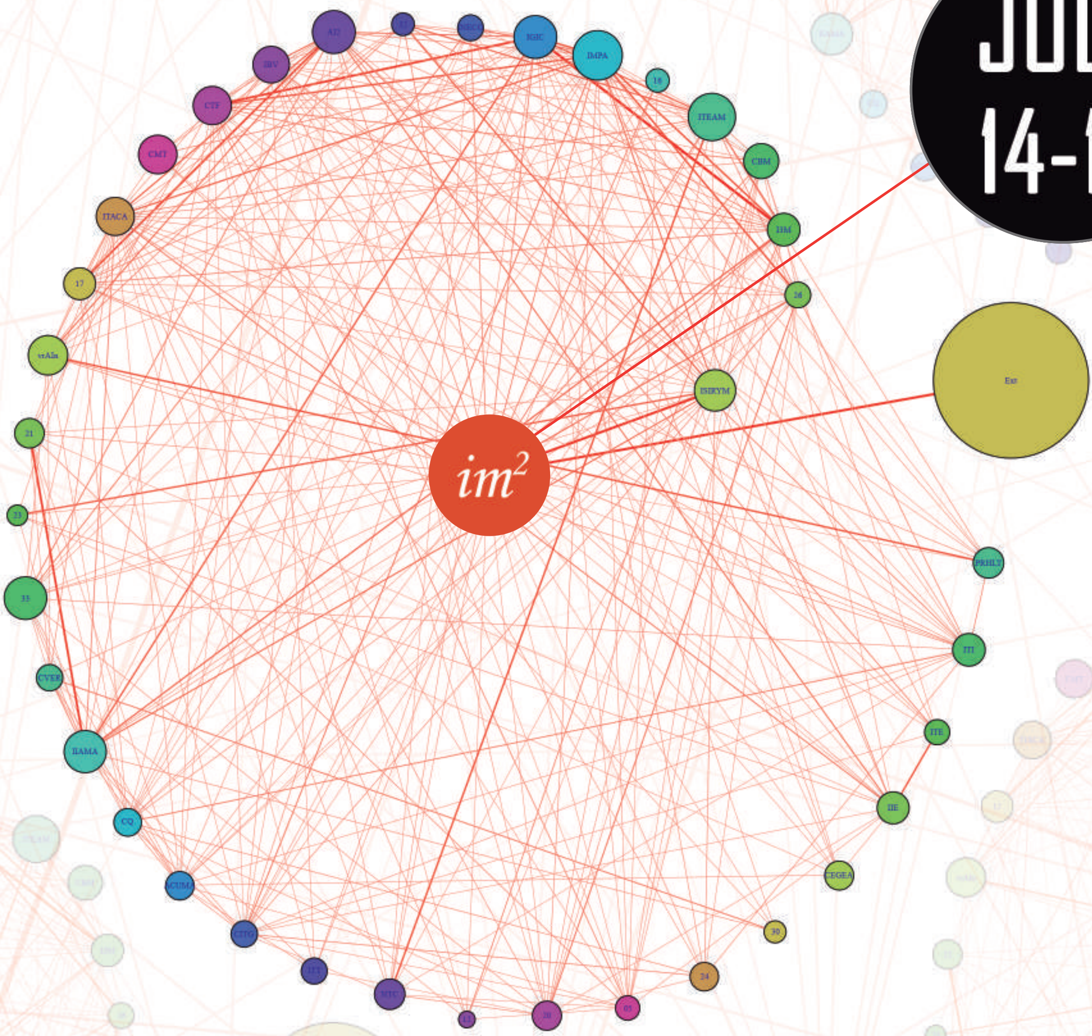
MODELLING FOR ENGINEERING & HUMAN BEHAVIOUR 2021

JULY 14-16



Edited by

Juan Ramón Torregrosa Juan Carlos Cortés Antonio Hervás Antoni Vidal Elena López-Navarro

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

im2 Instituto Universitario de Matemática Multidisciplinar

Modelling for Engineering & Human Behaviour 2021

València, July 14th-16th, 2021

This book includes the extended abstracts of papers presented at XXIII Edition of the Mathematical Modelling Conference Series at the Institute for Multidisciplinary Mathematics *Mathematical Modelling in Engineering & Human Behaviour*.

