Scaled and squared subdiagonal Padé approximation for the matrix exponential

Güttel, Stefan and Yuji, Nakatsukasa

2015

MIMS EPrint: **2015.46**

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

SCALED AND SQUARED SUBDIAGONAL PADÉ APPROXIMATION FOR THE MATRIX EXPONENTIAL

STEFAN GÜTTEL* AND YUJI NAKATSUKASA†

Abstract. The scaling and squaring method is the most widely used algorithm for computing the exponential of a square matrix A. We introduce an efficient variant that uses a much smaller squaring factor when $\|A\| \gg 1$ and a *subdiagonal* Padé approximant of low degree, thereby significantly reducing the overall cost and avoiding the potential instability caused by overscaling, while giving forward error of the same magnitude as the standard algorithm. The new algorithm performs well if a rough estimate of the rightmost eigenvalue of A is available and the rightmost eigenvalues do not have widely varying imaginary parts, and it achieves significant speedup over the conventional algorithm especially when A is of large norm. Our algorithm uses the partial fraction form to evaluate the Padé approximant, which makes it suitable for parallelization and directly applicable to computing the action of the matrix exponential $\exp(A)\boldsymbol{b}$, where $\boldsymbol{b}$ is a vector or a tall skinny matrix. For this problem the significantly smaller squaring factor has an even pronounced benefit for efficiency when evaluating the action of the Padé approximant.

Key words. matrix exponential, scaling and squaring, subdiagonal Padé approximation, stable matrix, conditioning, matrix function times a vector

AMS subject classifications. 15A23, 65F30, 65G50

1. Introduction. For a given matrix $A \in \mathbb{C}^{n\times n}$, the matrix exponential e^A is among the most important matrix functions, one reason being its close connection to ordinary differential equations; see, e.g., [42, Chapter 8]. We explore the scaling and squaring method [2, 23] for its computation, which is the most commonly used in practice, and implemented for example in MATLAB's `expm`. It is based on a type $(2^s m, 2^s m)$ rational approximation to the exponential in a region of the complex plane containing the eigenvalues $\Lambda(A)$, where s and m are suitably chosen integer parameters. The scaling and squaring method computes e^A by first choosing an integer s such that $A/2^s$ has norm of order 1, then taking a rational type (m, m) Padé approximation $r(z)$ to e^z, where m is chosen such that $e^{A/2^s} \approx r(A/2^s)$. Typically, $m = 13$ is chosen when $\|A\|_1$ exceeds about 5; see [23]. The method then computes $e^A \approx (r(A/2^s))^{2^s}$ via repeated squaring of the matrix.

Our work is motivated by the following observation. The standard choice of s roughly satisfies $2^s = O(\|A\|)$. Therefore the degree of the rational functions underlying the scaling and squaring method is $2^s m = 2^s \times 13 = O(\|A\|)$. When $\|A\| \gg 1$, this is a rational function of extremely high degree relative to the degree 16 of the best uniform rational approximation to the exponential function on the negative real axis which achieves an error of $O(u)$, where $u \approx 1.1 \times 10^{-16}$ is the unit roundoff in double precision arithmetic. The use of such best rational approximants appears to have been limited to cases where the matrix is Hermitian, or known to have negative real or nearly real eigenvalues [14, 38]; this is perhaps reasonable as it is nontrivial to compute the best rational approximant in a complex region.

In this work we investigate the high-degree rational approximant underlying the scaling and squaring method and observe that in many cases rational functions of

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

†Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan (`nakatsukasa@mist.i.u-tokyo.ac.jp`, `http://www.opt.mist.i.u-tokyo.ac.jp/~nakatsukasa`). Supported by JSPS Scientific Research Grant No. 26870149 and EPSRC grant EP/I005293/1.

much lower degree suffice for computing e^A in a forward stable manner. The key facts we are using are the following.

1. Due to the exponential decay of the exponential function, $e^z \approx 0$ if z has large negative real part. Consequently, the rational approximant $r(z) = (p_{k,m}(z)/q_{k,m}(z))^{2^s}$ is a good approximation for such z if $|p_{k,m}(z)/q_{k,m}(z)|$ is sufficiently smaller than 1 and 2^s is reasonably large. For example, with $|p_{k,m}(z)/q_{k,m}(z)| = 0.1$ and $s = 4$ we have $(p_{k,m}(z)/q_{k,m}(z))^{2^s} = 10^{-16}$, which approximates e^z to full machine precision for z with $\mathrm{Re}(z) \le \ln 10^{-16} \simeq -36.8$. We will show that this simple fact makes subdiagonal Padé approximants computationally more attractive than diagonal Padé approximants.
2. It is known [22, p. 240] that the relative condition number of e^A, defined by

$$\kappa_{\exp}(A) = \frac{\|L_{\exp}(A)\|\|A\|}{\|e^A\|}, \tag{1.1}$$

has lower and upper bounds

$$\|A\| \le \kappa_{\exp}(A) \le \frac{e^{\|A\|}\|A\|}{\|e^A\|}. \tag{1.2}$$

Here $\|\cdot\|$ denotes any consistent matrix norm and $L_{\exp}(A)$ is the Fréchet derivative of the matrix exponential at A. The lower bound $\|A\| \le \kappa_{\exp}(A)$ will be of particular interest, as it shows that a forward stable computation is allowed to have an error of $O(u\|A\|)$, and the algorithm can take advantage of this for improved efficiency.
3. Due to the low degree of the Padé approximant employed in our algorithm, its partial fraction form can be evaluated accurately without severe subtractive cancellation effects. We will also argue in section 4.1.1 that the condition numbers of the linear systems involved are under control. Moreover, the partial fraction form allows for some coarse grain parallelism in its evaluation.

Conventionally, the parameters in the scaling and squaring method are chosen to minimize cost while ensuring that the truncation error of the Padé approximant is of order unit roundoff [3, 23]. An important aspect here is that this approach takes no account of the roundoff errors in finite precision arithmetic, and indeed the stability of the method is still an open problem. The algorithm that we introduce does not resolve this issue either; however, we show that for normal matrices, for which the standard scaling and squaring method is known to be forward stable in finite precision arithmetic, our algorithm is also forward stable. Furthermore, we provide sufficient conditions under which our method is forward stable, and illustrate through experiments that the method performs stably for an even larger class of matrices.

Our implementation uses the partial fraction or product form to evaluate the Padé approximant at a matrix argument, and this allows us to easily adapt the algorithm for computing $e^A\boldsymbol{b}$ for a vector (or tall and skinny matrix) $\boldsymbol{b}$ via solving systems of linear equations. The reduced scaling parameter s then leads to an even pronounced efficiency improvement when evaluating the action of the Padé approximant.

Due to the relations with ordinary differential equations, many matrices one wants to compute the exponential of correspond to discretizations of unbounded differential operators and are of large norm and matrix size. The resulting large scaling parameter s and the high degree m of the Padé approximant render the conventional scaling and squaring method inefficient. On the other hand, specialized methods for the direct

computation of $e^A\boldsymbol{b}$ exist, most prominently, those based on polynomial or rational Krylov spaces (see, e.g., [14, 15, 17, 28, 31, 39]). However, these methods typically do not come with the same level of robustness as the conventional scaling and squaring method, at least if A is non-Hermitian, as various parameters need to be tuned to make these methods work efficiently (selection of poles, dimension of the Krylov space, restart lengths, and so on). Methods based directly on rational (best) approximation, like the Carathéodory–Fejér [33] or contour-based methods [38, 45], are applicable only if A is (nearly) Hermitian. Our method presented in this paper tries to fill a gap between these different approaches. We present an efficient method with robustness for an important class of not-necessarily Hermitian matrices arising often in practice, namely matrices whose rightmost eigenvalues have moderate imaginary parts. For example, sectorial matrices have this property. When the rightmost eigenvalues vary largely in imaginary parts, the direct application of our method may give inaccurate results. We suggest possible remedies to overcome this issue.

The structure of this paper is as follows. In section 2 we investigate scaled and squared rational approximation to the exponential. In section 3 we describe our new algorithm `sexpm`, with its name derived from the MATLAB command `expm` with the prefix `s`, which is a reminder of the four s's: subdiagonal scaling and squaring with shifting. We analyze the forward stability in section 4. Section 5 considers the computation of $e^A\boldsymbol{b}$. In section 6 we compare `sexpm` with other standard methods. Numerical experiments in section 7 illustrate the efficiency and performance of `sexpm`.

Notation. $\|A\|$ denotes any consistent matrix norm, $\kappa(A)$ is the standard 2-norm condition number, and $\kappa_{\exp}$ is the condition number for the exponential. We assume the use of IEEE double precision arithmetic with unit roundoff $u \approx 1.1 \times 10^{-16}$.

1.1. Initial shifting. It will prove useful to introduce the shift $A_\sigma = A - \sigma I$ so that the eigenvalues of A_σ are in the left half-plane. The scalar $\sigma \in \mathbb{C}$ can be obtained as an estimated rightmost eigenvalue of A. We take σ to be real if A is a real matrix to ensure the computed e^A is also real, but otherwise σ can be complex. To obtain the exponential of the original matrix we use the identity $e^A = e^\sigma e^{A_\sigma}$.

We shall make use of $\|A\|$, which is a lower bound for $\kappa_{\exp}(A)$, cf. (1.2). Hence it is important that the shift operation does not significantly increase this norm. Fortunately this is guaranteed as can be seen from $\|A_\sigma\|_2 \le \|A\|_2 + \|\sigma I\|_2 \lesssim 2\|A\|_2$, using the fact that σ is an estimate for the rightmost eigenvalue of A and $\max_i |\lambda_i(A)| \le \|A\|_2$.

We note that the idea of shifting the matrix was used by Ward [44] (who also uses balancing) with the goal of reducing the matrix norm and thereby the scaling parameter. There is a fundamental difference between the purpose of the shift, as we shift so that the rightmost eigenvalues are near the origin and $\|e^{A_\sigma}\| = O(1)$, allowing us to examine the quality of the rational approximant being used; see the next section.

Several approaches exist for estimating the rightmost eigenvalues [11, 25] but these are beyond the scope of this paper and we assume that the user provides an estimate for σ. We demonstrate in section 7.2.2 that the estimate is allowed to have error $O(2^s)$, where s is the scaling parameter. Note that in many applications where exponentials play a role, the matrices A are stable with no eigenvalues in the right half-plane, in which case shifting is not necessary anyway.

For notational simplicity in the following we will take $A \leftarrow A_\sigma$, that is, A denotes the shifted matrix with rightmost eigenvalue ≈ 0.

2. Rational approximants underlying the scaling and squaring method. The starting point of our investigation is to view the scaling and squaring method as

a global rational approximation to the exponential function. The conventional scaling and squaring method proceeds as follows:

1. Choose s so that $\|A/2^s\| \simeq O(1)$.
2. Evaluate the diagonal Padé approximant $r_{m,m}(A/2^s)$. If $\|A\| \gg 1$, $m = 13$ is chosen.
3. Obtain an approximation to e^A by repeated squaring: $e^A \approx (r_{m,m}(A/2^s))^{2^s}$.

