Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®

Journal of Computational and Applied Mathematics 161 (2003) 203–216

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

Computing a matrix function for exponential integrators[☆]

Ya Yan Lu[*]

Department of Mathematics, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

Received 14 February 2003

Abstract

An efficient numerical method is developed for evaluating $\varphi(A)$, where A is a symmetric matrix and φ is the function defined by $\varphi(x)=(e^x-1)/x=1+x/2+x^2/6+\cdots$. This matrix function is useful in the so-called exponential integrators for differential equations. In particular, it is related to the exact solution of the ODE system $dy/dt=Ay+b$, where A and b are t-independent. Our method avoids the eigenvalue decomposition of the matrix A and it requires about $10n^3/3$ operations for a general symmetric $n\times n$ matrix. When the matrix is tridiagonal, the required number of operations is only $O(n^2)$ and it can be further reduced to $O(n)$ if only a column of the matrix function is needed. These efficient schemes for tridiagonal matrices are particularly useful when the Lanczos method is used to calculate the product of this matrix function (for a large symmetric matrix) with a given vector.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Matrix function; Exponential integrator; Chebyshev rational approximation; Lanczos method

1. Introduction

The function φ defined by

$$\varphi(x)=1+\frac{x}{2}+\frac{x^2}{3!}+\frac{x^3}{4!}+\cdots \tag{1}$$

has been found useful in the so-called exponential integrators [8] for solving differential equations. Consider the following initial value problem of a linear system of ODEs:

$$\frac{dy}{dt}=Ay+b \quad \text{for } t>0, \tag{2}$$

$$y(0)=y_0. \tag{3}$$

[☆] This research was partially supported by the CityU research Grant #7001229.
[*] Tel.: +85227887436; fax: +85227887446.
E-mail address: mayylu@math.cityu.edu.hk (Y.Y. Lu).

0377-0427/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2003.08.006

When the square matrix A and the column vector b are independent of t, the exact solution of (2) and (3) is well known:

$$y(t) = -A^{-1}b + \mathrm{e}^{tA}(y_0 + A^{-1}b) \tag{4}$$

$$= \mathrm{e}^{tA}y_0 + t\varphi(tA)b \tag{5}$$

$$= y_0 + t\varphi(tA)(b + Ay_0). \tag{6}$$

When the inhomogeneous term b is zero, the formulae are simply $y(t) = \mathrm{e}^{tA}y_0$. For a nonzero vector b, formula (6) is convenient to use and it is valid even when A is singular. Therefore, it is desirable to develop an efficient numerical method for evaluating the matrix function $\varphi(tA)$, or the product of $\varphi(tA)$ with a given vector.

For the more general case where A and b are t-dependent, a simple numerical scheme can be obtained if we replace A and b in each step by their values at the midpoint and use the exact solution of the approximate equation as the numerical solution. This gives rise to the following midpoint exponential method:

$$y_1 = y_0 + h\varphi(hA_{1/2})(b_{1/2} + A_{1/2}y_0), \tag{7}$$

where $h = t_1 - t_0$ is the step size ($t_0 = 0$ here), $A_{1/2} = A(t_{1/2})$ and $b_{1/2} = b(t_{1/2})$ for $t_{1/2} = t_0 + h/2$, and y_1 is the numerical solution at t_1, namely, $y_1 \approx y(t_1)$. If A and b are t-independent on the interval $[t_0, t_1]$, y_1 given by (7) reproduces the exact solution at t_1. However, for the general variable coefficient case, (7) is only a second-order method, same as, for example, the implicit midpoint method

$$\frac{y_1 - y_0}{h} = A_{1/2}\frac{y_0 + y_1}{2} + b_{1/2}. \tag{8}$$

Nevertheless, the midpoint exponential method (7) can still be useful. For example, for highly oscillatory problems, A and b often vary with t slowly, compared with the solution. In this case, we expect that (7) allows a larger step size h than (8). This fact has been analyzed in [6]. In fact, exponential integrators for the more general nonlinear equation $y' = f(y)$ have been developed and proved to be useful in [8]. A key operation in these methods is the multiplication of $\varphi(hA)$ with a given vector in each step, where A is the Jacobian matrix of f evaluated on the numerical solution at the beginning of the step and h is the step size.

Motivated by its application in exponential integrators, we develop an efficient numerical algorithm for calculating $\varphi(A)$ in this paper. The scalar h appeared in (7) or other exponential methods [8] has been dropped for simplicity. The matrix A is assumed to be real symmetric (or complex Hermitian). For a general $n \times n$ symmetric matrix A, our method requires $\frac{10}{3}n^3 + \mathrm{O}(n^2)$ floating point operations. For a symmetric tridiagonal matrix, the required number of operations is $\mathrm{O}(n^2)$. The standard procedure for computing a function of a symmetric matrix is to use the eigenvalue decomposition. This would require at least $5n^3$ operations, if Cuppen's divide and conquer method [2] is used, and more if the standard QR method is used instead. Furthermore, for a symmetric tridiagonal matrix, the method based on the eigenvalue decomposition still requires $\mathrm{O}(n^3)$ operations. Another approach for calculating $\varphi(A)$ is to combine a Padé approximation with a scaling and repeated doubling, similar to the well-known method for matrix exponential [12]. This method was developed in [8] based on $\varphi(2x) = (\mathrm{e}^x + 1)\varphi(x)/2$ and $\mathrm{e}^{2x} = \mathrm{e}^x\mathrm{e}^x$. However, the accuracy of this method depends on the norm of A. When A has a large norm, this method becomes expensive and not as accurate.

On the other hand, the method developed in this paper guarantees a relative accuracy (say, in matrix 2-norm) bounded by a small multiple of the machine epsilon.

