

On the computation of High Order Pseudospectral Derivatives

Bruno Costa ^{a,2} Wai Sun Don ^{b,1}

^a*Departamento de Matemática Aplicada, IM-UFRJ, Caixa Postal 68530, Rio de Janeiro, RJ, C.E.P. 21945-970, Brazil. E-Mail: bcosta@labma.ufrj.br*

^b*Division of Applied Mathematics, Brown University, Providence, Rhode Island 02912. E-Mail: wsdon@hydra.cfm.brown.edu*

Abstract

A study is presented on the computation of pseudospectral differentiation matrices for higher derivatives using the general Lagrangian polynomial interpolation formulation. The diagonal elements of the differentiation matrices are computed as the negative row sum of the off-diagonal elements and we show why this technique should be used instead of the explicit formula that is usually given in the literature. An efficient recursive algorithm for computing the higher order differentiation matrices are derived. For the Even-Odd Decomposition Algorithm a similar efficient recursive algorithm is also provided. The Chebyshev and Legendre collocation methods commonly used in applications are one of the special case.

Key words: Lagrangian interpolation, High Order Differentiation Matrix, Roundoff error

1 Introduction

When solving Partial Differential Equations via pseudospectral methods (see Gottlieb and Orszag [2] or Canuto, et. al. [3]) care must be taken with the roundoff error issue when higher derivatives or a large number of points N are involved. For instance, the utilization of Chebyshev collocation methods incurs in a roundoff error of order $O(N^{2k}\epsilon)$, where k is the order of the PDE and ϵ is the machine zero. It is obvious that this can ruin the computed solution even if k and N are not large (see [1]).

¹ Supported by grant from AFOSR 9620-96-1-0150

² Supported by grant from NSF DMS-9705229

In [4], Don and Solomonoff showed how to compute higher derivatives with an optimal roundoff error for the Chebyshev collocation method, although the explicit formulae for the differentiation matrices became difficult and complicated to evaluate as k increased. The works of Huang and Sloan [9] and Welfert [10] provided recursive formulae on k for the computation of the differentiation matrices of various types of collocation points, however not much attention was devoted to the efficiency and accuracy achieved in the evaluation of the matrices elements. In this paper, we present efficient and accurate recursive algorithms for evaluating the entries of the differentiation matrices for higher derivatives using the well known Lagrangian Interpolation formulation. The Chebyshev and Legendre collocation methods commonly used in applications are one of the special case.

In section 2, an introduction to the Lagrangian interpolation formula and its derivative is given. We also suggest an old technique for computing the diagonal elements of the differentiation matrices provided that the off-diagonal elements are also computed accurately. An explanation is given in section 3 for why this technique should be used instead of the explicit formula given in the literature. The recursive formula for the differentiation matrices of higher order is given in section 4. Since the Even-Odd Decomposition Algorithms (see [7]) provide substantial saving in storage and operation count, a recursive formula for computing the Even and Odd differentiation matrices of higher order are also derived in section 5.

2 Lagrangian Interpolation and Differentiation

In this section, we introduce the well known Lagrangian Interpolation formula and its differentiation. The reader is referred to [5] for details.

Let $\{-1 = x_0 < x_1 < \dots < x_{N-1} < x_N = 1\}$ be a set of $N + 1$ distinct collocation points and $I(u)(x)$ be a general Lagrangian interpolation polynomial of degree N interpolating the function $u(x)$ at x_j , i.e.

$$I(u)(x) = \sum_{j=0}^N u(x_j)g_j(x) \quad , \quad g_j(x) = \frac{P(x)}{P'(x_j)(x - x_j)} \quad , \quad (1)$$

with

$$P(x) = \prod_{j=0}^N (x - x_j) \quad \text{and} \quad P'(x) = \sum_{n=0}^N \prod_{l=0, l \neq n}^N (x - x_l) \quad . \quad (2)$$

$g_j(x)$ is a polynomial of degree N and $g_j(x_k) = \delta_{kj}$, $I(u)(x_j) = u(x_j)$ for $j = 0, \dots, N$.

Differentiation :

To compute the first derivative of the $I(u)$ at x_k , the general Lagrangian interpolation formula (1) is differentiated yielding

$$\frac{d}{dx}I(u)(x_k) = \sum_{j=0}^N d_{kj}u(x_j) = D\vec{u} \quad , \quad (3)$$

where $d_{kj} = g'_j(x_k)$. D is the $(N + 1) \times (N + 1)$ differentiation matrix and \vec{u} is the vector with $\vec{u}_k = I(u)(x_k)$.

