Computing machine-efficient polynomial approximations

NICOLAS BRISEBARRE
Université J. Monnet, St-Étienne and LIP-E.N.S. Lyon
JEAN-MICHEL MULLER
CNRS, LIP-ENS Lyon
and
ARNAUD TISSERAND
INRIA, LIP-ENS Lyon

Polynomial approximations are almost always used when implementing functions on a computing system. In most cases, the polynomial that best approximates (for a given distance and in a given interval) a function has coefficients that are not exactly representable with a finite number of bits. And yet, the polynomial approximations that are actually implemented do have coefficients that are represented with a finite - and sometimes small - number of bits: this is due to the finiteness of the floating-point representations (for software implementations), and to the need to have small, hence fast and/or inexpensive, multipliers (for hardware implementations). We then have to consider polynomial approximations for which the degree-i coefficient has at most m_i fractional bits: in other words, it is a rational number with denominator 2^{m_i}. We provide a general and efficient method for finding the best polynomial approximation under this constraint. Moreover, our method also applies if some other constraints (such as requiring some coefficients to be equal to some predefined constants, or minimizing relative error instead of absolute error) are required.

Categories and Subject Descriptors: G.1.0 [**Numerical Analysis**]: General—*Computer arithmetic*; G.1.2 [**Numerical Analysis**]: Approximation; B.2.4 [**Arithmetic and Logic Structures**]: High-Speed Arithmetic

General Terms: Algorithms, Performance

Additional Key Words and Phrases: polynomial approximation, minimax approximation, floating-point arithmetic, Chebyshev polynomials, polytopes, linear programming

Authors' addresses: N. Brisebarre, LArAl, Université J. Monnet, 23, rue du Dr P. Michelon, F-42023 Saint-Étienne Cedex, France and LIP/Arénaire (CNRS-ENS Lyon-INRIA-UCBL), ENS Lyon, 46 Allée d'Italie, F-69364 Lyon Cedex 07 France; email: Nicolas.Brisebarre@ens-lyon.fr; J.-M. Muller, LIP/Arénaire (CNRS-ENS Lyon-INRIA-UCBL), ENS Lyon, 46 Allée d'Italie, F-69364 Lyon Cedex 07 France; email: Jean-Michel.Muller@ens-lyon.fr; Arnaud Tisserand, INRIA, LIP/Arénaire (CNRS-ENS Lyon-INRIA-UCBL), ENS Lyon, 46 Allée d'Italie, F-69364 Lyon Cedex 07 France; email: Arnaud.Tisserand@ens-lyon.fr.

Table I. Latencies (in number of cycles) of double precision floating-point addition, multiplication and division on some recent processors.

Processor	FP add	FP mult.	FP div.
Pentium IV	5	7	38
PowerPC 750	3	4	31
UltraSPARC III	4	4	24
Alpha21264	4	4	15
Athlon K6-III	3	3	20

1. INTRODUCTION

The basic floating-point operations that are implemented in hardware on modern processors are addition/subtraction, multiplication, and sometimes division and/or fused multiply-add, i.e., an expression of the form $xy + z$, computed with one final rounding only. Moreover, division is frequently much slower than addition or multiplication (see Table I), and sometimes, for instance on the Itanium, it is not hardwired at all. In such cases, it is implemented as a sequence of fused multiply and add operations, using an iterative algorithm.

Therefore, when trying to implement a given, regular enough, function f, it seems reasonable to try to avoid divisions, and to use additions, subtractions, multiplications and possibly fused multiply-adds. Since the only functions of one variable that one can implement using a finite number of these operations and comparisons are piecewise polynomials, a natural choice is to focus on piecewise polynomial approximations to f. Indeed, most recent software-oriented elementary function algorithms use polynomial approximations [Markstein 2000; Muller 1997; Story and Tang 1999; Cornea et al. 2002].

Two kinds of polynomial approximations are used: the approximations that minimize the "average error," called *least squares approximations*, and the approximations that minimize the worst-case error, called *least maximum approximations*, or *minimax approximations*. In both cases, we want to minimize a distance $||p - f||$, where p is a polynomial of a given degree. For least squares approximations, that distance is:

$$||p - f||_{2,[a,b]} = \left(\int_a^b w(x) \left(f(x) - p(x) \right)^2 dx \right)^{1/2},$$

where w is a continuous *weight function*, that can be used to select parts of $[a, b]$ where we want the approximation to be more accurate. For minimax approximations, the distance is:

$$||p - f||_{\infty,[a,b]} = \sup_{a \le x \le b} |p(x) - f(x)|.$$

One could also consider distances such as

$$||p - f||_{\mathrm{rel},[a,b]} = \sup_{a \le x \le b} \frac{1}{|f(x)|} |p(x) - f(x)|.$$

The least squares approximations are computed by a projection method using orthogonal polynomials. Minimax approximations are computed using an algorithm due to Remez [Remes 1934; Hart et al. 1968]. See [Markstein 2000; Muller 1997] for recent presentations of elementary function algorithms.

In this paper, we are concerned with minimax approximations using distance $||p-f||_{\infty,[a,b]}$. And yet, our general method of Section 3.2 also applies to distance $||p-f||_{\mathrm{rel},[a,b]}$. Our approximations will be used in finite-precision arithmetic. Hence, the computed polynomial coefficients are usually *rounded*: let $m_0, m_1, \ldots, m_n$ be a fixed finite sequence of natural integers, the coefficient p_i of the minimax approximation

$$p(x) = p_0 + p_1 x + \cdots + p_n x^n$$

is rounded to, say, the nearest multiple of 2^{-m_i}. By doing that, we obtain a slightly different polynomial approximation $\hat{p}$. But *we have no guarantee that $\hat{p}$ is the best minimax approximation to f among the polynomials whose degree-i coefficient is a multiple of* 2^{-m_i}. The aim of this paper is to give a way of finding this "best truncated approximation". We have two goals in mind:

—rather low precision (say, around 24 bits), hardware-oriented, for specific-purpose implementations. In such cases, to minimize multiplier sizes (which increases speed and saves silicon area), the values of m_i, for $i \geq 1$, should be very small. The degrees of the polynomial approximations are low. Typical recent examples are given in [Wei et al. 2001; Pineiro et al. 2001]. Roughly speaking, what matters here is to reduce the cost (in terms of delay and area) without making the accuracy unacceptable;

—single-precision or double-precision, software-oriented, for implementation on current general-purpose microprocessors. Using table-driven methods, such as the ones suggested by Tang [1989; 1990; 1991; 1992], the degree of the polynomial approximations can be made rather low. Roughly speaking, what matters in that case is to get very high accuracy, without making the cost (in terms of delay and memory) unacceptable.

One could object that the m_i's are not necessarily known *a priori*. For instance, if one wishes a coefficient to be exactly representable in double precision arithmetic, one needs to know the order of magnitude of that coefficient to know what value of m_i corresponds to that wish. And yet, in practice, good approximations of the same degree to a given function have coefficients that are very close (the approach given in Section 3.1 allows to show that), so that using our approach with possibly two different values of m_i if the degree-i coefficient of the minimax approximation is very close to a power of i suffices.

It is important to notice that our polynomial approximations will be computed once for all, and will be used very frequently (indeed, several billion times, for an elementary function program put in a widely distributed library). Hence, if it remains reasonably fast, the speed of an algorithm that computes adequate approximations is not extremely important. However, in the practical cases we have studied so far, our method will very quickly give a result.

In this paper, we provide a general and efficient method for finding the "best truncated approximation(s)" (it is not necessarily unique). It consists in building a polytope $\mathfrak{P}$ of $\mathbb{R}^{n+1}$, to which the numerators of the coefficients of this (these) "best truncated approximation(s)" belong, such that $\mathfrak{P}$ contains a number as small as possible of points of $\mathbb{Z}^{n+1}$. Once it is achieved, we do an exhaustive search by computing the norms[1]

$$\left\| \frac{a_0}{2^{m_0}} + \frac{a_1}{2^{m_1}} x + \cdots + \frac{a_n}{2^{m_n}} x^n - f \right\|_{\infty,[a,b]}$$

with $(a_0, a_1, \ldots, a_n) \in \mathfrak{P} \cap \mathbb{Z}^{n+1}$.