I.S.B.N.: 978-84-09-36287-5

November 30th, 2021
Report any problems with this document to imm@imm.upv.es.

Edited by: I.U. de Matemàtica Multidisciplinar, Universitat Politècnica de València.
J.R. Torregrosa, J-C. Cortés, J. A. Hervás, A. Vidal-Ferràndiz and E. López-Navarro

im^2
Instituto Universitario
de Matemática Multidisciplinar

Contents

12

Accurate approximation of the Hyperbolic matrix cosine using Bernoulli matrix polynomials

E. Defez⋆,[1] J.J. Ibáñez⋆ J.M. Alonso♮ J. Peinado♭ and J. Sastre∗

(⋆) Instituto Universitario de Matemática Multidisciplinar,
(♮) Instituto de Instrumentación para Imagen Molecular,
(♭) Departamento de Sistemas Informáticos y Computación,
(∗) Instituto de Telecomunicaciones y Aplicaciones Multimedia,
Universitat Politècnica de València
Camí de Vera s/n, Valencia, Spain.

1 Introduction and motivation

The evaluation of matrix functions plays an important and relevant role in many scientific applications because matrix functions have proven to be an efficient tool in applications such as reduced order models [1], [2, pp. 275–303], image denoising [3] and graph neural networks [4], among others.
Among the different matrix functions, we must highlight hyperbolic matrix functions. The computation of the hyperbolic matrix functions has received remarkable attention in the last decades due to its usefulness in the solution of systems of partial differential problems, see references [5,6] for example. For this reason, several algorithms have been provided recently for computing these matrix functions, looking for high precision in the approximation and economy of computational cost, see [7, pp.403–407], [8–11] and references therein.
Also, the generalizations of some known classical special functions into matrix framework are important both from the theoretical and applied point of view. These new extensions (Laguerre, Hermite, Chebyshev, Jacobi matrix polynomials, etc.) have proved to be very useful in various fields such as physics, engineering, statistics and telecommunications. Recently, Bernoulli polynomials $B_n(x)$, who are defined in [12] as the coefficients of the generating function

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

and that have the explicit expression for $B_n(x)$

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

where the *Bernoulli numbers* are defined by $\mathcal{B}_n = B_n(0)$, satisfying the explicit recurrence

$$\mathcal{B}_0 = 1, \mathcal{B}_k = -\sum_{i=0}^{k-1} \frac{\mathcal{B}_i}{k+1-i}, k \geq 1. \tag{3}$$

[1]edefez@imm.upv.es

have been generalized to the matrix framework in [13]: For a matrix $A \in \mathbb{C}^{r\times r}$, the nth Bernoulli matrix polynomial it is defined by the expression

$$B_n(A) = \sum_{k=0}^{n} \binom{n}{k} \mathcal{B}_k A^{n-k}. \tag{4}$$

This matrix polynomials have the series expansion

$$e^{At} = \left(\frac{e^t - 1}{t}\right) \sum_{n\geq 0} \frac{B_n(A)t^n}{n!}\ ,\ |t| < 2\pi. \tag{5}$$

To obtain *practical approximations of the exponential matrix* using the expansion (5), let's take "s" as the scaling of the matrix A and take the degree of the approximation "m", and then

$$e^{A2^{-s}} \approx (e-1) \sum_{n=0}^{m} \frac{B_n\left(A2^{-s}\right)}{n!}. \tag{6}$$

The use of expansion (5) to approximate matrix exponential with good results of precision and computational cost can be found in [13]. For a matrix $A \in \mathbb{C}^{r\times r}$, using expression (5) we obtain

$$\cosh(A) = \sinh(1) \sum_{n\geq 0} \frac{B_{2n}(A)}{(2n)!} + (\cosh(1) - 1) \sum_{n\geq 0} \frac{B_{2n+1}(A)}{(2n+1)!}. \tag{7}$$

