein. In time-varying networks the matrix logarithm is required [39].
- A number of problems in imaging make use of the matrix logarithm, including image registration [12], patch modeling-based skin detection [57], and in-betweening in computer animations [75].
- In optics, the Mueller matrix M is a real 4×4 matrix associated with an element that alters the polarization of light. One method for determining the diattenutation, retardance, and depolarization properties of M involves computing a pth root with $p \approx 10^5$ [22], [70]. The logarithm of M also provides understanding of the underlying medium that M describes [71]. A related Jones matrix can be represented in terms of the matrix exponential [13].

3 Matrix Function Algorithms

There is now a large literature on matrix function algorithms, of which a survey as of 2010 is given in [50]. It may not be clear to users from different fields which algorithms represent the current state-of-the-art. We list the algorithms that we consider to be preferred for a few common matrix functions, for the case where a factorization of A can be explicitly computed and full precision is required.

- Exponential: scaling and squaring algorithm (Al-Mohy and Higham, 2009) [4].
- Logarithm: inverse scaling and squaring algorithm (Al-Mohy, Higham, and Relton, 2012, 2013) [7], [8].
- Square root: Schur algorithm (Björck and Hammarling, 1983) [19], or real version for real matrices (Higham, 1987) [45]. An algorithm with blocking provides performance improvements (Deadman, Higham, and Ralha, 2013) [26].
- Real matrix power A^t with $t \in \mathbb{R}$: Schur–Padé algorithm (Higham and Lin, 2013) [54].
- Cosine and sine (Al-Mohy, Higham, and Relton, 2015) [9].
- Inverse cosine, inverse sine, inverse hyperbolic cosine, and inverse hyperbolic sine ($\operatorname{acos} A$, $\operatorname{acos} A$, $\operatorname{acosh} A$, $\operatorname{asinh} A$) (Aprahamian and Higham, 2016) [11].

Table 1: Availability of recommended algorithms.

	MATLAB built-in Sec. 4	MATLAB Third party Secs 7, 8	NAG Library Sec. 17	SciPy Sec. 19.1	Julia Sec. 21	R Sec. 15
e^A [4]	√	√	√	√	×	√
$\log A$ [7], [8]	√	√	√	√	√	×
$\cos A$, $\sin A$ [9]	×	√	×	×	×	×
acos(h)(A), asin(h)(A) [11]	×	√	×	×	×	×
$A^{1/2}$ [19], [26], [45]	√	√	√	√	√	√
A^t [54]	×	√	√	√	×	×
$f(A)$ [23]	√	–	√	×	×	×
Estimation of cond(f, A)	×	√	√	×	×	×
$e^A b$ [6][a]	×	√	√	√	×	×
$L_{\exp}$ [3]	×	√	√	√	×	√
$L_{\log}$ [8]	×	√	√	×	×	×
L_{x^t} [54]	×	√	√	×	×	×

[a]Krylov methods are also available; see the following sections.

- General matrix function with derivatives of the underlying scalar function available: Schur–Parlett algorithm (Davies and Higham, 2003) [23]. This uses the recurrence of Parlett (1976) [73].

- Function of a symmetric or Hermitian matrix: diagonalization (spectral decomposition).

In many applications of matrix functions the matrix is not known exactly, due to data errors or errors in previous computations. Even with exact data the computation of a matrix function is subject to rounding errors. It is therefore important to understand the sensitivity of the matrix function to perturbations in the data, which is determined by the Fréchet derivative, denoted by L_f. The recommended algorithms for computing the Fréchet derivative are as follows.

- Exponential: scaling and squaring algorithm (Al-Mohy and Higham, 2009) [3].

- Logarithm: inverse scaling and squaring algorithm (Al-Mohy, Higham, and Relton, 2013) [8].

- Real matrix power A^t with $t \in \mathbb{R}$: Schur–Padé algorithm (Higham and Lin, 2013) [54].

- General matrix function: complex step algorithm (Al-Mohy and Higham, 2010) [5] or use of a block 2×2 matrix formula [50, Sec. 7.3].

Table 1 summarizes the availability of the above algorithms in several key sources of software. Details are provided in the following sections.

The worst-case sensitivity of a matrix function over all perturbations is measured by the condition number, cond(f, A) [47, Chap. 3]. The recommended way to estimate the condition number is by using one of the above algorithms for the Fréchet derivative in conjunction with [47, Alg. 3.22] and the block matrix 1-norm estimator of [56]; an implementation is the function `funm_condest1` in the Matrix Function Toolbox (see Section 7). We encourage users to compute a condition number estimate whenever possible.

A rather different problem is to compute $f(A)b$, where b is a vector—the action of $f(A)$ on a vector—without explicitly forming $f(A)$. Such problems can involve very large, sparse matrices, in which case matrix factorization may not be possible and methods that require only matrix–vector products with A are needed. The sensitivity of this problem is investigated by Deadman [24].

Codes for the $f(A)b$ problem are described in some of the following sections.

4 MATLAB Built-in Functions

MATLAB has a number of built-in commands for evaluating functions of matrices.

- `expm`: matrix exponential by scaling and squaring algorithm (Al-Mohy and Higham, 2009) [4]. MATLAB R2006a–R2015a used the earlier algorithm of (Higham, 2005, 2009) [46], [48], which could suffer from overscaling [65].

  Also included for pedagogical and historical interest are three older algorithms.

  - `expmdemo1`: matrix exponential by an older scaling and squaring algorithm [36, Alg. 9.3.1]. This is an M-file implementation of the algorithm that was used by `expm` in MATLAB 7 (R14SP3) and earlier versions.
  - `expmdemo2`: matrix exponential by Taylor series.
  - `expmdemo3`: matrix exponential by eigenvalue decomposition.

- `logm`: matrix logarithm by inverse scaling and squaring algorithm (Al-Mohy, Higham, and Relton, 2012, 2013) [7], [8]. MATLAB R2008a–R2015a used the Schur–Parlett algorithm combined with inverse scaling and squaring (Higham, 2008) [47, Alg. 11.11].

- `mpower`, `^`: arbitrary matrix power via eigendecomposition. Not that this approach can be numerically unstable for noninteger powers of highly nonnormal matrices.