If we denote

$$c_k = P'(x_k) = \prod_{l=0, l \neq k}^N (x_k - x_l) \quad , \quad (4)$$

then the first derivative of $g_j(x)$ gives

$$g'_j(x) = \frac{1}{x - x_j} \left[\frac{P'(x)}{c_j} - g_j(x) \right] \quad . \quad (5)$$

The entries of the differentiation matrix D becomes

$$d_{kj} = \begin{cases} \frac{c_k}{c_j}(x_k - x_j)^{-1} & k \neq j \\ \sum_{l=0, l \neq k}^N (x_k - x_l)^{-1} & k = j \end{cases} \quad , \quad (6)$$

where the diagonal elements are computed using the L'Hospital rule on (5).

Remark :

For large value of N , roundoff error will build up in the computation of the ratio $\frac{c_k}{c_j}$ in (6) due to the fact that the terms $x_k - x_l$ in c_k are being accumulated within the register of the CPU. To avoid this problem, the ratio $\frac{c_k}{c_j}$ should be computed as

$$b_k = \sum_{l=0, l \neq k}^N \ln(|x(k) - x(l)|) \quad , \quad \frac{c_k}{c_j} = (-1)^{k+j} e^{b_k - b_j} \quad . \quad (7)$$

The cutoff value of N in which this method for computing $\frac{c_k}{c_j}$ should be used depends on the internal real data model of the computer. For instance, the IBM SP2 with internal double precision has the smallest representation of order $O(10^{-307})$, the critical N would be around 512-640 for the Legendre-Gauss-Lobatto points.

3 Computation of the Diagonal Elements

For specific types of collocation points a closed form formula for the diagonal elements of the differentiation matrix can be obtained explicitly. For example, for the Chebyshev-Gauss-Lobatto collocation points, the diagonal elements are given by

$$d_{00} = -d_{NN} = \frac{(2N^2 + 1)}{6}, \quad d_{kk} = -\frac{x_k}{2(1 - x_k^2)}, \quad k = 1, \dots, N - 1 \quad .$$

In this section we study a technique to compute the diagonal elements $d_{kk}^{(m)}$ of the differentiation matrix of order m , which consists of enforcing the fact that the row sum of the differentiation matrices must be zero [6], i.e.

$$\sum_{j=0}^N d_{kj}^{(m)} = 0 \quad , \quad k = 0, \dots, N \quad m \geq 1 \quad , \quad (8)$$

where the superscript (m) denotes the differentiation matrix for the m -th derivative.

From this, the diagonal element $d_{kk}^{(m)}$ can be computed as

$$d_{kk}^{(m)} = - \sum_{j=0, j \neq k}^N d_{kj}^{(m)} \quad , \quad k = 0, \dots, N \quad m \geq 1 \quad . \quad (9)$$

Using this technique, Baltensperger et al. [8] showed numerically that the roundoff error growth for Chebyshev collocation methods is of order $O(N^2\epsilon)$, where ϵ is the machine zero. Similar results for the Legendre collocation methods was also shown by Baltensperger et al. (private communications). Table I shows the absolute maximum error for the first and second derivative of $u(x) = \sin(2x)$ using the Standard Legendre Collocation (SLC) methods, Lagrangian formulation (6) and Lagrangian formulation (6) with the diagonal elements being computed according to (9). All three algorithms are based on

Table I
 l_∞ error for the m -th derivative of $\sin(2x)$ using Legendre-Gauss-Lobatto points.

N	$m = 1$			$m = 2$		
	(SLC)	(6)	(6)&(9)	(SLC)	(6)	(6)&(9)
32	0.12E-10	0.47E-12	0.44E-13	0.46E-8	0.66E-10	0.38E-10
64	0.28E-10	0.95E-11	0.74E-12	0.52E-7	0.23E-7	0.10E-8
128	0.68E-8	0.10E-9	0.16E-10	0.36E-4	0.87E-6	0.59E-7
256	0.24E-6	0.16E-8	0.54E-11	0.51E-2	0.28E-4	0.51E-6
512	0.83E-6	0.23E-7	0.44E-9	0.58E-1	0.21E-2	0.20E-4
1024	0.19E-4	0.46E-6	0.54E-9	0.65E+1	0.78E-1	0.16E-3

the Legendre-Gauss-Lobatto points. It is clear from the table that, among all three algorithms, the Lagrangian formulation (6) with equation (9) is the most accurate formulation while the Standard Legendre Collocation methods (SLC) is the worst.

