CHEBINT: Operations on multivariate Chebyshev approximations

Koen Poppe *Ronald Cools*

Report TW 603, November 2011

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

CHEBINT: Operations on multivariate Chebyshev approximations

Koen Poppe *Ronald Cools*

Report TW 603, November 2011

Department of Computer Science, K.U.Leuven

Abstract

We detail the implementation of basic operations on multivariate Chebyshev approximations. In most cases, they can be derived directly from well known properties of univariate Chebyshev polynomials. Besides addition, subtraction and multiplication, we discuss integration, indefinite differentiation, indefinite integration and interpolation. The latter three, can be written as matrix-vector products of a structured sparse matrix and the respectively the component vector and function values which might be used in the context of differential equations.

Keywords : Multivariate Chebyshev polynomials, addition, subtraction, multiplication, indefinite differential, indefinite integration, interpolation, structured sparse matrix-vector formulation
MSC : Primary : 65D25, 65D30, 65D32

CHEBINT: Operations on multivariate Chebyshev approximations*

KOEN POPPE and RONALD COOLS

November, 2011

Abstract

We detail the implementation of basic operations on multivariate Chebyshev approximations. In most cases, they can be derived directly from well known properties of univariate Chebyshev polynomials. Besides addition, subtraction and multiplication, we discuss integration, indefinite differentiation, indefinite integration and interpolation. The latter three, can be written as matrix-vector products of a structured sparse matrix and the respectively the component vector and function values which might be used in the context of differential equations.

1 Introduction

In the following, we consider s-dimensional Chebyshev approximations on hyper-rectangular domain $C_s = [-1,1]^s$. Although it is better to work with normalised Chebyshev polynomials, in order to avoid notational clutter, we will resort to the unnormalised ones. These multivariate Chebyshev polynomials are defined as

$$T_{\vec{h}}(\vec{x}) \coloneqq \prod_{r=1}^{s} T_{h_r}(x_r) \quad \text{where} \quad T_0(x) \coloneqq 1 \quad \text{and} \quad T_h(x) \coloneqq \cos\left(h \arccos\left(x\right)\right), \tag{1}$$

and are orthogonal with respect to the continuous scalar product with Chebyshev weight function $\omega(\vec{x}) \coloneqq \pi^{-s} \prod_{r=1}^{s} (1-x_r^2)^{-\frac{1}{2}}$. All polynomials up to a degree n, defined as $|\vec{h}| \coloneqq \sum_{r=1}^{s} |h_r|$, form a basis for the polynomial function space $\mathcal{P}_n^s$. The dimension of this polynomial space with degree n is $\dim(\mathcal{P}_n^s) \coloneqq \binom{s+n}{s}$.

The approximations we consider are linear combinations of the Chebyshev polynomials:

$$A(\vec{x}) \coloneqq \sum_{\vec{h},|\vec{h}|\leq n_A} a_{\vec{h}}\, T_{\vec{h}}(\vec{x}), \tag{2}$$

where the components $a_{\vec{h}}$ can be determined using hyperinterpolation as presented in [3] and calculated efficiently for certain types of Chebyshev lattices using the fast Fourier transform as derived in [6].

It is clear that the tuples $(\vec{h}, a_{\vec{h}})$ contain all information to describe the approximation. Evidently, that is how we will store these approximations.

In the following, several basic operations on Chebyshev approximations will be described in terms of this representation. This allows us to work efficiently with the multivariate representation of a function, as explored for the univariate case in CHEBFUN [7] and described as *symbolics with the speed of numerics*.

Moreover, describing differentiation as matrix operation on a component vector allows us to approximate the solution of specific types of differential equations.

*This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimisation), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its author(s).

2 Basic operations

2.1 Addition/subtraction: $C(\vec{x}) = A(\vec{x}) \pm B(\vec{x})$

Adding or subtracting Chebyshev approximations is rather straightforward from a logical point of view: the components of the resulting approximation are simply the sum or the difference of the corresponding components in both terms. Taking care of the possible difference in degree, the operation can be defined naturally as

$$C(\vec{x}) = A(\vec{x}) \pm B(\vec{x}) \quad \Longleftrightarrow \quad c_{\vec{h}} = \begin{cases} a_{\vec{h}} \pm b_{\vec{h}} & \text{if } |\vec{h}| \le \min(n_A, n_B), \\ a_{\vec{h}} & \text{if } n_B < |\vec{h}| \le n_A, \\ \pm b_{\vec{h}} & \text{if } n_A < |\vec{h}| \le n_B. \end{cases} \tag{3}$$

