MIXED-PRECISION PATERSON–STOCKMEYER METHOD FOR EVALUATING POLYNOMIALS OF MATRICES*

XIAOBO LIU†

Abstract. The Paterson–Stockmeyer method is an evaluation scheme for matrix polynomials with scalar coefficients that arise in many state-of-the-art algorithms based on polynomial or rational approximation, for example, those for computing transcendental matrix functions. We derive a mixed-precision version of the Paterson–Stockmeyer method that is particularly useful for evaluating matrix polynomials with scalar coefficients of decaying magnitude. The key idea is to perform computations on data of small magnitude in low precision, and rounding error analysis is provided for the use of lower-than-working precisions. We focus on the evaluation of the Taylor approximants of the matrix exponential and show the applicability of our method to the existing scaling and squaring algorithms, particularly when the norm of the input matrix (which in practical algorithms is often scaled towards to origin) is sufficiently small. We also demonstrate through experiments the general applicability of our method to the computation of the polynomials from the Padé approximant of the matrix exponential and the Taylor approximant of the matrix cosine. Numerical experiments show our mixed-precision Paterson–Stockmeyer algorithms can be more efficient than its fixed-precision counterpart while delivering the same level of accuracy.

Key words. Paterson–Stockmeyer method, mixed-precision algorithm, rounding error analysis, polynomial evaluation, polynomial of matrices, matrix function

MSC codes. 15A16, 65F60, 65G50

1. Introduction.

The Paterson–Stockmeyer (PS) method [19] is an evaluation scheme for matrix polynomials with scalar coefficients that is used in many state-of-the-art algorithms based on polynomial or rational approximants for computing transcendental functions of matrices, for example, the matrix exponential [8], [21], the matrix logarithm [7], and the matrix trigonometric and hyperbolic functions and their inverses [2], [3], [4], [20]. In the PS scheme, a matrix polynomial $p_m(X) = \sum_{i=0}^{m} b_i X^i$ at $X \in \mathbb{C}^{n\times n}$ is written as

$$p_m(X) = \sum_{i=0}^{r} B_i \cdot (X^s)^i, \quad r = \lfloor m/s \rfloor, \tag{1.1}$$

where s is an integer parameter and

$$B_i = \begin{cases} \displaystyle\sum_{j=0}^{s-1} b_{si+j} X^j, & i = 0, \dots, r-1, \\ \displaystyle\sum_{j=0}^{m-sr} b_{sr+j} X^j, & i = r. \end{cases}$$

The first s positive powers of X are computed once the parameter s is chosen; then (1.1) is evaluated by the matrix version of Horner's method, with each coefficient polynomial B_i formed via explicit powers reusing the computed powers of X. In the evaluation of polynomials of matrices, matrix multiplications have the highest asymptotic cost amongst all the matrix operations, so it is sensible to measure the

*Version of December 23, 2023.

Funding: This work was supported by the Royal Society.

†Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (xiaobo.liu@manchester.ac.uk).

efficiency of an evaluation scheme by the number of matrix multiplications required. This quantity is known to be minimised by setting $s = \sqrt{m}$ (which is not necessarily an integer) for a given matrix polynomial $p_m(X)$, and the practical choices of $s = \lfloor\sqrt{m}\rfloor$ or $s = \lceil\sqrt{m}\rceil$ yield exactly the same cost [6], [9, pp. 29–30], which is about $2\sqrt{m}$. The above discussion is most relevant to the case when the polynomial p is dense (most of the coefficients b_i are nonzero), and the economics of the evaluation can be rather different if p_m is sparse, which is not the focus of the work.

One downside of the PS method is that it requires $(s+2)n^2 = O(\sqrt{m}n^2)$ memory locations including the storage of the first s powers of X, in contrast to the n^2 elements of storage from Horner's method. Van Loan [17] proposed a modification of the PS method which reduces the storage requirement to $4n^2$ by computing p_m one column at a time, at the price of about 40% extra flops. A more efficient block variant of Van Loan's algorithm is developed in [16], where it is shown that the computation of all the three mentioned schemes can be accelerated by reducing the argument matrix to its Schur form if the degree m is sufficiently large—so the savings from performing matrix multiplications between triangular (instead of full) matrices outweigh the extra costs in reducing the matrix to Schur form.

In this work we aim to utilize multiple precisions in the computation of p by the PS method so as to achieve $\|p - \widehat{p}\| \lesssim cnu\|p\|$, where c is some mild constant, given that its scalar coefficients b_i enjoy a certain fast decaying property. Our idea is inspired by the fundamental fact that computations performed on data of small magnitude can use low precision. For example, in the computation of $X = C + AB$ where $|A||B| \ll |C|$ then the matrix product AB can be computed in lower precision than the subsequent summation without significantly impacting the overall accuracy.

We begin in section 2 by stating the main theorems which are the building block for analysing the errors in the evaluation of $p_m(X)$ in (1.1) and discuss the evaluation scheme following from the error analysis. In section 3, we apply the framework derived in the previous section to Taylor approximants of the matrix exponential and show the applicability of our method to existing scaling and squaring algorithms for that function, particularly when the norm of the input matrix is sufficiently small, in which case accuracy of the mixed-precision method is shown to be guaranteed. Numerical experiments are presented to demonstrate the accuracy and potential efficiency of the algorithms. In section 4, we illustrate with examples the general applicability of our framework to the computation of matrix polynomials with scalar coefficients that decay in modulus. Conclusions are drawn in section 5.

Throughout this work we denote by $\|\cdot\|$ any consistent matrix norm, by $\mathbb{N}$ the set of nonnegative integers, and by $\mathbb{N}^+$ the set of positive integers. We denote by u the unit roundoff of the floating-point arithmetic. An inequality expressed as "$a \ll b$" can be read as "a is sufficiently less than b".

2. Rounding error analysis and evaluation scheme. We use the standard model of floating-point arithmetic [10, sect. 2.2]

$$\mathrm{fl}(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \quad |\delta| \le u, \tag{2.1}$$

where x and y are floating-point numbers and op denotes addition, subtraction, multiplication, or division. For matrix multiplication, we have [10, sect. 3.5]

$$\mathrm{fl}(AB) = AB + E, \quad |E| \le \gamma_n |A||B|, \tag{2.2}$$

where $\gamma_n := nu/(1-nu)$, assuming $nu < 1$.

If more than one precision is involved in a computation, we will use the operator $\mathrm{fl}_j(\cdot)$ to denote an operation executed in precision u_j and $\gamma_n^j := nu_j/(1-nu_j)$, assuming $nu_j < 1$. Define $\theta_{i,j} = u_i/u_j$, $i,j = 0\colon r$, so we have

$$\gamma_n^i = \gamma_{n\theta_{i,j}}^j. \tag{2.3}$$

The evaluation of $p_m(X)$ in (1.1) is customarily performed via Horner's method, that is, we compute

$$p_m(X) = B_0 + X^s\Big(B_1 + X^s\Big(B_2 + X^s\big(B_3 + \cdots + X^s(B_{r-1} + X^s B_r)\big)\Big)\Big) \tag{2.4}$$

starting from the quantities in the innermost brackets. In this paper we are most interested in the case where the $|b_i|$ decay quickly, so we have, for some positive integer $\nu \in [1, r]$,

$$\|B_i\|\|X^s\| = \tau_i\|B_{i-1}\|, \quad \tau_i \ll 1, \quad i = \nu\colon r. \tag{2.5}$$

This set of conditions can be written as, for $i = \nu\colon r$,

$$\|b_{si}I + b_{si+1}X + \cdots + b_{si+s-1}X^{s-1}\|\|X^s\| \ll \|b_{si-s}I + b_{si-s+1}X + \cdots + b_{si-1}X^{s-1}\|,$$

which will hold if $\|X\|$ is sufficiently small. The intuition is that the dominant terms in B_i and B_{i-1} are linear combination of powers of X generally from the same set with each pair of corresponding scalar coefficients being s indices apart from the series $\{b_i\}$, of which the modulus decays rapidly. Consider simply $X = \left[\begin{smallmatrix} -1 & 1 \\ 2 & 1 \end{smallmatrix}\right]$ with $b_i = 1/i!$ and $s = 6$, for example; we have

$$\begin{aligned}\|B_2\|_1\|X^s\|_1 \approx \left\|\frac{1}{12!}I + \frac{1}{13!}X\right\|_1 \|X^s\|_1 &= 6.5 \times 10^{-8} \\ &\ll 1.8 \times 10^{-3} = \left\|\frac{1}{6!}I + \frac{1}{7!}X\right\|_1 \approx \|B_1\|_1,\end{aligned}$$

where the dominant terms in B_1 and B_2 are both from $\{I, X\}$. Later we will discuss to what extent the conditions (2.5) can hold for the polynomial $p_m(X)$.

Define the polynomial

$$q(X) := B_{\nu-1} + X^s\Big(B_\nu + X^s\big(B_{\nu+1} + \cdots + X^s(B_{r-1} + X^s B_r)\big)\Big), \tag{2.6}$$

which is exactly $p_m(X)$ if $\nu = 1$. Assuming that (2.5) is satisfied, our idea for computing $q(X)$ is to start with the lowest precision in forming the matrix product in the innermost brackets, and then gradually and adaptively increase the precision (up to the working precision) for the subsequent matrix products outwards, aiming to still deliver the full working precision accuracy for the computation of $q(X)$. The following theorem provides a rounding error bound on the forward error of the process, and the proof is given in Appendix A.