- `sqrtm`: matrix square root by Schur method with recursive blocking (Björck and Hammarling, 1983) [19], (Deadman, Higham, and Ralha, 2013) [26] and with condition number estimate.

- `funm`: Schur–Parlett algorithm for general functions (Davies and Higham, 2003) [23]. It has built-in support for the matrix cosine, sine, hyperbolic cosine, and hyperbolic sine.

- `polyvalm`: evaluate polynomial with matrix argument.

5 Symbolic Math Toolbox

The Symbolic Math Toolbox [64] is a MATLAB toolbox that carries out computations with symbolic variables and also provides variable precision arithmetic. The toolbox overloads the following functions for both symbolic and variable precision matrix arguments. For variable precision arguments the function `digits` can be used to specify the number of digits of precision required.

- `expm`: matrix exponential.

- `logm`: matrix logarithm.

- `sqrtm`: matrix square root.

- `mpower`, `^`: arbitrary matrix power via eigendecomposition.

- `funm`: general matrix function.

The Symbolic Math Toolbox also contains the MuPAD computer algebra system, which provides some additional matrix function capabilities for matrices with numeric (not symbolic) entries.

- `numeric::expMatrix`: computes the matrix exponential or the action of the matrix exponential on another matrix or vector. The numerical precision used can be specified by the environment variable `DIGITS`. The exponential is evaluated using a choice of diagonalization, interpolation, a Taylor series (apparently without scaling and squaring), or (for $e^{A}b$ only) a Krylov subspace method.

- `numeric::fMatrix`: for a diagonalizable matrix, computes an arbitrary function of the matrix via a diagonalization.

6 The Advanpix Multiprecision Computing Toolbox

The Advanpix Multiprecision Toolbox [1] is an extension to MATLAB for computing with arbitrary precision. The toolbox provides arbitrary precision analogues to the built-in MATLAB matrix functions as well as some trigonometric matrix functions. It is specifically optimized for quadruple precision.

The following matrix function routines are available in the toolbox: `funm`, `expm`, `sqrtm`, `logm`, `sinm`, `cosm`, `sinhm`, and `coshm`. The first four have the same calling sequences as their MATLAB counterparts.

7 The Matrix Function Toolbox

The Matrix Function Toolbox (Higham, 2008) [44] contains MATLAB implementations of many of the algorithms described in the book *Functions of Matrices: Theory and Computation* [47], including

- trigonometric matrix functions,
- condition number evaluation and estimation,
- Fréchet derivative evaluation,
- polar decomposition,
- iterative methods for computing matrix roots,
- $f(A)b$ via Arnoldi method.

The toolbox is documented in [47, App. D] and its contents are summarized in Table 2.

Table 2: Contents of Matrix Function Toolbox.

`arnoldi`	Arnoldi iteration
`ascent_seq`	Ascent sequence for square (singular) matrix.
`cosm`	Matrix cosine by double angle algorithm.
`cosm_pade`	Evaluate Padé approximation to the matrix cosine.
`cosmsinm`	Matrix cosine and sine by double angle algorithm.

Table 2: (continued)

cosmsinm_pade	Evaluate Padé approximations to matrix cosine and sine.
expm_cond	Relative condition number of matrix exponential.
expm_frechet_pade	Fréchet derivative of matrix exponential via Padé approximation.
expm_frechet_quad	Fréchet derivative of matrix exponential via quadrature.
fab_arnoldi	$f(A)b$ approximated by Arnoldi method.
funm_condest1	Estimate of 1-norm condition number of matrix function.
funm_condest_fro	Estimate of Frobenius norm condition number of matrix function.
funm_ev	Evaluate general matrix function via eigensystem.
funm_simple	Simplified Schur–Parlett method for function of a matrix.
logm_cond	Relative condition number of matrix logarithm.
logm_frechet_pade	Fréchet derivative of matrix logarithm via Padé approximation.
logm_iss	Matrix logarithm by inverse scaling and squaring method.
logm_pade_pf	Evaluate Padé approximant to matrix logarithm by partial fraction form.
mft_test	Test the Matrix Function Toolbox.
mft_tolerance	Convergence tolerance for matrix iterations.
polar_newton	Polar decomposition by scaled Newton iteration.
polar_svd	Canonical polar decomposition via singular value decomposition.
polyvalm_ps	Evaluate polynomial at matrix argument by Paterson–Stockmeyer algorithm.
power_binary	Power of matrix by binary powering (repeated squaring).
quasitriang_struct	Block structure of upper quasitriangular matrix.
riccati_xaxb	Solve Riccati equation $XAX = B$ in positive definite matrices.
rootpm_newton	Coupled Newton iteration for matrix pth root.
rootpm_real	pth root of real matrix via real Schur form.
rootpm_schur_newton	Matrix pth root by Schur–Newton method.
rootpm_sign	Matrix pth root via matrix sign function.
signm	Matrix sign decomposition.
signm_newton	Matrix sign function by Newton iteration.
sqrtm_db	Matrix square root by Denman–Beavers iteration.
sqrtm_dbp	Matrix square root by product form of Denman–Beavers iteration.
sqrtm_newton	Matrix square root by Newton iteration (unstable).
sqrtm_newton_full	Matrix square root by full Newton method.
sqrtm_pd	Square root of positive definite matrix via polar decomposition.
sqrtm_pulay	Matrix square root by Pulay iteration.
sqrtm_real	Square root of real matrix by real Schur method.
sqrtm_triang_min_norm	Estimated minimum norm square root of triangular matrix.
sylvsol	Solve Sylvester equation.

8 Other MATLAB Functions

Various other MATLAB functions implementing algorithms developed in research papers are freely available online. We organize them according to the problem that they treat: $f(A)$ or the action of $f(A)$ on a vector, $f(A)b$.

8.1 $f(A)$

- Al-Mohy, Higham, and Relton (2013) [8]: an algorithm for the matrix logarithm (a real version of the algorithm in [7]) together with Fréchet derivatives and condition number estimates. Available from http://www.mathworks.com/matlabcentral/fileexchange/38894-matrix-logarithm-with-frechet-derivatives-and-condition-number.

