Int. J. Nonlinear Anal. Appl. 14 (2023) 1, 2881–2900
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.26081.3222

Approximating the matrix exponential, sine and cosine via the spectral method

Arezo Shakeri[a], Mahmoud Behroozifar[b,*]

[a]Department of Mathematics and Physics, Faculty of Science and Technology, University of Stavanger, Stavanger, Rogaland, Norway

[b]Department of Mathematics, Faculty of Science, Babol Noshirvani University of Technology, Babol, Mazandaran, Iran

(Communicated by Saman Babaie-Kafaki)

Abstract

This article is arranged to introduce three different algorithms for computing the matrix exponential, cosine and sine functions At for $0 \leq t \leq b$, for all $b \in \mathbb{R}^+$. To achieve this purpose, we deal with the spectral method based on Bernstein polynomials. Bernstein polynomials are briefly introduced and utilized to approximate the functions. The operational matrix of integration of Bernstein polynomials is stated and employed to reduce the dynamic systems to the linear algebraic systems. It is required to solve n linear algebraic systems for evaluating the matrix functions. By presenting the CPU time, it is displayed that the methods require a low amount of running time. Also, error analysis is discussed in detail. The outstanding point of this method is that the approximate exponential, cosine and sine matrix At_0, for all $t_0 \in [0, L]$ can be obtained with only one execution of the algorithm. These three different algorithms have common parts that can be used to practically reduce the computational volume. Some examples are provided to show the high performance of the methods.

Keywords: Matrix exponential function, Matrix cosine function, Matrix sine function, Spectral method, Operational matrix of integration, Bernstein polynomial
2020 MSC: Primary 65F60; Secondary 15A16, 65M70

1 Introduction

There are many methods for calculating the matrix exponential, cosine and sine functions. Matrix functions play effective roles in many fields of sciences and technologies, for example, engineering, physics and other sciences. As well, mathematical models of many physical, biological, and economic processes involve systems of linear ordinary differential equations (ODEs) with constant coefficients

$$x'(t) = Ax(t) \tag{1.1}$$

where A is a known, scalar, real or complex square matrix [22]. Solution of ODE (1.1) is

$$x(t) = e^{tA}x_0 \tag{1.2}$$

*Corresponding author
Email addresses: arezoshakeri93@gmail.com (Arezo Shakeri), m_behroozifar@nit.ac.ir, behroozifar2@gmail.com (Mahmoud Behroozifar)

Received: January 2022 *Accepted:* July 2022

such that $x(0) = x_0$ is an initial vector. In [15], a method to calculate the exponential matrix is presented with the goal of minimizing the mathematical prerequisites. In [14], the method of [15] is improved in view of computational aspects to calculate the exponential matrix. Reference [22] is a nice reference for the interested readers to see other methods for computing or approximating the matrix exponential function. In [22], several methods are elaborated and error bound of approximation of some methods is stated. Some proposed methods in [22] are listed as the series methods, ordinary differential equation methods, polynomial methods, matrix decomposition methods, splitting methods, Krylov space methods. In practice, consideration of computational stability and efficiency indicates that some of the methods are preferable to others, but none of them are completely satisfactory. In fact, these methods are applicable to the wide classes of matrices, but a method that works only on matrices with distinct eigenvalues will not be highly regarded [22]. Interested readers can also refer to [2, 3, 10, 17, 18, 20] and the references therein for more new information about matrix functions and the content about them.
In [24], the action of the large sparse matrix exponential is computed by combining incomplete orthogonalization and adaptive strategies for changing the step size and the dimension of the Krylov subspace. In [21], a software package (Expokit) is provided to compute the exponential matrix. In [1], an important method to compute the matrix exponential function is mentioned which is a reliable method. In [1], the scaling and squaring method for the matrix exponential is based on the approximation $e^A \approx (r_m(2^{-s}A))^{2^s}$, where $r_m(x)$ is the $[m/m]$ Padé approximate to e^x and the integers m and s are to be chosen.

In [19], a recursive algorithm is analyzed for computing the matrix cosine using the double-angle formula

$$\cos(2A) = 2\cos^2(A) - I.$$

Method of [19] must be implemented much times to calculate the cosine of a matrix which this is one of the defects of this method. In [6], an efficient algorithm to compute the matrix cosine based on Hermite matrix polynomial expansions has been proposed, improving the algorithms proposed by the authors in [5, 7]. Matrix exponential, cosine and sine functions can be expressed as the power series, respectively :

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \frac{1}{4!}A^4t^4 + \dots$$

$$\cos(At) = I - \frac{1}{2!}A^2t^2 + \frac{1}{4!}A^4t^4 - \frac{1}{6!}A^6t^6 + \dots$$

$$\sin(At) = At - \frac{1}{3!}A^3t^3 + \frac{1}{5!}A^5t^5 - \frac{1}{7!}A^7t^7 + \dots ,$$

where A is a constant $n \times n$ matrix. It is clear that the mentioned series are infinite and, in practice, they cannot be used.

Matrix functions arise most repeatedly in connection with the solution of differential systems and control theory [6]. For example, it is well known that the wave equation

$$\nu^2 \frac{\partial^2 \psi}{\partial \chi^2} = \frac{\partial^2 \psi}{\partial t^2} \tag{1.3}$$

plays an important role in many areas of engineering and applied sciences. When the wave equation (1.3) is solved by the spatially semi-discretization, the following system of second order differential equation is achieved

$$\begin{cases} y''(t) + A^2 y(t) = 0 \\ y(0) = y_0, \qquad y'(0) = y_1 \end{cases} \tag{1.4}$$

where A is a known square matrix and y_0 and y_1 are vectors. Problem (1.4) has the exact solution as

$$y(t) = \cos(At)y_0 + \sin(At)A^{-1}y_1 \tag{1.5}$$

where A denotes any non-singular matrix; see [19] for details. As result, matrix trigonometric functions play important roles in the solution of the second order differential systems, similarly, the matrix exponential e^{At} in first order differential systems [22].

Spectral method is applied in different topics of science to approximate the solution of differential equations, e.g. nonlinear Schrödinger equation [8], hyperbolic telegraph equation [16], time fractional diffusion equation [13], parabolic equation subject to specification of the mass [26].

Spectral method has not been utilized to approximate the matrix functions, yet. This motivates us to present the new methods to approximately compute the values of matrix exponential, cosine and sine functions. Our aim is to propose a method which that matrices e^{At}, $\cos(At)$, and $\sin(At)$ can be obtained for different values of t e.g. $t = \frac{1}{2}, \frac{\sqrt{2}}{2}, \frac{\pi}{5} \cdots, 1$ by one algorithm execution. Also, we plan to design methods based on the spectral method, that the singularity and eigenvalues of the matrix do not make any restrictions for them.

This paper is arranged in four sections: section 2 is assigned to the needed definitions and theorems. In section 3, we present the methods to approximate matrix functions. Illustrative examples are exhibited in section 4. Finally, section 5 summarizes the results.

2 Preliminaries

Objective of this section is to exhibit some definitions and concepts which are employed in this paper.

2.1 Bernstein polynomials

Bernstein polynomials of mth degree are introduced on the interval $[a, b]$ as [25, 26]

$$B_{i,m}(x) = \binom{m}{i} \frac{(x-a)^i (b-x)^{m-i}}{(b-a)^m}, \quad 0 \le i \le m$$

where $\binom{m}{i} = \frac{m!}{i!(m-i)!}$. These Bernstein polynomials form a basis on $[a, b]$. Bernstein polynomial of mth degree includes $m+1$ polynomials of degree m. The ith Bernstein polynomials of degree m can be derived using the following recursive relation

$$B_{i,m}(x) = \frac{(b-x)}{b-a} B_{i,m-1}(x) + \frac{x-a}{b-a} B_{i-1,m-1}(x), \quad i = 0, 1, \ldots, m$$

where $B_{i,m}(x) = 0$, if $i < 0$ or $i > m$. It can easily be shown that the Bernstein polynomials are positive, linear independent. It is important to notice that the Bernstein-Vandermonde matrix A is a strictly totally positive matrix [4, 9] which the points satisfy $0 < t_0 < t_1 < \cdots < t_m < 1$ which $A = [a_{i+1,j+1}]$ and $a_{i+1,j+1} = B_{j,m}(t_i)$ for $i, j = 0, \cdots, m$.

2.2 Function approximation

Suppose that $H = L^2[a, b]$ where $a, b \in \mathbb{R}$ and let $\{B_{0,m}, B_{1,m}, \cdots, B_{m,m}\} \subset H$ is the set of Bernstein polynomials of degree m and

$$Y = \text{Span}\{B_{0,m}, B_{1,m}, \cdots, B_{m,m}\}$$

and function f is an arbitrary element in H. Since Y is a finite dimensional vector space, function f has a unique best approximation out of Y [12], say $y_0 \in Y$, that is

$$\exists y_0 \in Y;\ \forall y \in Y\ ||f - y_0||_2 \le ||f - y||_2,$$

where $||f||_2 = \sqrt{\langle f, f \rangle}$ and $\langle f, g \rangle = \int_a^b f(t)g(t)dt$.

In [25], it is proved that a unique coefficient vector $c^T = [c_0, c_1, \cdots, c_m]$ exists such as

$$f \approx y_0 = \sum_{i=0}^{m} c_i B_{i,m} = c^T \phi, \tag{2.1}$$

where $\phi^T = [B_{0,m}, B_{1,m}, \cdots, B_{m,m}]$ and c^T can be obtained

$$c^T = \left(\int_a^b f(x)\phi(x)^T dx \right) Q^{-1},$$

which $Q = \langle \phi\ ,\ \phi \rangle = \int_a^b \phi(x)\phi(x)^T dx$ is said dual matrix of ϕ. Since the Bernstein polynomials are independent thus matrix Q is invertible [12].