THEOREM 2.1. *If $\|\widehat{B}_i - B_i\| \le u_i\|B_i\|$, $i = \nu - 1\colon r$ and $\|\widehat{Y} - Y\| \le u_\nu\|Y\|$ where $Y \equiv X^s$, then for the matrix $\widehat{q}$, computed in finite precision, for $q(X)$ in (2.6), the evaluation scheme*

$\widehat{\varphi}_r = \widehat{B}_r$
for $j = r\colon -1\colon \nu$
  $\widehat{\varphi}_{j-1} = \mathrm{fl}_{j-1}(\widehat{B}_{j-1} + \mathrm{fl}_j(\widehat{Y}\widehat{\varphi}_j))$
end
$\widehat{q} = \widehat{\varphi}_{\nu-1}$

satisfies

$$\|\widehat{q}-q(X)\| \le \gamma_{f_r}^{\nu-1}\|X^s\|^{r-\nu+1}\|B_r\|+\gamma_{f_{r-1}}^{\nu-1}\|X^s\|^{r-\nu}\|B_{r-1}\|+\cdots+\gamma_{f_{\nu-1}}^{\nu-1}\|B_{\nu-1}\|, \tag{2.7}$$

where

$$\begin{aligned} f_r &= \frac{nu_r}{u_{\nu-1}} + \frac{(n+1)}{u_{\nu-1}}(u_{r-1}+u_{r-2}+\cdots+u_\nu)+1,\\ f_{r-1} &= \frac{(n+2)u_{r-1}}{u_{\nu-1}} + \frac{(n+1)}{u_{\nu-1}}(u_{r-2}+u_{r-3}+\cdots+u_\nu)+1,\\ &\vdots\\ f_\nu &= \frac{(n+2)u_\nu}{u_{\nu-1}}+1,\\ f_{\nu-1} &= 2. \end{aligned}$$

The constants f_i in (2.7) can be bounded above by

$$f_i \le (n+2)\left(\frac{u_i}{u_{\nu-1}}+\frac{u_{i-1}}{u_{\nu-1}}+\cdots+\frac{u_\nu}{u_{\nu-1}}\right)+1, \quad i=\nu\colon r.$$

If (2.5) is satisfied, then in Theorem 2.1 we choose the precisions

$$u_i = \frac{\|B_{\nu-1}\|u_{\nu-1}}{\|B_i\|\|X^s\|^{i-\nu+1}}, \quad i=\nu\colon r, \tag{2.8}$$

which implies $u_{i-1}=\tau_i u_i$, and take $\tau=\max_i \tau_i \ll 1$ to have

$$u_{i-1} \le \tau u_i, \quad i=\nu\colon r. \tag{2.9}$$

It follows that, for $i=\nu\colon r$,

$$\begin{aligned} \frac{u_{\nu-1}f_i}{u_i} &\le (n+2)\left(1+\frac{u_{i-1}}{u_i}+\cdots+\frac{u_{\nu-1}}{u_i}\right)\\ &\le (n+2)\left(1+\tau+\cdots+\tau^{i-\nu+1}\right)\\ &= n+2+O(n\tau), \end{aligned}$$

and therefore we have $f_i \lesssim (n+n\tau+2)\,u_i/u_{\nu-1}$ and thus

$$f_i\|B_i\|\|X^s\|^{i-\nu+1} \lesssim \frac{(n+n\tau+2)u_i}{u_{\nu-1}}\|B_i\|\|X^s\|^{i-\nu+1} = (n+n\tau+2)\|B_{\nu-1}\|.$$

Then, since $i\gamma_k^{\nu-1} \le \gamma_{ik}^{\nu-1}$ [10, Lem. 3.3] we have the bound

$$\begin{aligned} \|\widehat{q}-q(X)\| &\le \gamma_{f_r}^{\nu-1}\|X^s\|^{r-\nu+1}\|B_r\| + \gamma_{f_{r-1}}^{\nu-1}\|X^s\|^{r-\nu}\|B_{r-1}\| + \cdots + \gamma_{f_{\nu-1}}^{\nu-1}\|B_{\nu-1}\|\\ &\lesssim (r-\nu+1)\gamma_{(n+n\tau+2)\|B_{\nu-1}\|}^{\nu-1} + \gamma_2^{\nu-1}\|B_{\nu-1}\|\\ &\lesssim \frac{(r-\nu+1)(n+n\tau+2)+2}{1-(n+n\tau+2)\|B_{\nu-1}\|u_{\nu-1}}\|B_{\nu-1}\|u_{\nu-1}\\ &\approx \frac{(r-\nu+1)n}{1-(n+n\tau+2)\|B_{\nu-1}\|u_{\nu-1}}\|q(X)\|u_{\nu-1}. \end{aligned}$$

Therefore, if $((1+\tau)n+2)\,\|B_{\nu-1}\|u_{\nu-1} \ll 1$, then we can choose the precisions u_i by (2.8) such that the computed matrix $\widehat{q}$ of $q(X)$ has approximately a normwise relative error of $(r-\nu+1)nu_{\nu-1}$, where $r=\lfloor m/s\rfloor$, and, in particular, if (2.5) holds for $\nu=1$ then the computed matrix $\widehat{q}_m$ of $q_m(X)$ has approximately a normwise relative error of rnu_0.

For $s=1$ the PS scheme (2.4) reduces to Horner's method, in which case the conditions (2.5) become

$$|b_i|\|X\| = \tau_i|b_{i-1}|, \quad \tau_i \ll 1, \quad i=\nu\colon m, \tag{2.10}$$

for some positive integer $\nu\in[1,m]$. We can obtain an analogous result to Theorem 2.1. In this case it can be shown that if $((1+\tau)n+2)\,|b_{\nu-1}|u_{\nu-1}\ll 1$, then we can choose the precisions by

$$u_i = \frac{|b_{\nu-1}|u_{\nu-1}}{|b_i|\|X\|^{i-\nu+1}}, \quad i=\nu\colon m$$

such that the computed matrix $\widehat{q}$ of $q(X)$ has approximately a normwise relative error of $(m-\nu+1)nu_{\nu-1}$, and, in particular, if (2.10) holds for $\nu=1$ then the computed matrix $\widehat{q}_m$ of $q_m(X)$ has approximately a normwise relative error of mnu_0.

The requirement (2.10) is made between any two consecutive coefficients and it can only hold if $\|X\|$ is sufficiently small and the decay rate of $|b_i|$ is sufficiently large. On the other hand, the PS scheme with sufficiently large s can mitigate this potentially very strict requirement. The requirement in (2.10) is on adjacent coefficients b_{i-1} and b_i, but in (2.5) the dominant terms are s indices apart so the condition is more likely to be satisfied. Also, the error bound associated with the PS scheme is smaller than that of Horner's method by at most a factor of approximately $r/m=\lfloor m/s\rfloor/m\approx 1/s$. We henceforth focus on the PS evaluation scheme (2.4) in the general case (s is not necessarily equal to 1).

The framework of the mixed-precision PS scheme is that we exploit lower precisions $u_r\ge u_{r-1}\ge,\dots,\ge u_\nu$ in the computation of $q(X)$ and then perform the matrix products and sums in the evaluation of

$$p_m(X) = B_0 + X^s\Big(B_1 + X^s\Big(B_2+\cdots+X^s\big(B_{\nu-3}+X^s(B_{\nu-2}+X^s q(X))\big)\Big)\Big) \tag{2.11}$$

in the working precision $u:=u_{\nu-1}$. The required powers of X are formed explicitly and each B_i is formed by reusing these computed powers, which involves only matrix scaling and additions, so we will form the powers of X in the working precision u. From the earlier discussion, the computed matrix $\widehat{q}$ of $q(X)$ has approximately a normwise relative error bounded above by $(r-\nu+1)nu$, which is satisfactory for the evaluation of (2.11) in precision u. However, this error bound is from Theorem 2.1 and is subject to $\|\widehat{Y}-Y\|\le u_\nu\|Y\|$, where $Y\equiv X^s$, and $\|\widehat{B}_i-B_i\|\le u_i\|B_i\|$, $i=\nu-1\colon r$. Among the latter requirements, we just need to ensure that $\|\widehat{B}_{\nu-1}-B_{\nu-1}\|\le u\|B_{\nu-1}\|$ is satisfied by the choice of the precisions (2.8), as we do the matrix scaling and additions required in assembling the B_i in the working precision u. In fact, since $B_{\nu-1}$ is only involved in the final matrix summation in the evaluation of $q(X)$ via (2.6), it is not hard to see that we can ease the condition to

$$\|\widehat{B}_{\nu-1}-B_{\nu-1}\|\le cnu\|B_{\nu-1}\|, \tag{2.12}$$

where c denotes some mild constant, and still achieve the same error bound on the computed $\widehat{q}$.

We next provide rounding errors analysis for the computation of matrix powers and polynomials and discuss its practical implications for our use case.

2.1. Powers and polynomials of matrices. We first derive an upper bound on the rounding errors in the computation of X^k in a fixed precision. In our results, when we write $X \in \mathbb{R}^{n\times n}$ it is understood that X is a matrix of floating-point numbers.

LEMMA 2.2. *For $X \in \mathbb{R}^{n\times n}$ and $X_t = X^t$ with $t \in \mathbb{N}^+$, the computed matrix $\widehat{X}_t = \mathrm{fl}(X^t)$ obtained by repeated multiplication $\widehat{X}_k = \mathrm{fl}(\widehat{X}_{k-1}X)$, $k = 1\colon t$, $\widehat{X}_0 = X$ in precision u satisfies $|\widehat{X}_t - X^t| \le \gamma_{(t-1)n}|X|^t$.*

Proof. For $t = 2$ the bound holds by (2.2). Suppose the bound holds for $t = k-1$:

$$|\widehat{X}_{k-1} - X^{k-1}| \le \gamma_{(k-2)n}|X|^{k-1}.$$

We have, for $t = k$,

$$\begin{aligned}|\widehat{X}_k - X^k| &\le |\widehat{X}_k - \widehat{X}_{k-1}X| + |\widehat{X}_{k-1}X - X^k| \le \gamma_n|\widehat{X}_{k-1}||X| + |\widehat{X}_{k-1} - X^{k-1}||X| \\ &\le \gamma_n(1+\gamma_{(k-2)n})|X|^{k-1}|X| + \gamma_{(k-2)n}|X|^k \le \gamma_{(k-1)n}|X|^k,\end{aligned}$$

where we have used [10, Lem. 3.3], and so the proof is completed by induction. ∎

Based upon Lemma 2.2, we obtain the following result, which bounds the forward error of a polynomial formed by assembling matrix powers that have already been computed via repeated multiplication. The proof is straightforward by induction and is thus omitted.