The last formula clearly reveals the rational function that underlies the method: $e^z \approx (r_{m,m}(z/2^s))^{2^s}$, that is, e^z is approximated by a scaled and squared Padé approximant of the exponential. For any two integers k and m, the type (k, m) Padé approximant of the exponential is explicitly known to be $r_{k,m}(z) = p_{k,m}(z)/q_{k,m}(z)$ with

$$p_{k,m}(z) = \sum_{j=0}^{k} \frac{(k+m-j)!\,k!}{(k+m)!\,(k-j)!}\,\frac{z^j}{j!}, \quad q_{k,m}(z) = \sum_{j=0}^{m} \frac{(k+m-j)!\,m!}{(k+m)!\,(m-j)!}\,\frac{(-z)^j}{j!}. \tag{2.1}$$

Suppose for simplicity that A is diagonalizable and $A = X\mathrm{diag}(\lambda_i)X^{-1}$. Then the method approximates $e^A = X\mathrm{diag}(e^{\lambda_i})X^{-1}$ by $Y := X\mathrm{diag}((r_{m_1,m}(\lambda_i/2^s))^{2^s})X^{-1} \approx e^A$, and the error satisfies

$$\|e^A - Y\|_2 \le \kappa_2(X) \max_{z\in\mathcal{D}} \left|(e^z - r(z/2^s))^{2^s}\right|, \tag{2.2}$$

where $\mathcal{D}$ is a region in the complex plane containing the eigenvalues $\Lambda(A)$, and $\kappa_2(X) = \|X\|_2\|X^{-1}\|_2$ is the 2-norm condition number of the eigenvector matrix. We may take $\mathcal{D} = \Lambda(A)$, in which case $\max_i \left|e^{\lambda_i} - (r(\lambda_i/2^s))^{2^s}\right|$ represents the exact error if A is a normal matrix, and its quality as an estimate of the error may deteriorate as $\kappa_2(X)$ grows. A bound $\|e^A - Y\|_2 \le 11.08 \max_{z\in\mathcal{W}(A)} \left|e^z - (r(z/2^s))^{2^s}\right|$ involving the field of values $\mathcal{W}(A) = \{\boldsymbol{x}^*A\boldsymbol{x} : \|\boldsymbol{x}\|_2 = 1\}$ is also known [8], and it does not depend explicitly on $\kappa_2(X)$. However, we will not use this bound in what follows as it can lead to crude overestimations and $\mathcal{W}(A)$ is difficult to compute.

We now focus on the term $\max_i \left|e^{\lambda_i} - (r(\lambda_i/2^s))^{2^s}\right|$, and to do so we examine $\left|e^z - (r(z/2^s))^{2^s}\right|$ for complex arguments z. If this error is small in a region $\mathcal{D}$ containing the eigenvalues of A, then Y is a good approximation to e^A. A similar analysis is given by Fair and Luke [13], who show that essentially the same argument holds for general matrices with nontrivial Jordan blocks.

Figures 2.1–2.6 visualize $|e^z - (r_{k,m}(z/2^s))^{2^s}|$, $s = 4$, for three representative choices of Padé types (k, m):

1. $k = 3,\ m = 4$: subdiagonal Padé approximant (Figures 2.1 and 2.2),
2. $k = m = 4$: diagonal Padé approximant (Figures 2.3 and 2.4), and
3. $k = m = 13$: diagonal Padé approximant that is usually employed by the standard method (Figures 2.5 and 2.6).

The right figures show the same function $|e^z - (r_{k,m}(z/2^s))^{2^s}|$ as the left but in a larger region, and the red dots are the poles of $r_{k,m}(z)$, shown for reference only (note that these are not the poles of $r_{k,m}(z/2^s)$, which are larger by the factor 2^s).

Padé approximants with $m \le k + 2$ are known to have all their poles in the right half-plane [41]. If in addition $k \le m$ then they are known to be *A-acceptable* [43], which means the modulus $|r_{k,m}(z)|$ is bounded by 1 in the left half-plane. A-acceptable subdiagonal Padé approximants have the stronger property that they are L-acceptable [18], which means that the modulus decays to 0 as $z \to -\infty$ (which is also obvious from degree consideration).

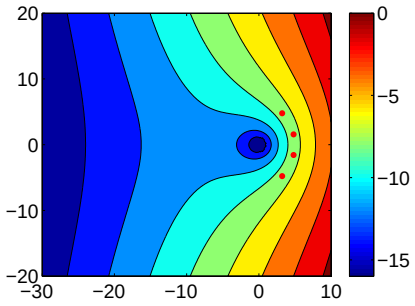

FIG. 2.1. $k = 3$, $m = 4$ *(subdiagonal Padé)*, $s = 4$.

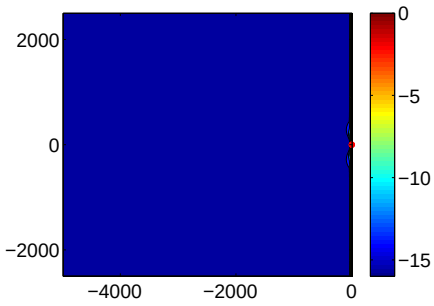

FIG. 2.2. *Larger region of left plot.*

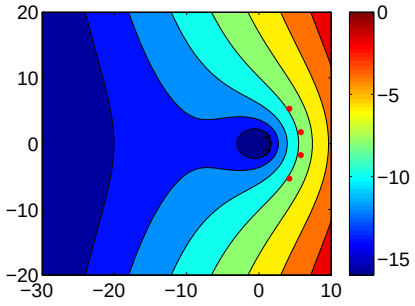

FIG. 2.3. $k = m = 4$ *(diagonal Padé)*, $s = 4$.

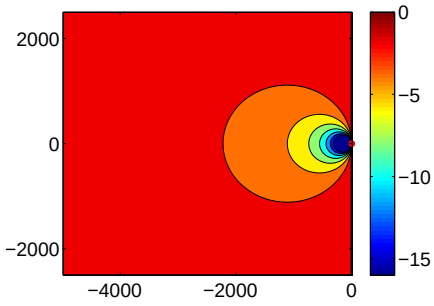

FIG. 2.4. *Larger region of left plot.*

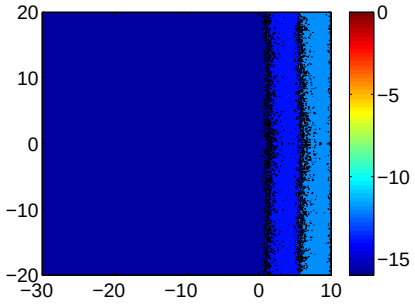

FIG. 2.5. $k = m = 13$ *(standard method)*, $s = 4$.

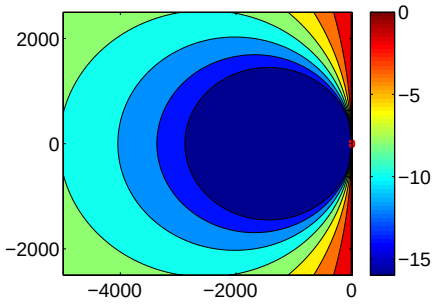

FIG. 2.6. *Larger region of left plot.*

Comparing Figures 2.1 and 2.2 with Figures 2.3 and 2.4, we clearly see the benefits of using a subdiagonal Padé approximant instead of a diagonal one. Even though the diagonal Padé approximant has better accuracy near the origin (as it should since it has one degree higher accuracy there), globally the subdiagonal Padé approximant provides a much better approximation to e^z when $\mathrm{Re}(z)$ is negative with large absolute value. Indeed, the type $(3, 4)$ Padé approximant is more accurate than the standard $(13, 13)$ Padé approximant if A has eigenvalues with real part -10^3 or smaller, as can be seen in Figures 2.2 and 2.6.

This phenomenon is crucial to the cost savings in our algorithm and hence warrants further explanation. With a subdiagonal Padé approximant $r_{m-1,m}(z/2^s) = p_{m-1}(z/2^s)/q_m(z/2^s)$ we have $|r_{m-1,m}(z/2^s)| \ll 1$ for sufficiently large $|z|$ since $|p_{m-1}(z/2^s)| \ll |q_m(z/2^s)|$. For example, when $s = 4$ we have

$$|(r_{m-1,m}(z/2^s))^{2^s}| = |r_{m-1,m}(z/16)|^{16},$$

which shows that as long as $|r_{m-1,m}(z/16)| < 0.1$ we have $|(r_{m-1,m}(z/16))^{16}| = O(u)$.

Moreover, since $|e^z| = e^{\mathrm{Re}(z)}$, we have $|e^z| = O(u)$ for z with large negative real parts. Indeed if the real part of z satisfies $\mathrm{Re}(z) < -\log 10^{-16} \simeq -36.8$, then $|e^z| \lesssim u$, which is of negligible size compared with $O(1)$.

Combining the above two observations, we conclude that in the inequality

$$|(r_{m-1,m}(z/2^s))^{2^s} - e^z| \le |(r_{m-1,m}(z/2^s))^{2^s}| + |e^z|$$

both terms on the right are negligibly small provided that $\mathrm{Re}(z) \le -36.8$, which we can verify in Figures 2.1 and 2.2.

3. Scaled and squared subdiagonal Padé approximation for e^A. We now describe our algorithm `sexpm`. The standard scaling and squaring method is known to be forward stable for certain types of matrices, including normal matrices [5], and we aim to design an algorithm that retains this property. Recall that forward stability means that the forward error is about the same order as that of a backward stable algorithm [21, Sec. 1.6], so we generally accept forward errors $\|e^A - fl(e^A)\|$ of size $u\,\kappa_{\exp}(A)\,\|e^A\| \gtrsim u\,\kappa_{\exp}(A)$, recalling from section 1.1 that $||e^A|| \gtrsim 1$ due to the initial shift. Since by (1.2) the condition number $\kappa_{\exp}(A)$ is bounded below by $||A||$, we can take $u||A||$ to be a lower bound for the error accepted in a forward stable algorithm. The guiding principle for the parameter choice is therefore to control the approximation error $|e^z - (r_{k,m}(z/2^s))^{2^s}|$ to be no larger than $u||A||$ for values of z including those in the spectrum of A.

3.1. Choice of the parameters. We determine the scaling parameter s and the Padé degree (k, m) based on the value or estimate of $\|A\|_2$. Specifically, we choose the parameters to minimize the cost $s+m$ (which measures the total number of matrix multiplications and inversions) while ensuring that the approximation error satisfies $|e^z - (r_{k,m}(z/2^s))^{2^s}| < u\|A\|_2$ on $[-\|A\|_2, 0]$. Note that this choice is based on the approximation errors on the real axis, and does not control the error in a complex region. However, we recall from the figures in the previous section that the error grows only moderately as the imaginary part of z grows, especially when the real part of z is large and negative.

We thus obtain the parameters in Tables 3.1 (for $\|A\|_2 \ge 1$) and 3.2 ($\|A\|_2 \le 1$).

TABLE 3.1

Parameters $s, (k, m)$ depending on (an estimate of) $\|A\|_2$ and the corresponding uniform approximation error on $[-\|A\|_2, 0]$ and subtractive cancellation error; both errors should be a moderate multiple of $\max(u\|A\|_2, u)$.

$\|A\|_2$	$1\sim$	$200\sim$	$10^4\sim$	$10^6\sim$	$10^9\sim$	$10^{11}\sim$	$10^{12}\sim$	$10^{14}\sim$
$s,(k,m)$	4,(5, 4)	4,(4, 5)	4,(3, 4)	3,(3, 4)	2,(3, 4)	2,(2, 3)	2,(1, 2)	1,(1, 2)
flops/n^3	16	18	16	14	12	10	8	6
approx. error	1.5e-14	1.4e-14	1.4e-11	1.7e-9	4.2e-6	1.6e-5	8.3e-4	9.6e-3
subtract. canc.	1.7e-14	9.6e-15	2.7e-15	2.7e-15	2.7e-15	1.1e-15	3.3e-16	3.3e-16

TABLE 3.2
Parameters $s, (k, m)$ and errors for $\|A\| \leq 1$.

$\|A\|_2$	~ 1	~ 0.5	~ 0.3	~ 0.15	~ 0.07	$\sim 10^{-2}$	$\sim 10^{-4}$	$\sim 10^{-5}$	$\sim 10^{-8}$
$s, (k, m)$	4,(4, 3)	3,(4, 3)	2,(4, 3)	1,(4, 3)	0,(4, 3)	0,(3, 2)	0,(3, 0)	0,(2, 0)	0,(1, 0)
flops/n^3	14	12	10	8	6	4	4	2	0
approx. error	3.8e-15	2.1e-15	2.3e-15	1.3e-16	3.3e-16	3.3e-16	4.4e-16	2.2e-16	1.1e-16
subtract. canc.	6.8e-15	6.8e-15	6.8e-15	6.8e-15	2.7e-15	1.8e-15	2.2e-16	2.2e-16	2.2e-16

Note that the flop counts in Table 3.1 decrease as $\|A\|_2$ grows. This is because as $\|A\|_2$ decreases, the lower bound for the condition number $\kappa_{\exp}(A)$ also decreases, which forces the approximation quality to be more stringent. This is in stark contrast to the standard scaling and squaring method, in which the scaling parameter s needs to grow with $\|A\|_2$, resulting in a more expensive method with arithmetic cost about $2(7 + \log_2 \|A\|_2)n^3$ flops. Clearly, the larger $\|A\|_2$ is, the more efficient our method `sexpm` becomes relative to the standard method.