The usual rationale for using this technique is that the differentiation of a constant function must be equal to 0. However, the effects of applying the formula (9) on functions other than the constant have yet to be studied. Numerical experiments with several test functions seems to support the use of this technique. As far as we know, no satisfactory proof of the validity of the technique has been given in the literature. Here, we submit a simple explanation of why and how it work.

Let's define a new function $h_n(x) = u(x) - C_n$ and $C_n = u_n = u(x_n)$ is a constant depending on n . The function $h_n(x)$ has the property that $h_n(x_n) = 0$.

For each fixed n and $k = 0, \dots, N \quad m \geq 1$, we have

$$\begin{aligned}
 \frac{d^m}{dx^m} h_n(x_k) &= \sum_{j=0}^N d_{kj}^{(m)} h_n(x_j) = \sum_{j=0, j \neq n}^N d_{kj}^{(m)} h_n(x_j) \quad (\text{since } h_n(x_n) = 0) \\
 &= \sum_{j=0, j \neq n}^N d_{kj}^{(m)} u(x_j) - \sum_{j=0, j \neq n}^N d_{kj}^{(m)} u(x_n) \\
 \\
 \frac{d^m}{dx^m} u(x_k) &= \sum_{j=0}^N d_{kj}^{(m)} u(x_j) \\
 &= \sum_{j=0, j \neq n}^N d_{kj}^{(m)} u(x_j) + d_{kn}^{(m)} u(x_n)
 \end{aligned}$$

Since subtracting a constant from $u(x)$ does not alter its derivatives,

$$h_n^{(m)}(x_k) = u^{(m)}(x_k) \iff d_{kn}^{(m)} = - \sum_{j=0, j \neq n}^N d_{kj}^{(m)} \quad \text{or} \quad \sum_{j=0}^N d_{kj}^{(m)} = 0 \quad .$$

for any function $u(x)$ including the constant function.

The advantages for using (8) are, namely, the reduction of roundoff error and convenience formula for computing the diagonal elements over the traditional formula.

The magnitudes of $d_{kn}^{(m)}$ are largest at one of these indexes $n = k - 1, k, k + 1$. By choosing $n = k$, the diagonal elements can be computed using (9). Since $h_k(x_k) = 0$, the large roundoff error contribution from the diagonal elements becomes zero.

Hence, as long as the off-diagonal elements of the differentiation matrix are computed accurately, the diagonal elements computed according to (9) yield better roundoff error in the differentiation process.

With this, a greatly simplified, accurate and efficient algorithm can be derived for derivatives of polynomial interpolation as shown in next section.

4 High Order Derivatives Formulae

To compute the m -th derivative of the $I(u)$ at x_j , the general Lagrangian interpolation formula (1) is differentiated m times yielding

$$\frac{d^m}{dx^m} I(u)(x_k) = \sum_{j=0}^N d_{kj}^{(m)} u(x_j) = D^{(m)} \vec{u} \quad , \quad (10)$$

where $d_{kj}^{(m)} = g_j^{(m)}(x_k)$. $D^{(m)}$ is the $(N + 1) \times (N + 1)$ differentiation matrix of order m and \vec{u} is the vector $\vec{u}_k = I(u)(x_k)$.

The expression in (5) can be easily generalized to yield a recursive formula for derivatives of higher order, namely

$$g_j^{(m)}(x) = \frac{1}{x - x_j} \left[\frac{P^{(m)}(x)}{c_j} - m g_j^{(m-1)}(x) \right] \quad . \quad (11)$$

Using the L'Hospital rule, it also yields

$$g_k^{(m)}(x_k) = \frac{1}{c_k(m+1)} P^{(m+1)}(x_k) \quad \text{or} \quad P^{(m)}(x_k) = mc_k g_k^{(m-1)}(x_k) \quad . \quad (12)$$

Off-Diagonal Elements : $k \neq j$

$$\begin{aligned} g_j^{(m)}(x_k) &= \frac{1}{x_k - x_j} \left[\frac{P^{(m)}(x_k)}{c_j} - m g_j^{(m-1)}(x_k) \right] \\ &= \frac{m}{x_k - x_j} \left[\frac{c_k}{c_j} g_k^{(m-1)}(x_k) - g_j^{(m-1)}(x_k) \right] \quad . \end{aligned}$$

Finally, using (6), we obtain

$$d_{kj}^{(m)} = m \left[d_{kk}^{(m-1)} d_{kj} - (x_k - x_j)^{-1} d_{kj}^{(m-1)} \right] \quad , \quad k \neq j \quad . \quad (13)$$

This algorithm is essentially the same as the one derived in [9,10], however, the derivation and resulting recursive formula of [9] are not as clear as the one given here, since there the diagonal elements are the first ones to be computed, turning the process of generating the matrices more complicated.