No doubt, when A is very large, evaluating $\varphi(A)$ is impractical, since our method still requires $O(n^3)$ operations. In practice, what is really needed in (7) and other exponential integrators is $\varphi(hA)v$ for a given vector v. In this case, the Krylov subspace method [4,7,3] can be used. For a symmetric matrix A, the Lanczos method is used for a tridiagonal reduction, then $\varphi(hA)v$ is approximated by

$$\varphi(hA)v \approx \|v\| V_m \varphi(hT_m) e_1^{(m)}, \tag{9}$$

where V_m is the matrix of the first m Lanczos vectors with $v/\|v\|$ as its first column, T_m is an $m \times m$ tridiagonal matrix, $e_1^{(m)}$ is the first column of the $m \times m$ identity matrix. A convergence criterion is needed to determine the value of m that gives a sufficiently accurate approximation. This can be based on the $(m, 1)$ entry of $\varphi(hT_m)$ [8]. However, if we use the eigenvalue decomposition to calculate the $(m, 1)$ entry of $\varphi(hT_m)$ for $m = 1, 2, \dots, m_*$ (assuming a satisfactory approximation is obtained for m_*), the total number of operations needed would be $O(m_*^4)$, since the eigenvalue decomposition of T_m requires $O(m^3)$ operations. This is a significant computation effort that can slow down the overall Lanczos algorithm. In this paper, we demonstrate that the $(m, 1)$ entry of $\varphi(hT_m)$ can be calculated in $O(m)$ operations. This gives rise to the total of $O(m_*^2)$ operations for the convergence test.

In Section 2, the details of our method are presented. An analysis for the error for computing $\varphi(A)$ by our method is provided in Section 3. Numerical examples are given in Section 4. The paper is completed with some remarks in Section 5.

2. The algorithm

For a symmetric matrix A, our method starts with the tridiagonal reduction $A = QTQ^{\mathrm{T}}$, where T is symmetric tridiagonal and Q is orthogonal. This is the standard first step for computing the eigenvalue decomposition of a symmetric matrix. For a number of matrix functions, efficient numerical methods [9–11] have been developed based on $f(A) = Qf(T)Q^{\mathrm{T}}$ and rational approximations to $f(T)$. This is in contrast to the standard approach where the eigenvalue decomposition of A (based on that of T) is computed first, then evaluate $f(A)$ by $f(A) = Vf(\Lambda)V^{\mathrm{T}}$, where Λ is the diagonal matrix of the eigenvalues and V is the orthogonal matrix of the eigenvectors.

For the square root and the logarithm of a symmetric positive definite matrix, the methods developed in [9,10] are based on Padé approximants for a scaled tridiagonal matrix $(1/\mu)T$. The scaling parameter μ is chosen to minimize the effort to approximate $\sqrt{T}$ or $\log(T)$ within a given error tolerance. For the exponential of a symmetric matrix, the method developed in [11] finds e^T in $O(n^2)$ operations, leading to the total of $\frac{10}{3}n^3 + O(n^2)$ operations for computing e^A. For the given tridiagonal matrix T, this method first calculates the largest eigenvalues λ_1 in $O(n)$ operations (for example, by the bisection method), then approximates e^T by a Chebyshev rational approximation of $e^{T-\lambda_1 I}$, since $e^T = e^{\lambda_1} e^{T-\lambda_1 I}$. More precisely, let R_p be the rational function that gives the smallest maximum error for approximating e^x on $(-\infty, 0]$, among all rational functions with the degrees of the denominator and the numerator less than or equal to p, we approximate $e^{T-\lambda_1 I}$ by $R_p(T - \lambda_1 I)$. For the standard double precision (64-bit floating point numbers), it is sufficient to take $p = 14$. The function R_p is given as a sum of prime fractions

$$R_p(x) = \alpha_0^{(p)} + \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{x - \theta_j^{(p)}}. \tag{10}$$

The matrix $R_p(T - \lambda_1 I)$ is then efficiently evaluated through

$$R_p(T - \lambda_1 I) = \alpha_0^{(p)} I + \sum_{j=1}^{p} \alpha_j^{(p)} [T - (\lambda_1 + \theta_j^{(p)}) I]^{-1}. \tag{11}$$

Similarly, for the function φ considered in this paper, the Chebyshev rational approximation on $(-\infty, 0]$ is useful. Let $\tilde{R}_q$ be the rational function of degree q (for polynomials in the denominator and the numerator) that minimizes the maximum error on $(-\infty, 0]$, we write $\tilde{R}_q$ in its prime fraction expansion

$$\tilde{R}_q(x) = \tilde{\alpha}_0^{(q)} + \sum_{j=1}^{q} \frac{\tilde{\alpha}_j^{(q)}}{x - \tilde{\theta}_j^{(q)}}. \tag{12}$$

For a negative semi-definite matrix T, we can use the following approximation:

$$\varphi(T) \approx S_q^{(1)}(T) = \tilde{\alpha}_0^{(q)} I + \sum_{j=1}^{q} \tilde{\alpha}_j^{(q)} [T - \tilde{\theta}_j^{(q)} I]^{-1}. \tag{13}$$

Let $\tilde{\varepsilon}_q = \max_{x \leqslant 0} |\varphi(x) - \tilde{R}_q(x)|$, we immediately have a bound on the absolute error in the matrix 2-norm

$$\|\varphi(T) - S_q^{(1)}\|_2 \leqslant \tilde{\varepsilon}_q.$$

However, if the largest eigenvalue λ_1 is far away from zero (and negative), we could have poor relative accuracy, since $\|\varphi(T)\|_2 = \varphi(\lambda_1)$ could be quite small. For this purpose, we propose to use (13) only when $-1 \leqslant \lambda_1 \leqslant 0$. If $\lambda_1 < -1$, we derive an approximation of $\varphi(T)$ from $\varphi(T) = T^{-1}(e^T - I)$ and the Chebyshev rational approximation of e^T. More precisely, from the approximation of e^T by $e^{\lambda_1} R_p(T - \lambda_1 I)$ in (11), we have

$$\varphi(T) \approx S_p^{(2)}(T) = T^{-1} \left[(e^{\lambda_1} \alpha_0^{(p)} - 1) I + e^{\lambda_1} \sum_{j=1}^{p} \alpha_j^{(p)} [T - (\lambda_1 + \theta_j^{(p)}) I]^{-1} \right]. \tag{14}$$

The above two approximations ($S_q^{(1)}$ and $S_p^{(2)}$) to $\varphi(T)$ should not be used if the largest eigenvalue is positive. Unlike the exponential function, we can not simply shift the matrix by $-\lambda_1 I$ and calculate $\varphi(T - \lambda_1 I)$, since $\varphi(T)$ does not have a simple relationship with $\varphi(T - \lambda_1 I)$. In principle, once the largest eigenvalue of T is calculated, we could proceed to calculate a Chebyshev rational approximation to $\varphi(x)$ on $(-\infty, \lambda_1]$ and use it to approximate $\varphi(T)$. This is not practical, because the Chebyshev approximation is not easy to calculate. Fortunately, a different approximation to $\varphi(x)$ can be derived from the Chebyshev rational approximation of e^x. Let

$$B = \begin{pmatrix} x & 1 \\ 0 & 0 \end{pmatrix},$$

it is easy to see that

$$e^B = \begin{pmatrix} e^x & \varphi(x) \\ 0 & 1 \end{pmatrix}.$$

Therefore,

$$e^B = e^{\lambda_1} e^{B-\lambda_1 I} \approx e^{\lambda_1} \left\{ \alpha_0^{(p)} I + \sum_{j=1}^{p} \alpha_j^{(p)} [B - (\lambda_1 + \theta_j^{(p)}) I]^{-1} \right\}.$$