Note that unlike the Taylor (and Hermite) polynomials that are even or odd, depending on the parity of the polynomial degree n, the Bernoulli matrix polynomials do not verify this property, so in the development of $\cosh(A)$ all Bernoulli polynomials are needed (and not just the even-numbered). We can also obtain, for $C \in \mathbb{C}^{r\times r}$, the expression:

$$\cosh(C) = \sinh(1) \sum_{n\geq 0} \frac{2^{2n} B_{2n}\left(\frac{1}{2}(C+I)\right)}{(2n)!}. \tag{8}$$

The objective of this work is to present algorithms based on the approximations (7) and (8) for the matrix hyperbolic cosine, trying to choose the most precise and with the lowest computational cost.

2 The proposed Algorithms

From (7) one gets the approximation

$$\cosh(A) \approx \sinh(1) \sum_{n=0}^{m} \frac{B_{2n}(A)}{(2n)!} + (\cosh(1) - 1) \sum_{n=0}^{m} \frac{B_{2n+1}(A)}{(2n+1)!}, \tag{9}$$

and from (8) one gets the alternative approximation

$$\cosh(C) \approx \sinh(1) \sum_{n=0}^{m} \frac{2^{2n} B_{2n}\left(\frac{1}{2}(C+I)\right)}{(2n)!}. \tag{10}$$

We are going to try to compare algorithms based on the approximations in practice (9)-(10). As different algorithms are going to be used, we will to establish the following identification code denoted by *coshmber_x_y*, where the argument is chosen according to the following criteria:

- We denote $x = 1$ if we use directly formula (9).

Numerical test 1			
$E(coshmber\_1\_3) < E(coshmber\_1\_4)$	1.23%	$E(coshmber\_1\_3) < E(coshmber\_1\_5)$	0.61%
$E(coshmber\_1\_3) > E(coshmber\_1\_4)$	40.49%	$E(coshmber\_1\_3) > E(coshmber\_1\_5)$	0.00%
$E(coshmber\_1\_3) = E(coshmber\_1\_4)$	58.28%	$E(coshmber\_1\_3) = E(coshmber\_1\_5)$	99.39%

Table 1: Errors in test 1

- We denote $x = 2$ if we use directly formula (10).
- We use $x = 3$ if formula (10) is used, but terms with odd powers have been removed.

By other hand, we have the argument $y \in \{3, 4, 5\}$, it is chosen according to the following criteria:

- We denote $y = 3$ if the evaluation of m and s use a norm estimation, similar to the given in reference [14].
- We denote $y = 4$ if the evaluation of m and s use other algorithm for the norm estimation, see reference [14] for more details.
- We denote $y = 5$ if the evaluation of m and s is made without norm estimation (calculating the norms), see [14].

Our algorithm has been compared with algorithm *funmcosh*. This functions is *funm* MATLAB function to compute matrix functions, such as the matrix hyperbolic cosine. All computations was implemented on MATLAB 2020b.

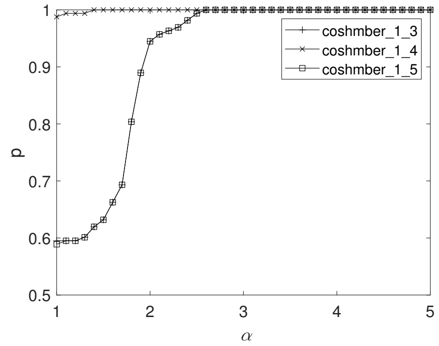
Matrices and numerical test

For the numerical experiments a set of 153 test matrices matrices has been selected: 60 diagonalizable (Hadamard matrices), 60 non-diagonalizable, 39 from toolbox [15] and 13 from Eigtool [16]. Size 128×128. We have performed a series of experiments to determine the best algorithm choice. First we carry out the following tests:

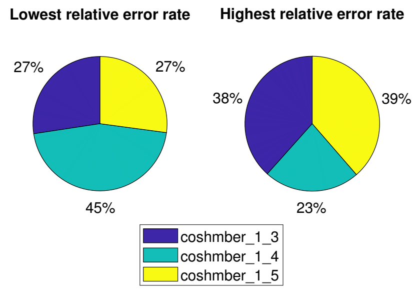
- test 1: we compare each $coshmber\_1\_3$, $coshmber\_1\_4$, $coshmber\_1\_5$.
- test 2: we compare each $coshmber\_2\_3$, $coshmber\_2\_4$, $coshmber\_2\_5$.
- test 3: we compare each $coshmber\_3\_3$, $coshmber\_3\_4$, $coshmber\_3\_5$.

