Fast Taylor polynomial evaluation for the matrix cosine

Conference Paper · July 2018

5 authors, including:

Jorge Sastre Martínez
Universitat Politècnica de València
77 PUBLICATIONS **524** CITATIONS

Jacinto Javier Ibáñez
Universitat Politècnica de València
51 PUBLICATIONS **204** CITATIONS

Pedro Alonso
Universitat Politècnica de València
96 PUBLICATIONS **392** CITATIONS

Jesús Peinado
Universitat Politècnica de València
33 PUBLICATIONS **90** CITATIONS

Proceedings of the 18th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2018
July 9–14, 2018.

Fast Taylor polynomial evaluation for the matrix cosine

Jorge Sastre[1], Javier Ibáñez[2], Pedro Alonso[3], Jesús Peinado[3] and Emilio Defez[4]

[1] *Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politècnica de València*

[2] *Instituto de Instrumentación para Imagen Molecular, Universitat Politècnica de València*

[3] *Department of Information Systems and Computation, Universitat Politècnica de València*

[4] *Instituto de Matemática Multidisciplinar, Universitat Politècnica de València*

emails: `jsastrem@upv.es`, `jjibanez@dsic.upv.es`, `palonso@upv.es`, `jpeinado@dsic.upv.es`, `edefez@imm.upv.es`

Abstract

In this work we introduce a new method to compute the matrix cosine. It is based on recent new matrix polynomial evaluation methods for the Taylor approximation and forward and backward error analysis. The matrix polynomial evaluation methods allow to evaluate the Taylor polynomial approximation of the cosine function more efficiently than using Paterson-Stockmeyer method. A MATLAB implementation of the new algorithm is provided, giving better efficiency and accuracy than state-of-the-art algorithms.

Key words: matrix, cosine, computation, Taylor, fast matrix polynomial evaluation

1 Introduction

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

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

 ISBN: 978-84-697-7861-6

Algorithm 1 Given a matrix $A \in \mathbb{C}^{n\times n}$ and a maximum order $m_M \in \mathbb{N}$, this algorithm computes $C = \cos(A)$ by a Taylor approximation of order $2m_k \leq 2m_M$.

1: $B = A^2$
2: Scaling phase: Choose $m_k \leqslant m_M$ and an integer scaling parameter s for the Taylor approximation with scaling.
3: $B = B/4^s$
4: Compute $C = P_{m_k}(B)$
5: **for** $i = 1 : s$ **do**
6: $\quad C = 2C^2 - I$
7: **end for**

for which we can consider a Taylor approximation of order $2m$ to obtain the polynomial of order m

$$P_m(\bar{A}) = \sum_{i=0}^{m} p_i \bar{A}^i, \tag{1}$$

where $p_i = \frac{(-1)^i}{(2i)!}$ and $\bar{A} = A^2$.

Algorithm 1 shows a general algorithm for computing the matrix cosine based on Taylor approximation.

2 The proposed Taylor algorithm

So far Step 3 of Algorithm 1 was traditionally carried out by using the Paterson-Stockmeyer's method [5]. In this paper, we propose using the general matrix polynomial evaluation methods from [6] to evaluate $C = P_m(B)$ more efficiently than Paterson-Stockmeyer method. They are based on approximations of orders $m = 8$, 12, and 15. For $m = 1$, 2 and 4, similarly to (10) from [10], the Taylor polynomials $P_m(B)$ can be computed by using the following expressions:

$$\begin{aligned} P_1(B) &= -B/2 + I, \\ P_2(B) &= (B^2/12 - B)/2 + I, \\ P_4(B) &= ((((B^2/56 - B)/30 + I)B^2/12 - B)/2 + I. \end{aligned}$$

Following [6, Ex. 3.1] $P_8(B)$ can be evaluated by using the following formulae:

$$\begin{aligned} y_{02}(B) &= B^2(c_1B^2 + c_2B), \\ P_8(B) &= (y_{02}(B) + c_3B^2 + c_4B)(y_{02}(B) + c_5B^2) \\ &\quad + c_6 y_{02}(B) + B^2/24 - B/2 + I, \end{aligned} \tag{2}$$

ISBN: 978-84-697-7861-6

with a cost of 3 matrix products, being c_i coefficients that can be found in [6, Table 4]. With that cost the maximum approximation order available with Paterson–Stockmeyer is $m = 6$. Following [9, Sec. 3.2], similarly to [6, Ex. 5.1], with a cost of 4 matrix products it is possible to obtain a Taylor based approximation $P_{16}(B)$ of the matrix cosine of order $m = 15$, but it results not very stable according to the stability analysis proposed in [6, Ex. 3.1]. Using (34) and (35) from [6] for a cost of 4 matrix products we can evaluate $P_{12}(B)$ in stable manner, where the maximum approximation order available with Paterson–Stockmeyer is $m = 9$ [6]. Using (34) and (35) from [6] it is possible to evaluate $P_{16}(B)$ with 5 matrix products and several possibilities of real coefficients. In this case the stability check proposed in [6, Ex. 3.1] gives not good enough results. The stability can be improved using expression (52) from [6], with $s = 3$ and $p = 3$, giving a Taylor approximation order of $m = 15$. With the same cost the maximum approximation order available with Paterson–Stockmeyer is $m = 12$. The scaling algorithm is similar to [7, Alg. 1], using a combination of absolute forward error analysis from [8] and relative backward error analysis from [7].