THEOREM 2.3. *If the first t positive powers of X are formed by repeated multiplication in precision u (with $\widehat{X}_t = \mathrm{fl}(X^t)$ denoting the computed matrix) and $\psi = \sum_{j=0}^{t} a_j X^j$ is evaluated in precision u by*

$\widehat{\varphi}_0 = \mathrm{fl}(a_0 I)$
for $j = 1\colon t$
  $\widehat{\varphi}_j = \mathrm{fl}(\widehat{\varphi}_{j-1} + \mathrm{fl}(a_j \widehat{X}_j))$
end
$\widehat{\psi} = \widehat{\varphi}_t$

then the computed $\widehat{\psi}$ satisfies

$$|\widehat{\psi} - \psi(X)| \le \gamma_t|a_0|I + \sum_{j=1}^{t} \gamma_{(j-1)(n-1)+t+1}|a_j||X|^j. \tag{2.13}$$

The bound of the theorem can be pessimistic, in the sense that inequalities such as $|X^i| \le |X|^i$ are used in the derivation, but as an a priori bound it cannot be improved without further assumptions. The bound is immediately applicable to the computed polynomial $\widehat{B}_{\nu-1}$ in precision $u = u_{\nu-1}$, and we have

$$\begin{aligned}|\widehat{B}_{\nu-1} - B_{\nu-1}(X)| &\le \gamma_{s-1}|b_{s(\nu-1)}|I + \sum_{j=0}^{s-1} \gamma_{(j-1)(n-1)+s}|b_{s(\nu-1)+j}||X|^j \\ &\le \gamma_{(s-2)(n-1)+s} \sum_{j=0}^{s-1} |b_{s(\nu-1)+j}||X|^j =: \gamma_{(s-2)n+2}\widetilde{B}_{\nu-1}(|X|),\end{aligned} \tag{2.14}$$

where $\widetilde{B}_{\nu-1}(X) = \sum_{j=0}^{s-1}|b_{s(\nu-1)+j}|X^j$. Hence a sufficient condition for (2.12) to hold is $\|B_{\nu-1}(X)\| \approx \|\widetilde{B}_{\nu-1}(|X|)\|$, which is true if there is no significant cancellation

in forming $B_{\nu-1}(X)$. This is the case, for example, when the coefficients $b_{s(\nu-1)+j}$, $j = 0\colon s-1$ are one-signed and $X > 0$, or, when the $|b_{s(\nu-1)+j}|$ decay rapidly and there is no significant cancellation in forming the first few terms of $B_{\nu-1}(X)$.

On the other hand, since all the required powers of X are formed in precision u, we have $\|\widehat{Y} - Y\| \le \gamma_{(s-1)n}\|X\|^s$ from Lemma 2.2. Therefore, from (2.9) we have

$$\|\widehat{Y} - Y\| \lesssim snu\|X\|^s = sn\tau_\nu u_\nu \|X\|^s \lesssim u_\nu \|X^s\| \tag{2.15}$$

if $sn\tau_\nu\|X\|^s \lesssim \|X^s\|$. In any case, the validity of this relation will depend on the matrix X and τ_ν. A special instance is when $X \ne 0$ is nilpotent with index s (so $X^s = 0$), where the condition $\|\widehat{Y} - Y\| \le u_\nu\|X^s\|$ cannot possibly be satisfied ($\widehat{Y}$ can contain significant rounding errors), but we are not interested in this case where the evaluation of (2.4) becomes trivial because $p_m(X) = B_0$.

3. Taylor approximants to the matrix exponential. In this section we consider the concrete setting when the matrix polynomial $p_m(X)$ of (1.1) is the truncated Taylor approximant of order m to the matrix exponential of X, where the coefficients $b_i = 1/i!$ decay super-exponentially. We can show that if the 1-norm of X is sufficiently small, then in general the conditions (2.5) are satisfied with $\nu = 1$ and accuracy of the computed polynomial $\widehat{p}_m$ of $p_m(X)$ is guaranteed.

Suppose $\|X\|_1 = \sigma$ for some $0 < \sigma \le s/\mathrm{e}$, where $\mathrm{e} \approx 2.718$ is Euler's constant. We have, for $i = 2\colon r$,

$$\begin{aligned}\tau_i = \frac{\|B_i\|_1\|X^s\|_1}{\|B_{i-1}\|_1} &= \frac{\left\|\frac{1}{(is)!}I + \frac{1}{(is+1)!}X + \cdots + \frac{1}{(is+s-1)!}X^{s-1}\right\|_1 \|X^s\|_1}{\left\|\frac{1}{((i-1)s)!}I + \frac{1}{((i-1)s+1)!}X + \cdots + \frac{1}{((i-1)s+s-1)!}X^{s-1}\right\|_1} \\ &\le \frac{\frac{1}{(is)!}\left(1 + \frac{\sigma}{is+1} + \cdots + \frac{\sigma^{s-1}}{(is+1)^{s-1}}\right)s!}{\frac{1}{((i-1)s)!} - \frac{\sigma}{((i-1)s+1)!}\left(1 + \frac{\sigma}{(i-1)s+2} + \cdots + \frac{\sigma^{s-2}}{((i-1)s+2)^{s-2}}\right)} =: \gamma(s,i).\end{aligned}$$

Since $\sigma \le s/\mathrm{e}$ implies, for $i \ge 2$,

$$r_1 := \frac{\sigma}{(i-1)s+2} < \frac{1}{\mathrm{e}} < 1, \quad r_2 := \frac{\sigma}{is+1} < \frac{1}{2\mathrm{e}} < 1,$$

it follows that, for r_1 and r_2 sufficiently close to zero (which is true when $\sigma \approx 0$ or when s or i is large),

$$\begin{aligned}\gamma(s,i) &= \frac{\frac{s!}{(is)!}\cdot\frac{1-r_2^s}{1-r_2}}{\frac{1}{((i-1)s)!} - \frac{\sigma}{((i-1)s+1)!}\cdot\frac{1-r_1^{s-1}}{1-r_1}} \approx \frac{s!(is-s)!}{(is)!\left(1 - \frac{\sigma}{(i-1)s+1}\right)} \\ &\le \left(1 - \frac{1}{\mathrm{e}(i-1)}\right)^{-1}\binom{is}{s}^{-1} \le \frac{\mathrm{e}}{\mathrm{e}-1}\cdot\frac{s^s}{(is)^s} = \frac{\mathrm{e}}{\mathrm{e}-1}i^{-s}.\end{aligned} \tag{3.1}$$

This shows that, for a chosen s, $\tau_i \le \gamma(s,i)$ (recall that τ_i is from (2.5)) decreases at least polynomially as i increases, and that, for a fixed $i \ge 2$, τ_i decreases at least exponentially as s increases.

We have

$$\|B_1\|_1 \le \frac{1}{s!}\left(1 + \frac{\sigma}{s} + \cdots + \frac{\sigma^{s-1}}{s^{s-1}}\right) = \frac{1-(\sigma/s)^s}{s!(1-\sigma/s)} \le \frac{1}{(s-\sigma)(s-1)!}. \tag{3.2}$$

Both the bound and $\|B_1\|_1$ tend to $1/s!$ as $\sigma \to 0$, which shows this bound tends to be a good approximation to $\|B_1\|_1$ as $\|X\|_1$ tends to zero. We noticed that $\sqrt[s]{s!}$ can be very well approximated by $s/\mathrm{e} + 1 \geq \sigma + 1$ for s in a practical range (we noted that Stirling's approximation gives $\sqrt[s]{s!} \sim s\,\sqrt[2s]{2\pi s}/\mathrm{e}$ for large s), and so from (3.2) we have

$$\begin{aligned} \tau_1 = \frac{\|B_1\|_1\|X^s\|_1}{\|B_0\|_1} &\leq \frac{\mathrm{e}\|X\|_1^s}{(\mathrm{e}-1)s!\|B_0\|_1} \cdot \frac{\|X^s\|_1}{\|X\|_1^s} \\ &\lesssim \frac{(s/\mathrm{e})^s}{s!\|B_0\|_1} \cdot \frac{\|X^s\|_1}{\|X\|_1^s} \leq \frac{1}{\|B_0\|_1} \cdot \frac{\|X^s\|_1}{\|X\|_1^s}, \end{aligned} \tag{3.3}$$

where $\|B_0\|_1 \approx \|\mathrm{e}^X\|_1 \geq \mathrm{e}^{-\|X\|_1}$ [11, Thm. 10.10] and $\|X^s\|_1/\|X\|_1^s$ is bounded above by 1 but can be arbitrarily small. This shows that τ_1 is bounded above by a quantity which tends to zero as s increases.

Now consider the effects of rounding errors on the evaluation of $Y = X^s$ and B_0 in precision $u = u_0$ (as now we consider $\nu = 1$). From (2.15) we need to check whether we have $sn\tau_1\|X\|^s \lesssim \|X^s\|$. We have, using the approximation $B_0(X) \approx \mathrm{e}^X$ from (2.4) and (2.5),

$$\frac{sn\tau_1\|X\|_1^s}{\|X^s\|_1} = \frac{sn\|B_1\|_1\|X\|_1^s}{\|B_0\|_1} \lesssim \frac{sn \cdot s!}{\|\mathrm{e}^X\|_1} \cdot \frac{1-(\sigma/s)^s}{s!(1-\sigma/s)} \lesssim sn\mathrm{e}^{\|X\|_1},$$

which shows, given that $\|X\|_1$ is nicely bounded, $sn\tau_1\|X\|_1^s$ is approximately bounded above by a mild multiple of $\|X^s\|_1$, and therefore, we should expect $Y = X^s$ to be evaluated to satisfying accuracy. On the other hand, since B_0 has all positive coefficients, it follows from (2.14) that

$$\|\widehat{B}_0 - B_0(X)\|_1 \leq \gamma_{(s-2)n+2}B_0(\|X\|_1) \approx \gamma_{(s-2)n+2}\mathrm{e}^{\|X\|_1}.$$

We deduce, using [11, Thm. 10.10], that

$$\begin{aligned} \|\widehat{B}_0 - B_0(X)\|_1 &\lesssim \gamma_{(s-2)n+2}\mathrm{e}^{\|X\|_1}\mathrm{e}^{\|X\|_1}\|\mathrm{e}^X\|_1 \\ &\approx \gamma_{(s-2)n+2}\|B_0(X)\|_1\mathrm{e}^{2\|X\|_1} \leq \gamma_{(s-2)n+2}\|B_0(X)\|_1\mathrm{e}^{2s/\mathrm{e}}. \end{aligned}$$

Hence the relative error in $\widehat{B}_0$ is bounded approximately by $\gamma_{(s-2)n+2}\mathrm{e}^{2s/\mathrm{e}}$, which is a satisfactory bound for practical values of s. We have empirically found that $\widehat{B}_0$ is typically computed to close to full working precision (the relative error in $\widehat{B}_0$ is typically close to u) for matrices of varying size generated pseudo-randomly and from the MATLAB gallery. This is consistent with the analysis which shows the rounding errors in the evaluation of B_0 are nicely bounded, and it is also a possible consequence of the fact that the underlying BLAS in MATLAB uses blocked algorithms to reduce the error constant [12].