Theorem 2.1. [12] Suppose that H be a Hilbert space on $[a,b]$ and Y be a closed subspace of H such that $\dim Y < \infty$ and $\{y_1, y_2, \cdots, y_n\}$ is any basis for Y. Let x be an arbitrary element in H and y_0 be the unique best approximation to x out of Y. Then

$$||x - y_0||_2^2 = \frac{G(x, y_1, y_2, \cdots, y_n)}{G(y_1, y_2, \cdots, y_n)},$$

where

$$G(x, y_1, y_2, \cdots, y_n) = \begin{vmatrix} \langle x, x\rangle & \langle x, y_1\rangle & \cdots & \langle x, y_n\rangle \\ \langle y_1, x\rangle & \langle y_1, y_1\rangle & \cdots & \langle y_1, y_n\rangle \\ \vdots & \vdots & & \vdots \\ \langle y_n, x\rangle & \langle y_n, y_1\rangle & \cdots & \langle y_n, y_n\rangle \end{vmatrix}$$

where $\langle f, g\rangle = \int_a^b f(t)g(t)dt$ and is called inner product of f and g.

Consequently, if we assume $Y = \text{Span}\{B_{0,m}, B_{1,m}, \cdots, B_{m,m}\}$ on $[a,b]$, the absolute error can be written in a simple form via Theorem 2.1

$$||f - y_0||_2 = \frac{\det[\int_a^b \psi_{(t)}\psi_{(t)}^T dt]}{\det[\int_a^b \phi_{(t)}\phi_{(t)}^T dt]}$$

which $\phi^T = [B_{0,m}, B_{1,m}, \cdots, B_{m,m}]$ and $\psi^T = [f, B_{0,m}, B_{1,m}, \cdots, B_{m,m}]$.

Exact value of approximation error is presented by the Theorem 2.1 and in the following lemma we present an upper bound of approximation error.

Lemma 2.2. [26] Suppose that the function $g : [a,b] \to \mathbb{R}$ is $m+1$ times continuously differentiable, $g \in C^{m+1}[a,b]$, and $Y = Span\{B_{0,m}, B_{1,m}, \cdots, B_{m,m}\}$. If $c^T\phi$ be the best approximation g out of Y then the mean error bound is presented as follows:

$$\|g - c^T\phi\|_2 \leq \frac{M(b-a)^{\frac{2m+3}{2}}}{(m+1)!\sqrt{2m+3}},$$

where $M = \max\limits_{x\in[a,b]} |g^{(m+1)}(x)|$.

Lemma 2.2 shows that the method of approximation converges to f when $m \to \infty$.

Similarly to Eq. (2.1), a vector of function can also be approximated in terms of Bernstein polynomials. Consider $X^*(t) \in \mathbb{R}^n$ is an arbitrary vector of function, then

$$X^*(t) \approx C\phi(t) \tag{2.2}$$

which $C = \left(\int_a^b X^*(t)\phi(t)^T dt\right) Q^{-1}$. Note that C is an unknown $n \times (m+1)$ matrix and c is an unknown $(m+1) \times 1$ vector.

Definition 2.3. Vector $X(t)^T = [X_1(t), X_2(t), \cdots, X_n(t)]$ is called the best approximation $X^*(t)^T = [X_1^*(t), X_2^*(t), \cdots, X_n^*(t)]$ out of Y whenever $X_i(t)$ is the best approximation of vector $X_i^*(t)$ toward Y for $i = 1, 2, \ldots, n$.

Lemma 2.4. Let vector $X^*(t)^T = [X_1^*(t), X_2^*(t), \cdots, X_n^*(t)]$ in which $X_i^*(t) : [a,b] \to \mathbb{R}$ is $m+1$ times continuously differentiable, that is, $X_i^*(t) \in C^{m+1}[a,b]$ for $i = 1, 2, \ldots, n$. If $X(t)^T = [X_1(t), X_2(t), \cdots, X_n(t)]$ is the best approximation $X^*(t)$ out of
$Y = Span\{B_{0,m}, B_{1,m}, \cdots, B_{m,m}\}$ then the mean error bound is

$$\|X^*(t) - X(t)\|_2 \leq \frac{M\sqrt{n}\,(b-a)^{\frac{2m+3}{2}}}{(m+1)!\,\sqrt{2m+3}},$$

where $M = \max\{M_1, M_2, \cdots, M_n\}$ and $M_i = \max\limits_{t\in[a,b]} |\frac{d^{m+1}}{dt^{m+1}} X_i^*(t)|$ for $i = 1, 2, \ldots, n$.

Proof . Assume that $\tilde{X_i^*}(t)$ is the Taylor polynomial of m-degree for $X_i^*(t)$ about point a

$$\tilde{X_i^*}(t) = X_i^*(a) + (X_i^*)'(a)(t-a) + \frac{(X_i^*)''(a)}{2!}(t-a)^2 + \cdots + \frac{1}{m!}\frac{d^m}{dt^m}X_i^*(a)(t-a)^m$$

which

$$|X_i^*(t) - \tilde{X}_i^*(t)| \leq M_i \frac{(t-a)^{m+1}}{(m+1)!}. \tag{2.3}$$

Since $\tilde{X}_i^*(t) \in Y$, $X_i(t)$ is the best approximation of $X_i^*(t)$ out of Y and from Eq. (2.3), we have

$$\|X^* - X\|_2^2 = \int_a^b \sum_{i=1}^n |X_i^*(t) - X_i(t)|^2 \, dt \leq \int_a^b \sum_{i=1}^n \left| X_i^*(t) - \tilde{X}_i^*(t) \right|^2 dt \leq$$
$$\int_a^b \sum_{i=1}^n \left(M_i \frac{(t-a)^{m+1}}{(m+1)!} \right)^2 dt \leq \frac{M^2}{(m+1)!^2} \sum_{i=1}^n \int_a^b (t-a)^{2m+2} dt = \frac{M^2 \, n \, (b-a)^{2m+3}}{(m+1)!^2 \, (2m+3)},$$

taking square root gives the desired result. □ Lemma 2.4 demonstrates that $\lim_{m\to\infty} \|X^* - X\|_2 = 0$. Also, Lemma 2.4 shows that the upper bound of approximate error is a multiple of $\sqrt{n}$. That is why these methods offer high-precision approximate solutions even in large scale.

2.3 Operational matrices

Operational matrices of integration P of mth degree Berstein polynomials are defined as follows which are explained in details in [25]

$$\int_a^t \phi(r) dr \approx P\phi(t), \qquad a < t \leq b. \tag{2.4}$$

In the following, a new simple technique is explained to determine the integration operational matrix P. At first, we introduce

$$h_1(t) = \int_a^t \phi(r) dr$$

and then multiply the both sides of equality (2.4) from right hand in $\phi^T(t)$, so

$$h_1(t)\phi^T(t) \approx P\phi(t)\phi^T(t). \tag{2.5}$$

Taking integral from both sides of Eq. (2.5) with respect to t, it is yielded

$$\int_a^b h_1(t)\phi^T(t) dt \approx P \int_a^b \phi(t)\phi^T(t) dt = PQ,$$

where Q as previously introduced, is dual operational matrix of $\phi(t)$. Therefore, it is derived

$$P = \left(\int_a^b h_1(t)\phi^T(t) dt \right) Q^{-1}. \tag{2.6}$$

3 Methodology

In this section, we introduce different techniques to approximately compute the matrix exponential, cosine and sine functions. These methods have high efficiency and can be considered a good substitution for traditional methods. These methods based on solving linear algebraic equations set. Let A is a constant $n \times n$ matrix, the vector e_i and A_i are the ith column of the identify matrix and the ith column of matrix A, respectively.

3.1 Matrix exponential function

In the following, it is planned to approximate the vector $e^{At}e_i$ for $i = 1, 2, \ldots, n$. Eqs. (1.1) and (1.2) result that the following equation has the exact solution $e^{At}e_i$

$$\begin{cases} X'(t) = AX(t) \\ X(0) = e_i \end{cases} \tag{3.1}$$

for $i = 1, 2, \ldots, n$. Suppose that the approximate solution of Eq. (3.1) is denoted by $X_i(t)$, i.e. $X(t) = e^{At}e_i \approx X_i(t)$ for $i = 1, 2, \ldots, n$.
To achieve this goal, let $X_i'(t)$ is approximated by Bernstein polynomilal using Eq. (2.2), that is, $X_i'(t) \approx C\phi(t)$, which C is an unknown matrix, therefore

$$X_i(t) = X_i(0) + \int_0^t X_i'(s)ds \approx e_i + CP\phi(t) \tag{3.2}$$

where P is the integration operational matrix determined by 2.4. Substituting $X_i'(t) \approx C\phi(t)$ and Eq. (3.2) in Eq. (3.1) concludes

$$X_i'(t) = AX_i(t) \Longrightarrow C\phi(t) = A(e_i + CP\phi(t)) = A_i + ACP\phi(t) = E_i\phi(t) + ACP\phi(t) \tag{3.3}$$

which the matrix E_i is derived from $A_i = E_i\phi(t)$ using Eq. (2.2).
Eq. 3.3 gives the system

$$C - ACP = E_i,$$

this system is solved using Mathematica software to acquire the matrix C. After characterizing the matrix C, we have

$$X(t) = e^{At}e_i \approx X_i(t) = A_i + ACP\phi(t)$$

for $i = 1, 2, \ldots, n$.

Algorithm: To approximate the ith column of e^{At}, fulfill the following procedure for $i = 1, 2, \ldots, n$:

1. Approximate A_i as $A_i = E_i\phi(t)$,
2. To determine C, solve the set of algebraic equation $C - ACP = E_i$,
3. Calculate $X_i(t) = A_i + ACP\phi(t)$.