Matching the components for s-dimensional approximations when the degrees of both terms differ is not that easy from an implementation point of view. One possible strategy is similar to merge sorting: order the components of the approximations based on the lexicographic order of the coefficients $\vec{h}$, then run over both arrays at the same time and recombine/collect the right components. However, discriminating the components based on their degree proved more efficient, mainly due to the absence of explicit `for`-loops, and is therefore used in our MATLAB/Octave implementation:

Addition/subtraction

1. Select the common components of A and B, e.g., where $|\vec{h}| \le \min(n_A, n_B)$
2. Sort the common components based on $\vec{h}$ and add/subtract them
3. If $n_A > n_B$, add the remaining components from A
4. If $n_B > n_A$, add the remaining components from B, with a sign change in case of subtraction

A third option could be to put the components of A and B in a s-dimensional sparse array based on their coefficients. Then, using the addition/subtracting for that data type, the components of C can be found. However, retrieving the resulting coefficients would be less straightforward, not to mention the overhead of mapping s dimensions onto the sparse matrices that are limited to two dimensions in Matlab/Octave.

2.2 Multiplication: $C(\vec{x}) = A(\vec{x})\, B(\vec{x})$

Multiplying two approximations is feasible but possibly very expensive because the number of terms in the approximation increases dramatically. The product of two approximations of degree n increases the number of coefficients from $2\binom{s+n}{s}$ to $\binom{s+2n}{s}$, an increase with a factor of about 2^{s-1}. This operation should thus be avoided if possible because multiplying evaluations is much cheaper than evaluating the explicit product C.

The product operation is based on the one-dimensional identity

$$T_h(x)\, T_g(x) = \frac{1}{2}\, T_{h+g}(x) + \frac{1}{2}\, T_{|h-g|}(x), \tag{4}$$

which follows readily from the trigonometric definition of the univariate Chebyshev polynomials. It is straightforward to generalise this to s dimensions, what leads to

$$T_{\vec{h}}(\vec{x})\, T_{\vec{g}}(\vec{x}) = \frac{1}{2^s} \sum_{\vec{t} \in \{-1,1\}^s} T_{\vec{h}+\vec{t}\circ\vec{g}}(\vec{x}). \tag{5}$$

Applying (5) for all combinations of terms, one from $A(\vec{x})$ and one from $B(\vec{x})$, creates the contributions for the product approximation $C(\vec{x})$.

Note that division is nontrivial, also because it might lead to a rational function.

2.3 Differentiation: $B(\vec{x}) = \frac{\mathbf{d}A(\vec{x})}{\mathbf{d}x_t}$

Due to the product form of the multivariate Chebyshev polynomials, partial differentiation with respect to x_t $(1 \le t \le s)$ is as simple as one-dimensional differentiation. Using

$$\mathsf{d}T_h(x) = \begin{cases} 0 & \text{if } h = 0 \\ T_0(x) & \text{if } h = 1, \\ 4\,T_1(x) & \text{if } h = 2, \\ 2h\,T_{h-1}(x) + \frac{h}{h-2}\,\mathsf{d}T_{h-2}(x) & \text{otherwise,} \end{cases} \tag{6}$$

shows that the derivative of a Chebyshev base function has contributions from all lower degree base functions with the same parity as h. This follows from the alternative form derived by expanding the recursion in (6):

$$\mathsf{d}T_h(x) = \begin{cases} 2h \sum_{\gamma=1}^{\frac{h}{2}} T_{2\gamma-1}(x) & \text{if } h \text{ is even,} \\ 2h \sum_{\gamma=1}^{\frac{h-1}{2}} T_{2\gamma}(x) + hT_0(x) & \text{if } h \text{ is odd.} \end{cases} \tag{7}$$

If we differentiate a univariate Chebyshev approximation $A(x) = \sum_{h=0}^{n} a_h T_h(x)$, the result can be rewritten as $\sum_{h=0}^{n-1} b_h T_h(x)$. Graphically, the differentiated approximation (first column) is expanded in several contributions in terms of the base functions T_h (columns). Summation per base function than leads to the components b_h as depicted in the bottom row:

$\mathsf{d}A$	T_0	T_1	T_2	T_3	T_4	T_5	$\dots$	T_{n-1}
$a_0\,\mathsf{d}T_0$	0	0	0	0	0	0		0
$a_1\,\mathsf{d}T_1$	a_1	0	0	0	0	0		0
$a_2\,\mathsf{d}T_2$	0	$4\,a_2$	0	0	0	0		0
$a_3\,\mathsf{d}T_3$	$3\,a_3$	0	$6\,a_3$	0	0	0		0
$a_4\,\mathsf{d}T_4$	0	$8\,a_4$	0	$8\,a_4$	0	0		0
$a_5\,\mathsf{d}T_5$	$5\,a_5$	0	$10\,a_5$	0	$10\,a_5$	0		0
$a_6\,\mathsf{d}T_6$	0	$12\,a_6$	0	$12\,a_6$	0	$12\,a_6$		0
$\vdots$	$\ddots$		$\ddots$		$\ddots$		$\ddots$	0
$a_n\,\mathsf{d}T_n$								$(2n)\,a_n$
	b_0	b_1	b_2	b_3	b_4	b_5	$\dots$	b_{n-1}

It is clear that the components b_g, the sums per column, can be summarised as

$$b_g = \begin{cases} \sum_{\gamma=1}^{\lceil \frac{n}{2} \rceil} (2\gamma - 1)\,a_{2\gamma-1} & \text{if } g = 0, \\ \sum_{\gamma=\frac{g+1}{2}}^{\lfloor \frac{n}{2} \rfloor} 2(2\gamma)\,a_{2\gamma} & \text{if } g \text{ is odd,} \\ \sum_{\gamma=\frac{g+2}{2}}^{\lceil \frac{n}{2} \rceil} 2(2\gamma - 1)\,a_{2\gamma-1} & \text{if } g \text{ is even.} \end{cases} \tag{8}$$

It should be noted that these sums share trailing terms. Therefore, b_g can also be defined recursively and computed starting from b_{n-1} in order to avoid these duplicate summations

$$b_g = \begin{cases} 0 & \text{if } g \ge n, \\ a_1 + \frac{1}{2}b_2 & \text{if } g = 0, \\ 2\,(g+1)\,a_{g+1} + b_{g+2} & \text{otherwise.} \end{cases} \tag{9}$$

This is a common way of implementing the derivative of a Chebyshev polynomial (see i.e., [5]) and is also how CHEBFUN [7] does it.

Differentiating a multivariate Chebyshev polynomial is straightforward:

$$\frac{\mathsf{d}T_{\vec{h}}(\vec{x})}{\mathsf{d}x_t} = \frac{\mathsf{d}T_{h_t}(x_t)}{\mathsf{d}x_t} \prod_{r=1,\, r\neq t}^{s} T_{h_r}(x_r), \tag{10}$$

so to differentiate a multivariate Chebyshev approximation with respect to x_t, one can group the terms with equal coefficients h_r with $r = \{1, \ldots, s\} \backslash \{t\}$ and apply (9) to the remaining one-dimensional Chebyshev approximation terms.

Implementing the above requires a significant amount of bookkeeping, but by sorting the coefficients on dimensions $\{1, \ldots, s\} \backslash \{t\}$, the one-dimensional approximations are restructered as blocks in the accordingly sorted components vector (illustrated in Figure 1). This reduces the bookkeeping to one sort operation, which is evidently much cheaper than iterating over the coefficients to find the right terms.

Higher order derivatives are implemented as repeated differentiation.

