Algorithms Seminar 2006–2007,
F. Chyzak (ed.), INRIA

Available online at the URL
http://algo.inria.fr/seminars/.

Polynomial Approximation and Floating-Point Numbers

Sylvain Chevillard
Arenaire team, Lip, Ens Lyon

June 12, 2007

Summary by Marc Mezzarobba

Abstract

For purposes of numerical implementation of mathematical functions, one is interested in finding low degree polynomials with floating-point coefficients approximating a given function to a certain precision. The basic idea of the method presented here is to look for a constrained approximant interpolating the target function at the same points as the Chebyshev polynomial of given degree; which reduces to solving the shortest vector problem in a lattice. This method is simple, fast, flexible, and although heuristic, it usually gives solutions very close to the actual optimum. This is joint work with Serge Torres, and part of the speaker's PhD research, conducted under the direction of Nicolas Brisebarre and Jean-Michel Muller.

The general framework of this work—and the research subject of the Arenaire project—is the problem of certified numerical computation with mathematical functions. This includes hardware ("FPU"), fixed-precision software ("`libm`") and arbitrary precision software ("bigfloat") implementation of elementary functions as well as correct rounding for more complex operators, such as linear algebra operations. In all these cases, to implement a function f with inexact arithmetic, one may replace it by a simpler function $\tilde{f}$—here *simple* means easier to evaluate—, which in turn is computed using the available primitives, controlling the round-off error.

Here we are concerned with the choice of $\tilde{f}$ in the case where f is a real function we wish to evaluate in a fixed precision floating-point format using additions and multiplications. So we let $f : [a, b] \to \mathbb{R}$ (with $-\infty < a < b < \infty$) be a continuous real function, and we seek to approximate f by a polynomial $\tilde{f}$. (If the domain of the function we're trying to implement is too large, we may split it or use various range reduction technique to obtain a "small enough" segment.) Many usual functions come by definition with natural simple approximations, such as truncated Taylor series for analytic functions. However, they are usually inefficient: for instance, one needs 7 terms of the Taylor expansion of $\exp(1/2 + t)$ to approximate exp on $[-1, 2]$ with (absolute) error not exceeding 0.01, while there exists a polynomial of degree 4 achieving the same accuracy. So it is natural to look for polynomials of minimal degrees providing an approximation of a function to a given precision. We focus on *absolute* error; however, most of what follows adapts to the case of relative error.

Definition 1. Let $f : [a, b] \to \mathbb{R}$ be a continuous function and let $n \in \mathbb{N}$. A polynomial $\tilde{f} \in \mathbb{R}[x]$ of degree $\leq n$ minimizing the maximum absolute error

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

is called a minimax polynomial (or a best approximation polynomial in the sense of Chebyshev) of degree n of f .

Polynomial approximation has been studied from the mathematical point of view since the 19th century. The basic result about minimax approximants is the classical theorem of Chebyshev.

Theorem 1. *A continuous function $f : [a, b] \to \mathbb{R}$ has a unique minimax polynomial. This polynomial can be characterised as the unique $\tilde{f} \in \mathbb{R}[x]$ of degree $\leq n$ such that there exists $\epsilon = \pm 1$ and $n + 2$ points $x_0 < \cdots < x_{n+1}$ satisfying*

$$\forall j, \quad \tilde{f}(x_j) - f(x_j) = \epsilon(-1)^j \|\tilde{f} - f\|_\infty.$$

In words, a polynomial $\tilde{f}$ is the minimax polynomial of degree n of f if and only if the error function $\tilde{f} - f$ reaches is maximal absolute value $n + 2$ times, with alternating signs.

E. Rémès [9] gave an algorithm to compute (arbitrarily good approximations of) the minimax polynomial with quadratic convergence. But this is not the end of the story. Indeed, what we are interested in is approximation by polynomial with floating-point coefficients. Recall a (binary, fixed-precision) floating-point number is a number of the form $m \cdot 2^e$, where the mantissa m is an integer with a fixed number t of bits. Usually the exponent e is also constrained to lie in a certain range, but we will ignore that here. The IEEE 754 standard defines several widely used floating-point formats, including "single precision" ($t = 24$) and "double precision" ($t = 53$). So our true problem is the following.

