Bounds for Variable Degree Rational L_∞ Approximations to the Matrix Cosine

Ch. Tsitouras[a], V.N. Katsikis[b]

[a] *TEI of Sterea Hellas, GR34400, Psahna, Greece*
[b] *TEI of Piraeus, GR12244, Athens, Greece*

Abstract

In this work we derive new alternatives for efficient computation of the matrix cosine. We focus especially on the two classes of normal and nonnegative matrices and we present intervals of applications for rational L_∞ approximations of various degrees for these types of matrices in the lines of [3]. Our method relies on Remez algorithm for rational approximation while the innovation here is the choice of the starting set of non-symmetrical Chebyshev points. Only one Remez iteration is then enough to quickly approach infinitesimally close to the required L_∞ approximant.

Keywords: Matrix cosine, rational L_∞ approximation, Remez algorithm
2000 MSC: 65F30, 65F60, 65D20

1. Introduction

It is well known that in engineering applications many processes are described by second order differential equations and the exact solution for these equations is given in terms of trigonometric matrix functions sine and cosine. For example the following matrix differential problem

$$\frac{d^2y}{dt^2} + Ay = 0,\ y(0) = y_0,\ y'(0) = y'_0, \tag{1}$$

has solution

$$y(t) = \cos(\sqrt{A}t)y_0 + (\sqrt{A})^{-1}\sin(\sqrt{A}t)y'_0, \tag{2}$$

where $\sqrt{A}$ denotes any square root of matrix A. As a case in point, the problem described in equation (1), comes in front, when we are trying to solve the classical wave problem $v^2\frac{\partial^2 y}{\partial x^2} = \frac{\partial^2 y}{\partial t^2}$. To the best of our knowledge, the most efficient

Email addresses: tsitoura@teihal.gr (Ch. Tsitouras), vaskats@gmail.com (V.N. Katsikis)
URL: http://users.ntua.gr/tsitoura (Ch. Tsitouras), http://faculty.teipir.gr/vkatsikis (V.N. Katsikis)

and competitive algorithms for computing the matrix cosine are based on Padé rational approximations [3, 4] and on Hermite matrix polynomial series [1, 2].

In this work we are interested to compare rational approximations and, in particular the Padé approximation and the best L_∞ approximation.

Our main concern is to establish a competitive best rational L_∞ approximation method for the matrix cosine. For this reason, we focus on two large classes of matrices, normal matrices and nonnegative matrices (i.e., matrices for which each matrix element is a nonnegative real number). The proposed method seems to be advantageous over the Padé approximation in terms of digits of error as well as in number of multiplications. Our method relies on Remez algorithm for rational approximation while the innovation here is the choice of the starting set of non-symmetrical Chebyshev points. Only one modified Remez iteration is then enough to get infinitesimally close to the required L_∞ approximant.

This paper is organized as follows. Section 2 summarizes known facts on rational approximations and in particular the best L_∞ rational approximation and the Padé rational approximation. In Section 3, our proposed method based on Remez algorithm is described. Numerical experiments are presented in Section 4. Finally, conclusions are given in Section 5.

2. Preliminaries and notation

Rational functions $r_{km}(x) = \frac{p_k(x)}{q_m(x)}$, where p_k, q_m are polynomials in x of degree at most k, m, respectively, are of crucial importance for approximation. It is well known that a scalar approximation $f(x) \approx r_{km}(x)$ can be easily translated into a matrix approximation $f(A) \approx r_{km}(A)$, for $A \in \mathbb{C}^{n\times n}$, by applying the scalar approximation to the spectrum of the matrix A. In the case of normal matrices a good approximation on the spectrum imply a good matrix approximation. To see this, recall that by the spectral theorem if A is normal then $A = UDU^T$, where U is a unitary matrix and D is a diagonal matrix, in fact, the normal matrices are precisely those that are unitarily diagonizable . Note that, the class of normal matrices includes orthogonal matrices, symmetric matrices, skew-symmetric matrices and of course their complex analogues i.e., the unitary, Hermitian and skew-Hermitian matrices.

Let $\mathcal{R}_{k,m}$ denote the space of rational functions $r_{km}(x) = \frac{p_k(x)}{q_m(x)}$ with numerator p_k and denominator q_m polynomials in x of degree at most k, m, respectively.