- Al-Mohy, Higham, and Relton (2015) [9]: algorithms for the matrix cosine, the matrix sine, and for simultaneous evaluation of the matrix cosine and sine. Available from https://github.com/sdrelton/cosm_sinm and http://uk.mathworks.com/matlabcentral/fileexchange/53130-matrix-cosine-and-sine-functions.

- Aprahamian and Higham (2014) [10]: an algorithm for computing the matrix unwinding function. Available at http://eprints.ma.man.ac.uk/2094.

- Aprahamian and Higham (2016) [11]: algorithms for the matrix inverse cosine, inverse sine, inverse hyperbolic cosine, and inverse hyperbolic sine. Available at https://github.com/higham/matrix-inv-trig-hyp.

- Deadman and Higham (2014) [25]: a method for testing matrix function algorithms using identities. Available from https://github.com/edvindeadman/testing_matrix_functions.

- Fasi, Higham, and Iannazzo (2015) [31]: the Lambert W function of a matrix. Available from http://riccati.dm.unipi.it/software/lambertw.

- Greco and Iannazzo (2010) [38]: a binary powering Schur algorithm for matrix roots. Available from http://poisson.phc.unipi.it/~maxreen/bruno/codes/rootpm_real_2.m.

- Higham and Lin (2013) [54]: a Schur–Padé algorithm for fractional matrix powers together with Fréchet derivatives and condition estimates. Available from http://www.mathworks.com/matlabcentral/fileexchange/41621-fractional-matrix-powers-with-frechet-derivatives-and-condition-number-estimate

- Iannazzo and Manasse (2013) [58]: a Schur logarithmic algorithm for fractional powers of matrices. Available from http://poisson.phc.unipi.it/~maxreen/bruno/codes/pthrootlog.m.

8.2 $f(A)b$

- Al-Mohy and Higham (2011) [6]: a scaled Taylor series algorithm for the action of the matrix exponential on a vector. Available from http://www.mathworks.com/matlabcentral/fileexchange/29576-matrix-exponential-times-a-vector

- Caliari, Kandolf, Ostermann, and Rainer (2014) [20]: a method for computing the action of the matrix exponential (for a matrix with spectrum in the left half of the complex plane) based on interpolation at Leja points. Available from http://www.mathworks.com/matlabcentral/fileexchange/44039-matrix-exponential-times-a-vector

  Caliari, Kandolf, Ostermann, and Rainer (2015) [21]: a newer version of the Leja algorithm with no restriction on the spectrum of A. Available from https://numerical-analysis.uibk.ac.at/exp-int-software.

- Eiermann and Güttel (2008) [2], [28]: a restarted Krylov algorithm for computing $f(A)b$; it also implements deflated restarting [29] and harmonic Arnoldi approximation [32]. Available from http://www.guettel.com/funm_kryl.
- Frommer, Güttel and Schweitzer (2014) [33]: a quadrature-based restarted Krylov method for $f(A)b$. It implements deflated restarting [29]. Available from http://www.guettel.com/funm_quad.
- Güttel (2010): a rational Chebyshev series method for computing $\mathrm{e}^{A}b$, where A is symmetric and has no positive eigenvalues. Available from http://www.mathworks.com/matlabcentral/fileexchange/28199-matrix-exponential.
- Güttel and Knizhnerman (2011) [40]: a black-box rational Arnoldi method for computing $f(A)b$, where f is a Markov matrix function. Available from http://guettel.com/markovfunmv. See also [41].
- Hale, Higham and Trefethen (2008) [42]: algorithms for evaluating $f(A)$ and $f(A)b$ by contour integration. MATLAB code is embedded in the paper.
- Kloster and Gleich [62]: an algorithm for computing a column of the exponential of a stochastic matrix. Code is available from https://www.cs.purdue.edu/homes/dgleich/codes/nexpokit.

9 Expokit

Expokit (Sidje, 1998) [79], [80] is a package of Fortran and MATLAB codes to compute e^{A} (using scaling and squaring) and $\mathrm{e}^{A}b$ (using Krylov subspace methods). An R interface to Expokit is available at http://cran.r-project.org/web/packages/expoRkit.

10 EXPINT

EXPINT (Berland, Skaflestad and Wright, 2007) [16], [17] is a MATLAB package providing exponential integrators for ordinary differential equations. A large range of Runge–Kutta, multistep, and general linear integrators is available. The functions $\varphi_k(z) = \sum_{j=0}^{\infty} z^j/(j+k)!$ underlying the methods are evaluated at matrix arguments using Padé approximants with a scaling and squaring scheme.

11 GNU Octave

GNU Octave [34] is an open source problem-solving environment (PSE)[1] with a high-level programming language similar to (and mostly compatible with) MATLAB. It contains several matrix function routines.

- `expm`: matrix exponential by Ward's version of the scaling and squaring algorithm (1977) [83].
- `logm`: matrix logarithm by an inverse scaling and squaring algorithm (Higham, 2008) [47].
- `sqrtm`: matrix square root by the Schur method (Björck and Hammarling, 1983) [19].

[1] A PSE provides a programming language, an interactive command window with the display of graphics, and the ability to export graphics and more generally publish documents to HTML, PDF, TeX, and so on.

An extra package `linear-algebra` is available (http://octave.sourceforge.net/linear-algebra), which contains some additional matrix function routines.

- `thfm`: trigonometric and hyperbolic functions and their inverses. It implements textbook definitions, in terms of `expm`, `logm`, and `sqrtm`.
- `funm`: general matrix function via diagonalization.

12 Scilab

Scilab [78] is another open source PSE. Scilab syntax is similar to that of MATLAB and a code translator is available to convert code from MATLAB to Scilab. Scilab contains several matrix function routines.

- `expm`: matrix exponential using block diagonalization with a Padé approximant applied to each block.
- `logm`: matrix logarithm via diagonalization.
- `sqrtm`: matrix square root via diagonalization.
- Trigonometric and hyperbolic functions, which are implemented via their definitions in terms of the matrix exponential.
- `power`: matrix power using diagonalization for non-integer powers.

13 Maple

Maple contains some matrix function routines in its `LinearAlgebra` package. The matrix functions are computed symbolically using polynomial interpolation at the matrix eigenvalues.

