Elsevier Editorial System(tm) for Journal of Computational and Applied Mathematics
Manuscript Draft

Manuscript Number: CAM-D-19-01467

Title: New Hermite series expansion for computing the matrix hyperbolic cosine

Article Type: Research Paper

Section/Category: 65Yxx (Computer aspects of numerical algorithms)

Keywords: Hermite matrix approximation; Matrix hyperbolic cosine; Error analysis; GPU computing

Corresponding Author: Dr. Emilio Defez,

Corresponding Author's Institution:

First Author: Emilio Defez

Order of Authors: Emilio Defez; J. Javier Ibañez; Jesús Peinado; P. Alonso-Jordá; José M. Alonso

Abstract: There are, currently, very few implementations to compute the hyperbolic cosine of a matrix. This work tries to fill this gap. To this end, we first introduce both a new rational-polynomial Hermite matrix expansions and a formula for the forward relative error of Hermite approximation in exact arithmetic with a sharp bound for forward error. This matrix expansion allows to obtain a new accurate and efficient method for computing the hyperbolic matrix cosine. We present a MATLAB implementation based on this method, which shows a superior efficiency and a similar or better accuracy than other state-of-the-art methods. The algorithm developed on the basis of this method is also able to run on an NVIDIA GPU thanks to a MEX file that connects the Matlab script to the CUDA code.

HIGHLIGHTS

New Hermite series expansion for computing the matrix hyperbolic function

E. Defez, J. Ibáñez, J. Peinado, P. Alonso, José M. Alonso

June 27, 2019

Highlights:

- A new rational-polynomial Hermite matrix expansions have been developed for computing the matrix hyperbolic cosine.
- The proposed numerical algorithm shows a far superior efficiency, and higher accuracy that the given in the literature.
- A sequential MATLAB code has been implemented.
- A NVIDIA GPUs code has been implemented, which demonstrates its great computational capacity.

New Hermite series expansion for computing the matrix hyperbolic cosine

E. Defez[a,*], J. Ibáñez[b], J. Peinado[c], P. Alonso-Jordá[c], José M. Alonso[b]

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

[a]*Instituto de Matemática Multidisciplinar*
[b]*Instituto de Instrumentación para Imagen Molecular*
[c]*Departamento Sistemas Informáticos y Computación.*

Abstract

There are, currently, very few implementations to compute the hyperbolic cosine of a matrix. This work tries to fill this gap. To this end, we first introduce both a new rational-polynomial Hermite matrix expansions and a formula for the forward relative error of Hermite approximation in exact arithmetic with a sharp bound for forward error. This matrix expansion allows to obtain a new accurate and efficient method for computing the hyperbolic matrix cosine. We present a MATLAB implementation based on this method, which shows a superior efficiency and a similar or better accuracy than other state-of-the-art methods. The algorithm developed on the basis of this method is also able to run on an NVIDIA GPU thanks to a MEX file that connects the Matlab script to the CUDA code.

Keywords:
Hermite matrix approximation, Matrix hyperbolic cosine, Error analysis, GPU computing

1. Introduction

Coupled partial differential problems can often be found in many different areas of science and technology, for example, in biochemistry, in the study of elastic and inelastic contact problems of solids, cardiology, diffusion problems, magnetohydrodynamic flows, etc. (see [1, 2, 3, 4] and references therein). In particular, coupled hyperbolic systems appear in optics [5] and in the study of microwave heating processes [6].

Functions of a square matrix A frequently arise in many areas of science and technology, especially those that require the resolution of first and second order

[*]Corresponding author
Email addresses: edefez@imm.upv.es (E. Defez), jjibanez@dsic.upv.es (J. Ibáñez), jpeinado@dsic.upv.es (J. Peinado), palonso@upv.es (P. Alonso-Jordá), jmalonso@dsic.upv.es (José M. Alonso)

differential systems [7]. In particular, the hyperbolic matrix functions $\cosh(A)$ and $\sinh(A)$, defined in terms of the exponential matrix e^A as

$$\cosh(A) = \frac{e^A + e^{-A}}{2}, \quad \text{and} \quad \sinh(A) = \frac{e^A - e^{-A}}{2} ,$$

are involved in the solution of coupled hyperbolic systems of partial differential equations [8]. Moreover, we also can find applicability to these functions in other fields of science and engineering, e.g. communicability analysis in complex networks [9, 10, 11, 12].

A way of computing these matrix functions is to use the well-known relations:

$$\cosh(A) = \cos(iA), \quad \text{and} \quad \sinh(A) = i\cos\left(A - \frac{i\pi}{2}I\right), \; i^2 = -1,$$

provided there exists a method to compute $\cos(A)$. But this approach has the disadvantage of requiring complex arithmetic even being matrix A real.

There exist, however, alternative and more practical ways to evaluate these matrix functions. One of them uses Hermite matrix polynomials series expansions [13]. Other methods based on Taylor series have been studied to evaluate the action of these functions on vectors [14, 15].

1.1. Notation

Throughout this paper, we denote by $\mathbb{C}^{r\times r}$ the set of all the complex square matrices of size r. We denote by Θ and I, respectively, the zero and the identity matrix in $\mathbb{C}^{r\times r}$. If $A \in \mathbb{C}^{r\times r}$, we denote by $\sigma(A)$ the set of all the eigenvalues of A. For a real number x, $\lfloor x \rfloor$ denotes the lowest integer not less than x and $\lceil x \rceil$ denotes the highest integer not exceeding x.

If $f(z)$ and $g(z)$ are holomorphic functions in an open set Ω of the complex plane, and if $\sigma(A) \subset \Omega$, we denote by $f(A)$ and $g(A)$, respectively, the image by the Riesz-Dunford functional calculus of functions $f(z)$ and $g(z)$, respectively, acting on the matrix A, being $f(A)g(A) = g(A)f(A)$ [16, p. 558]. We say that matrix A is positive stable if $Re(z) > 0$ for every eigenvalue $z \in \sigma(A)$. In this case, let us denote $\sqrt{A} = A^{1/2} = \exp\left(\frac{1}{2}\log(A)\right)$ the image of the function $z^{1/2} = \exp\left(\frac{1}{2}\log(z)\right)$ by the Riesz-Dunford functional calculus, acting on the matrix A, where $\log(z)$ denotes the principal branch of the complex logarithm.

In this paper, we use consistent matrix norms, in particular, $\|A\|_2$ is the 2-norm. In tests we use the 1-norm of a matrix $A \in \mathbb{C}^{r\times r}$ defined by $\|A\|_1 = \sup_{x\neq 0} \frac{\|Ax\|_1}{\|x\|_1}$, where $\|\cdot\|_1$ denotes the vector 1-norm defined as $\|y\|_1 = |y_1| + \cdots + |y_r|$, $y \in \mathbb{C}^r$ [17, Chap. 2].

