Some tools for evaluating functions on a machine in a certified (yet efficient!) way

Nicolas Brisebarre

Journées nationales du GDR IM 2015, 2/2/2015


A joint work!

AriC team, LIP

Nicolas Brisebarre, Sylvain Chevillard, Guillaume Hanrot, Mioara Joldeș, Jean-Michel Muller, Arnaud Tisserand, Serge Torres

Computing

- Huge focus on “High Performance Computing”

Computing

- Huge focus on “High Performance Computing”
- Security issues regarding the data, the computations, the results: Cryptology

Computing

- Huge focus on "High Performance Computing"
- Security issues regarding the data, the computations, the results: Cryptology
- What about the accuracy and the reliability of these computations?

Floating Point (FP) Arithmetic

Given

$$\begin{cases} \text{a radix} & \beta \geq 2, \\ \text{a precision} & n \geq 1, \\ \text{a set of exponents} & E_{\mathsf{min}} \cdots E_{\mathsf{max}}. \end{cases}$$

A finite FP number x is represented by 2 integers:

- integer mantissa M, $\beta^{n-1} \leq |M| \leq \beta^n - 1$,
- exponent E, $E_{\mathsf{min}} \leq E \leq E_{\mathsf{max}}$

such that

$$x = \frac{M}{\beta^{n-1}} \times \beta^E.$$

We assume binary FP arithmetic (that is to say, $\beta = 2$).

IEEE Precisions

See `http://en.wikipedia.org/wiki/IEEE_floating_point` or (older)
`http://babbage.cs.qc.edu/courses/cs341/IEEE-754references.html`.

	precision n	min. exponent E_{min}	maximal exponent E_{max}
single (binary 32)	24	-126	127
double (binary 64)	53	-1022	1023
extended double	64	-16382	16383
quadruple (binary 128)	113	-16382	16383

We have $x = \frac{M}{2^{n-1}} \times 2^E$ with $2^{n-1} \leq |M| \leq 2^n - 1$

and $E_{\mathsf{min}} \leq E \leq E_{\mathsf{max}}$.

Various bugs

S. Rump's example (1988). Consider

$$f(a,b) = 333.75b^6 + a^2\left(11a^2b^2 - b^6 - 121b^4 - 2\right) + 5.5b^8 + \frac{a}{2b},$$

and try to compute $f(a,b)$ for $a = 77617.0$ and $b = 33096.0$. On an IBM 370 computer:

- 1.172603 in single precision;
- 1.1726039400531 in double precision;
- 1.172603940053178 in extended precision.

Various bugs

S. Rump's example (1988). Consider

$$f(a,b) = 333.75b^6 + a^2\left(11a^2b^2 - b^6 - 121b^4 - 2\right) + 5.5b^8 + \frac{a}{2b},$$

and try to compute $f(a,b)$ for $a = 77617.0$ and $b = 33096.0$. On an IBM 370 computer:

- 1.172603 in single precision;
- 1.1726039400531 in double precision;
- 1.172603940053178 in extended precision.

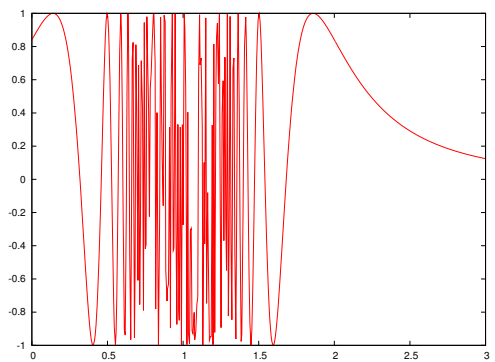
And yet, the exact result is $-0.8273960599\cdots$.
What about more recent systems? On a Pentium4 (gcc, Linux), Rump's C program returns

- 5.960604×10^{20} in single precision;
- 2.0317×10^{29} in double precision;
- $-9.38724 \times 10^{-323}$ in extended precision.

Various bugs

M. Joldeș. Rigorous Polynomial Approximations and Applications. PhD thesis, ENS Lyon, 2011.

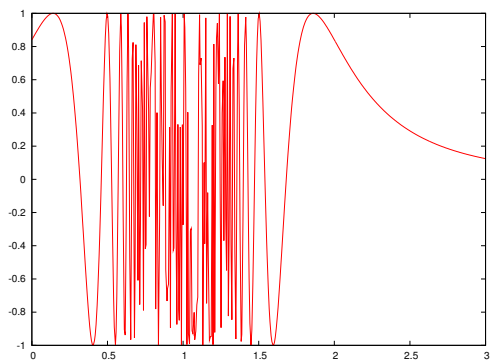
Let $J = \int_0^3 \sin\left(\frac{1}{(10^{-3} + (1-x)^2)^{3/2}}\right) \mathrm{d}x$.



Various bugs

M. Joldeș. Rigorous Polynomial Approximations and Applications. PhD thesis, ENS Lyon, 2011.

Let $J = \int_0^3 \sin\left(\frac{1}{(10^{-3} + (1-x)^2)^{3/2}}\right) \mathrm{d}x$.



- Maple18: 0.7499743685;
- Pari/GP: $0.79277309714790807 55$;
- Mathematica and Chebfun fail to answer;
- Chen, '06: 0.7578918118.

  WHAT IS THE CORRECT ANSWER?

Various bugs

W. Tucker. Validated Numerics. Princeton University Press, 2011.

Let $I = \int_0^8 \sin(x + e^x)\mathrm{d}x$. Let's evaluate it using MATLAB.

```
fcn_str = 'sin(x+exp(x))';
f = vectorize(inline(fcn_str));
a = 0; b = 8;
>> q = quad(f,a,b)
q =
    0.251102722027180
```

Actually, $I \in [0.3474, 0.3475]$...

A statement by Alston Householder

"It makes me nervous to fly on airplanes since I know they are designed using floating-point arithmetic." A. Householder

A statement by Alston Householder

"It makes me nervous to fly on airplanes since I know they are designed using floating-point arithmetic." A. Householder

Well, the situation is not that tragic! There are some useful computations that we can perform in a (sometimes fast and) certified way.

A statement by Alston Householder

"It makes me nervous to fly on airplanes since I know they are designed using floating-point arithmetic." A. Householder

Well, the situation is not that tragic! There are some useful computations that we can perform in a (sometimes fast and) certified way.

Some of these results can even be used to establish rigorous mathematical proofs (Tucker's proof of the existence of Lorentz attractor, Hales proof of Kepler conjecture, Helfgott proof of the ternary Goldbach conjecture).

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{\ }, \ldots$

Goal: evaluation of φ to a given accuracy η.

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f.

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f.
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f.
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.
- Step 4. Computation of a certified evalutation error of $p^\star$: GAPPA (G. Melquiond).

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 0. Computation of hardest-to-round cases: V. Lefèvre and J.-M. Muller.
- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f.
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.
- Step 4. Computation of a certified evalutation error of $p^\star$: GAPPA (G. Melquiond).

Applications

- Specific hardware implementations in low precision (~ 15 bits). Reduce the cost (time and silicon area) while keeping a correct accuracy;

- single or double IEEE precision software implementations. Get very high accuracy while keeping the (time and memory) cost acceptable.

Scientific Framework and Tools

Computer Arithmetic

Scientific Framework and Tools

Computer Arithmetic

- Numerical Analysis, Approximation Theory, Interval Analysis, Fine-tuned Implementation

Scientific Framework and Tools

Computer Arithmetic

- Numerical Analysis, Approximation Theory, Interval Analysis, Fine-tuned Implementation
- Algorithmic Number Theory, Computer Algebra, Functional Analysis, Complex Analysis, Logic, Formal Proof

Two Key Problems

- Quantization problem. When it comes to implementing a function or a filter, need for an (quasi-)optimal polynomial approximation or transfer function. Issue: the coefficients are machine numbers.

Two Key Problems

- Quantization problem. When it comes to implementing a function or a filter, need for an (quasi-)optimal polynomial approximation or transfer function. Issue: the coefficients are machine numbers.
- Rigorous approximation. Given f a solution of a linear ODE, compute a pair (P, Δ) where P is a polynomial and Δ an interval such that $f - P$ takes all its values in Δ.

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 0. Computation of hardest-to-round cases: V. Lefèvre and J.-M. Muller.
- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f.
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.
- Step 4. Computation of a certified evalutation error of $p^\star$: GAPPA (G. Melquiond).

Minimax Approximation

Reminder. Let $g : [a, b] \to \mathbb{R}$, $||g||_{[a,b]} = \sup_{a \le x \le b} |g(x)|$.

We denote $\mathbb{R}_n[X] = \{p \in \mathbb{R}[X]; \deg p \le n\}$.

Minimax Approximation

Reminder. Let $g : [a, b] \to \mathbb{R}$, $||g||_{[a,b]} = \sup_{a \leq x \leq b} |g(x)|$.

We denote $\mathbb{R}_n[X] = \{p \in \mathbb{R}[X]; \deg p \leq n\}$.

Minimax approximation: let $f : [a, b] \to \mathbb{R}$, $n \in \mathbb{N}$, we search for $p \in \mathbb{R}_n[X]$ s.t.

$$||p - f||_{[a,b]} = \inf_{q \in \mathbb{R}_n[X]} ||q - f||_{[a,b]}.$$

Minimax Approximation

Reminder. Let $g : [a, b] \to \mathbb{R}$, $||g||_{[a,b]} = \sup_{a \le x \le b} |g(x)|$.

We denote $\mathbb{R}_n[X] = \{p \in \mathbb{R}[X]; \deg p \le n\}$.

Minimax approximation: let $f : [a, b] \to \mathbb{R}$, $n \in \mathbb{N}$, we search for $p \in \mathbb{R}_n[X]$ s.t.

$$||p - f||_{[a,b]} = \inf_{q \in \mathbb{R}_n[X]} ||q - f||_{[a,b]}.$$

An algorithm due to Remez gives p (`minimax` function in Maple, also available in Sollya `http://sollya.gforge.inria.fr/`).

Minimax Approximation

Reminder. Let $g : [a, b] \to \mathbb{R}$, $||g||_{[a,b]} = \sup_{a \leq x \leq b} |g(x)|$.

We denote $\mathbb{R}_n[X] = \{p \in \mathbb{R}[X]; \deg p \leq n\}$.

Minimax approximation: let $f : [a, b] \to \mathbb{R}$, $n \in \mathbb{N}$, we search for $p \in \mathbb{R}_n[X]$ s.t.

$$||p - f||_{[a,b]} = \inf_{q \in \mathbb{R}_n[X]} ||q - f||_{[a,b]}.$$

An algorithm due to Remez gives p (`minimax` function in Maple, also available in Sollya `http://sollya.gforge.inria.fr/`).

Problem: we can't directly use minimax approx. in a computer since the coefficients of p can't be represented on a finite number of bits.

Approximation of the Function $\cos$ over $[0, \pi/4]$ by a Degree-3 Polynomial

Maple or Sollya tell us that the polynomial

$$p = 0.9998864206 + 0.00469021603x - 0.5303088665x^2 + 0.06304636099x^3$$

is $\sim$ the best approximant to $\cos$. We have
$\varepsilon = ||\cos - p||_{[0,\pi/4]} = 0.0001135879...$.

We look for $a_0, a_1, a_2, a_3 \in \mathbb{Z}$ such that

$$\max_{0 \le x \le \pi/4} \left| \cos x - \left(\frac{a_0}{2^{12}} + \frac{a_1}{2^{10}} x + \frac{a_2}{2^6} x^2 + \frac{a_3}{2^4} x^3 \right) \right|$$

is minimal.

Approximation of the Function cos over $[0, \pi/4]$ by a Degree-3 Polynomial

Maple or Sollya tell us that the polynomial

$$p = 0.9998864206 + 0.00469021603x - 0.5303088665x^2 + 0.06304636099x^3$$

is $\sim$ the best approximant to cos. We have
$\varepsilon = ||\cos - p||_{[0,\pi/4]} = 0.0001135879...$.

We look for $a_0, a_1, a_2, a_3 \in \mathbb{Z}$ such that

$$\max_{0 \le x \le \pi/4} \left| \cos x - \left(\frac{a_0}{2^{12}} + \frac{a_1}{2^{10}} x + \frac{a_2}{2^6} x^2 + \frac{a_3}{2^4} x^3 \right) \right|$$

is minimal.

The naive approach gives the polynomial

$$\hat{p} = \frac{2^{12}}{2^{12}} + \frac{5}{2^{10}} x - \frac{34}{2^6} x^2 + \frac{1}{2^4} x^3.$$

We have $\hat{\varepsilon} = ||\cos - \hat{p}||_{[0,\pi/4]} = 0.00069397...$.

Approximation of the Function $\cos$ over $[0, \pi/4]$ by a Degree-3 Polynomial

Maple or Sollya computes a polynomial p which is $\sim$ the best approximant to $\cos$. We have $\varepsilon = ||\cos - p||_{[0,\pi/4]} = 0.0001135879...$.
We look for $a_0, a_1, a_2, a_3 \in \mathbb{Z}$ such that

$$\max_{0 \le x \le \pi/4} \left| \cos x - \left(\frac{a_0}{2^{12}} + \frac{a_1}{2^{10}} x + \frac{a_2}{2^6} x^2 + \frac{a_3}{2^4} x^3 \right) \right|$$

is minimal.
The naive approach gives the polynomial $\hat{p}$ and
$\hat{\varepsilon} = ||\cos - \hat{p}||_{[0,\pi/4]} = 0.00069397...$

Approximation of the Function cos over $[0, \pi/4]$ by a Degree-3 Polynomial

Maple or Sollya computes a polynomial p which is $\sim$ the best approximant to cos. We have $\varepsilon = ||\cos - p||_{[0,\pi/4]} = 0.0001135879...$.
We look for $a_0, a_1, a_2, a_3 \in \mathbb{Z}$ such that

$$\max_{0 \le x \le \pi/4} \left| \cos x - \left(\frac{a_0}{2^{12}} + \frac{a_1}{2^{10}} x + \frac{a_2}{2^6} x^2 + \frac{a_3}{2^4} x^3 \right) \right|$$

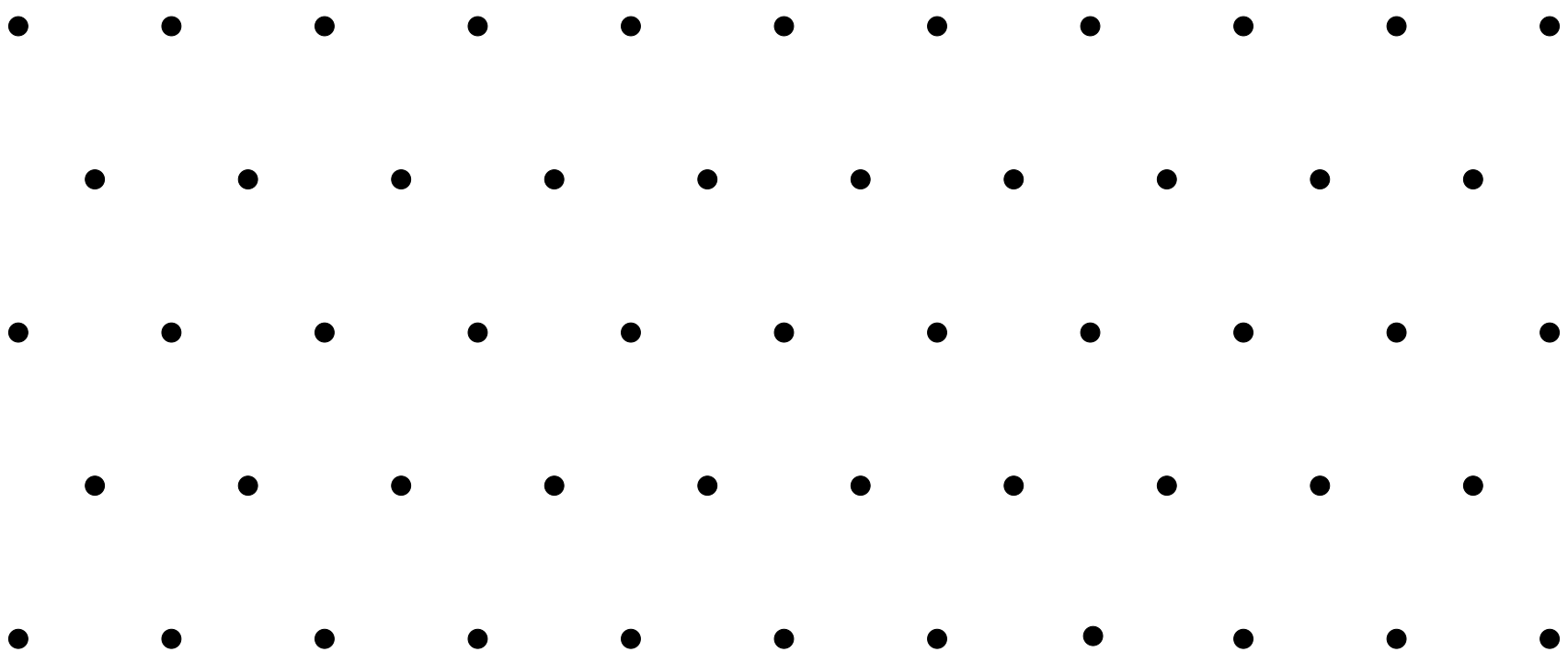
is minimal.
The naive approach gives the polynomial $\hat{p}$ and $\hat{\varepsilon} = ||\cos - \hat{p}||_{[0,\pi/4]} = 0.00069397...$ But the best "truncated" approximant:

$$p^\star = \frac{4095}{2^{12}} + \frac{6}{2^{10}} x - \frac{34}{2^6} x^2 + \frac{1}{2^4} x^3$$

which gives $||\cos - p^\star||_{[0,\pi/4]} = 0.0002441406250$.

In this example, we gain $-\log_2(0.35) \approx 1.5$ bits of accuracy.

An Approach based on Lattice Basis Reduction



An Approach based on Lattice Basis Reduction

Definition

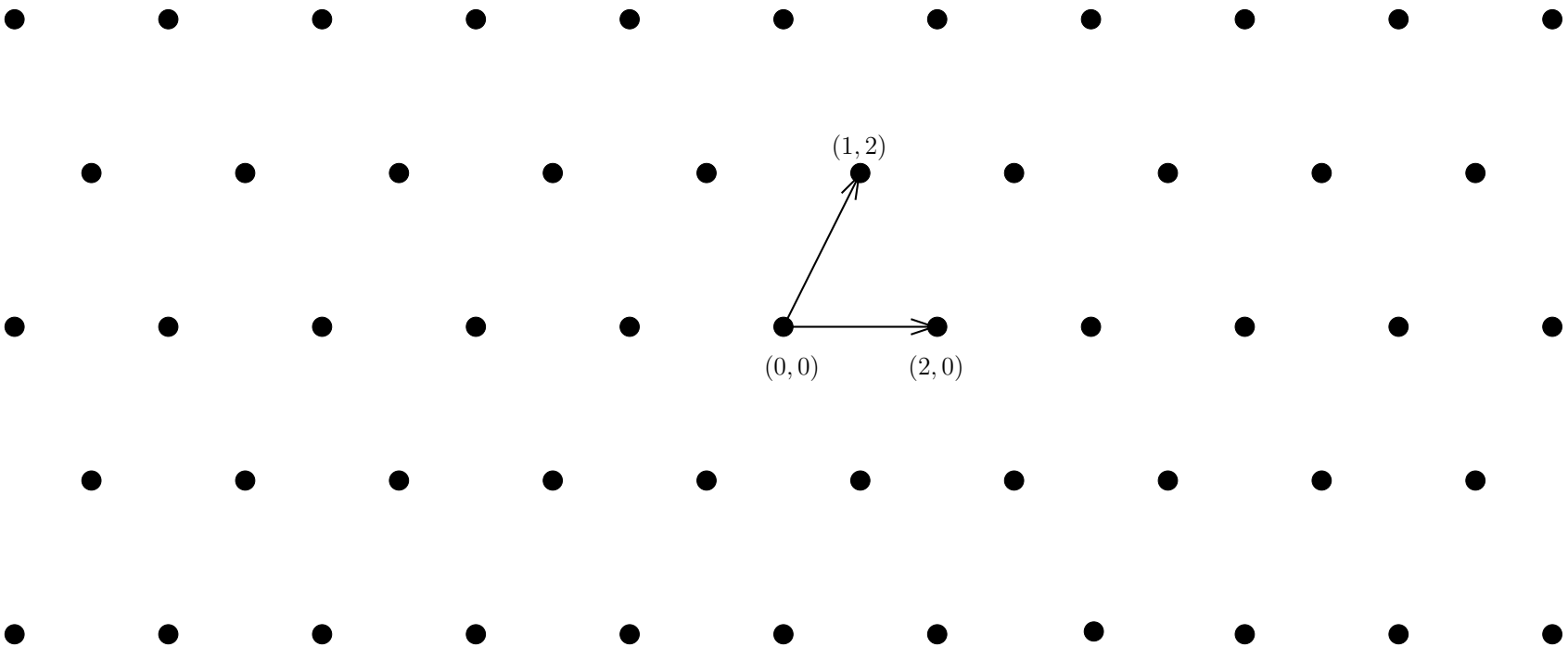
Let L be a nonempty subset of $\mathbb{R}^d$, L is a lattice iff there exists a set of vectors $b_1, \ldots, b_k$ $\mathbb{R}$-linearly independent such that

$$L = \mathbb{Z}.b_1 \oplus \cdots \oplus \mathbb{Z}.b_k.$$

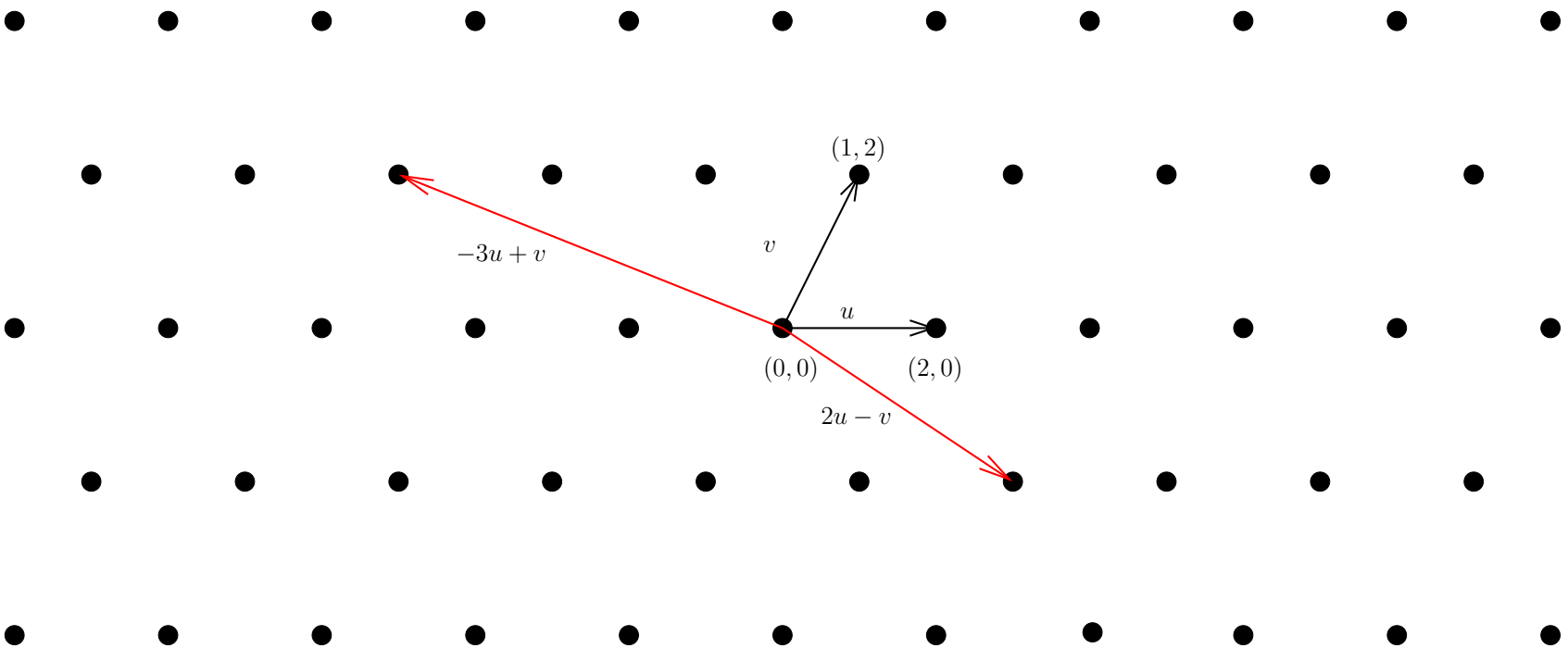
$(b_1, \ldots, b_k)$ is a basis of the lattice L.

Examples. $\mathbb{Z}^d$, every subgroup of $\mathbb{Z}^d$.

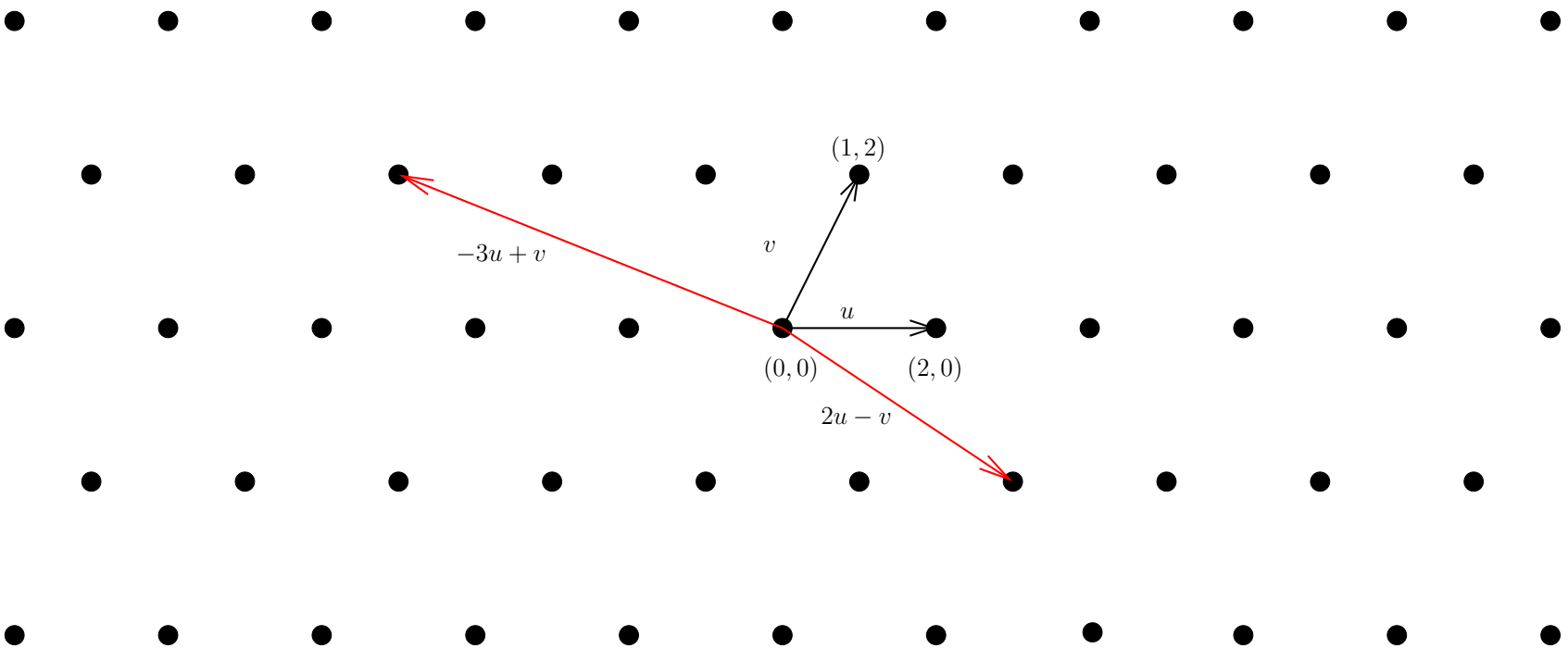
Example: The Lattice $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$



Example: The Lattice $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$

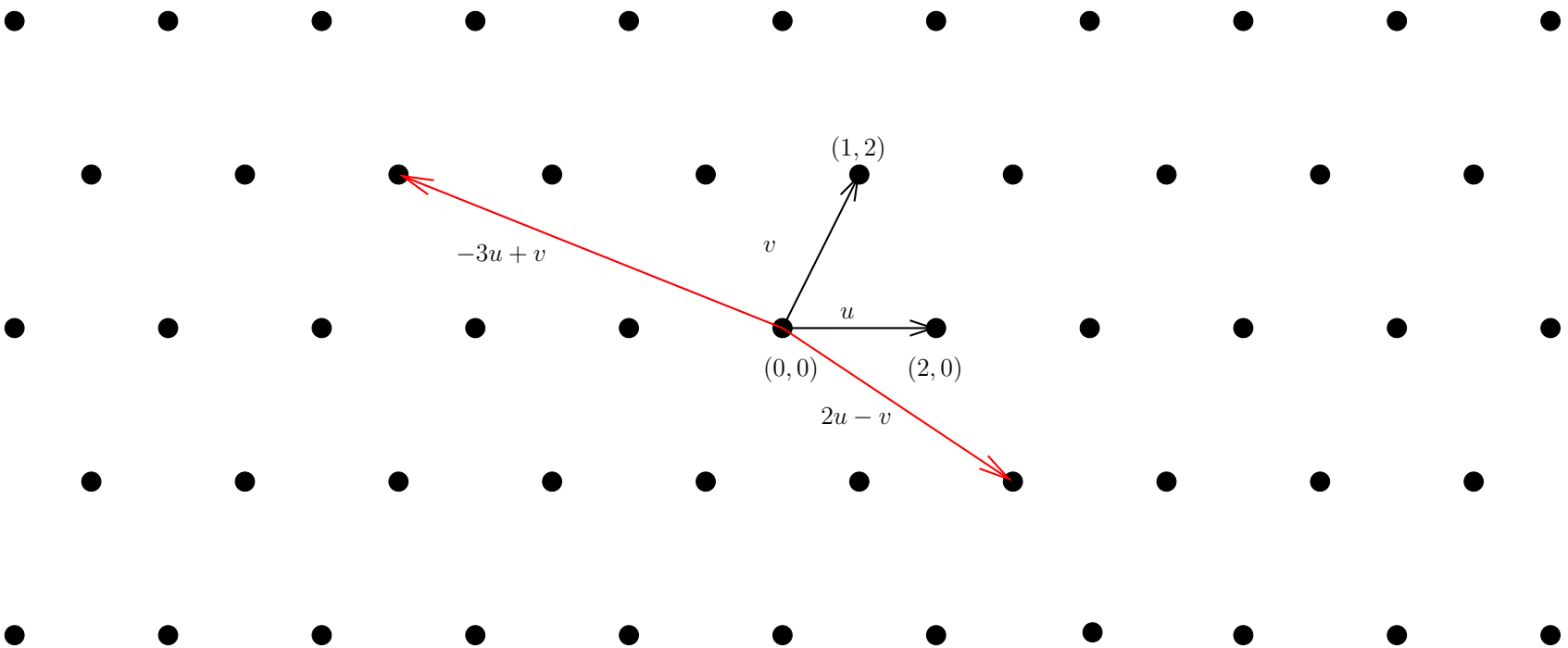


Example: The Lattice $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$



SVP (Shortest Vector Problem) and CVP (Closest Vector Problem)

Example: The Lattice $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$



SVP (Shortest Vector Problem) and CVP (Closest Vector Problem) are NP-hard.

Lenstra-Lenstra-Lovász Algorithm

SVP (Shortest Vector Problem) and CVP (Closest Vector Problem) are NP-hard.
Factoring Polynomials with Rational Coefficients, A. K. Lenstra, H. W. Lenstra and L. Lovász, Math. Annalen **261**, 515-534, 1982.

The LLL algorithm gives an approximate solution to SVP in polynomial time.

Babai's algorithm (based on LLL) gives an approximate solution to CVP in polynomial time.

Absolute Error Problem

We search for (one of the) best(s) polynomial of the form

$$p^\star = \frac{a_0^\star}{2^{m_0}} + \frac{a_1^\star}{2^{m_1}} X + \cdots + \frac{a_n^\star}{2^{m_n}} X^n$$

(where $a_i^\star \in \mathbb{Z}$ and $m_i \in \mathbb{Z}$) that minimizes $\|f - p\|_{[a,\, b]}$.

Discretize the continuous problem: we choose $x_1, \cdots, x_d$ points in $[a, b]$ such that $\frac{a_0^\star}{2^{m_0}} + \frac{a_1^\star}{2^{m_1}} x_i + \cdots + \frac{a_n^\star}{2^{m_n}} x_i^n$ is as close as possible to $f(x_i)$ for all $i = 1, \ldots, d$.

Absolute Error Problem

We search for (one of the) best(s) polynomial of the form

$$p^\star = \frac{a_0^\star}{2^{m_0}} + \frac{a_1^\star}{2^{m_1}} X + \cdots + \frac{a_n^\star}{2^{m_n}} X^n$$

(where $a_i^\star \in \mathbb{Z}$ and $m_i \in \mathbb{Z}$) that minimizes $\|f - p\|_{[a,\, b]}$.

Discretize the continuous problem: we choose $x_1, \cdots, x_d$ points in $[a, b]$ such that $\frac{a_0^\star}{2^{m_0}} + \frac{a_1^\star}{2^{m_1}} x_i + \cdots + \frac{a_n^\star}{2^{m_n}} x_i^n$ is as close as possible to $f(x_i)$ for all $i = 1, \ldots, d$.

Actually, this can be viewed as an instance of the Closest Vector Problem.

An Example from CRlibm

- CRlibm is a library designed to compute correctly rounded functions in an efficient way (target : IEEE double precision).

  `http://lipforge.ens-lyon.fr/www/crlibm/`

- It uses specific formats such as double-double or triple-double.

- Here is an example we worked on with C. Lauter, and which is used to compute $\arcsin(x)$ on $[0.79; 1]$.

Arcsine Function

After argument reduction we have the problem to approximate

$$g(z) = \frac{\arcsin(1 - (z + m)) - \frac{\pi}{2}}{\sqrt{2 \cdot (z + m)}}$$

where $0\text{xBFBC28F800009107} \leq z \leq 0\text{x3FBC28F7FFFF6EF1}$ (i.e. approximately $-0.110 \leq z \leq 0.110$) and
$m = 0\text{x3FBC28F80000910F} \simeq 0.110$.

Data

Target accuracy to achieve correct rounding : 2^{-119}.
The minimax of degree 21 is sufficient (error $= 2^{-119.83}$).
Each approximant is of the form

$$\underbrace{p_0}_{t.d.} + \underbrace{p_1}_{t.d.} x + \underbrace{p_2}_{d.d.} x^2 + \underbrace{\cdots}_{\cdots} + \underbrace{p_9}_{d.d.} x^9 + \underbrace{p_{10}}_{d.} x^{10} + \underbrace{\cdots}_{\cdots} + \underbrace{p_{21}}_{d.} x^{21}$$

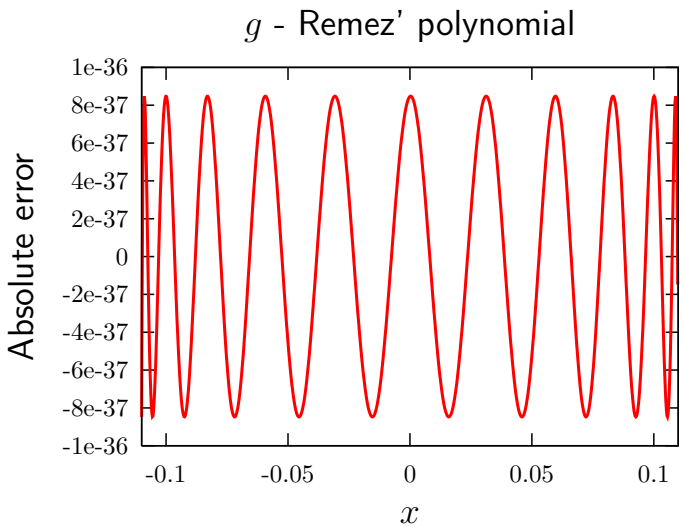
where the p_i are either double precision numbers (d.), a sum of two double precision numbers (d.d.), a sum of two double precision numbers (t.d.).

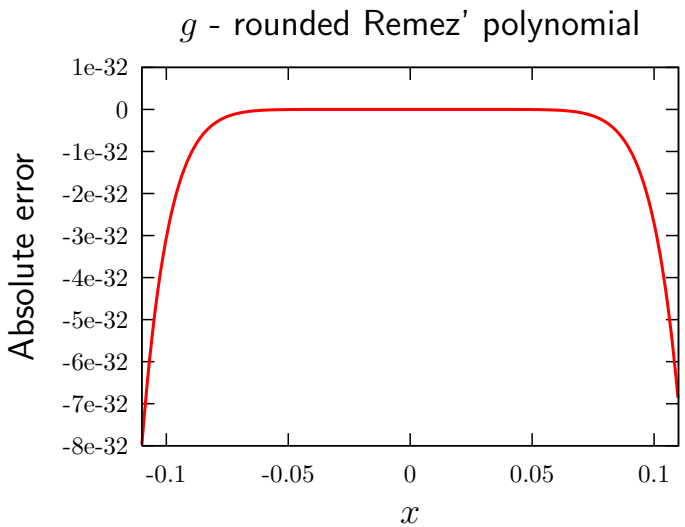
Figure : binary logarithm of the absolute error of several approximants

Target	-119
Minimax	-119.83
Rounded minimax	-103.31
Our polynomial	-119.77

Exact Minimax, Rounded Minimax, our Polynomial

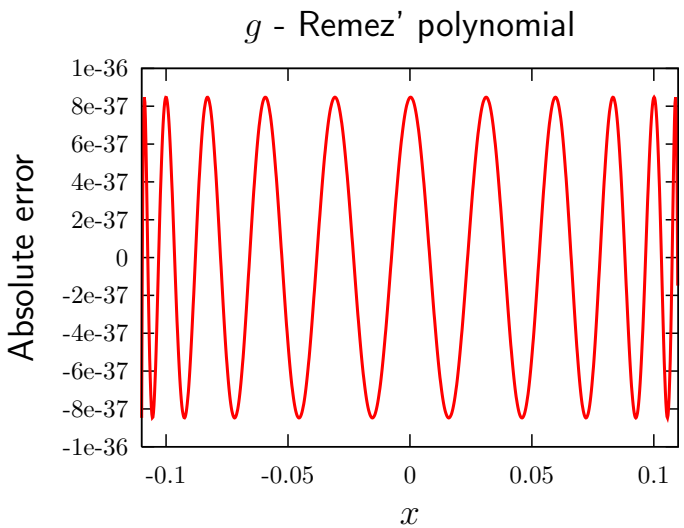
We save 16 bits with our method.

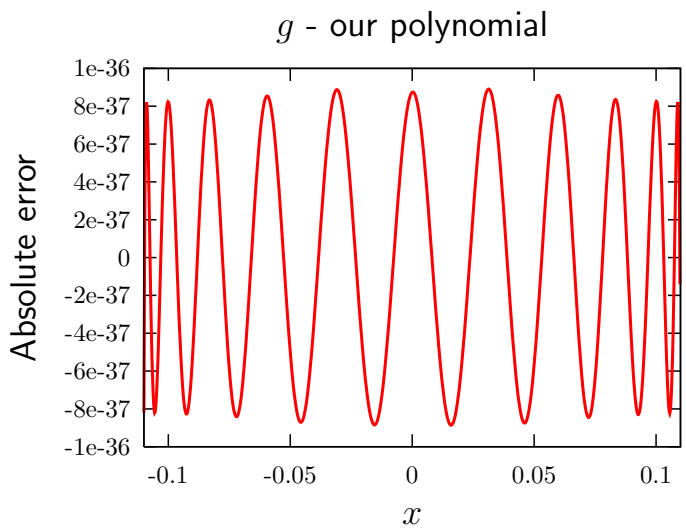




Exact Minimax, Rounded Minimax, our Polynomial

We save 16 bits with our method.





Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 0. Computation of hardest-to-round cases: V. Lefèvre and J.-M. Muller.
- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f .
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.
- Step 4. Computation of a certified evalutation error of $p^\star$: GAPPA (G. Melquiond).

What Kind of Problems can we (CM) Address ?

Currently we consider univariate functions f, "sufficiently smooth" over $[a, b]$.

What Kind of Problems can we (CM) Address ?

Currently we consider univariate functions f, "sufficiently smooth" over $[a, b]$.

Practical Examples:

- Computing supremum norms of approximation error functions:

$$\sup_{x \in [a,\, b]} \{|f(x) - g(x)|\},$$

  where g is a very good approximation of f.

- Rigorous quadrature:

$$J = \int_0^3 \sin\left(\frac{1}{(10^{-3} + (1-x)^2)^{3/2}}\right) \mathrm{d}x = ?$$

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
  (multiple precision IA libraries exist, e.g. MPFI[1])

[1] `http://gforge.inria.fr/projects/mpfi/`

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI[1])
- $\pi \in [3.1415, 3.1416]$

[1] http://gforge.inria.fr/projects/mpfi/

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
  (multiple precision IA libraries exist, e.g. MPFI[1])
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1, 2] + [-3, 2] = [-2, 4]$

[1] http://gforge.inria.fr/projects/mpfi/

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
  (multiple precision IA libraries exist, e.g. MPFI[1])
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1, 2] + [-3, 2] = [-2, 4]$
- Range bounding for functions
  Eg. $x \in [-1, 2], f(x) = x^2 - x + 1$
  $F(X) = X^2 - X + 1$
  $F([-1, 2]) = [-1, 2]^2 - [-1, 2] + [1, 1]$
  $F([-1, 2]) = [0, 4] - [-1, 2] + [1, 1]$
  $F([-1, 2]) = [-1, 6]$

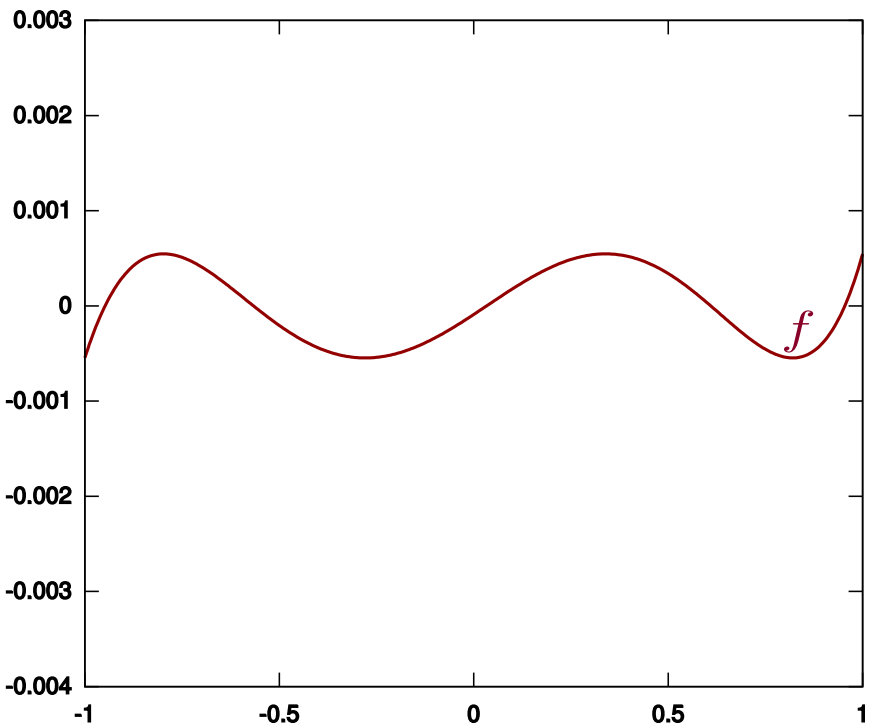
[1] `http://gforge.inria.fr/projects/mpfi/`

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
  (multiple precision IA libraries exist, e.g. MPFI[1])
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1, 2] + [-3, 2] = [-2, 4]$
- Range bounding for functions
  Eg. $x \in [-1, 2], f(x) = x^2 - x + 1$
  $F(X) = X^2 - X + 1$
  $F([-1, 2]) = [-1, 2]^2 - [-1, 2] + [1, 1]$
  $F([-1, 2]) = [0, 4] - [-1, 2] + [1, 1]$
  $F([-1, 2]) = [-1, 6]$
  $x \in [-1, 2], f(x) \in [-1, 6]$, but $\mathtt{Im}(f) = [3/4, 3]$

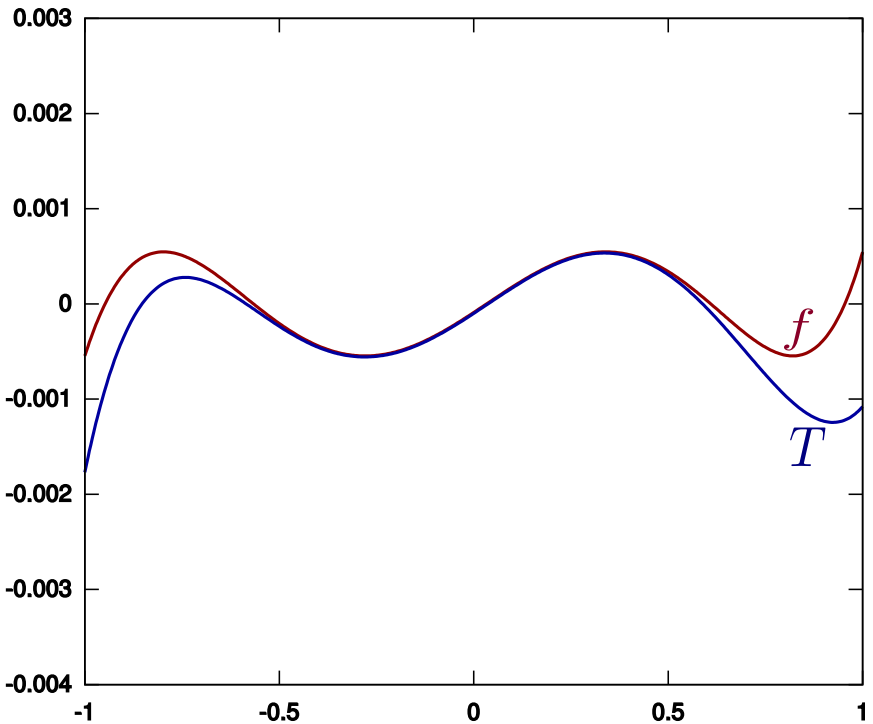
[1] `http://gforge.inria.fr/projects/mpfi/`

Rigorous Polynomial Approximations



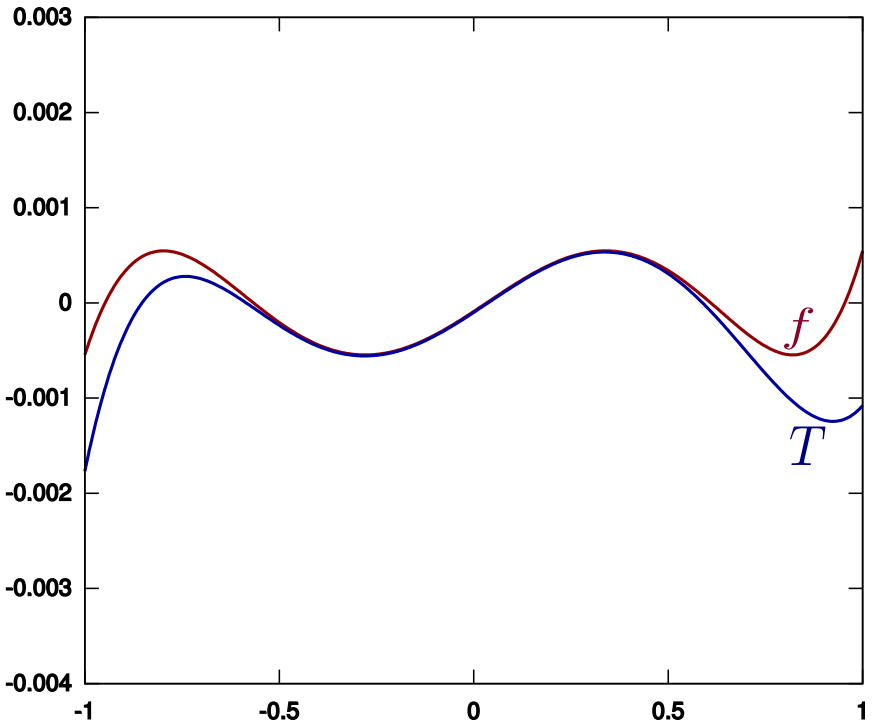
Rigorous Polynomial Approximations

f replaced with
- polynomial approximation T of degree n



Rigorous Polynomial Approximations

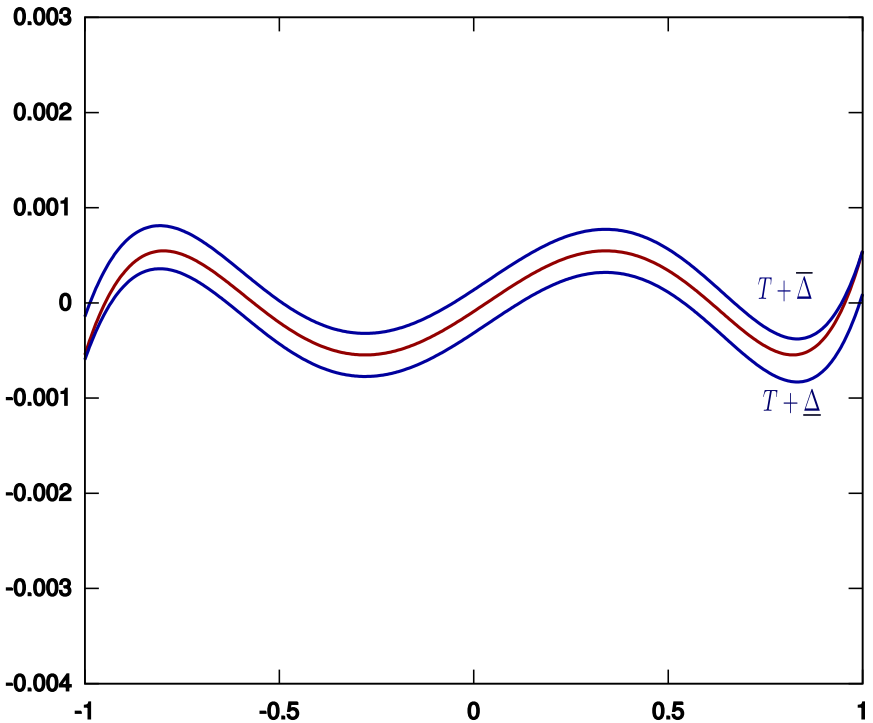
f replaced with
- polynomial approximation T of degree n
- interval $\mathbf{\Delta}$ s. t. $f(x) - T(x) \in \mathbf{\Delta}, \forall x \in [a, b]$



Rigorous Polynomial Approximations

f replaced with a rigorous polynomial approximation : $(T, \mathbf{\Delta})$

- polynomial approximation T of degree n
- interval $\mathbf{\Delta}$ s. t. $f(x) - T(x) \in \mathbf{\Delta}, \forall x \in [a, b]$



Main point: How to compute $(T, \mathbf{\Delta})$?

Rigorous Polynomial Approximations

(Straightforward) Idea: Consider Taylor approximations

Rigorous Polynomial Approximations

(Straightforward) Idea: Consider Taylor approximations

Let $n \in \mathbb{N}$, $n+1$ times differentiable function f over $[a,b]$ around x_0.

- $f(x) = \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!}}_{T(x)} + \underbrace{\Delta_n(x,\xi)}_{\text{remainder}}$
- $\Delta_n(x,\xi) = \frac{(x-x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$, $x \in [a,b]$, ξ lies strictly between x and x_0

Rigorous Polynomial Approximations

(Straightforward) Idea: Consider Taylor approximations

Let $n \in \mathbb{N}$, $n+1$ times differentiable function f over $[a,b]$ around x_0.

- $f(x) = \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!}}_{T(x)} + \underbrace{\Delta_n(x,\xi)}_{\text{remainder}}$
- $\Delta_n(x,\xi) = \frac{(x-x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$, $x \in [a,b]$, ξ lies strictly between x and x_0

- How to compute the coefficients $\frac{f^{(i)}(x_0)}{i!}$ of $T(x)$?
- How to compute an interval enclosure $\mathbf{\Delta}$ for $\Delta_n(x,\xi)$?

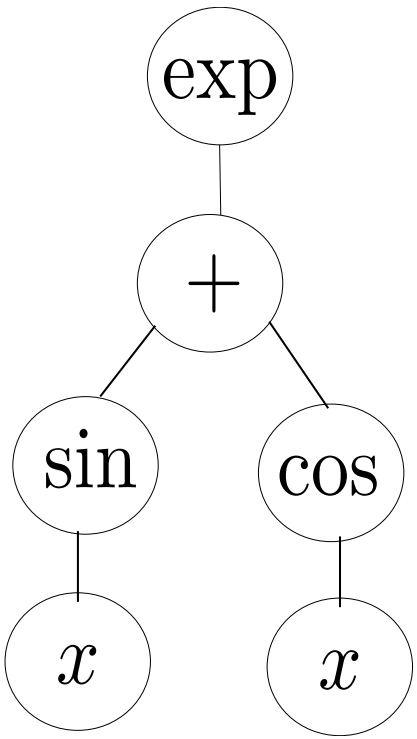
Taylor Models

R. Moore (1950') then M. Berz and K. Makino (1990').

Taylor Models - Two-step procedure

R. Moore (1950') then M. Berz and K. Makino (1990').

Example: $f_{\text{comp}}(x) = \exp(\sin(x) + \cos(x))$



Taylor Model Philosophy

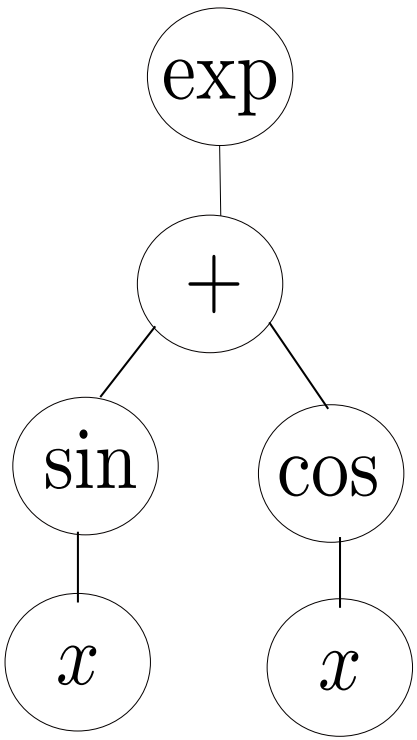
For bounding the remainders:

- For "basic functions" use Lagrange formula.

- For "composite functions" use a two-step procedure:
  - compute models (T, I) for all basic functions;
  - apply algebraic rules with these models, instead of operations with the corresponding functions.

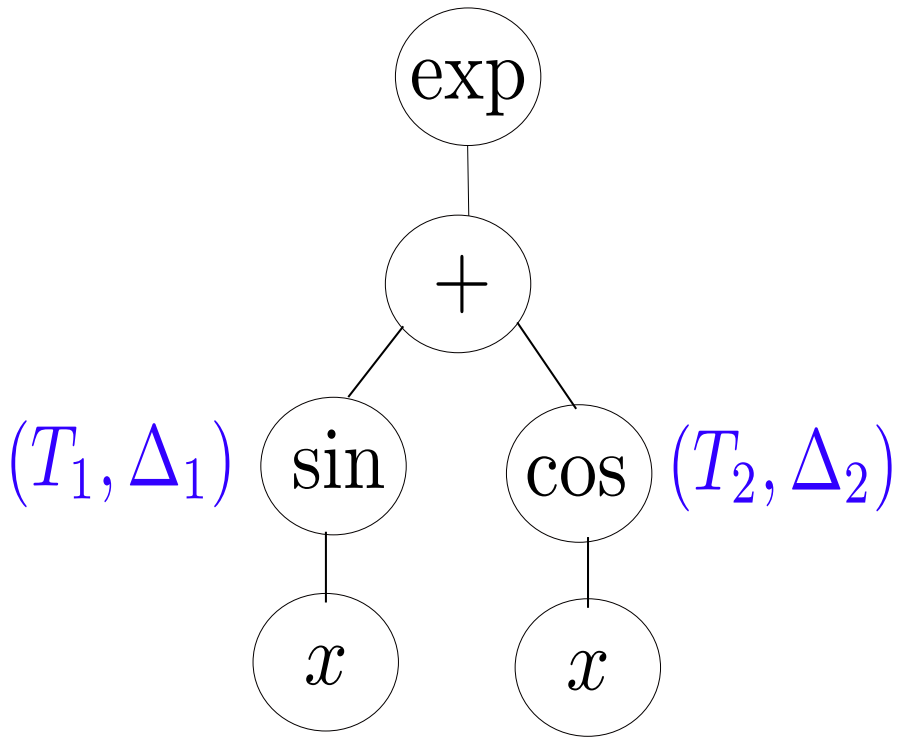
Taylor Models - Two-step procedure

Example: $f_{\text{comp}}(x) = \exp(\sin(x) + \cos(x))$



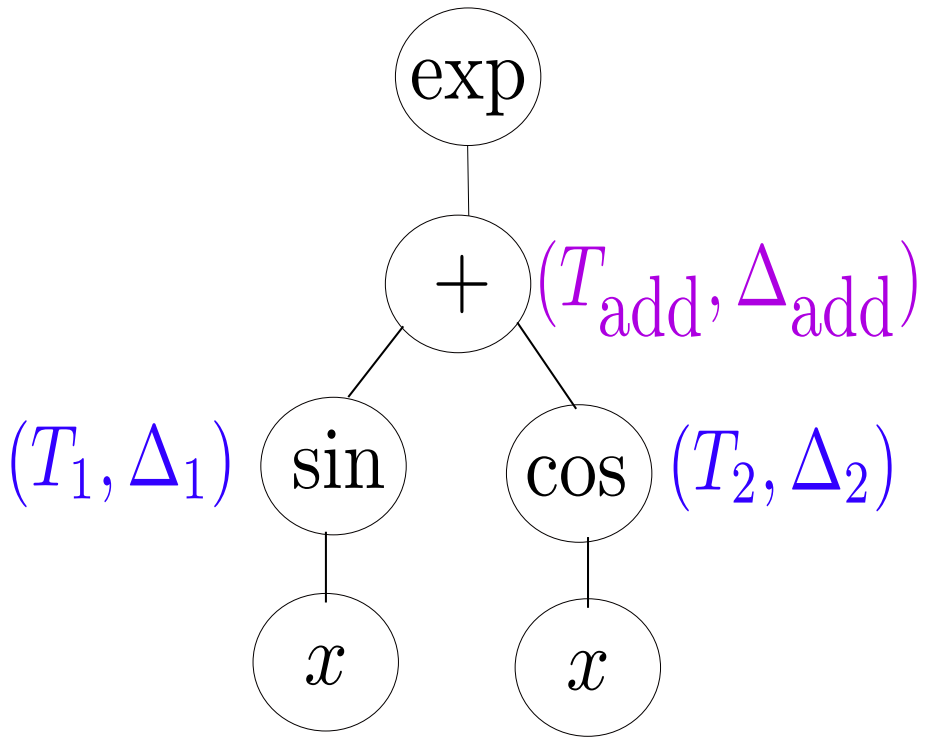
Taylor Models - Two-step procedure

Example: $f_{\text{comp}}(x) = \exp(\sin(x) + \cos(x))$



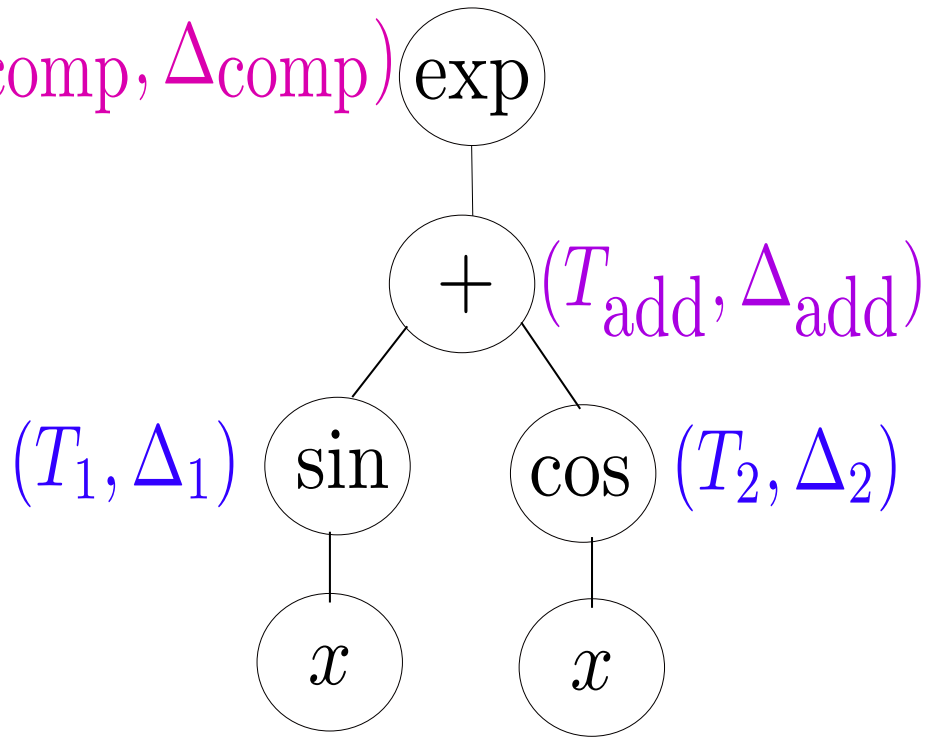
Taylor Models - Two-step procedure

Example: $f_{\text{comp}}(x) = \exp(\sin(x) + \cos(x))$



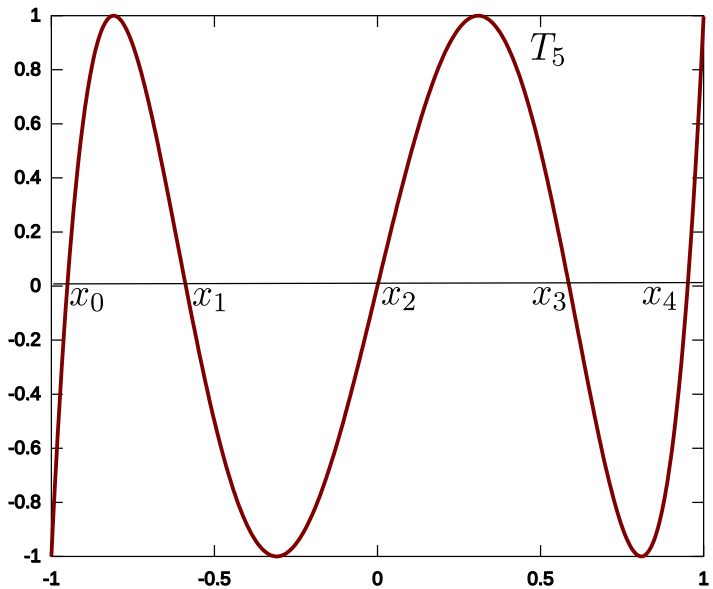
Taylor Models - Two-step procedure

Example: $f_{\text{comp}}(x) = \exp(\sin(x) + \cos(x))$



Quick Reminder on Chebyshev Polynomials

Over $[-1,1]$, $T_n(x) = \cos\left(n \arccos x\right), n \geq 0$.



$(T_n)_{n\in\mathbb{N}}$: orthogonal basis for the scalar product

$$(f,g) = \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\mathrm{d}x.$$

Chebyshev nodes: n roots in $[-1,1]$ of T_n, i.e. $x_i = \cos\left(\left(i+1/2\right)\pi/n\right)$, $i = 0,\ldots,n-1$.

Our Approach - Chebyshev Models

Basic idea:

- Use a polynomial approximation better than Taylor:
  - Chebyshev interpolation polynomial.
  - Chebyshev truncated series.
- Use again the two-step approach:
  - compute models (P, I) for basic functions;
  - apply algebraic rules with these models, instead of operations with the corresponding functions.

Chebyshev Models - Operations: Addition

Given two Chebyshev Models for f_1 and f_2, over $[a, b]$, degree n:
$f_1(x) - P_1(x) \in \mathbf{\Delta}_1$ and $f_2(x) - P_2(x) \in \mathbf{\Delta}_2$, $\forall x \in [a, b]$.

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

Chebyshev Models - Operations: Multiplication

Given two Chebyshev Models for f_1 and f_2, over $[a, b]$, degree n:
$f_1(x) - P_1(x) \in \mathbf{\Delta}_1$ and $f_2(x) - P_2(x) \in \mathbf{\Delta}_2$, $\forall x \in [a, b]$.

Multiplication
We need algebraic rule for: $(P_1, \mathbf{\Delta}_1) \cdot (P_2, \mathbf{\Delta}_2) = (P, \mathbf{\Delta})$ s.t.
$f_1(x) \cdot f_2(x) - P(x) \in \mathbf{\Delta}$, $\forall x \in [a, b]$

Chebyshev Models - Operations: Multiplication

Given two Chebyshev Models for f_1 and f_2, over $[a, b]$, degree n:
$f_1(x) - P_1(x) \in \boldsymbol{\Delta}_1$ and $f_2(x) - P_2(x) \in \boldsymbol{\Delta}_2$, $\forall x \in [a, b]$.

Multiplication
We need algebraic rule for: $(P_1, \boldsymbol{\Delta}_1) \cdot (P_2, \boldsymbol{\Delta}_2) = (P, \boldsymbol{\Delta})$ s.t.
$f_1(x) \cdot f_2(x) - P(x) \in \boldsymbol{\Delta}$, $\forall x \in [a, b]$

$$f_1(x) \cdot f_2(x) \in \underbrace{P_1 \cdot P_2}_{\underbrace{(P_1 \cdot P_2)_{0\ldots n}}_{P} + \underbrace{(P_1 \cdot P_2)_{n+1\ldots 2n}}_{\boldsymbol{I_1}}} + \underbrace{\boldsymbol{P_2} \cdot \boldsymbol{\Delta}_1 + \boldsymbol{P_1} \cdot \boldsymbol{\Delta}_2 + \boldsymbol{\Delta}_1 \cdot \boldsymbol{\Delta}_2}_{\boldsymbol{I_2}}.$$

$$\boldsymbol{\Delta} = \boldsymbol{I_1} + \boldsymbol{I_2}$$

Chebyshev Models - Operations: Multiplication

Given two Chebyshev Models for f_1 and f_2, over $[a, b]$, degree n:
$f_1(x) - P_1(x) \in \boldsymbol{\Delta}_1$ and $f_2(x) - P_2(x) \in \boldsymbol{\Delta}_2$, $\forall x \in [a, b]$.

Multiplication
We need algebraic rule for: $(P_1, \boldsymbol{\Delta}_1) \cdot (P_2, \boldsymbol{\Delta}_2) = (P, \boldsymbol{\Delta})$ s.t.
$f_1(x) \cdot f_2(x) - P(x) \in \boldsymbol{\Delta}$, $\forall x \in [a, b]$

$$f_1(x) \cdot f_2(x) \in \underbrace{P_1 \cdot P_2}_{P} + \underbrace{\boldsymbol{P_2} \cdot \boldsymbol{\Delta}_1 + \boldsymbol{P_1} \cdot \boldsymbol{\Delta}_2 + \boldsymbol{\Delta}_1 \cdot \boldsymbol{\Delta}_2}_{\boldsymbol{I_2}}.$$

$$\underbrace{(P_1 \cdot P_2)_{0\ldots n}}_{P} + \underbrace{(P_1 \cdot P_2)_{n+1\ldots 2n}}_{\boldsymbol{I_1}}$$

$$\boldsymbol{\Delta} = \boldsymbol{I_1} + \boldsymbol{I_2}$$

In our case, for bounding "$\boldsymbol{P}$s": $\boldsymbol{P} = p_0 + \sum_{i=1}^{n} p_i \cdot [-1, 1]$.

Chebyshev Models - Operations: Composition

Given CMs for f_1 over $[c,d]$, for f_2 over $[a,b]$, degree n:
$f_1(y) - P_1(y) \in \boldsymbol{\Delta}_1$, $\forall y \in [c,d]$ and $f_2(x) - P_2(x) \in \boldsymbol{\Delta}_2$, $\forall x \in [a,b]$.

Remark: $(f_1 \circ f_2)(x)$ is f_1 evaluated at $y = f_2(x)$.
We need: $f_2([a,b]) \subseteq [c,d]$, checked by $\boldsymbol{P_2} + \boldsymbol{\Delta}_2 \subseteq [c,d]$

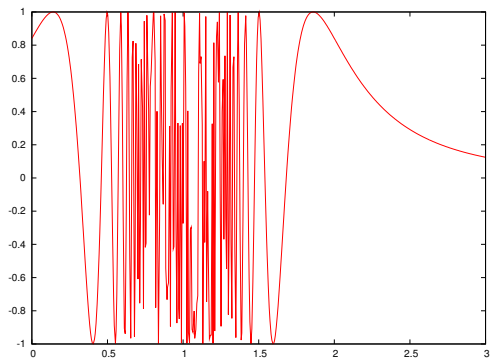
$$f_1(f_2(x)) \in P_1(P_2(x) + \boldsymbol{\Delta}_2) + \boldsymbol{\Delta}_1$$

Extract polynomial and remainder: P_1 can be evaluated using only additions and multiplications: Clenshaw's algorithm

Various bugs

M. Joldeș. Rigorous Polynomial Approximations and Applications. PhD thesis, ENS Lyon, 2011.

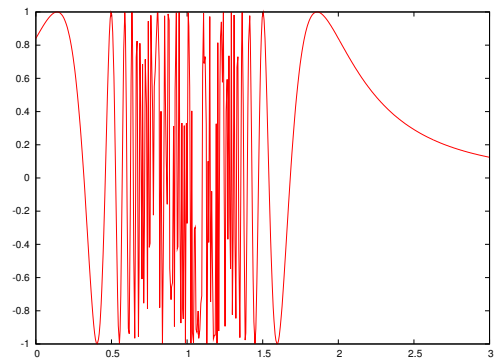
Let $J = \int_0^3 \sin\left(\frac{1}{(10^{-3} + (1-x)^2)^{3/2}}\right) \mathrm{d}x$.



Various bugs

M. Joldeș. Rigorous Polynomial Approximations and Applications. PhD thesis, ENS Lyon, 2011.

Let $J = \int_0^3 \sin\left(\frac{1}{(10^{-3} + (1-x)^2)^{3/2}}\right) \mathrm{d}x$.

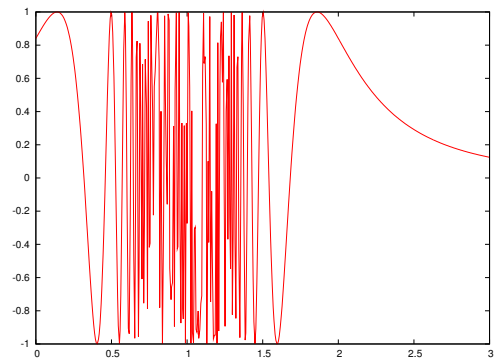


- Maple15: 0.7499743685;
- Pari/GP: $0.79277309714790807 55$;
- Mathematica, Chebfun fail to answer;
- Chen, '06: 0.7578918118.

Various bugs

M. Joldeș. Rigorous Polynomial Approximations and Applications. PhD thesis, ENS Lyon, 2011.

Let $J = \int_0^3 \sin\left(\frac{1}{(10^{-3} + (1-x)^2)^{3/2}}\right) \mathrm{d}x$.



- Maple15: 0.7499743685;
- Pari/GP: $0.79277309714790807 55$;
- Mathematica, Chebfun fail to answer;
- Chen, '06: 0.7578918118.

  Correct answer: $J \in 0.749974368527[1;3]$!

Current work

- Quantization problem. Digital filter synthesis (work with Silviu Filip and Guillaume Hanrot).
  Issue: the coefficients are machine numbers.

Current work

- Quantization problem. Digital filter synthesis (work with Silviu Filip and Guillaume Hanrot).
  Issue: the coefficients are machine numbers.

- Certified spectral methods. Given f a solution of a linear ODE, given $(\varphi_n)_{n\in\mathbb{N}}$ a family of orthogonal polynomials, compute a pair (P, Δ) where P is a polynomial expressed in the basis $(\varphi_n)_{n\in\mathbb{N}}$ and Δ an interval such that $f - P$ takes all its values in Δ.

Current work

- Quantization problem. Digital filter synthesis (work with Silviu Filip and Guillaume Hanrot).
  Issue: the coefficients are machine numbers.

- Certified spectral methods. Given f a solution of a linear ODE, given $(\varphi_n)_{n\in\mathbb{N}}$ a family of orthogonal polynomials, compute a pair (P, Δ) where P is a polynomial expressed in the basis $(\varphi_n)_{n\in\mathbb{N}}$ and Δ an interval such that $f - P$ takes all its values in Δ.

  Chebyshev basis: beautiful work by Alexandre Benoit, Mioara Joldeș and Marc Mezzarobba, based on very nice results by A. Benoit and Bruno Salvy.

Current work

- Quantization problem. Digital filter synthesis (work with Silviu Filip and Guillaume Hanrot).
  Issue: the coefficients are machine numbers.

- Certified spectral methods. Given f a solution of a linear ODE, given $(\varphi_n)_{n\in\mathbb{N}}$ a family of orthogonal polynomials, compute a pair (P, Δ) where P is a polynomial expressed in the basis $(\varphi_n)_{n\in\mathbb{N}}$ and Δ an interval such that $f - P$ takes all its values in Δ.

  Chebyshev basis: beautiful work by Alexandre Benoit, Mioara Joldeș and Marc Mezzarobba, based on very nice results by A. Benoit and Bruno Salvy.

  Gegenbauer polynomials: work with Thomas Grégoire.

Evaluation of Elementary Functions

$\exp, \ln, \cos, \sin, \arctan, \sqrt{}, \ldots$

Goal: evaluation of φ to a given accuracy η.

- Step 0. Computation of hardest-to-round cases: V. Lefèvre and J.-M. Muller.
- Step 1. Argument reduction (Payne & Hanek, Ng, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f .
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.
- Step 4. Computation of a certified evalutation error of $p^\star$: GAPPA (G. Melquiond).