3 Numerical experiments

We compare the new MATLAB function developed in this paper, `cosmpol`, with `cosm`, which is a Matlab function based on the Padé rational approximation for the matrix cosine [1]; and `cosmtay`, which is a code based on the Taylor series evaluated using Paterson-Stockmeyer using norm estimation [7].

As an example, we perform a test consisting of fifteen matrices with dimensions lower than or equal to 128 from the Eigtool MATLAB package [11] and forty four 128×128 real matrices from the function `matrix` of the Matrix Computation Toolbox [3]. We have eliminated the matrices for which we can not calculate the condition number of the matrix cosine function.

The *"exact"* matrix cosine has been computed following [2, Sec. 4.1] using MATLAB symbolic versions of a scaled Padé rational approximation from [1] and a scaled Taylor Paterson-Stockmeyer approximation [7, pp. 67]) both with 4096 decimal digit arithmetic and several orders m and/or scaling parameters s higher than the ones used by `cosm` and `cosmtay`, respectively, checking that their relative difference was small enough. The algorithm accuracy was tested by computing the relative error $\mathrm{E} = \|\cos(A) - \tilde{Y}\|_1 / \|\cos(A)\|_1$, where $\tilde{Y}$ is the computed solution and $\cos(A)$ is the exact solution. We also have used MATLAB function `funm_condest1` to estimate the condition number of the matrix 1-norm.

According to our results, we checked that the number of matrix products is 511, 558, and 673 for algorithms `cosmpol`, `cosmtay`, and `cosm`[1], respectively.

[1]The cost of the resolution of linear systems that appears in the code based on Padé approximations has been calculated as 4/3 products, because from a computational point of view, the cost of that operation compared to the cost of a matrix product is approximately equal to 4/3, see Table C.1 from [4, pp. 336].

 ISBN: 978-84-697-7861-6

Table 1: Relative error comparison of `cosmpol` with `cosm` and `cosmtay`, respectively.

E(`cosmpol`)<E(`cosm`)	71.27%	E(`cosmpol`)<E(`cosmtay`)	50.85%
E(`cosmpol`)>E(`cosm`)	27.73%	E(`cosmpol`)>E(`cosmtay`)	42.37%
E(`cosmpol`)=E(`cosm`)	1.69 %	E(`cosmpol`)=E(`cosmtay`)	6.78%

With regard to accuracy, Table 1 shows the percentage of cases in which the relative errors of `cosmpol` are, respectively, lower than, greater than, or equal to the relative errors of the other algorithms under test. Also, Figure 1 shows the normwise relative errors (a), the Performance Profile (b), and the ratio of relative errors (c), and the ratio of the matrix products (d). In the performance profile, the α coordinate varies between 1 and 5 in steps equal to 0.1, and the p coordinate is the probability that the considered algorithm has a relative error lower than or equal to α-times the smallest error over all methods. The ratios of relative errors are presented in decreasing order with respect to E(`cosmpol`)/E(`cosmtay`) and E(`cosmpol`)/E(`cosm`). The solid lines is the function $k_{\cos}u$, where $k_{\cos}$ is the condition number of matrix cosine function [4, Chapter 3] and $u = 2^{-53}$ is the unit roundoff in the double precision floating-point arithmetic.

In conclusion, we see that all the implementations have a similar numerical stability. Functions based on polynomial approximations are more accurate than the one based on Padé approximants, being the new function `cosmpol` slightly more accurate than `cosmtay`. `cosmpol` has 31.70% and 9.20% lower computational cost than `cosm` and `cosmtay`, respectively.

