Testing Matrix Function Algorithms Using Identities

EDVIN DEADMAN and NICHOLAS J. HIGHAM, The University of Manchester

Algorithms for computing matrix functions are typically tested by comparing the forward error with the product of the condition number and the unit roundoff. The forward error is computed with the aid of a reference solution, typically computed at high precision. An alternative approach is to use functional identities such as the "round-trip tests" $e^{\log A} = A$ and $(A^{1/p})^p = A$, as are currently employed in a SciPy test module. We show how a linearized perturbation analysis for a functional identity allows the determination of a maximum residual consistent with backward stability of the constituent matrix function evaluations. Comparison of this maximum residual with a computed residual provides a necessary test for backward stability. We also show how the actual linearized backward error for these relations can be computed. Our approach makes use of Fréchet derivatives and estimates of their norms. Numerical experiments show that the proposed approaches are able both to detect instability and to confirm stability.

Categories and Subject Descriptors: G.4 [**Mathematical Software**]: Algorithm design and analysis; G.1.3 [**Numerical Linear Algebra**]

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Matrix function, normwise relative error, functional identity, testing, forward error, backward error, MATLAB

ACM Reference Format:
Edvin Deadman and Nicholas J. Higham. 2016. Testing matrix function algorithms using identities. ACM Trans. Math. Softw. 42, 1, Article 4 (January 2016), 15 pages.
DOI: http://dx.doi.org/10.1145/2723157

1. INTRODUCTION

In recent years, much effort has been devoted to developing new and improved algorithms for computing functions of matrices $X = f(A)$, $A \in \mathbb{C}^{n\times n}$, where f is an underlying scalar function and $f(A) \in \mathbb{C}^{n\times n}$ is defined, for example, via the Jordan canonical form [Higham 2008, Section 1.2; Horn and Johnson 1991, Section 6.2]. Important functions include the exponential, logarithm, and fractional matrix powers. Most testing of such algorithms has been based on approximations to the forward error $\|X - \widehat{X}\|/\|X\|$ of the computed $\widehat{X}$. Here, X is computed by a trusted reference algorithm at the working precision or any algorithm using higher-precision arithmetic, or might be known exactly by the construction of the problem. The forward error is compared with the product of the condition number of the problem and the unit roundoff in order to see whether the algorithm is behaving in a forward stable way.

This work was supported by European Research Council Advanced Grant MATFUN (no. 267526). The second author was also supported by Engineering and Physical Sciences Research Council grant no. EP/I006702/1.
Authors' addresses: E. Deadman and N. J. Higham, School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK; emails: edvin.deadman@manchester.ac.uk, nick.higham@manchester.ac.uk.

DOI: http://dx.doi.org/10.1145/2723157

The forward error approach has drawbacks. A reference algorithm superior in accuracy to the algorithm being tested may not be available, and the exact solution is not always known. While the use of higher-precision arithmetic is valuable for algorithm development in a single programming environment, it is not appropriate in a software engineering setting in which implementations of an algorithm in a language such as C or Fortran are tested on multiple systems with different compilers. Furthermore, high-precision arithmetic is expensive—its cost limits the size of the test matrices that can be used.

In this work, we develop an approach originally employed by Cody [1993] for testing algorithms for evaluating elementary functions of scalars. Suppose that $Q(f_1, \dots, f_k) = 0$ is an identity in the functions $f_1, \dots, f_k$ and is such that $Q(f_1(A), \dots, f_k(A)) = 0$ for $A \in \mathbb{C}^{n\times n}$. Examples of such identities are $e^{\log A} = A$ and $\sin(2A) - 2\sin A\cos A = 0$; general results saying when scalar identities translate to the matrix case are available [Higham 2008, Theorems 1.16, 1.17]. If each of the function evaluations $F_j = f_j(A)$ were to be done in a backward stable way, then we would have computed matrices $\widehat{F}_j = f_j(A + \Delta A_j)$, $\|\Delta A_j\| \le u\|A\|$, where u is the unit roundoff. Suppose that we can solve

$$\max \|Q(f_1(A + \Delta A_1), \dots, f_k(A + \Delta A_k))\| \quad \text{subject to} \quad \|\Delta A_j\| \le u\|A\|, \quad j = 1 : k. \tag{1.1}$$

Then, by comparing the norm of the computed $\widehat{Q}$ with this maximum, we can tell whether the computations are consistent with backward stable function evaluations. The nonlinear optimization problem (1.1) is too difficult to solve in the form stated; thus, we will carry out a linearized analysis, using the Fréchet derivatives of the f_i, and estimate the maximum for the linearized problem.

We also develop an extension of this approach that computes a linearized backward error for the identities. Fréchet derivative expansions are used to obtain an underdetermined linear system that relates the residual to the backward error matrices. The backward error matrices, hence the backward error, are found by computing the minimal norm solution to this system.

The intended use of our identity-based tests is both to compare competing algorithms and to test that implementations of algorithms are working as expected. Since higher-precision arithmetic is not required, our tests are portable and are not limited to matrices of small dimensions. The use of identities also allows us to detect programming errors that might not otherwise be apparent, as, for example, if the code being tested is run in higher-precision arithmetic to obtain the reference solution.

This work does not represent the first use of identities to test matrix function algorithms. For example, Smith [2003], Guo and Higham [2006], Greco and Iannazzo [2010], Higham and Lin [2011], and Iannazzo and Manasse [2013] have all used the identity $X^p - A = 0$, where X is a pth root of A, to assess algorithms for computing matrix pth roots. The identity $e^{\log A} = A$ has been used by Davies and Higham [2003] and Dieci et al. [1996] to test algorithms for the matrix logarithm. Denman and Beavers, Jr. [1976] used the trace of the identity $\operatorname{sign}(A)^2 = I$ to test the convergence of iterations for computing the matrix sign function. Our contribution treats general identities and incorporates the effects of errors in each of the functions therein.

Identities are currently used to test matrix function codes in several numerical software libraries. The Python package SciPy [Jones et al. 2001] uses the identities $(A^{1/p})^p = A$ and $e^{\log A} = A$, which it refers to as "round-trip" tests in the source code on Github at https://github.com/scipy/scipy/blob/master/scipy/linalg/tests/test_matfuncs.py. The NAG Library [NAG Library] makes similar use of identities to provide additional tests for matrix function routines. The Matrix Function Toolbox [Higham 2008] uses the identities $\sin^2 A + \cos^2 A = I$ and $(A^{1/p})^p = A$ in the test code `mft_test.m`

distributed with the toolbox. In all these cases, heuristic tolerances are used in the tests. Our work provides the information needed to make more rigorous choices of tolerance.