Analysis of results of test 1

We compare algorithms $coshmber\_1\_3$, $coshmber\_1\_4$, $coshmber\_1\_5$, obtaining the following table 1 of results. With respect the computational cost, the total number of matrix products of each algorithm was: $coshmber\_1\_3$ (1940), $coshmber\_1\_4$ (1872) and $coshmber\_1\_5$ (1939).
Among the three proposed algorithms ($coshmber\_1\_3$, $coshmber\_1\_4$, $coshmber\_1\_5$) we choose algorithm $coshmber\_1\_4$ because $E(coshmber\_1\_3) > E(coshmber\_1\_4)$ in the 40.49% and the number of matrix products is 1872, therefore, this algorithm $coshmber\_1\_4$ has the lowest computational cost. Regarding errors, algorithms $coshmber\_1\_3$ and $coshmber\_1\_5$ are practically the same.



(a) Profile test 1.



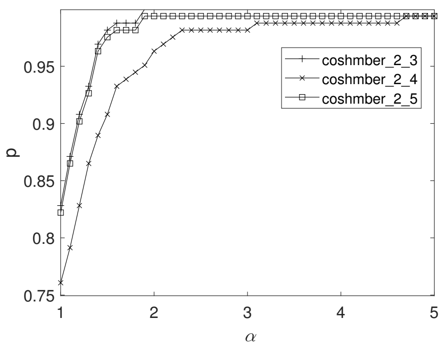
(b) Pie charts Test 1.

Numerical test 2			
$E(coshmber\_2\_3) < E(coshmber\_2\_4)$	23.93%	$E(coshmber\_2\_3) < E(coshmber\_2\_5)$	0.61%
$E(coshmber\_2\_3) > E(coshmber\_2\_4)$	17.18%	$E(coshmber\_2\_3) > E(coshmber\_2\_5)$	0.00%
$E(coshmber\_2\_3) = E(coshmber\_2\_4)$	58.90%	$E(coshmber\_2\_3) = E(coshmber\_2\_5)$	99.39%

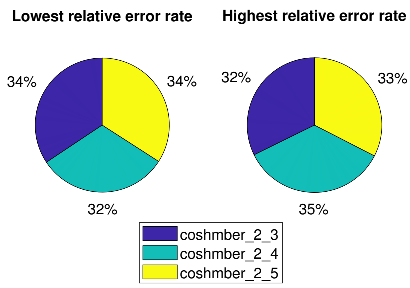
Table 2: Errors in test 2

Analysis of results of test 2

We compare algorithms *coshmber_2_3*, *coshmber_2_4*, *coshmber_2_5*, obtaining the table 2 of results. With respect the computational cost, the total number of matrix products of each algorithm was: *coshmber_2_3* (1940), *coshmber_2_4* (1872) and *coshmber_2_5* (1939).



(a) Profile test 2.



(b) Pie charts Test 2.

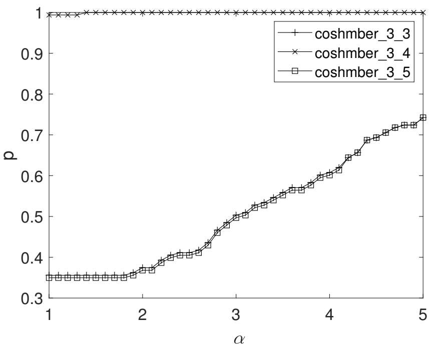
Among the three proposed algorithms (*coshmber_2_3*, *coshmber_2_4*, *coshmber_2_5*) we choose algorithm *coshmber_2_3* because $E(coshmber\_2\_3) < E(coshmber\_2\_4)$ in the 23.93% despite the fact that it has a higher computational cost (the number of matrix products is 1940). Regarding errors, algorithms *coshmber_2_3* and *coshmber_2_5* are practically the same.

