Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing

Nicolas Brisebarre, Mioara Maria Joldes

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing

Nicolas Brisebarre,
Mioara Joldeş

LIP, Arénaire
CNRS/ENSL/INRIA/UCBL/Université de Lyon
46, allée d'Italie, 69364 Lyon Cedex 07, France
Nicolas.Brisebarre@ens-lyon.fr, Mioara.Joldes@ens-lyon.fr

April 2010

Research Report No RR2010-13

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`




Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing

Nicolas Brisebarre, Mioara Joldeş

LIP, Arénaire

CNRS/ENSL/INRIA/UCBL/Université de Lyon

46, allée d'Italie, 69364 Lyon Cedex 07, France

Nicolas.Brisebarre@ens-lyon.fr, Mioara.Joldes@ens-lyon.fr

April 2010

Abstract

Performing numerical computations, yet being able to provide rigorous mathematical statements about the obtained result, is required in many domains like global optimization, ODE solving or integration. Taylor models, which associate to a function a pair made of a Taylor approximation polynomial and a rigorous remainder bound, are a widely used rigorous computation tool. This approach benefits from the advantages of numerical methods, but also gives the ability to make reliable statements about the approximated function. Despite the fact that approximation polynomials based on interpolation at Chebyshev nodes offer a quasi-optimal approximation to a function, together with several other useful features, an analogous to Taylor models, based on such polynomials, has not been yet well-established in the field of validated numerics. This paper presents a preliminary work for obtaining such interpolation polynomials together with validated interval bounds for approximating univariate functions. We propose two methods that make practical the use of this: one is based on a representation in Newton basis and the other uses Chebyshev polynomial basis. We compare the quality of the obtained remainders and the performance of the approaches to the ones provided by Taylor models.

Keywords: Rigorous Computing, Validated Numerics, Interpolation Polynomial, Chebyshev Polynomials, Taylor Models

1 Introduction

Computers are used nowadays to quickly give numerical solutions to various global optimization, ODE solving or integration problems. However, traditional numeric methods usually provide only approximate values for the solution. Bounds for the approximation errors are only sometimes available, are not guaranteed to be accurate or are sometimes unreliable. In contrast, validated computing aims at providing rigorously verified information about solutions, in order to complete proofs, or to give rigorous mathematical statements about the obtained result.

Interval arithmetic [20] is a classical tool to perform validated computations with floating-point arithmetic. Intervals are well-suited to represent enclosures of real numbers on a machine. However, they propagate only information about function values, and fail to convey much information about the other properties about the function itself. In particular, when modeling functions with interval arithmetic, splitting the domain in subintervals is usually required. For some cases, known as "high dependency problems" [5, 6, 12, 11] the number of necessary subintervals becomes unfeasible.

Taylor models [18, 5, 6], introduced by Berz and his group offer a remedy to this problem. They provide another way to rigorously manipulate and evaluate functions using floating-point arithmetic. They have been widely used for validated computing for global optimization and range bounding [17, 5, 11, 6], solutions of ODEs [22], quadrature [4], etc.

A Taylor model (TM) of order n for a function f which is supposed to be $n+1$ times continuously differentiable over an interval $[a, b]$, is a rigorous polynomial approximation of f. More specifically, it is a couple $(P, \mathbf{\Delta})$ formed by a polynomial P of degree n, and an interval part $\mathbf{\Delta}$, such that $f(x) - P(x) \in \mathbf{\Delta}, \forall x \in [a, b]$. Roughly speaking, the polynomial can be seen as a Taylor expansion of the function at a given point. The interval $\mathbf{\Delta}$ provides the validation of the approximation, meaning that it provides an enclosure of all the approximation errors encountered (truncation, roundings).

A natural idea is to try to replace Taylor polynomials with better approximations such as minimax approximation, Chebyshev truncated series or interpolation polynomials (also called approximate Chebyshev truncated series when the points under consideration are Chebyshev nodes) for instance. The last two kind of approximations are of particular relevance for replacing Taylor polynomials since the series they define converge on domains better shaped for various usual applications than Taylor expansions (see, for instance, Section 2.7 of [9] for a more detailed account). Moreover, we can take advantage of numerous powerful techniques for computing these approximations. So far, the attempts for using these better approximations, in the context of rigorous computing, do not seem to have succeeded, see for example [18] for a comparison of existing techniques.

In this work we propose two approaches for computing models based on interpolation polynomials at Chebyshev nodes, what we call "Chebyshev interpolation models" (CM). The first method is based on Newton Basis and the second on Chebyshev polynomial basis. We believe that bringing a certified remainder to an approximate truncated Chebyshev series and providing effective tools for working with such models, opens the way to adapting to rigorous computing many numerical algorithms based on Chebyshev interpolation polynomials, for rigorous ODE solving, quadrature, etc.

The outline of the paper is the following. We first recall or prove various definitions and results required by our approaches in Section 2. In Section 3, we present in more details TMs and we discuss the use of better polynomial approximations. Then, we introduce the notion

of "Chebyshev interpolation models" in Section 4. The CMs are implemented using multiple precision interval arithmetic in order to perform rigorous computing and yet, for the sake of clarity, we present their implementation in exact arithmetic in Section 5. We give some results and a comparison of our models with TMs in Section 6. We end with a brief conclusion and a mention of our future works on the subject.

2 Some preliminary statements about interpolation and Chebyshev polynomials

We first give a very short reminder on Chebyshev polynomials. A detailed presentation can be found in [7, 28]. Then we state some interpolation results that we use in the sequel.

2.1 Some basic facts about Chebyshev polynomials

Over $[-1,1]$, Chebyshev polynomials can be defined as $T_n(x) = \cos(n \arccos x), n \geq 0$. Since we consider functions over any interval $I = [a,b]$, we define in the following the Chebyshev polynomials over I as $T_n^{[a,b]}(x) = T_n\left(\frac{2x-b-a}{b-a}\right)$.

$T_{n+1}^{[a,b]}$ has $n+1$ distinct real roots in $[a,b]$, called "Chebyshev nodes" since they are of utmost interest for interpolation:

$$x_i^* = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(i+1/2)\,\pi}{n+1}\right), i = 0, \ldots, n. \tag{1}$$

We now recall

Lemma 2.1 *The polynomial $W_{\overline{x^*}}(x) = \prod_{i=0}^{n}(x - x_i^*)$, is the monic degree-$n+1$ polynomial that minimizes the supremum norm over $[a,b]$ of all monic polynomials in $\mathbb{C}[x]$ of degree at most $n+1$. We have*

$$W_{\overline{x^*}}(x) = \frac{(b-a)^{n+1}}{2^{2n+1}} T_{n+1}^{[a,b]}(x)$$

and

$$\max_{x \in [a,b]} |W_{\overline{x^*}}(x)| = \frac{(b-a)^{n+1}}{2^{2n+1}}.$$

2.2 Brief overview of interpolation results used

Let $I = [a,b]$ be an interval. Let f be a function that is at least $n+1$ times continuously differentiable over I. Let $\{y_i, i = 0, \ldots, n\}$ be a set of $n+1$ points in I. There exists a unique polynomial P of degree $\leq n$ which interpolates f at these points [10]: $P(y_i) = f(y_i), \forall i = 0, \ldots, n$, or if y_i is repeated k times, $P^{(j)}(y_i) = f^{(j)}(y_i), \forall j = 0, \ldots, k-1$. If all the points are distinct, this is called Lagrange interpolation. In the extreme case that all the y_i are equal, P is just the Taylor polynomial of f at the considered point.

Several algorithms and interpolation formulae in various basis exist for representing P, for example monomial basis, Lagrange, Newton, Barycentric Lagrange, Chebyshev basis [10, 3, 30]. The numerical properties (stability) of these formulas have been widely studied in the literature [16].

Let us consider the polynomial P in Newton basis: $P(x) = \sum_{i=0}^{n} c_i N_i(x)$, where $N_0(x) = 1$ and $N_i = \prod_{j=0}^{i-1}(x - y_j), i = 1, \ldots, n$. The coefficients c_i are the divided-differences $f[y_0, \ldots, y_i]$ of f at the points $y_0, \ldots, y_i$. As mentioned above, if k points coincide, it suffices to take the successive $k-1$ derivatives of f. Note that c_i can be obtained thanks to the divided-differences algorithm [14, 30]. Moreover, the error between f and P is given [14, 30] by:

$$\forall x \in I, f(x) - P(x) = f[y_0, \ldots, y_n, x] W_{\overline{y}}(x). \tag{2}$$

By a repeated application of Rolle's theorem [10, 14, 30], we can prove that $\forall x \in I,\ \exists \xi \in (a, b)$ s.t.

$$f(x) - P(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) W_{\overline{y}}(x), \tag{3}$$

with $W_{\overline{y}}(x) = \prod_{i=0}^{n}(x - y_i)$. In the following we denote the right member of this formula for the error by $\Delta_n(x, \xi)$. Lemma 2.1 indicates that an optimal choice of interpolation points seems to be the Chebyshev nodes (1).

We will take advantage of the following lemma which generalizes Lemma 5.12 of [32]:

Lemma 2.2 *Under the assumptions on f and y_i above, if $f^{(n+1)}$ is increasing (resp. decreasing) over I, then $f[y_0, \ldots, y_n, x]$ is increasing (resp. decreasing) over I.*

Assume that $f^{(n+1)}$ is increasing. We know (see Chap. 4 of [30] for instance) that, for all $x \in [a, b]$, we have

$$f[y_0, \ldots, y_n, x] = \int_0^1 \int_0^{t_1} \cdots \int_0^{t_n} f^{(n+1)}(y_0 + t_1(y_1 - y_0) + \cdots + t_{n+1}(x - y_n)) \mathrm{d}t_1 \cdots \mathrm{d}t_{n+1}.$$

Let x, $y \in [a, b]$, $x \leq y$, let Z_n denote $y_0 + t_1(y_1 - y_0) + \cdots + t_n(y_n - y_{n-1}) - t_{n+1} y_n$, we notice that

$$f[y_0, \ldots, y_n, y] - f[y_0, \ldots, y_n, x] = \int_0^1 \int_0^{t_1} \cdots \int_0^{t_n} \left(f^{(n+1)}(Z_n + t_{n+1} y) - f^{(n+1)}(Z_n + t_{n+1} x) \right) \mathrm{d}t_1 \cdots \mathrm{d}t_{n+1}.$$

Since $t_{n+1} \geq 0$, we have $Z_n + t_{n+1} y \geq Z_n + t_{n+1} x$. As $f^{(n+1)}$ is increasing, it follows that the integrand is nonnegative, which implies $f[y_0, \ldots, y_n, y] \geq f[y_0, \ldots, y_n, x]$.

3 Taylor models vs. using better approximations

3.1 Basic principles of Taylor models

Computing a TM consists in computing a polynomial together with an interval bound by applying simple rules recursively on the structure of the function f. In fact, for functions like trigonometric, exponential, logarithmic functions, as well as operations like $1/x$ or the power function, all referred to in this article as basic functions (or as intrinsics in [17]), bounds for the remainders can be easily computed. For composite functions, TMs offer usually a

much tighter bound than the one directly computed for the whole function, for example using automatic differentiation [1, 25, 20]. A meaningful comparison for this phenomenon is provided in [11, 18]. Here we provide the reader with a quick overview of the situation. Consider the Taylor development of a composite function $v \circ u$, where u is a basic function defined on I and v is a basic function defined on J respectively. The Taylor remainder is given by the Lagrange formula: for all $x \in I$, there exists $\xi \in I$ such that

$$R(x) = \frac{(v \circ u)^{(n+1)}(\xi)}{(n+1)!} (x - x_0)^{n+1}.$$

When bounding the remainder by means of automatic differentiation, an interval $\boldsymbol{K}$ enclosing the values of the $n+1$-th derivative of this composite function $(v \circ u)^{(n+1)}(I)$ is obtained by performing many recursive operations involving enclosures of $u^{(i)}(I)$ and $v^{(i)}(J)$ which finally may produce a considerable overestimation in the remainder [11].

In contrast, TMs consider $v \circ u$ as a composition between two basic functions. Evaluating with interval arithmetic the n-th derivative of such simple functions can be done in a fast way using simple formulae and does not lead to serious overestimation.

Morevoer, most of the functions we deal with have Taylor series whose coefficients decrease. In particular, when the functions are analytic over a sufficiently large domain (a disk of radius > 1 suffices), the magnitude of the coefficients of the underlying polynomial decreases exponentially (this is a consequence of Cauchy's integral formula). Hence when performing the composition of two such models, the intervals contributing to the final remainder become smaller for monomials of higher degree. This leads to a reduced overestimation in the computed remainder.

In conclusion, we highlight that, in practice, it is significantly more suitable to use a two-step procedure for handling composite functions: first step consists in computing models $(P, \boldsymbol{\Delta})$ for all basic functions; second, apply algebraic rules specifically designed for handling operations with these mixed models instead of operations with the corresponding functions.

3.2 Substituting tighter polynomial approximations to Taylor polynomials

It is well known that Taylor polynomials are fairly poor approximations to functions, except perhaps on very small intervals. It is thus very tempting to try to work with other classes of polynomials.