3.2 Matrix cosine function

Eqs. (1.4) and (1.5) guide us to achieve a technique to approximate the matrix cosine function detailed as follows: From, Eqs. (1.4) and (1.5) we can realize that the exact solution of the dynamic system (3.4) is $X(t) = \cos(At)e_i$

$$\begin{cases} X''(t) + A^2X(t) = 0 & \text{i=1, 2, \ldots, n} \\ X(0) = e_i, \qquad X'(0) = 0. \end{cases} \tag{3.4}$$

In the following, we aim to bring forward the vector $X_i(t)$ as an approximate of the exact solution of the dynamic system 3.4, i.e. $X(t) = \cos(At)e_i \approx X_i(t)$. For this purpose, it is considered $X_i''(t) \approx C\phi(t)$ via Eq. (2.2), which C is an unknown matrix, so

$$X_i(t) = X_i(0) + tX_i'(0) + \int_0^t \int_0^r X_i''(s)dsdr \approx e_i + C\int_0^t \int_0^r \phi(s)dsdr = e_i + CP^2\phi(t) \tag{3.5}$$

where P is the integration operational matrix determined by 2.4. Replacing $X_i''(t) \approx C\phi(t)$ and Eq. (3.5) in Eq. (3.4) acquires

$$0 = C\phi(t) + A^2(e_i + CP^2\phi(t)) = C\phi(t) + A^2e_i + A^2CP^2\phi(t) = C\phi(t) + K_i\phi(t) + A^2CP^2\phi(t) \tag{3.6}$$

where K_i is achieved from $A^2e_i = K_i\phi(t)$. Matrix C is detected from the below system of algebraic equations and using Mathematica software

$$C + K_i + A^2CP^2 = 0$$

and then the approximate solution is obtained as $X_i(t) \approx e_i + CP^2\phi(t)$.

Algorithm: To approximate the ith column of $\cos(At)$, accomplish the following process for $i = 1, 2, \ldots, n$:

1. Estimate A^2e_i as $A^2e_i = K_i\phi(t)$,
2. To specify C, solve the set of algebraic equation $C + K_i + A^2CP^2 = 0$,
3. Compute $X_i(t) = e_i + CP^2\phi(t)$.

3.3 Matrix sine function

This part proceeds analogously to the subsection 3.2. Vector $X(t) = \sin(At)e_i$ for $i = 1, 2, \ldots, n$ is the exact solution of the dynamic system (3.7)

$$\begin{cases} X''(t) + A^2X(t) = 0 \\ X(0) = 0, \qquad X'(0) = A_i. \end{cases} \tag{3.7}$$

We aim to $X(t) = \sin(At)e_i \approx X_i(t)$, therefore consider that $X_i''(t) \approx C\phi(t)$. We have

$$X_i(t) = X_i(0) + tX_i'(0) + \int_0^t \int_0^r X_i''(s)dsdr \approx tA_i + C\int_0^t \int_0^r \phi(s)dsdr = tA_i + CP^2\phi(t). \tag{3.8}$$

By estimating $t = l^T\phi(t)$ and substituting it in Eq. (3.8), we lead to

$$X_i(t) \approx (A_i \otimes l^T)\phi(t) + CP^2\phi(t). \tag{3.9}$$

where $\otimes$ is tensor product. To distinguish matrix C, the following equations set is derive by putting $X_i''(t) \approx C\phi(t)$ and Eq. (3.9) in the dynamic system (3.7)

$$C + A^2(A_i \otimes l^T + CP^2) = 0,$$

then Eq. (3.9) discloses an approximate of $\sin(At)e_i$.

Algorithm: To approximate the ith column of $\sin(At)$, employ the following process for $i = 1, 2, \ldots, n$:

1. Approximate t as $t = l^T\phi(t)$,
2. To distinguish C, solve the set of algebraic equation $C + A^2(A_i \otimes l^T + CP^2) = 0$,
3. Calculate $X_i(t) = (A_i \otimes l^T)\phi(t) + CP^2\phi(t)$.

4 Numerical findings

In examples 4.1-4.4, the presented methods are examined on three different types of the matrices to trust the performance of the method. In these examples, plots of error vector in L^2-norm of matrix functions are used to investigate the approximate solution with the exact solution. Afterward, the residual vector is implemented to check the validity of approximate solution for example 4.4 because the scale of example is large and the exact solution is not available. The presented method has the ability to define on interval $[0, b], \forall b \in \mathbb{R}^+$, therefore different matrix functions can be achieved. The outstanding point of this method is that the approximate matrix functions of matrix $At, \forall t \in [0, b]$ can be obtained with only one execution of the method, so various matrix functions can be achieved by setting t. In the following, we concentrate on interval $[0, 1]$(i.e. $a = 0,\ b = 1$), so the Bernestein polynomial and operational matrices of integration on interval $[0, 1]$ are considered and approximation of the matrix functions are exhibited on $[0, 1]$.

The main difference between this method and the other existing method is that the proposed method determines the matrix function At for $0 \le t \le 1$, but other existing methods are implemented only on matrix A. Consequently, authors cannot make fair comparisons. As mentioned in [20], the methods presented in [2] are based on the identities $\cos(A) = \frac{e^{iA}+e^{-iA}}{2}$, $\sin(A) = \frac{e^{iA}-e^{-iA}}{2i}$ and the use of a diagonal Padé approximations of the exponential e^{iA}. Method of [20] is a similar with method of [2] for approaching matrices $\cos(A)$ and $\sin(A)$ with fewer product number, while our method estimates the matrix functions e^{At}, $\cos(At)$ and $\sin(At)$ for $\forall t \in [0, 1]$.

The simplicity of our method and the small number of processes are important advantages of the proposed method than method [2]. In algorithm of method [2], more than 45 steps are required to compute the matrix functions, but 3 steps are required for our method.

Example 4.1. Let the following matrix $A_{11\times 11}$

$$A=\begin{pmatrix}
-\frac{47}{40} & \frac{679}{160} & \frac{35}{32} & \frac{13}{160} & \frac{47}{40} & -\frac{47}{20} & -\frac{81}{160} & \frac{19}{160} & \frac{163}{160} & -\frac{163}{160} & -\frac{47}{80} \\
-\frac{3}{10} & \frac{71}{40} & \frac{3}{8} & -\frac{3}{40} & \frac{3}{10} & -\frac{3}{5} & -\frac{9}{40} & -\frac{9}{40} & \frac{27}{40} & -\frac{27}{40} & -\frac{3}{20} \\
-\frac{13}{8} & \frac{761}{160} & \frac{289}{160} & -\frac{13}{160} & \frac{61}{40} & -\frac{61}{20} & -\frac{27}{32} & -\frac{19}{160} & \frac{277}{160} & -\frac{277}{160} & -\frac{61}{80} \\
-\frac{219}{280} & \frac{687}{224} & \frac{1123}{1120} & -\frac{27}{224} & \frac{191}{280} & -\frac{191}{140} & -\frac{677}{1120} & \frac{27}{224} & \frac{27}{224} & -\frac{27}{224} & -\frac{191}{560} \\
\frac{69}{140} & -\frac{189}{80} & -\frac{17}{16} & \frac{79}{560} & -\frac{9}{140} & \frac{167}{210} & \frac{337}{560} & \frac{97}{560} & -\frac{19}{240} & \frac{19}{240} & \frac{129}{280} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & -\frac{1}{12} & \frac{1}{12} & 0 \\
-\frac{43}{56} & \frac{559}{224} & \frac{215}{224} & -\frac{43}{224} & \frac{43}{56} & -\frac{43}{28} & -\frac{97}{224} & \frac{43}{224} & \frac{43}{224} & -\frac{43}{224} & -\frac{43}{112} \\
\frac{9}{8} & -\frac{117}{32} & -\frac{45}{32} & \frac{9}{32} & -\frac{9}{8} & \frac{9}{4} & \frac{27}{32} & \frac{7}{32} & -\frac{25}{32} & \frac{25}{32} & \frac{9}{16} \\
\frac{7}{20} & -\frac{77}{80} & -\frac{21}{80} & -\frac{7}{80} & -\frac{7}{20} & \frac{7}{10} & \frac{7}{80} & \frac{7}{80} & -\frac{29}{80} & -\frac{19}{80} & \frac{7}{40} \\
\frac{7}{20} & -\frac{77}{80} & -\frac{21}{80} & -\frac{7}{80} & -\frac{7}{20} & \frac{7}{10} & \frac{7}{80} & \frac{7}{80} & -\frac{49}{80} & \frac{1}{80} & \frac{7}{40} \\
-\frac{51}{28} & \frac{4017}{560} & \frac{873}{560} & -\frac{221}{560} & \frac{347}{140} & -\frac{347}{70} & -\frac{103}{112} & -\frac{883}{560} & \frac{3669}{560} & -\frac{3669}{560} & -\frac{107}{280}
\end{pmatrix}$$

Matrix A is singular, diagonalizable and $\| A \|_2 = 17.17277$ and all eigenvalues are less than 1. For verifying the reliability of method, the plots of error vector in L^2-norm of matrix exponential function are exhibited in Figure 1, i.e. plots of $||X_i - X_i^*||_2$ for $i = 1, 4, 8, 11$ where X_i, X_i^*, X and X^* are the ith column of matrix X, the ith column of matrix X^*, the approximate matrix exponential function of A and the exact matrix exponential function of A, respectively. Figure 2 propose the plots of error vector in L^2-norm of the matrix cosine function of A and analogously Figure 3 for matrix sine function.

Since various matrix functions are obtained by specifying arbitrarily the value $t \in [0, 1]$, so Figures 1, 2, and 3 exhibit 2-norm error vector ith column of all these matrix functions. This means that this method needs to run once individually and then select the desired amount t.