For a positive stable matrix $A \in \mathbb{C}^{r\times r}$ the n-th Hermite matrix polynomial is defined in [18] by:

$$H_n(x,A) = n! \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \frac{(-1)^k \left(\sqrt{2A}\right)^{n-2k}}{k!(n-2k)!} x^{n-2k}, \tag{1}$$

which satisfies the three-term matrix recurrence:

$$\left.\begin{array}{l} H_n(x,A) = x\sqrt{2A} H_{n-1}(x,A) - 2(n-1)H_{n-2}(x,A)\ ,\ n \geq 1, \\ H_{-1}(x,A) = \Theta\ ,\ H_0(x,A) = I\ . \end{array}\right\} \tag{2}$$

The following upper bound of Hermite matrix polynomials

$$\left.\begin{array}{c} \|H_{2n}(x,A)\|_2 \leq g_n(x)\ ,\ n \geq 1 \\ \|H_{2n+1}(x,A)\|_2 \leq |x| \left\| \left(\frac{A}{2}\right)^{-\frac{1}{2}} \right\|_2 \frac{2g_n(x)}{n+1}\ ,\ n \geq 0 \end{array}\right\} \tag{3}$$

was demonstrated in [19], where function $g_n(x)$ is defined as

$$g_n(x) = \frac{(2n+1)!2^{2n}}{n!} \exp\left(\frac{5}{2}\|A\|_2 x^2\right), \qquad n \geq 0. \tag{4}$$

The Hermite matrix polynomial sequence $\{H_n(x,A)\}_{n\geq 0}$ has the following generating function [18]:

$$e^{xt\sqrt{2A}} = e^{t^2} \sum_{n\geq 0} \frac{H_n(x,A)}{n!} t^n\ ,$$

from which we can derive the following expressions for the matrix hyperbolic sine and cosine [20]:

$$\left.\begin{array}{rcl} \cosh\left(xt\sqrt{2A}\right) & = & e^{t^2} \sum_{n\geq 0} \frac{H_{2n}(x,A)}{(2n)!} t^{2n} \\ \sinh\left(xt\sqrt{2A}\right) & = & e^{t^2} \sum_{n\geq 0} \frac{H_{2n+1}(x,A)}{(2n+1)!} t^{2n+1} \end{array}\right\}\ ,\ x \in \mathbb{R}, |t| < \infty. \tag{5}$$

Recent developments, polynomial and rational, in series of Hermite polynomials have been obtained in [21]. These results have led to the development of a more accurate and efficient method compared with others proposed in the literature for the computation of, e.g. the matrix cosine function. In this paper we calculate the exact value of the following Hermite matrix polynomial series:

$$\sum_{n\geq 0} \frac{H_{2n+1}(x,A)}{(2n)!} t^{2n} := \mathcal{A}(x,t;A)\ , \tag{6}$$

$$\sum_{n\geq 0} \frac{H_{2n+2}(x,A)}{(2n+1)!} t^{2n+1} := \mathcal{B}(x,t;A)\ , \tag{7}$$

and

$$\sum_{n\geq 0} \frac{H_{2n+3}(x,A)}{(2n+1)!} t^{2n+1} := \mathcal{C}(x,t;A)\ , \tag{8}$$

which results in a new expansion of the hyperbolic matrix cosine in Hermite matrix polynomials. This method improves the one proposed in [13].

The organization of this paper is as follows. Section 2 presents the proof for the formulas (6)-(8) introduced in this paper. Section 3 deals with the new rational-polynomial Hermite matrix expansions for the hyperbolic matrix cosine. The proposed algorithm and its MATLAB implementation is presented in Section 4. The numerical results are presented in Section 5. Some final conclusions are given in Section 6.

2. A proof of formulas (6)-(8)

The aim is to calculate the exact value of matrix series $\mathcal{A}(x,t;A)$, $\mathcal{B}(x,t;A)$ and $\mathcal{C}(x,t;A)$ defined by (6)-(8). Next, we prove that both matrix series are convergent. Taking into account (3) we have

$$\left\| \frac{H_{2n+1}(x,A)}{(2n)!} t^{2n} \right\| \leq |x| \left\| \left(\frac{A}{2}\right)^{-\frac{1}{2}} \right\|_2 \frac{2g_n(x)}{(n+1)(2n)!} |t|^{2n}.$$

Since using (4), $\sum_{n\geq 0} \frac{g_n(x)}{(n+1)(2n)!}|t|^{2n}$ is convergent for $|t| < \infty$, the matrix series $\mathcal{A}(x,t;A)$ defined by (6) is convergent in any compact real interval. In the same way, we have

$$\left\| \frac{H_{2n+2}(x,A)}{(2n+1)!} t^{2n+1} \right\| \leq \frac{g_{n+1}(x)}{(2n+1)!} |t|^{2n+1}.$$

Since using (4), $\sum_{n\geq 0} \frac{g_{n+1}(x)}{(2n+1)!}|t|^{2n+1}$ is convergent for $|t| < \infty$, the matrix series $\mathcal{B}(x,t;A)$ defined by (7) is convergent in any compact real interval. Analogously and taking into account (3) again, we have

$$\left\| \frac{H_{2n+3}(x,A)}{(2n+1)!} t^{2n+1} \right\| \leq |x| \left\| \left(\frac{A}{2}\right)^{-\frac{1}{2}} \right\|_2 \frac{2g_{n+1}(x)}{(n+2)(2n+1)!} |t|^{2n+1}.$$

Since using (4), $\sum_{n\geq 0} \frac{g_{n+1}(x)}{(n+2)(2n+1)!}|t|^{2n+1}$ is convergent for $|t| < \infty$, the matrix series $\mathcal{C}(x,t;A)$ defined by (8) is convergent in any compact real interval. Using now (2), (6) and that $H_1(x,A) = \sqrt{2A}x$, we can write:

$$\begin{aligned}
\mathcal{A}(x,t;A) &= \left(x\sqrt{2A}\right)\sum_{n\geq 0}\frac{H_{2n}(x,A)}{(2n)!}t^{2n} - 2\sum_{n\geq 1}\frac{(2n)H_{2n-1}(x,A)}{(2n)!}t^{2n}\\
&= H_1(x,A)e^{-t^2}\cosh\left(xt\sqrt{2A}\right) - 2t\sum_{n\geq 1}\frac{H_{2n-1}(x,A)}{(2n-1)!}t^{2n-1}\\
&= H_1(x,A)e^{-t^2}\cosh\left(xt\sqrt{2A}\right) - 2t\sum_{n\geq 0}\frac{H_{2n+1}(x,A)}{(2n+1)!}t^{2n+1}\\
&= H_1(x,A)e^{-t^2}\cosh\left(xt\sqrt{2A}\right) - 2te^{-t^2}\sinh\left(xt\sqrt{2A}\right)\\
&= e^{-t^2}\left[H_1(x,A)\cosh\left(xt\sqrt{2A}\right) - 2t\sinh\left(xt\sqrt{2A}\right)\right].
\end{aligned}$$

The proof of (7) is similar giving:

$$\begin{aligned}
\mathcal{B}(x,t;A) &= \left(x\sqrt{2A}\right)\sum_{n\geq 0}\frac{H_{2n+1}(x,A)}{(2n+1)!}t^{2n+1} - 2\sum_{n\geq 0}\frac{(2n+1)H_{2n}(x,A)}{(2n+1)!}t^{2n}\\
&= H_1(x,A)e^{-t^2}\sinh\left(xt\sqrt{2A}\right) - 2t\sum_{n\geq 0}\frac{H_{2n}(x,A)}{(2n)!}t^{2n}\\
&= H_1(x,A)e^{-t^2}\sinh\left(xt\sqrt{2A}\right) - 2te^{-t^2}\cosh\left(xt\sqrt{2A}\right)\\
&= e^{-t^2}\left[H_1(x,A)\sinh\left(xt\sqrt{2A}\right) - 2t\cosh\left(xt\sqrt{2A}\right)\right].
\end{aligned}$$