Analysis of results of test 3

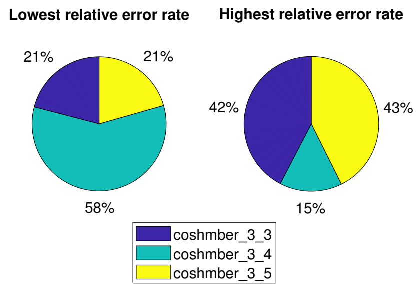
We compare algorithms *coshmber_2_3*, *coshmber_2_4*, *coshmber_2_5*, obtaining the table 3 of results. With respect the computational cost, the total number of matrix products of each algorithm was: *coshmber_3_3* (1435), *coshmber_3_4* (1336) and *coshmber_3_5* (1325).

Numerical test 3			
$E(coshmber\_3\_3) < E(coshmber\_3\_4)$	0.61%	$E(coshmber\_3\_3) < E(coshmber\_3\_5)$	0.61%
$E(coshmber\_3\_3) > E(coshmber\_3\_4)$	64.42%	$E(coshmber\_3\_3) > E(coshmber\_3\_5)$	0.00%
$E(coshmber\_3\_3) = E(coshmber\_3\_4)$	34.97%	$E(coshmber\_3\_3) = E(coshmber\_3\_5)$	99.39%

Table 3: Errors in test 3



(a) Profile test 3.

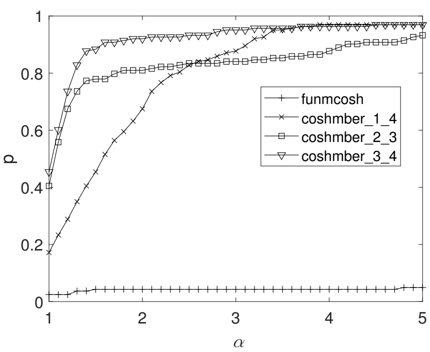


(b) Pie charts Test 3.

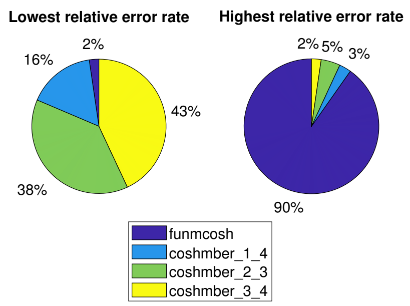
Among the three proposed algorithms (*coshmber*_3_3, *coshmber*_3_4, *coshmber*_3_5) we choose algorithm *coshmber*_3_4 because $E(coshmber\_3\_3) > E(coshmber\_3\_4)$ in the 64.42% and has a lower computational cost (the number of matrix products is 1336). Regarding errors, algorithms *coshmber*_3_3 and *coshmber*_2_5 are practically the same.

Analysis of results with MATLAB function *funmcosh* (Numerical test 4)

Finally, we will compare the selected algorithms *coshmber*_1_4, *coshmber*_2_3, *coshmber*_3_4 and the MATLAB function *funmcosh*, see Table 4. With respect the computational cost, the total number of matrix products of each algorithm was: *funmcosh*: (2282), *coshmber*_1_4 (1872), *coshmber*_2_3 (1940) and *coshmber*_3_4 (1336).



(a) Profile test 4.



(b) Pie charts Test 4.

In general, the relative error improvements over the MATLAB function *funmcosh* exceed 94% in all cases. Between algorithms *coshmber*_1_4, *coshmber*_2_3, *coshmber*_3_4, we choose algorithm *coshmber*_3_4 because it has a lower computational cost (the number of total matrix products is 1336).