Problem 1. *Given the function f, a degree bound n and a mantissa size t (and maybe a base b, if we wish to consider floating-point numbers is bases other than 2), find a polynomial $\tilde{f} \in 2^{-\infty}\mathbb{Z}[x]$ of degree $\leq n$, with floating-point coefficients of the specified format, minimizing $\|\tilde{f} - f\|_\infty$.*

Notice that unicity is no more ensured. The naive approach of rounding each coefficient of the real minimax polynomial to obtain a floating-point polynomial may yield poor results.

Example. Take $f : [0, 1] \to \mathbb{R}, x \mapsto \lg(1 + 2^{-x})$ and look for a polynomial approximation of degree ≤ 6 with single-precision coefficients. Then the real minimax corresponds to an error of $8.3 \cdot 10^{-10}$. Rounding each of its coefficients to the nearest single-precision IEEE floating-point number gives an error of $119 \cdot 10^{-10}$, while the optimal polynomial approximation in this format achieves $10.06 \cdot 10^{-10}$.

Similar issues have already been tackled by D. Kodek [7] in the context of signal processing, but his approach is limited to small mantissas (typically $t < 10$) and low degree ($n < 20$) polynomials. W. Kahan also claims to have an efficient method; however his work is not available.

The first general method for determining the optimal floating-point minimax polynomial is due to N. Brisebarre, J.-M. Muller and A. Tisserand [3]. Their idea is the following: when looking for a polynomial of degree n, first guess the exponents $e_0, \ldots, e_n$ of the coefficients and an approximate value ϵ of $\|\tilde{f} - f\|_\infty$. Then let $\tilde{f}(x) = m_0 \cdot 2^{e_0} + \cdots + m_n \cdot 2^{e_n} x^n$ and search for $(m_0, \ldots, m_n) \in \mathbb{Z}^{n+1}$ minimizing $\|\tilde{f} - f\|_\infty$ under the additional constraints

$$-\epsilon \leq \tilde{f}(x_i) - f(x_i) \leq \epsilon \qquad \text{for "many" } x_i \tag{1}$$

using P. Feautrier's integer linear programming tool PIP [4]. Repeat until the guesses are correct.

This method provides certified results and is flexible, in the sense that it allows to find minimax polynomials with various additional constraints. However, it runs in exponential time, and it is quite sensitive to the choice of ϵ and the x_i: if ϵ is underestimated, there will be no solution, but if it is too much overestimated, the search time becomes unacceptable. The new method presented here is a fast (polynomial time) heuristic that aimed to provide good starting points in order to

speed up the method of Brisebarre *et al.* Actually the polynomial it gives turned out to be better than expected, and often good enough to be used directly.

We make the same simplification as above, namely to suppose the values of the exponents have been guessed. Starting from the exponents of the rounded real minimax is usually a good choice. So the setting is the same as above, excepted that we replace the (1) by a "approximate interpolation constraints" for $n+1$ points $x_0, \ldots, x_n$:

$$\begin{bmatrix} \tilde{f}(x_0) \\ \vdots \\ \tilde{f}(x_n) \end{bmatrix} = m_0 \begin{bmatrix} 2^{e_0} \\ \vdots \\ 2^{e_0} \end{bmatrix} + \cdots + m_n \begin{bmatrix} 2^{e_n} x_0^n \\ \vdots \\ 2^{e_n} x_n^n \end{bmatrix} \simeq \begin{bmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{bmatrix}. \tag{2}$$