The (1,2) entry of the matrix $[B - (\lambda_1 + \theta_j^{(p)})I]^{-1}$ is $(x - \lambda_1 - \theta_j^{(p)})^{-1}(\lambda_1 + \theta_j^{(p)})^{-1}$, therefore,

$$\varphi(x) \approx e^{\lambda_1} \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{\lambda_1 + \theta_j^{(p)}} \frac{1}{x - \lambda_1 - \theta_j^{(p)}}.$$

This leads to the following approximation:

$$\varphi(T) \approx S_p^{(3)}(T) = e^{\lambda_1} \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{\lambda_1 + \theta_j^{(p)}} [T - (\lambda_1 + \theta_j^{(p)}) I]^{-1}. \tag{15}$$

Therefore, we first calculate the largest eigenvalue λ_1 in $O(n)$ operations and then approximate $\varphi(T)$ by (14), (13) or (15) for $\lambda_1 < -1$, $-1 \leqslant \lambda_1 \leqslant 0$ or $\lambda_1 > 0$, respectively. In our numerical implementation for double precision (64-bit) floating point arithmetics, we have chosen $p = 14$ for (14), $q = 14$ for (13) and $p = 16$ for (15). The corresponding maximum errors are

$$\varepsilon_{14} \approx 1.83 \times 10^{-14}, \qquad \varepsilon_{16} \approx 2.12 \times 10^{-16}, \qquad \tilde{\varepsilon}_{14} \approx 6.89 \times 10^{-16}.$$

The rational approximation coefficients (that is, $\{\alpha_j^{(p)}, \theta_j^{(p)}\}$ for $p = 14$, 16 and $\{\tilde{\alpha}_j^{(q)}, \tilde{\theta}_j^{(q)}\}$ for $q = 14$) are calculated in [5] and listed in the appendix. Our method is summarized in the following algorithm.

Algorithm 1. *Let A be a given symmetric matrix, the matrix function $\varphi(A)$ is calculated in the following steps using the coefficients $\alpha_j^{(p)}$, $\theta_j^{(p)}$, $\tilde{\alpha}_j^{(q)}$ and $\tilde{\theta}_j^{(q)}$ given in the appendix.*

1. *Calculate a symmetric tridiagonal matrix T and an orthogonal matrix Q, such that $A = QTQ^{\mathrm{T}}$;*
2. *Calculate the largest eigenvalue of T, say λ_1;*
3. *If $\lambda_1 < 0$, choose $p = 14$ and evaluate $\varphi(T)$ by*

$$\varphi(T) \approx T^{-1} \left[(e^{\lambda_1} \alpha_0^{(p)} - 1) I + e^{\lambda_1} \sum_{j=1}^{p} \alpha_j^{(p)} [T - (\lambda_1 + \theta_j^{(p)}) I]^{-1} \right],$$

if $-1 \leqslant \lambda_1 \leqslant 0$, choose $q = 14$ and evaluate $\varphi(T)$ by

$$\varphi(T) \approx \tilde{\alpha}_0^{(q)} I + \sum_{j=1}^{q} \tilde{\alpha}_j^{(q)} [T - \tilde{\theta}_j^{(q)} I]^{-1},$$

if $\lambda_1 > 0$, *choose* $p = 16$ *and evaluate* $\varphi(T)$ *by*

$$\varphi(T) \approx \mathrm{e}^{\lambda_1} \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{\lambda_1 + \theta_j^{(p)}} [T - (\lambda_1 + \theta_j^{(p)})I]^{-1};$$

4. *Calculate* $\varphi(A)$ *by* $\varphi(A) = Q\varphi(T)Q^{\mathrm{T}}$.

We notice that the first step is a standard procedure for computing eigenvalues of a symmetric matrix [1]. For a general dense $n \times n$ symmetric matrix, this tridiagonal reduction step requires about $\frac{4}{3}n^3$ operations with Q given implicitly as a product of Householder reflections. In the second step, the largest eigenvalue of T can be calculated in $\mathrm{O}(n)$ operations by, for example, the bisection method [1]. The required number of operations for the third step is clearly $\mathrm{O}(n^2)$. The last step requires about $2n^3$ operations (using the implicitly given matrix Q), the details can be found in [9]. Therefore, the total required number of operations for computing $\varphi(A)$ is $10n^3/3 + \mathrm{O}(n^2)$. If A is a symmetric tridiagonal matrix itself, then the first and last steps are not needed and the algorithm requires only $\mathrm{O}(n^2)$ operations.

If only the matrix vector product $\varphi(T)b$ is needed for a given vector b, the number of operations can be reduced to $\mathrm{O}(n)$, because the inverse matrices in the third step are reduced to solutions of linear systems with tridiagonal coefficient matrices. This is useful in the Lanczos method that finds an approximation for $\varphi(hA)v$, where A is a large symmetric (possibly sparse) matrix, h is a scalar and v is a given vector. As given in (9), the first column of $\varphi(hT_m)$ is required, where T_m is a $m \times m$ symmetric tridiagonal matrix. Since $m \ll n$, this step is usually negligible in the whole Lanczos process. However, for the convergence test in the Lanczos method, it is necessary to calculate the $(m, 1)$ entry of $\varphi(hT_m)$ for $m = 1, 2, \ldots, m_*$. With our algorithm, this can be done in $\mathrm{O}(m_*^2)$ operations. If the standard eigenvalue decomposition approach is used for this purpose, $\mathrm{O}(m_*^4)$ operations are required. Although m_* is certainly small compared with n, the $\mathrm{O}(m_*^4)$ operations can not be ignored. If A is a typical sparse matrix, the matrix vector multiplication involving A may only require $\mathrm{O}(n)$ operations, leading to the total of $\mathrm{O}(nm_*)$ operations for the Lanczos method (with m_* steps). Therefore, the $\mathrm{O}(m_*^4)$ convergence test can be a significant factor in the overall performance of the Lanczos method.

3. Error bounds

The Chebyshev rational approximations of e^x and $\varphi(x)$ on the interval $(-\infty, 0]$ are characterized by their respective maximum errors

$$\varepsilon_p = \max_{x \leqslant 0} |\mathrm{e}^x - R_p(x)|, \qquad \tilde{\varepsilon}_q = \max_{x \leqslant 0} |\varphi(x) - \tilde{R}_q(x)| \tag{16}$$

for R_p and $\tilde{R}_q$ defined in (10) and (12), respectively. For a given symmetric tridiagonal matrix T, if its largest eigenvalue λ_1 is negative or zero, we show that our approximate formulas always give a small relative accuracy that is bounded by a small multiple of ε_p or $\tilde{\varepsilon}_q$. If $\lambda_1 < -1$, we have

$$\varphi(x) \approx S_p^{(2)}(x) = \frac{1}{x}\left[\mathrm{e}^{\lambda_1}\alpha_0^{(p)} - 1 + \sum_{j=1}^{p} \frac{\mathrm{e}^{\lambda_1}\alpha_j^{(p)}}{x - \lambda_1 - \theta_j^{(p)}}\right].$$

For $x \leqslant \lambda_1$, we have

$$|\varphi(x) - S_p^{(2)}(x)| = \left|\frac{1}{x}(\mathrm{e}^{\lambda_1}\mathrm{e}^{x-\lambda_1} - 1) - S_p^{(2)}(x)\right| = \frac{\mathrm{e}^{\lambda_1}}{|x|}\,|\mathrm{e}^{x-\lambda_1} - R_p(x-\lambda_1)| \leqslant \frac{\mathrm{e}^{\lambda_1}\varepsilon_p}{|x|}.$$

Therefore,

$$|\varphi(x) - S_p^{(2)}(x)| \leqslant \frac{\mathrm{e}^{\lambda_1}}{1-\mathrm{e}^{\lambda_1}}\,\varepsilon_p|\varphi(\lambda_1)| < \frac{1}{\mathrm{e}-1}\varepsilon_p|\varphi(\lambda_1)|.$$

For the matrix 2-norm, we have $\|\varphi(T)\|_2 = \varphi(\lambda_1)$ and

$$\|\varphi(T) - S_p^{(2)}(T)\|_2 = \max_{1\leqslant k\leqslant n}|\varphi(\lambda_k) - S_p^{(2)}(\lambda_k)| < \frac{1}{\mathrm{e}-1}\,\varepsilon_p\|\varphi(T)\|_2,$$

where λ_k for $k = 1,2,\ldots,n$ are the eigenvalues of T.

If $-1 \leqslant \lambda \leqslant 0$, we approximate $\varphi(T)$ directly by its Chebyshev rational approximation $\tilde{R}_q(T)$. For $x \leqslant \lambda_1$, we have

$$|\varphi(x) - \tilde{R}_q(x)| \leqslant \tilde{\varepsilon}_q \leqslant \frac{1}{\varphi(-1)}\,\tilde{\varepsilon}_q\varphi(\lambda_1) = \frac{\mathrm{e}}{\mathrm{e}-1}\,\tilde{\varepsilon}_q\varphi(\lambda_1).$$

This leads to

$$\|\varphi(T) - \tilde{R}_q(T)\|_2 \leqslant \frac{\mathrm{e}}{\mathrm{e}-1}\,\tilde{\varepsilon}_q\|\varphi(T)\|_2.$$

When the largest eigenvalue λ_1 is positive, our approximation to $\varphi(T)$ is based on

$$\varphi(x) \approx S_p^3(x) = \mathrm{e}^{\lambda_1}\sum_{j=1}^{p}\frac{\alpha_j^{(p)}}{\lambda_1 + \theta_j^{(p)}}\,\frac{1}{x - \lambda_1 - \theta_j^{(p)}} \quad \text{for } x \leqslant \lambda_1.$$

The error of this approximation is directly related to the error of the Chebyshev rational approximation of e^x:

$$E_p(x) = \mathrm{e}^x - R_p(x)$$

for $R_p(x)$ given in (10). We first establish that if $x \neq 0$,

$$\varphi(x) - S_p^{(3)}(x) = \mathrm{e}^{\lambda_1}\,\frac{E_p(x-\lambda_1) - E_p(-\lambda_1)}{x} \tag{17}$$

and

$$\varphi(0) - S_p^{(3)}(0) = \mathrm{e}^{\lambda_1}E_p'(-\lambda_1). \tag{18}$$

In fact, if $x \neq 0$, we have

$$\varphi(x) = \frac{\mathrm{e}^x - 1}{x} = \frac{\mathrm{e}^{\lambda_1}}{x}(\mathrm{e}^{x-\lambda_1} - \mathrm{e}^{-\lambda_1}) = \frac{\mathrm{e}^{\lambda_1}}{x}[R_p(x-\lambda_1) + E_p(x-\lambda_1) - R_p(-\lambda_1) - E_p(-\lambda_1)].$$

It is easy to see that

$$\frac{\mathrm{e}^{\lambda_1}}{x}[R_p(x-\lambda_1) - R_p(-\lambda_1)] = \frac{\mathrm{e}^{\lambda_1}}{x}\sum_{j=1}^{p}\alpha_j^{(p)}\left(\frac{1}{x-\lambda_1-\theta_j^{(p)}} + \frac{1}{\lambda_1+\theta_j^{(p)}}\right) = S_p^{(3)}.$$

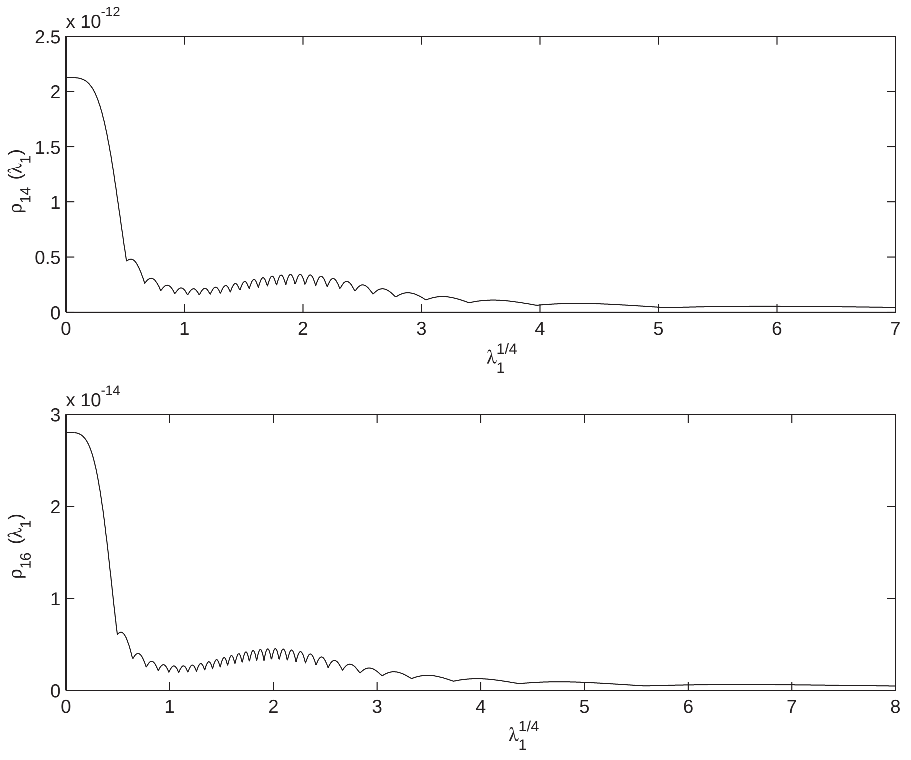

Fig. 1. The function $\rho_p(\lambda_1)$ for $p = 14$ and 16. The horizontal axis is $\lambda_1^{1/4}$.

This immediately leads to (17). For $x = 0$, result (18) can be directly verified. If we introduce the following function:

$$\rho_p(\lambda_1) = \frac{e^{\lambda_1}}{\varphi(\lambda_1)} \max_{x \leqslant \lambda_1} \left| \frac{E_p(x-\lambda_1) - E_p(-\lambda_1)}{x} \right|$$

(with $E_p'(-\lambda_1)$ replacing the divided difference above when $x = 0$), we have

$$|\varphi(x) - S_p^{(3)}(x)| \leqslant \rho_p(\lambda_1)\varphi(\lambda_1) \quad \text{for } x \leqslant \lambda_1.$$

This leads to

$$\|\varphi(T) - S_p^{(3)}(T)\|_2 \leqslant \rho_p(\lambda_1)\|\varphi(T)\|.$$

In Fig. 1, the function ρ_p for $p = 14$ and 16 is shown. It seems that at least for the two values of p in Fig. 1, $\rho_p(\lambda_1)$ (for $\lambda_1 \geqslant 0$) is bounded by $\rho_p(0)$. Notice that the horizontal axis is actually $\lambda_1^{1/4}$. If $\rho_p(\lambda_1)$ is plotted against λ_1 directly, we will only see a sharp decrease near $\lambda_1 = 0$. In order to evaluate $\rho_p(\lambda_1)$, it is necessary to find the maximum of the function $|x^{-1}(E_p(x-\lambda_1) - E_p(-\lambda_1))|$ for all x in the semi-infinite interval $(-\infty, \lambda_1]$. Fortunately, the interval for x can be reduced. In fact, from the theory of Chebyshev rational approximation, we know that the function $E_p(x)$ has $2p$ critical points on the negative real axis. Let these critical points be $x_1, x_2, \ldots, x_{2p}$, satisfying

$$0 = x_0 > x_1 > x_2 > \cdots > x_{2p} > x_{2p+1} = \infty,$$

we have

$$E_{x_j} = (-1)^j \varepsilon_p \quad \text{for } j = 0, 1, 2, \ldots, 2p+1.$$

Now, if $-\lambda_1$ is close to x_m and $E_p(-\lambda_1)$ has the same sign as $E_p(-\lambda_1)$, then

$$\max_{x \leqslant \lambda_1} |x^{-1}(E_p(x-\lambda_1) - E_p(-\lambda_1))| = \max_{x_{m+1} \leqslant x \leqslant x_{m-1}} |x^{-1}(E_p(x-\lambda_1) - E_p(-\lambda_1))|. \tag{19}$$

Of course, the above is only for the most general case. If $m=0$, or $2p+1$, we should replace x_{m-1} by x_0, or x_{m+1} by x_{2p+1}, respectively. Furthermore, if $E_p(-\lambda_1)=0$ and $-\lambda_1 \in (x_{m+1}, x_m)$, we only need to search the maximum in $[x_{m+1}, x_m]$. To understand (19), we notice that $(E_p(x-\lambda_1) - E_p(-\lambda_1))/x$ is the slope of the line segment connecting the two points on the E_p function at $-\lambda_1$ and $x - \lambda_1$. If x is outside the interval $[x_{m+1}, x_{m-1}]$, it simply can not give a larger slope (in absolute value). Of course, the maximum on the interval must equal to $E_p'(\xi)$ for some $\xi \in [x_{m+1}, x_{m-1}]$. On the other hand, the function $\rho_p(\lambda_1)$ has the additional factor of $e^{\lambda_1}/\varphi(\lambda_1)$ which is asymptotic to λ_1 for large λ_1. However, this increasing factor in ρ_p is balanced by the general decreasing trend of the maximum in (19) (as λ_1 increases). The gaps between x_{m+1} and x_m increases rapidly if m increases and the corresponding derivative $E_p'(\xi)$ (ξ depends on λ_1) generally decreases as λ_1 increases. This explains that the function $\rho_p(\lambda_1)$ is still uniformly bounded for all $\lambda_1 \geqslant 0$.

From the results in this section, we conclude that we can reach a relative accuracy around the machine epsilon (for standard double precision) for our matrix approximation of $\varphi(T)$. However, we obtain the same bounds for the corresponding approximation of $\varphi(A)$ in the matrix 2-norm. For example, if $\lambda_1 > 0$, we have

$$\|\varphi(A) - QS_p^{(3)}(T)Q^{\mathrm{T}}\|_2 = \|\varphi(T) - S_p^{(3)}(T)\|_2 \leqslant \rho_p(\lambda_1)\|\varphi(T)\|_2 = \rho_p(\lambda_1)\|\varphi(A)\|_2.$$

4. Numerical examples

We have implemented our algorithm in double precision based on $p=14$ in (13), $q=14$ in (13) and $p=16$ in (15). To access the accuracy of our method, we compare our numerical solutions with the solutions obtained from a quadruple precision routine based on the eigenvalue decomposition. The quadruple precision program for $\varphi(A)$ relies on a simple modification of the EISPACK routines `TRED2`, `IMTQL2` and `PYTHAG` for the symmetric matrix eigenvalue problem. The accuracy of the modified programs is independently verified by a multi-precision calculation in MAPLE. With the quadruple precision solution as the “exact solution”, we have calculated the following relative errors in various matrix norms:

$$e_f = \frac{\|\varphi(A) - \varphi_*(A)\|_f}{\|\varphi(A)\|_f}, \qquad e_1 = \frac{\|\varphi(A) - \varphi_*(A)\|_1}{\|\varphi(A)\|_1}, \qquad e_2 = \frac{\|\varphi(A) - \varphi_*(A)\|_2}{\|\varphi(A)\|_2},$$

where $\varphi_*(A)$ is the numerical solution based on our new method. We point out that the numerical solutions from a double precision eigenvalue decomposition routine cannot be used as “exact solutions” for calculating the errors of our solutions, since the errors are on the same order of magnitude. On the other hand, this double precision eigenvalue decomposition program is used for comparison of execution time, to verify the theoretical claim that our method is more efficient.

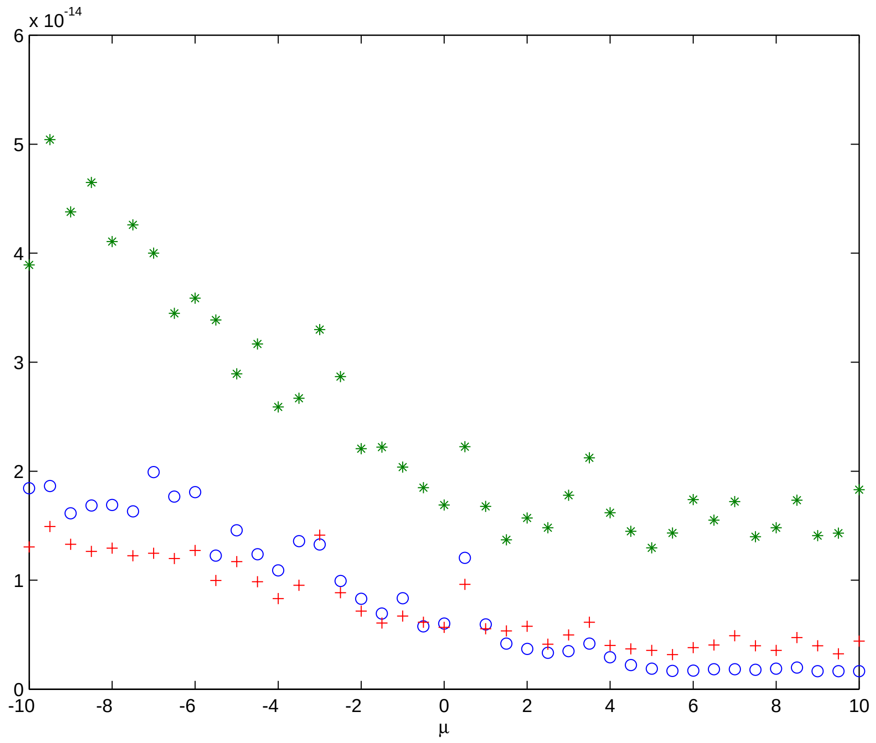

Fig. 2. Relative errors for computing $\varphi(A) = \varphi(A_0 - \mu I)$ in matrix 2-norm (+), Frobenius norm (o) and matrix 1-norm (∗). A_0 is the 100×100 matrix given in (20).

Example 1. The (i, j) entry of the $n \times n$ matrix A_0 is

$$a_{ij} = \frac{1}{2 + (i - j)^2}. \tag{20}$$

Since different approximations are used when the largest eigenvalue is in different intervals, we let $A = A_0 - \mu I$ and calculate the relative errors for $n = 100$ and various values of μ. The results are shown in Fig. 2.

Example 2. The $n \times n$ matrix A_0 is obtained from a second-order finite difference discretization of the negative Laplacian on an unit square with Dirichlet boundary conditions. The (i, j) entry of A_0 is given by

$$a_{ij} = \begin{cases} 4 & \text{if } i = j, \\ -1 & \text{if } |i - j| = p, \\ -1 & \text{if } |i - j| = 1 \text{ and } (i + j) \bmod (2p) \neq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{21}$$

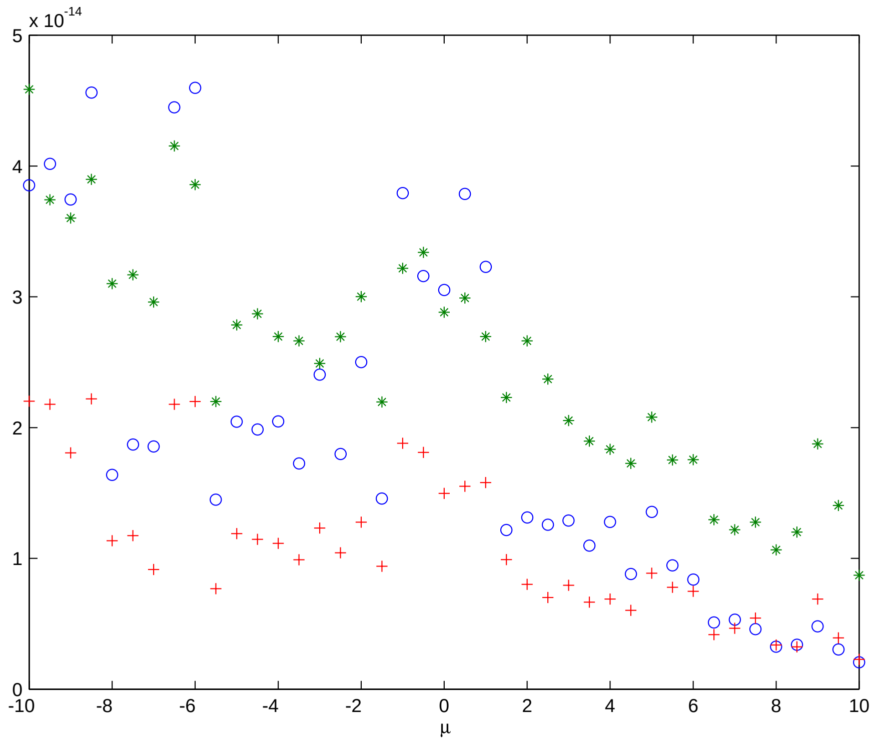

Fig. 3. Relative errors for computing $\varphi(A)=\varphi(A_0-\mu I)$ in matrix 2-norm (+), Frobenius norm (o) and matrix 1-norm (∗). A_0 is the 100×100 matrix given in (21).

where $n=p^2$. For a number of values of μ, we let $A=A_0-\mu I$ and calculate the matrix function $\varphi(A)$. The relative errors in various matrix norms are given in Fig. 3. These results are very satisfactory. Our numerical solutions are expected to have 14 correct digits.