The method presented here is very flexible since it applies also when we impose supplementary constraints on the "truncated polynomials", and/or when distance $||.||_{\mathrm{rel},[a,b]}$ is considered. For example, the search can be restricted to odd or even polynomials or, more generally, to polynomials with some fixed coefficients. This is frequently useful: one may for instance wish the computed value of $\exp(x)$ to be exactly one if $x = 0$ (hence, requiring the degree-0 coefficient of the polynomial approximation to be 1).

Of course, one would like to take into account the roundoff error that occurs during polynomial evaluation: getting the polynomial, with constraints on the size of the coefficients, that minimizes the total (approximation plus roundoff) error would be extremely useful. Although we are currently working on that problem, we do not yet have a solution: first, it is very algorithm-and-architecture dependent (for instance, some architectures have an extended internal precision), second, since the largest roundoff error and the largest approximation error are extremely unlikely to be attained at the same points exactly, the total error is difficult to predict accurately.

And yet, here are a few observations that lead us to believe that in many practical cases, our approach will give us polynomials that will be very close to these "ideal" approximations. Please notice that these observations are merely intuitive feelings, and that one can always build cases for which the "ideal" approximations differ from the ones we compute.

(1) Good approximations of the same degree to a given function have coefficients that are very close in practice. Indeed, the approach given in Section 3.1 allows to show that.

(2) When evaluating two polynomials whose coefficients are very close, on variables that belong to the same input interval, the largest roundoff errors will be very close too.

(3) In all practical cases, the approximation error oscillates slowly, whereas the roundoff error varies very quickly, so that if the input interval is reasonably small, an error very close to the maximum error is reached near any point. This is illustrated in Figures 1 and 2: we have defined p as the polynomial obtained by rounding to the nearest double precision number each coefficient of the degree-5 minimax approximation to e^x in $[0, 1/128]$. Figure 1 shows the difference $p(x) - e^x$ (approximation error), and Figure 2 shows the difference between the computed value and the exact value of $p(x)$, assuming Horner's scheme is used, in double precision arithmetic.

[1]So far, we have computed these norms using the `infnorm` function of Maple. Our research group is working on a C implementation that will use multiple precision interval arithmetic to get certified upper and lower bounds on the infinite norm of a regular enough function.

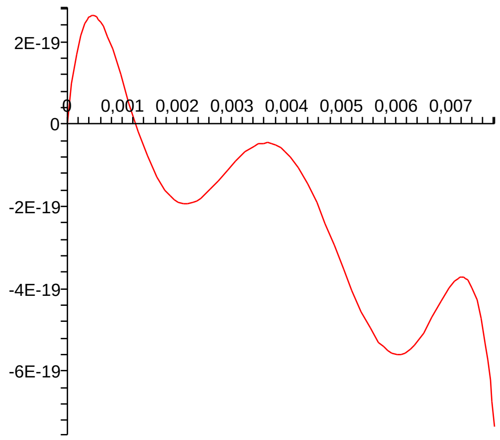

Fig. 1. Approximation error: we have plotted the difference $p(x) - e^x$.

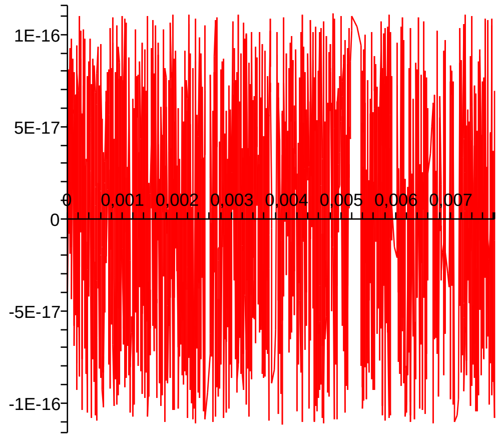

Fig. 2. Roundoff error: we have plotted the difference between the exact and computed values of $p(x)$. The computations are performed using Horner's scheme, in double precision, without using a larger internal format.

These observations tend to indicate that for all candidate polynomials the roundoff errors will be very close, and the total error will be close to the sum of the approximation and roundoff errors. Hence, the "best" polynomial when considering the approximation error only will be very close to the "best" polynomial when considering approximation and roundoff errors.

Of course, these observations are not proofs: they just result from some experiments, and we are far from being able to solve the general problem of finding the "best" polynomial, with size constraints on coefficients, when considering approximation and roundoff errors. Hopefully, these remarks will one day help to build a more general method.

The outline of the paper is the following. We give an account of Chebyshev polynomials and some of their properties in Section 2. In Section 3, we first provide a method based on Chebyshev polynomials that partially answers to the problem and then, we give a general and efficient method based on polytopes that finds a "best truncated approximation" of a function f over a compact interval $[a, b]$. Despite the fact that it is less efficient and general than the polytope method, we present the method based on Chebyshev polynomials because this approach seems interesting in itself, is simple and gives results easy to use and, moreover, might be useful in other problems. We end Section 3 with a remark illustrating the flexibility of our method. We finish with some examples in Section 4. We complete the paper with three appendices. In the first one, we collect the proofs of the statements given in Section 2. In the second one, we prove a lemma, used in Subsection 3.2, that implies in particular the existence of a best truncated polynomial approximation. In the last one, we give a worked example of the methods presented here.

To end this introduction, let us mention that a C implementation of our method is in process and also that the method applies to some signal processing problems, namely finding the rational linear combination of cosines with constraints on the size in bits of the rational coefficients in order to implement (in software or hardware) digital FIR filters. This will be the purpose of a future paper.

As we only deal with the supremum norm, wherever there is no ambiguity, we will write $||\cdot||_I$ instead of $||\cdot||_{\infty,I}$ where I is any real set.

2. SOME REMINDERS ON CHEBYSHEV POLYNOMIALS

Definition 1 Chebyshev polynomials. *The Chebyshev polynomials can be defined either by the recurrence relation*

$$\begin{cases} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_n(x) &= 2xT_{n-1}(x) - T_{n-2}(x); \end{cases}$$

or by

$$T_n(x) = \begin{cases} \cos\left(n \cos^{-1} x\right) & \text{for } |x| \le 1, \\ \cosh\left(n \cosh^{-1} x\right) & \text{for } x > 1. \end{cases}$$

A presentation of Chebyshev polynomials can be found in [Borwein and Erdélyi 1995] and especially in [Rivlin 1990]. These polynomials play a central role in approximation theory. The following property is easily derived from Definition 1.

Property 1. *For $n \ge 0$, we have*

$$T_n(x) = \frac{n}{2} \sum_{k=0}^{\lfloor n/2 \rfloor} (-1)^k \frac{(n-k-1)!}{k!(n-2k)!} (2x)^{n-2k}.$$

Hence, T_n has degree n and its leading coefficient is 2^{n-1}. It has n real roots, all strictly between -1 and 1.

We recall that a *monic* polynomial is a polynomial whose leading coefficient is 1. The following statement is a well known and remarkable property of Chebyshev Polynomials.

Property 2 Monic polynomials of smallest norm. *Let $a, b \in \mathbb{R}$, $a < b$. The monic degree-n polynomial having the smallest $||\cdot||_{[a,b]}$ norm is*

$$\frac{(b-a)^n}{2^{2n-1}} T_n\left(\frac{2x-b-a}{b-a}\right).$$

In the following, we will make use of the polynomials

$$T_n^*(x) = T_n(2x-1).$$

We have (see [Fox and Parker 1972, Chap. 3] for example) $T_n^*(x) = T_{2n}(x^{1/2})$, hence all the coefficients of T_n^* are nonzero integers.

Now, we state two propositions that generalize Property 2 when dealing with intervals of the form $[0, a]$ and $[-a, a]$.