In the next section, we introduce some necessary background on Fréchet derivatives and the condition number of a matrix function. In Section 3, we develop the use of identities for compositions of functions, while Section 4 treats the product of functions. In Section 5, we show how to compute linearized backward errors from residuals of the identities using both direct and iterative methods. Numerical experiments in Section 6 demonstrate the ability of the tests to discriminate between stable and unstable behavior. Concluding remarks are given in Section 7.

2. LINEAR OPERATORS, FRÉCHET DERIVATIVES, AND THE CONDITION NUMBER OF A MATRIX FUNCTION

Consider a linear operator $L : \mathbb{C}^{m\times n} \to \mathbb{C}^{p\times q}$. The norm of L is

$$\|L\| = \max_{\|E\|=1} \|L(E)\|, \tag{2.2}$$

where we are using $\|\cdot\|$ to denote both a norm on $\mathbb{C}^{m\times n}$ and a norm on $\mathbb{C}^{p\times q}$. The operator 1-norm $\|L\|_1$ is obtained when we choose the standard matrix 1-norms on $\mathbb{C}^{m\times n}$ and $\mathbb{C}^{p\times q}$. Since L is linear, we can write

$$\operatorname{vec}(L(E)) = K \operatorname{vec}(E), \tag{2.3}$$

where the vec operator stacks the columns of a matrix on top of each other. The matrix $K \in \mathbb{C}^{mn\times pq}$ is called the Kronecker matrix and depends on L but not E. The following result shows that we can estimate the 1-norm of the linear operator L by the 1-norm of the matrix K at the cost of a factor $\max(n, q)$ uncertainty.

LEMMA 2.1. *The linear operator $L : \mathbb{C}^{m\times n} \to \mathbb{C}^{p\times q}$ and its Kronecker matrix K satisfy*

$$\frac{\|L\|_1}{n} \le \|K\|_1 \le q\|L\|_1.$$

PROOF. The proof is essentially the same as that of Higham [2008, Lem. 3.18], which is a special case of this result. For $E \in \mathbb{C}^{m\times n}$, we have $\|E\|_1 \le \|\operatorname{vec}(E)\|_1 \le n\|E\|_1$ (with equality on the left for $E = ee_1^T$ and on the right for $E = ee^T$, where e is the vector of ones and e_1 is the first unit vector). Hence, using Equation (2.3), for nonzero E, we have

$$\frac{1}{n}\frac{\|L(E)\|_1}{\|E\|_1} \le \frac{\|K\operatorname{vec}(E)\|_1}{\|\operatorname{vec}(E)\|_1} \le q\frac{\|L(E)\|_1}{\|E\|_1}.$$

Maximizing over all E gives the result. □

To estimate $\|L\|_1$, we will use the following algorithm, which is essentially Higham [2008, Algorithm 3.22] modified to use a block 1-norm estimator. We need the adjoint of L, $L^\star : \mathbb{C}^{p\times q} \to \mathbb{C}^{m\times n}$, which is defined by the condition

$$\langle L(G), H\rangle = \langle G, L^\star(H)\rangle \tag{2.4}$$

for all $G \in \mathbb{C}^{m\times n}$, $H \in \mathbb{C}^{p\times q}$. Here, the scalar product $\langle X, Y\rangle = \operatorname{trace}(Y^*X) = y^*x$, where $y = \operatorname{vec}(Y)$ and $x = \operatorname{vec}(X)$. Note that $\langle Kx, y\rangle = \langle L(X), Y\rangle$, hence

$$\langle x, K^*y\rangle = \langle Kx, y\rangle = \langle L(X), Y\rangle = \langle X, L^\star(Y)\rangle = \langle x, \operatorname{vec}(L^\star(Y))\rangle.$$

Since this is true for all x, we have

$$K^*y = \operatorname{vec}(L^\star(Y)). \tag{2.5}$$

ALGORITHM 2.2 (1-NORM ESTIMATOR FOR LINEAR OPERATOR). *Given a linear operator $L : \mathbb{C}^{m\times n} \to \mathbb{C}^{p\times q}$ and the ability to compute $L(E)$ and $L^\star(E)$ for any E, this algorithm produces an estimate γ of $\|L\|_1$. More precisely, $\gamma \le \|K\|_1$, where $\|K\|_1 \in [n^{-1}\|L\|_1, q\|L\|_1]$.*

1 Apply the block 1-norm estimator of Higham and Tisseur [2000], with parameter t (the number of columns in the iteration matrix) set to 2, to the matrix K, noting that $Kx \equiv \mathrm{vec}(L(X))$ and $K^*x \equiv \mathrm{vec}(L^\star(X))$, where $\mathrm{vec}(X) = x$.

Note that K is a rectangular matrix. Although it is stated for square matrices, the 1-norm estimation algorithm of Higham and Tisseur [2000] extends naturally to rectangular matrices without needing to pad them with zeros to make them square. The latter algorithm requires about $4t$ matrix–vector products in total and produces estimates almost always within a factor 3 of the true norm.

Let f be defined on an open subset of $\mathbb{C}$ and consider the corresponding matrix function. The relative condition number $\mathrm{cond}(f, A)$ for $A \in \mathbb{C}^{n\times n}$ is

$$\mathrm{cond}(f, A) = \lim_{\epsilon\to 0} \sup_{\|\Delta A\| \le \epsilon \|A\|} \frac{\|f(A+\Delta A) - f(A)\|}{\epsilon \|f(A)\|}.$$

The function f is Fréchet differentiable at $A \in \mathbb{C}^{n\times n}$ if there exists a linear operator $L_f(A, \cdot) : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$, called the Fréchet derivative, such that, for all $E \in \mathbb{C}^{n\times n}$

$$f(A+E) = f(A) + L_f(A, E) + o(\|E\|). \tag{2.6}$$

If f is Fréchet differentiable, then the relative condition number can be expressed in terms of the Fréchet derivative as in Higham [2008, Thm. 3.1]:

$$\mathrm{cond}(f, A) = \|L_f(A)\| \frac{\|A\|}{\|f(A)\|}. \tag{2.7}$$

To estimate the condition number, we need to estimate the norm of $L_f(A)$. This can be done using Algorithm 2.2 if we can compute the adjoint of $L_f(A)$. In most cases of interest for matrix functions f, including for any analytic function having a Taylor series with real coefficients, $L_f^\star(A, X) = L_f(A, X^*)^*$; see Higham and Lin [2013, sec. 6] for details.

Let $\widehat{X}$ denote the computed approximation to a matrix function $X = f(A)$. The normwise relative forward error is $\|X - \widehat{X}\|/\|X\|$. The normwise relative backward error is

$$\eta(\widehat{X}) = \min\left\{ \frac{\|\Delta A\|}{\|A\|} : \widehat{X} = f(A+\Delta A) \right\}. \tag{2.8}$$

Of course, $\eta(\widehat{X})$ could be undefined, for example, if $\widehat{X}$ is singular and f is the exponential. A useful rule of thumb following from these definitions is that the forward error is approximately bounded by the product of the condition number and the backward error.

3. COMPOSITION OF MATRIX FUNCTIONS

Consider Fréchet differentiable matrix functions f and g satisfying the identity $A = f(g(A))$ in some region. Such identities include $e^{\log A} = A$, $(A^{1/2})^2 = A$, and $\sin(\sin^{-1} A) = A$. We assume that a computed approximation $\widehat{X}$ to $f(g(A))$ can be written

$$\widehat{X} = f(g(A+E_1) + E_2), \tag{3.9}$$

$$\|E_1\| \le \epsilon\|A\|, \quad \|E_2\| \le \epsilon\|g(A)\|. \tag{3.10}$$

This assumption can be interpreted as saying that the f and g evaluations are both backward stable in floating point arithmetic if ϵ is a small multiple of the unit roundoff.

The question of interest is how large the normwise relative residual of the identity,

$$\text{res} = \frac{\|\widehat{X} - A\|}{\|A\|},$$

can be. Note first that, by the chain rule for Fréchet derivatives [Higham 2008, Theorem 3.4],

$$L_f\big(g(A), L_g(A, E_1)\big) = E_1.$$

The residual of the identity, $R = \widehat{X} - A$, satisfies

$$\begin{aligned} R &= f(g(A + E_1) + E_2) - A \\ &= f(g(A) + L_g(A, E_1) + o(\|E_1\|) + E_2) - A \\ &= L_f(g(A), L_g(A, E_1)) + L_f(g(A), E_2) + o(\|E_1\|) + o(\|E_2\|) + o(\|L_g(A, E_1)\|) \\ &= E_1 + L_f(g(A), E_2) + o(\|E_1\|) + o(\|E_2\|) + o(\|L_g(A, E_1)\|), \end{aligned} \tag{3.11}$$

where L_f and L_g are the Fréchet derivatives of f and g, respectively.

We would like to know more about the $o(\|\cdot\|)$ terms in Equation (2.6) and hence in Equation (3.11). Al-Mohy and Higham [2010, Theorem 1] show that, if $f : U \to \mathbb{C}$ has a power series expansion, where U is an open subset of $\mathbb{C}$, then for $\alpha \in \mathbb{C}$ and $E \in \mathbb{C}^{n\times n}$ such that the eigenvalues of $A + \alpha E$ lie within the radius of convergence of the power series,

$$f(A + \alpha E) = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!} G_f^{(k)}(A, E), \tag{3.12}$$

where the kth Gâteaux derivative in the direction E, $G_f^{(k)}(A, E)$, is given by

$$G_f^{(k)}(A, E) = \frac{d^k}{dt^k} f(A + tE)\bigg|_{t=0}.$$

Note that, under our assumption on f, the Fréchet derivative in the direction E, $L_f(A, E)$, is equal to the Gâteaux derivative in the direction E, $G_f^{(1)}(A, E)$ [Higham 2008, Section 3.2; Nashed 1966, Section 8, Remark. 3]. This expansion shows that the $o(\|\cdot\|)$ term in Equation (2.6), hence every such term in Equation (3.11), is in fact $O(\|\cdot\|^2) = O(\epsilon^2)$. Therefore, we can drop these terms and obtain a first-order bound with error $O(\epsilon^2)$. We note that the constants in the dropped terms depend on second- and higher-order Gâteaux derivatives. Dropping these terms may not be valid if the higher-order Gâteaux derivatives are significantly larger in norm than the first Gâteaux derivative.

The relative residual can therefore be approximated by

$$\text{res} := \frac{\|R\|}{\|A\|} \approx \frac{\|E_1 + L_f(g(A), E_2)\|}{\|A\|}. \tag{3.13}$$

The maximum possible value that the relative residual can take under the assumptions (Equation (3.10)) is

$$\text{res}_{\max} = \max_{\substack{\|E_1\| \le \epsilon \|A\| \\ \|E_2\| \le \epsilon \|g(A)\|}} \frac{\|E_1 + L_f(g(A), E_2)\|}{\|A\|}. \tag{3.14}$$

Since E_1 and E_2 are independent and arbitrary, we have that

$$\begin{aligned}\mathrm{res}_{\max} &= \epsilon + \frac{\epsilon\|g(A)\|}{\|A\|} \max_{\|E_2\|\le 1} \|L_f(g(A), E_2)\| \\ &= \epsilon(1 + \mathrm{cond}(f, g(A))), \end{aligned} \tag{3.15}$$

using Equations (2.2) and (2.7).

Our test compares the relative residual res with $\mathrm{res}_{\max}$ in the 1-norm. To do so, we need to estimate $\mathrm{cond}(f, g(A))$, which can be done by using Algorithm 2.2 to estimate $\|L_f(g(A))\|_1$.

As mentioned in Section 1, the residual of the identity $(A^{1/p})^p = A$ is often used to test algorithms for principal matrix pth roots $A^{1/p}$, where p is a positive integer. Guo and Higham [2006] give what is essentially a specialized version of the analysis of this section with $E_1 = 0$, in order to derive an appropriate residual test for algorithms for computing pth roots.

We note that, for the identity $f(g(A)) = I$, the term $L_f(g(A), L_g(A, E_1)) \equiv L_{f\circ g}(A, E_1)$ vanishes. Hence, there is no E_1 term in Equation (3.11); thus, Equation (3.15) becomes $\mathrm{res}_{\max} = \epsilon\,\mathrm{cond}(f, g(A))$.

4. PRODUCT OF MATRIX FUNCTIONS AND COSINE–SINE IDENTITY

We now suppose that the product of two matrix functions $f(A)g(A)$ is known *a priori*, for example, $e^A e^{-A} = I$ or $A^{2/3}A^{1/3} = A$. Assume that the computed product $\widehat{X}$ of $f(A)g(A)$ is backward stable in the sense that

$$\widehat{X} = f(A + E_1)g(A + E_2), \tag{4.16}$$

$$\|E_1\| \le \epsilon\|A\|, \quad \|E_2\| \le \epsilon\|A\|. \tag{4.17}$$

The residual is then given by

$$\begin{aligned} R &= \widehat{X} - f(A)g(A) \\ &= f(A + E_1)g(A + E_2) - f(A)g(A) \\ &= L_f(A, E_1)g(A) + f(A)L_g(A, E_2) + L_f(A, E_1)L_g(A, E_2) + o(\|E_1\|) + o(\|E_2\|). \end{aligned} \tag{4.18}$$

