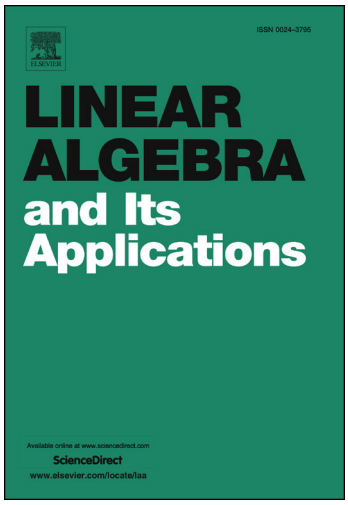

On the algorithm by Al-Mohy and Higham for computing the action of the matrix exponential: A posteriori roundoff error estimation

Thomas M. Fischer

PII: S0024-3795(17)30342-7
DOI: http://dx.doi.org/10.1016/j.laa.2017.05.042
Reference: LAA 14190

To appear in: *Linear Algebra and its Applications*

Received date: 23 September 2016
Accepted date: 25 May 2017

Please cite this article in press as: T.M. Fischer, On the algorithm by Al-Mohy and Higham for computing the action of the matrix exponential: A posteriori roundoff error estimation, *Linear Algebra Appl.* (2017), http://dx.doi.org/10.1016/j.laa.2017.05.042

On the algorithm by Al-Mohy and Higham for computing the action of the matrix exponential: A posteriori roundoff error estimation

Thomas M. Fischer

Department of Mathematics and Natural Sciences, University of Applied Sciences, D-64295 Darmstadt, Germany

Abstract

The algorithm by Al-Mohy and Higham (2011) computes an approximation to $e^A b$ for given A and b, where A is an n-by-n matrix and b is, for example, a vector of dimension n. It uses a scaling together with a truncated Taylor series approximation to the exponential of the scaled matrix. In this paper, a method is developed for estimating the roundoff error of the computed solution. An asymptotic expansion of this error for small values of the unit roundoff is the basis of the method. The roundoff error is further expressed in terms of sums of rounding errors, which occur during the computation. A second approximation to $e^A b$, which is computed with a lower precision than the first one, is used to evaluate these rounding errors. The result is an upper bound on the normwise relative roundoff error. Further, an algorithm is proposed for computing the error bound. The cost for performing this algorithm depends on the type of problem and the accuracy, which is required for the error estimate. In case that all computations are performed with standard precisions, this cost can be expressed in terms of the number of computed matrix-vector products and is bounded from above by two times the cost for computing $e^A b$.

Keywords: matrix exponential, Taylor series, roundoff error, asymptotic expansion, a posteriori error estimate
2010 MSC: 65F60

Email address: thomas.fischer@h-da.de (Thomas M. Fischer)

1. Introduction

Special numerical techniques are needed to compute $e^A b$ for $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$, if A is large and sparse. A computation of e^A should be avoided in this case, since the exponential of a sparse matrix is generally full. Methods, which consider a sparsity of A, can be used, for example, as part of exponential integrators for solving large systems of ordinary differential equations (see e.g. [16]). There is a great variety of such methods, let us briefly describe some of them:

Krylov subspace methods reduce the $e^A b$ problem to a corresponding problem for a small dense matrix (see e.g. [11], [8], [15]). For that purpose, projection techniques like the Lanczos and Arnoldi processes are used to build an orthonormal basis of the Krylov subspace, which is defined with respect to A and b. The influence of finite precision arithmetic on the accuracy of the Lanczos approximation to $e^A b$ was analysed, for example, in [9], where A was assumed to be real symmetric.

Chebyshev methods, which are mainly designed to compute $e^A b$ for real symmetric A, are based on expansions of the scalar exponential function in Chebyshev polynomials (see e.g. [3]). The method by Sheehan et al. [19] uses a Laguerre series expansion instead, whereas an interpolation of the scalar exponential function at Leja points is the basis of the method by Caliari et al. [4], [5]. The use of Chebyshev methods requires some knowledge of the spectrum of A, as is also true for the Leja point method. In addition, a scaling of A is generally needed to improve the accuracy of the computed solution in case of both the Laguerre and the Leja point method.

The methods mentioned so far use approximations to $e^A b$, which are polynomial in A. These methods multiply A into vectors only and are therefore particularly efficient when A is large and sparse. In contrast to that, rational approaches like rational Krylov methods (see e.g. [12] and references therein) and contour integral methods (see e.g. [21]) reduce the $e^A b$ problem to the solution of linear systems of equations, whose coefficient matrices are shifted versions of A. The applicability of such rational approaches depends on how efficient these linear systems can be solved.

Our main focus of interest is the algorithm by Al-Mohy and Higham [2]. This algorithm computes an approximation to $e^A b$ of the form

$$e^A b \approx \left(T_m(s^{-1}A)\right)^s b, \tag{1.1}$$

where s is a positive integer, which is used to perform a scaling of A, and

$T_m(s^{-1}A)$ is the m-th order truncated Taylor series approximation to the exponential of the scaled matrix for some $m \geq 1$. The parameters m and s are determined so that the normwise relative backward error of this approximation is bounded by a given tolerance $\delta > 0$. Further, A is generally replaced by a certain shifted matrix in (1.1). With regard to accuracy and efficiency, this algorithm was compared with a large number of different methods for computing $e^A b$ (see [2], [4]). The concept of a backward error tolerance has recently been extended to the Leja method in [5].

The present analysis deals with the question how far rounding errors can affect the accuracy of a solution, which is computed to $e^A b$ by using the algorithm by Al-Mohy and Higham. It is assumed that computations are performed in floating point arithmetic, with $\epsilon > 0$ being the unit roundoff. In addition it is helpful to assign the $e^A b$ problems to the following cases:

(P1) The norm of the scaled matrix is small, which means that the number s of iterations for undoing the scaling may be large.

(P2) The norm of the scaled matrix is large.

Since the scaling condition in [2] is based on a certain nonnegative functional of A, the second case can happen only if A is nonnormal. A strong influence of rounding errors is further restricted to $e^A b$ problems having a large condition number, provided that the algorithm behaves in a forward stable manner.

The stability of some one-step methods for computing $e^A b$ was analysed in [10]. The obtained results were shown to apply also to the algorithm by Al-Mohy and Higham, provided that problems are of Type (P1) and $\delta = \epsilon$. According to these results, the algorithm is backward stable for Hermitian A and forward stable, for example, for skew-Hermitian A. It further follows from that analysis that the computed vector has a small normwise relative backward error, although in a general sense, for all problems of Type (P1). Problems of Type (P2) were, however, not considered in [10].

The aim of this paper is to develop a method, which can be used to estimate the roundoff error of the computed solution for both types of problems. An asymptotic expansion of this error for small values of $m\epsilon$ forms the basis of the method. In this context, the parameter m is considered to be a function of δ in order to combine δ with ϵ when ϵ decreases. The roundoff error is further expressed in terms of sums of rounding errors, which occur during the computation. To evaluate these rounding errors in practice, the algorithm is performed a second time and an approximation to $e^A b$ is computed by simulating an arithmetic of lower precision. It is shown that, if

the unit roundoff ϵ_1 associated with the lower precision satisfies $\epsilon \ll \epsilon_1 \leq \delta$, then the roundoff error of this approximation is of the order $m\epsilon_1$ and can be determined by neglecting higher order terms. The roundoff error of the solution computed with precision ϵ can therefore be estimated *a posteriori* by using the roundoff error of the solution computed with precision ϵ_1 and the difference of both approximations to $e^A b$. The result is an upper bound on the normwise relative roundoff error. It also applies to the case that $b \in \mathbb{C}^{n \times n_0}$ for some $n_0 \leq n$.

Further, an algorithm is proposed for computing the error bound. In case that ϵ and ϵ_1 are defined by standard precisions, the cost for performing this algorithm can be expressed in terms of the number of computed matrix-vector products and is bounded from above by two times the cost for computing $e^A b$. The computed solution can be regarded as sufficiently accurate, if the estimated normwise relative roundoff error is of the order $m\epsilon_1$. By varying ϵ_1, the accuracy of the error estimate can be controlled, if necessary. Numerical experiments show the applicability of the method to the kind of problems, which are typically computed by using the algorithm by Al-Mohy and Higham.

Information about the extent to which rounding errors can be amplified is also contained in the condition number of the $e^A b$ problem. However, the cost for computing an upper bound on the condition number is relatively high, especially for large values of s, since it is proportional to the square of the cost for computing $e^A b$ (see [7]).

The paper is organized as follows. The main features of the algorithm by Al-Mohy and Higham are described in Section 2, with emphasis on the choice of the parameter s. In addition it is analysed in Section 3 how the parameter m can be defined when δ goes to zero. An asymptotic expansion of the roundoff error for small values of $m\epsilon$ is given in Section 4. This expansion is basic for deriving both a priori and a posteriori error estimates. A result on forward stability, which differs from that in [10] in that the values for δ and ϵ may be different, is shown in Section 5. However, this result applies to certain problems only. Therefore, the main focus of the paper is on a general method for estimating the roundoff error. This method, which performs an a posteriori error estimation on the basis of two different approximations to $e^A b$, is described and analysed in Section 6. An algorithm for computing the error bound is formulated in Section 7. Some numerical experiments are given in Section 8. Conclusions for the practical use of the estimation procedure are summarized in Section 9.

Unless otherwise stated, $\|\cdot\|$ is any vector norm or the corresponding subordinate matrix norm.

2. Features of the algorithm

The algorithm by Al-Mohy and Higham [2, Algorithm 3.2] computes $e^A b$ for $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^{n\times n_0}$, with n_0 being sufficiently small. Without loss of generality, we assume that $n_0 = 1$, i.e. $b \in \mathbb{C}^n$. For preprocessing the data, the algorithm uses two techniques: balancing (the use of which is optional) and a shifting of A. The shifted matrix is given by $A - \mu I$ with

$$\mu := \frac{1}{n}\mathrm{trace}(A), \tag{2.1}$$

where it is assumed throughout this paper that $A-\mu I \neq 0$. (The case $A = \mu I$ is trivial.) Balancing is based on a diagonal similarity transformation of A and is not always recommended (see [2, p. 496]). We omit balancing, since otherwise the results of an error analysis depend on the condition number of the transformation matrix.

For reasons of numerical stability, the norm or, as will be discussed further below, a certain nonnegative functional of the shifted matrix is reduced by using a scaling, if necessary. Let $A_1 = s^{-1}(A - \mu I)$ be the scaled and shifted matrix, where s is a positive integer, which is assumed to be sufficiently large. The matrix exponential of A_1 is then approximated by a truncated Taylor series of the form

$$T_m(A_1) = \sum_{i=0}^{m} \frac{1}{i!} A_1^i, \tag{2.2}$$

with a degree $m \geq 1$.

