Testing matrix function algorithms using identities

Deadman, Edvin and Higham, Nicholas J.

2014

MIMS EPrint: **2014.13**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Testing Matrix Function Algorithms Using Identities

EDVIN DEADMAN, The University of Manchester
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, SciPy, Python, MATLAB, NAG Library

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, Sec. 1.2]. Important functions include the exponential, the 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.

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 and 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 $Q(f_1, \dots, f_k) = 0$ 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 $\sin(2A) - 2\sin A \cos A = 0$, and

This work was supported by European Research Council Advanced Grant MATFUN (267526). The second author was also supported by Engineering and Physical Sciences Research Council grant EP/I006702/1. Author's addresses: Edvin Deadman and Nicholas J. Higham, School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK; email: edvin.deadman@manchester.ac.uk, nick.higham@manchester.ac.uk.

general results saying when scalar identities translate to the matrix case are available [Higham 2008, Thms. 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 we can solve

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

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, so 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, by using the Fréchet derivative expansions to obtain an underdetermined linear system whose minimal norm solution provides the backward error matrices.

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], and Iannazzo and Manasse [2013] have all used the identity $X^p - A = 0$ 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. 01] 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 `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] 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.

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

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

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 (2.3),

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

Maximizing over all E gives the result. □

To estimate $\|L\|_1$ we will use the following algorithm, which is essentially [Higham 2008, Alg. 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 $\langle Kx, y\rangle = \langle L(X), Y\rangle$ and 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 $\|K\|_1 \in \left[n^{-1}\|L\|_1, q\|L\|_1\right]$.*

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 $Kx \equiv \operatorname{vec}(L(X))$ and $K^*x \equiv \operatorname{vec}(L^\star(X))$,
where $\operatorname{vec}(X) = x$.

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.

A 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

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

The relative condition number $\mathrm{cond}(f,A)$ of a matrix function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ is

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

and it can be expressed in terms of the Fréchet derivative as [Higham 2008, Thm. 3.1]

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

In order to apply Algorithm 2.2 we need the adjoint of L_f. 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 $A = f(g(A))$. 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. An alternative interpretation is that E_2 is a forward error for the evaluation of g, in which case the assumption is that the evaluation of f is exact and that of g mixed forward–backward stable [Higham 2002, p. 7]. The question of interest is how large the normwise relative residual of the identity,

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

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

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

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 (2.6) and hence in (3.11). Al-Mohy and Higham [2010, Thm. 1] show that if $f : \mathbb{C} \to \mathbb{C}$ has a power series expansion then, for $\alpha \in \mathbb{C}$ and $E \in \mathbb{C}^{n\times n}$ such that $A + \alpha E$ lies in 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

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

Note that for $k = 1$, $G_f^{(k)}(A, E)$ is equal to the Fréchet derivative $L_f(A, E)$ under our assumption on f [Higham 2008, Sec, 3.2]. This expansion shows that the $o(\|\cdot\|)$ term in (2.6), and hence every such term in (3.11), is in fact $O(\|\cdot\|^2) = O(\epsilon^2)$. We can therefore 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 (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

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

using (2.2) and (2.7).

Our test compares the relative residual res with $\text{res}_{\max}$ in the 1-norm. To do so we need to estimate $\text{cond}(f, g(A))$, 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 $A^{1/p}$, were 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 $f(g(A)) = I$ there is no E_1 term in (3.11) and so (3.15) becomes $\mathrm{res}_{\max} = \epsilon \,\mathrm{cond}(f, g(A))$.

4. PRODUCT OF MATRIX FUNCTIONS AND COSINE–SINE IDENTITY

We suppose now 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 (3.12). The maximum possible value that the relative residual $\mathrm{res} = \|R\|/\|f(A)g(A)\|$ can take subject to (4.17) is, to first order,

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

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

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

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

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

Hence, in the 1-norm,

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

We can estimate $\|L_{pd}(A)\|_1$ 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_{pd} is given by

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

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

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

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 (4.21) the maximum value of $\|R\|_1$ is

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

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

$$L_{sc}^\star(A,Y) = \left[L_s^\star\big(A, Y\sin A^* + (\sin A^*)Y\big), L_c^\star\big(A, Y\cos A^* + (\cos A^*)Y\big)\right]. \tag{4.22}$$

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

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

Assuming that $f(g(A)) = A$, by (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

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

This equation can be rewritten

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

and the minimum norm solution to this underdetermined system, which we will refer to as the linearized normwise relative backward error, approximates $\eta_{cp}(\widehat{X})$.