Proposition 1. *Let $a \in (0, +\infty)$, define*

$$\alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_n x^n = T_n^*\left(\frac{x}{a}\right).$$

Let k be an integer, $0 \le k \le n$, the polynomial

$$\frac{1}{\alpha_k} T_n^*\left(\frac{x}{a}\right)$$

has the smallest $||\cdot||_{[0,a]}$ norm among the polynomials of degree at most n with a degree-k coefficient equal to 1. That norm is $|1/\alpha_k|$.

Remark 1. Moreover, when $k = n = 0$ or $1 \le k \le n$, we can show that this polynomial is the only one having this property. We do not give the proof of this uniqueness property in this paper since we only need the existence result in the sequel.

Proposition 2. *Let $a \in (0, +\infty)$, $p \in \mathbb{N}$, define*

$$\beta_{0,p} + \beta_{1,p} x + \beta_{2,p} x^2 + \cdots + \beta_{p,p} x^p = T_p\left(\frac{x}{a}\right).$$

Let k and n be integers, $0 \le k \le n$.

—If k and n are both even or odd, the polynomial

$$\frac{1}{\beta_{k,n}} T_n\left(\frac{x}{a}\right)$$

has the smallest $||\cdot||_{[-a,a]}$ norm among the polynomials of degree at most n with a degree-k coefficient equal to 1. That norm is $|1/\beta_{k,n}|$.

—Else, the polynomial

$$\frac{1}{\beta_{k,n-1}} T_{n-1}\left(\frac{x}{a}\right)$$

has the smallest $||\cdot||_{[-a,a]}$ norm among the polynomials of degree at most n with a degree-k coefficient equal to 1. That norm is $|1/\beta_{k,n-1}|$.

3. GETTING THE "TRUNCATED" POLYNOMIAL THAT IS CLOSEST TO A FUNCTION ON A COMPACT INTERVAL

Let a, b be two real numbers, let f be a function defined on $[a,b]$ and m_0, m_1, ..., m_n be $n+1$ integers. Define $\mathcal{P}_n^{[m_0,m_1,\dots,m_n]}$ as the set of the polynomials of degree less than or equal to n whose degree-i coefficient is a multiple of 2^{-m_i} for all i between 0 and n (we will call these polynomials "truncated polynomials"), that is to say

$$\mathcal{P}_n^{[m_0,m_1,\dots,m_n]} = \left\{ \frac{a_0}{2^{m_0}} + \frac{a_1}{2^{m_1}}x + \cdots + \frac{a_n}{2^{m_n}}x^n,\ a_0,\dots,a_n \in \mathbb{Z} \right\}.$$

Let p be the minimax approximation to f on $[a,b]$. Define $\hat{p}$ as the polynomial whose degree-i coefficient is obtained by rounding the degree-i coefficient of p to the nearest multiple of 2^{-m_i} (with an arbitrary choice in case of a tie) for $i = 0, \dots, n$: $\hat{p}$ is an element of $\mathcal{P}_n^{[m_0,m_1,\dots,m_n]}$.

Also define ϵ and $\hat{\epsilon}$ as

$$\epsilon = ||f-p||_{[a,b]} \text{ and } \hat{\epsilon} = ||f-\hat{p}||_{[a,b]}.$$

We assume that $\hat{\epsilon} \neq 0$.

We state our problem as follows. Let $K \geq \epsilon$, we are looking for a truncated polynomial $p^\star \in \mathcal{P}_n^{[m_0,m_1,\dots,m_n]}$ such that

$$||f-p^\star||_{[a,b]} = \min_{q \in \mathcal{P}_n^{[m_0,m_1,\dots,m_n]}} ||f-q||_{[a,b]}$$

and

$$||f-p^\star||_{[a,b]} \leq K. \tag{1}$$

Lemma 2 in Appendix 2 implies that the number of truncated polynomials satisfying (1) is finite.

When $K = \hat{\epsilon}$, this problem has a solution since $\hat{p}$ satisfies (1). It should be noticed that, in that case, $p^\star$ is not necessarily equal to $\hat{p}$.

We can put, for example, $K = \lambda\hat{\epsilon}$ with $\lambda \in [\epsilon/\hat{\epsilon}, 1]$.

3.1 A partial approach through Chebyshev polynomials

The term "partial" refers to the fact that the intervals we can deal with in this subsection must be of the form $[0,a]$ or $[-a,a]$ where $a > 0$. This restriction comes from the following two problems:

(1) we do not have in the general case $[a,b]$ a simple analogous of Propositions 1 and 2,

(2) from a given polynomial and a given interval, a simple change of variable allows to reduce the initial approximation problem to another approximation problem with an interval of the form $[0,a]$ or $[-a,a]$. Unfortunately, this change of variables (that does not keep the size of the coefficients invariant) leads to a system of inequalities of the form of those we face in Subsection 3.2. Then, we have

to perform additional operations in order to produce candidate polynomials. Doing so, we lose the main interest -its simplicity- of the approach through Chebyshev polynomials in the cases $[0, a]$ and $[-a, a]$.

We will only deal with intervals $[0, a]$ where $a > 0$ since the method presented in the following easily adapts to intervals $[-a, a]$ where $a > 0$.

In this subsection, we compute bounds such that if the coefficients of a polynomial $q \in \mathcal{P}_n^{[m_0,m_1,\ldots,m_n]}$ are not within these bounds, then

$$||p - q||_{[0,a]} > \epsilon + K.$$

Knowing these bounds will make possible an exhaustive searching of $p^\star$. To do that, consider a polynomial q whose degree-i coefficient is $p_i + \delta_i$. From Proposition 1, we have

$$||q - p||_{[0,a]} \geq \frac{|\delta_i|}{|\alpha_i|},$$

where α_i is the nonzero degree-i coefficient of $T_n^*(x/a)$. Now, if q is at a distance greater than $\epsilon + K$ from p, it cannot be $p^\star$ since

$$||q - f||_{[0,a]} \geq ||q - p||_{[0,a]} - ||p - f||_{[0,a]} > K.$$

Therefore, if there exists i, $0 \leq i \leq n$, such that

$$|\delta_i| > (\epsilon + K)|\alpha_i|$$

then $||q - p||_{[0,a]} > \epsilon + K$ and therefore $q \neq p^\star$. Hence, the degree-i coefficient of $p^\star$ necessarily lies in the interval $[p_i - (\epsilon + K)|\alpha_i|, p_i + (\epsilon + K)|\alpha_i|]$. Thus we have

$$\underbrace{\lceil 2^{m_i}(p_i - (\epsilon + K)|\alpha_i|)\rceil}_{c_i} \leq 2^{m_i} p_i^\star \leq \underbrace{\lfloor 2^{m_i}(p_i + (\epsilon + K)|\alpha_i|)\rfloor}_{d_i}, \tag{2}$$

since $2^{m_i} p_i^\star$ is an integer. Note that, as $0 \in [0, a]$, Condition (1) implies in particular

$$f(0) - K \leq p_0^\star \leq f(0) + K$$

i.e., since $2^{m_0} p_0^\star$ is an integer,

$$\underbrace{\lceil 2^{m_0}(f(0) - K)\rceil}_{c_0'} \leq 2^{m_0} p_0^\star \leq \underbrace{\lfloor 2^{m_0}(f(0) + K)\rfloor}_{d_0'}.$$

We replace c_0 with $\max(c_0, c_0')$ and d_0 with $\min(d_0, d_0')$.

For $i = 0, \ldots, n$, we have $d_i - c_i + 1$ possible values for the integer $2^{m_i} p_i^\star$. This means that we have $\prod_{i=0}^{n}(d_i - c_i + 1)$ candidate polynomials. If this amount is small enough, we search for $p^\star$ by computing the norms $||f - q||_{[0,a]}$, q running among the possible polynomials. Otherwise, we need an additional step to decrease the number of candidates. Hence, we now give a method for this purpose. It allows us to reduce the number of candidate polynomials dramatically and applies more generally to intervals of the form $[a, b]$ where a and b are any real numbers.

3.2 A general and efficient approach through polytopes

From now on, we will deal with intervals of the form $[a, b]$ where a and b are any real numbers.