The backward error of this approximation was analysed in [2, p. 493] to determine an optimal choice of m and s. It follows from that analysis that for every $\delta > 0$ there exist positive integers m and s such that $T_m(A_1) = e^{A_1+E_m}$ for some matrix $E_m \in \mathbb{C}^{n\times n}$, which commutes with A_1 and satisfies the estimate

$$\|E_m\| \leq \delta \|A_1\|. \tag{2.3}$$

In fact, for given positive integers m and p, for which $m + 1 \geq p(p-1)$, it is sufficient to choose

$$s := \max\left(\left\lceil \frac{\alpha_p(A - \mu I)}{\theta_m} \right\rceil, 1\right) \tag{2.4}$$

for satisfying (2.3), where

$$\alpha_p(X) := \max\left(\|X^p\|^{\frac{1}{p}}, \|X^{p+1}\|^{\frac{1}{p+1}}\right) \leq \|X\|$$

is a nonnegative functional of $X \in \mathbb{C}^{n\times n}$ and $\theta_m > 0$ is a certain constant, which depends on δ (see [2, p. 494]). For the further analysis, it is helpful to describe in more detail how this constant is defined:

We first note that, if s is sufficiently large, then the principal logarithm of $e^{-A_1}T_m(A_1)$ exists and has a power series expansion of the form

$$h_{m+1}(A_1) := \log\left(e^{-A_1}T_m(A_1)\right) = \sum_{i=m+1}^{\infty} \lambda_i A_1^i \tag{2.5}$$

with $\lambda_i \in \mathbb{R}$ for $i = m+1, m+2, \ldots$ (see [2, p. 493]). Let $\rho_1 > 0$ be the radius of convergence of the series $\sum_{i=m+1}^{\infty} \lambda_i \rho^i$ and let

$$\widetilde{h}_{m+1}(\rho) := \sum_{i=m+1}^{\infty} |\lambda_i|\, \rho^i \tag{2.6}$$

for $0 \leq \rho < \rho_1$. According to a result of [1, Theorem 4.2] we then have

$$\|h_{m+1}(A_1)\| \leq \widetilde{h}_{m+1}(\alpha_p(A_1)), \tag{2.7}$$

provided that $\alpha_p(A_1) < \rho_1$. In addition, for some selected values of δ and m, it was shown in [2, Table 3.1] that the expression

$$\theta_m := \max\left\{0 < \rho < \rho_1 : \frac{\widetilde{h}_{m+1}(\rho)}{\rho} \leq \delta\right\} \tag{2.8}$$

is well defined. For that purpose, high precision arithmetic was used to evaluate the series in (2.6), and a zero-finder to compute the maximum value in (2.8) (see [14, Appendix]). It follows from (2.4), (2.7), and (2.8) that $\|h_{m+1}(A_1)\| \leq \delta\alpha_p(A_1)$ and, hence, (2.3) is satisfied with $E_m = h_{m+1}(A_1)$.

The values of m and p are finally determined so that the algorithm minimizes the number of computed matrix-vector products, where this number is regarded as a measure for the computational cost. However, since computations are performed with finite precision, m is bounded from above by some positive integer $m_{\max}$ (see [2, p. 494]).

We remark that a scaling condition of the type (2.4) was first proposed in [1]. In some cases, for example when A is normal and $\|\cdot\|$ is the matrix 2-norm, this condition simplifies to $s = q$, with

$$q := \left\lceil \frac{\|A - \mu I\|}{\theta_m} \right\rceil . \tag{2.9}$$

Note that $q \geq 1$ because of $A - \mu I \neq 0$.

Generally we have $s \leq q$, where s and q can differ by orders of magnitude if A is nonnormal and $\|A - \mu I\|$ is large (see e.g. [2, Eq. (3.8)]). Therefore, the purpose of using $\alpha_p(A-\mu I)$ instead of $\|A - \mu I\|$ in (2.4) is to reduce the number of iterations for undoing the scaling. This helps, for example, to avoid overscaling phenomena (see [1, p. 974]). Moreover, it makes the algorithm more efficient and often more accurate, as was observed experimentally in a similar context (see [1, pp. 985-988]).

Note further that the $e^A b$ problem is of Type (P1) if $s \approx q$ (i.e. if $\|A - \mu I\|$ is small or $\alpha_p(A - \mu I) \approx \|A - \mu I\|$) and of Type (P2) otherwise.

The main part of the algorithm by Al-Mohy and Higham [2, Algorithm 3.2] can now be described as follows:

Algorithm 2.1. Let $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$ and let μ be defined by (2.1). Further, for a given error tolerance $\delta > 0$, let s and θ_m be defined by (2.4) and (2.8), respectively, and let $A_1 = s^{-1}(A - \mu I)$. This algorithm then computes $e^A b$ on the basis of two nested loops: In the outer loop, an approximation v_s of $e^A b$ is determined by computing

$$v_k = e^{s^{-1}\mu} \left(T_m(A_1) v_{k-1}\right) \tag{2.10}$$

successively for $k = 1, \ldots, s$, with $v_0 = b$. In the inner loop, $r_m^{(k)} = T_m(A_1)v_{k-1}$ in (2.10) is evaluated by computing

$$r_j^{(k)} = r_{j-1}^{(k)} + w_j^{(k)}, \quad w_j^{(k)} = \frac{1}{sj}\left((A - \mu I)\, w_{j-1}^{(k)}\right) \tag{2.11}$$

successively for $j = 1, \ldots, m$, with $r_0^{(k)} = w_0^{(k)} = v_{k-1}$.

In addition, a premature terminating of the truncated Taylor series is used to reduce the number of iterations in (2.11) (see [2, Eq. (3.15)]). We do not consider an early termination of the algorithm until we come back to this point in Section 7.

3. Approximations for small error tolerances

In this section we analyse how m can be defined so that θ_m is bounded from below by some constant $\theta > 0$ when δ goes to zero. To begin with, we replace θ_m in (2.4) by any $\theta > 0$ and consider m a function of δ and θ. This is more convenient than to deal with condition (2.4) directly, since a numerical evaluation of θ_m can be avoided.

It follows from a result of [10, Lemma 3.1] that, for every $\theta > 0$ and every $A_1 \in \mathbb{C}^{n\times n}$ with $\|A_1\| \leq \theta$, the value of m can be chosen so that it does not grow faster than a constant times $\log \frac{1}{\delta}$ as $\delta \to 0$ for satisfying (2.3). The following lemma shows that this result remains true if $\|A_1\| \leq \theta$ is replaced by the weaker condition $\alpha_p(A_1) \leq \theta$:

Lemma 3.1. *Let $\eta > 0$ be the solution of $\eta e^{1+\eta} = 1$. Let $\theta > 0$ and let*

$$0 < \delta \leq \delta_1 := \frac{e^\theta}{c_1} \left(\frac{\theta e}{m_1}\right)^{m_1}, \tag{3.1}$$

where $m_1 := \left\lceil \frac{\theta}{\eta} \right\rceil$ and $c_1 := \left((2\pi (m_1+1))^{\frac{1}{2}} - 1 \right) \theta$. Further let $m = \overline{m}(\delta)$ be a positive integer defined by

$$m \left(\frac{c_1\delta}{e^\theta}\right)^{\frac{1}{m}} \leq \theta e < (m+1) \left(\frac{c_1\delta}{e^\theta}\right)^{\frac{1}{m+1}}. \tag{3.2}$$

We then have

$$0 \leq m - m_1 \leq \frac{1}{\eta} \log \frac{\delta_1}{\delta}. \tag{3.3}$$

In addition for every positive integer p satisfying $m + 1 \geq p\,(p-1)$ and every $A_1 \in \mathbb{C}^{n\times n}$ with $\alpha_p(A_1) \leq \theta$ there exists a matrix $E_m \in \mathbb{C}^{n\times n}$ such that $T_m(A_1) = e^{A_1+E_m}$ and $\|E_m\| \leq \delta\alpha_p(A_1)$.

Proof. As was shown in [10, Lemma 3.1], m is well defined by (3.2) and satisfies (3.3). Further, we want to express E_m in terms of the principal logarithm of $e^{-A_1}T_m(A_1)$, i.e. $E_m = \log(I - F_m)$, where $F_m := I - e^{-A_1}T_m(A_1)$. For that purpose, it is sufficient to show that $\|F_m\| < 1$ (see e.g. [18, Lemma 1]):

Note that F_m admits the representation

$$F_m = \frac{1}{m!} A_1^{m+1} \int_0^1 e^{-tA_1} t^m dt$$

(see e.g. [18, p. 32]), which can be used to expand F_m in a power series. This series is given by $F_m = \sum_{i=m+1}^{\infty} \omega_i A_1^i$, with

$$\omega_i = \frac{1}{m!} \frac{(-1)^{i-m-1}}{i\,(i-m-1)!}$$

for $i = m+1, m+2, \ldots$ Let

$$\phi_m(\rho) := \sum_{i=m+1}^{\infty} |\omega_i|\, \rho^i \tag{3.4}$$

for $\rho \geq 0$. We then conclude from a result of [1, Theorem 4.2] that $\|F_m\| \leq \phi_m(\alpha_p(A_1))$, provided that p is a positive integer satisfying $m+1 \geq p\,(p-1)$. Since $\alpha_p(A_1) \leq \theta$ and

$$\phi_m(\rho) = \frac{1}{m!}\rho^{m+1} \sum_{i=0}^{\infty} \frac{1}{(i+m+1)\,i!}\rho^i \leq \frac{1}{(m+1)!}\rho^{m+1}e^{\rho}$$

for $\rho \geq 0$, we therefore have

$$\|F_m\| \leq \phi_m(\alpha_p(A_1)) \leq \frac{1}{(m+1)!}\alpha_p(A_1)\theta^m e^{\theta}.$$

To estimate this further, we can proceed similar as in [10, p. 6] and obtain

$$\|F_m\| \leq \phi_m(\alpha_p(A_1)) \leq (2\pi\,(m_1+1))^{-\frac{1}{2}} \min\left(\frac{c_1}{\theta}\delta\alpha_p(A_1), 1\right), \tag{3.5}$$

from which follows that $\|F_m\| < 1$ and $\|E_m\| \leq \frac{\|F_m\|}{1-\|F_m\|} \leq \delta\alpha_p(A_1)$. $\square$

Note that for every positive integer m there exists $\theta > 0$ so that (3.1) and (3.2) are satisfied, if only $\delta > 0$ is sufficiently small. Assume, for example, that $m \leq m_{\max} = 55$ (see [2, p. 494]). It is then sufficient to choose

$$\delta \leq \left(\left((2\pi\,(m_{\max}+1))^{\frac{1}{2}} - 1\right)\eta m_{\max}\right)^{-1} \approx 0.0037$$

for satisfying (3.1) and (3.2) for some $\theta > 0$.

The result of the next lemma shows how the definitions of θ_m and $m = \overline{m}(\delta)$ in (2.8) and (3.2) are related to each other:

Lemma 3.2. *Let $\theta > 0$ and let $\delta > 0$ be sufficiently small so that* (3.1) *is satisfied. Further let $m = \overline{m}(\delta)$ be a positive integer defined by* (3.2) *and*

assume that the maximum value in (2.8) *exists so that* θ_m *is well defined. We then have*

$$\theta_m \geq \theta. \tag{3.6}$$

Proof. Let $A_1 \in \mathbb{C}^{n\times n}$ be a matrix so that $\alpha_p(A_1) \leq \theta$ for some positive integer p satisfying $m+1 \geq p\,(p-1)$. With the notation from the proof of Lemma 3.1 we then have

$$E_m = \log\,(I - F_m) = -\sum_{j=1}^{\infty} \frac{F_m^j}{j},$$

since $\|F_m\| < 1$. By using the power series expansion of F_m, we further obtain $E_m = \sum_{i=m+1}^{\infty} \lambda_i A_1^i$ with $\lambda_i = -\Pi_i(\omega_{m+1}, \ldots, \omega_i)$, where Π_i are polynomials in $\omega_{m+1}, \ldots, \omega_i$ having nonnegative coefficients for $i = m+1, m+2, \ldots$. Consequently we have

$$|\lambda_i| \leq \Pi_i(|\omega_{m+1}|\,, \ldots, |\omega_i|) \tag{3.7}$$

for $i = m+1, m+2, \ldots$. Note that E_m corresponds to $h_{m+1}(A_1)$ in (2.5).