Working analogously for (8):

$$\begin{aligned}
\mathcal{C}(x,t;A) &= x\sqrt{2A}\sum_{n\geq 0}\frac{H_{2n+2}(x,A)}{(2n+1)!}t^{2n+1} - 2\sum_{n\geq 0}\frac{(2n+2)H_{2n+1}(x,A)}{(2n+1)!}t^{2n+1}\\
&= H_1(x,A)\mathcal{B}(x,t;A) - 2\left(\sum_{n\geq 0}\frac{(2n+1)H_{2n+1}(x,A)}{(2n+1)!}t^{2n+1}\right.\\
&+ \left.\sum_{n\geq 0}\frac{H_{2n+1}(x,A)}{(2n+1)!}t^{2n+1}\right)\\
&= H_1(x,A)\mathcal{B}(x,t;A) - 2\left(\sum_{n\geq 0}\frac{H_{2n+1}(x,A)}{(2n)!}t^{2n+1} + e^{-t^2}\sinh\left(xt\sqrt{2A}\right)\right)\\
&= H_1(x,A)\mathcal{B}(x,t;A) - 2t\mathcal{A}(x,t;A) - 2e^{-t^2}\sinh\left(xt\sqrt{2A}\right).
\end{aligned}$$

Taking into account the values of $\mathcal{A}(x,t;A)$ and $\mathcal{B}(x,t;A)$, we get

$$\begin{aligned}
\mathcal{C}(x,t;A) &= H_1(x,A)\mathcal{B}(x,t;A) - 2t\mathcal{A}(x,t;A) - 2e^{-t^2}\sinh\left(xt\sqrt{2A}\right)\\
&= e^{-t^2}\left[\left(H_1(x,A)^2 + (4t^2-2)I\right)\sinh\left(xt\sqrt{2A}\right) - 4tH_1(x,A)\cosh\left(xt\sqrt{2A}\right)\right].
\end{aligned}$$

By (1), we have that $H_1(x,A) = \sqrt{2A}x, H_2(x,A) = 2x^2A - 2I$, and we can rewrite the last expression of $\mathcal{C}(x,t;A)$ in the form

$$\mathcal{C}(x,t;A) := e^{-t^2}\left[\left(H_2(x,A) + 4t^2I\right)\sinh\left(xt\sqrt{2A}\right) - 4tH_1(x,A)\cosh\left(xt\sqrt{2A}\right)\right].$$

Summarizing, the following result has been established:

Theorem 2.1. *Let $A \in \mathbb{C}^{r\times r}$ be a positive stable matrix, $x \in \mathbb{R}, |t| < +\infty$. Then*

$$\left.\begin{aligned}
\sum_{n\geq 0}\frac{H_{2n+1}(x,A)}{(2n)!}t^{2n} &= e^{-t^2}\left[H_1(x,A)\cosh\left(xt\sqrt{2A}\right) - 2t\sinh\left(xt\sqrt{2A}\right)\right],\\
\sum_{n\geq 0}\frac{H_{2n+2}(x,A)}{(2n+1)!}t^{2n+1} &= e^{-t^2}\left[H_1(x,A)\sinh\left(xt\sqrt{2A}\right) - 2t\cosh\left(xt\sqrt{2A}\right)\right],\\
\sum_{n\geq 0}\frac{H_{2n+3}(x,A)}{(2n+1)!}t^{2n+1} &= e^{-t^2}\left[\left(H_2(x,A) + 4t^2I\right)\sinh\left(xt\sqrt{2A}\right) - 4tH_1(x,A)\cosh\left(xt\sqrt{2A}\right)\right].
\end{aligned}\right\} \tag{9}$$

Taking into account that the Hermite matrix polynomial $H_n(x,A)$ coincides with the Hermite polynomial $H_n(x)$ taking $r = 1$ and $A = 2$ (see [18] for more details) we get the following corollary:

Corollary 1. *Let $\{H_n(x)\}_{n\geq 0}$ be the sequence of Hermite polynomials, $x \in \mathbb{R}, |t| < +\infty$. Then*

$$\left.\begin{aligned}
\sum_{n\geq 0}\frac{H_{2n+1}(x)}{(2n)!}t^{2n} &= e^{-t^2}\left[H_1(x)\cosh(2xt) - 2t\sinh(2xt)\right],\\
\sum_{n\geq 0}\frac{H_{2n+2}(x)}{(2n+1)!}t^{2n+1} &= e^{-t^2}\left[H_1(x)\sinh(2xt) - 2t\cosh(2xt)\right],\\
\sum_{n\geq 0}\frac{H_{2n+3}(x)}{(2n+1)!}t^{2n+1} &= e^{-t^2}\left[\left(H_2(x) + 4t^2\right)\sinh(2xt) - 4tH_1(x)\cosh(2xt)\right].
\end{aligned}\right\} \tag{10}$$

Formulas (10) are new in the literature of Hermite polynomials and special functions.

3. Some new Hermite matrix series expansions for the hyperbolic matrix cosine

Let $A \in \mathbb{C}^{r\times r}$ be a positive stable matrix, then the matrix polynomial $H_1(x,A) = \sqrt{2A}x$ is invertible if $x \neq 0$. Substituting $\sinh\left(xt\sqrt{2A}\right)$ given in (5) into the first expression of (9) we obtain a new rational expression for the hyperbolic matrix cosine in terms of Hermite matrix polynomials:

$$\cosh\left(xt\sqrt{2A}\right) = e^{t^2}\left(\sum_{n\geq 0}\frac{H_{2n+1}(x,A)}{(2n)!}\left(1+\frac{2t^2}{2n+1}\right)t^{2n}\right)\left[H_1(x,A)\right]^{-1}, \quad x\in\mathbb{R}\sim\{0\}, |t|<+\infty. \tag{11}$$

Substituting $\sinh\left(xt\sqrt{2A}\right)$ given in (5) into the second expression of (9) and using the three-term matrix recurrence (2) we obtain the expression of $\cosh\left(xt\sqrt{2A}\right)$ given in (5).

On the other hand, replacing the expression of $\sin\left(xt\sqrt{2A}\right)$ given in (5) into the third expression of (9), we obtain another new rational expression for the hyperbolic matrix cosine in terms of Hermite matrix polynomials:

$$\cosh\left(xt\sqrt{2A}\right) =$$

$$= \frac{-e^{t^2}}{4}\left[\sum_{n\geq 0}\frac{H_{2n+3}(x,A)}{(2n+1)!}t^{2n} - \left(H_2(x,A)+4t^2I\right)\star\left(\sum_{n\geq 0}\frac{H_{2n+1}(x,A)}{(2n+1)!}t^{2n+1}\right)\right]\left[H_1(x,A)\right]^{-1}, \quad x\in\mathbb{R}\sim\{0\}, |t|<+\infty. \tag{12}$$

Comparing (12) with (11), we observe that there is always one matrix product more when evaluating (12), the matrix product remarked by symbol "$\star$" in (12). Due to the importance of reducing the number of matrix products, see [22, 23, 24] for more details, we will focus mainly on the expansion (11).

