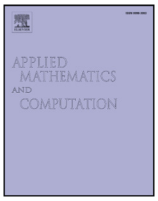
Applied Mathematics and Computation 312 (2017) 66–77

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc



Two algorithms for computing the matrix cosine function[☆]

Jorge Sastre[a], Javier Ibáñez[b,*], Pedro Alonso[c], Jesús Peinado[b], Emilio Defez[d]

[a] Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, España
[b] Instituto de Instrumentación para Imagen Molecular, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, España
[c] Department of Information Systems and Computation, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, España
[d] Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, España

ARTICLE INFO

Keywords:
Matrix cosine
Scaling and recovering method
Taylor series
Forward error analysis
Backward error analysis
MATLAB

ABSTRACT

The computation of matrix trigonometric functions has received remarkable attention in the last decades due to its usefulness in the solution of systems of second order linear differential equations. Several state-of-the-art algorithms have been provided recently for computing these matrix functions. In this work, we present two efficient algorithms based on Taylor series with forward and backward error analysis for computing the matrix cosine. A MATLAB implementation of the algorithms is compared to state-of-the-art algorithms, with excellent performance in both accuracy and cost.

1. Introduction

Many engineering processes are described by second order differential equations, whose solution is given in terms of the trigonometric matrix functions sine and cosine. Examples arise in the spatial semi-discretization of the wave equation or in mechanical systems without damping, where their solutions can be expressed in terms of integrals involving the matrix sine and cosine [1,2]. Several state-of-the-art algorithms have been provided recently for computing these matrix functions using polynomial and rational approximations with scaling and recovering techniques [3–6]. In order to reduce computational costs Paterson–Stockmeyer method [7] is used to evaluate the matrix polynomials arising in these approximations.

In the Taylor algorithm proposed in [4] we used sharp absolute forward error bounds. In the Taylor algorithm proposed in [6] we improved the previous algorithm using relative error bounds based on backward error bounds of the matrix exponentials involved in $\cos(A)$. Those error bounds do not guarantee that the cosine backward error bound in exact arithmetic is less than the unit roundoff in double precision arithmetic [6, Section 2]. However, according to the tests, that algorithm improved the accuracy with respect to the previous Taylor algorithm at the expense of some increase in cost (measured in flops). The algorithm proposed in [6] was also superior in both accuracy and cost to the version of the scaling and recovering Padé state-of-the-art algorithm in [5] not using the Schur decomposition.

Other algorithms based on approximations on L_∞ for normal and nonnegative matrices have been presented recently in [8]. In this work, we focus on general matrices and algorithms using approximations at the origin. We present two

☆ This work has been supported by Spanish Ministerio de Economía y Competitividad and the European Regional Development Fund (ERDF) grant TIN2014-59294-P.

* Corresponding author.
E-mail addresses: jsastrem@upv.es (J. Sastre), jjibanez@dsic.upv.es (J. Ibáñez), palonso@dsic.upv.es (P. Alonso), jpeinado@dsic.upv.es (J. Peinado), edefez@imm.upv.es (E. Defez).

http://dx.doi.org/10.1016/j.amc.2017.05.019
0096-3003/© 2017 Elsevier Inc. All rights reserved.

algorithms based on Taylor series that use Theorem 1 from [4] for computing the matrix cosine. We provide relative forward and backward error analysis for the matrix cosine Taylor approximation that improves even more the comparison to the algorithm in [5] with and without Schur decomposition in both accuracy and cost tests.

Throughout this paper $\mathbb{C}^{n\times n}$ denotes the set of complex matrices of size $n \times n$, I the identity matrix for this set, $\rho(X)$ the spectral radius of matrix X, and $\mathbb{N}$ the set of positive integers. In this work, we use the 1-norm to compute the actual norms. This paper is organized as follows. Section 2 presents a Taylor algorithm for computing the matrix cosine function. Section 3 deals with numerical tests and, finally, Section 4 gives some conclusions.