3.2.1 A previous attempt

The idea of using better approximation polynomials, for example Chebyshev truncated series, in the context of validated computations was introduced in [13] under the name ultra-arithmetic. As explained in [18], in their setting, the advantages of non-Taylor approximations cannot be explicitly maintained due to several drawbacks. It is noted in [18] that the Taylor representation is a special case, because for two functions f_1 and f_2 , the Taylor representation of order n, for the product $f_1 \cdot f_2$ can be obtained merely from the Taylor expansions of f_1 and f_2, simply by multiplying the polynomials and discarding the orders $n+1$ to $2n$. On the contrary, the Chebyshev truncated series of a product $f_1 \cdot f_2$ can in general not be obtained from the Chebyshev representations of the factors f_1 and f_2, and no operation analogous to TMs multiplication is given. Moreover, there is no systematic treatment of common basic functions. Finally, [18] explains that the methods developed in ultra-arithmetic can lead to

an increase in the magnitude of the coefficients, which will increase both the computational errors and the difficulty of finding good interval enclosures of the polynomials involved.

3.2.2 Minimax approximation

One may first try to directly use the minimax polynomial which is the polynomial which minimizes the supremum norm of the approximation error. In fact, such a polynomial can be obtained only through an iterative procedure, namely Remez algorithm, which can be considered too computationally expensive in our context. But the main drawback is linked to the computation of the remainder: obtaining a certified remainder in such a procedure, raises a significant dependency problem as discussed in [12, 11].

3.2.3 Chebyshev interpolation polynomial

The next natural idea is to consider an interpolation polynomial instead of a Taylor approximation: if the polynomial interpolates the function at Chebyshev nodes, such a polynomial, which we call Chebyshev interpolation polynomial, can usually provide a near-optimal approximation of f. See e.g. [24] which states an effective measure of this property.

Trefethen [31] uses the idea of representing the functions by Chebyshev interpolation polynomials. He chooses their expansion length in such a way as to maintain the accuracy of the approximation obtained close to machine precision. Moreover, the idea of using basic functions and then consider an algebra on them is used. The developed software, Chebfun was successfully used for numerically solving differential equations, quadrature problems. However, the Chebfun system does not provide any validation of the results obtained. As mentioned in [31], the aim of this system are numerical computations and no formal proof or safeguards are yet implemented to guarantee a validated computation, although it seems to be a future project.

3.2.4 Towards Chebyshev interpolation models

The interpolation polynomial itself is easy to compute and a wide variety of methods and various basis representations exist (we refer the reader to any numerical analysis book, see [30] for instance, for its computation). Another advantage of interpolation polynomials is that a closed formula, hence an explicit bound, for the remainder exists, cf. (2) and (3). Formula (3) induces that for an approximation polynomial of degree n, roughly speaking, compared to a Taylor remainder, the interpolation error will be scaled down by a factor of 2^{1-n}. For bounding the remainder, the only remaining difficulty is to bound the term $f^{(n+1)}(\xi)$ for $\xi \in I$. However, as briefly discussed above and in [11] the advantage of using directly this formula for the remainder can not be effectively maintained if the interval bound is obtained using automatic differentiation because of the overestimation. Hence in what follows we will try to adapt the "basic bricks" approach used by TMs to one using interpolation polynomials.

As explained above, a straightforward application of the principles and operations of TMs is not feasible when other better approximation polynomials are used. This paper seems to be a first attempt to combine the techniques and quasi-optimal remainder provided by interpolation for "basic bricks" with arithmetic rules similar to those used in TMs.

4 Chebyshev Interpolation Models

In the following we consider a Chebyshev Interpolation model (CM) of degree n for a function f over an interval I, as a couple $(P, \mathbf{\Delta})$, in the following sense: P is a polynomial of degree n which is "closely related" to the Chebyshev interpolation polynomial for f, $\mathbf{\Delta}$ is an interval enclosure for the remainder $f - P$. Specifically, from the mathematical point of view, for basic functions f, this polynomial P coincides with the Chebyshev interpolation polynomial and the computation of $\mathbf{\Delta}$ is detailed in 5.1.2. For composite functions, we will use some algebraic rules defined in Section 5 in a similar manner to the TMs arithmetic. We insist that our setting is multiple-precision interval arithmetic and all algorithms chosen are adapted for and implemented specifically for it. Since a thorough description of their implementation is tedious we indicate some implementation hints in 4.2. In what follows, we explain why we chose to use Newton and Chebyshev bases for performing operations over CMs.

4.1 Choice of representation basis

When computing an interpolation polynomial P two key choices are: the interpolating nodes, and the basis used for implementation. In this paper, we consider Chebyshev nodes. However, in what concerns the basis used for implementation, Lagrange basis suffers from numerical instability when poorly implemented [16]. It has been shown in [16, 3] that barycentric Lagrange basis has very good numerical properties and should be the "polynomial interpolation method of choice".

In our case, however, a third essential requirement emerges: one has to be able to find suitable algebraic rules for operations with models $(P, \mathbf{\Delta})$ that will not lead to overestimation of the remainder in the resulted model. This implies that addition, multiplication or composition of models made out of polynomials P, together with an interval remainder bound, has to be an effective process not only in terms of performance and quality of the polynomial obtained, but also in quality of remainder. We note, that in what concerns addition, the representation of P in any basis will suffice, since this is a linear operator.

For multiplication, as discussed in 3.2.1, it has not been obvious so far how to devise an efficient algorithm that, given two functions f_1 and f_2 and their models $(P_1, \mathbf{\Delta}_1)$ and $(P_2, \mathbf{\Delta}_2)$, where the polynomial part is obtained through an interpolation process, is able to efficiently compute a model $(P, \mathbf{\Delta})$ for $f_1 \cdot f_2$. Furthermore, composition rules $f_1 \circ f_2$ were even more difficult to obtain. On the contrary, these operations are straightforward for TMs, which is one of their incontestable advantages.

Consequently, we were led to searching for suitable basis representations for P such that all these requirements are successful. We eliminated a Lagrange basis representation for P not only because of its poor numerical properties [16], but also because of the $n-1$ terms containing each $n-1$ products of $x - x_i^*$. This leads to high overestimations on the interval remainder when a composition operation was tried.

In what concerns using barycentric Lagrange basis versus Newton basis, it is proved in [16] that the first has better numerical properties. However, Newton basis also has good numerical properties for certain orderings of points [16]. In our case, barycentric Lagrange has the disadvantage that the polynomial P is represented as $\left(\sum_{i=0}^{n} \frac{w_i}{x - x_i^*} f(x_i^*)\right) / \left(\sum_{i=0}^{n} \frac{w_i}{x - x_i^*}\right)$. It seems cumbersome to implement multiplication and composition of two models using this representation for P. We note that when composition should be implemented, the reciprocal

$1/(x - x_i^*)$ should be designed. This leads to considerable overestimations of the remainder.