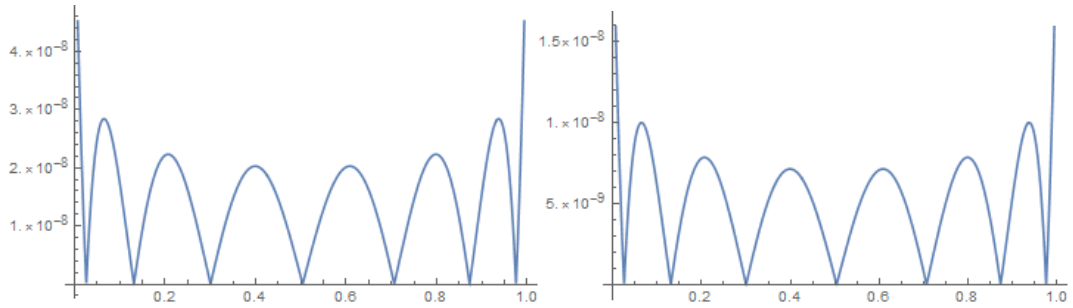
High precise of the method can be seen in Figures 1, 2, and 3 for a small value $m = 6$. It is clear that the error vanishes by increasing m by more computation time. We have tried to use a small amount for m to get an appropriate precision, otherwise, high accuracy can be reached by increasing m. This same result goes for other examples.

Also, CPU time of examples 4.1 is displayed in Table 1. Table 1 shows the low running time that it can be seen as one of the advantages of the method. The presented times in Table 1 are the needed times to calculate the approximate and exact solutions for all vectors and plot the error vectors. It may seem sounds that the CPU time of this method is not very small in comparison with the other existing methods, but this comparison is not correct, because the proposed method gives the matrix functions At for $\forall t \in [0, 1]$ but the other methods present matrix functions A. Therefore, this comparison is not suitable for this method. It is notable that the used PC is Intel(R) Core(TM) i7-7700K CPU @4.20GHz 4.20GHz.

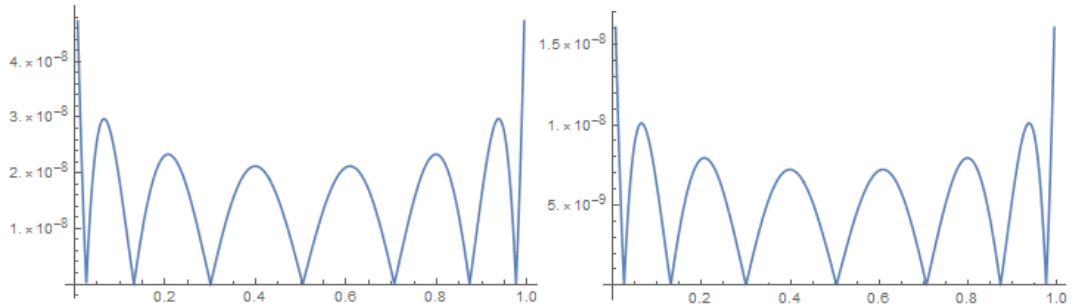
Table 1: CPU time of examples 4.1-4.3 for $m = 6$ in terms of seconds.

	e^{At}	$\cos(At)$	$\sin(At)$
Example 4.1	5	7	7
Example 4.2	14	50	52
Example 4.2	9	25	26

Example 4.2. Assume the following non singular and diagonazable matrix

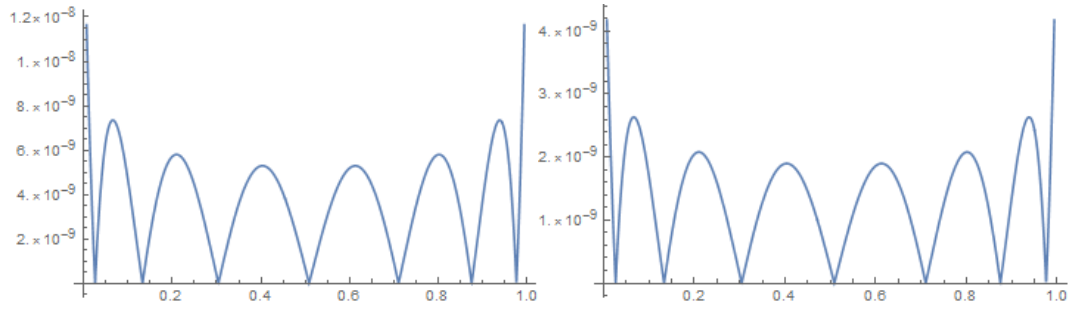


(a) Error vector of X_1 (b) Error vector of X_4

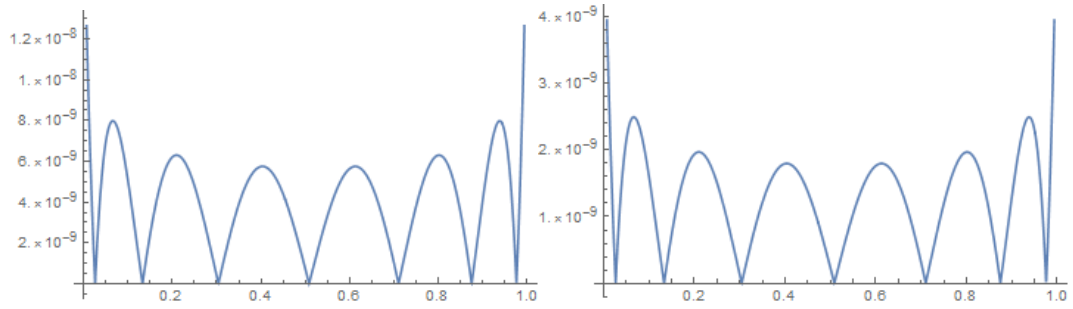


(c) Error vector of X_8 (d) Error vector of X_{11}

Figure 1: 2-norm error vector ith column of e^{At} on $[0, 1]$ for example 4.1.

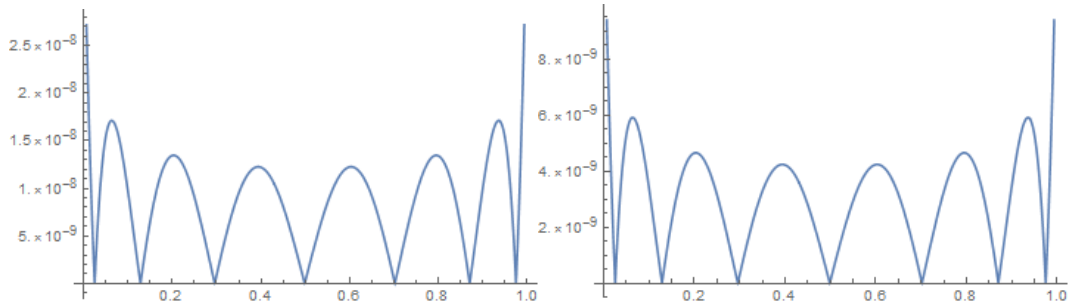


(a) Error vector of X_1 (b) Error vector of X_4

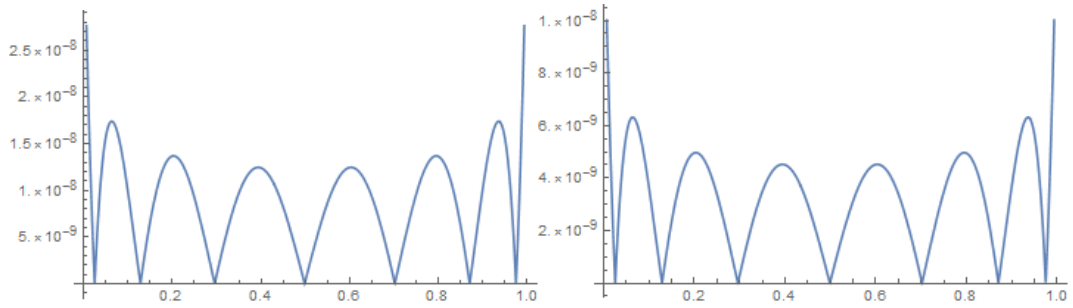


(c) Error vector of X_8 (d) Error vector of X_{11}

Figure 2: 2-norm error vector ith column of $\cos(At)$ on $[0, 1]$ for example 4.1.



(a) Error vector of X_1 (b) Error vector of X_4



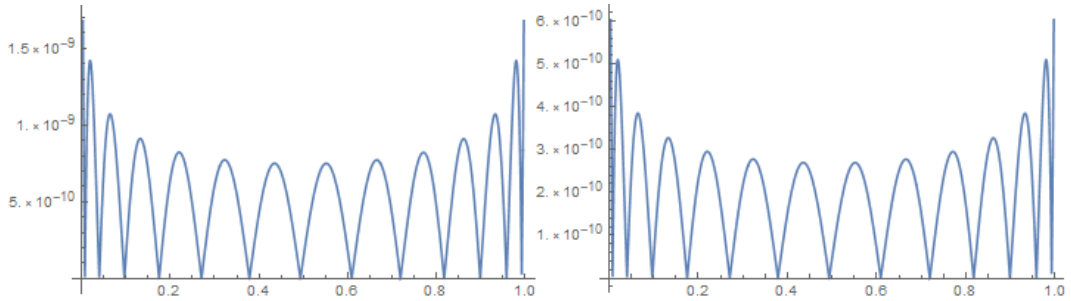
(c) Error vector of X_8 (d) Error vector of X_{11}

Figure 3: 2-norm error vector ith column of $\sin(At)$ on $[0,1]$ for example 4.1.