3.1. Applicability in the scaling and squaring algorithms for the matrix exponential. The discussion in the previous subsection implies that we could build a mixed-precision PS algorithm for the mth-order Taylor approximant to the matrix exponential under the constraint $\|X\|_1 \leq s/\mathrm{e}$, where the parameter s could exceed $\lceil\sqrt{m}\rceil$ in order for the desired decaying property of the polynomial coefficients (2.5) to hold, and accuracy of the algorithm could in general be guaranteed. However, in the stat-of-the-art algorithms for the matrix exponential [1], [8], which employ the scaling and squaring idea

$$\mathrm{e}^A = \left(\mathrm{e}^X\right)^{2^\ell} \approx p_m(X)^{2^\ell}, \quad \ell \in \mathbb{N}, \tag{3.4}$$

the admittable (scaled) matrix $X := 2^{-\ell}A$ can have a 1-norm that does not satisfy the constraint $\|X\|_1 \le s/\mathrm{e}$. This is because the thresholds for accepting certain X in these algorithms are determined by forward or backward truncation error bounds of the approximant to the exponential on the scaled matrix in exact arithmetic, and these thresholds in fact disregard the rounding errors in the computation of the approximant $p_m(X)$. For example, the Taylor-based algorithm of [8] requires that X satisfy

$$\|\mathrm{e}^X - p_m(X)\|_1 \le |\mathrm{e}^{\alpha_m(X)} - p_m(\alpha_m(X))| \le u\xi(X), \tag{3.5}$$

where $\xi(X)$ is some practical estimate to $\|\mathrm{e}^X\|_1$ and

$$\alpha_m(X) = \max\bigl(\|X^{d^*}\|_1^{1/d^*}, \|X^{d^*+1}\|_1^{1/(d^*+1)}\bigr),$$
$$d^* = \max_d\{d \in \mathbb{N}^+ \colon d(d-1) \le m+1\} = \left\lfloor \frac{1+\sqrt{4m+5}}{2} \right\rfloor.$$

In principle, the constraint $\|X\|_1 \le s/\mathrm{e}$ does not prevent the potential algorithm from being embedded into any existing scaling and squaring algorithm based on the Taylor approximants, for example, those employ the α_m-based bound (3.5). This is because, for a scaled matrix X accepted by one of such algorithms with $\|X\|_1 > s/\mathrm{e}$, one can always further scale X to $Z = 2^{-\ell_0}X$, $\ell_0 \in \mathbb{N}^+$ such that $\|Z\|_1 \le s/\mathrm{e}$ and Z remain admittable by the algorithm since $\alpha_m(Z) = 2^{-\ell_0}\alpha_m(X) < \alpha_m(X)$. In this way, instead of invoking (3.4), the algorithm will use the approximation

$$\mathrm{e}^A = \left(\mathrm{e}^{2^{-(\ell+\ell_0)}A}\right)^{2^{\ell+\ell_0}} \approx p_m(Z)^{2^{\ell+\ell_0}},$$

which, from our discussion in the previous subsection, has more refined bound on the rounding errors in the computed approximant $\widehat{p}_m$. But the algorithm can nevertheless require substantially more squaring steps in the final squaring phase, which is very sensitive to rounding errors [11, p. 247], because $\alpha_m(X)$ can be much smaller than $\|X\|_1$ for nonnormal X [1] and a matrix X accepted by an α_m-based bound can have huge 1-norm. For example, consider the matrix

$$A = \begin{bmatrix} -0.1 & 10^6 \\ 0 & -0.1 \end{bmatrix}, \tag{3.6}$$

for which the 1-norms of the powers of A decay exponentially and the Taylor-based algorithm of [8] with $u = 10^{-64}$ chooses $m = 42$ and accepts $X = A/2$ ($\ell = 1$), despite the large $(1,2)$ element. In this case $\alpha_m(X) \approx 0.66$ but $\|X\|_1 = 5 \times 10^5$ and with $s = \lceil\sqrt{m}\rceil$ the number of extra squarings required is $\ell_0 \ge \lceil\log_2(\mathrm{e}\|X\|_1/s)\rceil = 18$.

In fact, the scaling and squaring algorithms for the matrix exponential [1], [5], [8], [21], which disregard the occurrence of rounding errors in the computed approximant $\widehat{p}_m$ when determining the thresholds for accepting the scaled matrix, have been observed to work well in practice, even for X with a large 1-norm but a small or moderate $\alpha_m(X)$ associated with the chosen m. Disregarding the constraint $\|X\| \le s/\mathrm{e}$, which can cause overscaling issue for the scaling and squaring algorithms, does not appear to be harmful for the accuracy of a practical algorithm in the occurrence of rounding errors. Then in this case the question is to what extend (2.5) can be satisfied. We have found the condition often holds with $\nu = 1$ if $\|X^s\|$ is small, which is consistent with the discussion following (2.5). In the scaling and squaring algorithms that employ the α_m-based bound (3.5), for example, [8], the $\alpha_m(X)$ can be very small

Fragment 3.1: Computing B_0 and $Y = X^s$ in precision u.

function EVAL$B_0(\mathcal{X},\ s \in \mathbb{N}^+)$
▷ *Form the first* $s-1$ *positive powers of* X *in* $\mathcal{X}$ *and then compute* $B_0 = \sum_{j=0}^{s-1} X^j/j!$ *and* $Y = X^s$ *using elements of* $\mathcal{X}$.
$B_0 \leftarrow I$
for $j \leftarrow 1$ **to** $s-1$ **do**
  $\mathcal{X}_j \leftarrow \mathcal{X}_{j-1}X$
  $B_0 \leftarrow B_0 + \mathcal{X}_j/j!$
$Y \leftarrow \mathcal{X}_{s-1}X$
return $\mathcal{X}$, B_0, Y

(associated with the chosen m) on some tested matrices even if $\|X\|$ is large, in which case the value of $\|X^{d^*}\|_1$ is necessarily small, where $d^* \approx \sqrt{m}$ just matches the default parameter $s = \lfloor\sqrt{m}\rfloor$ or $s = \lceil\sqrt{m}\rceil$ in the fix-precision PS scheme, so (2.5) often holds with $\nu = 1$. In the least preferred case where (2.5) is not met for any $\nu \in [1, r]$, still, we can simply compute $q(X)$ from (2.6) in the working precision, in which case the algorithm recovers the fix-precision PS scheme.

Summarising the discussion in the section, we now present the mixed-precision PS algorithm in Algorithm 3.2, which is readily employable by the Taylor-based scaling and squaring algorithms for the matrix exponential. The algorithm takes the matrix $X \in \mathbb{C}^{n\times n}$, the order m of the used Taylor approximant, and the working precision u as input arguments. It starts with computing B_0 straightly in precision u with the default parameter $s = \lceil\sqrt{m}\rceil$ and then proceeds differently depending on $\|X\|_1$.

If $\|X\|_1 \le s/\mathrm{e}$, the algorithm increments s and updates B_0 and Y until the bound $\tau_1 \le \|X^s\|_1/\left(\|B_0\|_1\|X\|_1^s\right) \le 1$ from (3.3) is satisfied. Here we only check (2.5) for $i = 1$ since we have shown that the τ_i, $i = 2\colon r$ tend to decay at least polynomially as i increases (see (3.1)) and we found if the first condition of (2.5) is satisfied, then the remaining conditions therein are met for $\|X\|_1 \le s/\mathrm{e}$. After B_i and u_i are computed for all i the algorithm then executes the Horner's method (2.4) for $p_m(X)$ with the matrix products and sums done in the appropriate precisions. The economics of our mixed-precision PS scheme are different from its fixed-precision counterpart: the matrix multiplications performed in computing B_0 are potentially the most expensive while the others are potentially done in lower precisions, therefore a smaller s with a larger $r = \lfloor m/s\rfloor$ is potentially better for efficiency, if the cost of the matrix multiplications in lower precisions are negligible compared with those done in the working precision. Overall, Algorithm 3.2 requires $\lceil\sqrt{m}\rceil - 1 \le s - 1 \le m - 1$ matrix multiplications in precision u and one matrix multiplication in each of $u_i > u$, $i = 1\colon r$, where $1 \le r = \lfloor m/s\rfloor \le \lceil\sqrt{m}\rceil$ (when $s = m$ the PS scheme actually degenerates to evaluation via explicit powers and hence no matrix multiplications are formed in u_i). In the case of optimal efficiency, Algorithm 3.2 requires $\lceil\sqrt{m}\rceil - 1$ matrix multiplications in u, which is only approximately half of the matrix multiplications required by the fix-precision PS scheme.

For $\|X\|_1 > s/\mathrm{e}$ the algorithm sticks with $s = \lceil\sqrt{m}\rceil$ to form the B_i and then sets the u_i by $\max(u, \text{right-hand side of (2.8)})$, $i = 1\colon r$ (where a good estimation of $\|B_i\|_1$ can be obtained by using only the first few terms of B_i since $|b_i|$ decays quickly) until $u_i \ge \delta u$ (say, $i = \nu$), so lower precisions $u_r \ge u_{r-1} \ge \cdots \ge u_\nu$ are exploited in the computation of $q(X)$ and the remaining part of $p_m(X)$ is then computed via (2.11)

Algorithm 3.2: Mixed-precision Paterson–Stockmeyer scheme for the Taylor approximants of the matrix exponential.

Given $X \in \mathbb{C}^{n\times n}$ this algorithm computes an mth order Taylor approximant $P \equiv p_m(X)$ in the form of (1.1) using the Paterson–Stockmeyer scheme in floating-point arithmetic. The algorithm starts with the user-specified precision u and potentially uses multiple lower precisions $u_i \geq u$ aiming to produce a relative error of order nu. The pseudocode of EvalB$_0$ is given in Fragment 3.1.

