Efficient polynomial L_∞-approximations

ARITH 18 - Montpellier

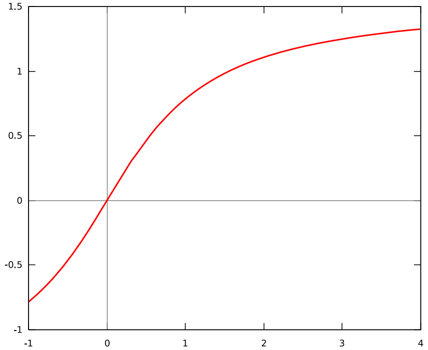
Nicolas Brisebarre **Sylvain Chevillard**

Laboratoire de l'informatique du parallélisme
Arenaire team

June 26, 2007

Contents

Functions approximation

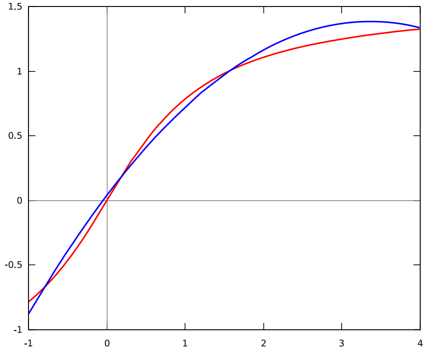


Graph of $f : x \mapsto \arctan(x)$

(interval $[-1, 4]$)

- Let f be a real valued function : $f : [a, b] \longrightarrow \mathbb{R}$.

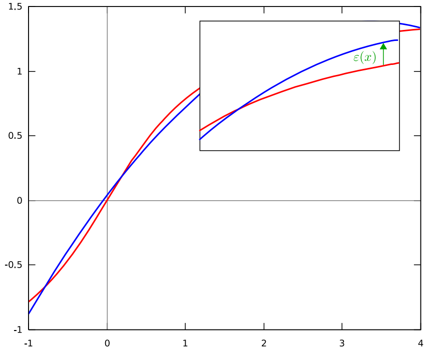
Functions approximation



($\mathbb{R}_n[X]$: set of polynomials with real coefficients and degree at most n). Here $n = 2$

- Let f be a real valued function : $f : [a, b] \longrightarrow \mathbb{R}$.
- Let $p \in \mathbb{R}_n[X]$ approximating f.

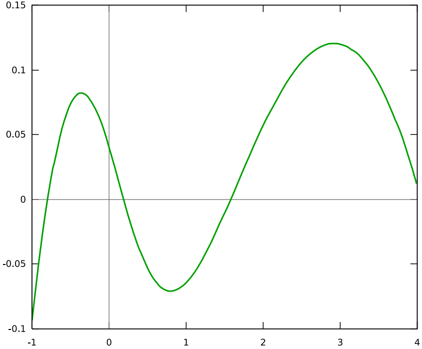
Functions approximation



($\mathbb{R}_n[X]$: set of polynomials with real coefficients and degree at most n). Here $n = 2$

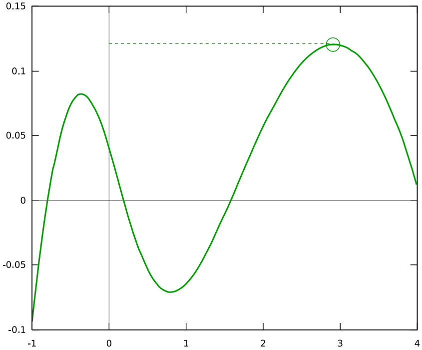
- Let f be a real valued function : $f : [a, b] \longrightarrow \mathbb{R}$.
- Let $p \in \mathbb{R}_n[X]$ approximating f.
- Approximation error at point x: $\varepsilon(x) = p(x) - f(x)$.