The rational function $r \in \mathcal{R}_{k,m}$ is a *best* L_∞ *approximation* to a function f on $[a, b]$ if it holds

$$\|r(x) - f(x)\|_\infty = \min_{s\in\mathcal{R}_{k,m}} \|s(x) - f(x)\|_\infty.$$

This kind of approximations are usually used for normal matrices where the error bounds for the scalar problem translate directly into error bounds for the matrix problem. Best L_∞ rational approximations can be calculated using the Remez algorithm, which is a standard algorithmic procedure in approximation theory.

The rational function $r \in \mathcal{R}_{k,m}$ is a $[k/m]$ *Padé approximant* to a given scalar function f if $q_m(0) = 1$ and

$$f(x) - r_{km}(x) = O(x^{k+m+1}).$$

Note that, for a given f, k and m a $[k/m]$ Padé approximant might not exist but if it exists then it is unique.

Especially for the function cos we are interested for rational forms where numerator and denominator are polynomials in even powers sharing the same degree. Thus let's denote by $\hat{r}_d$ these elements of $\mathcal{R}_{d,d}$.

When $\|A\| \lesssim 1$, $\cos(A)$ may be easily approximated using the $\hat{r}_8$ Padé approximation. For large $\|A\|$, the usual procedure is a scaling and squaring method that can reduce the norm. At this point it useful to keep in mind that Padé approximants have the same drawbacks as Taylor expansions: they are local (i.e., around one value) approximations only. In [4], the authors developed and algorithm that chooses s so that $2^{-s}\|A\| \leq 1$, approximates $\cos(A) \approx \hat{r}_8(2^{-s}A)$ and then one can obtain a good approximation at reasonable cost of the matrix cosine by using the cosine double angle formula. Several improvements to this method where made in [3].

Namely, variable degree Padé approximants were used according to the magnitude of the norm of the matrix A. We illustrate this in the following algorithm which follows the lines of [3, Algorithm 3.1].

Algorithm 1 Given a matrix $A \in \mathbb{C}^{n\times n}$ this algorithm approximates $C = cos(A)$. It uses the constants θ_d given in Table-2.

Require: Matrix $A \in \mathbb{C}^{n\times n}$

$B = A^2$
$\theta = ||B||_\infty^{1/2}$
for $d = [2\ 4\ 6\ 8\ 12\ 16]$
  if $\theta \leq \theta_d$
    $C = \hat{r}_d(A)$
    quit
  end
end
$s = \text{ceil}(\log_2(\theta/\theta_{16}))$
$B \leftarrow 4^{-s}B$
$C = \hat{r}_d(2^{-s}A)$
for $i = 1 : s$
  $C = 2C^2 - I$
end

Actually in [3] even $d = 20$ was used. But here we found that at maximum $d = 16$ is enough. It is obvious that if we avoid Padé and try L_∞ approximations then θ_d will be different. For the matrices under consideration values of θ_d are expected to be greater. Then a matrix multiplication can be saved.

3. The Remez algorithm for efficient computation of the matrix cosine

Muller [6] states that "It is no longer necessary to write specific software or to perform long paper and pencil calculations in order to compute polynomial or rational approximations of functions. Software such as Maple readily computes minimax or Chebyshev approximations."

Indeed using Maple we may for example approximate $\hat{r}_4$ that best approximates cos over the interval $[-1, 1]$, by typing:

```
with(numapprox):
Digits := 40:
sol := fnormal(minimax(cos(x),x=-1..1,[4, 4],1,'err'),16,10^(-11));
```

and after some seconds we get the result

$$\frac{0.9780991458613181 - 0.4459077731148132 \cdot x^2 + 0.02006287192090119 \cdot x^4}{0.9780991466002248 + 0.04314176272693444 \cdot x^2 + 0.0008799254301546418 \cdot x^4}$$

The maximum error observed in the interval is $\epsilon \approx 7.5545 \cdot 10^{-10}$.

Mathematica offers the same benefit. Typing:

```
In[1]:=<< FunctionApproximations`
In[2]:=N[Chop[MiniMaxApproximation[Cos[x], {x, {-1, 1},
                    4, 5}, WorkingPrecision -> 33, MaxIterations -> 200]
                    [[2, 1]], 10^-16], 16]
```

we get

$$\text{Out[2]} := \frac{0.9999999989884365 - 0.4558785528455367 \cdot x^2 + 0.02050650951514645 \cdot x^4}{1 + 0.04412140084820871 \cdot x^2 + 0.0009008930686455960 \cdot x^4}$$

But the maximum error observed in the interval is $\epsilon \approx 1.0115 \cdot 10^{-9}$. This erratic behavior is common with `MiniMaxApproximation` function of `Mathematica`. Actually we couldn't derive $\hat{r}_{12}$ or higher order with the latter function. Thus we developed another heuristic approach that approximates the correct minimal at much less time than the corresponding function of `Maple`.