Because of (3.7), the series in (2.6) and (3.4) satisfy the relation

$$\widetilde{h}_{m+1}(\rho) \leq \sum_{i=m+1}^{\infty} \Pi_i(|\omega_{m+1}|\,, \ldots, |\omega_i|)\rho^i = \sum_{j=1}^{\infty} \frac{\phi_m^j(\rho)}{j} \leq \frac{\phi_m(\rho)}{1-\phi_m(\rho)}$$

for all values of $\rho \geq 0$, for which $\phi_m(\rho) < 1$. It therefore follows from (3.5) (with ρ in place of $\alpha_p(A_1)$) that $\phi_m(\rho) < 1$ and $\widetilde{h}_{m+1}(\rho) \leq \delta\rho$ for $0 \leq \rho \leq \theta$. The proof then is an immediate result of (2.8). □

In the further sections of this paper it is assumed that m is the smallest positive integer, for which (3.6) is satisfied, for a given $\theta > 0$ and a sufficiently small $\delta > 0$. We then conclude from Lemma 3.2 that

$$m \leq \overline{m}(\delta). \tag{3.8}$$

In addition, the scaling parameter s is bounded from above by

$$s \leq \max\left(\left\lceil \frac{\alpha_p(A-\mu I)}{\theta} \right\rceil, 1\right) \tag{3.9}$$

for every positive integer p, for which $m+1 \geq p\,(p-1)$.

We remark that, in order that s is available for every positive integer m and every sufficiently small $\delta > 0$, we can use θ'_m in place of θ_m in (2.4), where $\theta'_m \leq \theta_m$ is the supremum of the set of all real numbers θ, which satisfy (3.2). It was shown in [10, p. 7] that θ'_m can easily be computed and differs only slightly from θ_m.

4. Asymptotic error analysis

The error, which is caused when $e^A b$ is computed by performing Algorithm 2.1 in floating point arithmetic, is the sum of an approximation error and a roundoff error, where the latter one represents all rounding errors, which occur during the computation, as a whole. The truncated Taylor series approximation to the matrix exponential was analysed in Section 2 and Section 3. We therefore concentrate on the roundoff error in this section. Our aim is to give an asymptotic expansion of this error for small values of $m\epsilon$. In addition, this error is expressed in terms of sums of rounding errors, which occur in forming the matrix-vector products and sums of two vectors in (2.11).

Without loss of generality, we can neglect an error of order ϵ in the data and assume that $A \in \mathbb{C}_{fl}^{n\times n}$ and $b \in \mathbb{C}_{fl}^{n}$, where $\mathbb{C}_{fl}$ denotes the set of all complex numbers, whose real and imaginary parts are floating point numbers. Further, all computations are assumed to be performed according to a model of arithmetic as described, for example, in [13, Eq. (2.4)].

We begin with an analysis of the method (2.11). Since the rounding errors, which are caused in computing $A - \mu I$, are negligible, we can assume that $A - \mu I \in \mathbb{C}_{fl}^{n\times n}$. In addition, we replace v_{k-1} by any vector $v \in \mathbb{C}_{fl}^{n}$ and drop the index k in $r_j^{(k)}$ and $w_j^{(k)}$. We then use the method (2.11) to compute $r_m = T_m(A_1)v$, where $A_1 \in \mathbb{C}^{n\times n}$ is defined by $A_1 = s^{-1}(A - \mu I)$. Let $\widetilde{r}_0 = \widetilde{w}_0 = v$ and let $\widetilde{r}_j$ and $\widetilde{w}_j$ be the computed vectors to r_j and w_j, respectively, for $j = 1, \ldots, m$. Assume further that the rounding errors, which occur in forming $(sj)^{-1}((A - \mu I)\widetilde{w}_{j-1})$ and $\widetilde{r}_{j-1} + \widetilde{w}_j$, add up to Δw_j and Δr_j, respectively, i.e.

$$\Delta r_j = \widetilde{r}_j - (\widetilde{r}_{j-1} + \widetilde{w}_j), \quad \Delta w_j = \widetilde{w}_j - \frac{1}{j} A_1 \widetilde{w}_{j-1} \tag{4.1}$$

for $j = 1, \ldots, m$.

In case A_1 has a sufficiently small norm, the numerical stability of the method (2.11) was studied in previous work: It follows from a result of [2, Lemma 4.1] that the computed vector $\widetilde{r}_m$ to r_m then has a normwise error of the order $m\epsilon$ and it was shown in [10, Lemma 3.2] that $\widetilde{r}_j - r_j = G_j v$ for some matrix $G_j \in \mathbb{C}^{n\times n}$, whose norm is of the order $j\epsilon$. In this context, norm means the 1-norm or the ∞-norm of a vector or a matrix.

In the following lemma, $|G_j|$ as well as $|D_j|$ and $|Z_j|$ are estimated, where $D_j, Z_j \in \mathbb{C}^{n\times n}$ are matrices, which can be used to express Δr_j and Δw_j

in terms of $\widetilde{r}_{j-1} + \widetilde{w}_j$ and $\widetilde{w}_{j-1}$, respectively. Note that, if $X = (x_{ik})$ and $Y = (y_{ik})$ are matrices of the same size, then $Y = |X|$ means that $y_{ik} = |x_{ik}|$ for all i and k.

Lemma 4.1. *Let m be a positive integer and assume that $\epsilon > 0$ is sufficiently small. Further let $\chi_n(\epsilon) := \frac{n+1}{1-n\epsilon}$ and let $\psi_n(\epsilon) := \frac{n+1}{1-(n+1)\epsilon}$. We then have*

$$\Delta r_j = D_j \left(\widetilde{r}_{j-1} + \widetilde{w}_j\right), \quad \Delta w_j = \frac{1}{j} Z_j \widetilde{w}_{j-1} \tag{4.2}$$

for some matrices $D_j, Z_j \in \mathbb{C}^{n\times n}$, where

$$|D_j| \leq \epsilon I, \quad |Z_j| \leq \chi_n(\epsilon)\epsilon \left|A_1\right| \tag{4.3}$$

for $j = 1, \ldots, m$. In addition we have $\widetilde{r}_j - r_j = G_j v$ for some matrix $G_j \in \mathbb{C}^{n\times n}$, where

$$|G_j| \leq \left(1 + \frac{1}{(n+1)\, j}\right) \psi_n(j\epsilon) j\epsilon T_j(|A_1|) \tag{4.4}$$

for $j = 1, \ldots, m$.

Proof. It follows from an analysis of the matrix-vector product (see e.g. [13]) and analyses of the division of a vector by a scalar and the sum of two vectors that

$$\widetilde{r}_j = (I + D_j)\left(\widetilde{r}_{j-1} + \widetilde{w}_j\right), \quad \widetilde{w}_j = \frac{1}{j}\left(I + C_j\right)\left(A_1 + H_j\right)\widetilde{w}_{j-1}$$

for some matrices $H_j, C_j, D_j \in \mathbb{C}^{n\times n}$, where

$$|H_j| \leq \frac{n\epsilon}{1-n\epsilon}\left|A_1\right|, \quad |C_j| \leq \epsilon I, \quad |D_j| \leq \epsilon I$$

for $j = 1, \ldots, m$. We conclude that (4.2) and (4.3) are satisfied by choosing $Z_j = H_j + C_j\left(A_1 + H_j\right)$ for $j = 1, \ldots, m$.

Let $P_j := \frac{1}{j!} A_1^j$ and let $\widetilde{P}_j := \frac{1}{j!}\left(A_1 + Z_j\right)\cdots\left(A_1 + Z_1\right)$, which means that $w_j = P_j v$ and $\widetilde{w}_j = \widetilde{P}_j v$ for $j = 1, \ldots, m$. We obtain from (4.3) that

$$\left|\widetilde{P}_j - P_j\right| \leq \left(\left(1 + \chi_n(\epsilon)\epsilon\right)^j - 1\right)\frac{1}{j!}\left|A_1\right|^j \leq \psi_n(j\epsilon) j\epsilon \frac{1}{j!}\left|A_1\right|^j \tag{4.5}$$

for $j = 1, \ldots, m$. It was further shown in [10, p. 8] that $\widetilde{r}_j - r_j = G_j v$, where

$$G_j = (I + D_j)\left(G_{j-1} + \widetilde{P}_j - P_j\right) + D_j T_j(A_1)$$

for $j = 1, \ldots, m$, with $G_0 = 0$. By using (4.3) and (4.5) we then obtain

$$|G_j| \leq (1+\epsilon)\left(|G_{j-1}| + \psi_n(j\epsilon) j\epsilon \frac{1}{j!}\,|A_1|^j\right) + \epsilon T_j(|A_1|),$$

for $j = 1, \ldots, m$, from which the proof follows by induction. □

We can now use the result of Lemma 4.1 to analyse Algorithm 2.1. Let $\widetilde{v}_0 = b$ and let $\widetilde{v}_k$ be the computed vector to v_k for $k = 1, \ldots, s$, which means that $\widetilde{v}_s$ is the computed solution to $e^A b$. Further let $\widetilde{r}_0^{(k)} = \widetilde{w}_0^{(k)} = \widetilde{v}_{k-1}$, let $\widetilde{r}_j^{(k)}$ and $\widetilde{w}_j^{(k)}$ be the computed vectors to $r_j^{(k)}$ and $w_j^{(k)}$, respectively, and let

$$\Delta r_j^{(k)} = \widetilde{r}_j^{(k)} - \left(\widetilde{r}_{j-1}^{(k)} + \widetilde{w}_j^{(k)}\right), \quad \Delta w_j^{(k)} = \widetilde{w}_j^{(k)} - \frac{1}{j} A_1 \widetilde{w}_{j-1}^{(k)} \tag{4.6}$$

for $j = 1, \ldots, m$ and $k = 1, \ldots, s$.

Note that $\widetilde{v}_{k-1}$ can be interpreted as the initial vector of the method (2.10), which means that $v = \widetilde{v}_{k-1}$ in Lemma 4.1. As a result, some other variables in Lemma 4.1 vary with k and are therefore supplied with an index k, for example $D_j^{(k)}$, $Z_j^{(k)}$, and $G_j^{(k)}$ are used in place of D_j, Z_j, and G_j, respectively. We then obtain

$$\widetilde{r}_j^{(k)} = \left(T_j(A_1) + G_j^{(k)}\right) \widetilde{v}_{k-1}$$

for $j = 1, \ldots, m$ and $k = 1, \ldots, s$. In addition the rounding errors, which occur in forming the product of $e^{s^{-1}\mu}$ and the computed vector to $T_m(A_1)v_{k-1}$ in (2.10), are negligible and can be ignored. Consequently we have

$$\widetilde{v}_k = e^{s^{-1}\mu}\widetilde{r}_m^{(k)} = e^{s^{-1}\mu}\left(T_m(A_1) + G_m^{(k)}\right)\widetilde{v}_{k-1} \tag{4.7}$$

for $k = 1, \ldots, s$, with $\widetilde{v}_0 = b$.

The following result shows that the roundoff error $\widetilde{v}_s - v_s$ of the computed solution has an asymptotic expansion for small values of $m\epsilon$, where the first order term of this expansion is a linear function in $G_m^{(1)}, \ldots, G_m^{(s)}$:

Theorem 4.1. *Assume that $\epsilon > 0$ and $\delta \geq \epsilon$ are sufficiently small. Let $\theta > 0$ and let m be the smallest positive integer, for which $\theta_m \geq \theta$. We then have*

$$\|\widetilde{v}_s - v_s\| = O(m\epsilon), \quad \|\widetilde{v}_s - v_s - \xi_s\| = O((m\epsilon)^2), \tag{4.8}$$

with $m \leq \overline{m}(\epsilon)$ and

$$\xi_s = e^{\mu} \sum_{k=1}^{s} \left((T_m(A_1))^{s-k}\, G_m^{(k)}\, (T_m(A_1))^{k-1}\, b\right). \tag{4.9}$$

Proof. Since $\delta \geq \epsilon$, it follows from (3.2) that $\overline{m}(\delta) \leq \overline{m}(\epsilon)$. We therefore obtain from (3.8) that $m \leq \overline{m}(\epsilon)$. Note further that s is bounded from above by a constant, which does not depend on δ or ϵ due to (3.9).