Analogous formulas to (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, Alg. 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 $K_{cp}(A)$, $K_{pd}(A)$, or $K_{sc}(A)$.*

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

7 end

Cost: $O(n^5)$ flops assuming 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. The Kronecker matrices are highly structured [Higham and Relton 2013a], [Higham and Relton 2013b], but it is not clear how to take advantage of this structure in using a direct method to solve the problem.

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

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

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 (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.

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 $\mathrm{res}_{\max}$ and also evaluate the linearized backward error of section 5, denoted η, which should be of order the unit roundoff 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 the computations were carried out in MATLAB R2013a.

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

— matrix exponential: scaling and squaring algorithms from [Al-Mohy and Higham 2009a], [Al-Mohy and Higham 2009b],
— matrix logarithm: inverse scaling and squaring algorithms from [Al-Mohy and Higham 2012], [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 & 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, $\mathrm{res}_{\max}$, the largest relative residual consistent with a backward stable computation and the backward error η (using Algorithm 5.1 and QR factorization [Golub and Van Loan 2013, Alg. 5.6.2]). An estimate η_{est} of η was also computed using LSMR with

Table I: Results for 10×10 random matrices with elements from uniform distribution on $[0, 1)$. The maximum values over 100 test matrices are displayed. A convergence parameter of 1e-4 was used for the LSMR algorithm.

Identity	Algorithms	$\mathrm{res}_{\max}$	$\mathrm{res}/\mathrm{res}_{\max}$	η	η/η_{est}
$e^{\log A} = A$	(Inv) scale & 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

Table II: Results for 10×10 Chebyshev matrix.

Identity	Algorithms	res	$\mathrm{res}_{\max}$	η	η_{est}	tol	it
$\log(e^A) = A$	Schur-Parlett	4.1e-4	5.7e-5	2.0e-7	1.8e-7	1e-12	10621
	(Inv) scale & square	2.6e-7	5.7e-5	1.8e-15	1.5e-15	1e-12	16226
$e^A e^{-A} = I$	Schur-Parlett	1.4e-3	1.9e-5	1.7e-4		test failed	
	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	$\mathrm{res}_{\max}$	η	η_{est}	tol	it
$\log(e^A) = A$	Schur-Parlett	1.1e-9	1.8e-14	3.4e-10	3.2e-10	1e-2	7
	(Inv) scale & square	8.2e-15	1.8e-14	3.9e-15	3.8e-15	1e-2	8
$e^A e^{-A} = I$	Schur-Parlett	7.7e-11	7.1e-15	2.5e-11	2.0e-11	1e-1	3
	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

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 indicated that the algorithms are performing 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 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 $\mathrm{res}_{\max}$ and backward errors η significantly larger than u. Conversely, for the other algorithms the relative residual did not exceed $\mathrm{res}_{\max}$ and the 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, and we found $\text{res} = 2.8 \times 10^{-12}$ and $\text{res}_{\max} = 2.9 \times 10^{-13}$, consistent with backward stable behaviour. On a 2.8 GHz Intel Core i7 MacBook Pro, computing $\text{res}_{\max}$ took 2.22 seconds. 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, so 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 $\text{res} = 5.8 \times 10^7$ and $\text{res}_{\max} = 1.4 \times 10^{10}$ (the large values are due to the Perron–Frobenius eigenvalue of size approximately 50). Computing $\text{res}_{\max}$ took 1.84 seconds.

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, and hundreds of iterations may be required. In comparison, evaluating $\text{res}_{\max}$ requires fewer than 20 Fréchet derivative evaluations.

Experiment 4: Matrices Prone to Overscaling. This first matrix in this experiment provides an example where 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], Higham [2009] (implemented in MATLAB R2013a as `expm`), causing a loss of accuracy (particularly in the (1,2) element) compared

Table IV: Results from using LSMR and ALSQR to compute η_{est} for a random 100×100 matrix with different choices of tolerance tol. 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 $\text{res}_{\max}$.

	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 $e^A e^{-A} = I$

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 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, though 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 [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], Higham [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 $\text{res}_{\max} = 2.8 \times 10^{36}$, 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 $\text{cond}(\exp, C) \approx 3.1 \times 10^{18}$, 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, while in the second a backward error for the matrix function evaluations is obtained by solving an underdetermined linear system. The methods are based on a linearized analysis. 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 can not attribute unstable behaviour to any particular matrix function evaluation in the test identity. Second errors in the constituent functions evaluations can cancel, as illustrated by Experiment 4, so 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 fewer than 20 Fréchet derivative evaluations for the identities considered here, with a cost of $O(n^3)$ flops. The method requires only that the matrix functions and their Fréchet derivatives can be evaluated. 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 4 Fréchet derivatives and hence 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.

Acknowledgements

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

A. DERIVATION OF ADJOINTS

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

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

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

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

Since this is true for all E, we conclude that

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

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

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

We conclude that

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

REFERENCES

AL-MOHY, A. H. AND HIGHAM, N. J. 2009a. Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM J. Matrix Anal. Appl. 30,* 4, 1639–1657.