We recall the following definitions from [Schrijver 2003].

Definitions 1. *Let $k \in \mathbb{N}$.*

A subset $\mathfrak{P}$ of $\mathbb{R}^k$ is called a polyhedron if there exists an $m \times k$ matrix A with real coefficients and a vector $b \in \mathbb{R}^m$ (for some $m \geq 0$) such that $\mathfrak{P} = \{x \in \mathbb{R}^k | Ax \leq b\}$.

A subset $\mathfrak{P}$ of $\mathbb{R}^k$ is called a polytope if $\mathfrak{P}$ is a bounded polyhedron.

A polyhedron (resp. polytope) $\mathfrak{P}$ is called rational if it is determined by a rational system of linear inequalities.

The $n+1$ inequalities given by (2) define a rational polytope of $\mathbb{R}^{n+1}$ which the numerators of $p^\star$ (i.e. the $2^{m_i} p_i^\star$) belong to. The idea[2] is to build a polytope $\mathfrak{P}$, still containing the $2^{m_i} p_i^\star$, such that $\mathfrak{P} \cap \mathbb{Z}^{n+1}$ is the smallest possible, which means that the number of candidate polynomials is the smallest possible, in order to reduce as much as possible the final step of computation of supremum norms. Once we get this polytope, we need an efficient way of producing these candidates, i.e., an efficient way of scanning the integer points (that is to say points with integer coordinates) of the rational polytope we built. Several algorithms allow to achieve that : the one given in [Ancourt and Irigoin 1991] uses the Fourier-Motzkin pairwise elimination, the one given in [Feautrier 1988; Collard et al. 1995] is a parameterized version of the Dual Simplex method and the one given in [Le Verge et al. 1994] is based on the dual representation of polyhedra used in Polylib [The Polylib Team 2004]. The last two algorithms allow us to produce in an "optimized" way[3] the loops in our final program of exhaustive search. Note that these algorithms have, at worst, the complexity of integer linear programming[4]. Now, let us give the details of the method.

First, we can notice that the previous approach handles the unknowns $p_i^\star$ separately, which seems unnatural. Hence, the basic aim of the method is to construct a polytope defined by inequalities that take into account in a more satisfying way the dependence relations between the unknowns. This polytope should contain less points of $\mathbb{Z}^{n+1}$ than the one built from Chebyshev polynomials' approach. The examples of Section 4 indicate that this seems to be the case (and the improvements can be dramatic).

Condition (1) means

$$f(x) - K \leq \sum_{j=0}^{n} p_j^\star x^j \leq f(x) + K \tag{3}$$

for all $x \in [a, b]$.

The idea is to consider inequalities (3) for a certain number (chosen by the user) of values of $x \in [a, b]$. As we want to build rational polytopes, these values must be rational numbers. We propose the following choice of values. Let A be a rational approximation to a greater than or equal to a, let B be a rational approximation

[2] After the submission of this paper, we read that this idea has already been proposed in [Habsieger and Salvy 1997], a paper dealing with number-theoretical issues, but the authors did not have any efficient method for scanning the integer points of the polytope.

[3] By "optimized", we mean that all points of the polytope are scanned only once.

[4] In fact, the algorithm given in [Feautrier 1988; Collard et al. 1995] has in the situation we are facing the complexity of rational linear programming.

to b less than or equal to b and such that $B > A$, let d be a nonzero natural integer chosen by the user. We show in Lemma 2 in Appendix 2 that we must have $d \geq n$ in order to ensure that we get a polytope (which implies the finiteness of the number of the "best truncated approximation(s)"). We consider the rational values[5] $x_i = A + \frac{i}{d}(B - A)$, $i = 0, \ldots, d$. Then again, since the polytope has to be rational, we compute, for $i = 0, \ldots, d$ two rational numbers l_i and u_i that are rational approximations to, respectively, $f(x_i) - K$ and $f(x_i) + K$ such that $l_i \leq f(x_i) - K$ and $u_i \geq f(x_i) + K$. The rational polytope $\mathfrak{P}$ searched is therefore defined by the inequalities

$$l_i \leq \sum_{j=0}^{n} p_j^\star x_i^j \leq u_i, \quad i = 0, \ldots, d. \tag{4}$$

If $\mathfrak{P} \cap \mathbb{Z}^{n+1}$ is small enough (this can be estimated thanks to Polylib), we start our exhaustive search by computing the norms

$$\left\| \frac{a_0}{2^{m_0}} + \frac{a_1}{2^{m_1}} x + \ldots \frac{a_n}{2^{m_n}} x^n - f \right\|_{[a,b]} \tag{5}$$

with $(a_0, a_1, \ldots, a_n) \in \mathfrak{P} \cap \mathbb{Z}^{n+1}$. This set can be scanned efficiently thanks to one of the algorithms given in [Ancourt and Irigoin 1991], [Feautrier 1988; Collard et al. 1995] or [Le Verge et al. 1994] that we quoted above. Or else, we increase the value of the parameter d in order to construct another rational polytope $\mathfrak{P}'$ that contains fewer elements of $\mathbb{Z}^{n+1}$. We must point out that assuming that the new parameter is greater than d does not necessarily lead to a new polytope with fewer elements of $\mathbb{Z}^{n+1}$ inside (it is easy to show such counter-examples) but it is reasonable to expect that, generally, a polytope built with a greater parameter should contain fewer elements of $\mathbb{Z}^{n+1}$ inside since it is defined from a larger number of inequalities (4) or in other words, as our discretization method is done using a larger number of rational points, which should allow to restrict the number of possible candidates since there are more conditions to satisfy. It is indeed the case if we choose any positive integer multiple of d as new parameter. In this case, the new polytope $\mathfrak{P}'$, associated to the parameter νd, with $\nu \in \mathbb{N}^*$, is a subset of $\mathfrak{P}$: $\mathfrak{P}$ is built from the set of rational points $\{A + \frac{i}{d}(B - A)\}_{i=0,\ldots,d}$ which is a subset of $\{A + \frac{j}{\nu d}(B - A)\}_{j=0,\ldots,\nu d}$ from which $\mathfrak{P}'$ is built.

Remark 2. As we said in the introduction, this method is very flexible. We give some examples to illustrate that.

—If we restrict our search to odd truncated polynomials (in this case, we put $n = 2k + 1$ and we must have $d \geq k$), it suffices to replace inequalities (4) with

$$l_i \leq \sum_{j=0}^{k} p_j^\star x_i^{2j+1} \leq u_i, \quad i = 0, \ldots, d$$

to create a polytope $\mathfrak{P}$ of $\mathbb{R}^{k+1}$ whose points with integer coordinates we scan.

[5] Choosing equally-spaced rational values seems a natural choice, when dealing with regular enough functions. And yet, in a few cases, we get a better result with very irregularly-spaced points. This is something we plan to investigate in the near future.

—If we restrict our search to truncated polynomials whose some coefficients have fixed values. For example, if we assume that the truncated polynomials sought have constant term equal to 1, it suffices to replace inequalities (4) with

$$l_i \leq 1 + \sum_{j=1}^{n} p_j^\star x_i^{2j+1} \leq u_i, \quad i = 0, \ldots, d$$

to create a polytope $\mathfrak{P}$ of $\mathbb{R}^n$ whose we scan the points with integer coordinates (we must have $d \geq n-1$).

—Our method also applies to the search for the best truncated polynomial with respect to the relative error distance $||\cdot||_{\mathrm{rel},[a,b]}$ defined in the introduction. In this case, we can state the problem as follows. Let $K \geq 0$, we search for a truncated polynomial $p^\star \in \mathcal{P}_n^{[m_0,m_1,\ldots,m_n]}$ such that

$$||f - p^\star||_{\mathrm{rel},[a,b]} = \min_{q \in \mathcal{P}_n^{[m_0,m_1,\ldots,m_n]}} ||f - q||_{\mathrm{rel},[a,b]}$$

and

$$||f - p^\star||_{\mathrm{rel},[a,b]} \leq K. \tag{6}$$

It suffices to consider the inequalities

$$-K|f(x)| - f(x) \leq \sum_{j=0}^{n} p_j^\star x^j \leq K|f(x)| + f(x)$$

for at least $n+1$ distinct rational values of $x \in [a,b]$ to make a polytope to which we apply the method presented above.