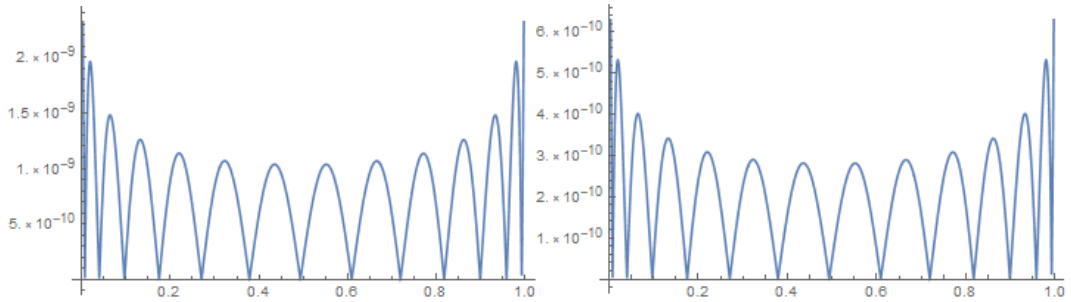
$$
\mathrm{A}=\begin{pmatrix}
\frac{227}{40} & -\frac{2319}{160} & -\frac{139}{32} & -\frac{53}{160} & -\frac{187}{40} & \frac{187}{20} & \frac{321}{160} & \frac{501}{160} & -\frac{1803}{160} & \frac{1803}{160} & \frac{187}{80} \\
\frac{9}{5} & -\frac{113}{20} & -\frac{9}{4} & \frac{9}{20} & -\frac{9}{5} & \frac{18}{5} & \frac{27}{20} & \frac{27}{20} & -\frac{81}{20} & \frac{81}{20} & \frac{9}{10} \\
-\frac{1}{8} & \frac{401}{160} & \frac{329}{160} & -\frac{213}{160} & \frac{21}{40} & -\frac{21}{20} & -\frac{51}{32} & -\frac{139}{160} & \frac{157}{160} & -\frac{157}{160} & -\frac{21}{80} \\
-\frac{213}{40} & \frac{2817}{160} & \frac{1049}{160} & -\frac{101}{160} & \frac{229}{40} & -\frac{229}{20} & \frac{241}{160} & \frac{357}{160} & \frac{69}{160} & -\frac{69}{160} & -\frac{229}{80} \\
\frac{27}{5} & -\frac{111}{5} & -4 & \frac{8}{5} & -\frac{37}{5} & \frac{44}{5} & \frac{14}{5} & \frac{29}{5} & -\frac{82}{5} & \frac{82}{5} & \frac{6}{5} \\
0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & \frac{13}{4} & -\frac{13}{4} & 0 \\
\frac{39}{8} & -\frac{507}{32} & -\frac{195}{32} & \frac{39}{32} & -\frac{39}{8} & \frac{39}{4} & -\frac{43}{32} & -\frac{39}{32} & -\frac{39}{32} & \frac{39}{32} & \frac{39}{16} \\
\frac{17}{8} & -\frac{221}{32} & -\frac{85}{32} & \frac{17}{32} & -\frac{17}{8} & \frac{17}{4} & \frac{51}{32} & \frac{47}{32} & -\frac{129}{32} & \frac{129}{32} & \frac{17}{16} \\
5 & -\frac{55}{4} & -\frac{15}{4} & -\frac{5}{4} & -5 & 10 & \frac{5}{4} & \frac{5}{4} & -\frac{17}{2} & \frac{11}{2} & \frac{5}{2} \\
5 & -\frac{55}{4} & -\frac{15}{4} & -\frac{5}{4} & -5 & 10 & \frac{5}{4} & \frac{5}{4} & -\frac{35}{4} & \frac{23}{4} & \frac{5}{2} \\
7 & -\frac{671}{20} & -\frac{79}{20} & \frac{43}{20} & -\frac{61}{5} & \frac{122}{5} & \frac{33}{4} & \frac{269}{20} & -\frac{947}{20} & \frac{947}{20} & \frac{11}{10}
\end{pmatrix}
$$

whose eigenvalues are $\left\{-5,-5,-3,-3,2,2,1,\frac{1}{4},\frac{1}{5},\frac{1}{5},-\frac{1}{8}\right\}$. Some eigenvalues of A are more than or equal to 1 and $\| A \|_2 = 99.10244$. Findings of example 4.2 are disclosed in Figures 4, 5 and 6 with $m = 12$. From the results of example 4.2, the accuracy of the method is good but in comparison with example 4.1, the accuracy of the method is decreased owing to some eigenvalues of matrix A are more than or equal to 1. This example well shows the effect of eigenvalues on method accuracy. Table 1 presents the calculation time of method for example 4.2 which displays that the processing time is normal. It is noteworthy that matrices e^{At}, $\cos(At)$, and $\sin(At)$ can be obtained for different values of t e.g. $t = \frac{1}{2}, \frac{\sqrt{2}}{2}, \frac{\pi}{5} \cdots, 1$ by one run of each algorithm.

Example 4.3. In this example, we aim to consider a singular and non-diagonalizable matrix to more survey the

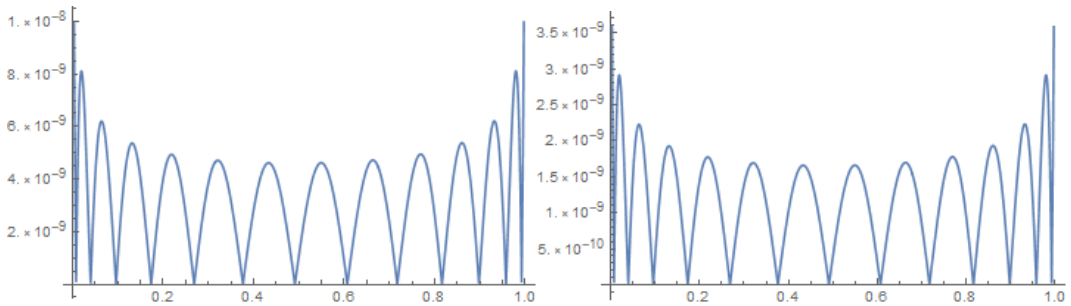


(a) Error vector of X_1 (b) Error vector of X_4

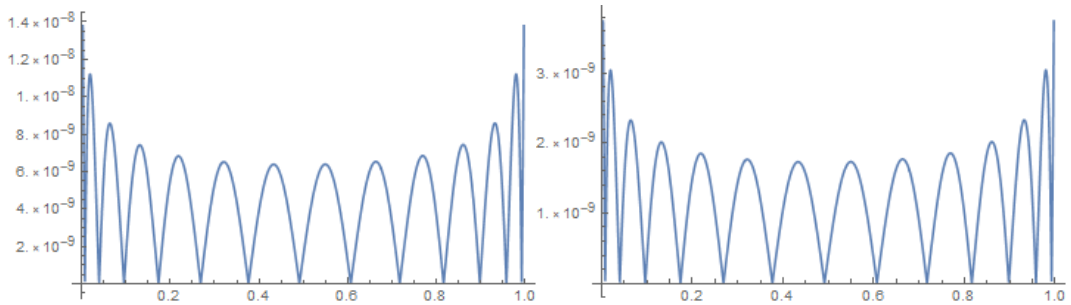


(c) Error vector of X_8 (d) Error vector of X_{11}

Figure 4: 2-norm error vector ith column of e^{At} on $[0, 1]$ for example 4.2.

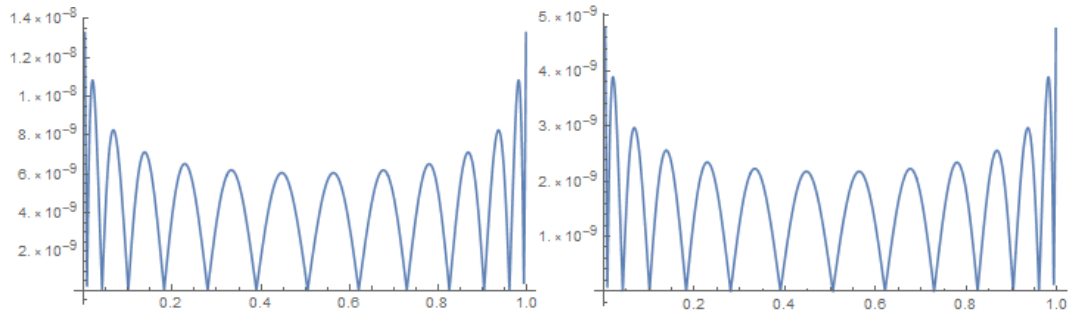


(a) Error vector of X_1 (b) Error vector of X_4

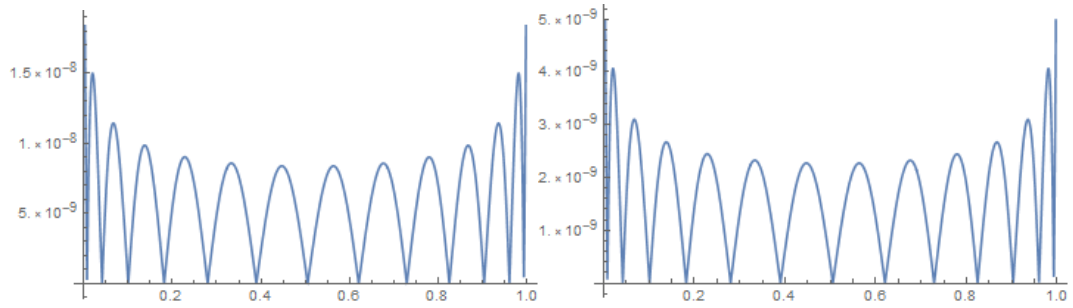


(c) Error vector of X_8 (d) Error vector of X_{11}

Figure 5: 2-norm error vector ith column of matrix cosine function for example 4.2.



