New matrix series expansions for the matrix cosine approximation*

Emilio Defez⋆, Javier Ibáñez†, José M. Alonso†, Jesús Peinado§, Pedro Alonso-Jordá♮

⋆ Instituto de Matemática Multidisciplinar.

† Instituto de Instrumentación para Imagen Molecular.

§ Departamento de Sistemas Informáticos y Computación.

♮ Grupo Interdisciplinar de Computación y Comunicaciones.

Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia, España.

edefez@imm.upv.es, {jjibanez, jmalonso, jpeinado, palonso,}@dsic.upv.es

1 Introduction and notation

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. Recently, several state-of-the-art algorithms have been provided for computing these matrix functions, see [1–4], in particular for the matrix cosine function.

Among the proposed methods for the approximate computation of the matrix cosine, two fundamental ones stand out: those based on rational approximations [1,5–7], and those related to polynomial approximations, using

*Acknowledgements: This work has been partially supported by Spanish Ministerio de Economía y Competitividad and European Regional Development Fund (ERDF) grants TIN2017-89314-P and by the Programa de Apoyo a la Investigación y Desarrollo 2018 of the Universitat Politècnica de València (PAID-06-18) grants SP20180016.

either Taylor series developments [8,9] or serial developments of Hermite matrix polynomials [10]. In general, polynomial approximations showed to be more efficient than the rational algorithms in tests because they are more accurate despite a slightly higher cost.

Bernoulli polynomials and Bernoulli numbers have been extensively used in several areas of mathematics (an excelent survey about Bernoulli polynomials and its applicacions can be found in [11]).

In this paper, we will present a new series development of the matrix cosine in terms of the Bernoulli matrix polynomials. We are going to verify that its use allows obtaining a new and competitive method for the approximation of the matrix cosine.

The organization of the paper is as follows: In section 2, we will obtain two serial developments of the matrix cosine in terms of the Bernoulli matrix polynomials. In section 3, we will present the different numerical tests performed. Conclusions are given in section 4.

Throughout this paper, we denote by $\mathbb{C}^{r\times r}$ the set of all the complex square matrices of size r. Besides, we denote I as the identity matrix in $\mathbb{C}^{r\times r}$. A polynomial of degree m is given by an expression of the form $P_m(t) = a_m t^m + a_{m-1}t^{m-1} + \cdots + a_1 t + a_0$, where t is a real variable and a_j, for $0 \leq j \leq m$, are complex numbers. Moreover, we can define the matrix polynomial $P_m(B)$ for $B \in \mathbb{C}^{r\times r}$ as $P_m(B) = a_m B^m + a_{m-1}B^{m-1} + \cdots + a_1 B + a_0 I$. As usual, the matrix norm $\|\cdots\|$ denotes any subordinate matrix norm; in particular $\|\cdots\|_1$ is the usual $1-$norm.

2 On Bernoulli matrix polynomials

The Bernoulli polynomials $B_n(x)$ are defined in [12, p.588] as the coefficients of the generating function

$$g(x,t) = \frac{te^{tx}}{e^t - 1} = \sum_{n\geq 0} \frac{B_n(x)}{n!} t^n \ , \ |t| < 2\pi, \tag{1}$$

where $g(x,t)$ is an holomorphic function in $\mathbb{C}$ for the variable t (it has an avoidable singularity in $t = 0$). Bernoulli polynomials $B_n(x)$ has the explicit

expression

$$B_n(x) = \sum_{k=0}^{n} \binom{n}{k} B_k x^{n-k}, \tag{2}$$

where the Bernoulli numbers are defined by $B_n = B_n(0)$. Therefore, it follows that the Bernoulli numbers satisfy

$$\frac{z}{e^z - 1} = \sum_{n\geq 0} \frac{B_n}{n!} z^n \ , \ |z| < 2\pi, \tag{3}$$

where

$$B_0 = 1, B_k = -\sum_{i=0}^{k-1} \binom{k}{i} \frac{B_i}{k+1-i}, k \geq 1. \tag{4}$$