Note that when $\|A\|_2 = O(1)$ or smaller, Table 3.1 recommends the use of a superdiagonal Padé approximant, which gives better accuracy near the origin at roughly the same cost. However, the speedup compared to the standard method is negligible in this easy case. In fact, even a simple Taylor series approximation would suffice in this case, not requiring any matrix inverses. For the majority of the paper our treatment hence focuses on the subdiagonal case for $\|A\|_2 \gg 1$.

3.2. Evaluating the Padé approximant. As Table 3.1 shows for $\|A\|_2 \gg 1$, our method requires the evaluation of a subdiagonal Padé approximant $r_{m-1,m}$ at a matrix argument $\widetilde{A} := A/2^s$. This can be done in a number of ways summarized below.

1. Partial fraction. Use the expression

$$r_{m-1,m}(z) = \sum_{i=1}^{m} \frac{a_i}{z - b_i}$$

and evaluate

$$r_{m-1,m}(\widetilde{A}) = \sum_{i=1}^{m} a_i(\widetilde{A} - b_i I)^{-1}. \tag{3.1}$$

For superdiagonal Padé we use $r_{m+1,m}(z) = \alpha_0 + \alpha_1 z + \sum_{i=1}^{m} \frac{a_i}{z-b_i}$ to evaluate $r_{m+1,m}(\widetilde{A}) = \alpha_0 I + \alpha_1 \widetilde{A} + \sum_{i=1}^{m} a_i(\widetilde{A} - b_i I)^{-1}$.

2. Product form. For the subdiagonal Padé approximant, express $r_{m-1,m}(z) = p_{m-1}(z)/q_m(z)$ as

$$r_{m-1,m}(z) = \frac{\prod_{i=1}^{m-1}(z - t_i)}{\prod_{i=1}^{m}(z - b_i)},$$

and evaluate $r_{m-1,m}(z)$ by the recursion $X_1 = I$,

$$X_{i+1} = (\widetilde{A} - t_i I)(\widetilde{A} - b_i I)^{-1} X_i$$

for $i = 1, \ldots, m-1$, and $r_{m-1,m}(\widetilde{A}) = (\widetilde{A} - b_m I)^{-1} X_m$. The superdiagonal case is analogous.

3. Direct evaluation. Compute $P = p_k(\widetilde{A})$ and $Q = q_m(\widetilde{A})$ independently, then obtain $r_{k,m}(\widetilde{A}) = PQ^{-1}$.
4. Continued fraction. Use the representation

$$r_{m-1,m}(z) = b_0 + \cfrac{a_1 z}{b_1 + \cfrac{a_2 z}{b_2 + \cfrac{a_3 z}{b_3 + \ddots}}}$$

and evaluate $r_{m-1,m}(\widetilde{A})$ in a bottom-up fashion (top-down reduces to the direct evaluation).

We argue below that both the partial fraction form and the product form have desirable stability properties for our method. Among the two, partial fraction form is usually more efficient and can be implemented in parallel. We use the partial fraction form unless a relative accuracy of 10^{-13} or better is desired (when this is possible, which necessarily means that $\|A\| \leq 10^3$).

One downside of both partial fraction and product forms is that they involve complex arithmetic even when A is real. With the partial fraction form, we can at least exploit the fact that the poles b_i and residues a_i in (3.1) appear in complex conjugate pairs, and that for real A we have

$$a_i(\widetilde{A} - b_i I)^{-1} + \bar{a}_i(\widetilde{A} - \bar{b}_i I)^{-1} = 2\mathrm{Re}(a_i(\widetilde{A} - b_i I)^{-1}).$$

This leads to a reduction in evaluation cost by nearly a half.

When A is real, the identity

$$a_i(z - b_i I)^{-1} + \bar{a}_i(z - \bar{b}_i I)^{-1} = \frac{a_i(z - \bar{b}_i) + \bar{a}_i(z - b_i)}{(z - b_i)(z - \bar{b}_i)} = \frac{2(z\mathrm{Re}(a_i) - \mathrm{Re}(a_i b_i))}{z^2 - 2z\mathrm{Re}(b_i) + |b_i|^2}$$

circumvents complex arithmetic, however, is not recommended numerically as the matrix $\widetilde{A}$ can still be of large norm and hence the squaring in the denominator may lead to severe subtractive cancellation.

The standard scaling and squaring method uses direct evaluation [22, p. 246]. There are two reasons for this: (i) $p_k(\widetilde{A})$ and $q_m(\widetilde{A})$ are closely related, so that for a type $(13, 13)$ Padé approximant both can be evaluated synchronously using 6 matrix multiplications; (ii) $q_m(\widetilde{A})$ is shown to be well conditioned, so there is little danger in the inversion.

Unfortunately the second argument is not valid in our method because $\|\widetilde{A}\|$ is not of $O(1)$, which is a consequence of the bounded scaling parameter $s \leq 4$. Indeed it is easy to verify that in our method $q_m(\widetilde{A})$ is highly ill-conditioned if A has large norm, resulting in instability of direct evaluation. This instability is avoided by the partial fraction or product form.

The continued fraction form has the advantage that if A is real then $r_{m-1,m}(\widetilde{A})$ can be computed using real arithmetic only. However, the intermediate inversions are generally more ill-conditioned than those in the partial fraction form, again giving inaccurate results [29], which we also observed through experiments. Moreover, continued fractions are not as amenable to parallelization as partial fractions are. For these reasons we do not further consider continued fractions.

3.3. Pseudocode. The pseudocode below summarizes our algorithm `sexpm` for computing the matrix exponential[1]. Recall that the matrix A is assumed to be already shifted so that the largest real part of all eigenvalues is (approximately) 0. For completeness, we make explicit the role played by the shift σ.

Algorithm 3.1 `sexpm` : Compute e^A for $A \in \mathbb{C}^{n\times n}$.

1. Estimate the rightmost eigenvalue σ of A, and take $A_\sigma = A - \sigma I$.
2. Estimate $\|A_\sigma\|_2$ and choose $s, (k, m)$ from the second row of Tables 3.1 and 3.2.
3. Evaluate Padé approximant: $r_{k,m}(A_\sigma/2^s) = p_k(A_\sigma/2^s)q_m(A_\sigma/2^s)^{-1}$ via partial fraction or product form.
4. Perform repeated squaring: $e^\sigma(r_{k,m}(A_\sigma/2^s))^{2^s} \approx e^A$.

4. Forward stability. Higham [23] established backward stability of scaling and squaring in exact arithmetic, though backward stability in finite precision arithmetic remains an open problem. Regarding forward stability, several results exist [5, 9, 10]. Among these, of particular relevance is that the standard scaling and squaring method is forward stable for normal matrices, as shown by Arioli, Codenotti, and Fassino [5]; see also [22, p. 248].

Here we examine the stability of our method. As in the analysis of [5] we denote by $fl(x)$ a computed approximation to x, so the output of `sexpm` is $fl((r_{k,m}(A/2^s))^{2^s}) =: fl(e^A)$. We ignore the errors resulting from scaling the matrix and shifting, as these operations are done essentially at the scalar level. Then the errors that arise in a scaling and squaring algorithm can be separated into three components (for definiteness we use the spectral norm):

1. Truncation (approximation) error. This is the error that results if the method is executed in exact arithmetic (recall $\widetilde{A} = A/2^s$):
$$E_t = \|(r_{k,m}(\widetilde{A}))^{2^s} - e^A\|_2.$$
2. Error in evaluating the Padé approximant:
$$E_p = \|fl(r_{k,m}(\widetilde{A})) - r_{k,m}(\widetilde{A})\|_2.$$
3. Error in the squaring phase:
$$E_s = \|fl(fl(r_{k,m}(\widetilde{A}))^{2^s}) - fl(r_{k,m}(\widetilde{A}))^{2^s}\|_2.$$

The subscripts in E_s and E_p reflect the relevant parameters that determine the operations. Using the triangular inequality, the error in the computed $fl(e^A) = fl(fl(r_{k,m}(\widetilde{A}))^{2^s})$ can be bounded as

$$\begin{aligned}\|e^A - fl(e^A)\|_2 &\le \|e^A - (r_{k,m}(\widetilde{A}))^{2^s}\|_2 + \|(r_{k,m}(\widetilde{A}))^{2^s} - fl(e^A)\|_2 \\ &\le E_t + \|(r_{k,m}(\widetilde{A}))^{2^s} - (fl(r_{k,m}(\widetilde{A})))^{2^s}\|_2 + \|(fl(r_{k,m}(\widetilde{A})))^{2^s} - fl(e^A)\|_2 \\ &= E_t + \|(r_{k,m}(\widetilde{A}))^{2^s} - fl(r_{k,m}(\widetilde{A}))^{2^s}\|_2 + E_s. \qquad (4.1)\end{aligned}$$

By defining $B := r_{k,m}(\widetilde{A})$ and $B + \Delta B := fl(r_{k,m}(\widetilde{A}))$, the second term is

$$\|B^{2^s} - (B + \Delta B)^{2^s}\|_2 \lesssim 2^s\|\Delta B\|_2 + O(\|\Delta B\|_2^2),$$

[1] A MATLAB implementation of `sexpm` is available at `http://www.opt.mist.i.u-tokyo.ac.jp/~nakatsukasa/expmpade.htm`

where we used $\|B\|_2 \approx 1$. Since $\|\Delta B\|_2 = \|E_p\|_2$, altogether we conclude that

$$\|e^A - fl(e^A)\|_2 \lesssim E_t + 2^s E_p + E_s. \tag{4.2}$$

Since in `sexpm` we have $s \le 4$ and hence $2^s = O(1)$, we conclude that for achieving forward stability it suffices to ensure that E_t, E_p, E_s are controlled to be $O(u\|A\|_2)$.

4.1. Error analysis in the Padé evaluation. The most intricate of the three sources of error is the evaluation of the Padé approximant, which we examine in detail.

4.1.1. Inversion of ill-conditioned matrices. Evaluating the Padé approximant at a matrix argument involves matrix inversions. Since the relative error in a computed matrix inverse M^{-1} can be as large as $O(u\kappa_2(M)) = O(u\|M\|_2\|M^{-1}\|_2)$, care is needed to avoid the inversion of matrices that are too ill-conditioned.

The standard scaling and squaring method chooses the parameters s and m in a way that guarantees that the quotient matrix in the type (m, m) Padé approximant is well conditioned [22, p. 240], so that computing its inverse can be done without severe numerical error. The new method `sexpm`, however, does not guarantee this. In fact the denominator matrix can be highly ill-conditioned; see also section 6.2. Here we shall argue that this still does not affect the forward stability of the method.

Recall from section 2 that the poles of the Padé approximant $r_{k,m}$ are in the right half-plane. Therefore, denoting by $\widetilde{A} = X\widetilde{\Lambda}X^{-1}$ an eigenvalue decomposition of $\widetilde{A}$, the inversions that we perform are of the form

$$(\widetilde{A} - bI)^{-1} = X^{-1}(\widetilde{\Lambda} - bI)X.$$

As is well known [21, Ch. 4], the relative error in the computed inverse is proportional to the condition number of the matrix, so we have

$$\|fl((\widetilde{A} - bI)^{-1}) - (\widetilde{A} - bI)^{-1}\|_2 = O(u\kappa_2(\widetilde{A} - bI)\|(\widetilde{A} - bI)^{-1}\|_2). \tag{4.3}$$

We first examine the $\kappa_2(\widetilde{A} - bI)$ term in (4.3). We have

$$\kappa_2(\widetilde{A} - bI) \le \kappa_2(X) \cdot \kappa_2(\widetilde{\Lambda} - bI).$$

Note that when A (and hence $\widetilde{A}$) is normal the eigenvector matrix X can be taken as unitary, and so $\kappa_2(\widetilde{A} - bI) = \kappa_2(\widetilde{\Lambda} - bI)$. In general, $\mathrm{Re}(b) \ge 1$ in all the Padé approximants that we employ in Table 3.1: the precise values of b_i are $3.2128 \pm 4.7731i$ and $4.7872 \pm 1.5675i$ when $m = 4$, and other values give similar results with $0 \le \mathrm{Re}(b_i) = O(1)$, as can be verified from (2.1) by finding the roots of the denominator polynomial. Recalling that the matrix $\widetilde{A}$ is initially shifted so that its eigenvalues lie in the left half-plane, we have $\min_i |\widetilde{\lambda}_i - b| \ge 1$, and therefore

$$\kappa_2(\widetilde{\Lambda} - bI) = O\left(\frac{\kappa_2(X)|\widetilde{\lambda}_{\max}|}{\min_i |\widetilde{\lambda}_i - b|}\right) = O(\kappa_2(X)|\widetilde{\lambda}_{\max}|) = O(\kappa_2(X)\|\widetilde{A}\|_2), \tag{4.4}$$

which shows that the condition number of the matrix to be inverted is bounded roughly by $\kappa_2(X)\|\widetilde{A}\|_2$.