The term $L_f(A, E_1)L_g(A, E_2)$ is of second order, as are the $o(\cdot)$ terms in view of Equation (3.12). The maximum possible value that the relative residual $\mathrm{res} = \|R\|/\|f(A)g(A)\|$ can take subject to Equation (4.17) is, to first order,

$$\mathrm{res}_{\max} = \epsilon \frac{\|A\|}{\|f(A)g(A)\|} \max_{\substack{\|E_1\|\le 1\\ \|E_2\|\le 1}} \|L_f(A, E_1)g(A) + f(A)L_g(A, E_2)\|. \tag{4.19}$$

We now define the linear operator $L_{\mathrm{prod}}(A, \cdot) : \mathbb{C}^{n\times 2n} \to \mathbb{C}^{n\times n}$, where

$$L_{\mathrm{prod}}(A, E) = L_f(A, E_1)g(A) + f(A)L_g(A, E_2), \tag{4.20}$$

with $E = [E_1, E_2] \in \mathbb{C}^{n\times 2n}$. The 1-norm of $L_{\mathrm{prod}}(A)$ is

$$\|L_{\mathrm{prod}}(A)\|_1 = \max_{\|E\|_1\le 1} \|L_{\mathrm{prod}}(A, E)\|_1 = \max_{\substack{\|E_1\|_1\le 1\\ \|E_2\|_1\le 1}} \|L_f(A, E_1)g(A) + f(A)L_g(A, E_2)\|_1.$$

Hence, in the 1-norm,

$$\mathrm{res}_{\max} = \frac{\epsilon\|A\|_1}{\|f(A)g(A)\|_1} \|L_{\mathrm{prod}}(A)\|_1.$$

We can estimate $\|L_{\mathrm{prod}}(A)\|_1$ to within a factor $2n$ (in view of Lemma 2.1) by applying Algorithm 2.2, using the fact, proved in the appendix, that the adjoint of L_{prod} is given

by

$$L^\star_{\text{prod}}(A, Y) = [L^\star_f(A, Yg(A)^*), L^\star_g(A,\ f(A)^*Y)].$$

We indicate how the analysis can be carried out for the identity $\sin^2 A + \cos^2 A = I$, which is useful for testing algorithms for the matrix sine and cosine. We assume that we have a backward stable computed $\widehat{X}$ such that

$$\begin{aligned}\widehat{X} &= \sin^2(A + E_1) + \cos^2(A + E_2),\\ &\|E_1\| \le \epsilon\|A\|, \quad \|E_2\| \le \epsilon\|A\|.\end{aligned}$$

Then, the residual of the identity is, neglecting higher-order terms in E_1 and E_2,

$$\begin{aligned}R &= \sin^2(A + E_1) + \cos^2(A + E_2) - I\\ &\approx (\sin A)L_s(A, E_1) + L_s(A, E_1)\sin A + (\cos A)L_c(A, E_2) + L_c(A, E_2)\cos A\\ &\equiv L_{sc}(A, E),\end{aligned} \tag{4.21}$$

where L_s and L_c are the Fréchet derivatives of the matrix sine and cosine, respectively. For the linear operator $L_{sc}(A) : \mathbb{C}^{n\times 2n} \to \mathbb{C}^{n\times n}$ defined by Equation (4.21), the maximum value of $\|R\|_1$ is

$$\text{res}_{\max} = \epsilon\|A\|_1\|L_{sc}(A)\|_1.$$

To apply Algorithm 2.2, we need the fact, proved in the appendix, that

$$L^\star_{sc}(A, Y) = \left[L^\star_s\left(A, Y\sin A^* + (\sin A^*)Y\right), L^\star_c\left(A, Y\cos A^* + (\cos A^*)Y\right)\right]. \tag{4.22}$$

When applying Algorithm 2.2 to L_{prod} or L_{sc}, the cost is dominated by the evaluation of Fréchet derivatives. The block 1-norm estimator of Higham and Tisseur [2000] requires the evaluation of about 8 Fréchet derivatives in total.

5. OBTAINING BACKWARD ERRORS FROM THE RESIDUALS

The tests developed in Sections 3 and 4 are able to show only that a computation is behaving in a way consistent with backward stability, because a suitably small relative residual (to first order) is a necessary condition but not a sufficient condition for backward stability. It is desirable to be able to obtain (or at least estimate) the backward errors in the matrix function computations directly from the residuals in the identities.

For the composition of functions, the backward error of a computed $\widehat{X} \approx f(g(A))$ is

$$\eta_{cp}(\widehat{X}) = \min_{E_1, E_2}\left\{\max\left(\frac{\|E_1\|}{\|A\|}, \frac{\|E_2\|}{\|g(A)\|}\right) : \widehat{X} = f\left(g(A + E_1) + E_2\right)\right\}.$$

Note that, by setting $E_2 = 0$, we recover the standard definition of backward error for the matrix function $h = f \circ g$, but this does not reflect the independent evaluations of f and g assumed here. Assuming that $f(g(A)) = A$, by Equation (3.11) the constraints can be rewritten in terms of the residual $R_{cp} = \widehat{X} - A$ as

$$R_{cp} = f(g(A + E_1) + E_2) - A \approx E_1 + L_f(g(A), E_2) =: L_{cp}(A, E),$$

where $E = [E_1, E_2] \in \mathbb{C}^{n\times 2n}$ and the approximation is correct to first order. Applying the vec operator yields

$$\text{vec}(R_{cp}) = \text{vec}(L_{cp}(A, E)) = K_{cp}(A)\,\text{vec}(E).$$

This equation can be rewritten as follows:

$$\widetilde{K}_{cp}\widetilde{E} \equiv K_{cp}(A)D \cdot D^{-1}\,\text{vec}(E) = \text{vec}(R_{cp}), \qquad D = \text{diag}(\|A\|I, \|g(A)\|I), \tag{5.23}$$

and the minimum norm solution to this underdetermined system, which we will refer to as the linearized normwise relative backward error, approximates $\eta_{cp}(\widehat{X})$. Note that for the identity $f(g(A)) = I$ there is no E_1 term in Equation (3.11); hence, $L_{cp}(A, E)$ reduces to $L_f(g(A), E_2)$.

Analogous formulas to Equation (5.23) hold for the product of functions and the sine and cosine identity, though the scaling matrix D is not needed in these cases since the backward error matrices E_1 and E_2 both perturb A. In each case, the $n^2 \times 2n^2$ Kronecker matrix can be computed explicitly using the following algorithm, which is very similar to Higham [2008, Algorithm 3.17].

ALGORITHM 5.1. *Given $A \in \mathbb{C}^{n\times n}$ and functions f and g together with their Fréchet derivatives, this algorithm computes $K_{cp}(A)$, $K_{\text{prod}}(A)$, or $K_{sc}(A)$.*

```
1  for j = 1 : 2n
2      for i = 1 : n
3          Z = e_i e_j^T ∈ R^{n×2n} (zero apart from 1 in the (i, j) position).
4          Compute Y = L_x(A, Z) %. The subscript x denotes cp, prod, or sc.
5          K_x(:, (j − 1)n + i) = vec(Y)
6      end
7  end
```

Cost: $O(n^5)$ flops assuming that evaluation of f, g, and their Fréchet derivatives costs $O(n^3)$ flops.

The overall cost of computing the backward error estimate is $O(n^6)$ flops if we explicitly form the Kronecker matrices and then find the minimum 2-norm solution by QR factorization. This clearly limits n. Higham and Relton [2014a, 2014b] have investigated evaluating Kronecker matrices within the context of higher-order Fréchet derivatives. However, it is not clear how to take advantage of any structure in the Kronecker matrices used here when using a direct method to solve the problem.

Instead, we can find the minimum 2-norm solution to Equation (5.23) using any method that requires only matrix–vector products with $\widetilde{K}_{cp}$. Indeed, if $\text{vec}(Z) = Dx$, then

$$\widetilde{K}_{cp}x = K_{cp}(A)Dx = \text{vec}(L_{cp}(A, Z)),$$

and $L_{cp}(A, Z)$ can be evaluated in $O(n^3)$ flops. Similarly, matrix–vector products $\widetilde{K}_{cp}^* x = DK_{cp}(A)^* x$ can be evaluated using Equation (2.5). Therefore, Krylov subspace methods such as LSQR [Paige and Saunders 1982] or the more recent LSMR [Fong and Saunders 2011] can be used at a cost of $O(n^3)$ flops per iteration and $O(n^2)$ storage. Each iteration of LSMR or LSQR requires two matrix–vector products: one with the original matrix and one with its conjugate transpose.

6. NUMERICAL EXPERIMENTS

The numerical tests described in this section use both random matrices and specific matrices selected from the literature that are known to cause difficulties for certain matrix function algorithms. In each experiment, we compare the relative residual in the matrix function identity, res, with the estimated $\text{res}_{\max}$, computed using $\epsilon = u$, the unit roundoff in double precision. We also evaluate the linearized backward error of Section 5, denoted η, which should be of order u if the algorithms are performing stably. The purpose of the experiments is to see whether our tests can reliably detect whether methods are behaving stably or unstably. All computations were carried out in MATLAB R2013a.

Unless otherwise stated, the matrix functions and Fréchet derivatives were computed using the following algorithms:

Table I. Results for 10×10 Random Matrices with Elements from Uniform Distribution on $[0, 1)$

Identity	Algorithms	$\text{res}_{\max}$	$\text{res}/\text{res}_{\max}$	η	η/η_{est}
$e^{\log A} = A$	(Inv) scale and square	6.9e-14	0.19	2.4e-15	1.0020
$(A^{0.2})^5 = A$	Schur-Padé	1.6e-14	0.68	4.6e-15	1.0000
$e^A e^{-A} = I$	Scaling-squaring	3.4e-12	0.10	1.3e-14	1.0002
$A^{2/3}A^{1/3} = A$	Schur-Padé	4.8e-11	0.24	1.3e-14	1.0040
$\sin^2 A + \cos^2 A = I$	Schur-Parlett	2.9e-14	1.05	6.3e-14	1.0300

Note: The maximum values over 100 test matrices are displayed. A convergence parameter of 1e-4 was used for the LSMR algorithm.

—*Matrix exponential:* scaling and squaring algorithms from Al-Mohy and Higham [2009a, 2009b]
—*Matrix logarithm:* inverse scaling and squaring algorithms from Al-Mohy and Higham [2012] and Al-Mohy et al. [2013]
—*Real matrix powers:* Schur–Padé algorithm [Higham and Lin 2013]
—*General matrix functions:* Schur–Parlett algorithm [Davies and Higham 2003]

The latter algorithm is implemented in the MATLAB function `funm` and explicit links to codes for the other algorithms are given in Higham and Deadman [2014]. The first two algorithms are abbreviated in the tables as "(Inv) scale and square"

Fréchet derivatives of trigonometric matrix functions were evaluated using either finite differences or the complex step approximation [Al-Mohy and Higham 2010].

Experiment 1: *Random Matrices*. We generated 100 10×10 random matrices with elements from the uniform distribution on $[0, 1)$. Matrix function algorithms would be expected to perform stably with such relatively well-behaved matrices. Various identities were tested involving the composition and product of matrix functions. For the fractional powers, the random matrices were squared if necessary to remove negative eigenvalues.

For each matrix and identity we computed res, the relative residual of the computed matrix, $\text{res}_{\max}$, the largest relative residual consistent with a backward stable computation, and the linearized backward error η (using Algorithm 5.1 and QR factorization [Golub and Van Loan 2013, Algorithm 5.6.2]). An estimate η_{est} of η was also computed using LSMR with a convergence tolerance of 1e-4, which ensured in this experiment that η_{est} was within an order of magnitude of η (the choice of tolerance is investigated further in subsequent experiments). The results are summarized in Table I, which shows the largest values encountered over the 100 test cases. Both testing methods indicate that the algorithms have performed stably. LSMR typically required between 10 and 150 iterations.

Experiment 2: *The Chebyshev and Forsythe Matrices*. The 10×10 Chebyshev spectral differentiation matrix and the 10×10 Forsythe matrix (available in MATLAB as `gallery('chebspec',10)` and `gallery('forsythe',10)`) are known to be difficult test matrices for the Schur-Parlett algorithm because of their eigenvalue distributions [Davies and Higham 2003]. The (inverse) scaling and squaring algorithms are not affected by the eigenvalue distribution. Note that the identity $\log(e^A) = A$ is valid provided that the matrix unwinding function [Aprahamian and Higham 2014] vanishes, which is the case here. The results are shown in Tables II and III. In the tables "it" is the number of iterations for LSMR with a convergence parameter tol to produce η_{est} agreeing with η to one significant figure.

In each of the tests, use of the Schur–Parlett algorithm led to a relative residual considerably larger than $\text{res}_{\max}$ and backward errors η significantly larger than u. Conversely, for the other algorithms, the relative residual did not exceed $\text{res}_{\max}$ and the

Table II. Results for 10×10 Chebyshev Matrix

Identity	Algorithms	res	$\text{res}_{\max}$	η	η_{est}	tol	it
	Schur-Parlett	4.1e-4	5.7e-5	2.0e-7	1.8e-7	1e-12	10621
$\log(e^A) = A$	(Inv) scale and square	2.6e-7	5.7e-5	1.8e-15	1.5e-15	1e-12	16226
	Schur-Parlett	1.4e-3	1.9e-5	1.7e-4		test failed	
$e^A e^{-A} = I$	Scaling-squaring	3.6e-7	1.9e-5	1.5e-12		test failed	
$\sin^2 A + \cos^2 A = I$	Schur-Parlett	6.5e-3	1.3e-8	4.7e-5		test failed	

Table III. Results for 10×10 Forsythe Matrix

Identity	Algorithms	res	$\text{res}_{\max}$	η	η_{est}	tol	it
	Schur-Parlett	1.1e-9	1.8e-14	3.4e-10	3.2e-10	1e-2	7
$\log(e^A) = A$	(Inv) scale and square	8.2e-15	1.8e-14	3.9e-15	3.8e-15	1e-2	8
	Schur-Parlett	7.7e-11	7.1e-15	2.5e-11	2.0e-11	1e-1	3
$e^A e^{-A} = I$	Scaling-squaring	2.2e-15	7.1e-15	4.5e-16	4.4e-16	1e-3	11
$\sin^2 A + \cos^2 A = I$	Schur-Parlett	3.1e-10	8.0e-15	4.3e-6	3.6e-6	1e-8	825

backward errors were considerably closer to u. As a check that our first-order analysis is correctly describing the behavior, we computed backward errors of all the constituent function evaluations using high-precision arithmetic; the results were entirely consistent with the residual and res and η values. For example, for the exponential of the Forsythe matrix, the Schur-Parlett algorithm gave a normwise relative backward error of 3.0×10^{-10}, whereas the scaling and squaring algorithm gave a normwise relative backward error of 1.7×10^{-15}.

For each test, we attempted to find the largest tolerance for LSMR such that η_{est} agreed with η to one significant figure. When the identities $e^A e^{-A} = I$ and $\sin^2 A + \cos^2 A = I$ were tested with the Chebyshev matrix, we were unable to find such a tolerance, allowing up to 20,000 iterations. Note that the condition numbers of the corresponding Kronecker matrices were of the order 10^{14}.

We conclude that the relative residual check and the backward error computation are able to detect that the Schur-Parlett algorithm did not perform in a backward stable manner and to reveal a clear distinction with the (inverse) scaling and squaring algorithms, which did perform in a backward stable manner. However, LSMR was not always able to return reliable backward error estimates.

Experiment 3: *Large Matrices and the Convergence of Iterative Methods*. The rate of convergence of Krylov subspace methods can potentially be improved by using a preconditioner. Without any information about the Kronecker matrix, finding a preconditioner prior to the solution of the least-squares problem is, in general, not possible. However Baglama et al. [2013] have devised an augmented LSQR method, ALSQR, which uses approximations of singular vectors, computed in the initial iterations, to augment the Krylov subspaces and improve convergence.

To compare LSMR and ALSQR, a 100×100 matrix with elements from the uniform distribution on $[0, 1)$ was used. The identity $\sin^2 A + \cos^2 A = I$ was tested; we found $\text{res} = 3.2 \times 10^{-12}$ and $\text{res}_{\max} = 7.1 \times 10^{-11}$, consistent with backward stable behavior. On a 2.8GHz Intel Core i7 MacBook Pro, computing $\text{res}_{\max}$ took 2.22s. For both LSMR and ALSQR, we recorded η_{est} and the time taken to compute it for various choices of tolerance. The speed of the methods is dependent on the machine architecture, thus, a better measure of performance is the number of matrix–vector multiplications mul involving the Kronecker matrix.

We repeated the experiment using the identity $e^A e^{-A} = I$. We found that $\text{res} = 5.8 \times 10^7$ and $\text{res}_{\max} = 1.4 \times 10^{10}$ (the large values are due to A having a spectral radius ≈ 50). Computing $\text{res}_{\max}$ took 1.84s.

Table IV. Results from Using LSMR and ALSQR to Compute η_{est} for a Random 100 × 100 Matrix with Different Choices of Tolerance tol

	LSMR			ALSQR		
tol	η_{est}	mul	t_η/t_{res}	η_{est}	mul	t_η/t_{res}
1e-1	6.6e-17	7	0.97	1.1e-15	79	11.4
1e-2	7.5e-16	45	6.57	2.9e-15	567	81.6
1e-3	1.5e-15	211	29.9	1.1e-14	6955	1004
1e-4	2.9e-15	911	129.7	2.3e-14	63829	9366
1e-5	5.9e-15	4007	549.2	–	–	–

(a) Results for the identity $\sin^2 A + \cos^2 A = I$

	LSMR			ALSQR		
tol	η_{est}	mul	t_η/t_{res}	η_{est}	mul	t_η/t_{res}
1e-1	1.2e-16	7	0.88	1.5e-16	9	1.22
1e-2	2.2e-16	19	2.35	1.0e-15	49	6.12
1e-3	1.1e-15	73	9.21	3.2e-15	97	11.9
1e-4	2.9e-15	207	25.8	5.2e-15	147	18.6
1e-5	4.8e-15	471	57.9	5.5e-15	173	21.2

(b) Results for the identity $e^A e^{-A} = I$

Note: The number of matrix–vector products is given by mul and t_η/t_{res} denotes the time taken to compute η_{est} divided by the time taken to compute $\text{res}_{\max}$.

The results from the experiments are shown in Table IV. Perusal of the η_{est} values suggests that using LSMR, a tolerance of 1e-3 is sufficient to obtain order-of-magnitude estimates of η. To obtain estimates accurate to one significant figure, tolerances smaller than 1e-5 are required. For $e^A e^{-A} = I$, ALSQR converges more quickly than LMSR at the most stringent tolerances. However, for the identity $\sin^2 A + \cos^2 A = I$, ALSQR performs poorly: as the tolerance was decreased, we were unable to obtain a value of η_{est} in a reasonable time.

For small tol, both iterative methods are far more expensive than evaluating $\text{res}_{\max}$. For example, a single iteration of LSMR requires the computation of four Fréchet derivatives of the underlying matrix functions (since L_{prod} and L_{sc} both involve two such derivatives), and hundreds of iterations may be required. In comparison, evaluating $\text{res}_{\max}$ requires about 16 Fréchet derivative evaluations.

Experiment 4: *Matrices Prone to Overscaling*. This first matrix in this experiment provides an example in which the use of identities fails to reveal instability. The matrix

$$B = \begin{bmatrix} 1 & 10^8 \\ 0 & -1 \end{bmatrix}$$

was found in Al-Mohy and Higham [2009b] to cause overscaling in the scaling and squaring algorithm of Higham [2005, 2009] (implemented in MATLAB R2013a as `expm`), causing a loss of accuracy (particularly in the (1,2) element) compared with the Schur-Parlett algorithm and the improved scaling and squaring algorithm of Al-Mohy and Higham [2009b].