Note that $B_3 = B_5 = \cdots = B_{2k+1} = 0$, for $k \geq 1$. For a matrix $A \in \mathbb{C}^{r\times r}$, we define the $m-th$ Bernoulli matrix polynomial by the expression

$$B_m(A) = \sum_{k=0}^{m} \binom{m}{k} B_k A^{m-k}. \tag{5}$$

We can use the series expansion

$$e^{At} = \left(\frac{e^t - 1}{t}\right) \sum_{n\geq 0} \frac{B_n(A) t^n}{n!} \ , \ |t| < 2\pi, \tag{6}$$

to obtain approximations of the matrix exponential. A method based in (6) to approximate the exponential matrix has been presented in [13].

From (6), we obtain the following expression for the matrix cosine and sine:

$$\left.\begin{aligned} \cos(A) &= (\cos(1) - 1) \sum_{n\geq 0} \frac{(-1)^n B_{2n+1}(A)}{(2n+1)!} + \sin(1) \sum_{n\geq 0} \frac{(-1)^n B_{2n}(A)}{(2n)!}, \\ \sin(A) &= \sin(1) \sum_{n\geq 0} \frac{(-1)^n B_{2n+1}(A)}{(2n+1)!} - (\cos(1) - 1) \sum_{n\geq 0} \frac{(-1)^n B_{2n}(A)}{(2n)!}. \end{aligned}\right\} \tag{7}$$

Note that unlike the Taylor (and Hermite) polynomials that are even or odd, depending on the parity of the polynomial degree n, the Bernoulli polynomials do not verify this property. Thus, in the development of $\cos(A)$ and $\sin(A)$, all Bernoulli polynomials are needed (and not just the even-numbered ones).

Replacing in (6) the value t for it and $-it$ respectively and taking the arithmetic mean, we obtain the expression

$$\sum_{n\geq 0} \frac{(-1)^n B_{2n}(A)}{(2n)!} t^{2n} = \frac{t}{2\sin\left(\frac{t}{2}\right)} \left(\cos\left(tA - \frac{t}{2}I\right)\right) \ , \ |t| < 2\pi. \tag{8}$$

Taking $t = 2$ in (8) it follows that

$$\cos(A) = \sin(1) \sum_{n\geq 0} \frac{(-1)^n 2^{2n} B_{2n}\left(\frac{A+I}{2}\right)}{(2n)!}, \tag{9}$$

Note that in formula (9) only even grade Bernoulli's polynomials appear.

3 Numerical Experiments

Having in mind expressions (7) and (9), two different approximations are given to compute cosine matrix function.

To test the proposed method and the two distinct approximations, and to compare them with other approaches, the following algorithms have been implemented on MATLAB R2018b:

- *cosmber*. New code based on the new developments of Bernoulli matrix polynomials (formulae (7) and (9)). The maximum value of m to be used is $m = 36$, with even and odd terms.
- *cosmtay*. Code based on the Taylor series for the cosine [8]. It will provide a maximum value of $m = 16$, considering only the even terms, which would be equivalent to $m = 32$ using the even and odd terms.
- *cosmtayher*. Code based on the Hermite series for the cosine [10]. As mentioned before, it will provide a maximum value of $m = 16$.

- *cosm*. Code based on the Padé rational approximation for the cosine [7].

The following sets of matrices have been used:

a) Diagonalizable matrices. The matrices have been obtained as $A = V \cdot D \cdot V^T$, where D is a diagonal matrix (with complex or real values) and matrix V is an orthogonal matrix, $V = H/16$, where H is a Hadamard matrix. We have $2.18 \leq \|A\|_1 \leq 207.52$. The matrix cosine is exactly calculated as $\cos(A) = V \cdot \cos(D) \cdot V^T$.

b) Non-diagonalizables matrices. The matrices have been computed as $A = V \cdot J \cdot V^{-1}$, where J is a Jordan matrix with complex eigenvalues with module less than 10 and random algebraic multiplicity between 1 and 5. Matrix V is a random matrix with elements in the interval $[-0.5, 0.5]$. We have $1279.16 \leq \|A\|_1 \leq 87886.4$. The matrix cosine is exactly calculated as $\cos(A) = V \cdot \cos(J) \cdot V^{-1}$.

c) Matrices from the Matrix Computation Toolbox [14] and from the **Eigtool Matlab package** [15]. These matrices have been chosen because they have more varied and significant characteristics.