This "discretisation" step is what makes the rigorous analysis of the method difficult (and the reason we call it heuristic). Let $\mathbf{b}_i = (2^{e_i} x_0^i, \ldots, 2^{e_i} x_n^i)$ and $\mathbf{f} = (f(x_0), \ldots, f(x_n))$. Equation (2) rewrites as

$$m_0 \mathbf{b}_0 + \ldots m_n \mathbf{b}_n \simeq \mathbf{f} \tag{3}$$

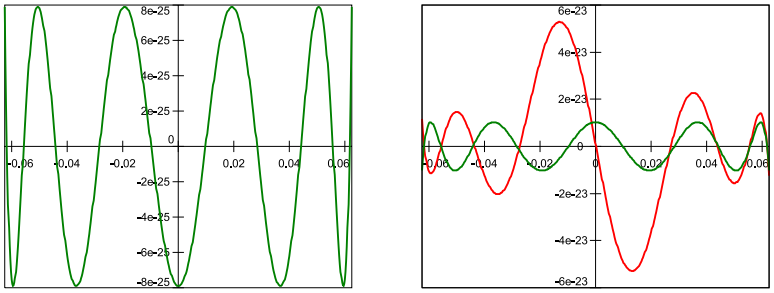
which can be understood as: find the vector closest to $\mathbf{f}$ in the lattice $\mathbb{Z}\mathbf{b}_0 + \ldots \mathbb{Z}\mathbf{b}_n \subset \mathbb{R}^{n+1}$. Let us recall the classical closest lattice vector problem (CVP).

Problem 2. *Let $\Gamma \subset \mathbb{R}^n$ be a lattice (that is, a discrete subgroup) of rank $k \leq n$, and let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. Given a basis $(\mathbf{b}_0, \ldots, \mathbf{b}_k)$ of Γ and a vector $x \in \mathbb{R}^n$, find $y^* \in \Gamma$ minimizing $\|y^* - x\|$.*

The Euclidean CVP is known to be NP-hard to solve exactly [10] or even within a subpolynomial (in n) factor, and *not* NP-hard to approximate to a factor $\sqrt{n/\log n}$ [5], still no algorithm is known that gives a polynomial approximation factor. On the practical side, it can be solved exactly using a (super-exponential) algorithm due to Kannan [6], but building on the celebrated LLL lattice reduction algorithm [8], Babai [1] gave a polynomial algorithm that finds a lattice vector y "pretty close" to a given x, namely such that $\|y - x\|_2 \leq 2^{n/2} \|y^* - x\|_2$. This is the algorithm we use. This exponential approximation factor may seem large, but (as does LLL) Babai's algorithm usually gives better results as one would expect looking at the bound.

The choice of the norm implied by the $\simeq$ sign in Equation (3) is somewhat arbitrary. What we really want is a uniform approximation; however, choosing the Euclidean norm allows us to use Babai's algorithm, and this is what we do. (If necessary, we may refine the result by looking for a closer vector w.r.t. $\|\cdot\|_\infty$ among those we get if we perturb the y we found by a linear combination with small coefficients of the "pretty short" vectors of the LLL-reduced basis of Γ.)

So the algorithm is as follows. We are given the function f, a degree n and a floating-point precision t, and we are looking for a floating-point polynomial $\tilde{f}$ (with this degree and this precision) close to f. We first compute (an approximation of) the real minimax $f^* = a_0^* + \cdots + a_n^* x^n$ by Rémès' algorithm. This gives us a first guess $e_i = 1 - t + |\lg |a_i^*||$ for the values of the exponents of $\tilde{f}$. The critical step is to choose the interpolation points x_i in order to avoid Runge's phenomenon. According to the intuition that $\tilde{f}$ will be close to f^*, and in view of Theorem 1, we use the $n+1$ points x such that $f^*(x) = f(x)$. We finally solve the closest vector problem 3 by Babai's algorithm as explained above. Of course, the guessed exponents may not be optimal. So we use the following heuristic: if one of the computed coefficients a_i has a mantissa m_i that doesn't fit on t bits (which suggests our choice of exponents was wrong), we repeat the process after shifting e_i by $|\lg m_i|$ (that is, replacing $a_i^a st$ by a_i in the formula used to guess the exponents) to take into account the order of magnitude of a_i. There is no proof that this converges; but it did after only two or three iterations on all the examples the speaker tried, even when the initial exponents were far off.