$s \leftarrow \lceil \sqrt{m} \rceil$
$u_0 \leftarrow u$
$\mathcal{X}_0 \leftarrow I$
$[\mathcal{X}, B_0, Y] \leftarrow \textsc{EvalB}_0(\mathcal{X}, s)$
if $\|X\|_1 \leq s/\mathrm{e}$ **then**
  while $\|Y\|_1 > \|B_0\|_1 \|X\|_1^s$ **and** $s < m$ **do**
    $B_0 \leftarrow B_0 + Y/s!$
    $s \leftarrow s + 1$
    $\mathcal{X}_s \leftarrow XY$ in precision u_0
    $Y \leftarrow \mathcal{X}_s$
  $r \leftarrow \lfloor m/s \rfloor$
  for $i \leftarrow 1$ **to** r **do**
    Form B_i using elements in $\mathcal{X}$ and then estimate $\|B_i\|_1$
    $u_i \leftarrow \|B_0\|_1 u_0 / (\|B_i\|_1 \|Y\|_1^i)$
else
  $r \leftarrow \lfloor m/s \rfloor$
  $\nu \leftarrow r + 1$
  for $i \leftarrow 1$ **to** r **do**
    Form B_i using elements in $\mathcal{X}$ and then estimate $\|B_i\|_1$
    $u_i \leftarrow \max\left(u, \|B_0\|_1 u_0 / (\|B_i\|_1 \|Y\|_1^i)\right)$
    if $u_i \geq \delta u$ **then**
      $\nu = i$, **break**
  for $i \leftarrow \nu + 1$ **to** r **do**
    Form B_i using elements in $\mathcal{X}$ and then estimate $\|B_i\|_1$
    $u_i \leftarrow \|B_{\nu-1}\|_1 u_0 / (\|B_i\|_1 \|Y\|_1^{i-\nu+1})$
$P = B_r$
for $i \leftarrow r$ **down to** 1 **do**
  Compute $P \leftarrow PY$ in precision u_i
  Form $P \leftarrow P + B_{i-1}$ in precision u_{i-1}
return P

in the working precision u. We set by default $\delta = 10$ (so the algorithm will switch to a lower precision when appropriate, even if the number of significant digits decreases by just 1). In this case, Algorithm 3.2 requires $\lceil \sqrt{m} \rceil + \nu - 2$ matrix multiplications in precision u and one matrix multiplication in each of $u_\nu, u_{\nu+1}, \ldots, u_r$, where $1 \leq \nu \leq r$. Obviously, a smaller ν means more matrix products are performed in precisions lower than u and when $u_i \geq \delta u$ does not hold for any i, the algorithm recovers the fix-

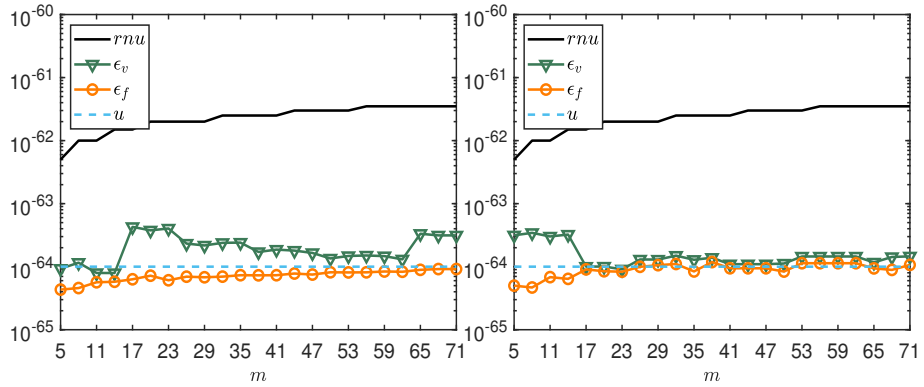

FIG. 3.1. *The relative errors* $\epsilon_v = \|\widehat{p}_m - p_m(X)\|_1 / \|p_m(X)\|_1$ *produced by Algorithm 3.2 against different values of* m *on normalised matrices with* $n = 50$ *and* $\|X\|_1 = \lceil\sqrt{m}\rceil/e$*, in comparison with the relative errors* ϵ_f *produced by the fixed-precision Paterson-Stockmeyer with* $s = \lceil\sqrt{m}\rceil$ *in precision* $u = 10^{-64}$*. Left:* $X =$ `rand(n)`*. Right:* $X =$ `randn(n)`*.*

precision PS scheme. Since the algorithm now does not have restriction on $\|X\|$ and our analysis only accounts for the potential use of lower precisions, the caveat is that there is no guarantee of the accuracy in the part of the PS scheme done in the working precision.

We remark that the framework of Algorithm 3.2 can be adjusted with little modification for the computation of other matrix polynomials with scalar coefficients that quickly decay in modulus, such as the polynomials in the numerator and denominator of the Padé approximants of exponential-like functions. We will further comment on the generality of the framework in section 4.

3.2. Numerical experiments. All our experiments are performed using the 64-bit version of MATLAB 2023b on a desktop equipped with an Intel i7-6700 processor running at 3.40GHz and with 64GB of RAM. The code uses the `chop`[1] function [15] to simulate the bfloat16 half precision and the Advanpix Multiprecision Computing Toolbox (Version 5.1.1.15444) [18] for simulating precisions other than half, single, and double precisions.

3.2.1. Behaviour in variable-precision arithmetic. In the setting of variable-precision arithmetic, the algorithms start with u and can potentially use internally multiple lower precisions $u_i \geq u$, $i = 1\colon r$ that can be arbitrarily chosen.

We first test Algorithm 3.2 on pseudorandom matrices normalised such that $\|X\|_1 = \lceil\sqrt{m}\rceil/\mathrm{e}$ against different values of m in the working precision $u = 10^{-64}$. In general, for the chosen degree m the norm of the highest order term $X^m/m!$ should not be smaller than the unit roundoff u. In the experiments we therefore choose the value for the largest tested m from the condition that m is the smallest integer such that

$$\frac{\lceil\sqrt{m}\rceil^m}{m!\mathrm{e}^m} \leq u. \tag{3.7}$$

Figure 3.1 reports the results, where the reference solution is computed by using the Multiprecision Computing Toolbox with 400 decimal digits of precision. We observe

[1] https://github.com/higham/chop

TABLE 3.1

The minimal degree m such that the difference between $p_m(X)$ and e^X in the 1-norm is no larger than u. The d_i represents the equivalent decimal digits of precision u_i and C_p is approximately the complexity reduction in percentage of Algorithm 3.2 compared with its fixed-precision counterpart in precision u. The d_i such that $d_i \le d_0/2$ (d_0 is the decimal digits of the working precision u) is highlighted in red text.

(u, m)	(s, r)	$(d_1, d_2, \ldots, d_r)$	C_p
$(10^{-32}, 37)$	$(7, 5)$	$(30, 25, 18, 11, 3)$	20.7%
$(10^{-64}, 60)$	$(8, 7)$	$(61, 55, 47, 38, 28, 18, 7)$	21.6%
$(10^{-128}, 99)$	$(10, 9)$	$(124, 115, 104, 92, 78, 64, 49, 34, 18)$	20.6%
$(10^{-256}, 169)$	$(13, 13)$	$(249, 237, 221, 203, 184, 164, 143, 121, 99, 75, 52, 28, 3)$	24.2%

in all tested cases the algorithm is producing a relative error of order u, which is comparable to that of its fixed-precision counterpart and is a couple of magnitudes smaller than the error bound rnu derived in section 2. We repeated the experiments in a working precision of 256 digits, finding similar behavior of the algorithm.

We next fix the test matrix to be $X =$ `gallery('cauchy',n)` with $n = 100$ and compute the Taylor approximant from its matrix exponential with several choices of (u, m). Now $\|X\|_1 \approx 4.20$ and we report in Table 3.1 some important algorithmic characteristics on the matrix. We see from the table that the default $s = \lceil \sqrt{m} \rceil$ is chosen in all tested cases and that $\tau_i = u_{i-1}/u_i$ is in general decreasing, which is consistent with our analysis (see (3.1)). Recall that the algorithm requires $s - 1 + r$ matrix multiplications and we see that approximately a fifth of the matrix multiplications were performed in precision $u^{1/2}$ or much lower in all tested cases. If the algorithmic complexity (measured in number of matrix multiplications) is assumed to be linearly proportional to the number of digits used, then we can calculate the approximate percentage of complexity reduction of Algorithm 3.2 compared with its fixed-precision counterpart in precision u as

$$C_p = \frac{r \log_{10} u + \sum_{i=1}^{r} d_i}{(s + r - 1) \log_{10} u}. \tag{3.8}$$

Note that the assumption on the algorithmic complexity is realistic yet slightly pessimistic because scalar multiplications and divisions scale even faster with the number of digits. As shown, the new algorithm is able to reduce the computational cost of computing the exponential of these test matrices by a percentage between 20.6% and 24.2%.

Finally, we test Algorithm 3.2 on various matrices from the literature of the matrix exponential [1], [8] and from the `gallery` group of Anymatrix [13], [14]. For these matrices, the condition $\|X\|_1 \le s/e$ is often not satisfied and the degree m of the approximant is given data. The 97 non-Hermitian test matrices are of size ranging from 2 to 100 and are possibly scaled with the scaling factor chosen by the Taylor-based algorithm of [8], which is the state-of-the-art aiming for arbitrary precision environment. The Hermitian matrices are excluded because they are handled by diagonalization instead of scaling and squaring by the algorithm. Again, we see from Figure 3.2 (*a*)–(*b*) that the relative errors produced by the algorithm are close to that of its fixed-precision counterpart in all cases and in most cases no larger than rnu. On the other hand, we see that in most cases the computational cost of Algorithm 3.2 is at least 25% lower compared with its fixed-precision counterpart on the real test set, and in few cases the savings can reach 40%.