- `MatrixExponential`: exponential of a matrix.
- `MatrixPower`: general (non-integer) power of a matrix.
- `MatrixFunction`: general function of a matrix. The function is supplied in the form of an analytic expression by the user.

14 Mathematica

Mathematica evaluates the functions listed below for numeric or symbolic matrices.

- `MatrixFunction`: evaluates a general matrix function. The Schur–Parlett algorithm (Davies and Higham, 2003) [23] is used for numeric matrices (derivatives are computed symbolically), and the Jordan form is used for symbolic matrices.
- `MatrixExp`: the matrix exponential is computed by scaling and squaring (Higham, 2005) [46], [48]. This function can also compute the action of the matrix exponential on a vector, using Krylov methods.
- `MatrixLog`: the matrix logarithm is evaluated via a Schur decomposition for numeric matrices or the Jordan form for symbolic matrices.

15 R

Three packages from CRAN (the Comprehensive R Archive Network) include codes for matrix functions.

15.1 Expm

Goulet, Dutang, Maechler, Firth, Shapira, and Stadelmann have written the R package expm [37]. It contains codes not only for the matrix exponential but also for the logarithm and square root.

- `expm`: the default method corresponds to option `Higham08.b`, which uses the algorithm of (Higham, 2005) [46], [48] with balancing. The option `AlMohy-Hi09` implements the algorithm of Al-Mohy and Higham [4]. Also available are options for using an eigendecomposition and other Padé and Taylor-based methods.
- `expAtv` uses a Krylov method of Sidje (1998) [80] to compute the action of the matrix exponential on a vector.
- `expmCond`: computes or approximates the 1-norm or Frobenius norm condition number of the matrix exponential using the algorithm of Al-Mohy and Higham (2009) [3] or methods from [47, Sec. 3.4].
- `expmFrechet`: Fréchet derivative of matrix exponential (Al-Mohy and Higham, 2009) [3].
- `logm`: matrix logarithm by inverse scaling and squaring (Higham, 2008) [47, Alg. 11.9].
- `sqrtm`: matrix square root by the Schur method (Björck and Hammarling, 1983) [19].
- `matpow`: positive integer powers via binary powering.

15.2 pbdDMAT

The pbdDMAT (Programming with Big Data—Distributed Matrix Methods) R package [77] includes a code `expm` that implements the scaling and squaring algorithm of Al-Mohy and Higham (2009) [4] for the matrix exponential.

15.3 Matrix

The Matrix R package [14] contains a code `expm` that is "a translation of the implementation of the corresponding Octave function contributed to the Octave project by A. Scottedward Hodel". The `expm` code in the Expm package (see Section 15.1) is to be preferred.

16 φ Functions in Fortran 95

Koikari (2009) [63] has written Fortran 95 software for computing the functions $\varphi_\ell(z) = \sum_{k=0}^{\infty} z^k/(k+\ell)!$ by scaling and squaring and by a block Schur–Parlett algorithm. The code is available as a supplement to the paper on the ACM website.

17 NAG Library

The NAG Library [67] has a large set of matrix function routines in its Chapter F01, covering computation of matrix functions and their Fréchet derivatives and estimation of the condition numbers of matrix functions.

- NAG Fortran Library Mark 23 and NAG Toolbox for MATLAB Mark 23 (released 2011):
  - Matrix exponential using scaling and squaring algorithm (Higham, 2005) [46], [48].
  - Function of real symmetric or Hermitian matrix via eigendecomposition.
- NAG C Library Mark 23, released 2012, also contains:
  - Schur–Parlett algorithm for general functions and for cos, sin, cosh, sinh, exp (Davies and Higham, 2003) [23].
  - Matrix logarithm by Schur–Parlett algorithm with inverse scaling and squaring algorithm (Higham, 2008) [47].
- NAG Fortran Library Mark 24 and the NAG Toolbox for MATLAB Mark 24 (released 2013) also contain:
  - Action of the matrix exponential by scaled Taylor series algorithm (Al-Mohy and Higham, 2011) [6].
  - Condition number estimation in the 1-norm for general matrix functions and for cos, sin, cosh, sinh, exp.
- The NAG C Library Mark 24 (2014), NAG Fortran Library Mark 25 (2015), and NAG Toolbox for MATLAB Mark 25 (2015) also contain:
  - Improved scaling and squaring algorithms for matrix exponential and logarithm (Al-Mohy and Higham, 2009, 2012) [4], [7], with computation of the logarithm of a real matrix in real arithmetic (Al-Mohy, Higham, and Relton, 2013) [8].
  - Matrix square root using Schur method with blocking (including real arithmetic algorithm of Higham [45]) [19], [26].
  - Matrix power $A^p, p \in \mathbb{R}$, via Schur–Padé algorithm (Higham and Lin, 2011, 2013) [53], [54].
  - Latest Fréchet derivative and condition number algorithms for the matrix exponential, logarithm, and real powers [3], [8], [54].

Documentation can be found at: `http://www.nag.co.uk/support_documentation.asp`. A complete list of all of the NAG matrix function routines, together with their Mark of introduction and the algorithms used, is given in Table 3.

18 SLICOT

SLICOT (Subroutine Library in Systems and Control Theory) [81] includes Fortran 77 codes for three matrix functions.

- `MB05MD` computes the exponential of a real matrix by diagonalization, optionally with balancing.

Table 3: Matrix Function Routines in the NAG Library.