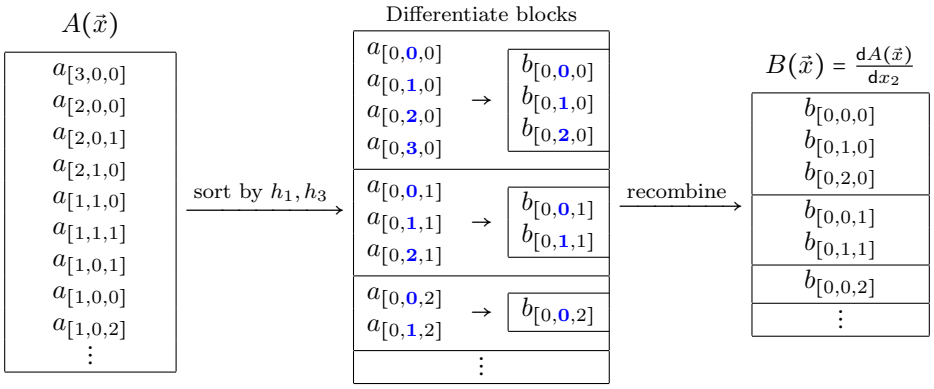


Figure 1: Overview of the differentiation of a Chebyshev approximation $A(\vec{x}) = \sum_{\vec{h}, |\vec{h}| \leq n} a_{\vec{h}} T_{\vec{h}}(\vec{x})$ with respect to x_2. The proposed algorithm sorts the components on h_1 and h_3, thus decomposing the approximation into one-dimensional blocks. Differentiating f is then accomplished by applying the univariate differentiation from (9) to all blocks and a recombination phase to construct $B(\vec{x}) := \frac{\mathrm{d}A(\vec{x})}{\mathrm{d}x_2}$.

2.4 Indefinite integration: $B(\vec{x}) = \int A(\vec{x}) \, \mathbf{d}x_t$

Integration is similar to differentiation because an analogous reasoning allows us to focus only on the integration of a one-dimensional Chebyshev approximation. Note that integrating a Chebyshev approximation increases the degree by one and leaves us with an undetermined arbitrary constant. In the following, we will define that constant so that the commonly used relation $\int (h+1) \, x_t^h \, \mathrm{d}x_t = x_t^{h+1}$ is satisfied.

Based on Chebyshev polynomial identities, the following relation can be derived:

$$\int T_h(x) \, \mathrm{d}x = \begin{cases} T_1(x) & \text{if } h = 0, \\ \frac{1}{4} \left(T_2(x) + T_0(x) \right) & \text{if } h = 1, \\ \frac{T_{h+1}(x)}{2(h+1)} - \frac{T_{h-1}(x)}{2(h-1)} & \text{if } h \geq 2 \text{ and } h \text{ is even}, \\ \frac{T_{h+1}(x)}{2(h+1)} - \frac{T_{h-1}(x)}{2(h-1)} + \frac{(-1)^{\frac{h-1}{2}} \, h}{(h+1)(h-1)} \, T_0(x) & \text{if } h \geq 3 \text{ and } h \text{ is odd}. \end{cases} \tag{11}$$

In contrast to the derivative (7), integrating a single Chebyshev polynomial (11) leads to three terms at most. Writing this in the same tabular form as before, the integrand of an univariate Chebyshev approximation

$A(x) = \sum_h^n a_h T_h(x)$ (first column) expands in the base functions T_h (columns):

$\int A(x)\,\mathrm{d}x$	T_0	T_1	T_2	T_3	T_4	T_5	T_6	$\dots$
$a_0\,\mathrm{d}T_0$	0	a_0	0	0	0	0	0	
$a_1\,\mathrm{d}T_1$	$\frac{a_1}{4}$	0	$\frac{a_1}{4}$	0	0	0	0	
$a_2\,\mathrm{d}T_2$	0	$-\frac{a_2}{2}$	0	$\frac{a_2}{6}$	0	0	0	
$a_3\,\mathrm{d}T_3$	$-\frac{3\,a_3}{8}$	0	$-\frac{a_3}{4}$	0	$\frac{a_3}{8}$	0	0	
$a_4\,\mathrm{d}T_4$	0	0	0	$-\frac{a_4}{6}$	0	$\frac{a_4}{10}$	0	
$a_5\,\mathrm{d}T_5$	$\frac{5\,a_5}{24}$	0	0	0	$-\frac{a_5}{8}$	0	$\frac{a_5}{12}$	
$\vdots$	$\vdots$					$\ddots$		$\ddots$
	b_0	b_1	b_2	b_3	b_4	b_5	b_6	$\dots$

Summing the contributions column by column provides an expression for the components of the approximation that is the indefinite integral. Assuming that $n > 1$,

$$b_g = \begin{cases} \frac{a_1}{4} + \sum_{\gamma=2}^{\lfloor \frac{n+1}{2} \rfloor} \frac{(-1)^{\gamma-1}\,(2\gamma-1)}{(2\gamma)(2\gamma-2)}\,a_{2\gamma-1} & \text{if } g = 0, \\ a_0 - \frac{a_2}{2} & \text{if } g = 1, \\ \frac{a_{g-1}}{2g} - \frac{a_{g+1}}{2g} & \text{if } 2 \le g \le n-1, \\ \frac{a_{g-1}}{2g} & \text{if } n \le g \le n+1. \end{cases} \tag{12}$$

Using the same sorting algorithm as with differentiation, the integration of a multivariate Chebyshev approximation can be decomposed in the integration of univariate blocks. Multiple integration is analogously implemented as repeated integration.