Diagonal Elements : $k = j$

In order to compute the diagonal elements of the m -th order derivative matrix $d_{kk}^{(m)}$, one needs to compute the $(m+1)$ -th derivative of the polynomial $P(x)$ in equation (12), which is not an easy task to be accomplished. In [10], the diagonal elements $k = j$ are computed first and later used for generating the off-diagonal elements. Here we proposed instead *to compute first the off-diagonal elements and then use the row sum formula (9) to compute the diagonal elements.*

Remark :

- An alternative expression for computing the diagonal elements is simply $d_{kk}^{(m)} = \sum_{j=0}^N d_{kj}^{(1)} d_{jk}^{(m-1)}$ as in [9, eqn. 3.13]. This is not an accurate formulation since it involves product of form like $(x_k - x_j)^{-1} (x_k - x_j)^{-1} \dots$ resulting in large roundoff error for large N and m .
- Note that the recursive formula for computing high order differentiation matrices $d_{kj}^{(m)}$, $m > 1$ involves only $\{x_k, d_{kj}, d_{kj}^{(m-1)}\}$.
- If the grid points x_k are the Chebyshev (Gauss, Radau or Lobatto) points, a trigonometric identity should be used for computing the terms $x_k - x_j$ (see [1]). This is to avoid roundoff error due to the subtraction of two numbers which are very close to each other.

- The (anti)-centrosymmetry property of the differentiation matrices

$$d_{N-k, N-j}^{(m)} = (-1)^m d_{kj}^{(m)} \quad , \quad k = 0, \dots, \frac{N}{2} \quad j = 0, \dots, N$$

should also be used when applicable (see [1]).

Algorithm :

- For $m = 1$, compute the elements of the first differentiation matrix d_{kj} as accurate as possible by making use of (6) and (9).
- For $m > 1$, (13) is used to compute the Off-Diagonal elements $d_{kj}^{(m)}$, $k \neq j$ while (9) is used to compute the Diagonal elements $d_{kk}^{(m)}$ of the m -th order differentiation matrices .

5 The Even Odd Decomposition Algorithm

In this section, a formula is derived for recursively obtaining the high order differentiation matrices of the Even-Odd Decomposition (EOD) Algorithm when the first order differentiation matrices ($m = 1$) $E^{(1)}$ and $O^{(1)}$ have been already computed. For a detailed discussion of the Even-Odd Decomposition Algorithm, see [7].

The EOD Algorithm is composed of three steps:

For $k = 0, \dots, \frac{N}{2}$

- the vector $\{u_k\}$ is decomposed into its even $\{e_k\}$ and odd $\{o_k\}$ parts:

$$e_k = \frac{1}{2}(u_k + u_{N-k}) \quad , \quad o_k = \frac{1}{2}(u_k - u_{N-k}) \quad .$$

- the m -th derivatives of $\{e_k\}$ and $\{o_k\}$ are computed by multiplying them with matrices $E^{(m)}$ and $O^{(m)}$ with entries $E_{kj}^{(m)}$ and $O_{kj}^{(m)}$ respectively, as

$$e_k^{(m)} = \sum_{j=0}^{\frac{N}{2}} E_{kj}^{(m)} e_j \quad , \quad o_k^{(m)} = \sum_{j=0}^{\frac{N}{2}} O_{kj}^{(m)} o_j \quad ,$$

where $E_{kj}^{(m)}$ and $O_{kj}^{(m)}$ are obtained from the elements $d_{kj}^{(m)}$ of the original differentiation matrix.

- $\{u_k^{(m)}\}$ is constructed from $\{e_k^{(m)}\}$ and $\{o_k^{(m)}\}$:

$$u_k^{(m)} = (o_k^{(m)} + e_k^{(m)}) \quad , \quad u_{N-k}^{(m)} = (-1)^{m-1} (o_k^{(m)} - e_k^{(m)}) \quad .$$

Even and Odd Differentiation Matrices : $\{E^{(m)}, O^{(m)}\}$

In the following discussion, N is assumed to be an odd number. The statements pertaining to the case of even N will be enclosed within $\{ \}$.