We note that Newton basis can be seen as an "immediate extension" to Taylor basis, so it is a natural try to make and leads to an adaptation of TMs algorithms. In what follows we consider the Newton basis at Chebyshev nodes taken in decreasing order, as given by formula (1). This order has good numerical properties, see for example Section 5.3 [15]. In Section 5 we show how multiplications and composition operations can be implemented.

As for Chebyshev basis, we were led to this choice mainly because of the remarkable properties and success of numerical methods based on Chebyshev series expansions [9, 3].

Moreover, as we will see in Section 5.5, the rapid decreasing of the coefficients in the representations in the last two bases allow for small overestimation when handling CMs.

We now give some basic operations that will be used during the operations on the models.

Operations with polynomials in Newton basis Consider two polynomials of degree n in Newton basis: $P(x) = \sum_{i=0}^{n} p_i N_i(x)$ and $Q(x) = \sum_{i=0}^{n} q_i N_i(x)$.

Addition. Adding two polynomials in Newton basis is straightforward: $P(x) + Q(x) = \sum_{i=0}^{n} (p_i + q_i) N_i(x)$

Multiplication. When multiplying two polynomials in Newton basis, we need that "the lower part" of the result be represented in this basis also, and the "upper part", should be represented such that it can be easily bounded with interval arithmetic. Hence, we chose to represent the product PQ in the basis $N_0, \ldots, N_n, W_{\overline{x^*}} N_0, \ldots, W_{\overline{x^*}} N_{n-1}$. We have $P(x) \cdot Q(x) = G(x) + W_{\overline{x^*}}(x) H(x)$, where $G(x) = \sum_{i=0}^{n} g_i N_i(x)$ and $H(x) = \sum_{i=0}^{n-1} h_i N_i(x)$.

One of the advantages of this representation is that we gave an exact interval bound for $W_{\overline{x^*}}$ in Section 2.1. Moreover, it can be easily shown that $G(x)$ is the Chebyshev interpolation polynomial of $P(x) \cdot Q(x)$, i.e. of fg.

In order to compute the coefficients of G and H one can use several techniques. We mention that based on [8] a conversion back to monomial basis, followed by an interpolation seems to be the fastest asymptotically (needing $O(M(n) \log n)$ operations,where $M(n)$ denotes the cost of multiplying univariate polynomials of degree less than n). However, in our current multiple precision interval arithmetic implementation, we use an $O(n^2)$ algorithm based on applying the divided differences algorithm first for computing the coefficients of G and then for computing the coefficients of $H = (PQ - G)/W_{\overline{x^*}}$.

Interval Range Bounding. We use a Horner-like algorithm [15] for bounding the range of a polynomial in Newton basis, which takes $O(n)$ operations.

Operations with polynomials in Chebyshev basis Consider two polynomials of degree n in Chebyshev basis: $P(x) = \sum_{i=0}^{n} p_i T_i^{[a,b]}(x)$ and $Q(x) = \sum_{i=0}^{n} q_i T_i^{[a,b]}(x)$.

Addition. Adding two polynomials in Chebyshev basis is straightforward: $P(x) + Q(x) = \sum_{i=0}^{n} (p_i + q_i) T_i^{[a,b]}(x)$ and takes $O(n)$ operations.

Multiplication. Their product can be expressed in Chebyshev basis as follows: $P(x) \cdot Q(x) = \sum_{k=0}^{2n} c_k T_k^{[a,b]}(x)$, where $c_k = (\sum_{|i-j|=k} p_i \cdot q_j + \sum_{i+j=k} p_i \cdot q_j)/2$. This identity can be

obtained noting that $T_i^{[a,b]}(x) \cdot T_j^{[a,b]}(x) = (T_{i+j}^{[a,b]} + T_{|i-j|}^{[a,b]})/2$ [19]. The cost using this simple identity is $O(n^2)$ operations.

Interval Range Bounding. For our purpose, we will use the following identity for interval bounding range of a polynomial in Chebyshev basis, which takes $O(n)$ operations: $\forall x \in [a,b], P(x) \in p_0 + \sum_{i=1}^{n} p_i \cdot [-1,1]$.

4.2 Rigorous implementation of the interpolation process

In the following, we denote an interval $\boldsymbol{x}$ as a pair $\boldsymbol{x} = [\underline{x}, \overline{x}]$. For a polynomial P with real number coefficients, we denote by $\boldsymbol{P}$ a polynomial obtained by replacing its coefficients with intervals which enclose them. We denote by $B(\boldsymbol{P}(\boldsymbol{x}))$ an interval enclosure for $\boldsymbol{P}(x)$, when $x \in \boldsymbol{x}$.

We implemented all the operations involved using multiple-precision interval arithmetic and all the algorithms we use are straightforwardly adaptable in this context. Hence, from the implementation point of view, the only notable change is that instead of polynomials with real number coefficients we have polynomials with tight interval coefficients. This means that in our implementation, for a function f we obtain a model $(\boldsymbol{P}, \boldsymbol{\Delta})$, such that $\forall x \in [a,b],\ f(x) \in \boldsymbol{P}(x) + \boldsymbol{\Delta}$. This design choice allows us to take into account all the rounding errors, because for each operation with interval arithmetic an outward rounding is performed. Moreover, it is proved in [27] that when evaluating a function φ over a point interval $\boldsymbol{x} = [x,x]$, the interval enclosure of $\varphi(x)$ can be made arbitrarily tight by increasing the precision used for evaluation. In our case the computations needed for the coefficients of the polynomial are done with initially almost point intervals, so the overestimation of the coefficients can be made as small as possible by increasing the precision used.

However, if needed, a certified floating-point approximation polynomial can be easily obtained from a CM $(\boldsymbol{P}, \boldsymbol{\Delta})$. Let us consider $\boldsymbol{P}(x) = \sum_{i=0}^{n} \boldsymbol{c_i}\beta_i(x)$ where β_i is the ith element of the considered basis (Newton or Chebyshev). It suffices to take the middle points t_i of $\boldsymbol{c_i}$ as the coefficients of the approximation polynomial $\tilde{P}(x) = \sum_{i=0}^{n} t_i\beta_i(x)$ and then to compute a simple interval bound $\boldsymbol{\delta} = \sum_{i=0}^{n} [\underline{\boldsymbol{c_i}} - t_i, \overline{\boldsymbol{c_i}} - t_i] \cdot \boldsymbol{\beta_i(x)}$, using the methods presented in Section 4.1. Then the error between the function f and its approximation polynomial $\tilde{P}$ is bounded by $\boldsymbol{\delta} + \boldsymbol{\Delta}$.

5 Chebyshev Interpolation Models with exact arithmetic

We follow the two-step approach specific for TMs, as mentioned in Section 3. As detailed below, firstly, we compute models for basic functions; secondly we apply specifically designed algebraic operations on such models.