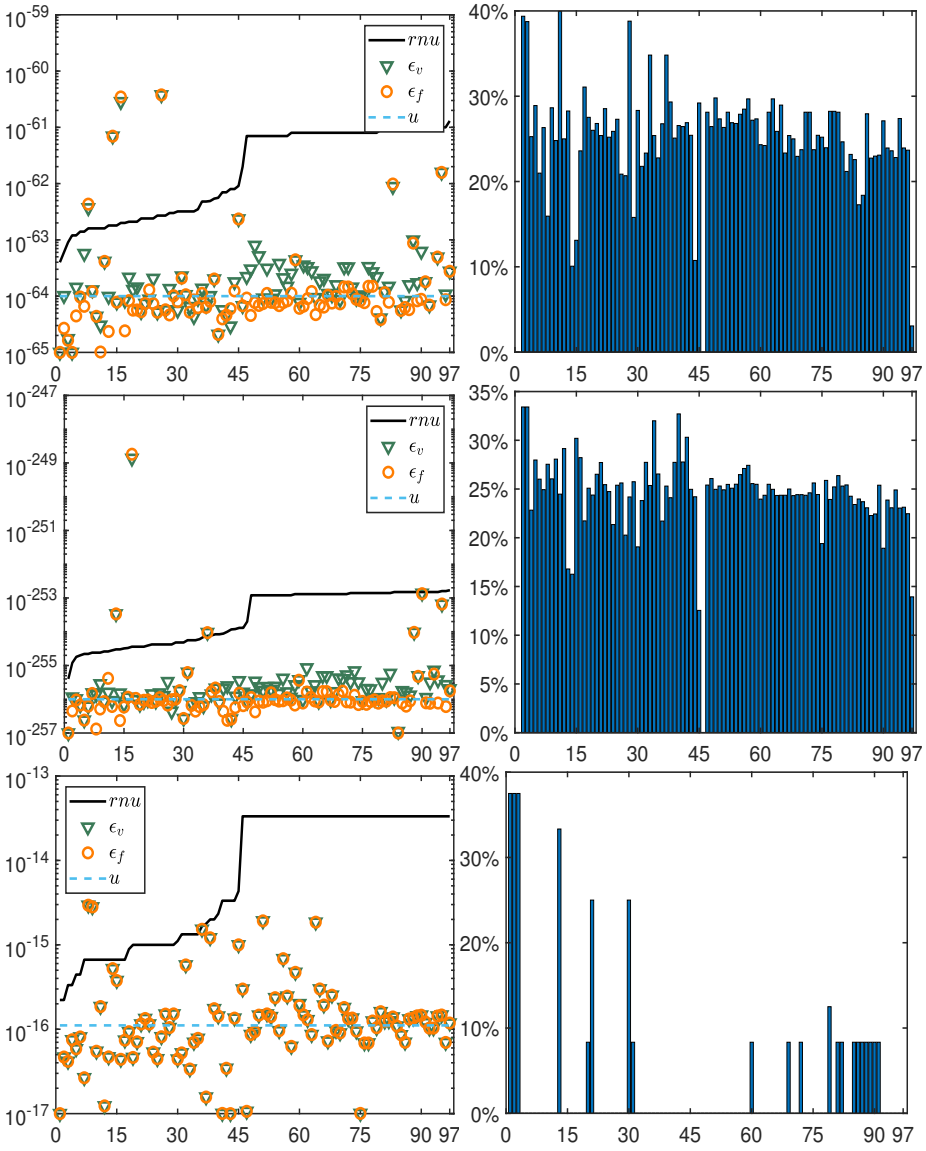

FIG. 3.2. *Left: The relative error* $\epsilon_v = \|\widehat{p}_m - p_m(X)\|_1/\|p_m(X)\|_1$ *produced by Algorithm 3.2 compared with the relative errors* ϵ_f *produced by the fixed-precision Paterson–Stockmeyer with* $s = \lceil\sqrt{m}\rceil$ *on various matrices with* $2 \le n \le 100$. *Right: The associated approximate percentages of complexity reduction* C_p *in (3.8) of the algorithm compared with its fixed-precision counterpart with* $s = \lceil\sqrt{m}\rceil$. *Top (a):* $u = 10^{-64}$. *Middle (b):* $u = 10^{-256}$. *Bottom (c):* $u = 2^{-53}$.

3.2.2. Low-precision variant of the algorithm. We also implemented Algorithm 3.2 using potentially only three precisions: fp64 (IEEE double), fp32 (IEEE single), and bfloat16 (by Google Brain[2]), aiming to achieve accuracy of level of the unit roundoff of double precision. This means the u_i chosen by (2.8) in the algorithm will be set to the nearest higher precision among $u_f = 2^{-8} \approx 3.9 \times 10^{-3}$,

[2] https://www.wikiwand.com/en/Google_Brain

$u_s = 2^{-24} \approx 6.0 \times 10^{-8}$, and $u = 2^{-53} \approx 1.1 \times 10^{-16}$. We are using bfloat16 instead of fp16 because the latter has the drawback of having a limited range, the smallest representable positive number being $x_{\min} \approx 5.96 \times 10^{-8}$, while the former has much larger exponential range with $x_{\min} \approx 1.18 \times 10^{-38}$ and this makes loss of accuracy due to underflow less likely to happen. This is particularly important for Algorithm 3.2, whose spirit is computing smaller numbers in a lower precision.

We test the low-precision variant of Algorithm 3.2 on the same set of 97 non-Hermitian matrices used in the previous subsection. The results are reported in Figure 3.2 (c). Similarly to the variable-precision variant, the algorithm is in all cases producing relative errors that are close to the double-precision counterpart and are no larger than order nu. But from the plot in the right panel, we note that the advantage in efficiency of the mixed-precision algorithm has greatly reduced, and in most cases the algorithm just degenerates to solely use double precision. The difference is largely because the chosen degree m is generally much smaller in a lower working precision, and this restricts more matrix products being formed in lower precisions and hence limits the efficiency gain of the mixed-precision algorithm. Also, in the double-single-half precision environment the algorithm loses the complete freedom to choose arbitrary precision and it in many cases has to use an unnecessarily higher precision.

4. A mixed-precision Paterson–Stockmeyer algorithm for general polynomials of matrices. Following the discussion in section 3.1, we now exploit the mixed-precision PS framework for the computation of general matrix polynomials with scalar coefficients decaying in modulus. The algorithm is presented as Algorithm 3.3.

We will test Algorithm 3.3 in the next subsections on different types of polynomials of matrices arising from the computation of exponential-like matrix functions.

4.1. Padé approximants to the matrix exponential. Again, we take the 97 non-Hermitian matrices from the same matrix set tested in the previous section with the degree m and the scaling parameter chosen by the Padé-based algorithm of [8] and use Algorithm 3.3 to compute the matrix polynomials p_{km} and q_{km}^{-1} from the $[k/m]$ Padé approximants $r_{km} = q_{km}^{-1} p_{km}$, where p_{km} and q_{km} are of degrees at most k and m, respectively.

Figure 4.1 (a)–(b) reports the result. We observe that the relative errors produced by Algorithm 3.3 have the same order of magnitude as those produced by the fixed-precision counterpart. Also, the reduction in computational complexity of the algorithm is on average around 10%, which is significantly smaller than the 25% reduction obtained when applying the algorithm to the Taylor Approximants of the matrix exponential (c.f. Figure 3.2 (a)). Again, the smaller (approximately halved) chosen degrees m in the matrix Padé approximants than the matrix Taylor approximant have prevented the algorithm from carrying out more low-precision matrix multiplications, outweighing the fact that the algorithm can potentially employ lower precisions early (since the scalar coefficients of the matrix polynomials from the Padé approximant decay much faster).

4.2. Taylor approximants to the matrix cosine. Next we test the algorithm on the Taylor approximants of the matrix cosine, denoted by c_m. The 98 test matrices are nonnormal and are from [2] (with the 2×2 nilpotent matrix therein excluded) and of size between 4 and 100. Most of the test matrices have only real elements and are set to be of size 100×100. The degree m and the scaling parameter for these matrices are determined by the Taylor-based scaling and recovering algorithm of [2].

Algorithm 3.3: Mixed-precision Paterson–Stockmeyer scheme for matrix polynomials with scalar coefficients

Given $X \in \mathbb{C}^{n\times n}$ and a set of polynomial coefficients $\{b_i\}_{i=0}^m$, this algorithm computes the matrix polynomial with scalar coefficients decaying in modulus $P \equiv p_m(X)$ in the form of (1.1) using the Paterson–Stockmeyer scheme in floating-point arithmetic. The algorithm starts with the user-specified precision u and potentially uses multiple lower precisions $u_i \geq u$ aiming to produce a relative error of order nu.

$s \leftarrow \lceil \sqrt{m} \rceil$
$r \leftarrow \lfloor m/s \rfloor$
$\nu \leftarrow r + 1$
$u_0 \leftarrow u$
$\mathcal{X}_0 \leftarrow I$
$B_0 \leftarrow b_0\mathcal{X}_0$ in precision u_0
for $j \leftarrow 1$ **to** $s-1$ **do**
  $\mathcal{X}_j \leftarrow \mathcal{X}_{j-1}X$ in precision u_0
  $B_0 \leftarrow B_0 + b_j\mathcal{X}_j$ in precision u_0
$Y \leftarrow \mathcal{X}_{s-1}X$ in precision u_0
for $i \leftarrow 1$ **to** r **do**
  Form B_i using elements in $\mathcal{X}$ and then estimate $\|B_i\|_1$
  $u_i \leftarrow \max\left(u, \|B_0\|_1 u_0/(\|B_i\|_1\|Y\|_1^i)\right)$
  if $u_i \geq \eta u$ **then**
    $\nu = i$, **break**
for $i \leftarrow \nu + 1$ **to** r **do**
  Form B_i using elements in $\mathcal{X}$ and then estimate $\|B_i\|_1$
  $u_i \leftarrow \|B_{\nu-1}\|_1 u_0/(\|B_i\|_1\|Y\|_1^{i-\nu+1})$
$P = B_r$
for $i \leftarrow r$ **down to** 1 **do**
  Compute $P \leftarrow PY$ in precision u_i
  Form $P \leftarrow P + B_{i-1}$ in precision u_{i-1}
return P

Figure 4.1 (c) shows that the accuracy of Algorithm 3.3 in this case is similar to that of the fixed-precision PS algorithm so is satisfying. The modulus of the coefficients in the Taylor approximants of the matrix cosine also decay considerably faster than that of the matrix exponential, for the same reasons discussed in section 4.1, the reduction in computational cost of the algorithm is marginal, at typically less than 5%. Moreover, in the algorithm we are actually evaluating the polynomial $c_m(X^2)$ via the PS scheme since the polynomial only contains even powers of X, and this makes the chosen degree m even smaller.