From (1), it follows that, for $x\neq 0$:

$$\begin{aligned} H_{2n+1}(x,A)\left[H_1(x,A)\right]^{-1} &= \frac{(2n+1)!}{x}\sum_{k=0}^{n}\frac{(-1)^k x^{2(n-k)+1}(2A)^{n-k}}{k!(2(n-k)+1)!} \\ &= \widetilde{H}_{2n+1}(x,A), \end{aligned} \tag{13}$$

where

$$\widetilde{H}_n(x,A) = n!\sum_{k=0}^{\lfloor \frac{n}{2}\rfloor}\frac{(-1)^k\left(\sqrt{2A}\right)^{n-2k-1}}{k!(n-2k)!}x^{n-2k}, \tag{14}$$

so the right hand side of (13) is still defined in the case where matrix A is singular. In this way, we can re-write the relation (11) in terms of the matrix polynomial $\widetilde{H}_{2n+1}(x,A)$:

$$\cosh\left(xt\sqrt{2A}\right) = e^{t^2}\left(\sum_{n\geq 0}\frac{\widetilde{H}_{2n+1}\left(x,A\right)}{(2n)!}\left(1+\frac{2t^2}{2n+1}\right)t^{2n}\right), \quad x\in\mathbb{R}, |t|<+\infty. \tag{15}$$

Replacing matrix A by $A^2/2$ in (15) we can avoid the square roots of matrices, and taking $x=\lambda$, $\lambda\neq 0$, $t=1/\lambda$, we finally obtain the expression

$$\cosh\left(A\right) = e^{\frac{1}{\lambda^2}}\left(\sum_{n\geq 0}\frac{\widetilde{H}_{2n+1}\left(\lambda,\frac{1}{2}A^2\right)}{(2n)!\lambda^{2n+1}}\left(1+\frac{2}{(2n+1)\lambda^2}\right)\right), 0<\lambda<+\infty. \tag{16}$$

4. Numerical approximations

Truncating the given series (16) until order m, we obtain the approximation $CH_m\left(\lambda,A\right)\approx\cosh\left(A\right)$ defined by

$$CH_m\left(\lambda,A\right) = e^{\frac{1}{\lambda^2}}\left(\sum_{n=0}^{m}\frac{\widetilde{H}_{2n+1}\left(\lambda,\frac{1}{2}A^2\right)}{(2n)!\lambda^{2n+1}}\left(1+\frac{2}{(2n+1)\lambda^2}\right)\right), 0<\lambda<+\infty. \tag{17}$$

Working analogously as we did to get the proof of formula (3.6) in [13] we get, for $x\neq 0$:

$$\left\|\widetilde{H}_{2n+1}\left(x,\frac{1}{2}A^2\right)\right\|_2 \leq (2n+1)!\frac{e\sinh\left(|x|\left\|A^2\right\|_2^{1/2}\right)}{|x|\left\|A^2\right\|_2^{1/2}}. \tag{18}$$

We can obtain the following expression for the approximation error

$$\begin{aligned}\left\|\cosh\left(A\right)-CH_m\left(\lambda,A\right)\right\|_2 &\leq e^{\frac{1}{\lambda^2}}\sum_{n\geq m+1}\frac{\left\|\widetilde{H}_{2n+1}\left(\lambda,\frac{1}{2}A^2\right)\right\|_2}{(2n)!\lambda^{2n+1}}\left(1+\frac{2}{(2n+1)\lambda^2}\right) \\ &\leq \frac{e^{1+\frac{1}{\lambda^2}}\sinh\left(\lambda\left\|A^2\right\|_2^{1/2}\right)}{\lambda^2\left\|A^2\right\|_2^{1/2}}\sum_{n\geq m+1}\frac{2n+1}{\lambda^{2n}}\left(1+\frac{2}{(2n+1)\lambda^2}\right).\end{aligned} \tag{19}$$

Taking $\lambda>1$, it follows that $\frac{2}{(2n+1)\lambda^2}<1$, and

$$\begin{aligned}\sum_{n\geq m+1}\frac{2n+1}{\lambda^{2n}}\left(1+\frac{2}{(2n+1)\lambda^2}\right) &\leq 2\sum_{n\geq m+1}\frac{2n+1}{\lambda^{2n}} \\ &= \frac{4+(4m+6)(\lambda^2-1)}{\lambda^{2m}\left(\lambda^2-1\right)^2},\end{aligned}$$

thus, from (19) we finally obtain:

$$\|\cosh(A) - CH_m(\lambda, A)\|_2 \leq \frac{e^{1+\frac{1}{\lambda^2}} \sinh\left(\lambda \left\|A^2\right\|_2^{1/2}\right)\left(4 + (4m+6)(\lambda^2 - 1)\right)}{\left\|A^2\right\|_2^{1/2} \lambda^{2m+2}\left(\lambda^2 - 1\right)^2}. \tag{20}$$

From expression (20) we can derive the optimal values $(\lambda_m; z_m)$ such that

$$z_m = \max\left\{ z = \left\|A^2\right\|_2 ; \frac{e^{1+\frac{1}{\lambda^2}} \sinh\left(\lambda z^{1/2}\right)\left(4 + (4m+6)(\lambda^2 - 1)\right)}{z^{1/2}\lambda^{2m+2}\left(\lambda^2 - 1\right)^2} < u \right\},$$

where u is the unit roundoff in IEEE double precision arithmetic ($u = 2^{-53}$). The optimal values of z and λ for each m have been obtained with an iterative MATLAB code. The results are given in Table 1.

Approximation CH_m (see (17)) can be expressed as polynomial with only even powers of matrix A:

$$CH_m(\lambda, A) = \sum_{i=0}^{m} p_i^{(\lambda)} A^{2i} = \sum_{i=0}^{m} p_i^{(\lambda)} B^i \equiv P_m^{(\lambda)}(B), \tag{21}$$

where be $B = A^2$. We present Algorithm 1 which computes the hyperbolic cosine of a matrix A by means the Paterson-Stockmeyer method ([25]).

Algorithm 1 Scaling and recovering algorithm for computing $C = \cosh(A)$, where $A \in \mathbb{C}^{r\times r}$, with $m_M = 16$ the maximum approximation order allowed.

```
1: $B = A^2$.
2: Choose optimal values of $m_k \in \{2, 4, 6, 9, 12, 16\} \leqslant m_M$ and the scaling
   parameter $s \in \mathbb{N} \cup \{0\}$.
3: $B = 4^{-s}B$                                        ▷ Scaling matrix $B$
4: Choose the corresponding $\lambda_{m_k}$ from Table 1.
5: Compute $C = P_{m_k}^{(\lambda_{m_k})}(B)$ by the Paterson-Stockmeyer method (See [26,
   Section 2]) .
6: for $i = 1 : s$ do                 ▷ Recovering the approximation of cosh($A$)
7:     $C = 2C^2 - I$                      ▷ Double angle formula of cosh($A$)
8: end for
```

The basic steps of this algorithm are the choosing of m_k and s (step 2), the computation of $P_{m_k}^{(\lambda_{m_k})}(B)$ (a complete study of how to Compute $P_{m_k}^{(\lambda_{m_k})}(B)$ can be seen in Section 2 from [26]), step 5, and the recovering steps 6-7.