5.1 Basic functions

We consider as basic functions the trigonometric, exponential, logarithmic functions, $1/x$, the power function, or any other function for which specific properties can be exploited. For such

functions we compute directly a model $(P, \mathbf{\Delta})$ formed by an interpolation polynomial P and an interval bound for the remainder $\mathbf{\Delta}$.

5.1.1 Computation of the interpolation polynomial

We mentioned in 2.2 how to express an interpolation polynomial in Newton basis, using the divided differences procedure. The number of operations necessary to compute the coefficients of the interpolation polynomial is $3n^2/2$ [15].

We now need to know how to represent it using Chebyshev basis. The Chebyshev interpolation polynomial P can be expressed using the collocation method [19] as follows: $P(x) = \sum_{i=0}^{n} p_i T_i^{[a,b]}(x)$, with $p_i = \sum_{k=0}^{n} \frac{2}{n+1} f(x_k^*) T_i^{[a,b]}(x_k^*)$.

Using directly this formula, the computation cost is $O(n^2)$ operations. It is known [23] that the usage of Fast Fourier Transform, can speed-up this computation to $O(n \log n)$ operations. However, note that in this case, an interval arithmetic adaptation of FFT should be considered.

5.1.2 Computation of the remainder

We can compute an enclosure of the remainder $f - P$ over $[a, b]$ using formula (3): $\Delta_n(x, \xi) = f^{(n+1)}(\xi) W_{\overline{x^*}}(x)/(n+1)!$, where $x, \xi \in [a, b]$. This reduces to bounding $f^{(n+1)}$ over $[a, b]$, which does not pose any problem for basic functions, since simple formulae are available for their derivatives. Moreover, $W_{\overline{x^*}}(x)$ can be bounded straightforwardly using Lemma 2.1.

We remark that, thanks to Lemma 2.2, an exact bound for the remainder can be computed when one can show that $f^{(n+1)}$ is increasing (resp. decreasing) over $[a, b]$. In such cases, one can use (2) and bound $f[x_0^*, \ldots, x_n^*, x]$ by $[\ f[x_0^*, \ldots, x_n^*, a],\ \ f[x_0^*, \ldots, x_n^*, b]\ ]$ (resp. $[\ f[x_0^*, \ldots, x_n^*, b],\ \ f[x_0^*, \ldots, x_n^*, a]\ ]$). For basic functions, checking whether $f^{(n+2)}$ has constant sign over $[a, b]$ is simple. This remark makes it possible to obtain smaller remainders and strengthens the effectiveness of the "basic bricks" approach.

5.1.3 Bounding the interpolation polynomial

When using this approach, one needs to compute bounds for the range of polynomials involved in such models. Several methods exist and a trade-off between their speed and the tightness of the bound is usually considered. For TMs, the fastest but "rough" method is a Horner-like interval evaluation. More complicated schemes exist, that usually give tighter bounds: LDB, QDB [5], or using a conversion to Bernstein basis [21, 32]. For univariate polynomials, slower techniques based on roots isolation can be used for a very tight polynomial bounding [29]. In our case, we focused on speed, and so, when considering interval range bounding for polynomials in Newton or Chebyshev basis we used two simple methods described in Section 4.1. Similarly to TMs, for a penalty in speed, more refined algorithms can also be plugged-in. However the simple techniques we used, proved effective in most of the examples we treated so far.

In what follows we consider two CMs for two functions f_1 and f_2, over the interval $[a, b]$, of degree n: $(\boldsymbol{P_1}, \mathbf{\Delta}_1)$ and $(\boldsymbol{P_2}, \mathbf{\Delta}_2)$. Similarly to TM arithmetic, we define the following operations.

5.2 Addition

Addition of two such models is done straightforwardly by adding the two polynomials and the remainder bounds as:

$$(\boldsymbol{P_1}, \boldsymbol{\Delta}_1) + (\boldsymbol{P_2}, \boldsymbol{\Delta}_2) = (\boldsymbol{P_1} + \boldsymbol{P_2}, \boldsymbol{\Delta}_1 + \boldsymbol{\Delta}_2).$$

It is obvious that since $f_1(x) \in \boldsymbol{P_1} + \boldsymbol{\Delta}_1$ and $f_2(x) \in \boldsymbol{P_2} + \boldsymbol{\Delta}_2$, $f_1(x) + f_2(x) \in \boldsymbol{P_1} + \boldsymbol{P_2} + \boldsymbol{\Delta}_1 + \boldsymbol{\Delta}_2$.

Note that adding two polynomials in Newton or Chebyshev basis is straightforward and has a linear complexity, see Section 4.1. For the sake of completeness, we mention that multiplying a CM with a constant, reduces to multiplying the polynomial and the remainder with the respective constant.

5.3 Multiplication

Consider $f_1(x) \in \boldsymbol{P_1} + \boldsymbol{\Delta}_1$ and $f_2(x) \in \boldsymbol{P_2} + \boldsymbol{\Delta}_2$. We have

$$f_1(x) \cdot f_2(x) \in \boldsymbol{P_1} \cdot \boldsymbol{P_2} + \boldsymbol{P_2} \cdot \boldsymbol{\Delta}_1 + \boldsymbol{P_1} \cdot \boldsymbol{\Delta}_2 + \boldsymbol{\Delta}_1 \cdot \boldsymbol{\Delta}_2.$$

We observe that $\boldsymbol{P_1} \cdot \boldsymbol{P_2}$ is a polynomial of degree $2n$. Depending on the basis used, we split it into two parts: the polynomial consisting of the terms that "do not exceed n", $(\boldsymbol{P_1} \cdot \boldsymbol{P_2})_{0\ldots n}$ and respectively the upper part $(\boldsymbol{P_1} \cdot \boldsymbol{P_2})_{n+1\ldots 2n}$, for the terms of the product $\boldsymbol{P_1} \cdot \boldsymbol{P_2}$ whose "order exceeds n".

Now, a CM for $f_1 \cdot f_2$ can be obtained by finding an interval bound for all the terms except $\boldsymbol{P} = (\boldsymbol{P_1} \cdot \boldsymbol{P_2})_{0\ldots n}$. Hence we have,

$$\boldsymbol{\Delta} = B((\boldsymbol{P_1} \cdot \boldsymbol{P_2})_{n+1\ldots 2n}) + B(\boldsymbol{P_2}) \cdot \boldsymbol{\Delta}_1 + B(\boldsymbol{P_1}) \cdot \boldsymbol{\Delta}_2 + \boldsymbol{\Delta}_1 \cdot \boldsymbol{\Delta}_2.$$

The interval bound for the polynomials involved can be computed as discussed in 5.1.3.

In our current setting, cf. Section 4.1, the number of operations necessary to multiply two such models is $O(n^2)$.