4. EXAMPLES

We implemented in Maple the method given in Subsection 3.1 and we started developing a C library that implements the method described in Subsection 3.2.

For producing the results presented in this section, we implemented the approach through polytopes in a C program of our own, based on the polyhedral library Polylib [The Polylib Team 2004]. This allows to treat a lot of examples but it is too roughly done to tackle with approximations of degree 15 to 20 for example. Our goal is to implement the method presented here in a C library, which should use Polylib and PIP [Feautrier 2003], that implements the parametric integer linear programming solver given in [Feautrier 1988; Collard et al. 1995], for scanning the integer points of the polytope. The choice of PIP instead of the algorithm given in [Le Verge et al. 1994] is due to the fact that our polytope may, in some cases, have a large amount of vertices (due to the need of a large amount of points x_j, i.e. constraints, when building the polytope) which can generate memory problems if we use [Le Verge et al. 1994]. We also need the MPFR multiple-precision library [The Spaces project 2004] since we face precision problems when forming the polytope, and the GMP library [Granlund 2002] since the polytope is defined by a matrix and a vector which may have very large integer coefficients. The GMP library is also used inside some Polylib computations.

Table II. Examples

	f	$[a,b]$	m	ϵ	$\hat{\epsilon}$	K
1	cos	$\left[0, \frac{\pi}{4}\right]$	$[12, 10, 6, 4]$	$1.135...10^{-4}$	$6.939...10^{-4}$	$\hat{\epsilon}/2$
2	exp	$\left[0, \frac{1}{2}\right]$	$[15, 14, 12, 10]$	$2.622...10^{-5}$	$3.963...10^{-5}$	$< \hat{\epsilon}$
3	exp	$\left[0, \log\left(1+\frac{1}{2048}\right)\right]$	$[56, 45, 33, 23]$	$1.184...10^{-17}$	$2.362...10^{-17}$	$< \hat{\epsilon}$
4	$\arctan(1+x)$	$\left[0, \frac{1}{4}\right]$	$[24, 21, 18, 17, 16]$	$2.381...10^{-8}$	$3.774...10^{-8}$	$< \hat{\epsilon}$
5	exp	$\left[-\frac{\log(2)}{256}, \frac{\log(2)}{256}\right]$	$[25, 17, 9]$	$8.270...10^{-10}$	$3.310...10^{-9}$	$< \hat{\epsilon}$
6	exp	$\left[-\frac{\log(2)}{256}, -\frac{\log(2)}{256}\right]$	$[28, 19, 9]$	$8.270...10^{-10}$	$3.310...10^{-9}$	$< \hat{\epsilon}$
7	$\frac{\log(3/4+x)}{\log(2)}$	$[-1/4, 1/4]$	$[12, 9, 7, 5]$	$6.371...10^{-4}$	$7.731...10^{-4}$	$< \hat{\epsilon}$
8	$\frac{\log(\sqrt{2}/2+x)}{\log(2)}$	$\left[\frac{1-\sqrt{2}}{2}, \frac{2-\sqrt{2}}{2}\right]$	$[12, 9, 7, 5]$	$6.371...10^{-4}$	$9.347...10^{-4}$	$< \hat{\epsilon}$

Table III. Corresponding results

	Chebyshev	Polytope (d)	T_1	T_2	Gain of accuracy in bits
1	330	1 (4)	0.62	0.26	≈ 1.5
2	84357	9 (20)	0.51	2.18	≈ 0.375
3	9346920	15 (20)	0.99	1.77	≈ 0.22
4	192346275	1 (20)	0.15	0.55	≈ 0.08
5	1	0 (4)	0.05	0	0
6	4	1 (4)	0.03	0.10	≈ 0.41
7	38016	2 (15)	0.13	0.69	≈ 0.06
8		12 (20)	0.59	5.16	≈ 0.26

We put some examples in Table II and we group the corresponding results in Table III. In the last column of Table II, the notation $< \hat{\epsilon}$ means that we chose a value slightly smaller than $\hat{\epsilon}$, namely $\hat{\epsilon}$ rounded down to four fractional digits.

In the first column of Table III, one can find the amount of candidates given by Chebyshev's approach. In the second, we give the number of candidates given by the approach through polytopes and the corresponding parameter d. T_1 denotes the time in seconds for producing the candidate polynomials with the polytope method. T_2 denotes the time in seconds for computing the norms (5), which, for the moment (cf. footnote 1), is done with Maple 9, in which the value of `Digits` was fixed equal to 20. In the last column, we give the gain of accuracy in bits that we get if we use the best truncated polynomial instead of polynomial $\hat{p}$.

All the computations were done on a 2.53 GHz Pentium 4 computer running Debian Linux with 256 MB RAM. The compiler used was `gcc` without optimization.

4.1 Choice of the examples

These examples correspond to common cases that occur in elementary and special function approximation [Muller 1997]. Examples 1, 2 and 4 are of immediate interest. Example 3 is of interest for implementing the exponential function in double precision arithmetic, using a table-driven method such as Tang's method [Tang 1991; 1992]. Examples 5 and 6 also correspond to a table-driven implementation of the exponential function, in single precision. Examples 7 and 8 aim at producing very cheap approximations to logarithms in $[1/2, 1]$, for special purpose applications.

Figure 3 shows how the number of integer points of the polytope drops when K decreases, in the case of example 2.

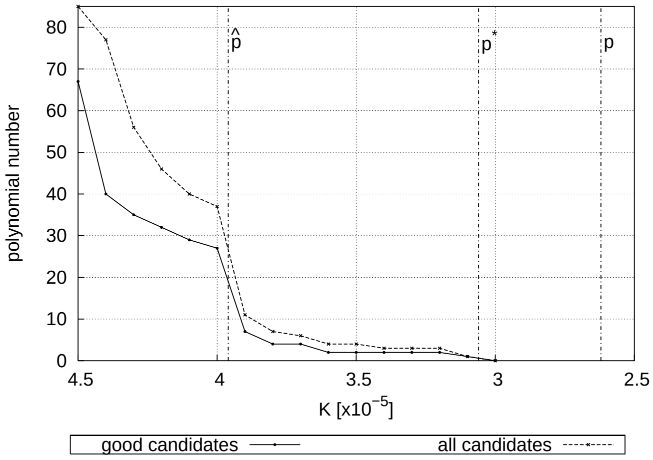

Fig. 3. This figure shows how the number of integer points of the polytope drops when K decreases, in the case of example 2. The term "all candidates" means the integer points of the polytope. By "good candidate" we mean the points that fulfill the requirement (1).

Acknowledgements

We would like to thank the referees, who greatly helped to improve the quality of the manuscript. We would also like to thank Nicolas Boullis and Serge Torres, who greatly helped computing the examples, and Paul Feautrier and Cédric Bastoul, whose expertise in polyhedric computations has been helpful.

REFERENCES

ANCOURT, C. AND IRIGOIN, F. 1991. Scanning polyhedra with do loops. In *Proceedings of the 3rd ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP'91).* ACM Press, New York, 39–50.

BORWEIN, P. AND ERDÉLYI, T. 1995. *Polynomials and Polynomials Inequalities.* Graduate Texts in Mathematics, 161. Springer-Verlag, New York.

COLLARD, J.-F., FEAUTRIER, P., AND RISSET, T. 1995. Construction of do loops from systems of affine constraints. *Parallel Processing Letters 5*, 421–436.

CORNEA, M., HARRISON, J., AND TANG, P. T. P. 2002. *Scientific Computing on Itanium-Based Systems.* Intel Press.

FEAUTRIER, P. 1988. Parametric integer programming. *RAIRO Rech. Opér. 22,* 3, 243–268.

FEAUTRIER, P. 2003. PIP/Piplib, a parametric integer linear programming solver. `http://www.prism.uvsq.fr/~cedb/bastools/piplib.html`.

FOX, L. AND PARKER, I. B. 1972. *Chebyshev Polynomials in Numerical Analysis.* Oxford Mathematical Handbooks. Oxford University Press, London-New York-Toronto.

GRANLUND, T. 2002. GMP, the GNU multiple precision arithmetic library, version 4.1.2. `http://www.swox.com/gmp/`.

HABSIEGER, L. AND SALVY, B. 1997. On integer Chebyshev polynomials. *Mathematics of Computation 66,* 218, 763–770.

HART, J. F., CHENEY, E. W., LAWSON, C. L., MAEHLY, H. J., MESZTENYI, C. K., RICE, J. R., THACHER, H. G., AND WITZGALL, C. 1968. *Computer Approximations.* John Wiley & Sons, New York.

LE VERGE, H., VAN DONGEN, V., AND WILDE, D. K. 1994. Loop nest synthesis using the polyhedral library. Tech. Rep. INRIA Research Report RR-2288, INRIA. May.

Markstein, P. 2000. *IA-64 and Elementary Functions : Speed and Precision.* Hewlett-Packard Professional Books. Prentice Hall.

Muller, J.-M. 1997. *Elementary Functions, Algorithms and Implementation.* Birkhäuser, Boston.

Pineiro, J., Bruguera, J., and Muller, J.-M. 2001. Faithful powering computation using table look-up and a fused accumulation tree. In *Proc. of the 15th IEEE Symposium on Computer Arithmetic (Arith-15)*, Burgess and Ciminiera, Eds. IEEE Computer Society Press, Los Alamitos, CA, 40–58.

Remes, E. 1934. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation. *C.R. Acad. Sci. Paris 198*, 2063–2065.

Rivlin, T. J. 1990. *Chebyshev polynomials. From approximation theory to algebra (Second edition).* Pure and Applied Mathematics. John Wiley & Sons, New York.

Schrijver, A. 2003. *Combinatorial optimization. Polyhedra and efficiency. Vol. A.* Algorithms and Combinatorics, 24. Springer-Verlag, Berlin.

Story, S. and Tang, P. T. P. 1999. New algorithms for improved transcendental functions on IA-64. In *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, Koren and Kornerup, Eds. IEEE Computer Society Press, Los Alamitos, CA, 4–11.

Tang, P. T. P. 1989. Table-driven implementation of the exponential function in IEEE floating-point arithmetic. *ACM Transactions on Mathematical Software 15,* 2 (June), 144–157.

Tang, P. T. P. 1990. Table-driven implementation of the logarithm function in IEEE floating-point arithmetic. *ACM Transactions on Mathematical Software 16,* 4 (Dec.), 378–400.

Tang, P. T. P. 1991. Table lookup algorithms for elementary functions and their error analysis. In *Proceedings of the 10th IEEE Symposium on Computer Arithmetic*, P. Kornerup and D. W. Matula, Eds. IEEE Computer Society Press, Los Alamitos, CA, 232–236.

Tang, P. T. P. 1992. Table-driven implementation of the expm1 function in IEEE floating-point arithmetic. *ACM Transactions on Mathematical Software 18,* 2 (June), 211–222.

The Polylib Team. 2004. Polylib, a library of polyhedral functions, version 5.20.0. `http://icps.u-strasbg.fr/polylib/`.

The Spaces project. 2004. MPFR, the multiple precision floating point reliable library, version 2.0.3. `http://www.mpfr.org`.

Wei, B., Cao, J., and Cheng, J. 2001. High-performance architectures for elementary function generation. In *Proc. of the 15th IEEE Symposium on Computer Arithmetic (Arith-15)*, Burgess and Ciminiera, Eds. IEEE Computer Society Press, Los Alamitos, CA, 136–146.

Appendix 1. Proofs of Propositions 1 and 2

Proving those propositions first requires the following two results. The first one is well known.

Property 3. *There are exactly* $n+1$ *values* $x_0, x_1, x_2, \ldots, x_n$ *such that*

$$-1 = x_0 < x_1 < x_2 < \cdots < x_n = 1,$$

which satisfy

$$T_n(x_i) = (-1)^{n-i} \max_{x \in [-1,1]} |T_n(x)| \quad \forall i,\ i = 0, \ldots, n.$$

That is, the maximum absolute value of T_n *is attained at the* x_i*'s, and the sign of* T_n *alternates at these points.*

Proof. These extreme points are simply the points $\cos(k\pi/n)$, $k = 0, \ldots, n$. □

Lemma 1. *Let* $(\delta_i)_{i=0,\ldots,n}$ *be an increasing sequence of nonnegative integers and*

$$P(x) = a_0 x^{\delta_0} + \cdots + a_n x^{\delta_n} \in \mathbb{R}[x],$$

then either $P = 0$ *or* P *has at most* n *zeros in* $(0, +\infty)$.

Proof. By induction on n. For $n = 0$, it is straightforward. Now we assume that the property is true until the rank n. Let

$$P(x) = a_0 x^{\delta_0} + \cdots + a_n x^{\delta_n} + a_{n+1} x^{\delta_{n+1}} \in \mathbb{R}[x]$$

with $0 \leq \delta_0 < \cdots < \delta_{n+1}$ and $a_0 a_1 \ldots a_{n+1} \neq 0$. Assume that P has at least $n+2$ zeros in $(0, +\infty)$. Then $P_1 = P/x^{\delta_0}$ has at least $n+2$ zeros in $(0, +\infty)$.

Thus, the nonzero polynomial

$$P_1'(x) = (\delta_1 - \delta_0) a_1 x^{\delta_1 - \delta_0} + \cdots + (\delta_{n+1} - \delta_0) a_{n+1} x^{\delta_{n+1} - \delta_0}$$

has, from Rolle's Theorem, at least $n+1$ zeros in $(0, +\infty)$, which contradicts the induction hypothesis. □

Proof of Proposition 1. From Property 3, there exist $0 = \eta_0 < \eta_1 < \cdots < \eta_n = 1$ such that

$$\alpha_k^{-1} T_n^*(\eta_i) = \alpha_k^{-1} (-1)^{n-i} \|T_n^*(\cdot/a)\|_{[0,a]} = \alpha_k^{-1} (-1)^{n-i}.$$

Let $q(x) = \sum_{\substack{j=0,\\ j \neq k}}^{n} c_j x^j \in \mathbb{R}[x]$ satisfy $\|x^k - q(x)\|_{[0,a]} < |\alpha_k^{-1}|$. Then the polynomial $P(x) = \alpha_k^{-1} T_n^*(x) - (x^k - q(x))$ has the form $\sum_{\substack{j=0,\\ j \neq k}}^{n} d_j x^j$ and is not identically zero. As it changes sign between any two consecutive extrema of T_n^*, it has at least n zeros in $(0, +\infty)$. Hence, from Lemma 1, it must vanish identically which is the contradiction desired. □

Proof of Proposition 2. We assume that n is even since a straightforward adaptation of the proof in this case gives the proof of the case where n is odd.

First, we suppose k even. Let

$$P(x) = a_n x^n + \cdots + a_{k+1}x^{k+1} + x^k + a_{k-1}x^{k-1} + \cdots + a_0$$

such that $||P||_{[-a,a]} < |1/\beta_{k,n}|$. Then, for all $x \in [-a, a]$,

$$-\frac{1}{|\beta_{k,n}|} < Q(x) := \frac{P(x) + P(-x)}{2} < \frac{1}{|\beta_{k,n}|}$$

i.e., since k and n are both even, for all $x \in [-a, a]$,

$$-\frac{1}{|\beta_{k,n}|} < Q(x) = a_n x^n + \cdots + a_{k+2}x^{k+2} + x^k + a_{k-2}x^{k-2} + \cdots + a_0 < \frac{1}{|\beta_{k,n}|}.$$

From Property 3 and the inequality $||Q||_{[-a,a]} < |1/\beta_{k,n}| = ||T_n(\cdot/a)||_{[-a,a]}$, the polynomial $R(x) = Q(x) - T_n(x/a)/\beta_{k,n} = \sum_{\substack{j=0,\\ j\neq k/2}}^{n/2} c_j x^{2j}$ changes sign between two consecutive extrema of $T_n(x/a)$. Then, it has at least n distinct zeros in $[-a, a]$ and, more precisely, in $[-a, 0) \cup (0, a]$ since 0 is an extrema of $T_n(x/a)$ as n is even. Hence, the polynomial $R(\sqrt{x}) = \sum_{\substack{j=0,\\ j\neq k/2}}^{n/2} c_j x^j$ has at least $n/2$ distinct zeros in $(0, \sqrt{a}]$: it is identically zero from Lemma 1.