Next let's show how to compute the values of m_k and s. If $\cosh(A)$ is computed from the Taylor series, then the absolute forward error of the Hermite approximation of $\cosh(A)$, denoted by E_f, can be computed as

$$E_f = \left\|\cosh(A) - P_{m_k}^{(\lambda_k)}(B)\right\| = \left\|\sum_{i \geq \hat{m}_k} f_i B^i\right\|,$$

Table 1: Values of z_{m_k}, λ_{m_k} and Θ_{m_k} of the matrix function $\cosh(A)$.

m_k	z_{m_k}	λ_{m_k}	Θ_{m_k}
2	0.0020000000061361199	909.39256098888882	$3.0278415575147896 \cdot 10^{-5}$
4	0.079956209874370632	99.997970988888895	$3.6905278917160876 \cdot 10^{-3}$
6	0.34561400005673254	39.999499988888893	$1.7003229163751021 \cdot 10^{-1}$
9	1.1120032200657	17.997896988889799	$1.6336837269432252 \cdot 10^{0}$
12	2.2373014291079998	11.882978988901458	$6.2251021047024793 \cdot 10^{0}$
16	4.1086396680000004	7.9999999964157498	$2.0043654334857223 \cdot 10^{1}$

Table 2: Values $\hat{m}_k$, $\tilde{m}_k$, and $f_{\max}$.

	$m_1 = 2$	$m_2 = 4$	$m_3 = 6$	$m_4 = 9$	$m_5 = 12$	$m_6 = 16$
$\hat{m}_k$	1	2	2	4	6	9
$\tilde{m}_k$	1	2	3	10	13	17
$f_{m_k}(\max)$	0	0	$1.9 \cdot 10^{-17}$	$5.3 \cdot 10^{-19}$	$3.1 \cdot 10^{-26}$	$3.4 \cdot 10^{-39}$

where $\hat{m}_k \leq m_k$. We have tested that by eliminating the coefficients of the previous series whose absolute value is less than the unit roundoff in double precision floating-point arithmetic u, then we obtain an excellent accuracy. If $f_{\tilde{m}_k}$ is the first value of the above series which has not been rejected, then we obtain the following approximation:

$$E_f \cong \left\| \sum_{i \geqslant \tilde{m}_k} f_{m_k,i} B^i \right\|.$$

Table 2 shows values $\hat{m}_k$, $\tilde{m}_k$, and the maximum value rejected, $f_{m_k}(\max)$, for each $m_k \in \{2, 4, 6, 9, 12, 16\}$.

The scaling factor s and the order of the Hermite approximation m_k are obtained by simplifying the following theorem:

Theorem 4.1 ([27]). *Let $h_l(x) = \sum_{i \geq l} p_i x^i$ be a power series with radius of convergence w, $\tilde{h}_l(x) = \sum_{i \geq l} |p_i| x^i$, $B \in \mathbb{C}^{n \times n}$ with $\rho(B) < w$, $l \in \mathbb{N}$ and $t \in \mathbb{N}$ with $1 \leqslant t \leqslant l$. If t_0 is the multiple of t such that $l \leqslant t_0 \leqslant l + t - 1$ and*

$$\beta_t = \max\{d_j^{1/j} : j = t, l, l+1, \ldots, t_0 - 1, t_0 + 1, t_0 + 2, \ldots, l + t - 1\},$$

where d_j is an upper bound for $||B^j||$, $d_j \geqslant ||B^j||$, then

$$||h_l(B)|| \leqslant \tilde{h}_l(\beta_t).$$

If we apply Theorem 4.1 to the series $f_{\tilde{m}_k}(x) = \sum_{i \geqslant \tilde{m}_k} f_{m_k,i} x^i$ and $\tilde{f}_{\tilde{m}_k}(x) = \sum_{i \geq \tilde{m}_k} |f_{m_k,i}| \, x^i$, then

$$E_f = \|f_{\tilde{m}_k}(B)\| \leqslant \tilde{f}_{\tilde{m}_k}(\beta_t),$$

for every t, $1 \leq t \leq \tilde{m}_k$. Let Θ_{m_k} be

$$\Theta_{m_k} = \max\left\{\theta \geqslant 0 : \tilde{f}_{\tilde{m}_k}(\theta) = \sum_{i \geqslant \tilde{m}_k} |f_{m_k,i}|\,\theta^i \leqslant u\right\}, \qquad (22)$$

then using MATLAB (R2017b) Symbolic Math Toolbox with 200 series terms and a zero finder, we obtained the values Θ_{m_k} that verify (22) (see Table 1).

The optimal values m_k and s are obtained from the values of β_t of Theorem 4.1 and from the values Θ_{m_k} of Table 1. A complete study of this question was developed by the authors in [26, 21]. Next we reproduced that study.

Let be $\beta_{\min}^{(\tilde{m}_k)} = \min_{1 \leqslant t \leqslant \tilde{m}_k} \{\beta_t\}$. If there exists a value $m_k \leq 16$ such that $\beta_{\min}^{(\tilde{m}_k)} \leq \Theta_{m_k}$, then the forward error E_f is lower than u. In this case, we choose the lower order m_k such that $\beta_{\min}^{(\tilde{m}_k)} \leq \Theta_{m_k}$ and the scaling factor is $s = 0$. Otherwise, we choose the Hermite approximation of order 12 or 16 providing the lower cost, with

$$s = \max\left\{0, \left\lceil \frac{1}{2}\log\left(\frac{\beta_{\min}^{(\tilde{m}_k)}}{\Theta_{m_k}}\right)\right\rceil\right\},\ m_k = 12 \text{ or } 16.$$

For computing $\beta_{\min}^{(\tilde{m}_k)}$ we have used the following approximation:

$$\beta_{\min}^{(\tilde{m}_k)} \approx \max\left\{d_{\tilde{m}_k}^{1/\tilde{m}_k},\ d_{\tilde{m}_k+1}^{1/(\tilde{m}_k+1)}\right\},$$

where $d_{\tilde{m}_k}$ and $d_{\tilde{m}_k+1}$ are bounds of $\left\|B^{\tilde{m}_k}\right\|$ and $\left\|B^{\tilde{m}_k+1}\right\|$, respectively (see (16) from [28])). The bounds d_l, $l = \tilde{m}_k, \tilde{m}_{k+1}$ can be computed using products of norms of matrix powers previously computed. For example, for $m_k = 6$ the powers B^2 and B^3 must be computed, hence $\beta_{min}^{(3)}$ $(\tilde{m}_k = 3)$ can be computed as follows

$$\beta_{min}^{(3)} = \max\left\{\left\|B^3\right\|^{1/3}, \min\left\{\left\|B^3\right\|\|B\|, \left\|B^2\right\|^2\right\}^{1/4}\right\}.$$

The algorithm for computing the values m and s is analogous to Algorithm 2 from [26].

5. Experiments

In this section we show the results of accuracy and performance of the algorithm to compute the function cosh proposed. Using our CUDA implementation we also show the performance using an NVIDIA GPU.

5.1. Numerical experiments

Our MATLAB implementation, named `coshmtayher`, has been developed by modifying the MATLAB code `coshher` given in [13], by replacing the original Hermite approximation `coshher` by the new Hermite matrix polynomial developed in this paper and derived from (16). We compare the new MATLAB function, `coshmtayher`, with functions `coshher` and `funmcosh` defined as:

- `coshmtayher`. New code based on the new developments of Hermites matrix polynomials (16).
- `coshher`. Code based on the Hermite series for the hyperbolic cosine [13].
- `funmcosh`. MATLAB function `funm` to compute matrix functions, i.e. the hyperbolic matrix cosine.