5.4 Composition

When the model for $f_1 \circ f_2$ is needed, we can consider $(f_1 \circ f_2)(x)$ as function f_1 evaluated at point $y = f_2(x)$. Hence, we have to take into account the additional constraint that the image of f_2 has to be included in the definition range of f_1. This can be checked by a simple interval bound computation of $B(\boldsymbol{P_2}) + \boldsymbol{\Delta_2}$. Then we have:

$$(f_1 \circ f_2)(x) \in \boldsymbol{P_1}(f_2(x)) + \boldsymbol{\Delta_1} \in \boldsymbol{P_1}(\boldsymbol{P_2}(x) + \boldsymbol{\Delta_2}) + \boldsymbol{\Delta_1} \tag{4}$$

In this formula, the only polynomial coefficients and remainders involved are those of the CMs of f_1 and f_2 which are basic functions. As we have seen above, fairly simple formulæ exist for computing the coefficients and remainders of such functions. However, when using formula (4), it is not obvious how to extract a polynomial and a final remainder bound from it. In fact, we have to reduce this extraction process to performing just multiplications and additions of CMs. A similar idea is used for composing TMs [18, 32].

In our case, the difference is that $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are polynomials represented in Newton/Chebyshev basis, and not in the monomial basis. In consequence, for computing the composition, we had

to use a different algorithm. It is worth mentioning that a simple change of basis back and forth to monomial basis will not be successful. The problem is that the multiplications and additions used in such a composition process do not have to add to much overestimation to the final remainder. As we discussed in Section 3, for Taylor expansions of most of the functions we address, the size of the coefficients for the representation in the monomial basis is bounded by a decreasing sequence. Hence the contributions of the remainders in such a recursive algorithm are smaller and smaller. On the contrary, for interpolation polynomials, the coefficients represented in monomial basis oscillate too much and have poor numerical properties. Hence, a direct application of the principle of composing TMs will not be successful.

When using Newton basis, we perform the composition using a Horner-like algorithm [15]. It takes a linear number of operations between models.

Algorithm 1 *Horner-like composition of CMs in Newton basis: Composing* $(\boldsymbol{P_1}, \boldsymbol{\Delta_1})$ *with* $(\boldsymbol{P_2}, \boldsymbol{\Delta_2})$

*/*We denote by* $\boldsymbol{P_{1_j}}$ *the* j*th coefficient of* $\boldsymbol{P_1}$ **/*
*/*and by* x_j^* *the* j*th interpolation point for* $\boldsymbol{P_1}$ **/*
$(\boldsymbol{C_n}, \boldsymbol{R_n}) := (\boldsymbol{P_{1}}_n, [0,0])$
For $j = n-1, \ldots, 0$ *do*
$(\boldsymbol{C_j}, \boldsymbol{R_j}) := ((\boldsymbol{P_2}, \boldsymbol{\Delta_2}) - (x_j^*, [0,0])) \cdot (\boldsymbol{C_{j+1}}, \boldsymbol{R_{j+1}}) + (\boldsymbol{P_{1}}_j, [0,0])$;
Return $(\boldsymbol{C_0}, \boldsymbol{R_0} + \boldsymbol{\Delta_1})$

When using Chebyshev basis, we perform the composition using an adaptation of Clenshaw algorithm [19]. Algorithm 2 is used for efficient evaluation of a Chebyshev sum $\sum c_i T_i(x)$. It reduces evaluation of such polynomials to basic additions and multiplications and it is as efficient as Horner form for evaluating a polynomial as a sum of powers using nested multiplications. In our case, the variable x where the sum is to be evaluated is a CM, the multiplications and additions are operations between CMs. Moreover, using this algorithm, we perform a linear number of such operations between models.

Algorithm 2 *Clenshaw-like composition of CMs in Chebyshev basis: Composing* $(\boldsymbol{P_1}, \boldsymbol{\Delta_1})$ *with* $(\boldsymbol{P_2}, \boldsymbol{\Delta_2})$

$(\boldsymbol{C_{n+2}}, \boldsymbol{R_{n+2}}) := (0, [0,0])$ */*CMs for* 0 **/*
$(\boldsymbol{C_{n+1}}, \boldsymbol{R_{n+1}}) := (0, [0,0])$
*/*We denote by* $\boldsymbol{P_{1_j}}$ *the* j*th coefficient of* $\boldsymbol{P_1}$ **/*
For $j = n, \ldots, 1$ *do*
$(\boldsymbol{C_j}, \boldsymbol{R_j}) := 2 \cdot (\boldsymbol{P_2}, \boldsymbol{\Delta_2}) \cdot (\boldsymbol{C_{j+1}}, \boldsymbol{R_{j+1}}) - (\boldsymbol{C_{j+2}}, \boldsymbol{R_{j+2}}) + (\boldsymbol{P_{1}}_j, [0,0])$;
$(\boldsymbol{C}, \boldsymbol{R}) := (\boldsymbol{P_2}, \boldsymbol{\Delta_2}) \cdot (\boldsymbol{C_1}, \boldsymbol{R_1}) - (\boldsymbol{C_2}, \boldsymbol{R_2}) + (\boldsymbol{P_{1}}_0, [0,0])$
Return $(\boldsymbol{C}, \boldsymbol{R} + \boldsymbol{\Delta_1})$

5.5 Growth of the coefficients and overestimation

The overestimation does not grow too much during the recursion process. This is due to the nice convergence properties of the series expansions in Newton Basis or in Chebyshev polynomial basis. One can prove for instance that if the function under consideration is analytic over a domain of $\mathbb{C}$ which contains $[a, b]$, the coefficients of the expansion in Newton [14] or in Chebyshev [9] bases decrease exponentially. As with TMs, when composing two such models,

the intervals contributing to the final remainder become smaller for higher coefficients, which yields a reduced overestimation in the final remainder.

6 Experimental results

We implemented the two methods in Maple, using the IntpakX[1] package. Table 1 shows the quality of some absolute error bounds obtained with the methods that we presented. Each row of the table represents one example. The function f, the interval I and the degree n of the approximation polynomial are given in the first column. In the second and third columns, we give the upper-bound for the remainder obtained using a CM as explained in Section 5 when using polynomials in Newton (CM1) and Chebyshev (CM2) bases. We computed the exact errors between f and the polynomials from CM1 and CM2 and we provide in the third column the minimum of these, such that the user can observe the overestimation of the remainder. We also give the remainder bounds and the exact error obtained when an interpolating polynomial is directly used (directly means that the remainder is computed using (3) and automatic differentiation). Finally we present the remainder obtained using a TM and the exact error. The Taylor polynomial was developed in the middle of I and the necessary polynomials bounds were computed using a Horner scheme.