At first the new algorithm selects an initial set of Chebyshev-type nonsymmetrical points (say x_j) and fits a function $\hat{r}'_d$. In all cases checked, this first approximant $\hat{r}'_d$ was extremely close to the minimal. Then just one minimization of the sum of squares of the differences

$$\hat{r}_d(x'_j) - \cos(x'_j) + (-1)^j \epsilon$$

was enough to get the minimal ϵ, where ϵ is the norm of the maximum error and $x'_j,\ j = 1, 2, \cdots, 2d + 3$ the extremes of the difference $\hat{r}'_d - \cos$. The least squares approach overthrows "ill-posed" problems and allows us not to deal with

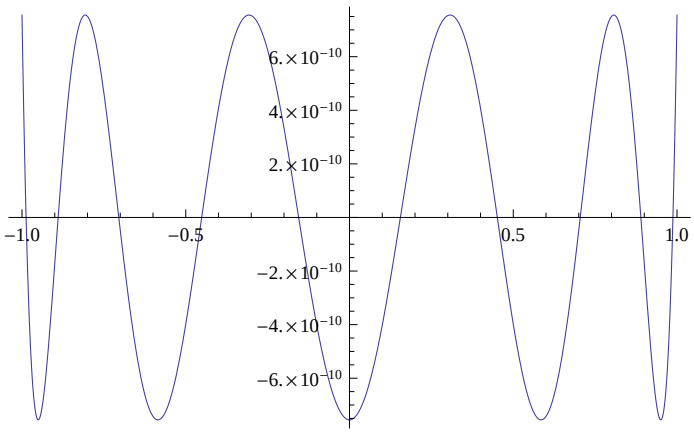

Figure 1: Error of $\hat{r}_4$ approximation by `minimax` and Algorithm-2 to cos

nullified coefficients of odd powers in polynomials of the rational forms. Thus only $d+2$ unknowns ($d+1$ coefficients of $\hat{r}_d$ and ϵ) are used to minimize the sum of squares of $2d+3$ factors. The algorithm is very fast since it is tuned for approaching cos function.

Algorithm 2 Given an even d and an interval $[-\theta, \theta]$ return the rational form $\hat{r}_d(x) = \frac{p_0+p_2x^2+\cdots+p_dx^d}{1+q_2x^2+\cdots+q_dx^d}$ that best approximates function cos in $[-\theta, \theta]$.

Require: Choose an even d and an interval $[-\theta, \theta]$

1: Choose points $x_{1.5d+0.75-\frac{j}{4}} = \theta \cdot \cos(\frac{\pi j}{6d})$, $j = 6d-1, 6d-5, 6d-9, \cdots, 3$

2: Find the least squares fit of $\hat{r}'_d$ to the data $(x_j,\ cosx_j)$, $j = 1, 2, \cdots, 1.5d$

3: Perform just one modified Remez iteration i.e.,

Compute points x' where $\hat{r}'_d(x) - \cos(x)$ has its $2d+3$ extremes

Find the following least squares sum for the coefficients of $\hat{r}_d$ and ϵ

$\sum_{j=1}^{j=2d+3}(\hat{r}_d(x'_j) - \cos(x'_j) + (-1)^j\epsilon)^2$.

In Figure-1 we observe the correct behavior of `minimax` and Algorithm-2 in evaluating $\hat{r}_4$ in L_∞. The extremes alternate signs in full amplification ϵ as expected by Chebyshev theorem [6, pg. 47]. In the contrary the corresponding result by `MiniMaxApproximation` fails as can be seen in Figure-2.

We also tested the ability of functions

1. `minimax` built-in with `Maple`
2. `cosi` implementing Algorithm-2 in `Mathematica` (see Appendix)
3. `MiniMaxApproximation` built-in with `Mathematica`

to produce $\hat{r}_d$ over the interval $[-1, 1]$ for $d = 4,\ 6,\ 8$. We summarized the results in Table-1 and observed that Algorithm-2 is by far the fastest that achieves the minimal.