2. Algorithms for computing matrix cosine

The matrix cosine can be defined for all $A \in \mathbb{C}^{n\times n}$ by

$$\cos(A) = \sum_{i=0}^{\infty} \frac{(-1)^i A^{2i}}{(2i)!},$$

and let

$$T_{2m}(A) = \sum_{i=0}^{m} \frac{(-1)^i B^i}{(2i)!} \equiv P_m(B), \tag{1}$$

be the Taylor approximation of order $2m$ of $\cos(A)$, where $B = A^2$. Since Taylor series are accurate only near the origin, in algorithms that use this approximation the norm of matrix B is reduced by scaling the matrix. Then, a Taylor approximation is computed, and finally the approximation of $\cos(A)$ is recovered by means of the double angle formula $\cos(2X) = 2\cos^2(X) - I$. Algorithm 1 shows a general algorithm for computing the matrix cosine based on Taylor approximation. By using the fact that $\sin(A) = \cos(A - \frac{\pi}{2}I)$, Algorithm 1 also can be easily used to compute the matrix sine.

Algorithm 1 Given a matrix $A \in \mathbb{C}^{n\times n}$, this algorithm computes $C = \cos(A)$ by Taylor series.

```
1: Select adequate values of $m$ and $s$                ▷ Phase I
2: $B = 4^{-s}A^2$
3: $C = P_m(B)$                     ▷ Phase II: Compute Taylor approximation
4: for $i = 1 : s$ do                   ▷ Phase III: Recovering $\cos(A)$
5:     $C = 2C^2 - I$
6: end for
```

In Phase I of Algorithm 1, m and s must be calculated so that the Taylor approximation of the scaled matrix is computed accurately and efficiently. In this phase some powers B^i, $i \geq 2$, are usually computed for estimating m and s and if so they are used in Phase II.

Phase II consists of computing the Taylor approximation (1). For clarity of the exposition we recall some results summarized in [6, Section 2]. Taylor matrix polynomial approximation (1), expressed as $P_m(B) = \sum_{i=0}^{m} p_i B^i$, $B \in \mathbb{C}^{n\times n}$, can be computed with optimal cost by the Paterson–Stockmeyer's method [7] choosing m from the set

$$\mathbb{M} = \{1, 2, 4, 6, 9, 12, 16, 20, 25, 30, 36, 42, \dots\},$$

where the elements of $\mathbb{M}$ are denoted as $m_1, m_2, m_3, \dots$ The algorithm computes first the powers B^i, $2 \leq i \leq q$ not computed in the previous phase, being $q = \lceil \sqrt{m_k} \rceil$ or $q = \lfloor \sqrt{m_k} \rfloor$ an integer divisor of m_k, $k \geq 1$, both values giving the same cost in terms of matrix products. Therefore, (1) can be computed efficiently as

$$\begin{aligned}
P_{m_k}(B) \; = & \\
(((p_{m_k}B^q \; + & \; p_{m_k-1}B^{q-1} + p_{m_k-2}B^{q-2} + \cdots + p_{m_k-q+1}B + p_{m_k-q}I)B^q \\
+ & \; p_{m_k-q-1}B^{q-1} + p_{m_k-q-2}B^{q-2} + \cdots + p_{m_k-2q+1}B + p_{m_k-2q}I)B^q \\
+ & \; p_{m_k-2q-1}B^{q-1} + p_{m_k-2q-2}B^{q-2} + \cdots + p_{m_k-3q+1}B + p_{m_k-3q}I)B^q \\
\cdots & \\
+ & \; p_{q-1}B^{q-1} + p_{q-2}B^{q-2} + \cdots + p_1B + p_0I.
\end{aligned} \tag{2}$$

Table 1 (page 11) shows the values of q for different values of m. From Table 4.1 from [9, p. 74] the cost of computing (1) with (2) is $\Pi_{m_k} = k$ matrix products, $k = 1, 2, \dots$

Finally, Phase III is necessary to obtain the cosine of matrix A from $\cos(4^{-s}B)$ computed previously in Phase II. If m_k is the order used and s is the scaling parameter, then the computational cost of Algorithm 1 is $2(k+s)n^3$ flops, and the storage cost is $(2+q_k)n^2$.

The difficulty of Algorithm 1 is to find appropriate values of m_k and s such that $\cos(A)$ is computed accurately with minimum cost. For that, in the following sections we will use Theorem 1: