LLL and polynomial approximation

Nicolas Brisebarre, Sylvain Chevillard

LLL and polynomial approximation

Nicolas Brisebarre **Sylvain Chevillard**

Arénaire project — CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE — ECOLE NORMALE SUPERIEURE DE LYON — INRIA — UCBL

The problem

In practice, to implement mathematical functions, one uses a polynomial approximation to the function. The coefficients of the polynomial will be stored into floating-point variables.

A naive method is to compute the best polynomial with real coefficients p^* and to round each coefficient to the nearest floating-point number, thus leading to a polynomial $\widehat{p}$. The polynomial $\widehat{p}$ may be quite inaccurate. On figure 1 in the *Example* box, the graph of the absolute errors $\widehat{p}-f$ and p^*-f in a real case are shown. The method described in this poster gives much better results (figure 2).

We address here the problem of finding a very good polynomial $\bar{p}$ with floating-point coefficients, to approximate a continuous function on an interval.

Lattices are a *key tool* in the method we propose.

IEEE-754 standard

The standard IEEE-754 defines how computers manipulate real numbers. The real numbers are approximated by the so-called *floating-point* numbers. Let us fix $t \in \mathbb{N}$.

The set of floating-point numbers of precision t is defined by $\mathcal{F}_t = \{m \cdot 2^{e-t}, (m,\, e) \in \mathbb{Z}^2\}$ m being written with exactly t bits.

In other words $x \in \mathcal{F}_t$ is a number of the form $x = 0.1m_2m_3 \ldots m_t \cdot 2^e$ where $m_i \in \{0,\, 1\}$.

Polynomial approximation

To evaluate a mathematical function f on several points of an interval, one generally replaces it by an approximation easier to evaluate.

A good choice consists in choosing a polynomial as an approximant because only additions and multiplications are required to evaluate it.

Using $p(x)$ as an approximation to $f(x)$ leads to an absolute error $\varepsilon(x) = p(x) - f(x)$ (one may also consider the relative error defined as $\varepsilon(x)/f(x)$).

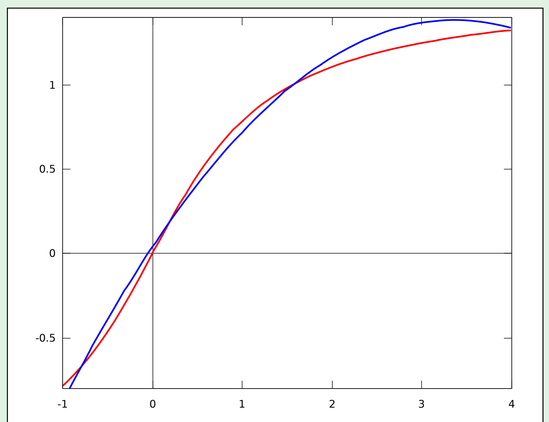

Fig. 1: $\arctan(x)$ and a polynomial $p(x)$ of degree 2 on $[-1,\, 4]$

The worst approximation quality corresponds to the maximum of the error function $|\varepsilon|$ on the interval $[a,\, b]$. This is called the *infinite norm* of ε on $[a,\, b]$. The figures show the example of a polynomial of degree 2 approximating the function $\arctan$ in $[-1,\, 4]$.

The *best approximation* problem consists in finding the polynomial with real coefficients leading to the smallest error in infinite norm. The existence of this polynomial and its uniqueness are ensured by theorems by Chebyshev and De la Vallée Poussin. Remez gave an iterative process that converges toward this polynomial.

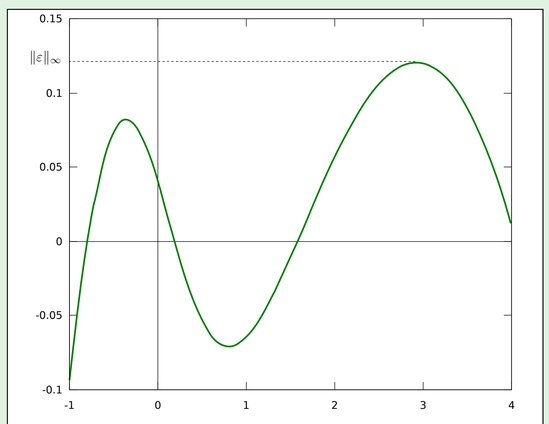

Fig. 2: the graph of the absolute error $\varepsilon(x) = p(x) - \arctan(x)$

Formalization

Our goal: find p with floating-point coefficients that approximates f well.

Thus, each coefficient a of p must be chosen of the form $a = m \cdot 2^e$ where m and e are two unknown integers.

e is the exponent of the coefficient, thus corresponds to its order of magnitude. We assume that it is the same as the corresponding one in p^*.

We obtain now a problem that we will solve with lattice reduction techniques: find p that approximates f well and with the following form: $m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} X + \cdots + m_n \cdot 2^{e_n} X^n$.

Resolution

Our goal: find p that approximates f well and with the following form: $m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} X + \cdots + m_n \cdot 2^{e_n} X^n$.

We use the idea of interpolation: we force the polynomial to be close to the function at some chosen points. Let $(x_0,\, \cdots,\, x_n)$ be $n+1$ points in $[a,\, b]$. We search $m_0,\, \cdots,\, m_n$ such that for all i

$$p(x_i) = m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} x_i + \cdots + m_n \cdot 2^{e_n} x_i^n \simeq f(x_i).$$

Rewritten with vectors:
$$\underbrace{m_0 \begin{pmatrix} 2^{e_0} \\ 2^{e_0} \\ \vdots \\ 2^{e_0} \end{pmatrix} + \cdots + m_n \begin{pmatrix} 2^{e_n} \cdot x_0^n \\ 2^{e_n} \cdot x_1^n \\ \vdots \\ 2^{e_n} \cdot x_n^n \end{pmatrix}}_{\Gamma \text{ of the form } \mathbb{Z}\overrightarrow{b_0} + \mathbb{Z}\overrightarrow{b_1} + \cdots + \mathbb{Z}\overrightarrow{b_n}} \simeq \underbrace{\begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}}_{\overrightarrow{t} \in \mathbb{R}^{n+1}} .$$

We recognize a closest vector problem!

Approximately solving CVP

The *closest vector problem (CVP)* is the following:
given a basis $(\overrightarrow{b_1},\, \cdots,\, \overrightarrow{b_n})$ of a lattice $\mathcal{B}$ and a vector $\overrightarrow{t}$, let $\mu = \mathrm{dist}(\overrightarrow{v},\, \mathcal{B})$. The problem is to find a vector $\overrightarrow{v} \in \mathcal{B}$ such that $\|\overrightarrow{v} - \overrightarrow{t}\| = \mu$.

The associated α-approximation problem consists in finding a vector $\overrightarrow{v}$ in the lattice such that $\|\overrightarrow{v} - \overrightarrow{t}\| \leq \alpha\mu$.

Babai's nearest plane algorithm uses an LLL-reduced basis of the lattice to solve the $2^{n/2}$ -approximation of CVP:

```
Input:
   an LLL-reduced basis (b1,...,bn)
   a vector t
Output:
   a 2^(n/2)-approximation of CVP

   (c1,...,cn) = GramSchmidt(b1,...,bn);
   v = t;
   for(j=n ; j > 0 ; j--) {
     v = v - [(v, cj)/(cj, cj)] * bj ;
   }
   return t-v;
```

In this pseudo-code,
[x] denotes the nearest integer to x;
(v, w) is the dot product of v and w.

Example

In this example, we try to approximate $f : \log_2(1 + 2^{-x})$ on $[0,\, 1]$ by a polynomial p of degree 6 with single precision coefficients (i.e. 24 bits of mantissa).

p^* denotes the best polynomial with real coefficients;
$\widehat{p}$ denotes the polynomial obtained by rounding each coefficient of p^* to the nearest floating-point number;
$\bar{p}$ denotes the polynomial that we obtain with our method.

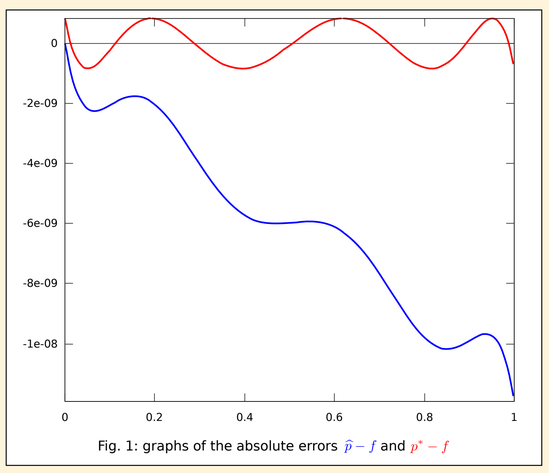

Fig. 1: graphs of the absolute errors $\widehat{p} - f$ and $p^* - f$

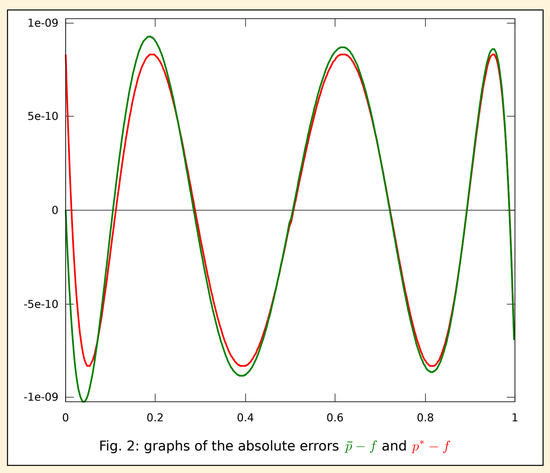

Fig. 2: graphs of the absolute errors $\bar{p} - f$ and $p^* - f$

As can be seen, $\bar{p}$ is far closer to f than $\widehat{p}$. Especially, $\bar{p}$ is almost as accurate as the best polynomial p^*.

For the $n+1=7$ points required by the method, we chose those where the red curve crosses the x-axis. A general theorem due to Chebyshev ensures that there are always at least $n+1$ points where the best polynomial with real coefficients p^* equals the function, i.e. where $p^* - f$ vanishes.