We have just proved that

$$P(x) = \frac{1}{\beta_{k,n}} T_n\left(\frac{x}{a}\right) + \frac{P(x) - P(-x)}{2}.$$

We recall that $|P(x)| < 1/|\beta_{k,n}|$ for all $x \in [-a, a]$. Therefore, we have, by substituting 1 and -1 to x

$$-\frac{1}{|\beta_{k,n}|} - \frac{1}{\beta_{k,n}} T_n\left(\frac{1}{a}\right) < \frac{P(1) - P(-1)}{2} < \frac{1}{|\beta_{k,n}|} - \frac{1}{\beta_{k,n}} T_n\left(\frac{1}{a}\right)$$

and

$$-\frac{1}{|\beta_{k,n}|} - \frac{1}{\beta_{k,n}} T_n\left(-\frac{1}{a}\right) < \frac{P(-1) - P(1)}{2} < \frac{1}{|\beta_{k,n}|} - \frac{1}{\beta_{k,n}} T_n\left(-\frac{1}{a}\right)$$

As n is even, we know that $T_n(1/a) = T_n(-1/a) = 1$. Hence, we obtain

$$0 < \frac{P(1) - P(-1)}{2} < 0$$

which is the contradiction desired.

Now, we suppose k odd. Let

$$P(x) = a_n x^n + \cdots + a_{k+1}x^{k+1} + x^k + a_{k-1}x^{k-1} + \cdots + a_0$$

such that $||P||_{[-a,a]} < |1/\beta_{k,n-1}|$. Then, for all $x \in [-a, a]$,

$$-\frac{1}{|\beta_{k,n-1}|} < Q(x) := \frac{P(x) - P(-x)}{2} < \frac{1}{|\beta_{k,n-1}|}$$

i.e., for all $x \in [-a, a]$,

$$-\frac{1}{|\beta_{k,n-1}|} < Q(x) = a_{n-1}x^{n-1}+\cdots+a_{k+2}x^{k+2}+x^k+a_{k-2}x^{k-2}+\cdots+a_1 < \frac{1}{|\beta_{k,n-1}|}.$$

From Property 3 and the inequality $||Q||_{[-a,a]} < |1/\beta_{k,n-1}| = ||T_{n-1}(\cdot/a)||_{[-a,a]}$, the polynomial $R(x) = Q(x) - T_{n-1}(x/a)/\beta_{k,n-1} = \sum_{\substack{j=0,\\ j\neq(k-1)/2}}^{n/2-1} c_j x^{2j+1}$ changes sign between two consecutive extrema of $T_{n-1}(x/a)$. Then, it has at least $n-1$ distinct zeros in $[-a, a]$ and, in particular, at least $n-2$ distinct zeros in $[-a, 0) \cup (0, a]$. Hence, the polynomial $R(\sqrt{x})/\sqrt{x} = \sum_{\substack{j=0,\\ j\neq(k-1)/2}}^{n/2-1} c_j x^j$ has at least $n/2-1$ distinct zeros in $(0, \sqrt{a}]$: it is identically zero from Lemma 1.

We have just obtained that

$$P(x) = \frac{1}{\beta_{k,n-1}} T_{n-1}\left(\frac{x}{a}\right) + \frac{P(x)+P(-x)}{2}.$$

Here again, we recall that $|P(x)| < 1/|\beta_{k,n-1}|$ for all $x \in [-a, a]$. Thus, it follows, by substituting 1 and -1 to x

$$-\frac{1}{|\beta_{k,n-1}|} - \frac{1}{\beta_{k,n-1}} T_n\left(\frac{1}{a}\right) < \frac{P(1)+P(-1)}{2} < \frac{1}{|\beta_{k,n-1}|} - \frac{1}{\beta_{k,n-1}} T_{n-1}\left(\frac{1}{a}\right)$$

and

$$-\frac{1}{|\beta_{k,n-1}|} - \frac{1}{\beta_{k,n-1}} T_{n-1}\left(-\frac{1}{a}\right) < \frac{P(-1)+P(1)}{2} < \frac{1}{|\beta_{k,n-1}|} - \frac{1}{\beta_{k,n-1}} T_{n-1}\left(-\frac{1}{a}\right)$$

As n is even, we have $T_{n-1}(1/a) = -T_{n-1}(-1/a) = 1$. Hence, we obtain

$$0 < \frac{P(1)+P(-1)}{2} < 0$$

which is the contradiction desired. □

Appendix 2.

We now prove the following statement.

Lemma 2. *Let $d, n \in \mathbb{N}$, let $x_0, \ldots, x_d, l_0, \ldots, l_d, u_0, \ldots, u_d \in \mathbb{R}$ (resp. $\mathbb{Q}$) such that $x_i \neq x_j$ if $i \neq j$, let $\mathfrak{P}$ the set defined by*

$$\mathfrak{P} = \left\{ (\alpha_0, \ldots, \alpha_n) \in \mathbb{R}^{n+1} \textit{ (resp. } \mathbb{Q}^{n+1}\textit{)} : l_i \leq \sum_{j=0}^{n} \alpha_j x_i^j \leq u_i \textit{ for } i = 0, \ldots, d \right\}.$$

If $d \geq n$, then $\mathfrak{P}$ is a polytope (resp. rational polytope). If $d < n$, then either $\mathfrak{P}$ is empty or $\mathfrak{P}$ is unbounded.

Proof. The set $\mathfrak{P}$ is obviously a polyhedron (resp. rational polyhedron). So, if we want to prove that $\mathfrak{P}$ is a polytope (resp. rational polytope), it suffices to show that it is bounded.

First, we assume $d \geq n$. Then, $\mathfrak{P}$ is contained in the set $\mathfrak{P}'$ defined by

$$\mathfrak{P}' = \left\{ (\alpha_0, \ldots, \alpha_n) \in \mathbb{R}^{n+1} : l_i \leq \sum_{j=0}^{n} \alpha_j x_i^j \leq u_i \text{ for } i = 0, \ldots, n \right\}.$$

Let

$$\begin{array}{rcl} \varphi : \mathbb{R}^{n+1} & \longrightarrow & \mathbb{R}^{n+1} \\ (\alpha_0, \ldots, \alpha_n) & \mapsto & (\sum_{j=0}^{n} \alpha_j x_0^j, \ldots, \sum_{j=0}^{n} \alpha_j x_n^j) \end{array}$$

The linear application φ is an isomorphism for the matrix

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{pmatrix}$$

is a nonsingular Vandermonde matrix since $x_i \neq x_j$ if $i \neq j$. The linear application φ^{-1}, defined on a finite dimensional $\mathbb{R}$-vector space, is continuous and the set $I_n = \prod_{i=0}^{n} [l_i, u_i]$ is a compact of $\mathbb{R}^{n+1}$. Thus, the set $\mathfrak{P}'$ which is equal to $\varphi^{-1}(I_n)$ is a compact of $\mathbb{R}^{n+1}$, which implies that $\mathfrak{P}$ is necessarily bounded.