5. Conclusions. In this work we have developed a mixed-precision Paterson–Stockmeyer (PS) method for evaluating matrix polynomials with scalar coefficients that decay in modulus. The key idea is to perform computations on data of small magnitude in low precision, and driven by this idea we show with rigorous error analysis that, if the coefficients satisfy a certain decaying property, then we can exploit

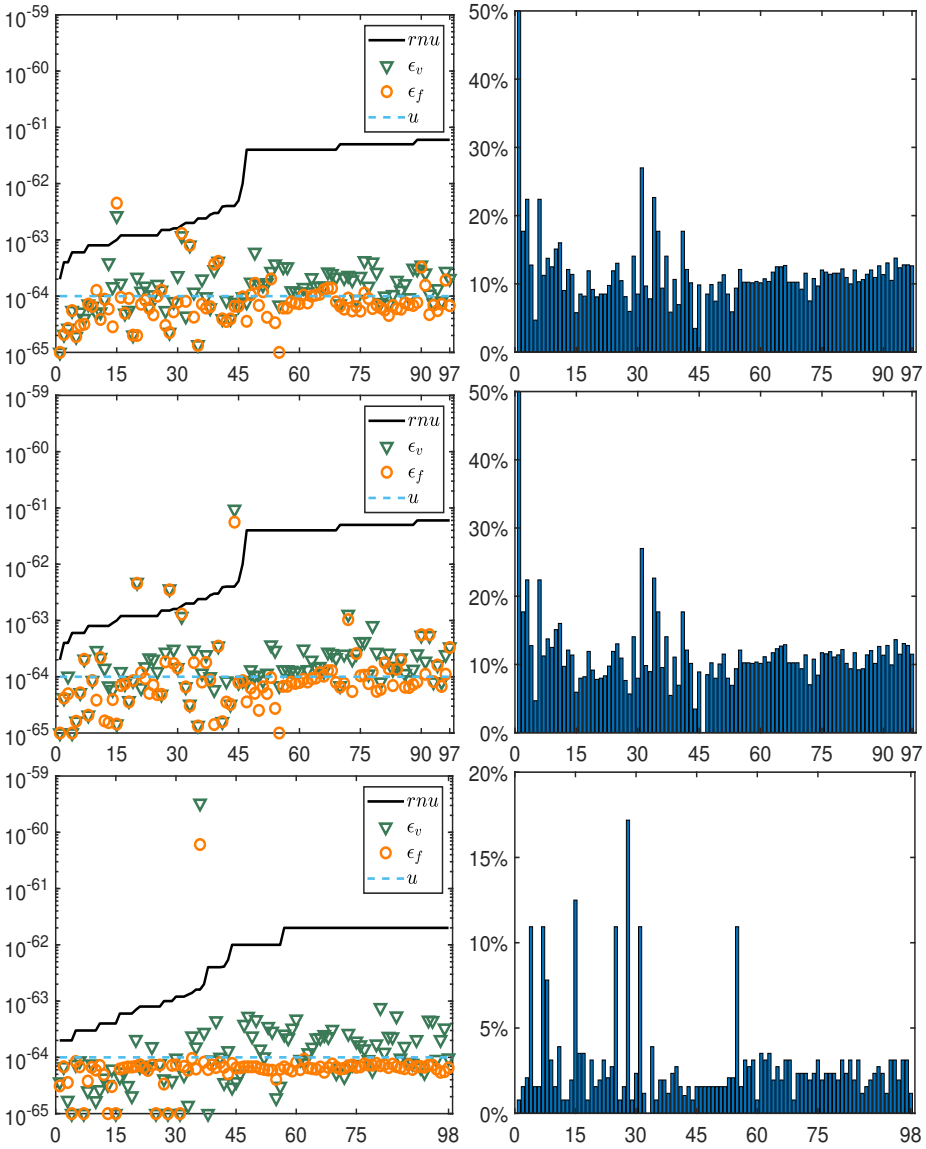

FIG. 4.1. *Left: The relative error* $\epsilon_v = \|\widehat{p}_m - p_m(X)\|_1 / \|p_m(X)\|_1$ *produced by Algorithm 3.3 in precision* $u = 10^{-64}$ *compared with the relative errors* ϵ_f *produced by the fixed-precision Paterson–Stockmeyer with* $s = \lceil \sqrt{m} \rceil$ *on various matrices with* $2 \le n \le 100$. *Right: The associated approximate percentages of algorithmic complexity reduction* C_p *in (3.8) of the algorithm compared with its fixed-precision counterpart with* $s = \lceil \sqrt{m} \rceil$. *Top (a):* $p_m = p_{mm}$ *from the* $[m/m]$ *Padé approximant of the matrix exponential. Middle (b):* $p_m = q_{mm}$ *from the* $[m/m]$ *Padé approximant of the matrix exponential. Bottom (c):* $p_m = c_m$ *from the Taylor approximant of the matrix cosine.*

a set of suitably chosen lower precisions (relative to the working precision) in the evaluation of the polynomial via a Horner scheme and still achieve the same level of accuracy as the evaluation being done solely in the working precision.

We applied the method to the computation of the Taylor approximants of the matrix exponential and showed the applicability of the mixed-precision PS framework to the existing scaling and squaring algorithms for the matrix exponential, particularly

when the norm of the input matrix is sufficiently small. The algorithm (Algorithm 3.2) puts no restriction on the input matrix and switches to lower precisions when appropriate, and it is readily employable by the Taylor-based scaling and squaring algorithms for the matrix exponential. Taking advantage of the generality of the mixed-precision PS framework, we finally designed an algorithm (Algorithm 3.3) for the computation of general polynomials of matrices and demonstrated its efficiency on polynomials from the Padé approximant of the matrix exponential and the Taylor approximant of the matrix cosine.

Numerical experiments show comparable accuracy of our mixed-precision PS algorithms to its fixed-precision counterparts (in the working precision). By measuring the computational complexity in terms of matrix multiplications and assuming the complexity is linearly proportional to the number of used digits, we find that the new algorithm is more efficient on various synthetic and real test problems in both variable precision arithmetic and the arithmetic that only involves IEEE double, IEEE single, and Google bfloat16 half precisions, and the reduction in complexity can be up to around 40% amongst our tested cases.

In principle, the new mixed-precision PS algorithm can be recommended for evaluating matrix polynomials with scalar coefficients that decay fast in modulus. Our MATLAB code is available from https://github.com/Xiaobo-Liu/mp-ps.

Appendix A. Proof of Theorem 2.1. The proof is by induction. Defining

$$E_j := \widehat{\varphi}_j - \varphi_j, \quad j = \nu - 1 : r,$$

we have $\|E_r\| = \|\widehat{\varphi}_r - \varphi_r\| = \|\widehat{B}_r - B_r\| \le u_r\|B_r\|$. Consider

$$\begin{aligned} E_{r-1} &= \widehat{\varphi}_{r-1} - \varphi_{r-1} = \mathrm{fl}_{r-1}\left(\widehat{B}_{r-1} + \mathrm{fl}_r(\widehat{Y}\widehat{\varphi}_r)\right) - B_{r-1} - Y\varphi_r \\ &:= E_{r-1,s} + E_{r-1,p} + E_{r-1,a}, \end{aligned}$$

where

$$\|E_{r-1,s}\| \le \gamma_1^{r-1}\|B_{r-1}\|, \quad \|E_{r-1,p}\| \le \gamma_n^r\|Y\|\|B_r\|,$$

and

$$\|E_{r-1,a}\| \le \gamma_1^{r-1}\|B_{r-1} + E_{r-1,s} + YB_r + E_{r-1,p}\|.$$

So by using (2.3) we have

$$\begin{aligned} \|E_{r-1}\| &\le \left(\gamma_n^r + \gamma_1^{r-1} + \gamma_n^r\gamma_1^{r-1}\right)\|Y\|\|B_r\| + \gamma_2^{r-1}\|B_{r-1}\| \\ &\le \gamma_{n\theta_{r,r-1}+1}^{r-1}\|Y\|\|B_r\| + \gamma_2^{r-1}\|B_{r-1}\|. \end{aligned}$$

Then we have

$$\begin{aligned} E_{r-2} &= \widehat{\varphi}_{r-2} - \varphi_{r-2} = \mathrm{fl}_{r-2}\left(\widehat{B}_{r-2} + \mathrm{fl}_{r-1}\left(\widehat{Y}(\varphi_{r-1} + E_{r-1})\right)\right) - B_{r-2} - Y\varphi_{r-1} \\ &= YE_{r-1} + E_{r-2,s} + E_{r-2,p} + E_{r-2,a}, \end{aligned}$$

where

$$\|E_{r-2,s}\| \le \gamma_1^{r-2}\|B_{r-2}\|, \quad \|E_{r-2,p}\| \le \gamma_n^{r-1}\|Y\|\|\varphi_{r-1} + E_{r-1}\|,$$

and

$$\|E_{r-2,a}\| \le \gamma_1^{r-2}\|B_{r-2} + E_{r-2,s} + Y\left(\varphi_{r-1} + E_{r-1}\right) + E_{r-2,p}\|,$$

so we have

$$\begin{aligned}\|E_{r-2}\| &\le \gamma_2^{r-2}\|B_{r-2}\| + \left(1 + \gamma_{n\theta_{r-1}+1}^{r-2}\right)\|Y\|\|E_{r-1}\| + \gamma_{n\theta_{r-1}+1}^{r-2}\|Y\|\|\varphi_{r-1}\| \\ &\le \gamma_{f_{r-2,r}}^{r-2}\|Y\|^2\|B_r\| + \gamma_{f_{r-2,r-1}}^{r-2}\|Y\|\|B_{r-1}\| + \gamma_{f_{r-2,r-2}}^{r-2}\|B_{r-2}\|,\end{aligned}$$

where

$$\begin{aligned}f_{r-2,r} &= n\theta_{r,r-2} + (n+1)\theta_{r-1,r-2} + 1 \\ f_{r-2,r-1} &= (n+2)\theta_{r-1,r-2} + 1 \\ f_{r-2,r-2} &= 2.\end{aligned}$$