5.2. Experiments description

The tests have been carried out using MATLAB (R2017b) running on an Apple Macintosh iMac 27" (iMac retina 5K 27" late 2015) with a quadcore INTEL i7-6700K 4Ghz processor and 16 Gb of RAM. The following tests were made using different matrices:

a) Test 1: One hundred diagonalizable 128×128 real matrices with 1-norm varying from 2.32 to 220.04. These matrices have the form $A = VDV^T$, where D is a diagonal matrix with real eigenvalues and V is an orthogonal matrix obtained as $V = H/16$, where H is the Hadamard matrix. The *"exact"* matrix hyperbolic cosine was computed as $\cosh(A) = V \cosh(D) V^T$ (see [7, pp. 10]), by using the MATLAB Symbolic Math Toolbox with 128 decimal digit arithmetic in all the computations.

b) Test 2: One hundred non-diagonalizable 128×128 real matrices whose 1-norm vary from 6.52 to 249.61. These matrices have the form $A = VJV^T$, where J is a Jordan matrix with real eigenvalues with algebraic multiplicity varying between 1 and 4, and V is an orthogonal matrix obtained as $V = H/16$, where H is the Hadamard matrix. The *"exact"* matrix hyperbolic cosine was computed as $\cosh(A) = V \cosh(J) V^T$.

c) Test 3: Comparison between `funmcosh`, `coshher` and `coshmtayher` for 13 test matrices from the Eigtool MATLAB package [29] with size 128×128 and 39 matrices from the matrix function literature with dimensions lower than or equal to 128 from the function `matrix` of the Matrix Computation Toolbox [30]. These matrices have been scaled so that they have 1-norm not exceeding 1024. The *"exact"* matrix hyperbolic cosine was computed by computing first the eigenvalue decomposition of matrix A, $A = VDV^{-1}$, by using the MATLAB function `eig`, and computing later $\cosh(A) = V \cosh(D) V^{-1}$, using the MATLAB's Symbolic Math Toolbox with 128 decimal digit arithmetic in all the computations.

Table 3: Matrix products (M) for Tests 1, 2, and 3 using MATLAB functions `coshmtayher`, `coshher` and `funmcosh`.

	`M(coshmtayher)`	`M(coshher)`	`M(funmcosh)`
Test 1	671	973	1400
Test 2	685	989	1400
Test 3	191	315	560

Table 4: Relative error comparison between `cosmtayher` vs `coshher` (row 1) and `cosmtayher` vs `funmcosh` (row 2) for Test 1, Test 2 and Test 3.

	Test 1	Test 2	Test 3
E(`coshmtayher`)<E(`coshher`)	47.50%	52%	57.50%
E(`coshmtayher`)<E(`funmcosh`)	100%	100%	97.50%

The following results show the computational costs and the algorithm accuracy for the functions under comparison, i.e. `coshmtayher`, `coshher`, and `funmcosh`, in the three tests described. The algorithm accuracy is tested by computing the relative error

$$\mathrm{E} = \frac{\| \cosh(A) - \tilde{Y}\|_1}{\|\cos(A)\|_1},$$

where $\tilde{Y}$ is the computed solution and $\cosh(A)$ is the exact solution.

Table 3 shows the computational costs of each algorithm represented in terms of the number of matrix products (`M()`) of each code, since the cost of the rest of the operations is negligible compared to matrix products for big enough matrices. Routine `funmcosh` has no matrix products, but a Real Schur Form reduction as the main cost. This is a computational cost of $28n^3$ [31]. This can be expressed as a minimum of 14 matrix products. The cost of a matrix product is $2n^3$ [32].

Table 4, on the other hand, shows the percentage of cases in which the relative errors of `coshmtayher` (New Hermite) are lower, greater or equal than the relative errors of `coshher`(Hermite) and `funmcosh`(funm).

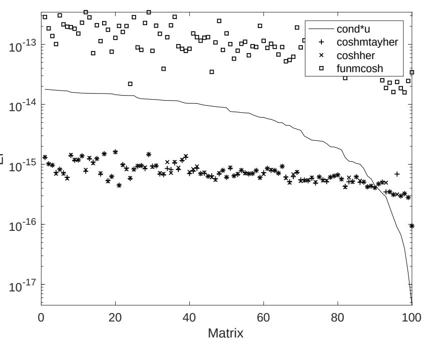
We have plotted in Figures 1, 2, and 3, for the three tests, respectively, the Normwise relative errors (a), the Performance Profiles (b), the ratio of relative errors (c) to show if these ratios are significant:

$$\mathrm{E}(\texttt{coshmtayher})/\mathrm{E}(\texttt{coshher}),\ \mathrm{E}(\texttt{coshmtayher})/\mathrm{E}(\texttt{funmcosh}),$$

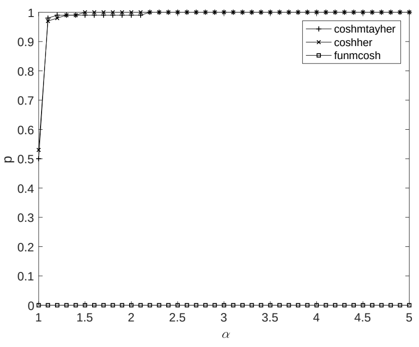
and the Ratio of the matrix products (d):

$$\mathrm{M}(\texttt{coshmtayher})/\mathrm{M}(\texttt{coshher}),\ \mathrm{M}(\texttt{coshmtayher})/\mathrm{M}(\texttt{funmcosh}).$$

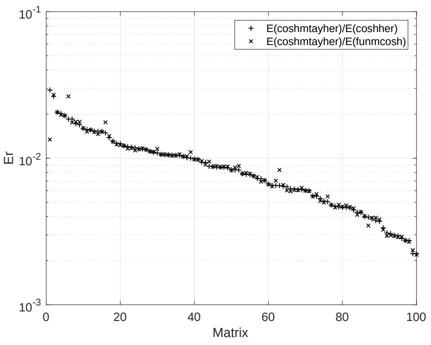
In the performance profile, the α coordinate varies between 1 and 5 in steps equal to 0.1, and the p coordinate is the probability that the considered algorithm has



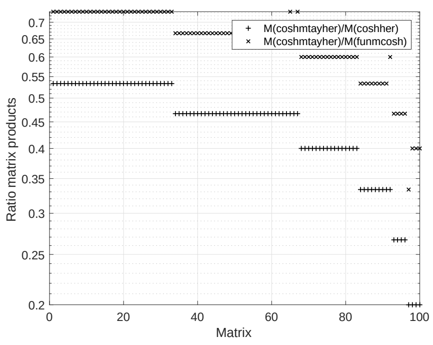
(a) Normwise relative errors.



(b) Performance profile.



(c) Ratio of relative errors.



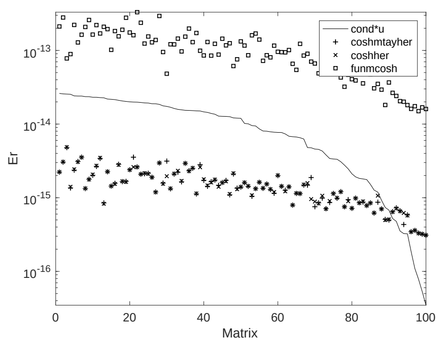
(d) Ratio of matrix products.

Figure 1: Experimental results for Test 1.