(a) Real minimax (b) First try

In practice, the running time of the algorithm is essentially that of Rémès' algorithm (which is called only once even if one wishes, say, to try several floating-point formats).

Example. Finally let us look at an example that shows how the above method can be tweaked to solve more complex cases with additional constraints. John Harrison (Intel) asked the speaker's team for a degree 9 polynomial approximating $f(x) = (2^x - 1)/x$ for $x \in [-1/16, 1/16]$ to a precision $\epsilon \lesssim 2^{-74} \simeq 5.30 \cdot 10^{-23}$. The coefficients were to be extended double precision ($t = 64$), except for the constant one, which was to be searched for as the sum of two "extended doubles".

Rémès' algorithm gives $\|f_8^* - f\|_\infty < 40.1 \cdot 10^{-23}$ and $\|f_9^* - f\|_\infty < 0.08 \cdot 10^{-23}$ for the minimax polynomials of degree 8 and 9 respectively, so degree 9 should indeed be a good choice for the target precision. It also provides interpolation points (Figure (a)). Applying the strategy above unchanged, we get $\tilde{f}$ s.t. $\|\tilde{f} - f\|_\infty < 5.32 \cdot 10^{-23}$: so the challenge was well chosen! (In comparison, rounding f_9^* gives $\|f_{\text{rnd}}^* - f\|_\infty \simeq 40.35 \cdot 10^{-23}$.) But look at Figure (b): $\tilde{f}$ does not respect the interpolation constraint! Indeed, we observe that the slope of $\tilde{f}$ in 0 is very constrained – a good polynomial $\tilde{f}$ should satisfy $\tilde{a}_1 \simeq (\lg 2)^2/2$. To take this into account, we look for $\tilde{g} = b_0 + b_2 x^2 + \ldots b_9 x^9$ approximating $g(x) = f(x) - a_1 x$. Adapting our strategy to enforce $g_1 = 0$, we finally get $\|\tilde{g} - g\|_\infty < 4.45 \cdot 10^{-23}$.

Although it worked on the last example above, Rémès' algorithm is not sufficient in general to compute real constrained minimax polynomials (such as g^* here), so a better algorithm is needed to generalize their use. Work is also in progress to extend the method exposed here to other types of approximants, such as rational polynomials and sums of cosines.

References

[1] Babai (László). – On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, vol. 6, n° 1, 1986, pp. 1–14.

[2] Brisebarre (Nicolas) and Chevillard (Sylvain). – Efficient polynomial L^∞-approximations. – Inria research report, 2006.

[3] Brisebarre (Nicolas), Muller (Jean-Michel), and Tisserand (Arnaud). – Computing machine-efficient polynomial approximations. *ACM Trans. Math. Softw*, vol. 32, n° 2, 2006.

[4] Feautrier (Paul). – Parametric integer programming. *Operations Research*, vol. 22, n° 3, 1988, pp. 243–268.

[5] Goldreich and Goldwasser. – On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, vol. 60, 2000.

[6] Kannan (Ravi). – *Minkowski's convex body theorem and integer programming.* – Technical Report n° CMU-CS-86-105, Computer Science Dept., Carnegie-Mellon University, 1986.

[7] Kodek (Dušan M.). – An approximation error lower bound for integer polynomial minimax approximation. *Elektroteh. vestn.*, vol. 69, n° 5, 2002, pp. 266–272.

[8] Lenstra (Arjen K.), Lenstra, Jr. (Hendrik W.), and Lovász (László). – Factoring polynomials with rational coefficients. *Mathematische Annalen*, vol. 261, 1982, pp. 515–534.

[9] Rémès (Evgeny). – Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation. *Comptes Rendus de l'Académie des Sciences*, vol. 198, 1934, pp. 2063–2065.

[10] van Emde Boas (Peter). – *Another NP-complete Partition Problem and the Complexity of Computing Short Vectors in Lattices.* – Technical Report n° 81-04, Mathematics Department, University of Amsterdam, 1981.

I wish to thank Sylvain Chevillard for providing very useful comments on a draft of this summary and allowing me to use the graphs (a) and (b) from his presentation.