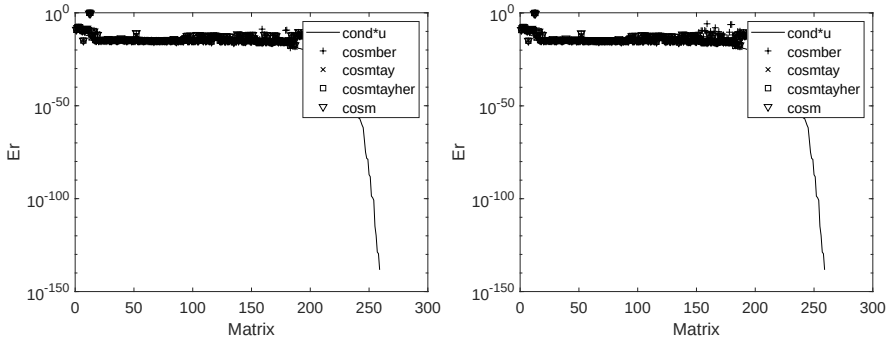
In the numerical test, we used 259 matrices of size 128×128: 100 from the diagonalizable set, 100 from the non-diagonalizable set, 42 from Matrix Computation Toolbox and 17 from Eigtool Matlab package. Results are given in Tables 1 and 2. The rows of each table show the percentage of cases in which the relative errors of `cosmber` (Bernoulli) is lower, greater or equal than the relative errors of `cosmtay` (Taylor), `cosmtayher` (Hermite) and `cosm` (Padé). Graphics of the Normwise relative errors and the Performance Profile are given in Figures 1 and 2. The total number of matrix products was: 3202 (*cosmber*), 2391 (*cosmtay*), 1782 (*cosmtayher*) and 3016 (*cosm*). Recall that in the Bernoulli implementation, the maximum value of m to be used was $m = 36$ considering all the terms and, in the rest of algorithms, was $m = 32$ but just having into account the even terms.

Table 1: Using approximation (7)

$E(cosmber) < E(cosmtay)$	58.30%
$E(cosmber) > E(cosmtay)$	41.70%
$E(cosmber) = E(cosmtay)$	0%
$E(cosmber) < E(cosmtayher)$	52.90%
$E(cosmber) > E(cosmtayher)$	47.10%
$E(cosmber) = E(cosmtayher)$	0%
$E(cosmber) < E(cosm)$	77.99%
$E(cosmber) > E(cosm)$	22.01%
$E(cosmber) = E(cosm)$	0%

Table 2: Using approximation (9)

$E(cosmber) < E(cosmtay)$	66.02%
$E(cosmber) > E(cosmtay)$	33.98%
$E(cosmber) = E(cosmtay)$	0%
$E(cosmber) < E(cosmtayher)$	60.23%
$E(cosmber) > E(cosmtayher)$	39.77%
$E(cosmber) = E(cosmtayher)$	0%
$E(cosmber) < E(cosm)$	76.06%
$E(cosmber) > E(cosm)$	23.94%
$E(cosmber) = E(cosm)$	0%

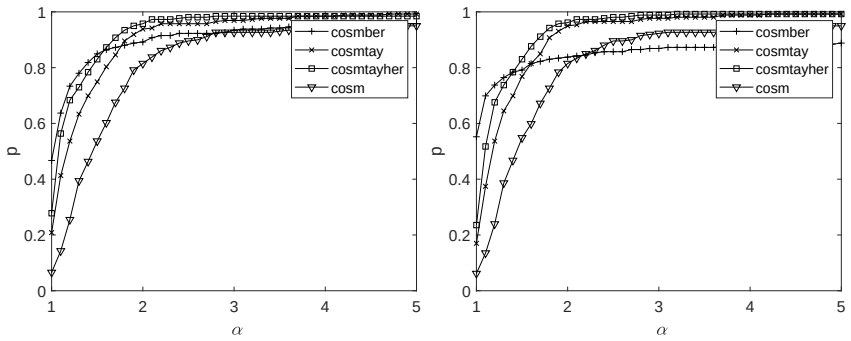


(a) Using formula (7) (b) Using formula (9)

Figure 1: Normwise relative errors.

4 Conclusions

In general, the implementation based on the new Bernoulli series (9) is more accurate than (7), comparing it with the one based on the Taylor series, algorithm (`cosmtay`) and Hermite series, algorithm (`cosmtayher`), and the one based in Padé rational approximation, algorithm (`cosm`).



(a) Using formula (7) (b) Using formula (9)

Figure 2: Performance Profile.

References

[1] Steven M. Serbin and Sybil A. Blalock. An algorithm for computing the matrix cosine. *SIAM Journal on Scientific Computing*, 1(2):198–204, 1980.

[2] Mehdi Dehghan and Masoud Hajarian. Computing matrix functions using mixed interpolation methods. *Mathematical and Computer Modelling*, 52(5-6):826–836, 2010.

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

[4] Pedro Alonso-Jordá, Jesús Peinado, Javier Ibáñez, Jorge Sastre, and Emilio Defez. Computing matrix trigonometric functions with GPUs through Matlab. *The Journal of Supercomputing*, pages 1–14, 2018.

[5] Ch Tsitouras and Vasilios N Katsikis. Bounds for variable degree rational L_∞ approximations to the matrix cosine. *Computer Physics Communications*, 185(11):2834–2840, 2014.

[6] S.M. Serbin. Rational approximations of trigonometric matrices with application to second-order systems of differential equations. *Applied Mathematics and Computation*, 5(1):75–92, 1979.

[7] Awad H. Al-Mohy, Nicholas J. Higham, and Samuel D. Relton. New algorithms for computing the matrix sine and cosine separately or simul-

taneously. *SIAM Journal on Scientific Computing*, 37(1):A456–A487, 2015.

[8] J. Sastre, J. Ibáñez, P. Alonso-Jordá, J. Peinado, and E. Defez. Two algorithms for computing the matrix cosine function. *Applied Mathematics and Computation*, 312:66–77, 2017.

[9] Jorge Sastre, Javier Ibáñez, Pedro Alonso-Jordá, Jesús Peinado, and Emilio Defez. Fast Taylor polynomial evaluation for the computation of the matrix cosine. *Journal of Computational and Applied Mathematics*, 354:641–650, 2019.

[10] Emilio Defez, Javier Ibáñez, Jesús Peinado, Jorge Sastre, and Pedro Alonso-Jordá. An efficient and accurate algorithm for computing the matrix cosine based on new Hermite approximations. *Journal of Computational and Applied Mathematics*, 348:1–13, 2019.

[11] Omran Kouba. Lecture Notes, Bernoulli Polynomials and Applications. *arXiv preprint arXiv:1309.7560*, 2013.

[12] Frank WJ Olver, Daniel W Lozier, Ronald F Boisvert, and Charles W Clark. *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge University Press, 2010.

[13] Emilio Defez, Javier Ibáñez, Jesús Peinado, Pedro Alonso-Jordá, and José M. Alonso. Computing matrix functions by matrix Bernoulli series. In *19th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE-2019)*, From 30th of Juny to 6th of July 2019. Poster presented at some conference, Rota (Cádiz), Spain.

[14] N. J. Higham. *The test matrix toolbox for MATLAB (Version 3.0)*. University of Manchester Manchester, 1995.

[15] TG Wright. Eigtool, version 2.1. *URL: web. comlab. ox. ac. uk/pseudospectra/eigtool*, 2009.