AL-MOHY, A. H. AND HIGHAM, N. J. 2009b. A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl. 31,* 3, 970–989.

AL-MOHY, A. H. AND HIGHAM, N. J. 2010. The complex step approximation to the Fréchet derivative of a matrix function. *Numer. Algorithms 53,* 1, 133–148.

AL-MOHY, A. H. AND HIGHAM, N. J. 2012. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM J. Sci. Comput. 34,* 4, C153–C169.

AL-MOHY, A. H., HIGHAM, N. J., AND RELTON, S. D. 2013. Computing the Fréchet derivative of the matrix logarithm and estimating the condition number. *SIAM J. Sci. Comput. 35,* 4, C394–C410.

APRAHAMIAN, M. AND HIGHAM, N. J. 2014. The matrix unwinding function, with an application to computing the matrix exponential. *SIAM J. Matrix Anal. Appl. 35,* 1, 88–109.

BAGLAMA, J., REICHEL, L., AND RICHMOND, D. 2013. An augmented LSQR method. *Numer. Algorithms 64,* 2, 263–293.

CODY, W. J. 1993. Algorithm 714. CELEFUNT: A portable test package for complex elementary functions. *ACM Trans. Math. Software 19,* 1, 1–21.

DAVIES, P. I. AND HIGHAM, N. J. 2003. A Schur–Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl. 25,* 2, 464–485.

DENMAN, E. D. AND BEAVERS, JR., A. N. 1976. The matrix sign function and computations in systems. *Appl. Math. Comput. 2,* 1, 63 – 94.

DIECI, L., MORINI, B., AND PAPINI, A. 1996. Computational techniques for real logarithms of matrices. *SIAM J. Matrix Anal. Appl. 17,* 3, 570–593.

FONG, D. C.-L. AND SAUNDERS, M. 2011. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput. 33,* 5, 2950–2971.

GOLUB, G. H. AND VAN LOAN, C. F. 2013. *Matrix Computations* Fourth Ed. Johns Hopkins University Press, Baltimore, MD, USA.

GRECO, F. AND IANNAZZO, B. 2010. A binary powering Schur algorithm for computing primary matrix roots. *Numerical Algorithms 55,* 1, 59–78.

GUO, C.-H. AND HIGHAM, N. J. 2006. A Schur–Newton method for the matrix pth root and its inverse. *SIAM J. Matrix Anal. Appl. 28,* 3, 788–804.

HIGHAM, N. J. The Matrix Function Toolbox. `http://www.maths.manchester.ac.uk/~higham/mftoolbox`.

HIGHAM, N. J. 2002. *Accuracy and Stability of Numerical Algorithms* Second Ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

HIGHAM, N. J. 2005. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl. 26,* 4, 1179–1193.

HIGHAM, N. J. 2008. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

HIGHAM, N. J. 2009. The scaling and squaring method for the matrix exponential revisited. *SIAM Rev. 51,* 4, 747–764.

HIGHAM, N. J. AND DEADMAN, E. 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. Feb.

HIGHAM, N. J. AND LIN, L. 2013. An improved Schur–Padé algorithm for fractional powers of a matrix and their Fréchet derivatives. *SIAM J. Matrix Anal. Appl. 34,* 3, 1341–1360.

HIGHAM, N. J. AND RELTON, S. D. 2013a. Estimating the condition number of the Fréchet derivative of a matrix function. MIMS EPrint 2013.84, Manchester Institute for Mathematical Sciences, The University of Manchester, UK. Dec.

HIGHAM, N. J. AND RELTON, S. D. 2013b. Higher order Fréchet derivatives of matrix functions and the level-2 condition number. MIMS EPrint 2013.58, Manchester Institute for Mathematical Sciences, The University of Manchester, UK. Nov.

HIGHAM, N. J. AND TISSEUR, F. 2000. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. *SIAM J. Matrix Anal. Appl. 21,* 4, 1185–1201.

IANNAZZO, B. AND MANASSE, C. 2013. A schur logarithmic algorithm for fractional powers of matrices. *SIAM J. Matrix Anal. Appl. 34,* 2, 794–813.

JONES, E., OLIPHANT, T., PETERSON, P., ET AL. 2001–. SciPy: Open source scientific tools for Python. `http://www.scipy.org/`.

MOLER, C. B. 2012. A balancing act for the matrix exponential. http://blogs.mathworks.com/cleve/2012/07/23/a-balancing-act-for-the-matrix-exponential/.

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

PAIGE, C. C. AND SAUNDERS, M. A. 1982. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software 8,* 1, 43–71.

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