The examples presented are representative for several situations, and we will detail them in what follows. The first five were analyzed in [11] for comparing the bounds obtained with interpolation vs. TMs. There, it was observed that in some cases the overestimation in the interpolation remainder is so big, that we can not benefit from using such a polynomial. We used them in order to highlight that CMs do not have this drawback and the remainders obtained with our methods have better quality than the TMs in all situations.

The first example presents a basic function which is analytic on the whole complex plane. There is almost no overestimation in this case, whatever method we use. The second is also a basic function. It has singularities in the complex plane (in $\pi/2 + \mathbb{Z}\pi$), but the interval I is relatively far from these singularities. All the methods present a relatively small overestimation. The third example is the same function but over a larger interval. In these case, the singularities are closer and Taylor polynomials are not very good approximations. The fourth and fifth examples are composite functions on larger intervals. The overestimation in the interpolation method becomes very large, rendering this method useless, while it stays reasonable with TMs and CMs.

The following examples $(6-8)$ are similar to some presented in [3]. There, the authors computed the minimax polynomials for these functions. Evidently, the polynomials obtained with CMs have a higher approximation error than the minimax, however, it is important to notice that in these tricky cases the remainder bound obtained for the CMs stays fairly good and it is much better than the one obtained from TMs.

Examples $8-9$ present the case when the definition domain of the function is close to a singularity. As seen in these examples, when a direct interpolation process is used for a composite function, unfortunately, one can not apply Lemma 2.2 for bounding the remainder. Consequently, the bound obtained for the remainder is highly overestimated. However, when using the approach based on "basic bricks" both TMs and CMs benefit from it, yielding a much better remainder bound.

[1] http://www.math.uni-wuppertal.de/~xsc/software/intpakX/

Example 10 deals with a function which is associated to the classical Runge phenomenon. Firstly, since the complex singularities of the function f defined by $f(x) = 1/(1+4x^2)$ are close to the definition interval I, the Taylor polynomial is not a good approximation. Then, the interpolation method gives unfeasible interval remainder bounds due to the overestimation of the $n+1$th derivative of the function f. TMs and CM1 fail both from the same cause: when computing a model for $f = g \circ h$ one needs to compose the models for $g(y) = 1/y$ and $h(x) = 1+4x^2$. The model for h is simple to compute. However, one has to take into account that the model of g has to be computed over an interval range enclosure of the model of h. When such an enclosure is computed, using the presented simple methods for polynomial range bounding we have an overestimation of the image of h. This leads in fact to an interval that contains 0. Hence, the model for g which is not defined in 0 can not be computed. On the contrary, the overestimation in polynomial range bounding with the method of CM2 is smaller and we have a feasible remainder in this case.

Example 11 is presented to emphasize the fact that similarly to TMs, the CMs reduce the dependency problem and can convey informations about the properties of functions. The CM for $f = \sin^2 + \cos^2$ over $[-1, 1]$, without performing any expression simplification, is in fact $(0.99999999997 + \tilde{P}(x), [-3.91 \cdot 10^{-9}, 3.91 \cdot 10^{-9}])$, where $\tilde{P}(x)$ is a polynomial with a tiny supremum norm. Hence, the information that f is in fact 1, is conveyed when using such models, while when using interval arithmetic directly, one would obtain an interval range of $[0.29, 1.71]$.

We do not give specific timings, since for the moment our implementation is rather a "proof of concept" one. The algorithms necessary for TMs seem to be slightly simpler and hence more efficient than the ones needed by our approaches: we observed a factor between 2 and 3 of speed-up in favor of TMs, in some examples. However, we note that in some cases, in order to attain the same quality for the remainder, TMs need a much higher polynomial degree, or they have to be applied over many subintervals, which favors the usage of CMs.

No	$f(x)$, I, n	CM1	CM2	Exact bound	Interpolation	Exact bound	TM	Exact bound
1	$\sin(x)$, [3, 4], 10	$\mathbf{1.19 \cdot 10^{-14}}$	$\mathbf{1.19 \cdot 10^{-14}}$	$1.13 \cdot 10^{-14}$	$\mathbf{1.19 \cdot 10^{-14}}$	$1.13 \cdot 10^{-14}$	$1.22 \cdot 10^{-11}$	$1.16 \cdot 10^{-11}$
2	$\arctan(x)$, [−0.25, 0.25], 15	$\mathbf{7.89 \cdot 10^{-15}}$	$\mathbf{7.89 \cdot 10^{-15}}$	$7.95 \cdot 10^{-17}$	$\mathbf{7.89 \cdot 10^{-15}}$	$7.95 \cdot 10^{-17}$	$2.58 \cdot 10^{-10}$	$3.24 \cdot 10^{-12}$
3	$\arctan(x)$, [−0.9, 0.9], 15	$\mathbf{5.10 \cdot 10^{-3}}$	$\mathbf{5.10 \cdot 10^{-3}}$	$1.76 \cdot 10^{-8}$	$\mathbf{5.10 \cdot 10^{-3}}$	$1.76 \cdot 10^{-8}$	$1.67 \cdot 10^{2}$	$5.70 \cdot 10^{-3}$
4	$\exp(1/\cos(x))$, [0, 1], 14	$6.69 \cdot 10^{-7}$	$\mathbf{5.22 \cdot 10^{-7}}$	$4.95 \cdot 10^{-7}$	0.11	$6.10 \cdot 10^{-7}$	$9.06 \cdot 10^{-3}$	$2.59 \cdot 10^{-3}$
5	$\frac{\exp(x)}{\log(2+x)\cos(x)}$, [0, 1], 15	$1.11 \cdot 10^{-8}$	$\mathbf{9.11 \cdot 10^{-9}}$	$2.21 \cdot 10^{-9}$	0.18	$2.68 \cdot 10^{-9}$	$1.18 \cdot 10^{-3}$	$3.38 \cdot 10^{-5}$
6	$\sin(\exp(x))$,[−1, 1], 10	$\mathbf{4.10 \cdot 10^{-6}}$	$9.47 \cdot 10^{-5}$	$3.72 \cdot 10^{-6}$	$4.42 \cdot 10^{-3}$	$3.72 \cdot 10^{-6}$	$2.96 \cdot 10^{-2}$	$1.55 \cdot 10^{-3}$
7	$\tanh(x+0.5) - \tanh(x-0.5)$, [−1, 1], 10	$8.48 \cdot 10^{-3}$	$\mathbf{1.75 \cdot 10^{-3}}$	$4.88 \cdot 10^{-7}$	$8.48 \cdot 10^{-3}$	$4.88 \cdot 10^{-7}$	8.68	$2.96 \cdot 10^{-3}$
8	$\sqrt{x+1.0001}$, [−1, 0], 10	$\mathbf{3.64 \cdot 10^{-2}}$	$\mathbf{3.64 \cdot 10^{-2}}$	$3.64 \cdot 10^{-2}$	$3.64 \cdot 10^{-2}$	$3.64 \cdot 10^{-2}$	0.11	0.11
9	$\sqrt{x+1.0001} \cdot \sin(x)$, [−1, 0], 10	$\mathbf{3.10 \cdot 10^{-2}}$	$3.32 \cdot 10^{-2}$	$3.08 \cdot 10^{-2}$	$3.21 \cdot 10^{33}$	$3.08 \cdot 10^{-2}$	0.12	$9.83 \cdot 10^{-2}$
10	$\frac{1}{1+4x^2}$, [−1, 1], 10	$+\infty$	$\mathbf{1.13 \cdot 10^{-2}}$	$6.17 \cdot 10^{-3}$	$1.50 \cdot 10^{7}$	$4.95 \cdot 10^{-3}$	$+\infty$	$8.20 \cdot 10^{2}$
11	$\sin^2(x) + \cos^2(x)$, [−1, 1], 10	$8.98 \cdot 10^{-9}$	$\mathbf{3.91 \cdot 10^{-9}}$	$2.1 \cdot 10^{-11}$	$8.44 \cdot 10^{-8}$	$2.29 \cdot 10^{-50}$	$8.74 \cdot 10^{-6}$	$5.57 \cdot 10^{-52}$