The $\|(\widetilde{A} - bI)^{-1}\|_2$ term in (4.3) can be bounded by $\kappa_2(X)$, again using the fact $\min_i |\widetilde{\lambda}_i - b| \ge 1$. Overall we conclude that

$$\|fl((\widetilde{A} - bI)^{-1}) - (\widetilde{A} - bI)^{-1}\|_2 = O(u\kappa_2(X)^2\|\widetilde{A}\|_2). \tag{4.5}$$

Recalling that the acceptable forward error in computing $e^{\widetilde{A}}$ is at least $O(u\|\widetilde{A}\|_2)$, we see that the inversions cause error of acceptable magnitude as long as $\kappa_2(X)$ is moderate.

4.1.2. Subtractive cancellation in the partial fraction form. Having computed the inverses $(\widetilde{A} - b_i I)^{-1}$, another source of error in the evaluation of (3.1) is in the summation. The potential danger is in subtractive cancellation, which may arise if an intermediate term during the summation $\sum_{i=1}^{\tilde{k}} a_i(\widetilde{A} - b_i I)^{-1}$ for some $\tilde{k}$ has much larger norm than $\|r_{m-1,m}(\widetilde{A})\|_2$. In our case $\|(\widetilde{A} - b_i I)^{-1}\|_2$ and $\|r_{m-1,m}(\widetilde{A})\|_2$ are both $O(\kappa_2(X))$ for all i, so when $\kappa_2(X)$ is moderate, problematic cancellation occurs only if $|a_i| \gg 1$ for some i.

This is another crucial benefit of using a low-degree Padé approximant: the residues a_i in the partial fraction (3.1) are all moderate, being less than 300 in absolute value. Therefore the summation causes the computed e^A to lose only about two digits of accuracy, which can be interpreted as an $O(100u)$ error. By contrast, the standard method makes the partial fraction form prohibitively unstable because the type $(13, 13)$ Padé approximant involves terms as large as $O(10^8)$

Nonetheless, there is one situation in which the use of partial fraction does not give the best possible numerical accuracy: when $\|A\|_2$ is small, say ≤ 100, so that $\kappa_{\exp}(A)$ is not ruled out to be $O(1)$, and one may want accuracy better than $O(100u)$. A cure for this is to use the product form, and our experiments indeed confirm that this gives improved accuracy when $\|A\|_2$ is small.

It may be rare that an error of $O(100u)$ is unacceptable, and in any case the subtractive cancellation error gets outsized by the inherent conditioning $\kappa_{\exp}(A)$ for any $\|A\|_2 \geq 100$. In our algorithm we therefore choose to use the partial fraction form by default, and allow an optional tolerance for the desired accuracy. If the desired accuracy is smaller than the error expected by subtractive cancellation, we switch to the product form evaluation.

4.2. Proof of forward stability for normal matrices. We now show that for near-normal matrices that have no rightmost eigenvalues with large imaginary parts, the forward stability of the scaling and squaring method is preserved in `sexpm`.

Our goal is to show that each term E_i on the right-hand side of (4.2) is bounded by $u\|e^A\|_2\kappa_{\exp}(A)$, which indicates that $fl(e^A)$ is computed in a forward stable manner. By (1.2) we have $\kappa_{\exp}(A) \geq \|A\|_2$ (in fact for normal matrices this is an equality) and $\|e^A\|_2 \approx 1$ (since we deal with the "shifted" matrix A with $\mathrm{Re}(\lambda(A)) \lesssim 0$), so it suffices to prove that $E_i \leq c_i u\|A\|_2$, where c_i is a modest constant.

THEOREM 4.1. *For a normal matrix $A \in \mathbb{C}^{n\times n}$ satisfying $\left|(r_{k,m}(z/2^s))^{2^s} - e^z\right| \leq u\|A\|_2$ for all $z \in \Lambda(A - \sigma I)$, Algorithm 3.1 computes an approximant $fl(e^A)$ to the matrix exponential e^A in a forward stable manner, that is,*

$$\frac{\|fl(e^A) - e^A\|_2}{\|e^A\|_2} = c_n\kappa_{\exp}u \tag{4.6}$$

for a modest constant c_n, which is a low-order polynomial in n but otherwise independent of A.

Proof. By (4.1), (4.2) and the fact $s \leq 4$, it suffices to prove that E_t, E_p and E_s are all bounded by $c\kappa_{\exp}u$. Below we examine each term.

Bounding E_t. The approximation error E_t is precisely equal to

$$E_t = \max_{z\in\Lambda(A)} \left|(r_{k,m}(z/2^s))^{2^s} - e^z\right|, \tag{4.7}$$

as we have $\kappa_2(X) = 1$ in (2.2). By assumption this is bounded by $u\|A\|_2$ by the choice of parameters s, m in Table 3.1.

Bounding E_p. The Padé approximant evaluation error depends on the evaluation scheme and needs careful analysis.

Suppose we use the partial fraction form $r_{m-1,m}(\widetilde{A}) = \sum_{i=1}^{m} a_i(\widetilde{A} - b_i I)^{-1}$ as in (3.1). The evaluation of matrix inverses $(\widetilde{A} - b_i I)^{-1}$ is the primary source of error here, and it is $O(u\|\widetilde{A}\|_2)$ by (4.5). The summation of the m terms introduces error $O(u \max_i \|(\widetilde{A} - b_i I)^{-1}\|_2 + 100u)$, where $100u$ comes from potential subtractive cancellations. This is $O(u\|A\|_2)$ unless $\|A\|_2 = O(1)$.

A similar argument proves the claim for the product form evaluation, noting that the involved poles are clearly the same as those in the partial fraction form. For the product form there is no subtractive cancellation, hence the overall error is $O(u\|A\|_2)$ regardless of $\|A\|_2$.

Bounding E_s. The error in the squaring phase is innocuous because the normality of A ensures that the hump effect does not occur, hence by the same analysis as in [5],

$$E_s = \|fl(fl(r_{k,m}(\widetilde{A}))^{2^s}) - fl(r_{k,m}(\widetilde{A}))^{2^s}\|_2 = O(ufl(r_{k,m}(\widetilde{A}))^{2^s}) = O(u),$$

where we used the fact $\|r_{k,m}(\widetilde{A})\|_2 \lesssim 1$.

Overall we have shown that $E_t + E_p + E_s \le c_n u\|A\|_2$, so by (4.2) we conclude that `sexpm` is forward stable for a normal matrix A. ∎

The assumption $\max_{z\in\Lambda(A)} \left|(r_{k,m}(z/2^s))^{2^s} - e^z\right| \le u\|A\|_2$ holds clearly if the eigenvalues are all real, but as discussed in section 2, it holds much more generally, unless A has a rightmost eigenvalue with large imaginary part.

As the proof indicates, strictly speaking, for $\|A\|_2 \lesssim 100$ the effect of subtractive cancellation of $O(100u)$ comes into play with the partial fractions evaluation, and the product form is necessary if accuracy of $O(\kappa_{\exp}u) \approx O(u)$ is required.

4.3. Nonnormal matrices. For highly nonnormal matrices, for which $\kappa_2(X) \gg 1$, the standard scaling and squaring method has not been proven to be forward stable, and the same applies to `sexpm`. Numerical experiments suggest that `sexpm` and `expm` usually give comparable accuracy. We briefly discuss the conditioning of $\kappa_{\exp}(A)$ for nonnormal A and argue why one can reasonably expect `sexpm` to be forward stable.

A crucial component in our algorithm is to take advantage of the lower bound $\kappa_{\exp}(A) \ge \|A\|$ for the condition number in (1.2). For nonnormal matrices, $\kappa_{\exp}(A)$ can be much larger than $\|A\|$. For diagonalizable A with eigendecomposition $A = X\Lambda X^{-1}$, Moler and Van Loan [27, 40] derive an upper bound to first order in $u\kappa_2(X)$,

$$\kappa_{\exp}(A) \le O(\kappa_2(X)^2\|A\|). \tag{4.8}$$

This bound is more informative than the upper bound in (1.2), which is extremely large when $\|A\| \gg 1$ and the rightmost eigenvalues of A are $O(1)$ or smaller. The bound (4.8) shows that the conditioning of the exponential satisfies $\|A\|_2 \le \kappa_{\exp}(A) \lesssim \|A\|_2\kappa_2(X)^2$, and so we can write

$$\kappa_{\exp}(A) = \kappa_2(X)^{\alpha_A}\|A\| \tag{4.9}$$

for some $\alpha_A \in [0, 2]$. In practice $\kappa_{\exp}(A)$ can be estimated by the algorithm in [1], and our experiments suggest that $\alpha_A = 1$ is a typical value, that is, $\kappa_{\exp}(A) \approx \kappa_2(X)\|A\|$. Hence the acceptable error of a forward stable algorithm is $O(u\kappa_2(X)\|A\|)$.

A straightforward extension of the proof of Theorem 4.1 to bound E_t, E_p and E_s for a nonnormal matrix A gives $E_t = O(u\kappa_2(X)\|A\|)$, $E_p = O(u(\kappa_2(X))^2\|A\|)$, and

$E_s = O(u \max_{1\le\tau\le s} fl(r_{k,m}(\widetilde{A}))^{2^\tau})$. E_s can grow if the hump effect occurs. It follows that `sexpm` is forward stable if (i) $E_t, E_p, E_s = O(u(\kappa_2(X))^{\alpha_A}\|A\|\|e^A\|)$ for α_A as in (4.9), and (ii) no hump effect of factor $(\kappa_2(X))^{\alpha_A}\|A\|\|e^A\|$ or more takes place in the repeated squaring phase.

Recalling that in `sexpm` overscaling is avoided by always taking $s \le 4$, we therefore argue that `sexpm` has potential advantage in stability compared with the standard algorithm. We discuss the overscaling issue further in section 6.1.

5. Computing the action on a vector. Here we consider computing $e^A\boldsymbol{b}$, the action of the matrix exponential on a vector (or a tall skinny matrix) $\boldsymbol{b} \in \mathbb{C}^{n\times\ell}$. The goal is to compute $e^A\boldsymbol{b}$ without forming e^A explicitly, and our method allows this with no major modification. Using the partial fraction form of $r_{k,m}$, the action of $r_{k,m}(A)$ on a vector $\boldsymbol{b}$ is readily computable as

$$r_{k,m}(A)\boldsymbol{b} = \sum_{i=1}^{m} a_i(A - b_iI)^{-1}\boldsymbol{b} = \sum_{i=1}^{m} a_i\boldsymbol{x}_i, \tag{5.1}$$

which corresponds to summing the m solutions to the (block) linear systems $(A - b_iI)\boldsymbol{x}_i = \boldsymbol{b}$. We obtain $e^A\boldsymbol{b} \approx (r_{k,m})^{2^s}\boldsymbol{b}$ by repeating (5.1) $2^s(\le 16)$ times.

An analogous evaluation is possible using the product form, and as was the case with computing the whole matrix e^A, the product form has slightly better stability than partial fraction when $\|A\| = O(100)$.

Algorithm 5.1 `sexpmv`: Computing $e^A\boldsymbol{b}$ for $A \in \mathbb{C}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{C}^{n\times\ell}$.