(a) Error vector of X_1 (b) Error vector of X_4



(c) Error vector of X_8 (d) Error vector of X_{11}

Figure 6: 2-norm error vector ith column of $\sin(At)$ on $[0, 1]$ for example 4.2.

applicability of the method. Let matrix $A_{11\times 11}$ be

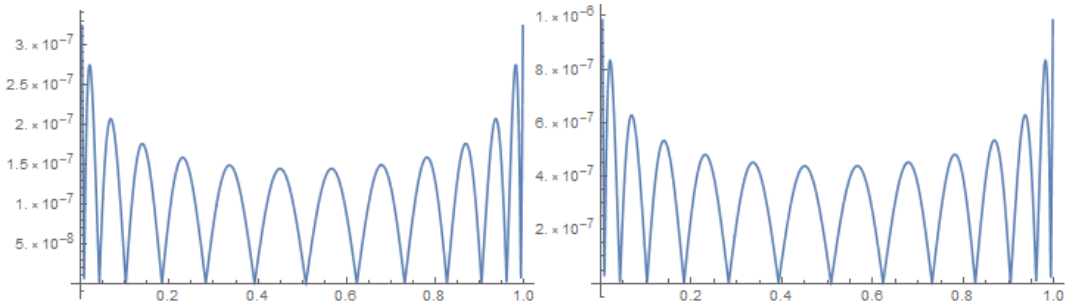
$$A = \begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 1 & 0 & -4 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & -3 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 5 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

which some eigenvalues are complex. In light of some eigenvalues of A are complex consequently matrix A is not diagonalizable and also $\| A \|_2 = 7.53299$. Figures 7, 8 and 9 disclose the numerical results of example 4.3 with $m = 12$ and the Table 1 indicates the calculation time. Survey of the validity of the approximate solution leads us to the accuracy of the method is good even when the given matrix is non-diagonalizable.

Example 4.4. In this example, we aim to test the presented methods on a high scale matrix. Then, matrix A with dimensions 128×128 is supposed and built using the following two commands in Mathematica^{TM} software:

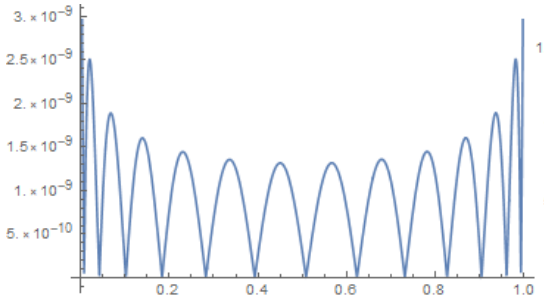
$$A = SparseArray[i\_, i\_ - > 2, i\_, j\_/; Abs[i - j] == 1 - > -1, 128, 128];$$
$$A[[120, 111]] = 50000;$$

Complex eigenvalues confirm that A is non-diagonalizable. It should be noted that $\| A \|_2 = 50000.00024$. Matrix exponential, cosine and sine functions are determined by $m = 12$. Value $m = 12$ is selected to mark the method can be implemented by more value of m even on the large scale. Considering the high number of columns, the ith column of matrix functions are specified for $i = 1, 5, 25, 125$. Since exact solutions are not available then residual vectors for

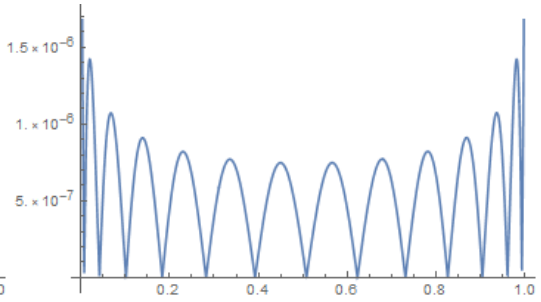


(a) Error vector of X_1

(b) Error vector of X_4

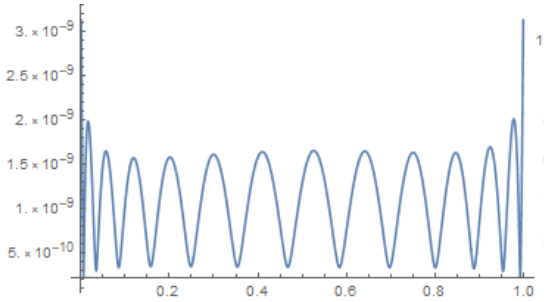


(c) Error vector of X_8

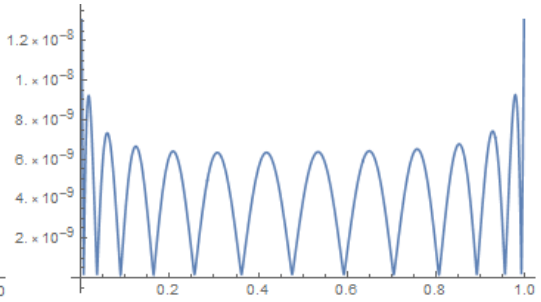


(d) Error vector of X_{11}

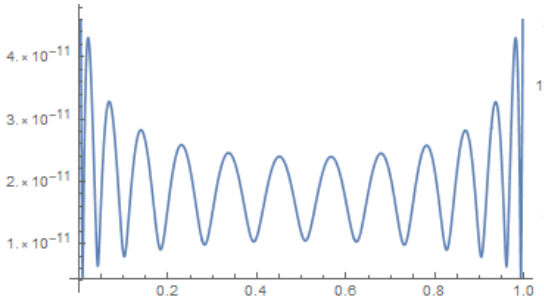
Figure 7: 2-norm error vector ith column of e^{At} on $[0, 1]$ for example 4.3.



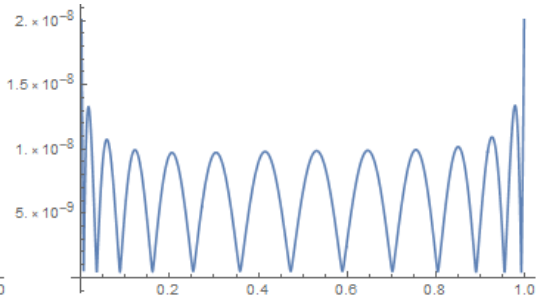
(a) Error vector of X_1



(b) Error vector of X_4

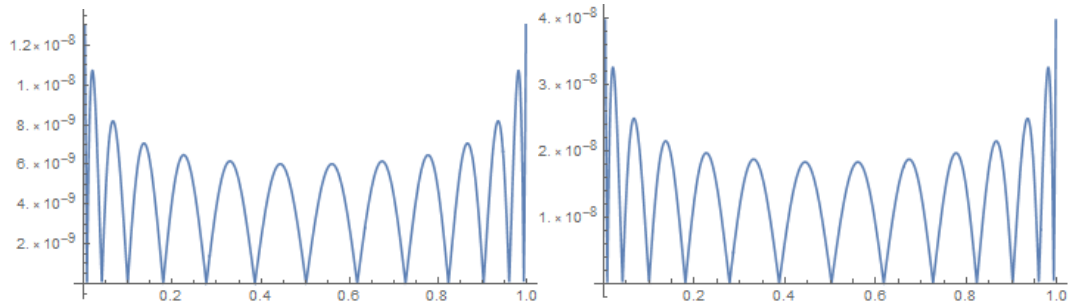


(c) Error vector of X_8

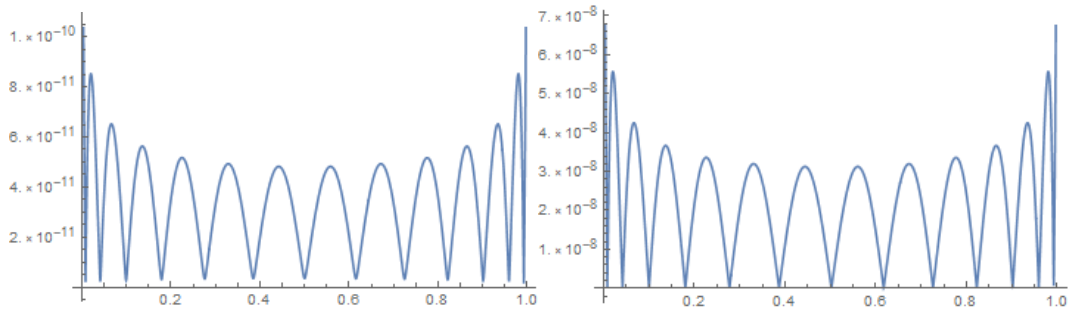


(d) Error vector of X_{11}

Figure 8: 2-norm error vector ith column of $\cos(At)$ on $[0, 1]$ for example 4.3.



(a) Error vector of X_1 (b) Error vector of X_4



(c) Error vector of X_8 (d) Error vector of X_{11}

Figure 9: 2-norm error vector ith column of $\sin(At)$ on $[0, 1]$ for example 4.3.

matrix exponential, cosine and sine functions are defined in L^2-norm respectively

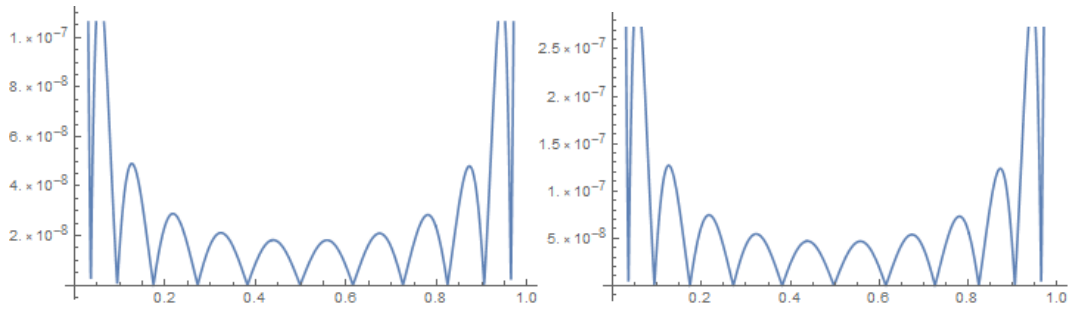
$$Exp - R_i(x) = ||X_i' - AX_i||_2,$$
$$Cos - R_i(x) = ||X_i'' + A^2X_i||_2,$$
$$Sin - R_i(x) = ||X_i'' + A^2X_i||_2$$

where X_i is the appropriate approximate solution. Residual vectors are applied to investigate whether the approximate solutions satisfy in their related equations.
Figures 10, 11 and 12 illustrate that the efficiency of the method on a large scale with an acceptable precise.