Short Name	Long Name	Purpose	Arithmetic	Algorithms Used	Marks of Introduction
F01EC	nag_real_gen_matrix_exp	Exponential	Real	Scaling & squaring [4]	FL22 & CL9
F01ED	nag_real_symm_matrix_exp	Exponential, symmetric matrix	Real	Diagonalization	FL23 & CL9
F01EF	nag_matop_real_symm_matrix_fun	Symmetric matrix function	Real	Diagonalization	FL23 & CL23
F01EJ	nag_matop_real_gen_matrix_log	Logarithm	Real	Scaling & squaring [7]	FL24 & CL23
F01EK	nag_matop_real_gen_matrix_fun_std	sin, cos, sinh, cosh or exp	Real	Schur–Parlett [23]	FL24 & CL23
F01EL	nag_matop_real_gen_matrix_fun_num	General user-provided function	Real	Schur–Parlett [23]	FL24 & CL24
F01EM	nag_matop_real_gen_matrix_fun_usd	General user-provided function	Real	Schur–Parlett [23]	FL24 & CL23
F01EN	nag_matop_real_gen_matrix_sqrt	Square root	Real	[19], [45], [26]	FL25 & CL24
F01EP	nag_matop_real_tri_matrix_sqrt	Upper triangular square root	Real	[19], [45], [26]	FL25 & CL24
F01EQ	nag_matop_real_gen_matrix_pow	General real power	Real	[53], [54]	FL25 & CL24
F01FC	nag_complex_gen_matrix_exp	Exponential	Complex	Scaling & squaring [4]	FL23 & CL23
F01FD	nag_complex_gen_matrix_exp	Exponential, Hermitian matrix	Complex	Diagonalization	FL23 & CL23
F01FF	nag_matop_complex_symm_matrix_fun	Hermitian matrix function	Complex	Diagonalization	FL23 & CL23
F01FJ	nag_matop_real_gen_matrix_log	Logarithm	Complex	Scaling & squaring [7]	FL24 & CL23
F01FK	nag_matop_real_gen_matrix_fun_std	sin, cos, sinh, cosh or exp	Complex	Schur–Parlett [23]	FL24 & CL23
F01FL	nag_matop_real_gen_matrix_fun_num	General user-provided function	Complex	Schur–Parlett [23]	FL24 & CL24
F01FM	nag_matop_real_gen_matrix_fun_usd	General user-provided function	Complex	Schur–Parlett [23]	FL24 & CL23
F01FN	nag_matop_real_gen_matrix_sqrt	Square root	Complex	[19], [26]	FL25 & CL24
F01FP	nag_matop_real_tri_matrix_sqrt	Upper triangular square root	Complex	[19], [26]	FL25 & CL24
F01FQ	nag_matop_real_gen_matrix_pow	General real power	Complex	[53], [54]	FL25 & CL24
F01GA	nag_matop_real_gen_matrix_actexp	Action of matrix exponential	Real	[6]	FL24 & CL24
F01GB	nag_matop_real_gen_matrix_actexp_rcomm	Action of matrix exponential	Real	[6] rev comm[1]	FL24 & CL24
F01HA	nag_matop_complex_gen_matrix_actexp	Action of matrix exponential	Complex	[6]	FL24 & CL24
F01HB	nag_matop_complex_gen_matrix_actexp_rcomm	Action of matrix exponential	Complex	[6] rev comm[1]	FL24 & CL24
F01JA	nag_matop_real_gen_matrix_cond_std	sin, cos, sinh, cosh, exp cond	Real	Schur–Parlett [23]	FL24 & CL24
F01JB	nag_matop_real_gen_matrix_cond_num	User function, condition	Real	Schur–Parlett [23]	FL24 & CL24
F01JC	nag_matop_real_gen_matrix_cond_usd	User function, condition	Real	Schur–Parlett [23]	FL24 & CL24
F01JD	nag_matop_real_gen_matrix_cond_sqrt	Square root, condition	Real	[19], [45], [26]	FL25 & CL24

[1] "Rev comm" denotes a reverse communication interface.

F01JE	nag_matop_real_gen_matrix_cond_pow	General real power, condition	Real	[54]	FL25 & CL24
F01JF	nag_matop_real_gen_matrix_frcht_pow	Real power, Fréchet derivative	Real	[54]	FL25 & CL24
F01JG	nag_matop_real_gen_matrix_cond_exp	Exponential, condition number	Real	[3]	FL25 & CL24
F01JH	nag_matop_real_gen_matrix_frcht_exp	Exponential, Fréchet derivative	Real	[3]	FL25 & CL24
F01JJ	nag_matop_real_gen_matrix_cond_log	Logarithm, condition number	Real	[7], [8]	FL25 & CL24
F01JK	nag_matop_real_gen_matrix_frcht_log	Logarithm, Fréchet derivative	Real	[7], [8]	FL25 & CL24
F01KA	nag_matop_complex_gen_matrix_cond_std	sin, cos, sinh, cosh, exp cond	Complex	Schur–Parlett [23]	FL24 & CL24
F01KB	nag_matop_complex_gen_matrix_cond_num	User function, condition	Complex	Schur–Parlett [23]	FL24 & CL24
F01KC	nag_matop_complex_gen_matrix_cond_usd	User function, condition	Complex	Schur–Parlett [23]	FL24 & CL24
F01KD	nag_matop_complex_gen_matrix_cond_sqrt	Square root, condition	Complex	[19], [26]	FL25 & CL24
F01KE	nag_matop_complex_gen_matrix_cond_pow	General real power, condition	Complex	[54]	FL25 & CL24
F01KF	nag_matop_complex_gen_matrix_frcht_pow	Real power, Fréchet derivative	Complex	[54]	FL25 & CL24
F01KG	nag_matop_complex_gen_matrix_cond_exp	Exponential, condition number	Complex	[3]	FL25 & CL24
F01KH	nag_matop_complex_gen_matrix_frcht_exp	Exponential, Fréchet derivative	Complex	[3]	FL25 & CL24
F01KJ	nag_matop_complex_gen_matrix_cond_log	Logarithm, condition number	Complex	[7], [8]	FL25 & CL24
F01KK	nag_matop_complex_gen_matrix_frcht_log	Logarithm, Fréchet derivative	Complex	[7], [8]	FL25 & CL24

- `MB05ND` computes the matrix exponential and an integral involving the matrix exponential for a real matrix
- `MB05OD` computes the exponential of a real matrix by Ward's version of the scaling and squaring algorithm (1977) [83] and returns an accuracy estimate.

19 Python

19.1 SciPy

SciPy [60] is a Python package for scientific computing. It has a number of matrix function codes.