Because of (2.10) and (4.7) we have

$$v_k = \left(e^{s^{-1}\mu} T_m(A_1)\right)^k b, \quad \widetilde{v}_k = e^{s^{-1}\mu k} \left(T_m(A_1) + G_m^{(k)}\right) \cdots \left(T_m(A_1) + G_m^{(1)}\right) b$$

for $k = 1, \ldots, s$. The proof then follows from (4.4). □

It is further important in order to estimate the roundoff error a posteriori that $\widetilde{v}_s - v_s$ can be expressed in terms of $\Delta r_j^{(k)}$ and $\Delta w_j^{(k)}$: We conclude from (2.10), (2.11), and (4.6) that $x_k = \widetilde{v}_k - v_k$ admits the representation

$$x_k = e^{s^{-1}\mu} \sigma_m^{(k)} \tag{4.10}$$

for $k = 1, \ldots, s$, where $\sigma_j^{(k)} = \widetilde{r}_j^{(k)} - r_j^{(k)}$ and $y_j^{(k)} = \widetilde{w}_j^{(k)} - w_j^{(k)}$ satisfy the equations

$$\sigma_j^{(k)} = \sigma_{j-1}^{(k)} + y_j^{(k)} + \Delta r_j^{(k)}, \quad y_j^{(k)} = \frac{1}{j} A_1 y_{j-1}^{(k)} + \Delta w_j^{(k)} \tag{4.11}$$

for $j = 1, \ldots, m$, with $\sigma_0^{(k)} = y_0^{(k)} = x_{k-1}$ and $x_0 = 0$. In this context, we use the notation

$$\widetilde{v}_s - v_s = Q(\Delta_s), \quad \Delta_s = \left(\Delta r_1^{(1)}, \Delta w_1^{(1)}, \ldots, \Delta r_m^{(s)}, \Delta w_m^{(s)}\right), \tag{4.12}$$

where $Q : \mathbb{C}^{n \times 2sm} \to \mathbb{C}^n$ is a linear operator, which is defined by (4.10) and (4.11).

5. A result on forward stability

In this section, the roundoff error of the computed solution is estimated in norm, where the estimated error is related to the condition number of the $e^A b$ problem. The result extends that of [10, Theorem 7.2], which applies when $\delta = \epsilon$, to the case that $\delta > \epsilon$. It gives an insight into the stability of the algorithm and is useful for a first assessment of the accuracy of a solution. This result is, however, relevant only to certain problems of Type (P1).

Let U be the space of all continuous functions $\mathcal{F} : [0, 1] \to \mathbb{C}^{n \times n}$, let $|||\mathcal{F}||| := \max_{0 \leq t \leq 1} \|\mathcal{F}(t)\|$ be the norm for $\mathcal{F} \in U$, and let

$$L(X, \mathcal{F}) := \int_0^1 e^{(1-t)X} \mathcal{F}(t) e^{tX} dt$$

for $X \in \mathbb{C}^{n\times n}$ and $\mathcal{F} \in U$. Further let

$$\lambda(A,b) := \max_{F\in\mathbb{C}^{n\times n},\|F\|\le 1} \|L(A,F)b\|, \quad \Lambda(A,b) := \sup_{\mathcal{F}\in U,|||\mathcal{F}|||\le 1} \|L(A,\mathcal{F})b\|$$

and let

$$\omega := \frac{\Lambda(A,b)}{\lambda(A,b)}, \quad \overline{\kappa}_{rel,1} := \frac{\lambda(A,b)\,\|A\| + \left\|e^A\right\|\,\|b\|}{\|e^A b\|} \tag{5.1}$$

for $b \neq 0$. Note that $\Lambda(A,b)$ corresponds to "$\mu(A,b)$" in [10, Eq. (4.11)] and $\overline{\kappa}_{rel,1}$ is the relative condition number, which was defined for the $e^A b$ problem in [10, Eq. (5.5)].

The following result shows that the roundoff error $\widetilde{v}_s - v_s$ of the computed solution can be estimated in terms of ω, $\overline{\kappa}_{rel,1}$, and the norm of $A_1 = s^{-1}(A - \mu I)$:

Theorem 5.1. *Assume that $\epsilon > 0$ and $\delta \ge \epsilon$ are sufficiently small. Let $\theta > 0$ and let m be the smallest positive integer, for which $\theta_m \ge \theta$. Further let $\|\cdot\|$ be the 1, 2, or ∞-norm, let $b \neq 0$, and let $v_s \neq 0$. We then have*

$$\frac{\|\widetilde{v}_1 - v_1\|}{\|v_1\|} \le \gamma e^{2\|A_1\|}\overline{\kappa}_{rel,1} m\epsilon + O(m\epsilon(\delta + m\epsilon)), \quad \text{if } s = 1, \tag{5.2}$$

$$\frac{\|\widetilde{v}_s - v_s\|}{\|v_s\|} \le 8\gamma\frac{e^{2\|A_1\|}}{\theta}\omega\overline{\kappa}_{rel,1} m\epsilon + O(m\epsilon(\delta + m\epsilon)), \quad \text{if } s \ge 2, \tag{5.3}$$

where $\gamma = \gamma_n := \left(1 + \frac{1}{(n+1)m}\right)\psi_n(m\epsilon)$, if $\|\cdot\|$ is the 1 or ∞-norm, and $\gamma = \gamma_{n^{3/2}}$, if $\|\cdot\|$ is the 2-norm.

Proof. As was described in Section 2, we have $T_m(A_1) = e^{A_1+E_m}$ for some matrix $E_m \in \mathbb{C}^{n\times n}$, which commutes with A_1 and satisfies (2.3). It follows that

$$\left\|\left(e^{s^{-1}\mu}T_m(A_1)\right)^k - e^{s^{-1}kA}\right\| = O(\delta), \quad \left\|v_k - e^{s^{-1}kA}b\right\| = O(\delta) \tag{5.4}$$

for $k = 1,\dots,s$. Further, in case that $\|\cdot\|$ is the 1 or ∞-norm, we obtain from (4.4) that

$$\left\|G_m^{(k)}\right\| \le \gamma e^{\|A_1\|} m\epsilon \tag{5.5}$$

for $k = 1,\dots,s$, with $\gamma = \gamma_n$. It can be shown that (5.5) remains true in case that $\|\cdot\|$ is the 2-norm, if only $\gamma = \gamma_{n^{3/2}}$ is used in place of $\gamma = \gamma_n$. We then

conclude from (4.8), (5.4), and (5.5) that

$$\left\| \widetilde{v}_s - v_s - e^{\mu} \sum_{k=1}^{s} \left(e^{(s-k)A_1} G_m^{(k)} e^{(k-1)A_1} b \right) \right\| = O(m\epsilon\,(\delta + m\epsilon)).$$

This means that

$$\left\| \widetilde{v}_s - v_s - s e^{\mu} L_s(A - \mu I, G_m^{(1)}, \ldots, G_m^{(s)}) b \right\| = O(m\epsilon\,(\delta + m\epsilon)), \tag{5.6}$$

where $L_s(X, F_1, \ldots, F_s)$ is a discrete form of $L(X, \mathcal{F})$, which is defined by

$$L_s(X, F_1, \ldots, F_s) := \frac{1}{s} \sum_{k=1}^{s} e^{\left(1 - s^{-1}k\right)X} F_k e^{s^{-1}(k-1)X}$$

for $X, F_1, \ldots, F_s \in \mathbb{C}^{n \times n}$ (see [10, Eq. (6.1)]).

If $s = 1$, then $A_1 = A - \mu I$. We conclude from (5.5) and (5.6) that

$$\|\widetilde{v}_1 - v_1\| \leq \gamma e^{2\|A_1\|} \left\| e^{A} \right\| \|b\|\, m\epsilon + O(m\epsilon\,(\delta + m\epsilon)),$$

from which follows that (5.2) is satisfied because of (5.1) and (5.4).

If $s \geq 2$, then

$$s < \frac{2}{\theta_m} \alpha_p(A - \mu I) \leq \frac{2}{\theta_m} \|A - \mu I\| \leq \frac{4}{\theta_m} \|A\|, \tag{5.7}$$

with $\theta_m \geq \theta$. According to a result of [10, Lemma 6.2], there exists $\mathcal{F} \in U$ so that

$$e^{\mu} L_s(A - \mu I, G_m^{(1)}, \ldots, G_m^{(s)}) = e^{\mu} L(A - \mu I, \mathcal{F}) = L(A, \mathcal{F}), \tag{5.8}$$

where

$$|||\mathcal{F}||| \leq 2e^{\|A_1\|} \max_{1 \leq k \leq s} \left\| G_m^{(k)} \right\|. \tag{5.9}$$

Since $\|L(A, \mathcal{F})b\| \leq \Lambda(A, b)\, |||\mathcal{F}|||$, we obtain from Eqs. (5.5) to (5.9) that

$$\|\widetilde{v}_s - v_s\| \leq 8\gamma \frac{e^{2\|A_1\|}}{\theta} \Lambda(A, b)\, \|A\|\, m\epsilon + O(m\epsilon\,(\delta + m\epsilon)).$$

The proof then is an immediate result of (5.1) and (5.4). □

Note that $\|A_1\| \leq s^{-1} q \theta_m$ because of (2.9). In addition we have $\theta_m \approx \theta$, since m is the smallest positive integer, for which $\theta_m \geq \theta$. It therefore follows

from Theorem 5.1 that it is sufficient for Algorithm 2.1 to behave in a forward stable manner, if $s \approx q$ and ω is bounded from above, by say, a small power of n. Some upper bounds for ω were given in [10, Corollaries 7.1-7.3]:

I. If A is Hermitian and $b \neq 0$, then $\omega \leq \sqrt{2}n$ in the 2-norm.

II. If A is skew-Hermitian and $b \neq 0$, then $\omega = 1$ in the 2-norm.

III. If A is real and essentially nonnegative and $b = \pm |b| \neq 0$, then $\omega \leq n$ in the 1-norm and the ∞-norm. (Note that a real matrix $A = (a_{ij})$ is called *essentially nonnegative* if $a_{ij} \geq 0$ for $i \neq j$.)

Since $s = q$, for example, when A is normal and $\|\cdot\|$ is the 2-norm, Algorithm 2.1 behaves in a forward stable manner in the Cases I and II. An example of data A and b, for which $s = q$ and the conditions of Case III are satisfied, is given in Section 8.

For later purposes, we briefly describe how a lower bound on $\overline{\kappa}_{rel,1}$ can be computed for small matrices A, at least: Let $K(A, b)$ be the Kronecker form of $L(A, .)b$ (see e.g. [10, Eq. (8.2)]) and let $\|\cdot\| = \|\cdot\|_1$. We then have

$$\kappa \leq \overline{\kappa}_{rel,1} \leq n\kappa, \tag{5.10}$$

where κ is defined by the expression for $\overline{\kappa}_{rel,1}$ in (5.1), but with $\|K(A, b)\|_1$ in place of $\lambda(A, b)$. Further, an algorithm similar to that of [7, Algorithm 3.1] can be used to compute κ, provided that n is sufficiently small.