Let's denote

$$l = N - j \quad , \quad N_2 = \begin{cases} \frac{N}{2} & \text{if } N \text{ is odd} \\ \frac{N}{2} - 1 & \text{if } N \text{ is even} \end{cases} \quad ,$$

then the EOD Algorithm yields two $\left(\frac{N}{2} + 1 \times \frac{N}{2} + 1\right)$ differentiation matrices $E^{(m)}$ and $O^{(m)}$, with entries

$$\begin{aligned} E_{kj}^{(m)} &= d_{kj}^{(m)} + d_{kl}^{(m)} \\ O_{kj}^{(m)} &= d_{kj}^{(m)} - d_{kl}^{(m)} \end{aligned} \quad , \quad k = 0, \dots, \frac{N}{2} \quad j = 0, \dots, N_2 \quad m \geq 1 \quad . \quad (14)$$

{ If N is an even number,

$$\begin{aligned} E_{k\frac{N}{2}}^{(m)} &= d_{k\frac{N}{2}}^{(m)} \\ O_{k\frac{N}{2}}^{(m)} &= 0 \end{aligned} \quad , \quad k = 0, \dots, \frac{N}{2} \quad m \geq 1 \quad . \quad (15)$$

}

The straightforward way of computing $E_{kj}^{(m)}$ and $O_{kj}^{(m)}$ is to form the entries of the full differentiation matrices $d_{kj}^{(m)}$ using the methods discussed in Sections 2 and 4 along with equations (14) and (15). This would require extra storage and computation.

The most efficient way is to form those matrices recursively as with the full matrices.

For $m = 1$:

For the Off-Diagonal Elements $j, k = 0, \dots, \frac{N}{2}$, $k \neq j$ equations (6) is used to compute $d_{kj}^{(1)}$ and $d_{k, N-j}^{(1)}$ (make use of the fact that $x_{N-j} = -x_j$), equations (14) and (15) can then be used to compute $E_{kj}^{(1)}$ and $O_{kj}^{(1)}$.

The Diagonal elements $E_{kk}^{(1)}$ and $O_{kk}^{(1)}$ can be computed using (20) and (21) respectively as discussed later in this section (see Diagonal Elements below).

For $m > 1$:

We define

$$F_{kj}^{(m)} = d_{kj}^{(m)} + \delta d_{kl}^{(m)} \quad , \quad \beta_{kj} = (x_k - x_j)^{-1} \quad , \quad \beta_{kl} = (x_k + x_j)^{-1} \quad (16)$$

where

$$F = \begin{cases} E \\ O \end{cases} \quad , \quad \delta = \begin{cases} 1 & \text{if } F = E \\ -1 & \text{if } F = O \end{cases} \quad .$$

Off-Diagonal Elements : $k \neq j$

From equation (13), we have

$$d_{kj}^{(m)} = m \left[d_{kk}^{(m-1)} d_{kj} - \beta_{kj} d_{kj}^{(m-1)} \right] \quad . \quad (17)$$

Thus, applying (17) to $F_{kj}^{(m)}$ in (16), we obtain

$$F_{kj}^{(m)} = m \left[d_{kk}^{(m-1)} F_{kj} - \left(\beta_{kj} d_{kj}^{(m-1)} + \delta \beta_{kl} d_{kl}^{(m-1)} \right) \right] \quad , \quad (18)$$

for $k = 0, \dots, N_2$, $j = 0, \dots, \frac{N}{2}$.

{ If N is an even number, the term for $k = \frac{N}{2}$ must be computed using

$$F_{kj}^{(m)} = m \left[E_{kk}^{(m-1)} F_{kj} - \left(\beta_{kj} d_{kj}^{(m-1)} + \delta \beta_{kl} d_{kl}^{(m-1)} \right) \right] \quad , \quad (19)$$

for $k = \frac{N}{2}$, $j = 0, \dots, \frac{N}{2}$. }

Diagonal Elements : $k = j$

Analogously to the high order differentiation matrices in section 4, the diagonal elements can be easily computed using the properties of the row sum of the matrices $E^{(m)}$ and $O^{(m)}$.