2.5 Interpolation

Interpolation is just a matter of evaluating the Chebyshev approximation. Using the Clenshaw algorithm [2] in one dimension, this can be done efficiently by exploiting the three term recurrence relation for the (unnormalised) Chebyshev polynomials. Evaluating $A(x) = \sum_{h=0}^{n} a_h\,T_h(x)$ boils down to the following algorithm:

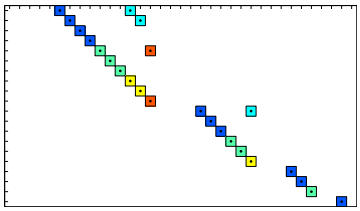
1. $b_{n+2} \coloneqq 0$, $b_{n+1} \coloneqq 0$
2. for $h = n : -1 : 1$, $b_h \coloneqq 2\,x\,b_{h+1} - b_{h+2} + a_h$
3. $A(x) \coloneqq x\,b_1 - b_2 + a_0$

Due to the product nature of the multivariate Chebyshev polynomials, the same algorithm can be used s times to evaluate a s-dimensional approximation.

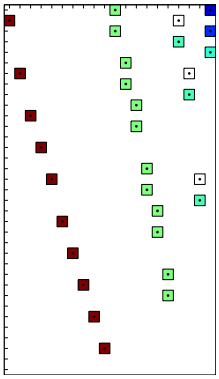
Our implementation uses a vectorised version of the algorithm where the N points are evaluated at once. Also, note that only the last two b_h's must be kept, which reduces the memory requirements from $\mathcal{O}\left(\binom{s+n}{s} N\right)$ to $\mathcal{O}(3\,s\,N)$.

3 Basic operations as matrix-vector product

The unary operations from the previous section can all be written as multiplication of a (sparse) matrix and $\vec{a} = \left[a_{\vec{h}}\right]_{\vec{h}}$, the column vector with the components $a_{\vec{h}}$ of the approximation $A(\vec{x})$. Note that the actual matrices depend on the ordering of the components and that the number of rows in these matrices also differs.



(a) Differentiation matrix $\mathbf{D}_{x_2}$ for a three-dimensional Chebyshev approximation with degree $n = 4$.



(b) Indefinite integration matrix $\mathbf{J}_{x_1}$ for a three-dimensional Chebyshev approximation with degree $n = 3$.

Figure 2: Sparse structure of the matrixes representing operations on Chebyshev approximations.

3.1 Differentiation

Differentiation of an approximation $A(\vec{x})$ of degree n with respect to the variable x_t decreases the degree by (at least) one and can be written as the matrix operation

$$B(\vec{x}) = \frac{\mathsf{d}A(\vec{x})}{\mathsf{d}x_t} \qquad \Leftrightarrow \quad \vec{b} = \mathbf{D}_{x_t}\,\vec{a} \quad \text{with} \quad \mathbf{D}_{x_t} \in \mathbb{R}^{\dim(\mathcal{P}^s_{n-1})\times\dim(\mathcal{P}^s_n)}, \tag{13}$$

where the entries of $\mathbf{D}_{x_t}$ can be calculated using the relations from Section 2.3. An illustration of the sparse structure of the matrix $\mathbf{D}_{x_t}$ is given in Figure 2(a).

3.2 Indefinite integration

A similar reasoning applies for indefinite integration. The resulting matrix relation contains a non-square matrix because the degree increases by one. It is given by

$$B(\vec{x}) = \int A(\vec{x})\,\mathsf{d}x_t \quad \Leftrightarrow \quad \vec{b} = \mathbf{J}_{x_t}\,\vec{a} \quad \text{with} \quad \mathbf{J}_{x_t} \in \mathbb{R}^{\dim(\mathcal{P}^s_{n+1})\times\dim(\mathcal{P}^s_n)}. \tag{14}$$

The sparse structure of $\mathbf{J}_{x_t}$ is shown in Figure 2(b) while the non-zero entries can be derived directly from the relations in Section 2.4.