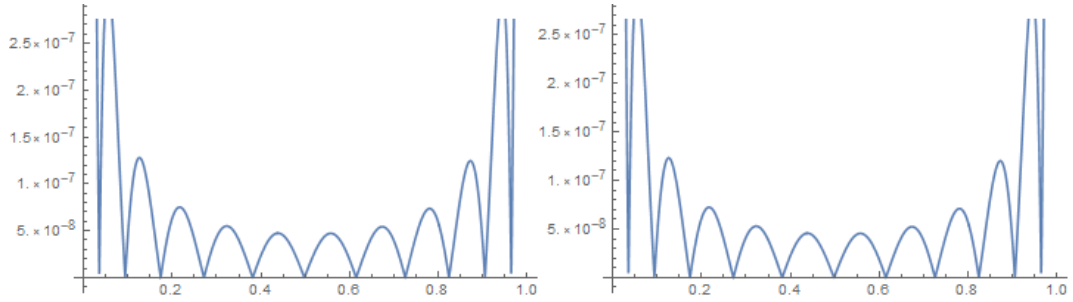
Example 4.5. In the previous examples, we intend to show the performance of the proposed method in different situations. In this example, our goal is to compare this method with some other methods. Therefore, the following matrix given in [20, 3] is assumed. Consider 2 special matrices (green) of the form

$$A = \begin{pmatrix} 1 & \lambda \\ 0 & -1 \end{pmatrix}$$

where $\lambda = 10000, 10^8$ and $\| A \|_2 = \lambda$. Presented method with $m = 12$ is executed for approaching matrix functions e^{At}, $\cos(At)$ and $\sin(At)$ and findings are displayed in Figures 13 and 14. According to the Figures 13 and 14, small values of approximate errors indicate the high accuracy of the given method towards the presented methods in [20, 3]. CPU time is revealed in Table 2. As another outstanding point of this method is that matrices P, Q, Q^{-1} are joint in the algorithm's matrix functions e^{At}, $\cos(At)$ and $\sin(At)$, so this reduces computation volume.

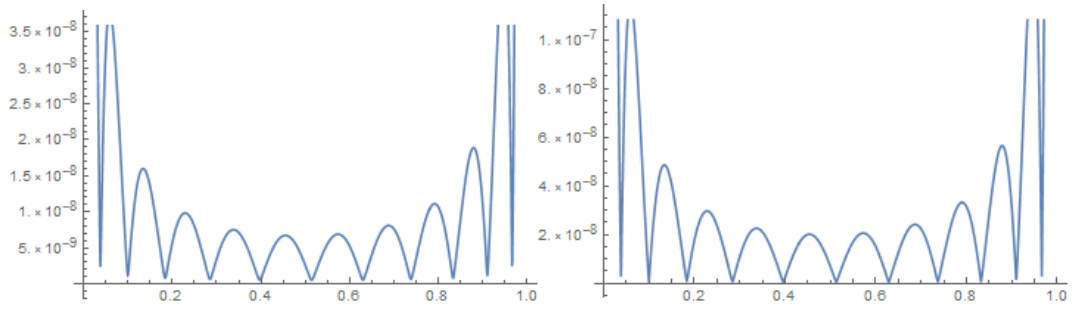


(a) Error vector of X_1 (b) Error vector of X_5

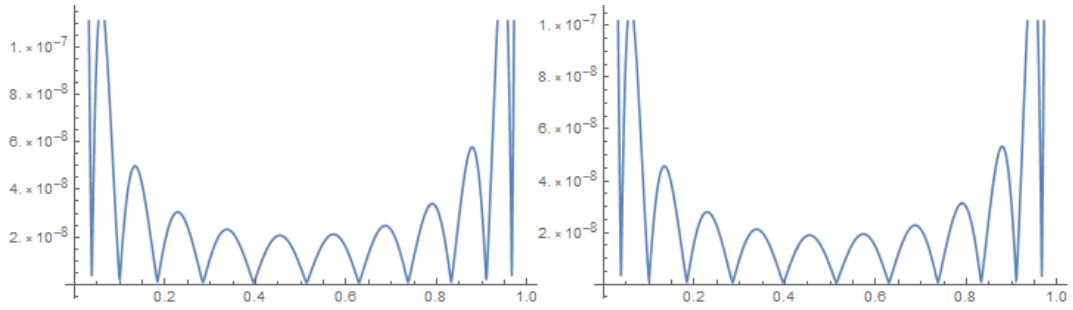


(c) Error vector of X_{25} (d) Error vector of X_{125}

Figure 10: Plots of residual vector in L^2-norm of $\cos(At)$ on $[0, 1]$ for example 4.4.

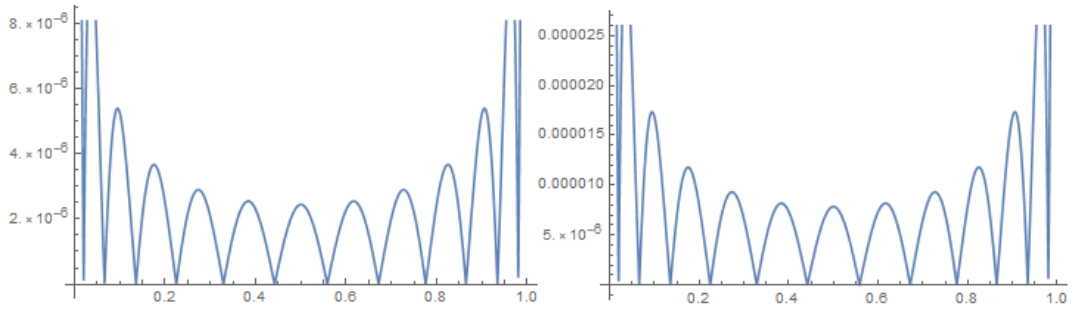


(a) Error vector of X_1 (b) Error vector of X_5



(c) Error vector of X_{25} (d) Error vector of X_{125}

Figure 11: Plots of residual vector in L^2-norm of $\sin(At)$ on $[0, 1]$ for example 4.4.



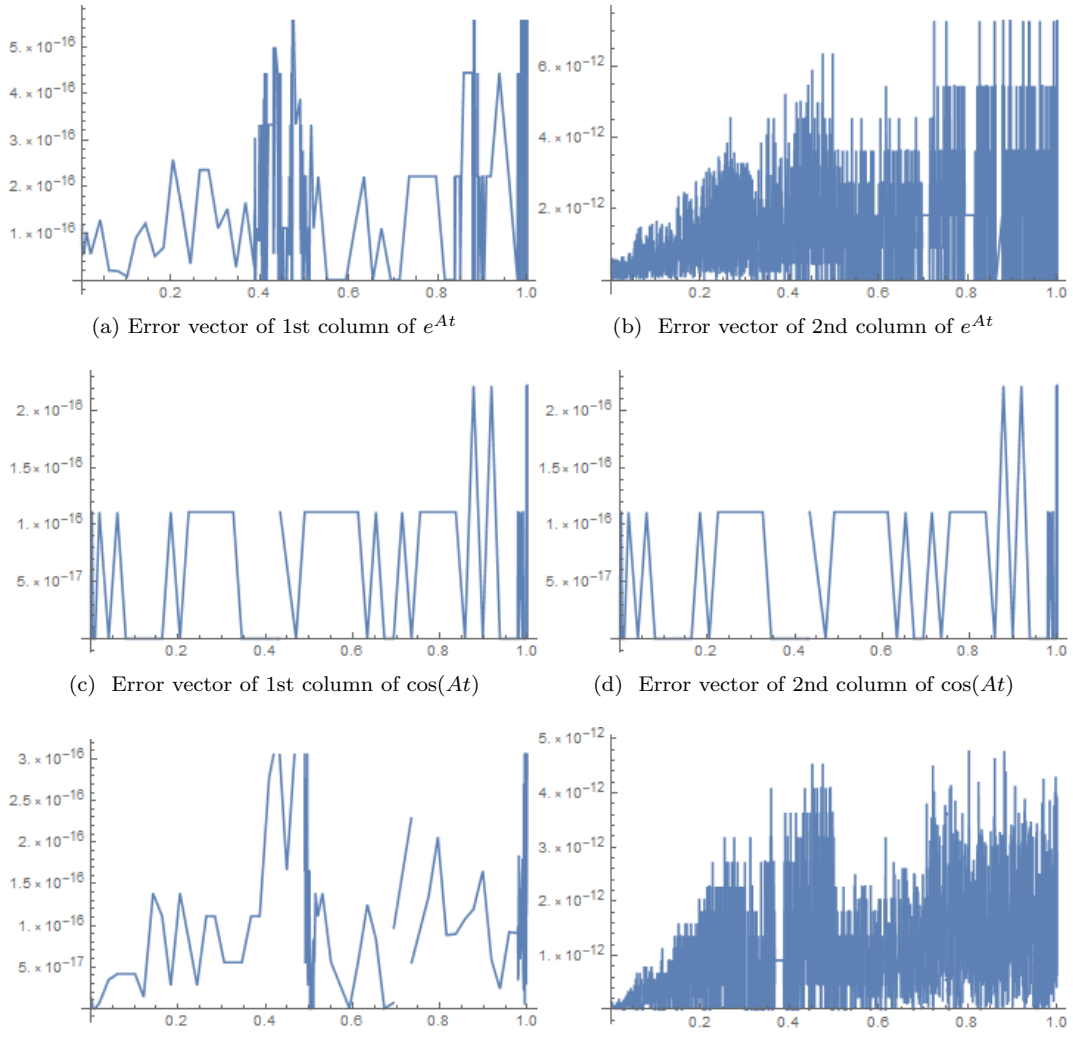
(a) Error vector of X_1 (b) Error vector of X_5

(c) Error vector of X_{25} (d) Error vector of X_{125}

Figure 12: Plots of residual vector in L^2-norm of e^{At} on $[0, 1]$ for example 4.4.