For instance, any derivative of the even function 1 vanishes, yielding the property that the sum of the elements of any row of $E^{(m)}$ is equal to zero, i.e., for $k = 0, \dots, \frac{N}{2}$

$$E_{kk}^{(m)} = - \sum_{j=0, j \neq k}^{\frac{N}{2}} E_{kj}^{(m)} \quad , \quad m \geq 1 \quad . \quad (20)$$

The analogous property for the matrix $O^{(m)}$ is the fact that the m -th derivative of the odd function x satisfies, for $k = 0, \dots, N_2$

$$\begin{aligned} O_{kk}^{(1)} &= -\frac{1}{x_k} \left(\sum_{j=0, j \neq k}^{\frac{N}{2}} O_{kj}^{(1)} x_j - 1 \right) \\ O_{kk}^{(m)} &= -\frac{1}{x_k} \sum_{j=0, j \neq k}^{\frac{N}{2}} O_{kj}^{(m)} x_j \quad m > 1 \end{aligned} \quad (21)$$

{ If N is an even number, $O_{kk}^{(m)} = 0 \quad , \quad k = \frac{N}{2}. \}$

Remark :

- $d_{kj}^{(m-1)}$ and $d_{kl}^{(m-1)}$ in equations (18) and (19) should be computed using (14), i.e.

$$\begin{aligned} d_{kj}^{(m-1)} &= \frac{1}{2}(E_{kj}^{(m-1)} + O_{kj}^{(m-1)}) \\ d_{kl}^{(m-1)} &= \frac{1}{2}(E_{kj}^{(m-1)} - O_{kj}^{(m-1)}) \quad . \end{aligned}$$

- The recursion formulation of the EOD differentiation matrices of order m only involves $x_k, E_{kj}, O_{kj}, E_{kj}^{(m-1)}, O_{kj}^{(m-1)}$.

{ If N is an even number,

- Equation (15) must be applied before using equations (20) and (21) to compute the diagonal elements for the first order ($m = 1$) Even and Odd differentiation matrix $E_{kk}^{(1)}$ and $O_{kk}^{(1)}$.
- $O_{k \frac{N}{2}}^{(m)} = 0 \quad , \quad k = 0, \dots, \frac{N}{2}$.
- If m is even (odd), any odd (even) function $f_{\frac{N}{2}} = f_{\frac{N}{2}}^{(m)} = 0$ implies that $O_{\frac{N}{2}j}^{(m)} = 0 \quad (E_{\frac{N}{2}j}^{(m)} = 0) \quad , \quad j = 0, \dots, \frac{N}{2}. \}$

Acknowledgments

The first author was supported by grant number NSF DMS-9705229 and by the research fund of Indiana University. The second author gratefully acknowledges discussions with J. P. Berrut and R. Baltensperger on this subject during the ICOSAHOM'98 conference. Research support for the second author was provided by AFOSR 9620-96-1-0150.

References

- [1] W. S. DON AND A. SOLOMONOFF, *Accuracy and speed in computing the Chebyshev collocation derivative*, SIAM J. Sci. Comp, **16**, No. 6, pp. 1253-1268, 1995
- [2] D. GOTTLIEB AND S. ORSZAG, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.
- [3] C. CANUTO, A. QUARTERONI, M. Y. HUSSAINI AND T. ZANG, *Spectral Methods in Fluid Mechanics*, Springer-Verlag, New York, 1988.
- [4] W. S. DON AND A. SOLOMONOFF, *Accuracy Enhancement for Higher Derivatives using Chebyshev Collocation and a Mapping Technique*, SIAM, Journal of Scientific Computing, **18**, No. 4, 1997
- [5] P. J. DAVIS *Interpolation and Approximation*, Dover Publications Inc. (1975)
- [6] A. BAYLISS, A. CLASS AND B. J. MATKOWSKY, *Roundoff Error in Computing Derivatives using the Chebyshev Differentiation Matrix*, J. Comp. Phy. **116**, pp. 380-383 (1995)
- [7] A. SOLOMONOFF, *A fast algorithm for spectral differentiation*, J. Comput. Phys. **98**, pp. 174-177 (1992)
- [8] R. BALTENSPERGER AND J. P. BERRUT, *The errors in calculating the pseudospectral differentiation matrices for Chebyshev-Gauss-Lobatto points*, Univerité de Fribourg (Suisse), Institut de Mathématiques (1997)
- [9] W. HUANG AND D. M. SLOAN, *The pseudospectral method for solving differential eigenvalue problems*, J. Comp. Phy. **111**, pp. 399-409 (1994)
- [10] B. D. WELFERT, *Generation of pseudospectral differentiation matrices I*, SIAM J. Numer. Anal. **34**, pp. 1640-1657 (1997)