1 Estimate $\|A\|_2$ and choose s, m from Tables 3.1 and 3.2.
2 Initialize $\boldsymbol{b}_0 = \boldsymbol{b}$.
3 Via partial fraction or product form evaluate

$$\boldsymbol{b}_i = r_{k,m}(A)\boldsymbol{b}_i = p_k(A)q_m(A)^{-1}\boldsymbol{b}_{i-1}, \quad i = 1 : 2^s.$$

4 Output $\boldsymbol{b}_{2^s} \approx e^A\boldsymbol{b}$.

The method requires $m \le 5$ solutions of linear systems, each of which is repeated 2^s times where $s \le 4$. Therefore the overall cost is about $2^s mM \le 80M$, where M denotes the cost of solving a linear system. However, it should be noted that the 2^s repetitions of linear system solves are with the same shifted matrices. Hence, if direct solvers are employed for their solution, LU factorizations need to be computed only once for each i in (3.1), and can be reused for all 2^s repetitions, hence requiring only m factorizations, where $m \le 5$ is the denominator degree.

The amenability for computing $e^A\boldsymbol{b}$ is in stark contrast to the standard method, which uses direct evaluation to form $p_k(A)q_m(A)^{-1}$. This can be overcome by using the product form (recall that the partial fraction form is highly unstable in the standard (13, 13) Padé function evaluation because of subtractive cancellation), but since the scaling parameter s is such that $2^s \approx \|A\|$, the cost is $O(2^sM) = O(\|A\|M)$, which is clearly prohibitive unless $\|A\| = O(1)$. Indeed the method in [3] relies on the Taylor series approximant instead of Padé approximants, and it is efficient only for A of moderate norms (or more generally when $\|A^k\|^{1/k} \ll \|A\|$ is moderate for some k), see Figure 7.7 in the experiments.

The stability results in section 4 for `sexpm` carry over to the vector problem `sexpmv` with essentially the same analysis.

6. Comparison with conventional scaling and squaring and other methods. We have already highlighted some differences between our `sexpm` and the standard `expm` [23] (or its improved version `expm_new` [2]). Here we discuss further aspects that contrast the methods.

6.1. Overscaling issue. The great cost saving in our algorithm stems from the observation that the exponential is negligible in magnitude at eigenvalues sufficiently far to the left of the rightmost ones (by more than $\ln 10^{-16} \simeq 36.8$), making it acceptable to use scaling parameters $s \leq 4$. The issue of overscaling, which refers to an unnecessarily large s, has long been known as a possible cause for instability in the scaling and squaring method [2, 9, 24, 27]. Our method almost completely solves this issue, leading to benefits in stability as discussed in section 4.3 and observed in our numerical experiments.

The `expm_new` algorithm developed in [2] also attempts to attenuate overscaling, but from a different standpoint of measuring the nonnormality of A via the quantities $\|A^p\|^{1/p}$ for $p = 1, 2, \ldots$, which are all bounded by $\|A\|$ but can be significantly smaller for nonnormal matrices. Detecting this can result in much smaller scaling factor than the conventional `expm` while maintaining stability in the Padé phase, but for normal matrices this makes no difference. `sexpm` reduces scaling based on a rational approximation viewpoint, and is effective whether or not the matrix is normal.

6.2. On previous arguments to prefer diagonal Padé approximants. Most existing approaches for computing the matrix exponential via Padé approximation use diagonal approximants. There are two main reasons for this [27, Sec. 3]:

1. Efficiency. Evaluating an (m, m) Padé approximant can be done at the same computational cost as for type $(m-1, m)$. Since a higher-order approximant is expected to give a better approximation, the diagonal (m, m) Padé approximant is preferred.
2. Stability. Evaluating a nondiagonal Padé approximant can lead to an ill-conditioned denominator matrix, leading to inversion of an ill-conditioned matrix. The denominator matrix is well-conditioned in the standard method.

We explain why our new method still uses a type $(m-1, m)$ Padé approximant. First, regarding efficiency, we have observed that m much smaller than that used by the standard algorithm is sufficient to maintain the accuracy of the rational approximant. Put another way, even though near the origin a type (m, m) approximant has smaller error, in a wider region (the region that contains the eigenvalues) the type $(m-1, m)$ approximant gives smaller error. Since an absolute accuracy of $O(u\|A\|)$ is sufficient (or the best we can hope for), a dramatic reduction in m and s is possible when the real parts of the eigenvalues vary widely.

Regarding stability, as discussed in section 4.1.1, the condition numbers of the matrices to be inverted in `sexpm` are bounded by $O(\kappa_2(X)\|A\|_2)$, indicating that the inversions do not cause error of unacceptable magnitude provided $\kappa_2(X) = O(1)$.

6.3. "Best" rational approximation. We argued that the parameters s and m in Table 3.1 are sufficient for our accuracy requirements, resulting in the use of a rational function of much lower degree compared to the choice $O(2^s m) \approx 13\, O(\|A\|_2)$ in the standard scaling and squaring method. However, our rational approximant has not necessarily the lowest degree possible to be a sufficient accurate approximation on some spectral region $\mathcal{D}$ of A. It is therefore natural to consider the use of best rational approximants, like, e.g., the lowest-degree rational function $r(z)$ for which $\sup_{z\in\mathcal{D}} |e^z - r(z)|$ is bounded by $O(u\|A\|)$.

When the matrix has only real eigenvalues, the best rational approximant is a well studied subject, and the best type (m,m) rational function on $(-\infty,\ell]$ for any $\ell \geq 0$ is known to have an error $O(9.28^{-m})$ [33]. The rational function can be computed numerically (essentially exactly) by the Carathéodory–Fejér method [36, Ch. 20],[37]. In double precision arithmetic $m = 13$ is sufficient to obtain approximation error $O(u)$ for $\ell = 0$, and indeed for Hermitian matrices, the use of best rational approximants to the exponential has been successfully implemented [14],[22, Sec. 10.7.1],[33].

By contrast, for general A with complex eigenvalues, the construction of computationally practical best or near-best rational approximants is more involved and seems to have received less attention. This work can be regarded as a first step towards filling this gap by using a rational function that is much closer to optimal than that of the standard scaling and squaring method. It is worth noting that the Carathéodory–Fejér approximation on $(-\infty,0]$ has the property that the error $|e^z - r(z)|$ is small if $\mathrm{Re}(z) < 0$ and $|\mathrm{Re}(z)| \gg 1$ (see [38] for an error plot). It can be implemented using m matrix inversions (linear system solves for the $e^A\boldsymbol{b}$ problem), but the approximation error grows quickly as $\mathrm{Re}(z)$ becomes positive, and so the choice of the initial shift is more crucial than in our method. Moreover, the approximation quality deteriorates rapidly as the imaginary parts of the eigenvalues grow.

6.4. Schur decomposition-based algorithms. Another contender for e^A is the algorithm based on the Schur decomposition $A = QTQ^*$, commonly called the Schur–Parlett algorithm [22, Ch. 9]; see [22, Sec. 10.4] for its specialization to e^A.

For `sexpm`, computing the Schur decomposition prior to the algorithm has two potential benefits: (i) it provides the eigenvalues, which informs us about the optimal shift σ and whether the rightmost eigenvalues with large imaginary parts are present, and (ii) the problem essentially reduces to computing e^T for the triangular matrix T, for which the matrix inversions in (3.1) are stable and efficient. Therefore, when there is possibility for rightmost eigenvalues with large imaginary parts, we recommend first computing the Schur decomposition. Furthermore, if the Schur decomposition reveals the presence of such eigenvalues, we can use one of remedies discussed in section 7.2.

6.5. Rational Krylov methods. These methods are applicable for the problem of computing $e^A\boldsymbol{b}$, when $\boldsymbol{b} \in \mathbb{C}^{n\times 1}$. They are based on extracting an approximation to $e^A\boldsymbol{b}$ from a d-dimensional rational Krylov space [32]

$$\begin{aligned}\mathcal{Q}_d(A,\boldsymbol{b}) = \mathrm{span}\big\{&(A-b_1I)^{-1}\boldsymbol{b}, (A-b_1I)^{-2}\boldsymbol{b}, \ldots, (A-b_1I)^{-s_1}\boldsymbol{b}, (A-b_2I)^{-1}\boldsymbol{b},\\ &(A-b_2I)^{-2}\boldsymbol{b}, \ldots, (A-b_2I)^{-s_2}\boldsymbol{b}, \ldots, (A-b_mI)^{-1}\boldsymbol{b}, \ldots, (A-b_mI)^{-s_m}\boldsymbol{b}\big\},\end{aligned}$$

defined for a set of distinct shifts $b_1, b_2, \ldots, b_m \in \mathbb{C}$ of multiplicities $s_1, s_2, \ldots, s_m$, with $d = s_1 + s_2 + \cdots + s_m$. The popular rational Arnoldi method for extracting an approximation $\boldsymbol{f}_d \approx e^A\boldsymbol{b}$ from $\mathcal{Q}_d(A,\boldsymbol{b})$ proceeds as follows: compute an orthonormal basis $V_d = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d] \in \mathbb{C}^{n\times d}$ of $\mathcal{Q}_d(A,\boldsymbol{b})$, and define $\boldsymbol{f}_d = V_d e^{A_d} V_d^*\boldsymbol{b}$, where $A_d = V_d^* A V_d$ typically is much smaller than A. The choice of optimal shifts for rational Krylov spaces for the approximation of matrix functions is an active research area. It is important to note here that the approximant $(r_{m-1,m}(A/2^s))^{2^s}\boldsymbol{b}$ computed by our method `sexpmv` is an element of $\mathcal{Q}_d(A,\boldsymbol{b})$, provided that the shifts and multiplicities have been chosen identical to the poles of the underlying squared Padé approximant. Another popular parameter choice is to use just a single repeated shift b_1 with multiplicity $s_1 = d$. The resulting method is then closely related to the *shift-and-invert Arnoldi method* and so-called *restricted-denominator rational approximants* [28]. For a Hermitian negative definite matrix A an optimal choice of b_1 based

on uniform scalar rational approximation of e^z on $(-\infty, 0]$ has been proposed in [12]. For further information about the choice of parameters in rational Krylov methods we refer to [17] and the references therein.

In view of the above discussion it is justified to interpret `sexpmv` as a rational Krylov method, however, without the need for orthogonalization of basis vectors or the evaluation of e^{A_d}. Moreover, `sexpmv` is non-iterative in nature and has the choice of parameters b_i and s_i built into it. It can therefore be viewed as a black-box method. One advantage of the rational Arnoldi method over `sexpmv`, described in [6] for a different class of matrix functions, is its potential for spectral adaptivity. This means that isolated eigenvalues are implicitly deflated by some of the eigenvalues of A_d (the so-called rational Ritz values) and hence do not contribute to the approximation error. While such effects can lead to considerable convergence speedup of the rational Arnoldi method, they are matrix-dependent and not well-understood for non-Hermitian matrices. Besides that, the rational Arnoldi method is not straightforwardly parallelized [34] and has a higher memory consumption (for storing of V_{d+1}) than `sexpmv`. We will compare `sexpmv` with a rational Krylov method in section 7.3.2.

7. Numerical experiments. All experiments are conducted in MATLAB version R2013a on a Core i7 machine with 16GB RAM. We refer to the *spread* of A as the largest distance between any two eigenvalues. We use random numbers generated by `randn` in MATLAB, seeded with `rng(0)` for reproducibility. We test `sexpm` on a collection of problems of dimensions ranging from 10 to above 10^4. Although matrices of dimension as small as 10 are not our main focus, we are able to compute the "exact" exponentials of such matrices using MATLAB's variable precision features. We compare `sexpm` with MATLAB's `expm` and its improved version `expm_new` [2].

In our experiments the shifts are chosen so that the rightmost eigenvalue is 0. The results are nearly identical as long as the rightmost eigenvalue is $O(1)$, as we demonstrate in section 7.2.2. To obtain an estimate of $\|A\|_2$, we always use the MATLAB command `normest(A,0.3)`. Such rough estimate is sufficient for choosing the parameters and the cost for the estimation is typically negligible.