Approximation error



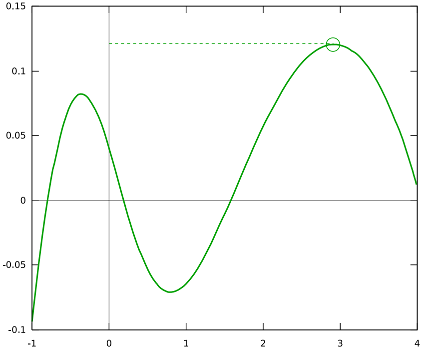
- $\varepsilon(x) = p(x) - f(x)$ over $[a,\ b]$

Approximation error



- $\varepsilon(x) = p(x) - f(x)$ over $[a,\ b]$
- $\max\{|\varepsilon(x)|,\ x \in [a,\ b]\}$

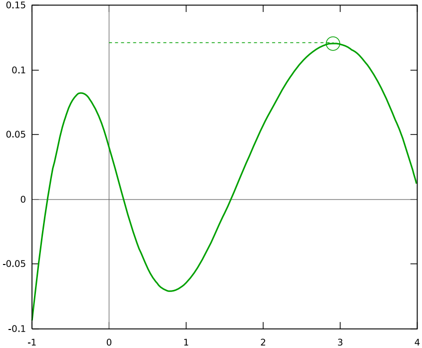
Approximation error



- $\varepsilon(x) = p(x) - f(x)$ over $[a, b]$
- $\max\{|\varepsilon(x)|, x \in [a, b]\}$

- Infinite norm: $\|\varepsilon\|_\infty = \|p - f\|_\infty = \max\{|\varepsilon(x)|, x \in [a, b]\}$

Approximation error



- $\varepsilon(x) = p(x) - f(x)$ over $[a,\ b]$
- $\max\{|\varepsilon(x)|,\ x \in [a,\ b]\}$

- Infinite norm: $\|\varepsilon\|_\infty = \|p - f\|_\infty = \max\{|\varepsilon(x)|,\ x \in [a,\ b]\}$
- Best approximation problem:

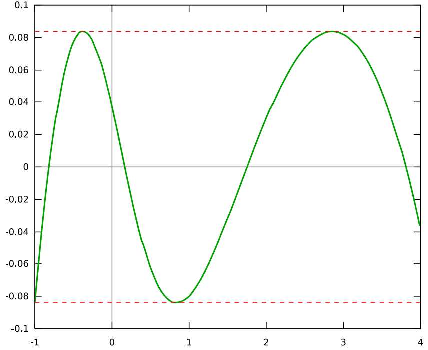
  given a degree n, find $p \in \mathbb{R}_n[X]$ minimizing $\|p - f\|_\infty$.

Theory of polynomial approximation

Facts:

- There exists a unique best approximation polynomial.

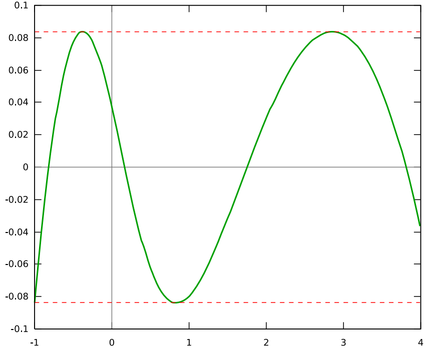
Theory of polynomial approximation



Facts:

- There exists a unique best approximation polynomial.
- Characterization: Chebyshev's theorem.

Theory of polynomial approximation



Facts:

- There exists a unique best approximation polynomial.
- Characterization: Chebyshev's theorem.
- To compute it: Remez' algorithm (`minimax` in Maple).

The problem

- Computers: finite memory.

The problem

- Computers: finite memory.
- `IEEE-754` standard: defines floating-point numbers.

The problem

- Computers: finite memory.
- `IEEE-754` standard: defines floating-point numbers.
- A floating-point number with radix 2 and precision t, is a number of the form $x = m \cdot 2^e$ where
  - $m \in \mathbb{Z}$ (written with exactly t bits) is called its mantissa;
  - $e \in \mathbb{Z}$ is its exponent.

The problem

- Computers: finite memory.
- `IEEE-754` standard: defines floating-point numbers.
- A floating-point number with radix 2 and precision t, is a number of the form $x = m \cdot 2^e$ where
  - $m \in \mathbb{Z}$ (written with exactly t bits) is called its mantissa;
  - $e \in \mathbb{Z}$ is its exponent.
- In practice: one has to store the coefficients into floating-point numbers.