- `sparse.linalg.expm`: matrix exponential by scaling and squaring algorithm (Al-Mohy and Higham, 2009) [4].
- `linalg.expm3`, `linalg.expm2`, `linalg.expm`: matrix exponential using Taylor series, eigenvalue decomposition, and scaling and squaring (Higham, 2005) [46], respectively.
- `linalg.logm`: matrix logarithm via inverse scaling and squaring algorithm (Al-Mohy and Higham, 2012) [7].
- `linalg.sinm`, `linalg.cosm`, `linalg.tanm`, `linalg.sinhm`, `linalg.coshm`, `linalg.tanhm`: implemented in terms of `linalg.expm`.
- `linalg.funm`: unblocked Schur–Parlett algorithm [36, Alg. 9.1.1], [47, Alg. 4.1.3].
- `linalg.fractional_matrix_power`: arbitrary real power of matrix by Schur–Padé algorithm (Higham and Lin, 2011, 2013) [53], [54].
- `linalg.signm`: matrix sign function via Newton iteration [47, Sec. 5.3].
- `linalg.expm_frechet`: Fréchet derivative of matrix exponential by scaling and squaring algorithm (Al-Mohy and Higham, 2009) [3].
- `linalg.expm_cond`: Frobenius norm condition number of matrix exponential.
- `linalg.sqrtm`: matrix square root by blocked version of Schur method of Björck and Hammarling (1983) [19], [26].
- `linalg.expm_multiply`: action of matrix exponential on a vector or matrix via scaled Taylor series algorithm (Al-Mohy and Higham, 2011) [6].
- `linalg.polar`: polar decomposition via the singular value decomposition [47, Chap. 8].

19.2 Python: SymPy

SymPy [82] is a Python library for symbolic mathematics. It contains some numerical matrix function capabilities in its `mpmath` module and variable precision arithmetic is supported. The precision is set using either `mp.prec` (to set the binary precision, measured in bits) or `mp.dps` (to set the decimal precision).

- `mpmath.expm`: matrix exponential by scaling and squaring with Taylor series or Padé approximant.

- `mpmath.logm`: matrix logarithm evaluated via inverse scaling and squaring and a Taylor series.

- `mpmath.sinm`, `mpmath.cosm`: matrix sine and cosine implemented via the matrix exponential.

- `mpmath.sqrtm`: matrix square root evaluated using the Denman–Beavers (1976) iteration [27].

- `mpmath.powm`: A^p for $p \in \mathbb{C}$, implemented as $\mathrm{e}^{p \log A}$.

Note that both SymPy and SciPy are available in Sage [76], an open source Python-based mathematical software package.

19.3 Other Python Functions

Python implementations of the algorithms in Deadman (2015) [24], for computing the condition number of $f(A)b$, are available at `https://github.com/edvindeadman/fAbcond`.

20 Java

A Java implementation by Bilge of the algorithm of Al-Mohy and Higham (2011) [6] for the action of the matrix exponential on a vector is available from `https://github.com/armanbilge/AMH11`.

21 Julia

Julia [18], [61] is an open-source, high-level, dynamic programming language designed specifically for high-performance numerical and scientific computing. Its extensive mathematical function library is largely written in Julia itself, but also includes calls to other libraries such as LAPACK and OpenBLAS. Four matrix function routines are available in the Julia Standard Library:

- `expm`: matrix exponential using the scaling and squaring algorithm of Higham (2005) [46], [48].

- `logm`: matrix logarithm by inverse scaling and squaring algorithm of Al-Mohy and Higham (2012) [7].

- `sqrtm`: matrix square root using the Schur method of Björck and Hammarling (1983) [19].

22 C++: Eigen

Niesen has written a matrix functions module for the Eigen C++ template library for linear algebra [69].

- `MatrixBase::exp()`: matrix exponential by scaling and squaring algorithm (Higham, 2005) [46], [48].

- `MatrixBase::sin()`, `MatrixBase::sinh()`, `MatrixBase::cos()`, `MatrixBase::cosh()` and `MatrixBase::matrixFunction()` are all based on the Schur–Parlett algorithm (Davies and Higham, 2003) [23].

- `MatrixBase::log()`: matrix logarithm by the Schur–Parlett algorithm with inverse scaling and squaring [47, Alg. 11.11].
- `MatrixBase::pow()`: real matrix powers A^t ($t \in \mathbb{R}$) using the Schur–Padé algorithm (Higham and Lin, 2011) [53].
- `MatrixBase::sqrt()`: matrix square root by Schur method (Björck and Hammarling, 1983) [19] and the real Schur method (Higham, 1987) [45].

23 The GNU Scientific Library

The GNU Scientific Library (GSL) is an open-source numerical library written in C (although wrappers exist for many other programming languages) [35]. GSL includes an undocumented function `gsl_linalg_exponential_ss` to compute the matrix exponential. This routine uses scaling and squaring and a truncated Taylor series. The scaling and truncation parameters are chosen as in Moler and Van Loan (2003) [66, Method 3].

Acknowledgements

We thank Awad Al-Mohy, Michael Croucher, Massimiliano Fasi, Stefan Güttel, Sven Hammarling, Peter Kandolf, Lijing Lin, and Samuel Relton for comments and suggestions.

References

[1] *Multiprecision Computing Toolbox*. Advanpix, Tokyo. http://www.advanpix.com.

[2] Martin Afanasjew, Michael Eiermann, Oliver G. Ernst, and Stefan Güttel. Implementation of a restarted Krylov subspace method for the evaluation of matrix functions. *Linear Algebra Appl.*, 429(10):2293–2314, 2008.

[3] Awad H. Al-Mohy and Nicholas J. Higham. Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM J. Matrix Anal. Appl.*, 30(4):1639–1657, 2009.

[4] Awad H. Al-Mohy and Nicholas J. Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 31(3):970–989, 2009.

[5] Awad H. Al-Mohy and Nicholas J. Higham. The complex step approximation to the Fréchet derivative of a matrix function. *Numer. Algorithms*, 53(1):133–148, 2010.

[6] Awad H. Al-Mohy and Nicholas J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM J. Sci. Comput.*, 33(2):488–511, 2011.

[7] Awad H. Al-Mohy and Nicholas J. Higham. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM J. Sci. Comput.*, 34(4):C153–C169, 2012.