6. A posteriori error estimation

A method for estimating the roundoff error is presented, which, unlike the error estimates of the previous section, makes use of the computed result. It is assumed that v_s is an approximation of $e^A b$, which is defined by Algorithm 2.1 with an error tolerance δ, and $\widetilde{v}_s$ is the computed vector to v_s in floating point arithmetic of precision ϵ, where $0 < \epsilon \ll \delta$. For ϵ and δ such a relation is often satisfied in applications. In order that the expression for the roundoff error in (4.12) is used for error estimation, it is necessary to compute $\Delta r_j^{(k)}$ and $\Delta w_j^{(k)}$ for all j and k with sufficiently high accuracy. To realize this in practice, the algorithm is performed a second time and v_s is computed by simulating an arithmetic of precision ϵ_1, where $\epsilon \ll \epsilon_1 \leq \delta$. Let $\widetilde{v}_{s,1}$ be the computed vector to v_s for this case. Since the main part of the computation is still performed in floating point arithmetic of precision ϵ, the result of Theorem 4.1 applies in a modified form so that the roundoff error of $\widetilde{v}_{s,1}$ can approximately be determined. As a result, the roundoff error of $\widetilde{v}_s$ can be estimated.

To change between the different precisions, we use a rounding operator of the form

$$\mathrm{rd}(f) = 2^{\beta(\|f\|)}\mathrm{prec2}(\mathrm{prec1}(2^{-\beta(\|f\|)}f)) \tag{6.1}$$

for $f \in \mathbb{C}^n_{fl}$, where $\beta(\rho) := \lceil \log_2(\rho) \rceil$ if $\rho > 0$ and $\beta(\rho) := 0$ otherwise. In this context, prec1 denotes an operator, which converts a vector from precision ϵ to precision ϵ_1 (by rounding), whereas prec2 converts a vector from precision ϵ_1 to precision ϵ (by filling the additional digits of precision with zeros). To avoid any overflow or underflow in low precision arithmetic, prec1 is applied to normalized vectors only.

In case that v_s is computed by simulating an arithmetic of precision ϵ_1, a rounding of the form (6.1) is performed after certain parts of the computation. The procedure is as follows:

Let $k \leq s$ be a positive integer and let $\widetilde{v}_{0,1} = b$. Assume that v_{k-1} in (2.10) was computed and $\widetilde{v}_{k-1,1}$ is the computed vector to v_{k-1}, if $k \geq 2$. Further let $j \leq m$ be a positive integer and let $\widetilde{r}^{(k)}_{0,1} = \widetilde{w}^{(k)}_{0,1} = \widetilde{v}_{k-1,1}$. Assume that $r^{(k)}_{j-1}$ and $w^{(k)}_{j-1}$ in (2.11) were computed and $\widetilde{r}^{(k)}_{j-1,1}$ and $\widetilde{w}^{(k)}_{j-1,1}$ are the computed vectors to $r^{(k)}_{j-1}$ and $w^{(k)}_{j-1}$, respectively, if $j \geq 2$. Then $w^{(k)}_j$ is approximated by

$$\widetilde{w}^{(k)}_{j,1} = \mathrm{rd}(\widetilde{w}^{(k)}_{j,2}), \tag{6.2}$$

where $\widetilde{w}^{(k)}_{j,2}$ is the computed vector to $(sj)^{-1}\left((A-\mu I)\,\widetilde{w}^{(k)}_{j-1,1}\right)$ in an arithmetic of precision ϵ. Further, $r^{(k)}_j$ is approximated by

$$\widetilde{r}^{(k)}_{j,1} = \mathrm{rd}(\widetilde{r}^{(k)}_{j,2}), \tag{6.3}$$

where $\widetilde{r}^{(k)}_{j,2}$ is the computed vector to $\widetilde{r}^{(k)}_{j-1,1} + \widetilde{w}^{(k)}_{j,1}$ in an arithmetic of precision ϵ. If $\widetilde{w}^{(k)}_{j,1}$ and $\widetilde{r}^{(k)}_{j,1}$ were obtained for $j = 1, \ldots, m$, then $\widetilde{v}_{k,1} = e^{s^{-1}\mu}\widetilde{r}^{(k)}_{m,1}$ is used as an approximation of v_k. By doing this for $k = 1, \ldots, s$, we finally obtain an approximation $\widetilde{v}_{s,1}$ of v_s.

To analyse the error $\widetilde{v}_{s,1} - v_s$ of the computed vector, we define

$$\Delta r^{(k)}_{j,1} = \widetilde{r}^{(k)}_{j,1} - \left(\widetilde{r}^{(k)}_{j-1,1} + \widetilde{w}^{(k)}_{j,1}\right), \quad \Delta w^{(k)}_{j,1} = \widetilde{w}^{(k)}_{j,1} - \frac{1}{j}A_1\widetilde{w}^{(k)}_{j-1,1} \tag{6.4}$$

for $j = 1, \ldots, m$ and $k = 1, \ldots, s$, where $A_1 = s^{-1}(A - \mu I)$. Further, we express $\widetilde{v}_{s,1} - v_s$ in terms of $\Delta r^{(k)}_{j,1}$ and $\Delta w^{(k)}_{j,1}$ by using the operator Q in

(4.12), i.e.

$$\widetilde{v}_{s,1} - v_s = Q(\Delta_{s,1}), \quad \Delta_{s,1} = \left(\Delta r_{1,1}^{(1)}, \Delta w_{1,1}^{(1)}, \ldots, \Delta r_{m,1}^{(s)}, \Delta w_{m,1}^{(s)}\right). \tag{6.5}$$

The following result shows that this error is of the order $m\epsilon_1$ and can be determined by neglecting terms of the order $m\epsilon$:

Theorem 6.1. *Assume that $\epsilon > 0$, $\epsilon_1 \gg \epsilon$, and $\delta \geq \epsilon_1$ are sufficiently small. Let $\theta > 0$ and let m be the smallest positive integer, for which $\theta_m \geq \theta$. We then have*

$$\|\widetilde{v}_{s,1} - v_s\| = O(m\epsilon_1), \quad \|\widetilde{v}_{s,1} - v_s - \xi_{s,1}\| = O(m\epsilon), \tag{6.6}$$

with $m \leq \overline{m}(\epsilon_1)$ and

$$\xi_{s,1} = Q(\widetilde{\Delta}_{s,1}), \quad \widetilde{\Delta}_{s,1} = \left(\widetilde{\Delta} r_{1,1}^{(1)}, \widetilde{\Delta} w_{1,1}^{(1)}, \ldots, \widetilde{\Delta} r_{m,1}^{(s)}, \widetilde{\Delta} w_{m,1}^{(s)}\right), \tag{6.7}$$

where $\widetilde{\Delta} r_{j,1}^{(k)} = \widetilde{r}_{j,1}^{(k)} - \widetilde{r}_{j,2}^{(k)}$ and $\widetilde{\Delta} w_{j,1}^{(k)} = \widetilde{w}_{j,1}^{(k)} - \widetilde{w}_{j,2}^{(k)}$ for $j = 1, \ldots, m$ and $k = 1, \ldots, s$.

Proof. It is helpful to express $\widetilde{r}_{j,1}^{(k)}$ and $\widetilde{w}_{j,1}^{(k)}$ in terms of $\widetilde{r}_{j-1,1}^{(k)} + \widetilde{w}_{j,1}^{(k)}$ and $\widetilde{w}_{j-1,1}^{(k)}$, respectively: Because of (6.2) and (6.3) we have

$$\widetilde{r}_{j,1}^{(k)} = \left(I + M_j^{(k)}\right) \widetilde{r}_{j,2}^{(k)}, \quad \widetilde{w}_{j,1}^{(k)} = \left(I + N_j^{(k)}\right) \widetilde{w}_{j,2}^{(k)} \tag{6.8}$$

for some matrices $M_j^{(k)}, N_j^{(k)} \in \mathbb{C}^{n \times n}$, where

$$\left|M_j^{(k)}\right| \leq \epsilon_1 I, \quad \left|N_j^{(k)}\right| \leq \epsilon_1 I \tag{6.9}$$

for $j = 1, \ldots, m$ and $k = 1, \ldots, s$. In addition it follows from Eqs. (4.1) to (4.3) that

$$\widetilde{r}_{j,2}^{(k)} = \left(I + D_j^{(k)}\right) \left(\widetilde{r}_{j-1,1}^{(k)} + \widetilde{w}_{j,1}^{(k)}\right), \quad \widetilde{w}_{j,2}^{(k)} = \frac{1}{j} \left(A_1 + Z_j^{(k)}\right) \widetilde{w}_{j-1,1}^{(k)} \tag{6.10}$$

for some matrices $D_j^{(k)}, Z_j^{(k)} \in \mathbb{C}^{n \times n}$, where

$$\left|D_j^{(k)}\right| \leq \epsilon I, \quad \left|Z_j^{(k)}\right| \leq \chi_n(\epsilon)\epsilon \left|A_1\right| \tag{6.11}$$

for $j = 1, \ldots, m$ and $k = 1, \ldots, s$. We conclude from (6.4), (6.8), and (6.10) that

$$\Delta r_{j,1}^{(k)} = D_{j,1}^{(k)} \left(\widetilde{r}_{j-1,1}^{(k)} + \widetilde{w}_{j,1}^{(k)} \right), \quad \Delta w_{j,1}^{(k)} = \frac{1}{j} Z_{j,1}^{(k)} \widetilde{w}_{j-1,1}^{(k)}, \tag{6.12}$$

with

$$D_{j,1}^{(k)} = M_j^{(k)} \left(I + D_j^{(k)} \right) + D_j^{(k)}, \quad Z_{j,1}^{(k)} = N_j^{(k)} \left(A_1 + Z_j^{(k)} \right) + Z_j^{(k)}.$$

Because of (6.9) and (6.11) we further have

$$\left| D_{j,1}^{(k)} \right| \leq \overline{\epsilon} I, \quad \left| Z_{j,1}^{(k)} \right| \leq \chi_n(\overline{\epsilon}) \overline{\epsilon} \, |A_1| \tag{6.13}$$

for $j = 1, \ldots, m$ and $k = 1, \ldots, s$, where $\overline{\epsilon} := \epsilon_1 \, (1 + \epsilon) + \epsilon$.

Apart from the notation, the estimates in (4.3) and (6.13) are of the same type. According to the result of Lemma 4.1, we then have $\widetilde{r}_{m,1}^{(k)} - r_m^{(k)} = G_{m,1}^{(k)} \widetilde{v}_{k-1,1}$ for some matrix $G_{m,1}^{(k)} \in \mathbb{C}^{n \times n}$, where

$$\left| G_{m,1}^{(k)} \right| \leq \left(1 + \frac{1}{(n+1) \, m} \right) \psi_n(m\overline{\epsilon}) m \overline{\epsilon} T_m(|A_1|) \tag{6.14}$$

for $k = 1, \ldots, s$.

It follows from (6.14) that the result of Theorem 4.1 remains true when $\widetilde{v}_s$, $G_m^{(k)}$, and ϵ are replaced by $\widetilde{v}_{s,1}$, $G_{m,1}^{(k)}$, and $\overline{\epsilon}$, respectively. Consequently we have

$$\| \widetilde{v}_{s,1} - v_s \| = O(m\overline{\epsilon}) = O(m\epsilon_1).$$

Because of (6.5) this means that $\| Q(\Delta_{s,1}) \| = O(m\overline{\epsilon})$, where the columns of $\Delta_{s,1}$ are defined by the expressions for $\Delta r_{j,1}^{(k)}$ and $\Delta w_{j,1}^{(k)}$ in (6.12). Note that the estimate for $Q(\Delta_{s,1})$ is based on those for $\left| D_{j,1}^{(k)} \right|$ and $\left| Z_{j,1}^{(k)} \right|$ in (6.13) but is otherwise not affected by the numerical values for the entries of $D_{j,1}^{(k)}$ and $Z_{j,1}^{(k)}$.