Table 1: Examples of bounds obtained by several methods

7 Conclusion and future work

We introduced two approaches for computing "Chebyshev interpolation models", a tool which is potentially useful in various rigorous computing applications. Currently, they achieve smaller remainders than Taylor models and require nevertheless more computing time.

This work is preliminary, in two senses. First, we did not address in this paper the opportunity to use Chebyshev truncated series instead of Chebyshev interpolation polynomials. Actually, this approach is a work in progress and seems very promising since the quality of

the remainder should remain at least as good as the one provided by CM and the complexity of basic bricks computations should be lowered. This issue of complexity has to be addressed if we want CMs to replace TMs in most of univariate applications. The techniques developed in [26, 2] could prove useful to achieve this goal. Secondly, we address the computation of such models for univariate functions only. A long-term project of ours is to extend these methods to the multivariate case.

The methods presented in this paper should be available in the months to come in the Sollya[1] tool.

References

[1] C. Bendsten and O. Stauning. TADIFF, a Flexible C++ Package for Automatic Differentiation Using Taylor Series. Technical Report 1997-x5-94, Technical Univ. of Denmark, April 1997.

[2] A. Benoit and B. Salvy. Chebyshev expansions for solutions of linear differential equations. In J. May, editor, *ISSAC '09: Proceedings of the twenty-second international symposium on Symbolic and algebraic computation*, pages 23–30, 2009.

[3] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Rev.*, 46(3):501–517, 2004.

[4] M. Berz and K. Makino. New methods for high-dimensional verified quadrature. *Reliable Computing*, 5(1):13–22, 1999.

[5] M. Berz and K. Makino. Rigorous global search using Taylor models. In *SNC '09: Proceedings of the 2009 conference on Symbolic numeric computation*, pages 11–20, New York, NY, USA, 2009. ACM.

[6] M. Berz, K. Makino, and Y.-K. Kim. Long-term stability of the tevatron by verified global optimization. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 558(1):1 – 10, 2006. Proceedings of the 8th International Computational Accelerator Physics Conference - ICAP 2004.

[7] P. Borwein and T. Erdélyi. *Polynomials and Polynomial Inequalities*. Graduate Texts in Mathematics, Vol. 161. Springer-Verlag, New York, NY, 1995.

[8] A. Bostan and É. Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, August 2005. Festschrift for the 70th Birthday of Arnold Schönhage.

[9] J. P. Boyd. *Chebyshev and Fourier spectral methods*. Dover Publications Inc., Mineola, NY, second edition, 2001.

[10] E. W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, 1966.

[11] S. Chevillard, J. Harrison, M. Joldes, and C. Lauter. Efficient and accurate computation of upper bounds of approximation errors. 40 pages, RRLIP2010-2, 2010.

[12] S. Chevillard, M. Joldes, and C. Lauter. Certified and fast computation of supremum norms of approximation errors. In *19th IEEE SYMPOSIUM on Computer Arithmetic*, pages 169–176, 2009.

[13] C. Epstein, W. Miranker, and T. Rivlin. Ultra-arithmetic i: Function data types. *Mathematics and Computers in Simulation*, 24(1):1 – 18, 1982.

[14] A. O. Gel′fond. *Calculus of finite differences*. Hindustan Pub. Corp., Delhi, 1971. Translated from the Russian, International Monographs on Advanced Mathematics and Physics.

[1] http://sollya.gforge.inria.fr/

[15] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2002.

[16] N. J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, 24(4):547–556, 2004.

[17] K. Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. PhD thesis, Michigan State University, East Lansing, Michigan, USA, 1998.

[18] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.

[19] J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. Chapman & Hall/CRC, Boca Raton, FL, 2003.

[20] R. E. Moore. *Methods and Applications of Interval Analysis*. Society for Industrial Mathematics, 1979.

[21] P. S. V. Nataraj and K. Kotecha. Global optimization with higher order inclusion function forms part 1: A combined taylor-bernstein form. *Reliable Computing*, 10(1):27–44, 2004.

[22] M. Neher, K. R. Jackson, and N. S. Nedialkov. On Taylor model based integration of ODEs. *SIAM J. Numer. Anal.*, 45:236–262, 2007.

[23] R. Platte and N. Trefethen. Chebfun: A new kind of numerical computing. Technical Report NA-08/13, Oxford University Computing Laboratory, Oct. 2008.

[24] M. J. D. Powell. On the maximum errors of polynomial approximations defined by interpolation and by least squares criteria. *Comput. J.*, 9:404–407, 1967.

[25] L. B. Rall. The arithmetic of differentiation. *Mathematics Magazine*, 59(5):275–282, 1986.

[26] L. Rebillard. *Étude théorique et algorithmique des séries de Chebyshev solutions d'équations différentielles holonomes*. PhD thesis, Institut National Polytechnique de Grenoble, 1998.

[27] N. Revol. Newton's algorithm using multiple precision interval arithmetic. *Numer. Algorithms*, 34(2):417–426, 2003.

[28] T. J. Rivlin. *Chebyshev polynomials. From approximation theory to algebra*. Pure and Applied Mathematics. John Wiley & Sons, New York, 2nd edition, 1990.

[29] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2004.

[30] M. Schatzman. *Numerical Analysis, A Mathematical Introduction*. Oxford University Press, 2002.

[31] L. N. Trefethen. Computing numerically with functions instead of numbers. *Mathematics in Computer Science*, 1(1):9–19, 2007.

[32] R. Zumkeller. *Global Optimization in Type Theory*. PhD thesis, École polytechnique, 2008.