Table 2: CPU time of examples 4.5 for $m = 12$ in terms of seconds.

	e^{At}	$\cos(At)$	$\sin(At)$
$\lambda = 10000$	4	6	4
$\lambda = 10^8$	4	6	4



(a) Error vector of 1st column of e^{At}

(b) Error vector of 2nd column of e^{At}

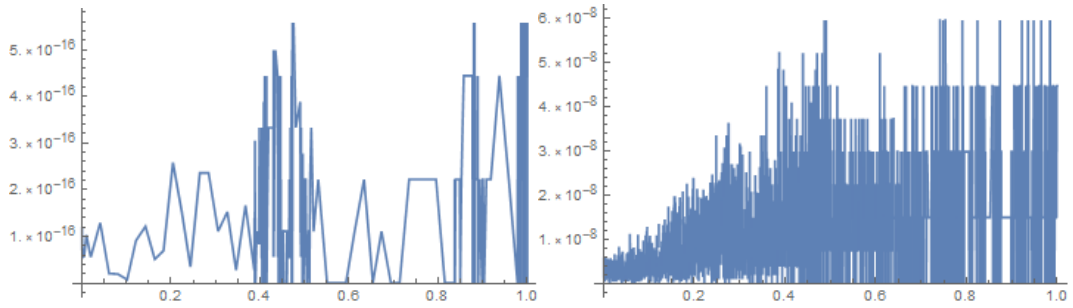
(c) Error vector of 1st column of $\cos(At)$

(d) Error vector of 2nd column of $\cos(At)$

(e) Error vector of 1st column of $\sin(At)$

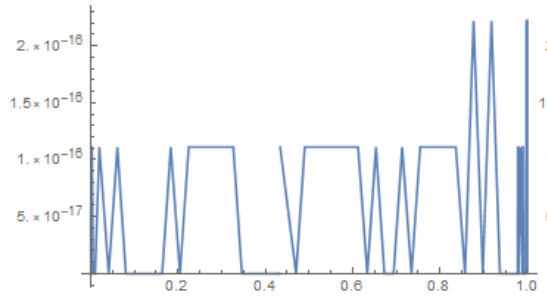
(f) Error vector of 2nd column of $\sin(At)$

Figure 13: 2-norm error vector ith column with $\lambda = 10000$ on $[0, 1]$ for example 4.5.

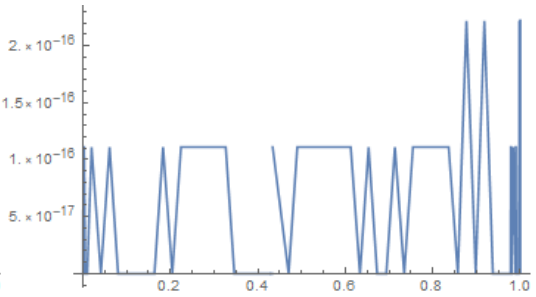


(a) Error vector of 1st column of e^{At}

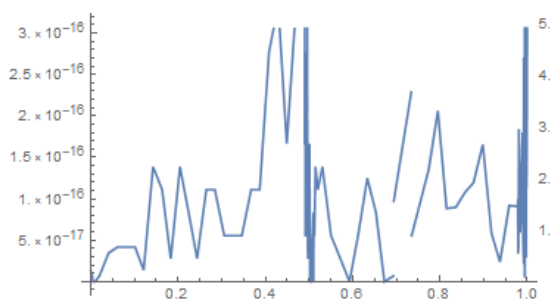
(b) Error vector of 2nd column of e^{At}



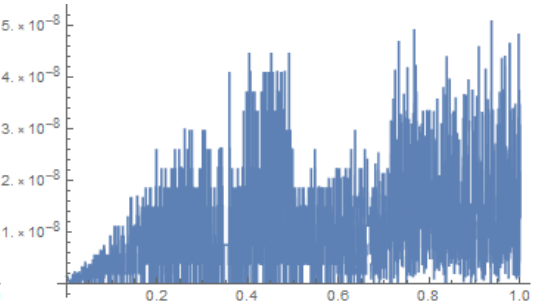
(c) Error vector of 1st column of $\cos(At)$



(d) Error vector of 2nd column of $\cos(At)$



(e) Error vector of 1st column of $\sin(At)$



(f) Error vector of 2nd column of $\sin(At)$

Figure 14: 2-norm error vector ith column with $\lambda = 10^8$ on $[0, 1]$ for example 4.5.

5 Conclusion

The objective of this article is to present the methods for estimating matrix exponential, cosine and sine functions. The proposed method determines the matrix exponential, cosine and sine functions At for $0 \leq t \leq 1$. Some properties of our method are listed as follows:

1. Singularity and eigenvalues of matrix A do not make any restrictions on the implementation of the mentioned method..
2. For the matrix exponential function, we require to solve n set of linear algebraic equations by constant coefficients.
3. For the matrix cosine and sine functions, n constant-coefficient linear algebraic equations systems are needed to solve.
4. It is proved that the upper bound of approximate error is a multiple of $\sqrt{n}$.
5. The presented method can be applied on interval $[0, b], \forall b \in \mathbb{R}$.
6. Approximate matrix functions of matrix At $\forall t \in [0, b]$ can be obtained with only one execution of the method.

References

[1] A.H. Al-Mohy and N.J. Higham, *A new scaling and squaring algorithm for the matrix exponential*, SIAM J. Matrix Anal. Appl. **31** (2010), 970–989.

[2] A.H. Al-Mohy, N.J. Higham and S.D. Relton, *New algorithms for computing the matrix sine and cosine separately or simultaneously*, SIAM J. Sci. Comput. **37** (2015), 56–87.

[3] P. Bader, S. Blanes and F. Casas, *Computing the matrix exponential with an optimized Taylor polynomial approximation*, Math. **7** (2019), 1174.

[4] J.M. Carnicer and J.M. Peña, *Shape preserving representations and optimality of the Bernstein basis*, Adv. Comput. Math. **1** (1993), no. 2, 173–196.

[5] E. Defez and L. Jódar, *Some applications of Hermite matrix polynomials series expansions*, J. Comput. Appl. Math. **99** (1998), 105–117.

[6] E. Defez, J. Sastre, J. IbIbáñez and P. Ruiz, *Computing matrix functions arising in engineering models with orthogonal matrix polynomials*, Math. Comput. Modell. **57** (2013), 1738–1743.

[7] E. Defez, J. Sastre, J. IbIbáñez and P.A. Ruiz, *Computing matrix functions solving coupled differential models*, Math. Comput. Modell. **50** (2009), 831–839.

[8] M. Dehghan and A. Taleei, *Numerical solution of nonlinear Schrödinger equation by using time-space pseudo-spectral method*, Numer. Meth. Part. Differ. Equ. **26** (2010), no. 4, 979–992.

[9] R.T. Farouki, *The Bernstein polynomial basis: A centennial retrospective*, Comput. Aided Geom. Design **29** (2012), 379–419.

[10] N.J. Higham and A.H. Al-Mohy, *Computing matrix functions*, Acta Numer. **19** (2010), 159–208.

[11] J.H. Hubbard and B.H. West, *Differential equations: A dynamical systems approach: Ordinary differential equations*, Springer-Verlag, New York, 2013.

[12] E. Kreyszig, *Introductory functional analysis with applications*,Wiley, New York, 1978.

[13] X. Li and C. Xu, *A space-time spectral method for the time fractional diffusion equation*, SIAM J. Numer. Anal. **47** (2009), 2108–2131.

[14] E. Liz, *Classroom note: A note on the matrix exponential*, SIAM Rev. **40** (1998), 700–702.

[15] I.E. Leonard, *The matrix exponential*, SIAM Rev. **38** (1996), 507–512.

[16] A. Saadatmandi and M. Dehghan, *Numerical solution of hyperbolic telegraph equation using the Chebyshev tau method*, Numer. Meth. Part. Differ. Equ. **26** (2010), 239–252.

[17] J. Sastre, J. IbIbáñez and E. Defez, *Boosting the computation of the matrix exponential*, Appl. Math. Comput. **340** (2019), 206–220.

[18] J. Sastre, J. Ibáñez, P. Alonso-Jordá, J. Peinado and E. Defez, *Fast Taylor polynomial evaluation for the computation of the matrix cosine*, J. Comput. Appl. Math. **354** (2019), 641–650.

[19] S.M. Serbin and S.A. Blalock, *An algorithm for computing the matrix cosine*, SIAM J. Sci. Statist. Comput. **1** (1980), no. 2, 198–204.

[20] M. Seydaoğlu, P. Bader, S. Blanes and F. Casas, *Computing the matrix sine and cosine simultaneously with a reduced number of products*, Appl. Numer. Math. **163** (2021), 96–107.

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

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

[23] J.A. Wood, *The chain rule for matrix exponential functions*, College Math. J. **35** (2004), 220–222.

[24] H.D. Vo and R.B. Sidje, *Approximating the large sparse matrix exponential using incomplete orthogonalization and Krylov subspaces of variable dimension*, Numer. Linear Algebra Appl. **24** (2017), 1–13.

[25] S.A. Yousefi and M. Behroozifar, *Operational matrices of Bernstein polynomials and their applications*, Int. J. Syst. Sci. **41** (2010), 709–716.

[26] S.A. Yousefi, M. Behroozifar and M. Dehghan, *The operational matrices of Bernstein polynomials for solving the parabolic equation subject to specification of the mass*, J. Comput. Appl. Math. **235** (2011), 5272–5283.