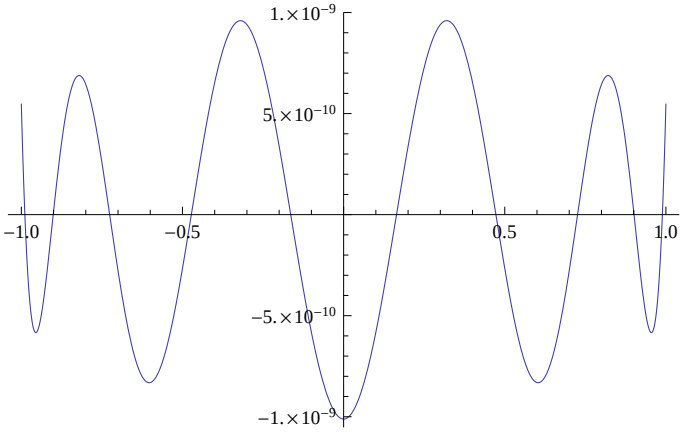

Figure 2: Error of $\hat{r}_4$ approximation by `MiniMaxApproximation` to cos

Table 1: Time in seconds and error for various minimax functions over the interval $[-1, 1]$.

	Maple		Algorithm-2		Mathematica	
d	time	error	time	error	time	error
4	4.4	$7.554 \cdot 10^{-10}$	0.094	$7.554 \cdot 10^{-10}$	0.156	$1.011 \cdot 10^{-9}$
6	7.6	$1.634 \cdot 10^{-15}$	0.469	$1.634 \cdot 10^{-15}$	0.282	$2.187 \cdot 10^{-15}$
8	16.3	$1.095 \cdot 10^{-21}$	0.485	$1.095 \cdot 10^{-21}$	0.469	$1.466 \cdot 10^{-21}$

The rational forms $\hat{r}_d$ that best approximate the function cos over the interval $[-\theta_d, \theta_d]$ can be found. The question that raises now is "how we choose θ_d's?".

We construct randomly 1000 matrices A_j, $j = 1, 2, \cdots, 1000$ with norms in $[0, \theta_d]$. These norms are more densely distributed in the right of the latter interval. We evaluate their true value of $\cos(A_j)$ along with their approximations $\hat{r}_d(A_j)$. θ_d is chosen such that

$$\max_{j=1,2,\cdots,1000} ||\hat{r}_d(A_j) - \cos(A_j)||_\infty \leq 10^{-16}.$$

The discussion above for the need for a fast and accurate algorithm is profound. The values found for normal or nonnegative matrices are given in Table-2. The corresponding values for Padé approximants [3, Table 3.1] are also given in this table. All $\hat{r}_d$'s derived have denominators with positive coefficients posing no problems in any area of the intervals of their application. We remark here that in order the functions $\hat{r}_d$ to be applicable to floating point arithmetic used in packages like MATLAB, then we have to truncate the coefficients derived by Algorithm-2 to 16 significant digits. The results in Table-2 were computed after this truncation was made.

Table 2: Bounds for Pade and L_∞ approximations.

	values of θ_d		
d	Pade	Normal	Nonnegative
2	0.006	0.011	0.01
4	0.11	0.22	0.17
6	0.43	0.85	0.65
8	0.98	2.0	1.5
12	2.6	5.3	3.8
16	4.7	7.6	6.7
20	7.1	–	–

4. Numerical experiments

Testing of the new algorithm was performed in MATLAB 7.4 in IEEE double precision arithmetic. We used a set of 52 test matrices that are built-in with MATLAB. Fourteen of these matrices are nonnegative non-normal and eighteen are normal. Thus 32 matrices of the total 52 are of interest in our work. The dimensions tried were $n = 5$ and $n = 10$. For comparison, we also applied `cosm` function from Matrix Computation Toolbox [5] that implements the variable order Padé approximation [3]. We evaluated the absolute error $||\hat{C} - C||_2$ where $\hat{C}$ is the computed approximation to C and the exact $C = \cos(A)$ is computed in 50 significant decimal digit arithmetic using MATLAB′s Symbolic Math Toolbox.