In addition, we obtain from (6.4) and (6.10) that

$$\Delta r_{j,1}^{(k)} - \widetilde{\Delta} r_{j,1}^{(k)} = D_j^{(k)} \left(\widetilde{r}_{j-1,1}^{(k)} + \widetilde{w}_{j,1}^{(k)} \right), \quad \Delta w_{j,1}^{(k)} - \widetilde{\Delta} w_{j,1}^{(k)} = \frac{1}{j} Z_j^{(k)} \widetilde{w}_{j-1,1}^{(k)},$$

where the right-hand sides of these equations differ from those in (6.12) only in that $D_j^{(k)}$ and $Z_j^{(k)}$ are used in place of $D_{j,1}^{(k)}$ and $Z_{j,1}^{(k)}$, respectively. By

exploiting the linearity of Q, we then conclude from (6.11) that in analogy to the estimate for $Q(\Delta_{s,1})$ we have

$$\|\widetilde{v}_{s,1} - v_s - \xi_{s,1}\| = \left\|Q(\Delta_{s,1} - \widetilde{\Delta}_{s,1})\right\| = O(m\epsilon),$$

which completes the proof. □

Note that the substractions $\widetilde{r}_{j,1}^{(k)} - \widetilde{r}_{j,2}^{(k)}$ and $\widetilde{w}_{j,1}^{(k)} - \widetilde{w}_{j,2}^{(k)}$ can be performed on the basis of the operations (6.2) and (6.3) without rounding. Since $\epsilon \ll \epsilon_1$, it can further be assumed that for every j and k the real and imaginary parts of the components of $\widetilde{\Delta} r_{j,1}^{(k)} = \widetilde{r}_{j,1}^{(k)} - \widetilde{r}_{j,2}^{(k)}$ and $\widetilde{\Delta} w_{j,1}^{(k)} = \widetilde{w}_{j,1}^{(k)} - \widetilde{w}_{j,2}^{(k)}$ in (6.7) have sufficiently accurate representations in the floating point number system so that the norm of $\xi_{s,1}$ can accurately be determined to within a few significant digits, at least.

We are now in a position to estimate the roundoff error of $\widetilde{v}_s$ on the basis of the result for the error of $\widetilde{v}_{s,1}$. Because of (6.6), the normwise absolute roundoff error satisfies the estimate

$$\|\widetilde{v}_s - v_s\| \leq \|\widetilde{v}_s - \widetilde{v}_{s,1}\| + \|\xi_{s,1}\| + O(m\epsilon).$$

With regard to the normwise relative roundoff error, the result is as follows:

Corollary 6.1. *Under the assumptions of Theorem* 6.1 *we have*

$$\frac{\|\widetilde{v}_s - v_s\|}{\|v_s\|} \leq \frac{d}{1-d} + O(m\epsilon), \quad \textit{with } d := \frac{\|\widetilde{v}_s - \widetilde{v}_{s,1}\| + \|\xi_{s,1}\|}{\|\widetilde{v}_s\|}, \tag{6.15}$$

provided that $v_s \neq 0$, $\widetilde{v}_s \neq 0$, *and* $d < 1$.

Since an explicit expression is given for $\xi_{s,1}$ in (6.7), the result of Theorem 6.1 and Corollary 6.1 remain true for the case that b and, consequently, v_s, $\widetilde{v}_s$, $\widetilde{v}_{s,1}$, and $\xi_{s,1}$ are elements of $\mathbb{C}^{n\times n_0}$ for some $n_0 \leq n$. This is, for example, important in applications, where exponential integrator methods are used in combination with standard discretization techniques to solve partial differential equations (see [4, Section 4]).

The roundoff error must be at least of the order $m\epsilon_1$ in order that it is relevant to the error bound in (6.15): Assume that the normwise relative roundoff error is much less than $m\epsilon_1$. We then conclude from (6.6) that $\|\xi_{s,1}\|$ and $\|\widetilde{v}_s - \widetilde{v}_{s,1}\|$ and, hence, d is of the order $m\epsilon_1$, which means that the roundoff error may strongly be overestimated by using (6.15). In this case, however, the order of d is at most slightly larger than $O(\delta)$, which is the order of the *approximation* error to v_s. The roundoff error is therefore regarded as small, and the computed solution as sufficiently accurate, if $d = O(m\epsilon_1)$.

Note further that, *if* the normwise relative roundoff error is sufficiently small, then $d \approx 2 \|\widetilde{v}_s - \widetilde{v}_{s,1}\| \, \|\widetilde{v}_s\|^{-1}$.

In practice, the maximum precision is determined by the unit roundoff ϵ. When carrying out computations in MATLAB, we have $\epsilon = 2^{-53}$ (double precision) and it is convenient to use $\delta = 2^{-24}$ (single precision) and $\epsilon_1 = 2^{-\beta_1}$, where $\beta_1 \geq 24$ is a sufficiently small integer. The results of the numerical experiments in Section 8 indicate that, for this choice of δ, ϵ, and ϵ_1, the present method for estimating the roundoff error applies to a wide range of problems.

A brief remark is in order regarding a procedure, which is often used for the lack of anything better to estimate the error $\widetilde{v}_s - e^A b$ of the computed solution:

Another solution to $e^A b$ is computed by using Algorithm 2.1 with an error tolerance $\delta' < \delta$, e.g. $\delta' = \epsilon$, without changing the precision of the floating point arithmetic. Let us denote this solution by $\widetilde{v}'_s$ and assume that $\widetilde{v}'_s$ is sufficiently accurate so that $\left\|\widetilde{v}'_s - e^A b\right\|$ can be neglected against $\|\widetilde{v}_s - \widetilde{v}'_s\|$. It then follows from

$$\widetilde{v}_s - e^A b = (\widetilde{v}_s - \widetilde{v}'_s) + \left(\widetilde{v}'_s - e^A b\right)$$

that the normwise relative error of $\widetilde{v}_s$ satisfies the estimate

$$\frac{\left\|\widetilde{v}_s - e^A b\right\|}{\|e^A b\|} \leq \frac{h}{1-h}, \quad \text{with } h := \frac{\|\widetilde{v}_s - \widetilde{v}'_s\|}{\|\widetilde{v}'_s\|}, \tag{6.16}$$

provided that $b \neq 0$, $\widetilde{v}'_s \neq 0$, and $h < 1$. However, if the accuracy of $\widetilde{v}'_s$ is poor, for example due to the effects of a large condition number of the $e^A b$ problem, then this estimate may be incorrect, especially when h is small. In this case, the accuracy of $\widetilde{v}_s - \widetilde{v}'_s$ suffers from cancellations. An example is given in Section 8.

7. Computing the error bound

In this section we present an algorithm for computing $\widetilde{v}_{s,1}$ and $\xi_{s,1}$ to make an evaluation of the error bound in (6.15) possible. The cost for performing this algorithm and computing the error bound is estimated and compared with the cost for computing $e^A b$. It is further shown how the algorithm can be modified to take account of a premature terminating of the truncated Taylor series in (2.11).

Algorithm 7.1. Let $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$ and let μ be defined by (2.1). Further let s and m be the number of steps in (2.10) and (2.11), respectively. Let $\mathrm{rd}(\cdot)$ be a rounding operator of the form (6.1). (A corresponding MATLAB function is given in Appendix A.) This algorithm then computes an approximation $v = \widetilde{v}_{s,1}$ to $e^A b$ and an approximation $x = \xi_{s,1}$ to the roundoff error of $\widetilde{v}_{s,1}$. All computations are performed in floating point arithmetic of precision ϵ, except when a vector is converted from precision ϵ to precision ϵ_1 by using $\mathrm{rd}(\cdot)$. The algorithm is based on the following MATLAB commands (with A and $e1$ in place of $A - \mu I$ and $e^{s^{-1}\mu}$, respectively):

```
1    v = b;   w = v;
2    x = zeros(size(b));   y = x;
3    for  k = 1 : s
4      for  j = 1 : m
5        ww = (A * w) / (s * j);   w = rd(ww);
6        vv = v + w;   v = rd(vv);
7        y = (A * y) / (s * j) + (w - ww);
8        x = (x + y) + (v - vv);
9      end
10     v = e1 * v;   w = v;
11     x = e1 * x;   y = x;
12   end
```

The cost for performing this algorithm is dominated by the number of computed matrix-vector products and can be expressed in terms of this number, provided that ϵ and ϵ_1 are defined by standard precisions. The procedure is similar to that of [2, pp. 494-495], where the cost for computing an approximate solution $\widetilde{v}_s$ to $e^A b$ was estimated: If s is sufficiently large, then this cost is mainly given by the cost for performing Algorithm 2.1, which is $C_{\mathrm{sol}} = sm$ matrix-vector products (see [2, Eq. (3.10)]). Additional costs result from norm estimation to determine s (see [2, Eq. (3.12)]). Correspondingly, the cost for performing Algorithm 7.1 and estimating the roundoff error of $\widetilde{v}_s$ is $C_{\mathrm{est}} = 2C_{\mathrm{sol}}$ matrix-vector products.

It is worth mentioning that, since m is bounded from above by $m_{\max} = 55$ (see [2, p. 494]), both C_{sol} and C_{est} primarily grow with s, where this growth is linear. In contrast to that, the cost for computing the condition number of the $e^A b$ problem is, for example, a quadratic function in s (see [7]).

In practice, an early termination of Algorithm 2.1 is used, if possible, to reduce the computational cost. In the present notation, the criterion of [2, Eq. (3.15)] for a premature terminating of the finite sequence $r_j^{(k)} = T_j(A_1)v_{k-1}$, $j = 1, \ldots, m$, in (2.11) reads as follows: If j_k is the first positive integer such that $j_k = m$ or

$$\left\|w_{j_k-1}^{(k)}\right\| + \left\|w_{j_k}^{(k)}\right\| \leq \delta \left\|r_{j_k}^{(k)}\right\|, \tag{7.1}$$

then $r_{j_k}^{(k)}$ is accepted as an approximation of $e^{A_1}v_{k-1}$. Note that the left-hand side of (7.1) is meant to estimate the norm of the tail of the truncated series, $\sum_{i=j_k+1}^{m} \frac{1}{i!}A_1^i v_{k-1}$ (see [2, p. 496]).

As a result, we can replace m by j_k in line 4 of Algorithm 7.1, which means that an early termination of Algorithm 2.1 causes an early termination of Algorithm 7.1. Moreover, the ratio of the cost for performing Algorithm 7.1 to the cost for performing Algorithm 2.1 does not change and is equal to two, if each cost is expressed in terms of the number of computed matrix-vector products.

8. Numerical experiments

We give some numerical experiments to show how the present method for estimating the roundoff error can be used in practice. Both the quality of the estimates and the efficiency of the method are analysed.

The error tolerance, which is needed to define an approximation v_s to $e^A b$, is set to $\delta = 2^{-24}$. For a given A, this determines m, θ_m, and s. Vectors $\widetilde{v}_s$ and $\widetilde{v}_{s,1}$ are then computed to v_s by using an arithmetic of precision $\epsilon = 2^{-53}$ and simulating an arithmetic of precision $\epsilon_1 = 2^{-\beta_1}$, respectively, where $\beta_1 \geq 24$ is assumed to be a sufficiently small integer.

If $\beta_1 = 24$ (i.e. $\epsilon_1 = \delta$), then prec1 and prec2 in (6.1) are replaced by the MATLAB standard functions "single" and "double", respectively.