The problem

- Computers: finite memory.
- `IEEE-754` standard: defines floating-point numbers.
- A floating-point number with radix 2 and precision t, is a number of the form $x = m \cdot 2^e$ where
  - $m \in \mathbb{Z}$ (written with exactly t bits) is called its mantissa;
  - $e \in \mathbb{Z}$ is its exponent.
- In practice: one has to store the coefficients into floating-point numbers.
- Naive method: compute the minimax with Remez' algorithm and a high precision. Then round each coefficient to the nearest floating-point number.

Failure of the naive method

- Example with $f(x) = \log_2(1 + 2^{-x})$:
  - on $[0; 1]$
  - approximated by a degree 6 polynomial
  - with single precision coefficients (24 bits).

Minimax	Naive method	Optimal
$8.3 \cdot 10^{-10}$	$119 \cdot 10^{-10}$	$10.06 \cdot 10^{-10}$

Failure of the naive method

- Example with $f(x) = \log_2(1 + 2^{-x})$:
  - on $[0; 1]$
  - approximated by a degree 6 polynomial
  - with single precision coefficients (24 bits).

Minimax	Naive method	Optimal
$8.3 \cdot 10^{-10}$	$119 \cdot 10^{-10}$	$10.06 \cdot 10^{-10}$

- The problem has been studied by
  - W. Kahan;
  - D. Kodek (precision $t < 10$, degree $n < 20$);
  - N. Brisebarre, J.-M. Muller and A. Tisserand (using linear programming).

Description of our method

Our goal: find p approximating f with the following form:

$$m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} X + \cdots + m_n \cdot 2^{e_n} X^n \qquad (m_i \in \mathbb{Z}).$$

Description of our method

Our goal: find p approximating f with the following form:

$$m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} X + \cdots + m_n \cdot 2^{e_n} X^n \qquad (m_i \in \mathbb{Z}).$$

- We use the idea of interpolation:

Description of our method

Our goal: find p approximating f with the following form:

$$m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} X + \cdots + m_n \cdot 2^{e_n} X^n \qquad (m_i \in \mathbb{Z}).$$

- We use the idea of interpolation:
  - we choose $n+1$ points $x_0, \cdots, x_n$ in $[a, b]$;

Description of our method

Our goal: find p approximating f with the following form:

$$m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} X + \cdots + m_n \cdot 2^{e_n} X^n \qquad (m_i \in \mathbb{Z}).$$

- We use the idea of interpolation:
  - we choose $n+1$ points $x_0, \cdots, x_n$ in $[a, b]$;
  - we search $m_0, \cdots, m_n$ such that for all i

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

Description of our method

Our goal: find p approximating f with the following form:

$$m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} X + \cdots + m_n \cdot 2^{e_n} X^n \qquad (m_i \in \mathbb{Z}).$$

- We use the idea of interpolation:
  - we choose $n+1$ points $x_0, \cdots, x_n$ in $[a, b]$;
  - we search $m_0, \cdots, m_n$ such that for all i

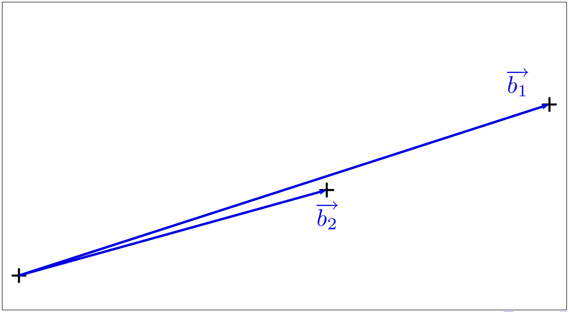
$$p(x_i) = m_0 \cdot 2^{e_0} + m_1 \cdot 2^{e_1} x_i + \cdots + m_n \cdot 2^{e_n} x_i^n \simeq f(x_i) \quad .$$

  - Rewritten with vectors:

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

Notions about lattices

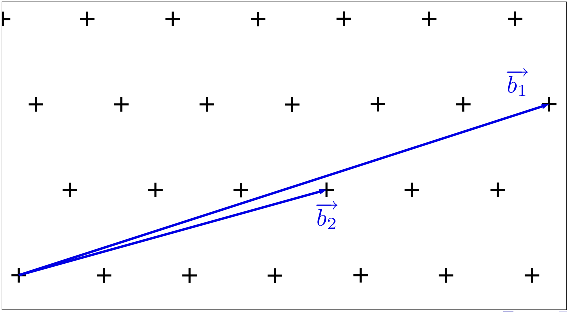
Let $(\overrightarrow{b_1}, \cdots, \overrightarrow{b_n})$ be a basis of a real vector space.