We tested the identity $e^B e^{-B} = I$. For both of the scaling and squaring algorithms, the relative residual and the estimated relative backward errors were all several orders of magnitude smaller than u. Following the discussion in Al-Mohy and Higham [2009b], it can be seen that the computed e^{-B} contains the same error in the $(1, 2)$ element as the computed e^B, and the (1,1) and (2,2) elements are interchanged. As a result, even if there is a large error in computing e^B, the computed value of e^{-B} remains very close

to the inverse of e^B. On computation of $e^B e^{-B}$, these errors effectively cancel out. Thus, for the matrix B, the identity $e^B e^{-B}$ is not a good test.

A similar phenomenon is possible with the identity $e^{\log A} = A$: an error of the form $2k\pi i I$ in the computed $\log A$ is removed by the exponentiation, although, of course, errors of this precise form are unlikely.

The matrix

$$C = \begin{bmatrix} 0 & 1 \times 10^{-8} & 0 \\ -(6 \times 10^{10} + 2 \times 10^{8})/3 & -3 & 2 \times 10^{10} \\ 200/3 & 0 & -200/3 \end{bmatrix}$$

was discussed in a blog post by Moler [2012]. It is an extreme example of a matrix that causes overscaling and loss of accuracy in the scaling and squaring algorithm of Higham [2005, 2009], but not in the improved scaling and squaring algorithm [Al-Mohy and Higham 2009b].

Using the improved scaling and squaring algorithm, the relative residual in the identity $e^C e^{-C} = I$ is 4.1×10^{21}, with $\mathrm{res}_{\max} = 2.8 \times 10^{36}$, and the relative backward error is 5.8×10^{-12}. Using `expm`, the relative residual is 3.0×10^{64} and the relative backward error is 4.7×10^{2}. Our methods correctly identify the preferred algorithm, even though $\mathrm{cond}(\exp, C) \approx 3.1 \times 10^{18}$, so that the validity of the first order bounds is questionable. Note that the large relative residuals are due to the small norm in the denominator of (4.19) compared with the terms in the numerator.

7. CONCLUDING REMARKS

We have proposed two methods for testing matrix function algorithms based on evaluating residuals in matrix function identities. In the first method, a maximum residual consistent with backward stability is computed. In the second method, a backward error for the matrix function evaluations is obtained by solving an underdetermined linear system. The methods are based on a linearized analysis, and both require only that the matrix functions and their Fréchet derivatives can be evaluated. In numerical experiments, both methods are able to distinguish between algorithms that are behaving stably and those that are not.

The methods have two intrinsic limitations. First, they cannot attribute unstable behavior to any particular matrix function evaluation in the test identity. Second, errors in the constituent function evaluations can cancel, as illustrated by Experiment 4, thus instability can fail to be detected, though such behavior can be expected to occur only for very special test matrices.

Computing the maximum residual in our first method requires about 8 Fréchet derivative evaluations for the identities considered here, with a cost of $O(n^3)$ flops. (In this count of 8, "one Fréchet derivative evaluation" involves the evaluation of the Fréchet derivatives of one or two underlying matrix functions, depending on the identity). Explicit solution of the linear system in our second method requires $O(n^6)$ flops, which severely restricts n. For large n, an iterative solver such as LSMR or ALSQR can be used, with each iteration involving the computation of 2 Fréchet derivatives costing $O(n^3)$ flops, but numerical experiments suggest that the convergence can be very slow.

This work is of particular benefit for testing implementations of matrix function algorithms for numerical software libraries. The use of identities avoids the need for the computation or storage of reference solutions and, in particular, does not require high-precision arithmetic. Our analysis quantifies the maximum size of a residual that is consistent with stable behavior and so provides a sound basis for choosing the tolerances in the tests.

APPENDIX

A. DERIVATION OF ADJOINTS

We first derive the adjoint $L_{\text{prod}}^{\star}(A) : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times 2n}$ of the linear operator $L_{\text{prod}}(A) : \mathbb{C}^{n\times 2n} \to \mathbb{C}^{n\times n}$ defined by

$$L_{\text{prod}}(A, E) = L_f(A, E_1)g(A) + f(A)L_g(A, E_2),$$

where $E = [E_1, E_2] \in \mathbb{C}^{n\times 2n}$. We have that

$$\begin{aligned}
\langle E, L_{\text{prod}}^{\star}(A, Y)\rangle &= \langle L_{\text{prod}}(A, E), Y\rangle \quad \text{by Equation (2.4)} \\
&= \langle L_f(A, E_1)g(A), Y\rangle + \langle f(A)L_g(A, E_2), Y\rangle \\
&= \langle L_f(A, E_1), Yg(A)^*\rangle + \langle L_g(A, E_2),\ f(A)^*Y\rangle \\
&= \langle E_1, L_f^{\star}(A, Yg(A)^*)\rangle + \langle E_2, L_g^{\star}(A,\ f(A)^*Y)\rangle.
\end{aligned}$$

Since this is true for all E, we conclude that

$$L_{\text{prod}}^{\star}(A, Y) = [L_f^{\star}(A, Yg(A)^*), L_g^{\star}(A,\ f(A)^*Y)].$$

For the trigonometric identity $\sin^2 A + \cos^2 A = I$, the same method can be applied to Equation (4.21). We find that

$$\begin{aligned}
\langle E, L_{sc}^{\star}(A, Y)\rangle &= \langle L_{sc}(A, E), Y\rangle \\
&= \langle (\sin A)L_s(A, E_1), Y\rangle + \langle L_s(A, E_1)\sin A, Y\rangle + \langle (\cos A)L_c(A, E_2), Y\rangle \\
&\quad + \langle L_c(A, E_2)\cos A, Y\rangle \\
&= \langle E_1, L_s^{\star}(A, (\sin A^*)Y)\rangle + \langle E_1, L_s^{\star}(A, Y\sin A^*)\rangle \\
&\quad + \langle E_2, L_c^{\star}(A, (\cos A^*)Y)\rangle + \langle E_2, L_c^{\star}(A, Y\cos A^*)\rangle.
\end{aligned}$$

We conclude that

$$L_{sc}^{\star}(A, Y) = [L_s^{\star}(A, Y\sin A^* + (\sin A^*)Y), L_c^{\star}(A, Y\cos A^* + (\cos A^*)Y)].$$

ACKNOWLEDGMENTS

We thank Jennifer Pestana for her advice on the iterative solution of the underdetermined linear system (5.23).