7.1. Computing the full matrix exponential e^A.

7.1.1. Varying eigenvalue spread. We generate fifty test matrices $A = X^{-1}\Lambda X$, where $X \in \mathbb{C}^{50\times 50}$ is fixed and obtained by `randsvd` with $\kappa_2(X) = 10$, and Λ forms a geometric sequence so that the eigenvalues have widely varying spread $10^{i/5}$ for $i = 1 : 50$ with the rightmost eigenvalue 0. We set the imaginary parts of each eigenvalue to be `randn/20` times the real part.

Figure 7.1 shows the errors and costs, in which the x-labels represent the matrices, arranged in increasing order of $\kappa_{\exp}(A)$. The "exact" solution e^A was computed by MATLAB's variable precision arithmetic `vpa` at 32 digit precision. Here and below the solid black lines show $u\kappa_{\exp}(A)$, a measure of the error accepted from a forward stable algorithm. We computed $\kappa_{\exp}(A)$ by the code `funm_condest1` available at [20] (see [1] for the algorithm). We also show $u\|A\|_2$, which by (1.2) is a lower bound for $\kappa_{\exp}(A)$ up to the difference in the choice of norms.

The error plot illustrates that all the methods perform in a forward stable manner, except when $\|A\|_2$ is $O(100)$. The error plot of `sexpm` shows a stagnation for matrices 40–50 at about 10^{-13}, which have small norms. This is due to subtractive cancellation effects, as described in section 4.1.2. As we explained there, this could be cured by using the product form instead of partial fractions, at the expense of increased cost.

The error plot of `sexpm` exhibits small jumps, which correspond to the values of

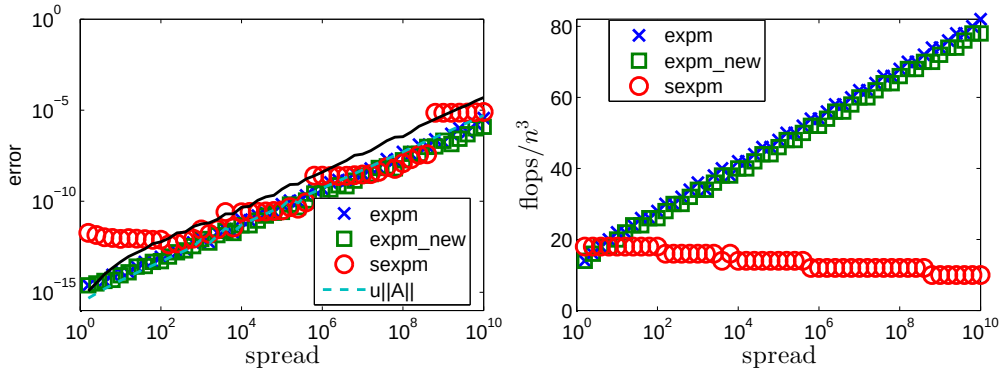

FIG. 7.1. *Results with matrices with near-real eigenvalues. Left: error* $\|fl(e^A) - e^A\|_F/\|e^A\|_F$, *along with the accepted error from a forward stable algorithm* $u\kappa_{\exp}(A)$ *(solid black line). Right: arithmetic cost/*n^3.

$\|A\|_2$ at which the Padé parameters are reduced. One may wonder what happens if the parameters are fixed at the largest degree $(4, 5)$ Padé. With such choice the errors become nearly identical to those of `expm` and `expm_new`. However, `sexpm` chooses the smallest s and m to reduce the cost without losing its forward stability (as the errors are never significantly above the black solid line $u\kappa_{\exp}(A)$).

The cost plot shows that `sexpm` is much more efficient than `expm` and `expm_new` when $\|A\|_2 \gg 1$.

7.1.2. Varying condition number of eigenvector matrix. Here we generate test matrices $A = X^{-1}\Lambda X$ of size $n = 50$, where now Λ is fixed to have spread 100 and X is obtained by `gallery('randsvd',n,k)` with widely varying condition numbers `k` taking logspaced values between 1 and 10^6.

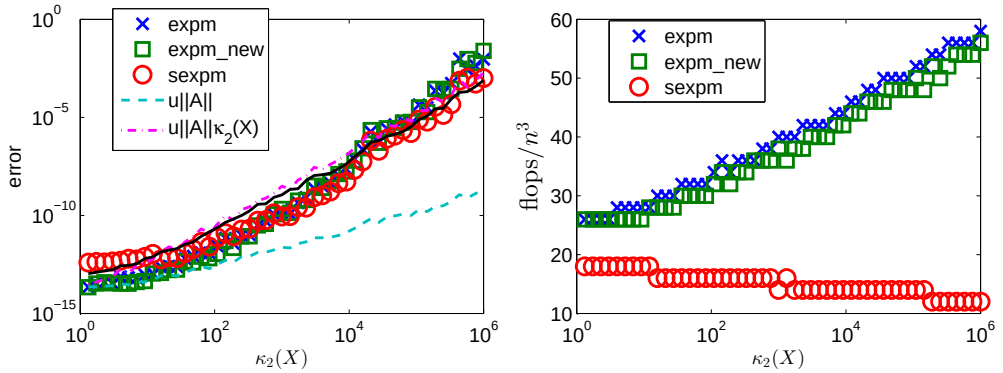

FIG. 7.2. *Results with matrices with real eigenvalues. Left: error* $\|fl(e^A) - e^A\|_F/\|e^A\|_F$, *along with the accepted error from a forward stable algorithm* $u\kappa_{\exp}(A)$ *(solid line), together with* $u\|A\|_2$ *(dashed line) and* $u\|A\|_2\kappa_2(X)$ *(dash-dotted line). Right: arithmetic cost/*n^3.

The error plot also shows $u\|A\|_2\kappa_2(X)$, and observe the remarkable agreement between the condition number $\kappa_{\exp}(A)$ and $u\|A\|_2\kappa_2(X)$. Recalling from (4.9) that $\|A\|_2 \le \kappa_{\exp}(A) \lesssim \|A\|_2\kappa_2(X)^2$, these experiments suggest that in practice $\alpha_A \approx 1$ is typically observed. Recalling the discussion in section 4.3, here we typically observe

$\|e^A\| \approx \kappa_2(X)$, and so $\alpha_A \gtrsim 1$ would indicate the forward stability of `sexpm` for diagonalizable matrices, provided no severe hump effect occurs. In this typical-case scenario `sexpm` is indeed behaving in a forward stable manner. We do not have a precise explanation for this observation, and leave it as an open problem.

7.1.3. Larger matrices. The previous experiments showed the costs in flops as the matrices were too small to measure runtime reliably. We now present runtime results for matrices of size $n = 2000$. The test matrices $A = X^{-1}\Lambda X$ have varying spread as described in section 7.1.1 and fixed eigenvectors with $\kappa_2(X) = 10$. Here computing the 'exact' e^A via variable precision arithmetic is infeasible, so we compute $X^{-1}e^{\Lambda}X$ instead. The results are given in Figure 7.3, in which we also show the results with the "naive" diagonalization-based method, which computes the eigendecomposition $A = X^{-1}\Lambda X$ via MATLAB's `eig`, and then takes $X^{-1}e^{\Lambda}X$.

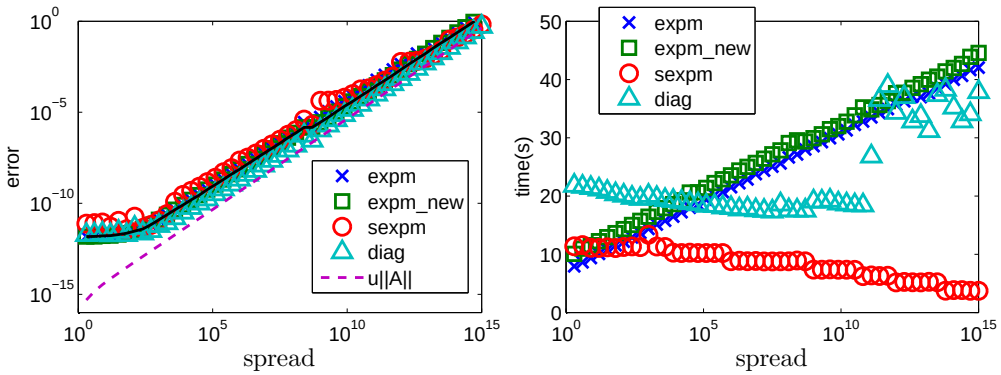

FIG. 7.3. 2000×2000 *matrices of varying spread. Left: Error. Right: Runtime(s).*

When $\|A\|_2 \gg 1$, `sexpm` is clearly the most efficient method giving about the same accuracy as the others, which all have a relative efficiency worsening with $\|A\|_2$.

Conversely, when $\|A\|_2 = O(1)$, the performance of the three methods does not differ much although `sexpm` is slightly slower than the other two. This is because when $\|A\|_2$ is small the standard methods also use a small scaling parameter and a low-degree Padé approximant, while `sexpm` involves complex arithmetic and does not employ an efficient scheme to evaluate the low-order Padé approximant.

7.1.4. Test matrices from Matrix Computation Toolbox. The previous experiments represented cases where `sexpm` is expected to work well, as those matrices are diagonalizable and have eigenvalues with small imaginary parts. We next test it with 10×10 matrices generated with the function `matrix` of the Matrix Computation Toolbox [19]. This set of test matrices includes difficult matrices such as those with nontrivial Jordan blocks, $\kappa_2(A) \gg 1$, or eigenvalues with large complex parts. The results are given in Figure 7.4.

Our method `sexpm` works fine for all matrices but one, for which it clearly gives an unstable result. We will discuss this problematic case in section 7.2.

Here again, using the product form instead of the partial fraction form reduces the errors of `sexpm` from about 10^{-13} to 10^{-15}.

7.2. Dealing with rightmost eigenvalues with large imaginary parts. The reason why `sexpm` failed for matrix 15 in the previous test is that this matrix has rightmost eigenvalues with widely varying imaginary parts as shown in Figure 7.5

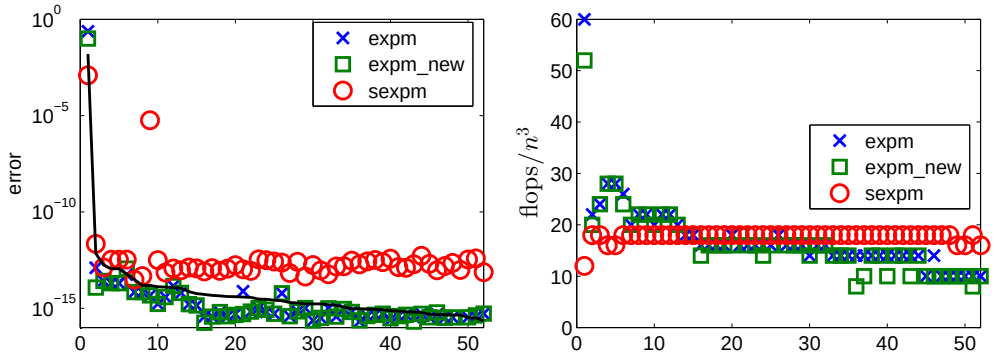

FIG. 7.4. *Test matrices from the toolbox. Left: error* $\|fl(e^A) - e^A\|_F / \|e^A\|_F$. *Right: flop count (accounting for* $O(n^3)$ *terms only) divided by* n^3.

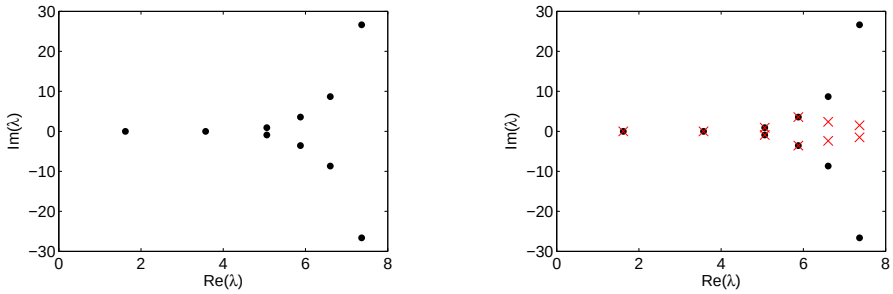

FIG. 7.5. *Left: eigenvalues of test matrix 15 for which* `sexpm` *fails. Note the presence of large imaginary parts in the rightmost eigenvalues. Right: same plot but with reduced imaginary parts of the rightmost eigenvalues, shown as red crosses. For this matrix the accuracy of* `sexpm` *is* $\approx 10^{-13}$. *The* x, y*-axes are not scaled equally in these plots.*

(left), and it is impossible to approximate e^z via low-degree rational functions across such a region. This is an inherent difficulty for any method based on (relatively) low-degree rational approximation, and `sexpm` is no exception. For example, if (after shifting so that $\mathrm{Re}(\lambda(A)) \le 0$) A has an eigenvalue at $\lambda_0 = 2000\pi i$, then in order to approximate the exponential in a convex region that contains both the origin and λ_0, the rational approximation needs to be of type (m, n) where $m + n \ge 2000$.