Table 3: Results for nonnegative, non-normal matrices, $n = 5$

	matrix	4	5	6	9	11	12	18
digits of	Pade	15.1	13.8	14.7	15.3	15.6	14.6	15.4
error	L_∞	15.4	13.8	15.1	15.2	15.0	14.3	15.5
multipli-	Pade	7	9	8	7	6	9	6
cations	L_∞	7	8	7	6	5	8	6

	matrix	24	33	35	43	45	49	52
digits of	Pade	15.4	15.2	15.3	14.0	15.4	14.0	15.2
error	L_∞	15.5	15.2	15.3	14.2	15.2	14.2	15.0
multipli-	Pade	7	7	7	11	7	11	7
cations	L_∞	6	7	6	11	6	11	6

For nonnegative matrices after interpreting Table-3 we observe that the same accuracies were achieved but our $\hat{r}_d$'s were able in 9 of the 14 cases to save a matrix multiplication. The same improvement was recorded in Table-4 for higher dimension matrices.

The results for normal matrices are even more pleasant. For $n = 5$ we found that in 16 of 18 cases we gained a multiplication without loss in accuracy. For

$n = 10$ this gain was raised to 17 matrices over 18 in total. In the latter case and for matrices numbered 17 and 42, no error was recorded since these matrices share extremely large norms.

Table 4: Results for nonnegative, non-normal matrices, $n = 10$

	matrix	4	5	6	9	11	12	18
digits of	Pade	14.5	7.8	14.1	14.8	15.3	13.3	15.4
error	L_∞	14.6	7.8	14.3	14.7	13.8	13.3	15.5
multipli-	Pade	8	11	9	7	6	11	6
cations	L_∞	8	10	8	7	5	10	6

	matrix	24	33	35	43	45	49	52
digits of	Pade	15.1	14.8	15.0	11.4	14.7	11.4	14.7
error	L_∞	15.3	14.9	15.1	11.0	14.7	11.0	14.6
multipli-	Pade	7	8	8	14	8	14	8
cations	L_∞	7	7	7	14	7	14	7

Table 5: Results for normal matrices, $n = 5$.

	matrix	1	7	10	17	20	22	25	26	27
digits of	Pade	15.5	11.6	14.7	3.6	15.4	15.2	14.1	14.8	15.4
error	L_∞	15.2	11.6	14.8	4.7	15.3	15.2	13.6	14.7	15.3
multipli-	Pade	6	12	8	27	6	7	9	8	6
cations	L_∞	5	11	7	26	6	6	8	7	5

	matrix	29	30	32	37	39	41	42	44	47
digits of	Pade	14.5	15.4	15.5	15.5	14.8	15.6	6.1	13.3	15.5
error	L_∞	14.5	15.4	15.3	15.4	14.8	15.5	6.4	12.9	15.5
multipli-	Pade	8	6	6	6	7	6	24	12	6
cations	L_∞	7	5	5	5	6	5	23	11	6

5. Conclusion

A new method for the computation of the matrix cosine is presented. Our method relies on Remez algorithm for rational approximation while the innovation here is the choice of the starting set of non-symmetrical Chebyshev points. Only one Remez iteration is then enough to get infinitesimally close to the required L_∞ approximant. The proposed algorithm, Algorithm-2, has proven to be advantageous over the Padé approximation (see Higham [3]), in terms of digits of error as well as in number of multiplications.

Table 6: Results for normal matrices, $n = 10$.

	matrix	1	7	10	17	20	22	25	26	27
digits of	Pade	14.8	12.2	13.1	$--$	14.8	14.5	13.0	14.0	15.1
error	L_∞	14.9	12.2	13.5	$--$	14.8	14.3	13.1	13.7	15.0
multipli-	Pade	6	12	10	66	7	8	11	10	6
cations	L_∞	5	11	9	65	6	7	10	9	5

	matrix	29	30	32	37	39	41	42	44	47
digits of	Pade	14.3	15.0	14.9	14.7	14.3	14.8	$--$	6.7	14.3
error	L_∞	14.0	14.9	14.9	14.7	14.4	14.7	$--$	7.0	14.3
multipli-	Pade	9	6	6	6	8	6	48	21	8
cations	L_∞	8	5	5	5	7	6	47	20	7

Acknowledgments

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: ARCHIMEDES III. Investing in knowledge society through the European Social Fund.

Appendix

Next we present the `Mathematica` package that computes the cosine as rational approximant of even power polynomials. Notice that we don't solve the linear system of the Remez iteration. Instead we find the minimum of the sum of squares of the equations involved.