Now, we assume $d < n$ and $\mathfrak{P}$ non empty. Then, we notice that $\mathfrak{P}$ is defined by the inequalities

$$\begin{array}{ccccc} l_0 - \sum_{j=d+1}^{n} \alpha_j x_0^j & \leq & \sum_{j=0}^{d} \alpha_j x_0^j & \leq & u_0 - \sum_{j=d+1}^{n} \alpha_j x_0^j, \\ \vdots & \leq & \vdots & \leq & \vdots \\ l_d - \sum_{j=d+1}^{n} \alpha_j x_d^j & \leq & \sum_{j=0}^{d} \alpha_j x_d^j & \leq & u_d - \sum_{j=d+1}^{n} \alpha_j x_d^j. \end{array}$$

As $x_i \neq x_j$ if $i \neq j$, the matrix

$$A = \begin{pmatrix} 1 & x_0 & \cdots & x_0^d \\ 1 & x_1 & \cdots & x_1^d \\ \vdots & \vdots & & \vdots \\ 1 & x_d & \cdots & x_d^d \end{pmatrix}$$

is nonsingular. Hence, in particular, the polyhedron $\mathfrak{P}$ contains the family of vectors

$$\left\{ A^{-1} \begin{pmatrix} u_0 - \sum_{j=d+1}^{n} \alpha_j x_0^j \\ u_1 - \sum_{j=d+1}^{n} \alpha_j x_1^j \\ \vdots \\ u_d - \sum_{j=d+1}^{n} \alpha_j x_d^j \end{pmatrix}, (\alpha_{d+1}, \ldots, \alpha_n) \in \mathbb{Z}^{n-d} \right\},$$

which proves that $\mathfrak{P}$ can not be bounded. $\square$

Appendix 3. A worked example

Let us give the details of the first example of Table II. We focus on the degree-3 approximation to the cosine function in $[0, \pi/4]$.

First, we determine (here using the `numapprox` package of Maple with `Digits` equal to 10) the degree-3 minimax polynomial associated to cos on $[0, \pi/4]$. We get

$$p = 0.9998864206 + (0.00469021603 + (-0.5303088665 + 0.06304636099x)x)x.$$

Then, $\epsilon = ||f - p||_{[0,\pi/4]} = 0.0001135879209$. This means that such an approximation is not good enough for single-precision implementation of the cosine function. It can be of interest for some special-purpose implementations. We have

$$\hat{p} = \frac{1}{16}x^3 - \frac{17}{32}x^2 + \frac{5}{1024}x + 1 \text{ and } \hat{\epsilon} = ||f - \hat{p}||_{[0,\pi/4]} = 0.0006939707.$$

Let us choose $K = \hat{\epsilon}/2$. Then, the approach that uses Chebyshev polynomials gives

—3 possible values between 4095/4096 and 4097/4096 for the degree-0 term,

—22 possible values between $-3/512$ and $15/1024$ for the degree-1 term,

—5 possible values between $-9/16$ and $-1/2$ for the degree-2 term,

—1 possible value equal to 1/16 for the degree-3 term.

Hence, the approach that uses Chebyshev polynomials provides 330 candidate polynomials. This is a reasonably small amount : the time necessary for the exhaustive computation of the norms (5) is small[6]. Hence, there is no need to use the approach based on polytopes. We use it anyway in the following to show how it works.

First, we define a subinterval $[A, B]$ of $[0, \pi/4]$ with rational bounds. Let $A = 0$, we choose a rational approximation B of $\pi/4$ such that $B \leq \pi/4$ and its numerator and denominator are small, in order to speed up the computations. The bigger the integers occuring in the data defining the polytope are, the slower the computations are performed. Hence, let $B = 7/9$, which is a convergent of the continued fraction expansion of $\pi/4$.

We choose a value of d equal to 4, i.e. the polytope is built from five rational points which are $0, \frac{1}{4}\cdot\frac{7}{9}, \frac{2}{4}\cdot\frac{7}{9}, \frac{3}{4}\cdot\frac{7}{9}, \frac{7}{9}$. Let $a_j^\star = 2^{m_j} p_j^\star$ for $j = 0, \ldots, 3$. Therefore, we want that the inequalities hereafter be satisfied:

$$\cos\left(\frac{7i}{36}\right) - K \leq \sum_{j=0}^{3} \frac{a_j^\star}{2^{m_j}} \left(\frac{7i}{36}\right)^j \leq \cos\left(\frac{7i}{36}\right) + K \text{ for } i = 0, \ldots, 4. \quad (7)$$

We still assume that $K = \hat{\epsilon}/2$. These inequalities define a polytope but it is not a rational one. Hence, we compute rational approximations of the lower and upper bounds in (7): for $i = 0, \ldots, 4$, we compute l_i and $u_i \in \mathbb{Q}$ such that $l_i \leq \cos\left(\frac{7i}{36}\right) - K$, $\cos\left(\frac{7i}{36}\right) + K \leq u_i$. Here again, our target is to prevent the increase of the size of the involved integers. Therefore, we can either compute convergents of the l_i and u_i or we can impose that, for each i, l_i and u_i have the same denominator as

[6]Around 1.5 second on a 2.53 GHz Pentium 4 computer running Debian Linux with 256 MB RAM.

Table IV. Number of points, and computational times (in C and in Maple) for various choices of d and K in Example 1.

d	K	time [s] T_1	time [s] T_2	# polynomials
9	$7.00\,10^{-4}$	2.60	1.68	7
9	$5.00\,10^{-4}$	1.59	0.98	4
9	$2.50\,10^{-4}$	0.25	0.26	1
9	$6.93\,10^{-4}$	2.55	1.40	6
18	$6.93\,10^{-4}$	2.84	1.40	6
36	$6.93\,10^{-4}$	3.37	1.40	6
72	$6.93\,10^{-4}$	7.39	1.40	6

$p^\star$: $(4095, 6, -34, 1)$

$2.44\,10^{-4}$ (gain 1.5 bits)

the $\frac{a_j^\star}{2^{m_j}}\left(\frac{7i}{36}\right)^j$ for $j = 0, \dots, 3$. We choose the second possibility, hence, for all $i = 0, \dots, 4$, we put

$$l_i = \left\lceil D_i(\cos\left(\frac{7i}{36}\right) - K)\right\rceil / D_i \text{ and } u_i = \left\lfloor D_i(\cos\left(\frac{7i}{36}\right) + K)\right\rfloor / D_i$$

where D_i is the least common multiple of the denominators of the $\frac{a_j^\star}{2^{m_j}}\left(\frac{7i}{36}\right)^j$ for $j = 0, \dots, 3$.

Hence, we obtain a rational polytope defined by $AX \leq B$ with

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 729 & 567 & 1764 & 1372 \\ -729 & -567 & -1764 & -1372 \\ 729 & 1134 & 7056 & 10976 \\ -729 & -1134 & -7056 & -10976 \\ 27 & 63 & 588 & 1372 \\ -27 & -63 & -588 & -1372 \\ 729 & 2268 & 28224 & 87808 \\ -729 & -2268 & -28224 & -87808 \end{pmatrix}, X = \begin{pmatrix} a_0^\star \\ a_1^\star \\ a_2^\star \\ a_3^\star \end{pmatrix}, B = \begin{pmatrix} 4097 \\ -4095 \\ 2930749 \\ -2928678 \\ 2764059 \\ -2761988 \\ 92341 \\ -92266 \\ 2128473 \\ -2126402 \end{pmatrix}.$$

Then, our current C implementation returns only one candidate $(4095, 6, -34, 1)$. We check that it satisfies condition (1). We therefore directly obtain

$$p^\star = \frac{1}{16}x^3 - \frac{17}{32}x^2 + \frac{3}{512}x + \frac{4095}{4096} \quad \text{and} \quad ||f - p^\star||_{[0,\pi/4]} = 0.0002441406250.$$

In this example, the distance between f and $p^\star$ is approximately 0.35 times the distance between f and $\hat{p}$. Using our approach saves around $-\log_2(0.35) \approx 1.5$ bits of accuracy.

Table IV gives some figures (number of points of the polytope, delay of computation) depending on the choices of d and K in this example. Here again, T_1 denotes the time in seconds for producing the candidate polynomials and T_2 denotes the time in seconds for computing the norms (5).