[8] Awad H. Al-Mohy, Nicholas J. Higham, and Samuel D. Relton. Computing the Fréchet derivative of the matrix logarithm and estimating the condition number. *SIAM J. Sci. Comput.*, 35(4):C394–C410, 2013.

[9] Awad H. Al-Mohy, Nicholas J. Higham, and Samuel D. Relton. New algorithms for computing the matrix sine and cosine separately or simultaneously. *SIAM J. Sci. Comput.*, 37 (1):A456–A487, 2015.

[10] Mary Aprahamian and Nicholas J. Higham. The matrix unwinding function, with an application to computing the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 35(1): 88–109, 2014.

[11] Mary Aprahamian and Nicholas J. Higham. Matrix inverse trigonometric and inverse hyperbolic functions: Theory and algorithms. MIMS EPrint 2016.4, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, January 2016. 25 pp.

[12] Vincent Arsigny, Olivier Commowick, Nicholas Ayache, and Xavier Pennec. A fast and log–Euclidean polyaffine framework for locally linear registration. *J. Math. Imaging Vis*, 33:222–238, 2009.

[13] Richard Barakat. Exponential versions of the Jones and Mueller-Jones polarization matrices. *J. Opt. Soc. Am. A*, 13(1):158–163, 1996.

[14] Douglas Bates and Martin Maechler. R package Matrix: Sparse and dense matrix classes and methods. `https://cran.r-project.org/web/packages/Matrix/index.html`.

[15] Michele Benzi, Ernesto Estrada, and Christine Klymko. Ranking hubs and authorities using matrix functions. *Linear Algebra Appl.*, 438(5):2447–2474, 2013.

[16] Håvard Berland, Bård Skaflestad, and Will Wright. EXPINT. `http://www.math.ntnu.no/num/expint/matlab.php`.

[17] Håvard Berland, Bård Skaflestad, and Will Wright. EXPINT—A MATLAB package for exponential integrators. *ACM Trans. Math. Software*, 33(1):Article 4, 2007.

[18] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. ArXiv preprint 1411.1607, 2014. 37 pp.

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

[20] Marco Caliari, Peter Kandolf, Alexander Ostermann, and Stefan Rainer. Comparison of software for computing the action of the matrix exponential. *BIT*, 54:113–128, 2014.

[21] Marco Caliari, Peter Kandolf, Alexander Ostermann, and Stefan Rainer. The Leja method revisited: Backward error analysis for the matrix exponential. Technical report, June 2015. 25 pp. ArXiv preprint 1506.08665.

[22] Russell A. Chipman. Mueller matrices. In *Handbook of Optics: Volume I—Geometrical and Physical Optics, Polarized Light, Components and Instruments*, Michael Bass, editor, McGraw-Hill, New York, third edition, 2010, pages 14.1–14.44.

[23] Philip I. Davies and Nicholas J. Higham. A Schur–Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.*, 25(2):464–485, 2003.

[24] Edvin Deadman. Estimating the condition number of $f(A)b$. *Numer. Algorithms*, 70(2): 287–308, 215.

[25] Edvin Deadman and Nicholas J. Higham. Testing matrix function algorithms using identities. MIMS EPrint 2014.13, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, March 2014. 15 pp. Revised January 2015. To appear in ACM Trans. Math. Software.

[26] Edvin Deadman, Nicholas J. Higham, and Rui Ralha. Blocked Schur algorithms for computing the matrix square root. In *Applied Parallel and Scientific Computing: 11th International Conference, PARA 2012, Helsinki, Finland*, P. Manninen and P. Öster, editors, volume 7782 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2013, pages 171–182.

[27] Eugene D. Denman and Alex N. Beavers, Jr. The matrix sign function and computations in systems. *Appl. Math. Comput.*, 2:63–94, 1976.

[28] Michael Eiermann and Oliver G. Ernst. A restarted Krylov subspace method for the evaluation of matrix functions. *SIAM J. Numer. Anal.*, 44(6):2481–2504, 2006.

[29] Michael Eiermann, Oliver G. Ernst, and Stefan Güttel. Deflated restarting for matrix functions. *SIAM J. Matrix Anal. Appl.*, 32(2):621–641, 2011.

[30] Ernesto Estrada and Desmond J. Higham. Network properties revealed through matrix functions. *SIAM Rev.*, 52(4):696–714, 2010.

[31] Massimiliano Fasi, Nicholas J. Higham, and Bruno Iannazzo. An algorithm for the matrix Lambert W function. *SIAM J. Matrix Anal. Appl.*, 36(2):669–685, 2015.

[32] Andreas Frommer, Stefan Güttel, and Marcel Schweitzer. Convergence of restarted Krylov subspace methods for Stieltjes functions of matrices. *SIAM J. Matrix Anal. Appl.*, 35(4): 1602–1624, 2014.

[33] Andreas Frommer, Stefan Güttel, and Marcel Schweitzer. Efficient and stable Arnoldi restarts for matrix functions based on quadrature. *SIAM J. Matrix Anal. Appl.*, 35(2): 661–683, 2014.

[34] GNU Octave. `http://www.octave.org`.

[35] GNU Scientific Library. `http://www.gnu.org/software/gsl`.

[36] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Fourth edition, Johns Hopkins University Press, Baltimore, MD, USA, 2013. xxi+756 pp. ISBN 978-1-4214-0794-4.

[37] Vincent Goulet, Christophe Dutang, Martin Maechler, David Firth, Marina Shapira, and Michael Stadelmann. R package expm. `http://cran.r-project.org/web/packages/expm/index.html`.

[38] Federico Greco and Bruno Iannazzo. A binary powering algorithm for computing primary matrix roots. *Numer. Algorithms*, 55(1):59–78, 2010.

[39] Peter Grindrod and Desmond J. Higham. A dynamical systems view of network centrality. *Proc. Roy. Soc. London Ser. A*, 470(2165), 2014.

[40] Stefan Güttel and Leonid Knizhnerman. Automated parameter selection for rational Arnoldi approximation of Markov functions. *Proc. Appl. Math. Mech.*, 11(1):15–18, 2011.

[41] Stefan Güttel and Leonid Knizhnerman. A black-box rational Arnoldi variant for Cauchy–Stieltjes matrix functions. *BIT*, 53(3):595616, 2013.