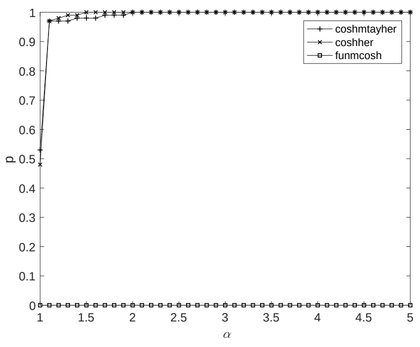
a relative error lower than or equal to α-times the smallest error over all methods. The ratios of relative errors are presented in decreasing order with respect to E(`coshmtayher`)/E(`coshher`) and E(`coshmtayher`)/E(`funmcosh`). The solid lines in figures 1a, 2a and 3a is the function $k_{\cos}u$, where $k_{\cos}$ is the condition number of matrix cosine function [7, Chapter 3] and $u = 2^{-53}$ is the unit roundoff in the double precision floating-point arithmetic. Our conclusions are:

- Regarding the normwise relative error shown in Figures 1a, 2a and 3a, in general, functions `coshmtayher` and `coshher` have a very good numerical stability. This can be appreciated seeing the distance from each matrix normwise relative error to the $cond * u$ line. In Figures 1a and 2a, the numerical stability is better because the relative errors are below the $cond * u$ line.

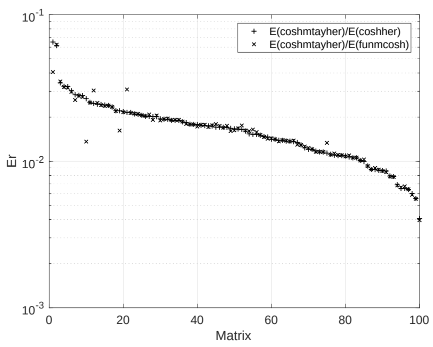
- The performance profile for the first two tests (Figures 1b, 2b) shows that accuraccy of the `coshmtayher` and the `coshher` methods is similar. Both of them have much better accuracy than the `funmcosh` method. For the third test, Figure 3b shows that the accuracy of function `coshmtayher` is a little better than the accuracy of function `coshher`.



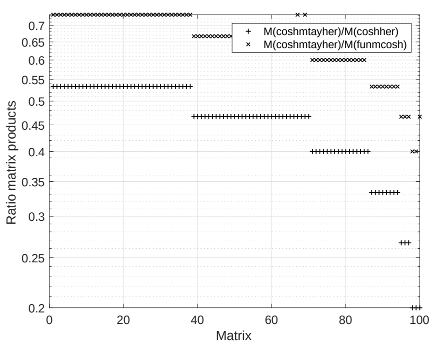
(a) Normwise relative errors.



(b) Performance profile.



(c) Ratio of relative errors.



(d) Ratio of matrix products.

Figure 2: Experimental results for Test 2.

- As is was shown in Table 3, function `coshmtayher` has significantly lower computational costs than the other two functions. The tests also confirm this result:

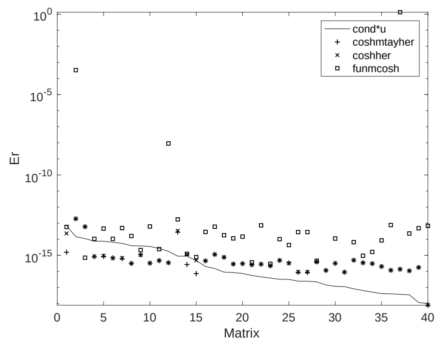
  Test 1 (Figure 1d):
  $\texttt{M(coshmtayher)} \in [0.2\,\texttt{M(coshher)}, 0.73\,\texttt{M(coshher)}]$,
  $\texttt{M(coshmtayher)} \in [0.33\,\texttt{M(funcosm)}, 0.53\,\texttt{M(funcosm)}]$.

  Test 2 (Figure 2d):
  $\texttt{M(coshmtayher)} \in [0.2\,\texttt{M(coshher)}, 0.8\,\texttt{M(coshher)}]$,
  $\texttt{M(coshmtayher)} \in [0.4\,\texttt{M(funcoshm)}, 0.73\,\texttt{M(funcoshm)}]$.

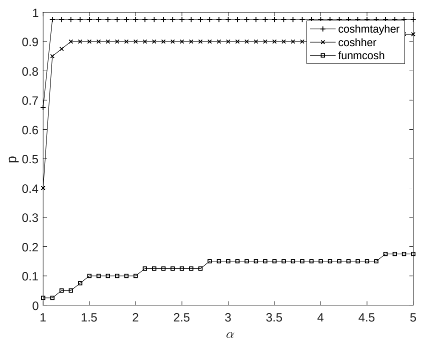
  Test 3 (Figure 3d):
  $\texttt{M(coshmtayher)} \in [0.33\,\texttt{M(coshher)}, 1\texttt{M(coshher)}]$,
  $\texttt{M(coshmtayher)} \in [0.26\,\texttt{M(funcoshm)}, 0.6\,\texttt{M(funcoshm)}]$.

5.3. Results in GPU

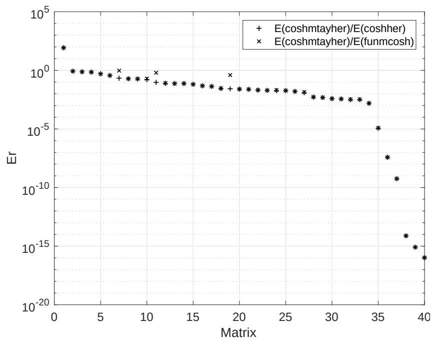
We have implemented an “accelerated” version that allows to execute our algorithm to compute the hyperbolic cosine on NVIDIA GPUs. Current GPUs



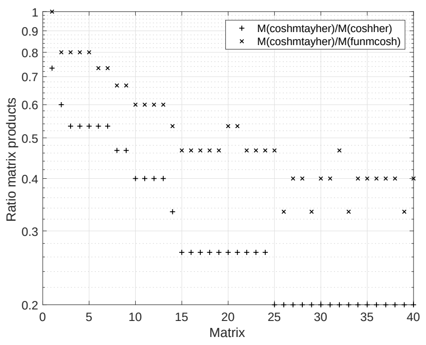
(a) Normwise relative errors.



(b) Performance profile.



(c) Ratio of relative errors.



(d) Ratio of matrix products.

Figure 3: Experimental results for Test 3.

are computational devices that allow to boost performance on data parallelism applications, i.e. applications that operate over many independent data. This is the case of matrix multiplication, which is a highly optimized operation for GPUs in its current implementation routine included in the CUBLAS [33] package. Our GPU algorithms are all based on polynomial evaluations which, in turn, results in intensive use of matrix products. The basic MATLAB algorithm is used in this case with some very costly operations (those based on matrix multiplication) addressed to the GPU through CUDA language by a means of implementing a MEX file.

We have carried out our experimental results on a computer equipped with two processors Intel Xeon CPU E5-2697 v2 @2.70GHz featuring 12 cores each. To obtain the algorithm performance on GPU we used one NVIDIATesla K20Xm (Kepler architecture) attached to the PCI of this workstation. This GPU features 2688 CUDA cores and 16 GB of memory.