If $\beta_1 > 24$ (i.e. $\epsilon_1 < \delta$), then the rounding operation in (6.1) is realized as follows: For every floating point number ϕ, we define

$$\mathrm{rd}(\phi) := 2^{\beta(|\phi|)-\beta_1}\mathrm{round}(2^{-\beta(|\phi|)+\beta_1}\phi), \tag{8.1}$$

where "round" is a MATLAB standard function, which rounds a floating point number to the nearest integer. Further, for every $f \in \mathbb{C}_{fl}^n$, we obtain $\mathrm{rd}(f)$ by rounding the real and imaginary parts of the components of f

according to the definition in (8.1). This kind of rounding is not extended to the case $\beta_1 = 24$, since it generally needs more computing time than the use of "single" and "double" as proposed. An algorithm for computing $\mathrm{rd}(f)$ is given in Appendix A.

Error estimations are primarily performed with $\epsilon_1 = \delta$. The only purpose of choosing $\epsilon_1 < \delta$ is to analyse whether an estimate can be improved, if necessary. In this context, the bound d on the roundoff error in (6.15) is supplied with an index β_1, i.e. $d = d_{\beta_1}$, if $\epsilon_1 = 2^{-\beta_1}$. Further, the early termination of Algorithm 2.1 and Algorithm 7.1 is activated, as described in Section 7. The cost for computing $e^A b$ (i.e. for performing Algorithm 2.1 and for norm estimation to determine s) and that for evaluating d are expressed in terms of the number of computed matrix-vector products, c_{sol} and c_{est}, respectively. Note that c_{sol} corresponds to "mv" in [2, Section 6] and we have $c_{\mathrm{est}} \leq 2c_{\mathrm{sol}}$ according to the cost analysis in Section 7. In case that $\epsilon_1 < \delta$, the rounding procedure (8.1) may have a strong influence on the computing time, so that the cost for estimating the roundoff error is not adequately considered by c_{est}. A cost estimation, which is based on CPU times, is then preferred.

Vector and matrix norm are equal to the 1-norm, except that condition (7.1) on a premature terminating of the truncated Taylor series is formulated in the ∞-norm (see [2, Algorithm 3.2]). If $\beta_1 = 24$, then the ∞-norm is also used in (6.1). All computations were carried out in MATLAB 2015a on a machine with a 3.60 GHz Intel Core i7-4790 processor and 8 GB RAM.

The matrices in these experiments are of the form $A = \tau M$, where $M \in \mathbb{C}^{n\times n}$ and $\tau > 0$ are specified for each case. They are sparse, except those of Experiment 2, and they are nonnormal, except those of Experiment 4 and some of Experiment 6. In addition, the matrices are of Type (P2) in Experiments 1 and 2, they are of Type (P1) in Experiments 3 to 5, and matrices of both types are considered in Experiment 6.

Experiment 1. Two $e^{\tau M}b$ problems are treated in this experiment: The first one is orani678 (see [20, Section 6.2] for $\tau = 10$ and [2, Experiment 7] for $\tau = 100$), where $M \in \mathbb{R}^{2529\times 2529}$ is obtained from the University of Florida Sparse Matrix Collection [6]. The second one is stabB767 (see [19, Section 4.2] and [2, Experiment 8] for some $0 < \tau \leq 1$), where $M \in \mathbb{R}^{55\times 55}$ is taken from [18, Section 15]. In both cases b is the vector of ones.

The results in Table 8.1 show that $d_{24} < m\delta$, which means that d_{24} is as small as possible. Moreover, $e^{\tau M}b$ is computed very efficiently due to the scaling condition (2.4), as is indicated by small values of s but large values

of q/s. Therefore, c_{sol} as well as $c_{\text{est}} < 2c_{\text{sol}}$ can be regarded as small.

Table 8.1. *Experiment* 1

	τ	$m\delta$	d_{24}	c_{sol}	$c_{\text{est}}/c_{\text{sol}}$	s	q/s
orani678	10	2.6*e*-6	2.7*e*-7	240	0.5	3	3.4*e*2
	100	3.3*e*-6	1.3*e*-6	702	1.5	21	3.7*e*2
stabB767	1/10	3.3*e*-6	1.2*e*-6	475	1.3	25	4.8*e*3
	1	3.3*e*-6	2.7*e*-6	2314	1.8	246	4.9*e*3

Experiment 2. A problem of the form $e^{M_\sigma}b$ is considered, where $M_\sigma \in \mathbb{R}^{10\times 10}$ is the matrix from [17, Example 4]. This matrix is defined by analytical expressions for both the Jordan canonical form and the eigensystem of M_σ, where $\sigma > 0$ is a parameter which controls the ill-conditioning of the eigensystem as $\sigma \to 0$. Further, b is the vector of ones. To ensure that the entries of M_σ and the components of the exact solution $e^{M_\sigma}b$ were accurately determined to 16 decimal places, they were evaluated in MATLAB by using variable-precision arithmetic, before they were converted to double precision.

As is shown in Table 8.2, the normwise relative error e_{rel} of the computed vector $\widetilde{v}_s$ to $e^{M_\sigma}b$ strongly increases as σ decreases. In addition, the lower bound κ on the condition number in (5.10) increases as well. This means that the loss of accuracy in the computed solutions is caused by rounding errors, which are enhanced by large condition numbers.

We further notice that d_{24} overestimates e_{rel} by far. However, d_{β_1} comes close to e_{rel} for every σ, when β_1 is increased, for example, from 24 to 38: d_{38} is by a factor of only 2.1 to 4.1 greater than e_{rel}. In contrast to that, the error of $\widetilde{v}_s$ could not be estimated by using (6.16): h is by orders of magnitude smaller than e_{rel}, due to cancellations as explained in Section 6. Since M_σ is small and $s = 1$, the computational cost is low.

Table 8.2. *Experiment* 2

σ	e_{rel}	d_{24}	d_{31}	d_{38}	h	s	q/s	κ
0.16	1.6*e*-7	3.9*e*-3	8.1*e*-5	6.5*e*-7	3.2*e*-12	1	1.6*e*5	1.4*e*11
0.14	7.5*e*-7	1.1*e*-2	1.3*e*-6	3.1*e*-6	3.4*e*-12	1	5.2*e*5	1.5*e*12
0.12	1.9*e*-5	7.6*e*-3	5.5*e*-4	3.9*e*-5	2.0*e*-12	1	1.9*e*6	2.4*e*13
0.10	2.3*e*-4	2.6*e*-2	2.8*e*-3	8.8*e*-4	2.2*e*-11	1	7.6*e*6	6.4*e*14

Experiment 3. In this experiment, $M \in \mathbb{C}^{50\times 50}$ is the lower Hessenberg matrix from [18, Section 14], except that $n = 50$ is used in place of $n = 25$. Further, b is the vector of ones. The 1-norm of $e^{\tau M}b$ then has a maximum

value of about 6.7e4 at $\tau = 23$ and reaches a second relative maximum value of about 2.8e2 at $\tau = 67$. These so-called "humps" are followed by two peaks of κ of the order 10^8 at $\tau = 54$ and $\tau = 104$ (see Table 8.3).

Similar as in the previous experiment, d_{24} overestimates the roundoff error of the computed solutions by far. This becomes obvious when β_1 is increased, for example, from 24 to 38: d_{38} is by orders of magnitude smaller than d_{24} and, in addition, close to h (see Table 8.3). Despite that, d_{38} is at $\tau = 54$ by a factor of about 80 and at $\tau = 104$ by a factor of about 10 greater than the prescribed error tolerance δ. This gives rise to the supposition that the accuracy of the computed solutions is strongly affected by rounding errors because of large condition numbers.

Since s is small, both c_{sol} and c_{est} are small: c_{sol} varies between 134 and 619, and c_{est} between 268 and 854. In addition, the ratio of the CPU time for evaluating d_{β_1} to the CPU time for computing $e^{\tau M}b$ was analysed. Typical values for this ratio are of the same order as $c_{\text{est}}/c_{\text{sol}}$, if $\beta_1 = 24$, and by a factor of 3.8 to 4.7 greater than $c_{\text{est}}/c_{\text{sol}}$, if $\beta_1 = 38$, for example.

Table 8.3. *Experiment* 3

τ	d_{24}	d_{31}	d_{38}	h	s	q/s	κ
30	2.1e-4	5.3e-7	5.6e-9	2.2e-9	5	1.0e0	5.6e4
54	3.3e-2	1.6e-4	4.8e-6	1.3e-6	9	1.1e0	2.9e8
80	3.5e-4	8.0e-6	9.2e-8	3.2e-8	12	1.0e0	6.9e6
104	3.8e-3	1.1e-4	6.0e-7	4.0e-7	16	1.0e0	1.3e8

Experiment 4. This experiment deals with two $e^{\tau M}b$ problems: The first one is bcspwr10 (see [2, Experiment 7] for $\tau = 10$), where $M \in \mathbb{R}^{5300\times 5300}$ is a symmetric matrix, which is obtained from the University of Florida Sparse Matrix Collection [6]. Further, $b = (1, 0, \ldots, 0, 1)^\top$. The formulation of this problem goes back to Sidje [20, Section 6.3]. The second one is poisson (see [21, pp. 659-660] and [2, Experiment 7] for $\tau = 1$), where $M \in \mathbb{R}^{9801\times 9801}$ and b are generated in MATLAB by using the commands

$$
\begin{aligned}
&M = -2500 * \text{gallery}('\text{poisson}', 99);\\
&g = (-0.98 : 0.02 : 0.98)';\\
&[R1, R2] = \text{meshgrid}(g, g); \quad r1 = R1(:); \quad r2 = R2(:);\\
&b = (1 - r1.\hat{}2) . * (1 - r2.\hat{}2) . * \exp(r1);
\end{aligned}
$$

Note that, in the latter case, M is a symmetric block tridiagonal matrix.

It follows from the results of Section 5 that $e^{\tau M}b$ is computed in a forward stable manner for each of these problems. As is shown in Table 8.4, d_{24} is less than $m\delta$, except for problem poisson if $\tau = 1$, where $d_{24} \approx 10m\delta$. We conclude that d_{24} is as small as possible or can be regarded as small. Note further that both $c_{\rm sol}$ and $c_{\rm est}$ are approximately proportional to s. They are large for problem poisson if $\tau = 1$, since $s = q$ is large.

Table 8.4. *Experiment* 4

	τ	$m\delta$	d_{24}	$c_{\rm sol}$	$c_{\rm est}/c_{\rm sol}$	s	q/s
bcspwr10	10	3.2*e*-6	3.1*e*-7	215	1.6	5	2.0*e*0
	100	3.3*e*-6	2.6*e*-6	1766	2.0	48	2.0*e*0
poisson	1/10	3.3*e*-6	3.0*e*-6	2969	2.0	75	1.0*e*0
	1	3.3*e*-6	3.4*e*-5	29255	2.0	749	1.0*e*0

Experiment 5. Let $N \geq 2$ be an integer and let $\zeta > 0$. The $e^{\tau M}b$ problem, which is treated in this experiment, corresponds to that from Caliari et al. [5, Example 6]: The starting point is the advection-diffusion equation

$$\frac{\partial u}{\partial \tau} = \sum_{j=1}^{2} \left(\frac{\partial^2 u}{\partial \rho_j^2} + \zeta \frac{\partial u}{\partial \rho_j} \right),$$

which is discretized with respect to $\rho_1, \rho_2 \in [0,1]$, for $\tau > 0$. For that purpose, centered finite differences are used to approximate $\frac{\partial u}{\partial \rho_j}$ and $\frac{\partial^2 u}{\partial \rho_j^2}$ on a uniform grid of mesh size $\frac{1}{N+1}$ in $[0,1]^2$. In addition, homogeneous Dirichlet boundary conditions are considered.

Let $\mathrm{Pe} := \frac{\zeta}{2(N+1)}$ be the grid Péclet number as defined in [5]. Further let

$$J_\ell := \{N + \ell, 2N + \ell, \ldots, (N-1)\,N + \ell\}$$

for $\ell = 0, 1$. The discretization matrix is then given by $M = (N+1)^2 X$, where $X = (x_{ik})$ is an N^2-by-N^2 matrix having the nonzero elements $x_{ik} = -4$, if $i = k$, $x_{ik} = 1 + \mathrm{Pe}$, if $i = k - 1$ and $i \notin J_0$, $x_{ik} = 1 + \mathrm{Pe}$, if $i = k - N$, $x_{ik} = 1 - \mathrm{Pe}$, if $i = k + 1$ and $i \notin J_1$, and $x_{ik} = 1 - \mathrm{Pe}$, if $i = k + N$. By increasing Pe the nonnormality of M can be controlled (see [5]). Further, b is determined from the initial value of $u(\rho_1, \rho_2, \tau)$ at $\tau = 0$, which is defined by $u_0(\rho_1, \rho_2) = 256\rho_1^2 (1-\rho_1)^2 \rho_2^2 (1-\rho_2)^2$.

Note that M is essentially nonnegative if and only if $\mathrm{Pe} \leq 1$, which is true for all sufficiently large N. In addition, we have $b \geq 0$ because of

$u_0(\rho_1, \rho_2) \geq 0$. Therefore, Case III of Section 5 applies to the data $A = \tau M$ and b.

We used the algorithm by Al-Mohy and Higham (with $\delta = 2^{-24}$ and not with $\delta = 2^{-53}$ as in [5]) to compute $e^{\tau M} b$ for $\tau = \frac{1}{200}$ and several values of N and Pe. In all cases we obtained $s = q$, which means that, according to the result of Theorem 5.1, the computations were performed in a forward stable manner. Moreover, we used Algorithm 7.1 with $\epsilon_1 = \delta$ to estimate the roundoff error of the computed solution. The results are shown in Table 8.5: d_{24} slightly increases with N, but is always less than $m\delta \approx 3.2e\text{-}6$ and is thus as small as possible. The changes of d_{24} with Pe are small. Further, we obtained $c_{\text{est}} = 2c_{\text{sol}}$ for $N = 50$ and $c_{\text{est}} \approx 1.9c_{\text{sol}}$ for $N = 100$ and $N = 150$. Since s is small, both c_{sol} and c_{est} can be regarded as small.

Table 8.5. *Experiment* 5

	$N = 50$			$N = 100$			$N = 150$		
Pe	d_{24}	c_{sol}	s	d_{24}	c_{sol}	s	d_{24}	c_{sol}	s
1/5	2.5*e*-7	152	4	1.2*e*-6	652	16	1.7*e*-6	1374	35
3/5	2.7*e*-7	152	4	8.2*e*-7	648	16	1.7*e*-6	1368	35
1	2.3*e*-7	151	4	6.9*e*-7	630	16	1.2*e*-6	1282	35

Experiment 6. In this experiment it is analysed how the cost for estimating the roundoff error depends on whether the estimation procedure uses $\epsilon_1 = \delta$ or $\epsilon_1 < \delta$. Since the number of computed matrix-vector products does not change with $\epsilon_1 = 2^{-\beta_1}$, this cost is generally estimated on the basis of the elapsed CPU time. The ratio of the CPU time for evaluating d_{β_1} to the CPU time for computing $e^{\tau M} b$ is denoted by η_{β_1}. The algorithm, which is used to perform the rounding operation in (6.1), is given in Appendix A. We again consider the problems orani678 and bcspwr10 from Experiments 1 and 4, respectively.

Table 8.6 gives typical values for η_{24}, η_{31}, and η_{38}. We note that η_{24} is of the same order as $c_{\text{est}}/c_{\text{sol}}$, which confirms the assumption that the cost is dominated by the number of computed matrix-vector products, if all computations are performed with standard precisions. On the other hand, η_{31} is by a factor of 5.3 to 12.5 greater than $c_{\text{est}}/c_{\text{sol}}$, and so is η_{38}. This means that the rounding procedure (8.1) has a strong influence on the computing time in case that $\epsilon_1 < \delta$. It is therefore recommended to perform estimations of the roundoff error primarily with $\epsilon_1 = \delta$ in order to avoid high costs. In principle, however, error estimates can be improved by decreasing ϵ_1, as is

indicated by the values for d_{24}, d_{31}, and d_{38} in Table 8.6.

Table 8.6. *Experiment* 6

	τ	d_{24}	d_{31}	d_{38}	$c_{\text{est}}/c_{\text{sol}}$	η_{24}	η_{31}	η_{38}
orani678	10	2.7e-7	2.0e-9	2.2e-11	0.5	0.6	2.8	2.8
	100	1.3e-6	1.2e-8	1.1e-10	1.5	1.6	8.0	7.9
bcspwr10	10	3.1e-7	4.5e-9	3.7e-11	1.6	2.2	18.6	18.7
	100	2.6e-6	2.7e-8	2.6e-10	2.0	2.9	24.9	25.0

9. Conclusions

The algorithm by Al-Mohy and Higham for computing $e^A b$ is supplied with a method, which estimates the roundoff error of the computed solution a posteriori. It is assumed that $\epsilon \ll \epsilon_1 \leq \delta$, where δ is a given tolerance for the approximation error, ϵ is the unit roundoff, and ϵ_1 is a parameter, which can be used to control the accuracy of the roundoff error estimate. The computed solution is considered to be sufficiently accurate, if the estimated normwise relative roundoff error is of the order $m\epsilon_1$.

In practice, computations are, for example, performed with $\delta = 2^{-24}$ (single precision) and $\epsilon = 2^{-53}$ (double precision). It is further convenient to choose $\epsilon_1 = \delta$ by default. For this choice of δ, ϵ, and ϵ_1, the required accuracy was obtained in all experiments, where the scaling parameter s and the influence of the condition number on the computed solution were sufficiently small. In case that the computed error bound is much greater than $m\delta$, the roundoff error may be large (as was demonstrated in Experiments 2 and 3). The method can then be used with $\epsilon_1 < \delta$ to analyse whether the error estimate can be improved.

The cost for computing a solution and, if $\epsilon_1 = \delta$, the cost for estimating the roundoff error can be expressed in terms of the number of computed matrix-vector products, c_{sol} and c_{est}, respectively. The ratio $c_{\text{est}}/c_{\text{sol}}$ is always bounded from above by two. However, c_{sol} and c_{est} vary with the type of problem:

If the problem is of Type (P1), then c_{sol} and c_{est} may be large because of a large value for s.

If the problem is of Type (P2), then c_{sol} and c_{est} can be regarded as small, since s is relatively small.

Additional costs, which result from using the rounding procedure (8.1) in case that $\epsilon_1 < \delta$, are not considered by c_{est}. A cost estimation on the basis

of CPU times is then preferred. The experimental findings showed that the ratio of the CPU time for estimating the roundoff error to the CPU time for computing $e^A b$ can exceed $c_{\text{est}}/c_{\text{sol}}$ in this case by far.

Acknowledgements

The author is grateful to the referees for constructive remarks and for pointing out reference [5].

Appendix A. An algorithm for computing rd(f)

Algorithm A.1. Let $f \in \mathbb{C}_{fl}^n$ and let β_1 be an integer satisfying $24 \leq \beta_1 < 53$. This algorithm then computes $z = \text{rd}(f)$, where the rounding operation in (6.1) is realized for the case that $\epsilon = 2^{-53}$ and $\epsilon_1 = 2^{-\beta_1}$. The algorithm is defined by the following MATLAB function (with $b1$ in place of β_1):

```
function z = rd(f, b1)
if b1 == 24
  c = ceil(log2(norm(f, 1)));
  z = 2^c * double(single(2^(-c) * f));
else
  for p = 1 : 2
    if p == 1, g = real(f); else g = imag(f); end
    a = abs(g);
    if max(a) > 0
      r = a == 0; a(r) = 1;
      c = ceil(log2(a)) - b1;
      g = 2.^c. * round(2.^(-c) .* g);
    end
    if p == 1, z1 = g; else z2 = g; end
  end
  z = z1 + 1i * z2;
end
```

[1] A.H. Al-Mohy, N.J. Higham, A new scaling and squaring algorithm for the matrix exponential, SIAM J. Matrix Anal. Appl. 31 (2009) 970-989.

[2] A.H. Al-Mohy, N.J. Higham, Computing the action of the matrix exponential, with an application to exponential integrators, SIAM J. Sci. Comput. 33 (2011) 488-511.

[3] L. Bergamaschi, M. Vianello, Efficient computation of the exponential operator for large, sparse, symmetric matrices, Numer. Linear Algebra Appl. 7 (2000) 27-45.

[4] M. Caliari, P. Kandolf, A. Ostermann, S. Rainer, Comparison of software for computing the action of the matrix exponential, BIT 54 (2014) 113-128.

[5] M. Caliari, P. Kandolf, A. Ostermann, S. Rainer, The Leja method revisited: Backward error analysis for the matrix exponential, SIAM J. Sci. Comput. 38 (2016) A1639-A1661.

[6] T.A. Davis, Y. Hu, The University of Florida sparse matrix collection, ACM Trans. Math. Software 38 (2011) 1-25.

[7] E. Deadman, Estimating the condition number of $f(A)b$, Numer. Algorithms 70 (2015) 287-308.

[8] V. Druskin, L. Knizhnerman, Krylov subspace approximation of eigenpairs and matrix functions in exact and computer arithmetic, Numer. Linear Algebra Appl. 2 (1995) 205-217.

[9] V. Druskin, A. Greenbaum, L. Knizhnerman, Using nonorthogonal Lanczos vectors in the computation of matrix functions, SIAM J. Sci. Comput. 19 (1998) 38-54.

[10] T.M. Fischer, On the stability of some algorithms for computing the action of the matrix exponential, Linear Algebra Appl. 443 (2014) 1-20.

[11] E. Gallopoulos, Y. Saad, Efficient solution of parabolic equations by Krylov approximation methods, SIAM J. Sci. Stat. Comput. 13 (1992) 1236-1264.

[12] S. Güttel, Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection, GAMM-Mitteilungen 36 (2013) 8-31.

[13] N.J. Higham, Accuracy and Stability of Numerical Algorithms, second ed., SIAM, Philadelphia, 2002.

[14] N.J. Higham, A.H. Al-Mohy, Computing matrix functions, Acta Numerica 19 (2010) 159-208.

[15] M. Hochbruck, C. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 34 (1997) 1911-1925.

[16] M. Hochbruck, A. Ostermann, Exponential integrators, Acta Numerica 19 (2010) 209-286.

[17] C.S. Kenney, A.J. Laub, A Schur-Fréchet algorithm for computing the logarithm and exponential of a matrix, SIAM J. Matrix Anal. Appl. 19 (1998) 640-663.

[18] C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev. 45 (2003) 3-49.

[19] B.N. Sheehan, Y. Saad, R.B. Sidje, Computing $\exp(-\tau A)b$ with Laguerre polynomials, Electron. Trans. Numer. Anal. 37 (2010) 147-165.

[20] R.B. Sidje, Expokit: A software package for computing matrix exponentials, ACM Trans. Math. Software 24 (1998) 130-156.

[21] L.N. Trefethen, J.A.C. Weideman, T. Schmelzer, Talbot quadratures and rational approximations, BIT 46 (2006) 653-670.