3.3 Interpolation

In general, interpolation of a function $f(\vec{x})$ with a linear combination of basis functions can be reduced to solving the linear system $\mathbf{P}^T\vec{a} = \vec{f}$ in which $\vec{f} = \left[f(\vec{x}_{\vec{\ell}})\right]_{\vec{\ell}}$, the column vector with the values at f at the interpolation points and $\mathbf{P} \in \mathbb{R}^{m\times N}$ is the matrix containing the evaluations of all m basis functions in the N interpolation points. Exactness for all basis functions requires a certain N. The optimal case is $N = m$ where the system has as many equations as unknowns but in general $N \geq m$. The (overdetermined) system is thus solved in an approximative sense, commonly by minimising the residue $\left\|\mathbf{P}^T\vec{a} - \vec{f}\right\|_2$ leading to the least squares solution from $(\mathbf{P}\,\mathbf{P}^T)\,\vec{a} = \mathbf{P}\,\vec{f}$.

However, due to the discrete orthogonality of the Chebyshev polynomials for specific point sets, $\mathbf{P}\,\mathbf{W}\,\mathbf{P}^T = \mathbf{I}_m$ with $\mathbf{W}$ the matrix with the weights for the points on its diagonal. Multiplying both sides of the original interpolation system with $\mathbf{P}\,\mathbf{W}$ leads to $\mathbf{P}\,\mathbf{W}\,\mathbf{P}^T\vec{a} = \mathbf{P}\,\mathbf{W}\,\vec{f}$ which evidently reduces to $\mathbf{I}\,\vec{a} = \vec{a} = \mathbf{P}\,\mathbf{W}\,\vec{f}$. Interpolation is thus a matrix-vector product. Also, by observing that $\mathbf{W}\,\vec{f} := \vec{f}_w$ is a vector with weighted function values, this proves to be equivalent with the cubature rules for interpolation as considered in [3].

In the specific case of the two-dimensional Padua point set and using the structure of $\mathbf{P}$, the product with $\mathbf{P}$ can be decomposed to improve the efficiency even further as explored in [1]. This publication also derived a fast Fourier transform (FFT) based method for the Padua point set. The latter is extendable to other point sets and higher dimensions as presented in [6].

4 Extra: solving differential equations

Using the above matrix formulations of differentiation operations on Chebyshev approximations, solving linear multivariate differential equations reduces to solving a linear system and boundary conditions similarly reduce to extra equations. This is a natural extension of what was explored for the univariate case with CHEBOPS [4] (now incorporated in CHEBFUN [7]).

Note that some kind of adaptivity is required to attain good results for general problems. CHEBFUN, however limited to the univariate case at the time of writing, provides this functionality: approximations may consist of several Chebyshev approximations that are joined together. This allows for good approximations of, for example, piecewise smooth functions and functions with large scale differences.

A second remark can be made on the requirement for the differential equation to be linear: in specific cases, non-linearities can be allowed. For example a term $x\,f(x,\ldots)$, where $f(x,\ldots)$ is the unknown function, can be formulated as matrix operation using the product operation and thus still results in a linear system.

5 Conclusion

We have presented several operations on multivariate Chebyshev approximations, provided some insight in their implementation in the context of CHEBINT and showed how differentiation, integration and interpolation can be written as a (sparse) matrix-vector product. The last section provided some preliminary insights in how to use these matrix formulations to solve certain types of differential equations, possibly with mild non-linearities.

References

[1] M. Caliari, S. De Marchi, A. Sommariva, and M. Vianello, *Padua2DM: fast interpolation and cubature at the Padua points in Matlab/Octave*, Numerical Algorithms, 56 (2011), pp. 45–60. 10.1007/s11075-010-9373-1.

[2] C. W. Clenshaw, *A note on the summation of chebyshev series*, Math. Tab. Wash., 9 (1955), pp. 118–120.

[3] R. Cools and K. Poppe, *Chebyshev lattices, a unifying framework for cubature with the Chebyshev weight function*, BIT Numerical Mathematics, 51 (2011), pp. 275–288.

[4] T. A. Driscoll, F. Bornemann, and L. N. Trefethen, *The CHEBOP system for automatic solution of differential equations*, BIT Numerical Mathematics, 48 (2008), pp. 701—723.

[5] J. Mason and D. Handscomb, *Chebyshev Polynomials*, 2003.

[6] K. Poppe and R. Cools, *CHEBINT: a MATLAB/Octave toolbox for fast multivariate integration and interpolation based on Chebyshev approximations over a hypercube*. In preparation.

[7] L. N. Trefethen et al., *Chebfun Version 4.0*, The Chebfun Development Team, 2011. http://www.maths.ox.ac.uk/chebfun/.