REFERENCES

Awad H. Al-Mohy and Nicholas J. Higham. 2009a. Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM Journal of Matrix Analysis and Applications* 30, 4, 1639–1657. DOI:http://dx.doi.org/10.1137/080716426

Awad H. Al-Mohy and Nicholas J. Higham. 2009b. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal of Matrix Analysis and Applications* 31, 3, 970–989. DOI:http://dx.doi.org/10.1137/09074721X

Awad H. Al-Mohy and Nicholas J. Higham. 2010. The complex step approximation to the Fréchet derivative of a matrix function. *Numerical Algorithms* 53, 1, 133–148. DOI:http://dx.doi.org/10.1007/s11075-009-9323-y

Awad H. Al-Mohy and Nicholas J. Higham. 2012. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM Journal on Scientific Computing* 34, 4, C153–C169. DOI:http://dx.doi.org/10.1137/110852553

Awad H. Al-Mohy, Nicholas J. Higham, and Samuel D. Relton. 2013. Computing the Fréchet derivative of the matrix logarithm and estimating the condition number. *SIAM Journal on Scientific Computing* 35, 4, C394–C410. DOI:http://dx.doi.org/10.1137/120885991

Mary Aprahamian and Nicholas J. Higham. 2014. The matrix unwinding function, with an application to computing the matrix exponential. *SIAM Journal on Matrix Analysis and Applications* 35, 1, 88–109. DOI:http://dx.doi.org/10.1137/130920137

James Baglama, Lothar Reichel, and Daniel Richmond. 2013. An augmented LSQR method. *Numerical Algorithms* 64, 2, 263–293. DOI:http://dx.doi.org/10.1007/s11075-012-9665-8

W. J. Cody. 1993. Algorithm 714. CELEFUNT: A portable test package for complex elementary functions. *ACM Transactions on Mathematical Software* 19, 1, 1–21.

Philip I. Davies and Nicholas J. Higham. 2003. A Schur-Parlett algorithm for computing matrix functions. *SIAM Journal on Matrix Analysis and Applications* 25, 2, 464–485. DOI:http://dx.doi.org/10.1137/S0895479802410815

Eugene D. Denman and Alex N. Beavers, Jr. 1976. The matrix sign function and computations in systems. *Applied Mathematics and Computation* 2, 1, 63-94. DOI:http://dx.doi.org/10.1016/0096-3003(76)90020-5

Luca Dieci, Benedetta Morini, and Alessandra Papini. 1996. Computational techniques for real logarithms of matrices. *SIAM Journal on Matrix Analysis and Applications* 17, 3, 570–593.

David Chin-Lung Fong and Michael Saunders. 2011. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing* 33, 5, 2950–2971. DOI:http://dx.doi.org/10.1137/10079687X

Gene H. Golub and Charles F. Van Loan. 2013. *Matrix Computations* (4th ed.). Johns Hopkins University Press, Baltimore, MD. xxi+756 pages.

Federico Greco and Bruno Iannazzo. 2010. A binary powering schur algorithm for computing primary matrix roots. *Numerical Algorithms* 55, 1 (January 2010), 59–78.

Chun-Hua Guo and Nicholas J. Higham. 2006. A Schur-Newton method for the matrix *p*th root and its inverse. *SIAM Journal on Matrix Analysis and Application* 28, 3, 788–804. DOI:http://dx.doi.org/10.1137/050643374

Nicholas J. Higham. 2008. The Matrix Function Toolbox. Retrieved December 29, 2015 from http://www.ma.man.ac.uk/higham/mftoolbox/.

Nicholas J. Higham. 2005. The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Application* 26, 4, 1179–1193. DOI:http://dx.doi.org/10.1137/04061101X

Nicholas J. Higham. 2008. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. xx+425 pages. DOI:http://dx.doi.org/10.1137/1.9780898717778

Nicholas J. Higham. 2009. The scaling and squaring method for the matrix exponential revisited. *SIAM Review* 51, 4, 747–764. DOI:http://dx.doi.org/10.1137/090768539

Nicholas J. Higham and Edvin Deadman. 2014. *A Catalogue of Software for Matrix Functions. Version 1.0*. MIMS EPrint 2014.8. Manchester Institute for Mathematical Sciences, The University of Manchester, UK. 19 pages.

Nicholas J. Higham and Lijing Lin. 2011. A Schur-Padé algorithm for fractional powers of a matrix. *SIAM Journal on Matrix Analysis and Application* 32, 3, 1056–1078. DOI:http://dx.doi.org/10.1137/10081232X

Nicholas J. Higham and Lijing Lin. 2013. An improved Schur–Padé algorithm for fractional powers of a matrix and their Fréchet derivatives. *SIAM Journal on Matrix Analysis and Application* 34, 3, 1341–1360. DOI:http://dx.doi.org/10.1137/130906118

Nicholas J. Higham and Samuel D. Relton. 2014a. Estimating the condition number of the Fréchet derivative of a matrix function. *SIAM Journal on Scientific Computing* 36, 6, C617–C634. DOI:http://dx.doi.org/10.1137/130950082

Nicholas J. Higham and Samuel D. Relton. 2014b. Higher order Fréchet derivatives of matrix functions and the level-2 condition number. *SIAM Journal on Matrix Analysis and Application* 35, 3, 1019–1037. DOI:http://dx.doi.org/10.1137/130945259

Nicholas J. Higham and Françoise Tisseur. 2000. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. *SIAM Journal on Matrix Analysis and Application* 21, 4, 1185–1201. DOI:http://dx.doi.org/10.1137/S0895479899356080

Roger A. Horn and Charles R. Johnson. 1991. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK. viii+607 pages.

Bruno Iannazzo and Carlo Manasse. 2013. A Schur logarithmic algorithm for fractional powers of matrices. *SIAM Journal on Matrix Analysis and Application* 34, 2, 794–813. DOI:http://dx.doi.org/10.1137/120877398

Eric Jones, Travis Oliphant, Pearu Peterson, and others. 2001. SciPy: Open Source Scientific Tools for Python. (2001). http://www.scipy.org/.

Cleve B. Moler. 2012. A Balancing Act for the Matrix Exponential. http://blogs.mathworks.com/cleve/2012/07/23/a-balancing-act-for-the-matrix-exponential/. (July 2012).

NAG Library. NAG Ltd., Oxford. http://www.nag.co.uk.

M. Z. Nashed. 1966. Some remarks on variations and differentials. *American Mathematical Monthly* 73, 4 (1966), 63–76.

Christopher C. Paige and Michael A. Saunders. 1982. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software* 8, 1, 43–71.

Matthew I. Smith. 2003. A Schur algorithm for computing matrix pth roots. *SIAM Journal on Matrix Analysis and Application* 24, 4, 971–989.

Received March 2014; revised September 2014; accepted January 2015