As inputs we insert,
order: The value d of the rational form $\hat{r}_d$.
bound: The bounds of the interval $[-\theta, \theta]$ where we want a $\hat{r}_d$ that best approximates cos.

In the output we get $\hat{r}_d$.

```
cosi[order_, bound_] := Module[{x0, y0, j1, a, b, res, temp, x1, uu, xx},
(*-- the initial points --*)
x0 = Table[Cos[Pi*j1/6/order], {j1, 6*order - 1, 1, -4}]*bound;y0 = Cos[x0];

(*-- the initial least squares rational approximant temp --*)
res = FindFit[Transpose[{x0, y0}],
          Sum[a[j1]*x^j1, {j1, 0, order, 2}]/(1 + Sum[b[j1]*x^j1, {j1, 2, order, 2}]),
              Join[Table[a[j1], {j1, 0, order, 2}], Table[b[j1], {j1, 2, order, 2}]],
              x, NormFunction -> (Norm[#, 2] &), {WorkingPrecision -> 33,
              MaxIterations -> 200}
              ];
temp = Sum[a[j1]*x^j1, {j1, 0, order, 2}]/(1 + Sum[b[j1]*x^j1, {j1, 2, order, 2}])
       /. res;
```

```
(*-- compute the extremes xx over the interval [-bound,bound] --*)
x1 = Table[Cos[Pi*j1/6/order], {j1, 6*order, 1, -2}]*bound;     (* find critical points *)
uu = Sort[Table[FindRoot[D[temp - Cos[x], x] == 0, {x, x1[[j1]]}, WorkingPrecision -> 33],
                                          {j1, 1, Length[x1]}][[All, 1, 2]], #1 < #2 &];
                                                          (* exclude saddle points *)
xx = Transpose[Select[Transpose[{uu, Join[uu[[2 ;; Length[uu]]], {1821}]}],
                                                    Abs[#[[1]] - #[[2]]] > .01 &]][[1]];
                                                             (* add boundaries *)
xx = Insert[Insert[xx, -bound - 10^-28, 1], bound - 10^-28, -1] + 10^-28;

(*-- perform a Remez iteration --*)
Return[Sum[a[j1]*x^j1, {j1, 0, order, 2}]/(1 + Sum[b[j1]*x^j1, {j1, 2, order, 2}]) /.
       FindMinimum[Total[Table[(Sum[a[j1]*xx[[j2]]^j1, {j1, 0, order, 2}]/
                               (1 + Sum[b[j1]*xx[[j2]]^j1, {j1, 2, order, 2}]) -
                                Cos[xx[[j2]]] + (-1)^j2*eps)^2, {j2, 1, Length[xx]}]
                         ],
                   Join[Transpose[{Join[Table[a[j1], {j1, 0, order, 2}],
                                        Table[b[j1], {j1, 2, order, 2}]
                                       ], res[[All, 2]]}
                                ], {{eps, 0}}
                       ], WorkingPrecision -> 31, MaxIterations -> 500
                  ][[2]]
       ]
                            ];
```

References

[1] E. Defez, J. Sastre, J.J. Ibáñez, P.A. Ruiz, *Computing matrix functions arising in engineering models with orthogonal matrix polynomials*, Math. Comput. Model., in press. http://dx.doi.org/10.1016/j.mcm.2011.11.022.

[2] E. Defez, J. Sastre, J. Ibáñez, P. Ruiz, J.C. Cortés. *Solving engineering models using matrix functions*, Modelling for Engineering and Human Behaviour 2011, Valencia, Spain, Sept. 69, 2011, 1-17.

[3] G.I. Hargreaves, N.J. Higham, *Efficient algorithms for the matrix cosine and sine*, Numer. Algorithms **40** (2005) 383-400.

[4] N.J. Higham, M.I. Smith, *Computing the matrix cosine*, Numer. Algorithms **34** (2003) 13-26.

[5] N.J. Higham, The Matrix Computation Toolbox, http://www.ma.man.ac.uk/higham/mctoolbox.

[6] J. M. Muller, Elementary Functions: Algorithms and Implementation, Birkhäuser, Boston, 2006.

[7] E. Remez, *Sur un procédé convergent d'approximations successives pour d'eterminer les polynômes d'approximation*, C.R. Acad. Sci., Paris, **198** (1934) 2063-2065.