Next, we study the efficiency of our method. For this purpose, we compare the execution time required by our new method and the standard method based on the eigenvalue decomposition. The latter is implemented using the LAPACK [1] routine for symmetric matrix eigenvalue problems based on Cuppen's the divide-and-conquer method [2]. All the programs are in double precision and compiled with the same option `fast-lsunperf` on a SUN Ultra 60 workstation with the compiler `f77` (version 5.0) from SUN Microsystems. Notice that we do not compile the LAPACK (and the BLAS) routines, since they are optimized and included in the SUN performance library. We have taken the matrix A_0 in Example 1 for various values of n. The required execution time (in seconds) are listed in Table 1. Three different values of μ are used. For $\mu=0$, 3 and 5, the largest eigenvalue of $A_0-\mu I$ (for all n listed) lies in the interval $(0,\infty)$, $(-1,0)$ and $(-\infty,0)$, respectively. Therefore, the three columns of T_n correspond to three different rational approximations. For the eigenvalue decomposition method, only the case $\mu=0$ is listed, since there is little difference for other values of μ. We have also implemented the eigenvalue method based on the QR algorithm [1] for symmetric matrix eigenvalue problems, the required execution time is longer. Our method has a clear advantage for all values of n in the table (from $n=100$ to 500). These results are consistent with the theoretical complexity of $\frac{10}{3}n^3+O(n^2)$.

Table 1
Time (in seconds) required for computing $\varphi(A)$ by the method in this paper T_n and the standard eigenvalue decomposition method T_{eig}, where $A = A_0 - \mu I$ as in Example 1

n	$T_n, \mu = 0$	$T_n, \mu = 3$	$T_n, \mu = 5$	$T_{eig}, \mu = 0$
100	0.065	0.053	0.060	0.068
150	0.164	0.152	0.153	0.233
200	0.321	0.300	0.300	0.603
250	0.574	0.610	0.537	1.11
300	0.973	0.966	0.885	1.85
350	1.45	1.48	1.37	2.43
400	2.37	2.37	2.14	3.73
450	3.17	3.19	3.07	5.24
500	4.66	4.46	4.47	7.20

5. Conclusion

We have developed an efficient numerical method for computing $\varphi(A)$ for a symmetric matrix A. Similar to our methods for matrix exponential [11], matrix square root [9] and matrix logarithm [10], it relies on the orthogonal reduction of the matrix A to a tridiagonal matrix T. The matrix function $\varphi(T)$ is then calculated by three rational approximations depending on the largest eigenvalue of A (or T). These rational approximations are given in the prime fraction form, so that $\varphi(T)$ can be easily evaluated. The method requires about 10/3 n^3 operations and it is more efficient than the standard method based on the eigenvalue decomposition. The accuracy and efficiency of our methods are demonstrated by numerical examples.

It is worthwhile to point out that if we only need to calculate the φ function of a symmetric tridiagonal matrix, the required number of operations is $O(n^2)$. Furthermore, if only one column of that matrix function is needed, the required number of operations is further reduced to $O(n)$. This feature is useful in the Lanczos method for evaluating $\varphi(A)v$, where A is large (possibly sparse) and symmetric, v is a given vector. While the Lanczos method approximates the vector $\varphi(A)v$ by a linear combination of first m Lanczos vectors, the number m is unknown and a convergence test that determines m is needed. This is reduced to computing the $(m, 1)$ entry of $\varphi(T_m)$ for various values of m. Our $O(m)$ method significantly speed up this process, since the standard method requires $O(m^3)$ operations. Similarly, if the Lanczos method is used to calculate $e^A v$, our method in [11] can be used to speed up the convergence test.

Appendix

For an even integer p, the Chebyshev rational approximation of the type $[p/p]$ for e^x on the interval $(-\infty, 0]$ can be written as

$$R_p(x) = \alpha_0^{(p)} + \sum_{j=1}^{p} \frac{\alpha_j^{(p)}}{x - \theta_j^{(p)}} = \alpha_0^{(p)} + 2\,\mathrm{Re} \sum_{j=1}^{p/2} \frac{\alpha_j^{(p)}}{x - \theta_j^{(p)}}$$

Table 2
Prime fraction coefficients of the [14/14] and [16/16] Chebyshev rational approximations of e^x for $x \leqslant 0$