Now assume a bound for $\|E_{r-k}\|$ of the following form:

$$\|E_{r-k}\| \le \gamma_{f_{r-k,r}}^{r-k}\|Y\|^k\|B_r\| + \gamma_{f_{r-k,r-1}}^{r-k}\|Y\|^{k-1}\|B_{r-1}\| + \cdots + \gamma_{f_{r-k,r-k}}^{r-k}\|B_{r-k}\|, \tag{A.1}$$

where

$$\begin{aligned}f_{r-k,r} &= n\theta_{r,r-k} + (n+1)\left(\theta_{r-1,r-k} + \theta_{r-2,r-k} + \cdots + \theta_{r-k+1,r-k}\right) + 1, \\ f_{r-k,r-1} &= (n+2)\theta_{r-1,r-k} + (n+1)\left(\theta_{r-2,r-k} + \theta_{r-3,r-k} + \cdots + \theta_{r-k+1,r-k}\right) + 1, \\ &\vdots \\ f_{r-k,r-k+1} &= (n+2)\theta_{r-k+1,r-k} + 1, \\ f_{r-k,r-k} &= 2.\end{aligned}$$

Then we have

$$\begin{aligned}E_{r-(k+1)} &= \widehat{\varphi}_{r-(k+1)} - \varphi_{r-(k+1)} \\ &= \mathrm{fl}_{r-(k+1)}\left(\widehat{B}_{r-(k+1)} + \mathrm{fl}_{r-k}\left(\widehat{Y}(\varphi_{r-k} + E_{r-k})\right)\right) - B_{r-(k+1)} - Y\varphi_{r-k} \\ &= YE_{r-k} + E_{r-(k+1),s} + E_{r-(k+1),p} + E_{r-(k+1),a},\end{aligned}$$

where

$$\left\|E_{r-(k+1),s}\right\| \le \gamma_1^{r-(k+1)}\left\|B_{r-(k+1)}\right\|, \quad \left\|E_{r-(k+1),p}\right\| \le \gamma_n^{r-k}\|Y\|\|\varphi_{r-k} + E_{r-k}\|,$$

and

$$\left\|E_{r-(k+1),a}\right\| \le \gamma_1^{r-(k+1)}\left\|B_{r-(k+1)} + E_{r-(k+1),s} + Y\left(\varphi_{r-k} + E_{r-k}\right) + E_{r-(k+1),p}\right\|,$$

and we have

$$\begin{aligned}\left\|E_{r-(k+1)}\right\| \le \gamma_2^{r-(k+1)}\left\|B_{r-(k+1)}\right\| &+ \left(1 + \gamma_{n\theta_{r-k,r-(k+1)}+1}^{r-(k+1)}\right)\|Y\|\|E_{r-k}\| \\ &+ \gamma_{n\theta_{r-k,r-(k+1)}+1}^{r-(k+1)}\|Y\|\|\varphi_{r-k}\|,\end{aligned}$$

where $\varphi_{r-k} = Y^kB_r + Y^{k-1}B_{r-1} + \cdots + Y^2B_{r-k+2} + YB_{r-k+1} + B_{r-k}$. Then by writing the gamma constants in (A.1) as

$$\gamma_{f_{r-k,j}}^{r-k} = \gamma_{\theta_{r-k,r-(k+1)}f_{r-k,j}}^{r-k+1}, \quad j = r-k\colon r,$$

and by noting the bounds

$$\gamma^{r-k+1}_{\theta_{r-k,r-(k+1)}f_{r-k,j}}\left(1+\gamma^{r-(k+1)}_{n\theta_{r-k,r-(k+1)}+1}\right)+\gamma^{r-(k+1)}_{n\theta_{r-k,r-(k+1)}+1}\le\gamma^{r-k+1}_{\theta_{r-k,r-(k+1)}(n+f_{r-k,j})+1},$$

we have

$$\left\|E_{r-(k+1)}\right\|\le\gamma^{r-(k+1)}_{f_{r-(k+1),r}}\|Y\|^{k+1}\|B_r\|+\gamma^{r-(k+1)}_{f_{r-(k+1),r-1}}\|Y\|^{k}\|B_{r-1}\|+\cdots+\gamma^{r-(k+1)}_{f_{r-(k+1),r-(k+1)}}\left\|B_{r-(k+1)}\right\|,$$

where $f_{r-(k+1),r-(k+1)}=2$ and

$$\begin{aligned}f_{r-(k+1),r}&=\theta_{r-k,r-(k+1)}\left(n+f_{r-k,r}\right)+1\\&=n\theta_{r,r-(k+1)}+\\&\quad(n+1)\left(\theta_{r-1,r-(k+1)}+\theta_{r-2,r-(k+1)}+\cdots+\theta_{r-k,r-(k+1)}\right)+1,\end{aligned}$$

and similarly, $f_{r-(k+1),j}=\theta_{r-k,r-(k+1)}\left(n+f_{r-k,j}\right)+1$ for $j=r-k\colon r-1$, so

$$\begin{aligned}f_{r-(k+1),r-1}&=(n+2)\theta_{r-1,r-(k+1)}+\\&\quad(n+1)\left(\theta_{r-2,r-(k+1)}+\theta_{r-3,r-(k+1)}+\cdots+\theta_{r-k,r-(k+1)}\right)+1,\\&\vdots\\f_{r-(k+1),r-k+1}&=(n+2)\theta_{r-k+1,r-(k+1)}+(n+1)\theta_{r-k,r-(k+1)}+1,\\f_{r-(k+1),r-k}&=(n+2)\theta_{r-k,r-(k+1)}+1.\end{aligned}$$

This proves (A.1) for $i=k+1$ and so the proof is completed by induction.

Acknowledgments. I wish to thank Prof. Nick Higham for financial support and for valuable comments on several versions of the draft manuscript. I also thank Dr. Massimiliano Fasi for useful suggestions for improving the manuscript.

REFERENCES

[1] 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, https://doi.org/10.1137/09074721X.

[2] A. H. Al-Mohy, N. J. Higham, and X. Liu, *Arbitrary precision algorithms for computing the matrix cosine and its Fréchet derivative*, SIAM J. Matrix Anal. Appl., 43 (2022), pp. 233–256, https://doi.org/10.1137/21m1441043.

[3] A. H. Al-Mohy, N. J. Higham, and S. D. Relton, *New algorithms for computing the matrix sine and cosine separately or simultaneously*, SIAM J. Sci. Comput., 37 (2015), pp. A456–A487, https://doi.org/10.1137/140973979.

[4] M. Aprahamian and N. J. Higham, *Matrix inverse trigonometric and inverse hyperbolic functions: Theory and algorithms*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1453–1477, https://doi.org/10.1137/16M1057577.

[5] P. Bader, S. Blanes, and F. Casas, *Computing the matrix exponential with an optimized Taylor polynomial approximation*, Mathematics, 7 (2019), p. 1174, https://doi.org/10.3390/math7121174.

[6] M. Fasi, *Optimality of the Paterson–Stockmeyer method for evaluating matrix polynomials and rational matrix functions*, Linear Algebra Appl., 574 (2019), pp. 182–200, https://doi.org/10.1016/j.laa.2019.04.001.

[7] M. Fasi and N. J. Higham, *Multiprecision algorithms for computing the matrix logarithm*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 472–491, https://doi.org/10.1137/17M1129866.

[8] M. Fasi and N. J. Higham, *An arbitrary precision scaling and squaring algorithm for the matrix exponential*, SIAM J. Matrix Anal. Appl., 40 (2019), pp. 1233–1256, https://doi.org/10.1137/18M1228876.

[9] G. Hargreaves, *Topics in matrix computations: Stability and efficiency of algorithms*, PhD thesis, University of Manchester, Manchester, England, Aug. 2005.

[10] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002, https://doi.org/10.1137/1.9780898718027.

[11] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008, https://doi.org/10.1137/1.9780898717778.

[12] N. J. Higham, *Numerical stability of algorithms at extreme scale and low precisions*, MIMS EPrint 2021.14, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Sept. 2021, http://eprints.maths.manchester.ac.uk/2833/. To appear in Proc. Int. Cong. Math.

[13] N. J. Higham and M. Mikaitis, *Anymatrix: An extensible MATLAB matrix collection*. https://github.com/mmikaitis/anymatrix.

[14] N. J. Higham and M. Mikaitis, *Anymatrix: An extensible MATLAB matrix collection*, Numer. Algorithms, 90 (2021), pp. 1175–1196, https://doi.org/10.1007/s11075-021-01226-2.

[15] N. J. Higham and S. Pranesh, *Exploiting lower precision arithmetic in solving symmetric positive definite linear systems and least squares problems*, SIAM J. Sci. Comput., 43 (2021), pp. A258–A277, https://doi.org/10.1137/19M1298263.

[16] N. Hoffman, O. Schwartz, and S. Toledo, *Efficient evaluation of matrix polynomials*, in Parallel Processing and Applied Mathematics: 12th International Conference, PPAM 2017, Lublin, Poland, R. Wyrzykowski, J. Dongarra, E. Deelman, and K. Karczewski, eds., vol. 10777 of Lecture Notes in Computer Science, Springer-Verlag, Cham, Switzerland, 2018, pp. 24–35, https://doi.org/10.1007/978-3-319-78024-5_3.

[17] C. V. Loan, *A note on the evaluation of matrix polynomials*, IEEE Trans. Automat. Contr., 24 (1979), pp. 320–321, https://doi.org/10.1109/TAC.1979.1102005.

[18] *Multiprecision Computing Toolbox*. Advanpix, Tokyo, 2023, http://www.advanpix.com. Version 5.1.1.15444.

[19] M. S. Paterson and L. J. Stockmeyer, *On the number of nonscalar multiplications necessary to evaluate polynomials*, SIAM J. Comput., 2 (1973), pp. 60–66, https://doi.org/10.1137/0202007.

[20] J. Sastre, J. Ibáñez, P. Alonso, J. Peinado, and E. Defez, *Two algorithms for computing the matrix cosine function*, Appl. Math. Comput., 312 (2017), pp. 66–77, https://doi.org/10.1016/j.amc.2017.05.019.

[21] J. Sastre, J. Ibáñez, E. Defez, and P. Ruiz, *New scaling-squaring taylor algorithms for computing the matrix exponential*, SIAM J. Sci. Comput., 37 (2015), pp. A439–A455, https://doi.org/10.1137/090763202.