Figure 4 shows the reduction in execution time when we use a GPU to accelerate the computations. The execution time for $n = 1000$ is very similar between CPU and GPU but, for $n = 2000$ the GPU performs twice the speed of the CPU. This difference augments as long as the problem size increases. We

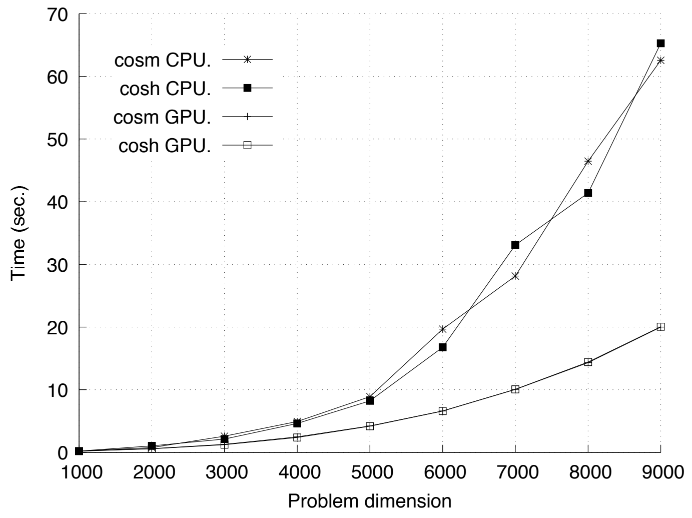


Figure 4: Execution time (sec.) of the algorithm to compute the cosine (cosm) and the algorithm to compute the hyperbolic cosine (cosh) on CPU and GPU large matrix sizes.

also compare the performance of both the former algorithm presented in [21] to compute the cosine of a matrix using Hermite matrix polynomials with the new one presented here to compute the hyperbolic cosine of a matrix using the same approximation. As it can be seen in the figure, both algorithms perform the same.

6. Conclusions

A new polynomial Hermite matrix algorithm has been developed in this work for computing the matrix hyperbolic cosine. We have implemented a MATLAB routine that is also capable of using an existing GPU in the system. This new algorithm has been compared with other MATLAB implementations and, at the light of the tests carried out, we have verified that the new algorithm behaves in a numerically stable manner showing very good results. One of the main conclusions is that MATLAB implementations based on the Hermite series (`coshher` and `coshmtayher`) have turned out to be more accurate and more efficient than others based on the *MATLAB* function `funm` when using algorithm `funmcosh`. In addition, the new implementation based on Hermite series (`coshmtayher`) presented here has lower computational cost than the other former version, also based on Hermite series (`coshher`), presenting, however, similar accuracy.

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.

References

[1] M. Sezgin, Magnetohydrodynamic flow in a rectangular duct, International journal for numerical methods in fluids 7 (7) (1987) 697–718.

[2] A. I. King, C. C. Chou, Mathematical modelling, simulation and experimental testing of biomechanical system crash response, Journal of biomechanics 9 (5) (1976) 301–317.

[3] L. Jódar, E. Navarro, J. Martin, Exact and analytic-numerical solutions of strongly coupled mixed diffusion problems, Proceedings of the Edinburgh Mathematical Society 43 (2) (2000) 269–293.

[4] V. Soler, E. Defez, M. Ferrer, J. Camacho, On exact series solution of strongly coupled mixed parabolic problems, Abstract and Applied Analysis 2013. doi:10.1155/2013/524514.

[5] P. K. Das, Optical signal processing, Springer,New York, 1991.

[6] D. M. Pozar, Microwave engineering, Addison-Wesley,New York, 1991.

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

[8] L. Jódar, E. Navarro, A. Posso, M. Casabán, Constructive solution of strongly coupled continuous hyperbolic mixed problems, Applied Numerical Mathematics 47 (3–4) (2003) 477–492.

[9] E. Estrada, D. J. Higham, N. Hatano, Communicability and multipartite structures in complex networks at negative absolute temperatures, Physical Review E 78 (2) (2008) 026102.

[10] E. Estrada, J. A. Rodríguez-Velázquez, Spectral measures of bipartivity in complex networks, Physical Review E 72 (4) (2005) 046105.

[11] E. Estrada, J. Gómez-Gardeñes, Network bipartivity and the transportation efficiency of european passenger airlines, Physica D: Nonlinear Phenomena 323 (2016) 57–63.

[12] J. Kunegis, G. Gröner, T. Gottron, Online dating recommender systems: The split-complex number approach, in: Proceedings of the 4th ACM RecSys workshop on Recommender systems and the social web, ACM, 2012, pp. 37–44.

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

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

[15] 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) (2018) A1696–A1713.

[16] N. Dunford, J. T. Schwartz, Linear operators, vol. I, Interscience, New York.

[17] G. H. Golub, C. F. Van Loan, Matrix computations, Johns Hopkins University Press, 1996.

[18] J. Jódar, R. Company, Hermite matrix polynomials and second order matrix differential equations, Approximation Theory and its Applications 12 (2) (1996) 20–30.

[19] E. Defez, A. Hervás, L. Jódar, A. Law, Bounding Hermite matrix polynomials, Mathematical and Computer Modelling 40 (1) (2004) 117–125.

[20] E. Defez, L. Jódar, Some applications of the Hermite matrix polynomials series expansions, Journal of Computational and Applied Mathematics 99 (1) (1998) 105–117.

[21] E. Defez, J. Ibáñez, J. Peinado, J. Sastre, P. Alonso-Jordá, An efficient and accurate algorithm for computing the matrix cosine based on new Hermite approximations, Journal of Computational and Applied Mathematics 348 (2019) 1–13.

[22] J. Sastre, J. Ibáñez, E. Defez, P. Ruiz, New scaling-squaring Taylor algorithms for computing the matrix exponential, SIAM Journal on Scientific Computing 37 (1) (2015) A439–A455.

[23] P. Alonso, J. Peinado, J. Ibáñez, J. Sastre, E. Defez, Computing matrix trigonometric functions with gpus through matlab, The Journal of Supercomputing (2018) 1–14.

[24] J. Sastre, Efficient evaluation of matrix polynomials, Linear Algebra and its Applications 539 (2018) 229–250.

[25] M. S. Paterson, L. J. Stockmeyer, On the number of nonscalar multiplications necessary to evaluate polynomials, SIAM Journal on Computing 2 (1) (1973) 60–66.

[26] J. Sastre, J. Ibáñez, P. Alonso, J. Peinado, E. Defez, Two algorithms for computing the matrix cosine function, Applied Mathematics and Computation 312 (2017) 66–77.

[27] J. Sastre, J. Ibáñez, P. Ruiz, E. Defez, Efficient computation of the matrix cosine, Applied Mathematics and Computation 219 (14) (2013) 7575–7585.

[28] P. Ruiz, J. Sastre, J. Ibáñez, E. Defez, High perfomance computing of the matrix exponential, J. Comput. Appl. Math. 291 (2016) 370–379.

[29] T. Wright, Eigtool, version 2.1.
URL `web.comlab.ox.ac.uk/pseudospectra/eigtool.`

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

[31] M. I. Smith, A Schur algorithm for computing matrix *p*th roots, SIAM J. Matrix Anal. Appl. 24 (4) (2003) 971–989.

[32] G. H. Golub, C. F. Van Loan, Matrix computations, Vol. 3, JHU Press, 2012.

[33] NVIDIA, CUDA. CUBLAS library (2009).