	Real part	Imaginary part
$\alpha_0^{(14)}$	0.18321743782470014452E − 13	
$\alpha_1^{(14)}$	−0.27875161940069948877E + 02	−0.10214733999018530056E + 03
$\alpha_2^{(14)}$	0.46933274488692083907E + 02	0.45643649768622350551E + 02
$\alpha_3^{(14)}$	−0.23498232090999260508E + 02	−0.58083591296613695955E + 01
$\alpha_4^{(14)}$	0.48071120988101914274E + 01	−0.13209793837461011947E + 01
$\alpha_5^{(14)}$	−0.37636003877978724711E + 00	0.33518347029383815648E + 00
$\alpha_6^{(14)}$	0.94390253106412059435E − 02	−0.17184791958420579140E − 01
$\alpha_7^{(14)}$	−0.71542880635038012562E − 04	0.14361043349477292252E − 03
$\theta_1^{(14)}$	0.56231425727425832215E + 01	0.11940690463436614062E + 01
$\theta_2^{(14)}$	0.50893450605771116159E + 01	0.35888240290260982584E + 01
$\theta_3^{(14)}$	0.39933697105748160696E + 01	0.60048316422335547350E + 01
$\theta_4^{(14)}$	0.22697838292270027481E + 01	0.84617379730382146007E + 01
$\theta_5^{(14)}$	−0.20875863825471482874E + 00	0.10991260561898794116E + 02
$\theta_6^{(14)}$	−0.37032750494286560601E + 01	0.13656371871480423129E + 02
$\theta_7^{(14)}$	−0.88977731864749491206E + 01	0.16630982619899044992E + 02
$\alpha_0^{(16)}$	0.21248537104952237488E − 15	
$\alpha_1^{(16)}$	−0.64500878025539644564E + 02	−0.22459440762652096092E + 03
$\alpha_2^{(16)}$	0.11339775178483930464E + 03	0.10194721704215856386E + 03
$\alpha_3^{(16)}$	−0.62518392463207919933E + 02	−0.11190391094283228881E + 02
$\alpha_4^{(16)}$	0.15059585270023467196E + 02	−0.57514052776421820767E + 01
$\alpha_5^{(16)}$	−0.14793007113558000013E + 01	0.17686588323782937902E + 01
$\alpha_6^{(16)}$	0.41023136835410020949E − 01	−0.15743466173455468195E + 00
$\alpha_7^{(16)}$	0.21151742182466031443E − 03	0.43892969647380673895E − 02
$\alpha_8^{(16)}$	−0.50901521865224928712E − 06	−0.24220017652852287986E − 04
$\theta_1^{(16)}$	0.64161776990994341857E + 01	0.11941223933701386699E + 01
$\theta_2^{(16)}$	0.59481522689511774823E + 01	0.35874573620183223162E + 01
$\theta_3^{(16)}$	0.49931747377179964192E + 01	0.59968817136039421951E + 01
$\theta_4^{(16)}$	0.35091036084149180718E + 01	0.84361989858843750942E + 01
$\theta_5^{(16)}$	0.14193758971856659905E + 01	0.10925363484496722585E + 02
$\theta_6^{(16)}$	−0.14139284624888862117E + 01	0.13497725698892745388E + 02
$\theta_7^{(16)}$	−0.52649713434426468908E + 01	0.16220221473167927305E + 02
$\theta_8^{(16)}$	−0.10843917078696988026E + 02	0.19277446167181652284E + 02

since the coefficients $\{\alpha_j^{(p)}, \theta_j^{(p)}\}$ appear as complex conjugate pairs. For $p = 14$ and 16, these coefficients are calculated in [5] and listed in Table 2. The set of coefficients for $p = 14$ listed in [4] are not correct. Similarly, for an even integer q, we write the $[q/q]$ Chebyshev rational approximation of $\varphi(x)$ on $(-\infty, 0]$ as

$$\tilde{R}_q(x) = \tilde{\alpha}_0^{(q)} + \sum_{j=1}^{q} \frac{\tilde{\alpha}_j^{(q)}}{x - \tilde{\theta}_j^{(q)}} = \tilde{\alpha}_0^{(q)} + 2\,\mathrm{Re} \sum_{j=1}^{q/2} \frac{\tilde{\alpha}_j^{(q)}}{x - \tilde{\theta}_j^{(q)}}.$$

Table 3
Prime fraction coefficients of the [14/14] Chebyshev rational approximation of $\varphi(x)$ for $x \leqslant 0$

	Real part	Imaginary part
$\tilde{\alpha}_0^{(14)}$	0.68944296265527394984E − 15	
$\tilde{\alpha}_1^{(14)}$	−0.16598679663720768703E + 02	−0.39025784287223383670E + 02
$\tilde{\alpha}_2^{(14)}$	0.22963504666229092280E + 02	0.90186818220061090091E + 01
$\tilde{\alpha}_3^{(14)}$	−0.75350149609204679786E + 01	0.30951732326685966968E + 01
$\tilde{\alpha}_4^{(14)}$	0.65440260116974146874E + 00	−0.12832270822767467541E + 01
$\tilde{\alpha}_5^{(14)}$	0.17992885377582909731E − 01	0.12021513848300774960E + 00
$\tilde{\alpha}_6^{(14)}$	−0.22224782352681356103E − 02	−0.31546051373084948534E − 02
$\tilde{\alpha}_7^{(14)}$	0.16950103692838164789E − 04	0.18407950619535128862E − 04
$\tilde{\theta}_1^{(14)}$	0.65586170606958520061E + 01	0.12541312162940416924E + 01
$\tilde{\theta}_2^{(14)}$	0.60329668674314355458E + 01	0.37686693138308950662E + 01
$\tilde{\theta}_3^{(14)}$	0.49527072954283340179E + 01	0.63037280204340004157E + 01
$\tilde{\theta}_4^{(14)}$	0.32515207076218489674E + 01	0.88794008802441251574E + 01
$\tilde{\theta}_5^{(14)}$	0.80133602893611439276E + 00	0.11529259279403978988E + 02
$\tilde{\theta}_6^{(14)}$	−0.26587124072174283827E + 01	0.14320672417208411550E + 02
$\tilde{\theta}_7^{(14)}$	−0.78095944003956373966E + 01	0.17439142275890278426E + 02

The coefficients for $q = 14$ are computed in [5] are listed in Table 3.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen, LAPACK Users' Guide, SIAM, Philadelphia, 1992.

[2] J.J.M. Cuppen, A divide and conquer method for the symmetric eigenproblem, Numer. Math. 36 (1981) 177–195.

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

[4] G. Gallopoulos, Y. Saad, Efficient solution of parabolic equations by Krylov approximation methods, SIAM J. Sci. Statist. Comput. 13 (1992) 1236–1264.

[5] P.L. Ho, Best rational approximation of the exponential and other functions, Undergraduate Final Year Project, Department of Mathematics, City University of Hong Kong, 1999.

[6] M. Hochbruck, C. Lubich, Exponential integrators for quantum-classical molecular dynamics, BIT 39 (4) (1999) 620–645.

[7] M. Hochbruck, C. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 34 (5) (1997) 1911–1925.

[8] M. Hochbruck, C. Lubich, H. Selhofer, Exponential integrators for large systems of differential equations, SIAM J. Sci. Comput. 19 (5) (1998) 1552–1574.

[9] Y.Y. Lu, A Padé approximation method for square roots of symmetric positive definite matrices, SIAM J. Matrix Anal. Appl. 19 (3) (1998) 833–845.

[10] Y.Y. Lu, Computing the logarithm of a symmetric positive definite matrix, Appl. Numer. Math. 26 (4) (1998) 483–496.

[11] Y.Y. Lu, Exponentials of symmetric matrices through tridiagonal reductions, Linear Algebra Appl. 279 (1998) 317–324.

[12] C.B. Moler, C.F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Rev. 20 (1978) 801–836.