Notions about lattices

Let $(\overrightarrow{b_1}, \cdots, \overrightarrow{b_n})$ be a basis of a real vector space. The set of all integer combinations of the $\overrightarrow{b_i}$ is called a lattice:

$$\Gamma = \mathbb{Z}\overrightarrow{b_1} + \mathbb{Z}\overrightarrow{b_2} + \cdots + \mathbb{Z}\overrightarrow{b_n} \quad .$$

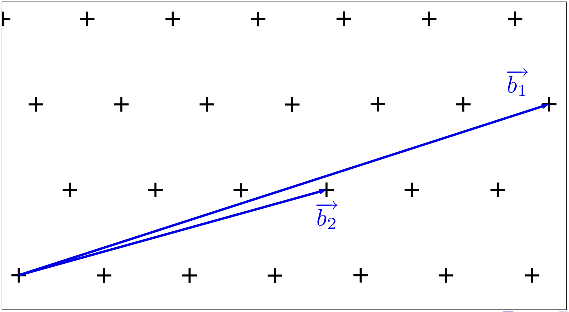


Notions about lattices

Let $(\overrightarrow{b_1}, \cdots, \overrightarrow{b_n})$ be a basis of a real vector space. The set of all integer combinations of the $\overrightarrow{b_i}$ is called a lattice:

$$\Gamma = \mathbb{Z}\overrightarrow{b_1} + \mathbb{Z}\overrightarrow{b_2} + \cdots + \mathbb{Z}\overrightarrow{b_n} \quad .$$

In general, a lattice has infinitely many bases.

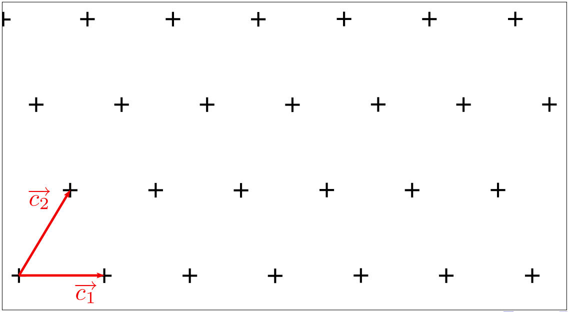


Notions about lattices

Let $(\overrightarrow{b_1}, \cdots, \overrightarrow{b_n})$ be a basis of a real vector space. The set of all integer combinations of the $\overrightarrow{b_i}$ is called a lattice:

$$\Gamma = \mathbb{Z}\overrightarrow{b_1} + \mathbb{Z}\overrightarrow{b_2} + \cdots + \mathbb{Z}\overrightarrow{b_n} \quad .$$

In general, a lattice has infinitely many bases.

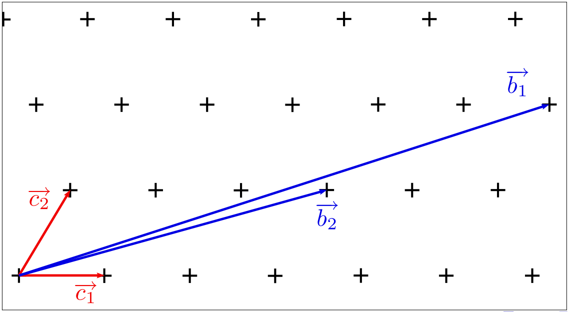


Notions about lattices

Let $(\overrightarrow{b_1}, \cdots, \overrightarrow{b_n})$ be a basis of a real vector space. The set of all integer combinations of the $\overrightarrow{b_i}$ is called a lattice:

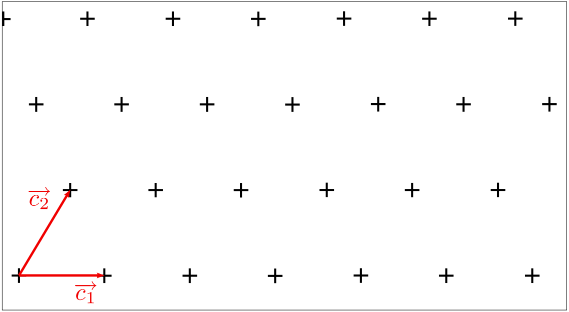
$$\Gamma = \mathbb{Z}\overrightarrow{b_1} + \mathbb{Z}\overrightarrow{b_2} + \cdots + \mathbb{Z}\overrightarrow{b_n} \quad .$$

In general, a lattice has infinitely many bases.



Notions about lattices

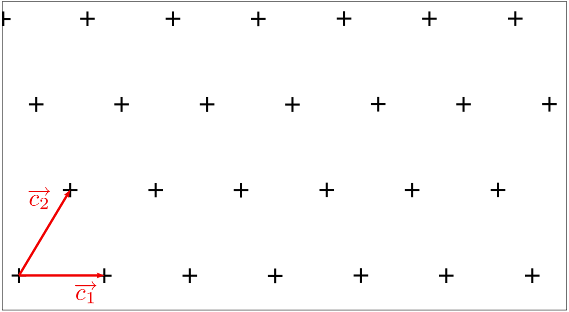
Algorithmic problems:



Notions about lattices

Algorithmic problems:

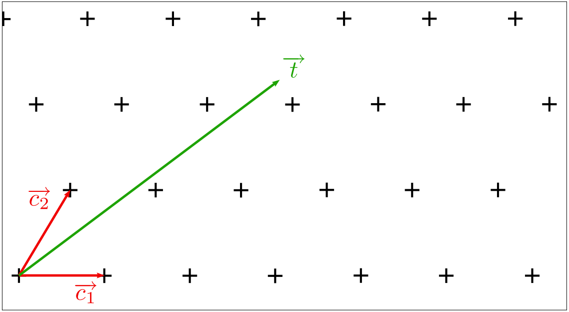
- Shortest vector problem (SVP)



Notions about lattices

Algorithmic problems:

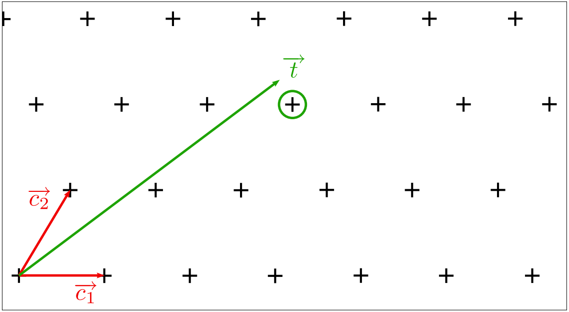
- Shortest vector problem (SVP)
- Closest vector problem (CVP)



Notions about lattices

Algorithmic problems:

- Shortest vector problem (SVP)
- Closest vector problem (CVP)

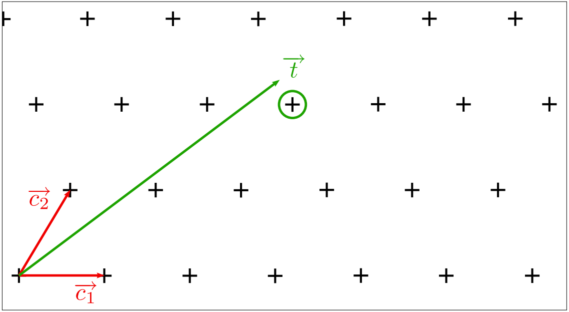


Notions about lattices

Algorithmic problems:

- Shortest vector problem (SVP)
- Closest vector problem (CVP)

LLL algorithm: Lenstra, Lenstra Jr. and Lovász.

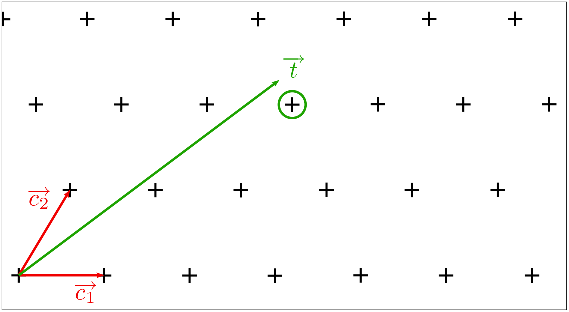


Notions about lattices

Algorithmic problems:

- Shortest vector problem (SVP)
- Closest vector problem (CVP)

LLL algorithm: finds pretty short vectors in polynomial time.

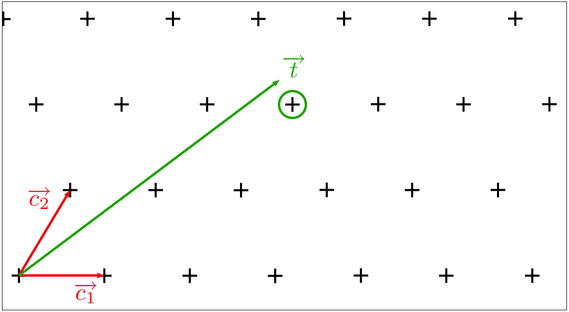


Notions about lattices

Algorithmic problems:

- Shortest vector problem (SVP)
- Closest vector problem (CVP)

LLL algorithm: used by Babai to solve an approximation of CVP.



A concrete and toy case

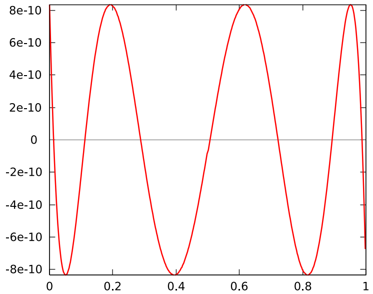
- We want to approximate $f : x \mapsto \log_2(1 + 2^{(-x)})$

A concrete and toy case

- We want to approximate $f : x \mapsto \log_2(1 + 2^{(-x)})$
  - on $[0, 1]$
  - by a polynomial of degree 6.
  - Each coefficient is stored in a single-precision number (24 bits).

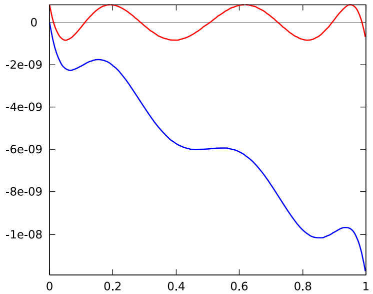
Datas

Best real	Naive method	Enhanced method
$8.34\mathrm{e}{-10}$	$119\mathrm{e}{-10}$	$49.9\mathrm{e}{-10}$



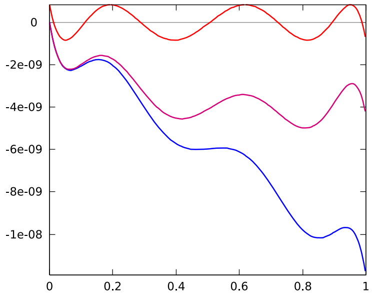
Datas

Best real	Naive method	Enhanced method
8.34e−10	119e−10	49.9e−10



Datas

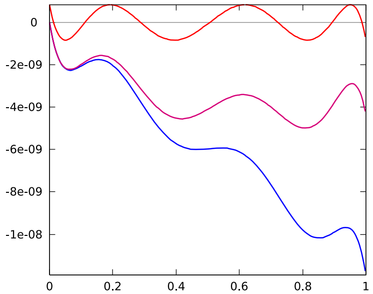
Best real	Naive method	Enhanced method
$8.34\mathrm{e}{-10}$	$119\mathrm{e}{-10}$	$49.9\mathrm{e}{-10}$



Datas

Best real	Naive method	Enhanced method
$8.34\mathrm{e}{-10}$	$119\mathrm{e}{-10}$	$49.9\mathrm{e}{-10}$

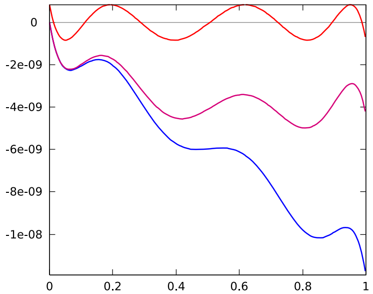
- How to choose the points?



Datas

Best real	Naive method	Enhanced method
$8.34\text{e}{-10}$	$119\text{e}{-10}$	$49.9\text{e}{-10}$

- How to choose the points?

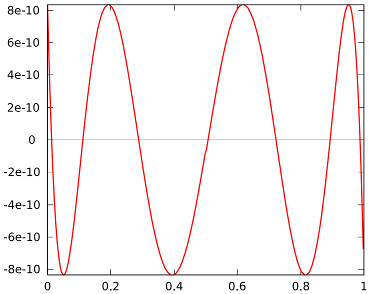


- We need $n+1$ points.

Datas

Best real	Naive method	Enhanced method
8.34e−10	119e−10	49.9e−10

- How to choose the points?

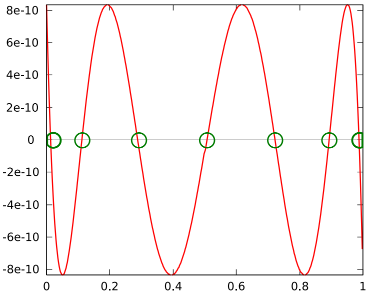


- We need $n+1$ points.
- They should correspond to the interpolation intuition.

Datas

Best real	Naive method	Enhanced method
$8.34\mathrm{e}{-10}$	$119\mathrm{e}{-10}$	$49.9\mathrm{e}{-10}$

- How to choose the points?



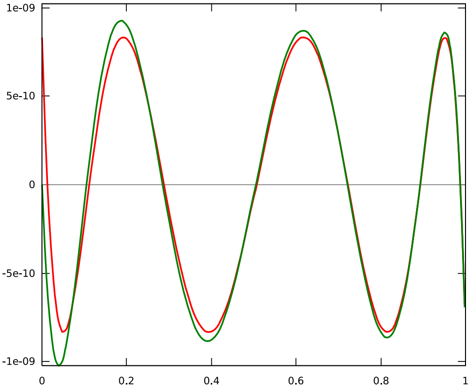
- We need $n+1$ points.
- They should correspond to the interpolation intuition.
- Chebyshev's theorem gives $n+1$ such points.

Results with our method

Best real	Naive method	Enhanced method	Our method
$8.34\mathrm{e}{-10}$	$119\mathrm{e}{-10}$	$49.9\mathrm{e}{-10}$	$10.24\mathrm{e}{-10}$

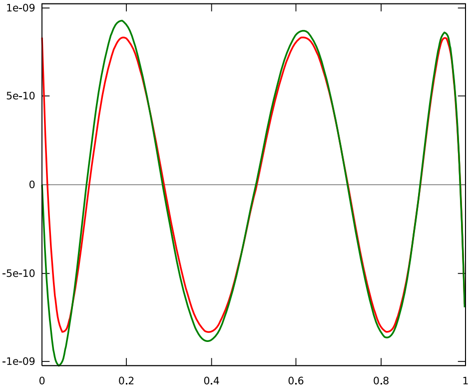
Results with our method

Best real	Naive method	Enhanced method	Our method
8.34e−10	119e−10	49.9e−10	10.24e−10



Results with our method

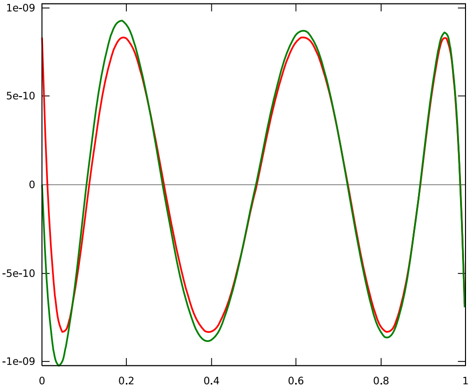
Best real	Naive method	Enhanced method	Our method
8.34e−10	119e−10	49.9e−10	10.24e−10



- Polynomial obtained in less than 1 second (Pentium III 1.2GHz)

Results with our method

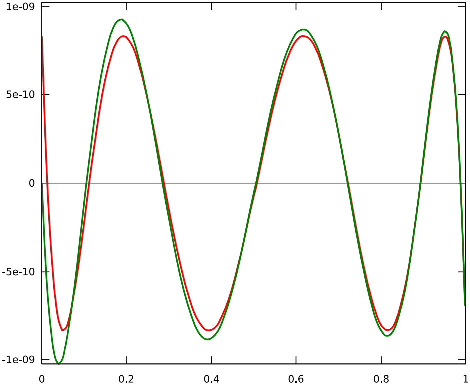
Best real	Naive method	Enhanced method	Our method
8.34e−10	119e−10	49.9e−10	10.24e−10



- Polynomial obtained in less than 1 second (Pentium III 1.2GHz)
- Degree 30 and precision ≈ 100 obtained in a few seconds

Results with our method

Best real	Naive method	Enhanced method	Our method
$8.34\mathrm{e}{-10}$	$119\mathrm{e}{-10}$	$49.9\mathrm{e}{-10}$	$10.24\mathrm{e}{-10}$



- Polynomial obtained in less than 1 second (Pentium III 1.2GHz)
- Degree 30 and precision ≈ 100 obtained in a few seconds
- Used in `CRlibm`

Conclusion

- We have developed an algorithm to find very good polynomial approximants with floating-point coefficients.

Conclusion

- We have developed an algorithm to find very good polynomial approximants with floating-point coefficients.
- The algorithm is a heuristic, but works well in practice.

Conclusion

- We have developed an algorithm to find very good polynomial approximants with floating-point coefficients.
- The algorithm is a heuristic, but works well in practice.
- The algorithm is flexible:
  - each coefficient may use a different floating-point format;
  - one may search polynomial with additional constraints: fix the value of some coefficients, search for an even polynomial;
  - one may optimize the relative error

$$\varepsilon(x) = \frac{p(x) - f(x)}{f(x)}$$

  instead of the absolute error.

Conclusion

- We have developed an algorithm to find very good polynomial approximants with floating-point coefficients.
- The algorithm is a heuristic, but works well in practice.
- The algorithm is flexible:
  - each coefficient may use a different floating-point format;
  - one may search polynomial with additional constraints: fix the value of some coefficients, search for an even polynomial;
  - one may optimize the relative error

$$\varepsilon(x) = \frac{p(x) - f(x)}{f(x)}$$

  instead of the absolute error.

Focus on polynomial approximation

- The definition often gives a natural way to find a polynomial approximation of f.
  $\hookrightarrow$ for instance: a truncated power series with a formally computed bound on the error.

Focus on polynomial approximation

- The definition often gives a natural way to find a polynomial approximation of f.
  $\hookrightarrow$ for instance: a truncated power series with a formally computed bound on the error.
- Truncated power series are useful but...

Focus on polynomial approximation

- The definition often gives a natural way to find a polynomial approximation of f.
  $\hookrightarrow$ for instance: a truncated power series with a formally computed bound on the error.
- Truncated power series are useful but…
  …usually inefficient in term of number of operations.
- Example: $\exp(x)$ on $[-1; 2]$ with an absolute error ≤ 0.01:
  - the series must be truncated to a degree 7 polynomial;
  - a degree 4 polynomial is sufficient.

Chebyshev's theorem

Theorem (Chebyshev)

Let f be a continuous function on $[a, b]$. Let $\mu = \inf\{\|f - p\|_\infty\}_{p \in \mathbb{R}_n[X]}$. Then, p satisfies $\|f - p\|_\infty = \mu$ if and only if there exist $n + 2$ points

$$x_0 < x_1 < \cdots < x_{n+1}$$

in $[a, b]$ such that

1. $\forall i \in [\![0, n+1]\!], |f(x_i) - p(x_i)| = \|f - p\|_\infty$
2. *For all $i \in [\![0, n]\!]$, the signs of $f(x_{i+1}) - p(x_{i+1})$ and $f(x_i) - p(x_i)$ are different.*