Numerical test 4	
$E(funmcosh) < E(coshmber\_1\_4)$	1.84%
$E(funmcosh) > E(coshmber\_1\_4)$	96.32%
$E(funmcosh) = E(coshmber\_1\_4)$	1.84%
$E(funmcosh) < E(coshmber\_2\_3)$	3.68%
$E(funmcosh) > E(coshmber\_2\_3)$	94.48%
$E(funmcosh) = E(coshmber\_2\_3)$	1.84%
$E(funmcosh) < E(coshmber\_3\_4)$	0.61%
$E(funmcosh) > E(coshmber\_3\_4)$	97.55%
$E(funmcosh) = E(coshmber\_3\_4)$	1.84%

Table 4: Errors in test 4

3 Conclusions

In this work, different variations of algorithms have been presented to calculate the matrix hyperbolic cosine based on new Bernoulli matrix polynomials series expansions (7) and (8). These algorithms have been tested on a battery of test matrices in order to select the best variants, both in terms of computational cost as in terms of error in the approximation. The best selection (algorithm $coshmber\_3\_4$) is based in formula (10), but terms with odd powers have been removed, and in the evaluation of m and s which use the algorithm for the norm estimation given in reference [14].

References

[1] V. Druskin, A. V. Mamonov, M. Zaslavsky, Multiscale s-fraction reduced-order models for massive wavefield simulations *Multiscale Modeling & Simulation*, 15(1):445–475, 2017.

[2] A. Frommer, V. Simoncini, Matrix functions. In Model order reduction: theory, research aspects and applications, Springer, New York (USA), 2008.

[3] V. May, Y. Keller, N. Sharon, Y. Shkolnisky, An algorithm for improving nonlocal means operators via low-rank approximation *IEEE Transactions on Image Processing*, 25(3):1340–1353, 2016.

[4] R. Levie, F. Monti, X. Bresson, M. M. Bronstein, Cayleynets: Graph convolutional neural networks with complex rational spectral filters *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[5] L. Jódar, E. Navarro, J. Martín, Exact and analytic-numerical solutions of strongly coupled mixed diffusion problems *Proceedings of the Edinburgh Mathematical Society*, 43:269–293, 2000.

[6] L. Jódar, E. Navarro, A. Posso, M. Casabán, Constructive solution of strongly coupled continuous hyperbolic mixed problems *Applied Numerical Mathematics*, 47(3):477–492, 2003.

[7] M.Fontes, M.Günther, N.Marheineke (Eds.), Progress in Industrial Mathematics at ECMI 2012, Mathematics in Industry, vol.19, Springer-Verlag, Berlin/Heidelberg, 2014.

[8] E. Defez,J. Sastre,J. Ibáñez, J. Peinado, M.Tung, A method to approximate the hyperbolic sine of a matrix *International Journal of Complex Systems in Science*, 4(1):41–45. 2014.

[9] E. Defez, J. Sastre, J. Ibanez, J. Peinado, Solving engineering models using hyperbolic matrix functions *Applied Mathematical Modelling*, 40(4):2837–2844. 2016.

[10] N. Higham, P. Kandolf, Computing the action of trigonometric and hyperbolic matrix functions *SIAM Journal on Scientific Computing*, 39(2):A613–A627, 2017.

[11] A. H. Al-Mohy, A Truncated Taylor Series Algorithm for Computing the Action of Trigonometric and Hyperbolic Matrix Functions *SIAM Journal on Scientific Computing*, 40(3):A1696–A1713. 2018.

[12] F. W. Olver, D. W. Lozier, R. F. Boisvert, C. W. Clark, NIST handbook of mathematical functions hardback and CD-ROM. Cambridge University Press, 2010.

[13] E. Defez, J. Ibáñez, P. Alonso-Jordá, J.M. Alonso, J. Peinado, On Bernoulli matrix polynomials and matrix exponential approximation *Journal of Computational and Applied Mathematics*, In Press, 2020.

[14] E. Defez, J. Ibáñez, J.M. Alonso, P. Alonso-Jordá, On Bernoulli series approximation for the matrix cosine *Mathematical Methods in the Applied Sciences*, In Press, 2020.

[15] N.J. Higham, The Test Matrix Toolbox for MATLAB *Numerical Analysis Report No. 237*, The University of Manchester, England, 1993.

[16] T.G. Wright, Eigtool, Version 2.1, 16, March 2009. *Available online at: http://www.comlab.ox.ac.uk/pseudospectra/eigtool/*.