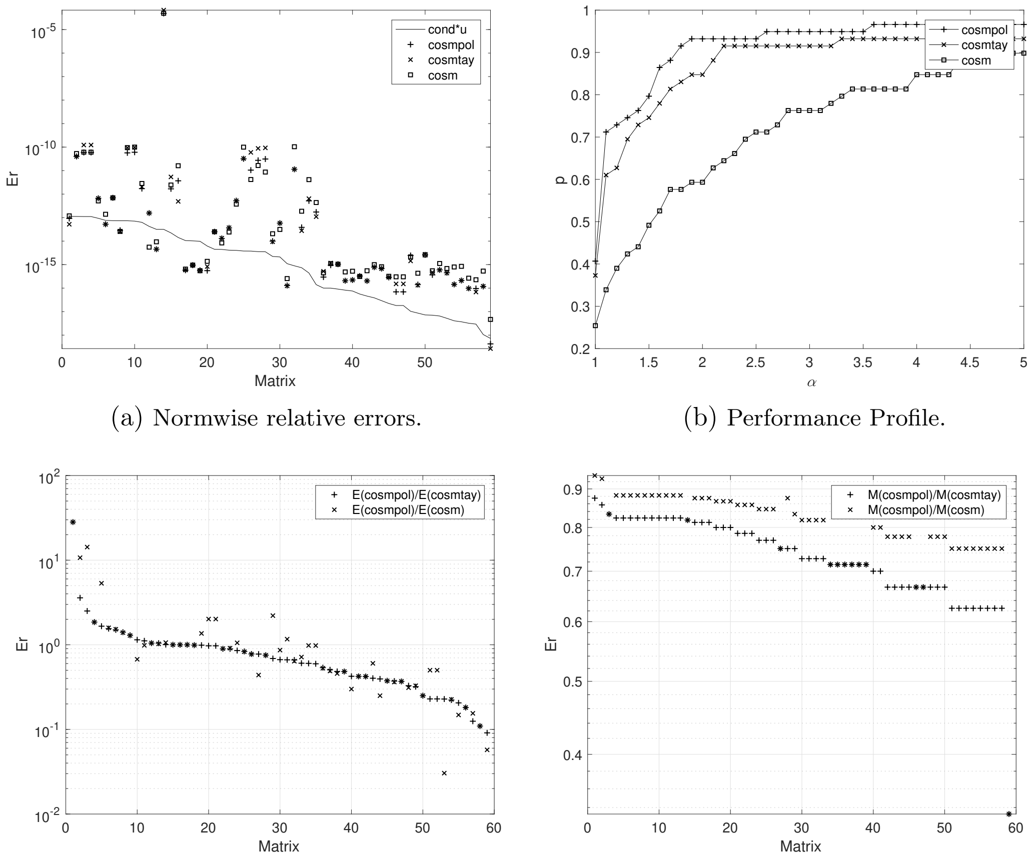
Acknowledgements

This work has been partially supported by Spanish Ministerio de Economía y Competitividad and European Regional Development Fund (ERDF) grants TIN2014-59294-P, TIN2017-89314-P, and TEC2015-67387-C4-1-R.

References

[1] Awad H. Al-Mohy, Nicholas J. Higham, and Samuel D. Relton. New algorithms for computing the matrix sine and cosine separately or simultaneously. *SIAM J. Sci. Comput.*, 37(1):A456–A487, 2015.

[2] P. Alonso, J. Ibáñez, J. Sastre, J. Peinado, and E. Defez. Efficient and accurate algorithms for computing matrix trigonometric functions. *J. Comput. Appl. Math.*, 309:325–332, 2017.



(a) Normwise relative errors.

(b) Performance Profile.

(c) Ratio of relative errors.

(d) Ratio of matrix products.

Figure 1: Relative performance and accuracy of algorithms `cosmpol`, `cosmtay`, and `cosm`.

[3] Nicholas J. Higham. The Test Matrix Toolbox for MATLAB. Numerical Analysis Report No. 237, inst-MCCM, Manchester, England, December 1993.

[4] Nicholas J. Higham. *Functions of Matrices: Theory and Computation.* SIAM, Philadelphia, PA, USA, 2008.

[5] Michael S. Paterson and Larry J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.*, 2(1):60–66, 1973.

[6] J. Sastre. Efficient evaluation of matrix polynomials. *Linear Alg. Appl.*, 539:229–250, 2018.

 ISBN: 978-84-697-7861-6

[7] J. Sastre, J. Ibáñez, P. Alonso, J. Peinado, and E. Defez. Two algorithms for computing the matrix cosine functions. *Appl. Math. Comput.*, 312:66–77, 2017.

[8] J. Sastre, J. Ibáñez, P. Ruiz, and E. Defez. Efficient computation of the matrix cosine. *Appl. Math. Comput.*, 219:7575–7585, 2013.

[9] J. Sastre, Javier J. Ibáñez, and E. Defez. Boosting the computation of the matrix exponential. *Submitted to Appl. Math. Comput.*

[10] J. Sastre, Javier J. Ibáñez, E. Defez, and Pedro A. Ruiz. Accurate matrix exponential computation to solve coupled differential models in engineering. *Math. Comput. Model.*, 54:1835–1840, 2011.

[11] T. G. Wright. Eigtool, version 2.1, 2009.

 ISBN: 978-84-697-7861-6