Here we mention three possible remedies for this difficulty. The first two are based on the fact that if $A = X\mathrm{diag}(\lambda_1, \ldots, \lambda_n)X^{-1}$, then e^A is equal to e^B for any matrix B of the form $B = A + 2\pi\mathrm{i}X\mathrm{diag}(k_1, \ldots, k_n)X^{-1}$, where k_i are integers.

Computing such B can be done as follows. Suppose that $\boldsymbol{y}_i$ and $\boldsymbol{x}_i$ are left and right eigenvectors of A corresponding to a simple eigenvalue λ_i, i.e., $A\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$, $\boldsymbol{y}_i^* A = \lambda_i \boldsymbol{y}_i^*$. Then the matrix

$$A + \Delta\lambda_i \frac{\boldsymbol{x}_i \boldsymbol{y}_i^*}{\boldsymbol{y}_i^* \boldsymbol{x}_i} \tag{7.1}$$

has the same eigenvalues as A except that λ_i is moved to $\lambda_i + \Delta\lambda_i$, and the eigenvectors are all preserved. Verifying this is easy using the orthogonality of left and right eigenvectors corresponding to distinct eigenvalues: $\boldsymbol{y}_j^* \boldsymbol{x}_i = 0$ if $\lambda_i \ne \lambda_j$. The above formula is well-defined so long as λ_i is a simple eigenvalue, for then $\boldsymbol{y}_i^* \boldsymbol{x}_i \ne 0$.

For the purpose of our algorithm, it is desirable to find $\Delta\lambda_i = 2\pi k_i \mathrm{i}$ such that the rightmost eigenvalues of B have small imaginary parts. Shifting them by multiples of $2\pi\mathrm{i}$ we can concentrate them within the strip $(-\pi, \pi]$. By (7.1), this can be done by computing the rightmost eigenvalues and their corresponding eigenvectors, both left and right. For example, the right plot of Figure 7.5 shows the eigenvalues after moving four rightmost eigenvalues to within the strip $(-\pi, \pi]$. This involves computing four eigenpairs, and improves the accuracy of `sexpm` to about 10^{-13} (we get $\approx 10^{-11}$ accuracy by moving two rightmost eigenvalues).

This process is effective when there are just a few rightmost eigenvalues with large imaginary parts, and computing their corresponding left and right eigenvectors is feasible (for example via a sparse eigenvalue solver, or from the Schur decomposition if available).

The second remedy is based on recent work by Aprahamian and Higham [4] which investigates the matrix unwinding function $\mathcal{U}(A)$, the matrix analogue of the scalar unwinding number, and shows that the exponentials of A and $A - 2\pi\mathrm{i}\mathcal{U}(A)$ are the same, and the imaginary parts of the eigenvalues of $A - 2\pi\mathrm{i}\mathcal{U}(A)$ lie in the interval $(-\pi, \pi]$. This effectively carries out the strategy just described for all eigenvalues. If we had an efficient method to compute $\mathcal{U}(A)$, computing the exponential of any A reduces to computing the exponential of a matrix whose eigenvalues have imaginary parts bounded by π in absolute value, leading to an ideal situation for `sexpm`. A similar idea was implemented by Ng [30]. The algorithm in [4] for computing $\mathcal{U}(A)$ is based on the Schur–Parlett recurrence and requires at least $28n^3$ flops, making this method of initial argument reduction the dominant cost. Its flop count will only be competitive with the standard `expm` if $\|A\| \geq O(2^{14})$, as each increase in scaling s results in $2n^3$ additional flops.

The third and probably simplest remedy is to use `sexpm` for computing $e^{A/s'}$ and then form $(e^{A/s'})^{s'}$. This corresponds to implicitly increasing the scaling factor; alternatively we can take s larger until the leftmost eigenvalues have imaginary parts $O(1)$ (but $\|A/2^s\| \gg 1$ is still allowed). In the vector case one may run `sexpmv` s' times, effectively computing $e^{A/s'}(\cdots(e^{A/s'}(e^{A/s'}\boldsymbol{b})))$. As this increases the computational cost by the factor s', it is only practical for moderate s'.

7.2.1. A problem from Cleve Moler's blog. Another difficult problem where `expm` gives completely inaccurate results with error 10^{18} is discussed by Moler [26]. This is due to overscaling, and `expm_new` resolves this problem giving error $\approx 2.2 \times 10^{-13}$. This matrix has norm $\|A\|_2 \approx 2.8 \times 10^{10}$, and `sexpm` yields error $\approx 9.5 \times 10^{-5}$ with scaling parameter $s = 2$ and a type $(3, 4)$ Padé approximant, which is still a forward stable result:

```
>> A = [             0 , 1/100000000 ,           0 ; ...
        -60200000000/3 ,          -3 , 20000000000 ; ...
                 200/3 ,           0 ,      -200/3 ];
>> sexpm(A)
ans =
     4.468493164532867e-01     1.540441841318744e-09     4.628116081523461e-01
    -5.742573897307440e+06    -1.528323594233649e-02    -4.527038378149220e+06
     4.477214802251823e-01     1.542705360507822e-09     4.634821497409599e-01
```

In experiments not reported here we have observed that `sexpm` would give an error of 1.8×10^{-12} if we choose $s = 4$, but as discussed in the previous sections, this is artificially accurate for a matrix of such large norm.

7.2.2. Accuracy dependence on the shift parameter. As noted in section 1.1, we employ an initial shifting $A \leftarrow A - \sigma I$ so that the rightmost eigenvalues of A have real part approximately 0. This requires an estimate or prior knowledge of the rightmost eigenvalues of the matrix, which may not be readily available. Here we examine the effect of misestimating σ on the accuracy of the computed e^A.

We take $n = 100$ and fixed the eigenvector matrix to have $\kappa_2(X) = 10^3$. The eigenvalues have a spread of 10^i for $i = 3:7$, with the rightmost eigenvalue being 0. This leads to 5 matrices A_i with norms $\|A_i\|_2 \approx 200 \times 10^i$. We vary σ over take 100 equispaced points on $[-10, 10]$ and run `sexpm` on $A_i - \sigma I$. Figure 7.6 shows the errors of the computed solutions.

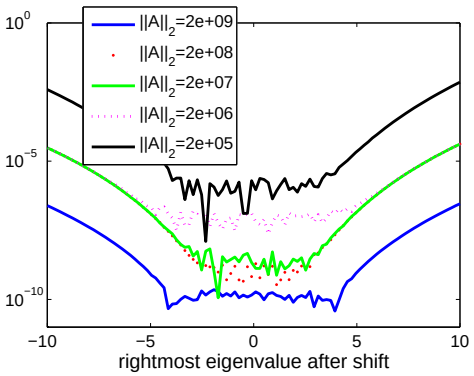

FIG. 7.6. *Errors as the initial shift is varied.*

Observe that for each i, the error does not deteriorate until $|\sigma| \gtrapprox 3$. This indicates that the shift σ does not have to be an accurate estimate of the rightmost eigenvalue, but an $O(2^s) = O(1)$ estimate is sufficient. Moreover, due to the conditioning $\kappa_{\exp}(A) \geq \|A\|$, a lower accuracy in σ becomes acceptable when $\|A\| \gg 1$.

7.3. Vector problem. We now test `sexpmv` for computing the action on a vector $e^A\boldsymbol{b}$. For this problem we mainly compare with `expmv` from [3].

7.3.1. Varying spread. We take test matrices as in section 7.1.1. The right part of Figure 7.7 illustrates the cost in logarithmic scale as follows: For `sexpmv` and `expmv`, it shows the number of matrix-vector products and linear system solves. For `expm_new`, it shows the number of total matrix-matrix multiplications and inversions (this is for reference only as full matrix inversions are to be avoided for computing $e^A\boldsymbol{b}$). We recall that only $m \leq 4$ LU factorizations are required for `sexpmv`.

While matrix-vector multiplication is usually much cheaper than solving a linear system (depending on the structure or sparsity of A), the cost of `expmv` generally grows rapidly with $\|A\|$, while that of `sexpmv` does not. Hence `sexpmv` is recommended whenever $\|A\| \gg 1$ and solving shifted linear systems $(A - bI)\boldsymbol{x} = \boldsymbol{b}$ is feasible.

7.3.2. A convection–diffusion problem. We consider the convection–diffusion problem described in [35], which has become a popular benchmark for discretizations of convection-dominated problems. The problem involves the computation of $\exp(-M^{-1}K)\boldsymbol{b}$, where $\{K, M\} \in \mathbb{R}^{n\times n}$ are finite element matrices of size $n = 2912$ (these matrices have been created for a similar test in [16, sec. 9.3]). The generalized eigenvalues of (K, M) are shown on the left of Figure 7.8. They all lie in

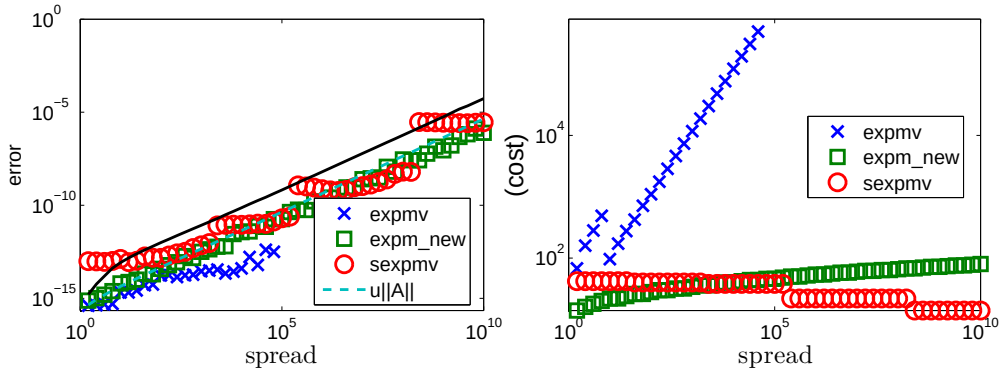

FIG. 7.7. *Left: error* $\|fl(e^A\boldsymbol{b}) - e^A\boldsymbol{b}\|_F/(\|e^A\|_F\|\boldsymbol{b}\|_2)$. *Right: for* `expm_new`*, the number of total matrix-matrix multiplications and inversions. For* `sexpmv` *and* `expmv`*, the number of matrix-vector products and linear system solves. Note the logarithmic scale.*

the left half-plane. It is straightforward to modify `sexpmv` to deal with a matrix $A = -M^{-1}K$ without inverting M by noting that $(A - b_iI)\boldsymbol{x} = \boldsymbol{b}$ is equivalent to $(K + b_iM)\boldsymbol{x} = M\boldsymbol{b}$. In all reported timings we reuse matrix LU factorizations of $(K + b_iM)$ whenever possible.

The runtimes and relative errors $\|e^A\boldsymbol{b} - fl(e^A\boldsymbol{b})\|_2/\|e^A\boldsymbol{b}\|_2$ achieved by the tested methods are shown in Table 7.1. Here `expmv` is not efficient because $\|A\|_2 = \|M^{-1}K\|_2 \approx 1.76\cdot 10^4 \gg 1$, balancing does not reduce the norm, and the quantity $\|A^p\|_2^{1/p}$ does not decrease much as p grows. The function `expmnew*b` computes the whole matrix e^A and hence is much slower than `sexpmv` for this problem. On the other hand, `sexpmv` gives slightly less accuracy than `expmnew*b`. This is due to the presence of eigenvalues $\approx -11.96 \pm 18.64$i. This issue can be resolved by any of the techniques discussed in section 7.2. We have tried the third technique which is to effectively halve the imaginary parts by using the identity $e^A\boldsymbol{b} = e^{A/2}(e^{A/2}\boldsymbol{b})$; this roughly doubles the cost but gives an accurate solution as Table 7.1 shows.

We also compare with the rational Arnoldi method (see section 6.5) using the orthogonalization algorithm `rat_krylov` implemented in the Rational Krylov Toolbox [7]. We try three different choices for the shift parameters b_i. The first and second choices correspond to the shift-and-invert Arnoldi method with constant shifts $b_i = 10$ or $b_i = 100$ $(i = 1, \ldots, 64)$, respectively. In the third choice we use as shifts the 64 poles of the scaled-and-squared Padé approximant $r_{k,m}(z/2^s)^{2^s}$ with parameters $k = 3$, $m = 4$, and $s = 4$. These are exactly the parameters employed by `sexpmv` for this problem. With this choice of shift parameters, the vector $r_{k,m}(A/2^s)^{2^s}\boldsymbol{b}$ computed by `sexpm` will be an element of the rational Krylov space $\mathcal{Q}_{64}(A, \boldsymbol{b})$. Note from Table 7.1 that `rat_krylov` is slightly faster with the Padé poles than with the constant choice of poles. Although in the latter case only one matrix factorization of the real matrix $(K - bM)$ is required, and in the former case two complex-valued matrices $(K - b_1M)$ and $(K - b_2M)$ need to be factorized, `rat_krylov` can exploit that the Padé shifts appear in complex conjugate pairs. This effectively halves the number of required linear system solves to 32.

The convergence of the three rational Arnoldi variants is shown on the right of Figure 7.8. It is interesting to see that both the constant choice $b_i = 100$ and the Padé shifts require almost the full 64 iterations to reach the stagnation accuracy. We

TABLE 7.1
Comparison of different methods for the convection–diffusion problem.

	runtime	relative error
`expmnew*b`	38	2.5e-13
`expmv`	×	×
`sexpmv`	0.2	1.6e-10
`sexpmv (x2)`	0.4	8.1e-13
`rat_krylov` (all shifts $b_i = 10$)	0.9	1.3e-10
`rat_krylov` (all shifts $b_i = 100$)	0.9	2.9e-12
`rat_krylov` (Padé shifts)	0.6	1.0e-12

do not claim that $b_i = 100$ is an optimal choice for a single shift, but by varying this shift manually we could not identify a choice that would lead to much faster convergence. This exemplifies the difficulty encountered with rational Krylov methods when choosing the shift parameters (at least for non-Hermitian A), and also shows that our subdiagonal Padé approximant is good in the sense that it is not outperformed by the shift-and-invert Arnoldi method. In terms of timings shown in Table 7.1, `sexpmv` is in fact slightly faster than `rat_krylov` as it does not require vector orthogonalizations.

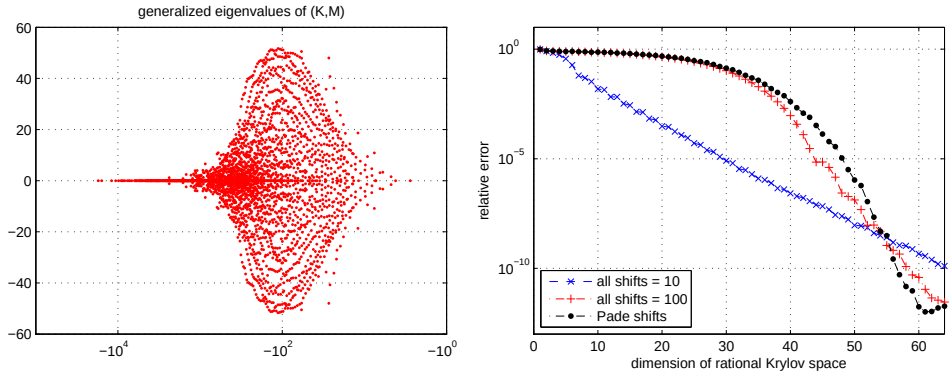

FIG. 7.8. *Left: generalized eigenvalues of (K, M) for the convection–diffusion problem. Right: convergence of the rational Arnoldi method run with three different choices for the shift parameters.*

8. Conclusion. By reexamining the rational approximant employed by the scaling and squaring method, we developed a new method `sexpm` for computing e^A and $e^A\boldsymbol{b}$ based on subdiagonal Padé approximants. `sexpm` and its vector variant `sexpmv` are effective when the rightmost eigenvalues do not have widely varying imaginary parts, and is efficient especially when $\|A\| \gg 1$. In Table 8.1 we summarize the recommended algorithm depending on (i) whether n is large (so a Schur decomposition is infeasible) and (ii) whether $|\mathrm{Im}(\mathrm{eig}(A))| \gg 1$ or not (or more specifically the imaginary parts of the rightmost eigenvalues).

TABLE 8.1
Recommended algorithm for e^A depending on size and $|\mathrm{Im}(\mathrm{eig}(A))|$ of the rightmost eigenvalues.

	$\lvert\mathrm{Im}(\mathrm{eig}(A))\rvert$ small	$\lvert\mathrm{Im}(\mathrm{eig}(A))\rvert$ large
n small to medium	`sexpm` or Schur+`sexpm`	reduce + `sexpm` or Schur–Parlett
n large	`sexpm`	reduce + `sexpm`

We established the forward stability of our algorithm for normal matrices. Our experiments indicated that it performs in a forward stable manner for a variety of nonnormal test matrices, although a precise statement is an open problem.

Acknowledgements. We thank Mary Aprahamian and Nick Higham for reading the manuscript and providing many helpful suggestions. We also thank Nick Trefethen and Charles Van Loan for their insightful comments, and Sam Relton for his help with the condition number estimation.

REFERENCES

[1] A. H. Al-Mohy and N. J. Higham, *Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation*, SIAM J. Matrix Anal. Appl., 30 (2009), pp. 1639–1657.
[2] A. H. Al-Mohy and N. J. Higham, *A new scaling and squaring algorithm for the matrix exponential*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 970–989.
[3] A. H. Al-Mohy and N. J. Higham, *Computing the action of the matrix exponential, with an application to exponential integrators*, SIAM J. Sci. Comp., 33 (2011), pp. 488–511.
[4] M. Aprahamian and N. J. Higham, *The matrix unwinding function, with an application to computing the matrix exponential*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 88–109.
[5] M. Arioli, B. Codenotti, and C. Fassino, *The Padé method for computing the matrix exponential*, Linear Algebra Appl., 240 (1996), pp. 111–130.
[6] B. Beckermann and S. Güttel, *Superlinear convergence of the rational Arnoldi method for the approximation of matrix functions*, Numer. Math., 121 (2012), pp. 205–236.
[7] M. Berljafa and S. Güttel, *A Rational Krylov Toolbox for MATLAB*, MIMS EPrint 2014.56, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2014. Available for download at `http://guettel.com/rktoolbox/`.
[8] M. Crouzeix, *Numerical range and functional calculus in Hilbert space*, J. Func. Anal., 244 (2007), pp. 668–690.
[9] L. Dieci and A. Papini, *Padé approximation for the exponential of a block triangular matrix*, Linear Algebra Appl., 308 (2000), pp. 183–202.
[10] L. Dieci and A. Papini, *Conditioning of the exponential of a block triangular matrix*, Numer. Algorithms, 28 (2001), pp. 137–150.
[11] H. C. Elman and M. Wu, *Lyapunov inverse iteration for computing a few rightmost eigenvalues of large generalized eigenvalue problems*, Tech. Report TR-5009, University of Maryland Department of Computer Science, 2012.
[12] J. Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comp., 27 (2006), pp. 1438–1457.
[13] W. Fair and Y. L. Luke, *Padé approximations to the operator exponential*, Numer. Math., 14 (1970), pp. 379–382.
[14] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by Krylov approximation methods*, SIAM J. Sci. Stat. Comp., 13 (1992), pp. 1236–1264.
[15] V. Grimm, *Resolvent Krylov subspace approximation to operator functions*, BIT, 52 (2012), pp. 639–659.
[16] S. Güttel, *Rational Krylov Methods for Operator Functions*, PhD thesis, TU Bergakademie Freiberg, Germany, 2010.
[17] S. Güttel, *Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection*, GAMM-Mitt., 36 (2013), pp. 8–31.
[18] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations*, vol. 2, Springer, 1991.
[19] N. J. Higham, *The Matrix Computation Toolbox*. Available online at `http://www.ma.man.ac.uk/~higham/mctoolbox`.
[20] N. J. Higham, *The Matrix Function Toolbox*. `http://www.ma.man.ac.uk/~higham/mftoolbox`.
[21] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, USA, second ed., 2002.
[22] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, PA, USA, 2008.
[23] N. J. Higham, *The scaling and squaring method for the matrix exponential revisited*, SIAM Rev., 51 (2009), pp. 747–764.

[24] C. S. Kenney and A. J. Laub, *A Schur–Fréchet algorithm for computing the logarithm and exponential of a matrix*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 640–663.
[25] K. Meerbergen and D. Roose, *Matrix transformations for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems*, IMA J. Numer. Anal., 16 (1996), pp. 297–346.
[26] C. Moler, *A balancing act for the matrix exponential*, July 2012. `http://blogs.mathworks.com/cleve/2012/07/23/a-balancing-act-for-the-matrix-exponential/`.
[27] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.
[28] I. Moret and P. Novati, *RD-rational approximations of the matrix exponential*, BIT, 44 (2004), pp. 595–615.
[29] M. Mori, *Approximation of exponential function of a matrix by continued fraction expansion*, Publications of the Research Institute for Mathematical Sciences, 10 (1974), pp. 257–269.
[30] K. C. Ng, *Contributions to the Computation of the Matrix Exponential*, PhD thesis, University of California, Berkeley, 1984.
[31] P. Novati, *Using the restricted-denominator rational Arnoldi method for exponential integrators*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1537–1558.
[32] A. Ruhe, *Rational Krylov algorithms for nonsymmetric eigenvalue problems*, IMA Vol. Math. Appl., 60 (1994), pp. 149–164.
[33] T. Schmelzer and L. N. Trefethen, *Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals*, Electron. Trans. Numer. Anal, 29 (2007), pp. 1–18.
[34] D. Skoogh, *A parallel rational Krylov algorithm for eigenvalue computations*, in Applied Parallel Computing Large Scale Scientific and Industrial Problems, B. Kågström, J. Dongarra, E. Elmroth, and J. Wasniewski, eds., vol. 1541 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 521–526.
[35] R. M. Smith and A. G. Hutton, *The numerical treatment of advection: A performance comparison of current methods*, Numer. Heat Transfer, 5 (1982), pp. 439–461.
[36] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013.
[37] L. N. Trefethen and M. H. Gutknecht, *The Carathéodory-Fejér method for real rational approximation*, SIAM J. Numer. Anal., 20 (1983), pp. 420–436.
[38] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer, *Talbot quadratures and rational approximations*, BIT, 46 (2006), pp. 653–670.
[39] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comp., 27 (2006), pp. 1438–1457.
[40] C. Van Loan, *The sensitivity of the matrix exponential*, SIAM J. Numer. Anal., 14 (1977), pp. 971–981.
[41] H. Van Rossum, *On the poles of Padé approximations to e^z*, Nieuw Archief voor Wiskunde, 19 (1971), p. 37.
[42] R. S. Varga, *Matrix Iterative Analysis*, vol. 27, Springer, 2009.
[43] G. Wanner, E. Hairer, and S. Nørsett, *Order stars and stability theorems*, BIT, 18 (1978), pp. 475–489.
[44] R. C. Ward, *Numerical computation of the matrix exponential with accuracy estimate*, SIAM J. Numer. Anal., 14 (1977), pp. 600–610.
[45] J. A. C. Weideman and L. N. Trefethen, *Parabolic and hyperbolic contours for computing the Bromwich integral*, Math. Comp., 76 (2007), pp. 1341–1356.