[42] Nicholas Hale, Nicholas J. Higham, and Lloyd N. Trefethen. Computing A^α, $\log(A)$, and related matrix functions by contour integrals. *SIAM J. Numer. Anal.*, 46(5):2505–2523, 2008.

[43] Timothy F. Havel, Igor Najfeld, and Ju-xing Yang. Matrix decompositions of two-dimensional nuclear magnetic resonance spectra. *Proc. Nat. Acad. Sci. USA*, 91:7962–7966, 1994.

[44] Nicholas J. Higham. The Matrix Function Toolbox. `http://www.maths.manchester.ac.uk/~higham/mftoolbox`.

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

[46] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, 2005.

[47] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. xx+425 pp. ISBN 978-0-898716-46-7.

[48] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Rev.*, 51(4):747–764, 2009.

[49] Nicholas J. Higham. Functions of matrices. In *Handbook of Linear Algebra*, Leslie Hogben, editor, Chapman and Hall/CRC, Boca Raton, FL, USA, second edition, 2014, pages 17.1–17.15.

[50] Nicholas J. Higham and Awad H. Al-Mohy. Computing matrix functions. *Acta Numerica*, 19:159–208, 2010.

[51] Nicholas J. Higham and Edvin Deadman. A catalogue of software for matrix functions. Version 1.0. MIMS EPrint 2014.8, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, February 2014. 19 pp.

[52] Nicholas J. Higham and Lijing Lin. On pth roots of stochastic matrices. *Linear Algebra Appl.*, 435(3):448–463, 2011.

[53] Nicholas J. Higham and Lijing Lin. A Schur–Padé algorithm for fractional powers of a matrix. *SIAM J. Matrix Anal. Appl.*, 32(3):1056–1078, 2011.

[54] Nicholas J. Higham and Lijing 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, 2013.

[55] Nicholas J. Higham and Lijing Lin. Matrix functions: A short course. In *Matrix Functions and Matrix Equations*, Zhaojun Bai, Weiguo Gao, and Yangfeng Su, editors, number 19 in *Series in Contemporary Applied Mathematics*, World Scientific, Singapore, 2015, pages 1–27.

[56] Nicholas J. Higham and Françoise Tisseur. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. *SIAM J. Matrix Anal. Appl.*, 21(4):1185–1201, 2000.

[57] Weiming Hu, Haiqiang Zuo, Ou Wu, Yunfei Chen, Zhongfei Zhang, and David Suter. Recognition of adult images, videos, and web page bags. *ACM Trans. Multimedia Comput. Commun. Appl.*, 78:28:1–28:24, 2011.

[58] Bruno Iannazzo and Carlo Manasse. A Schur logarithmic algorithm for fractional powers of matrices. *SIAM J. Matrix Anal. Appl.*, 34(2):794–813, 2013.

[59] Robert B. Israel, Jeffrey S. Rosenthal, and Jason Z. Wei. Finding generators for Markov chains via empirical transition matrices, with applications to credit ratings. *Math. Finance*, 11(2):245–265, 2001.

[60] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. `http://www.scipy.org/`.

[61] Julia. `http://julialang.org`.

[62] Kyle Kloster and David F. Gleich. Sublinear column-wise actions of the matrix exponential on social networks, 2013. `http://arxiv.org/abs/1310.3423`, revised March 2015. 20 pp.

[63] Souji Koikari. Algorithm 894: On a block Schur–Parlett algorithm for φ-functions based on the sep-inverse estimate. *ACM Trans. Math. Software*, 36(2):Article 12, 2009.

[64] *Symbolic Math Toolbox*. The MathWorks, Inc., Natick, MA, USA. `http://www.mathworks.co.uk/products/symbolic/`.

[65] Cleve B. Moler. A balancing act for the matrix exponential. `http://blogs.mathworks.com/cleve/2012/07/23/a-balancing-act-for-the-matrix-exponential/`, July 2012.

[66] Cleve B. Moler and Charles F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49, 2003.

[67] NAG Library. NAG Ltd., Oxford, UK. `http://www.nag.co.uk`.

[68] Igor Najfeld and Timothy F. Havel. Derivatives of the matrix exponential and their computation. *Adv. Appl. Math.*, 16:321–375, 1995.

[69] Jitse Niesen. Eigen matrix functions module. `http://eigen.tuxfamily.org/dox-devel//unsupported/group__MatrixFunctions__Module.html`.

[70] H. D. Noble and R. A. Chipman. Mueller matrix roots algorithm and computational considerations. *Opt. Express*, 20(1):17–31, 2012.

[71] Razvigor Ossikovski. Differential matrix formalism for depolarizing anisotropic media. *Optics Letters*, 36(12):2330–2332, 2011.

[72] Simon T. Parker, David M. Lorenzetti, and Michael D. Sohn. Implementing state-space methods for multizone contaminant transport. *Building and Environment*, 71:131–139, 2014.

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

[74] Maria Pusa and Jaakko Leppänen. Computing the matrix exponential in burnup calculations. *Nuclear Science and Engineering*, 164:140–150, 2010.

[75] Jarek Rossignac and Àlvar Vinacua. Steady affine motions and morphs. *ACM Trans. Graphics*, 30(5):116:1–116:16, 2011.

[76] The Sage Developers. Sage Mathematics Software. http://www.sagemath.org.

[77] Drew Schmidt, Wei-Chen Chen, George Ostrouchov, Pragneshkumar Patel, and R Core team. R package pbddmat: Programming with big data—distributed matrix methods. https://cran.r-project.org/web/packages/pbdDMAT/index.html.

[78] Scilab. http://www.scilab.org.

[79] Roger B. Sidje. Expokit. http://www.maths.uq.edu.au/expokit.

[80] Roger B. Sidje. Expokit: A software package for computing matrix exponentials. *ACM Trans. Math. Software*, 24(1):130–156, 1998.

[81] SLICOT: A subroutine library in systems and control theory. http://www.slicot.org/.

[82] SymPy. http://www.sympy.org.

[83] Robert C. Ward. Numerical computation of the matrix exponential with accuracy estimate. *SIAM J. Numer. Anal.*, 14(4):600–610, 1977.