

Polynomial approximation and floating-point numbers

Algorithms Project Seminar

Sylvain Chevillard

Advisors: Nicolas Brisebarre and Jean-Michel Muller
joint work with Serge Torres

Laboratoire de l'informatique du parallélisme
Arenaire team

June, 12. 2007

Contents

Scope of my researches

Approximation theory

Polynomial approximation with floating-point numbers

Lattices and LLL algorithm

A concrete case

Conclusion

Presentation of Arenaire

- ▶ Arenaire team : the main goal is the practical computation of mathematical functions.

Presentation of Arenaire

- ▶ Arenaire team : the main goal is the practical computation of mathematical functions.
- ▶ General scheme :
 - ▶ we want to compute a mathematical operator Θ ;
 - ▶ we may use an approximation $\hat{\Theta}$ of Θ ;
 - ▶ we implement it with inexact arithmetic, controlling the round-off error.

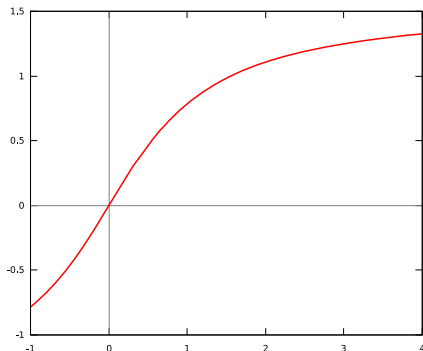
Presentation of Arenaire

- ▶ Arenaire team : the main goal is the practical computation of mathematical functions.
- ▶ General scheme :
 - ▶ we want to compute a mathematical operator Θ ;
 - ▶ we may use an approximation $\hat{\Theta}$ of Θ ;
 - ▶ we implement it with inexact arithmetic, controlling the round-off error.
- ▶ The scheme covers :
 - ▶ hardware implementation of mathematical functions ;
 - ▶ software implementation targeting IEEE correct rounding in double precision format ;
 - ▶ certified software implementation with arbitrary high precision ;
 - ▶ certified implementation of numerical algorithms (QR decomposition, lattice reduction...)

Presentation of Arenaire

- ▶ Arenaire team : the main goal is the practical computation of mathematical functions.
- ▶ General scheme :
 - ▶ we want to compute a mathematical operator Θ ;
 - ▶ we may use an approximation $\hat{\Theta}$ of Θ ;
 - ▶ we implement it with inexact arithmetic, controlling the round-off error.
- ▶ The scheme covers :
 - ▶ hardware implementation of mathematical functions ;
 - ▶ software implementation targeting IEEE correct rounding in double precision format ;
 - ▶ certified software implementation with arbitrary high precision ;
 - ▶ certified implementation of numerical algorithms (QR decomposition, lattice reduction...)

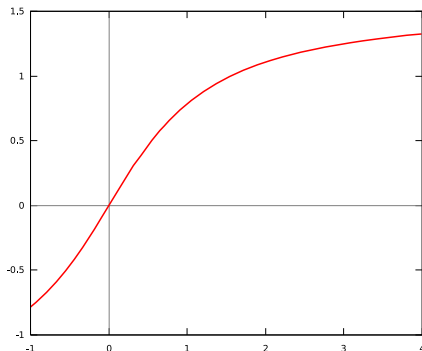
Why an approximation ?



- ▶ Let f be a real valued function : $f : \mathbb{R} \rightarrow \mathbb{R}$.

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

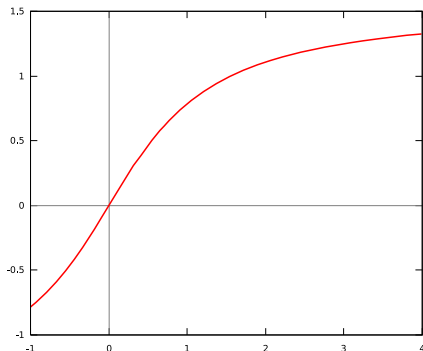
Why an approximation ?



- ▶ Let f be a real valued function : $f : \mathbb{R} \rightarrow \mathbb{R}$.
- ▶ The function may take irrational values : $f(x)$ is thus not exactly representable.

$$\arctan(1) = \pi/4 = 0.78539\dots$$

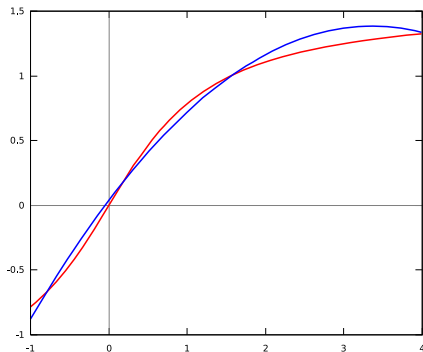
Why an approximation ?



$$\arctan(1) = 0.785 + \varepsilon, |\varepsilon| < 4e-4$$

- ▶ Let f be a real valued function : $f : \mathbb{R} \rightarrow \mathbb{R}$.
- ▶ The function may take irrational values : $f(x)$ is thus not exactly representable.
- ▶ We can only compute approximated values and hopefully bound the approximation error.

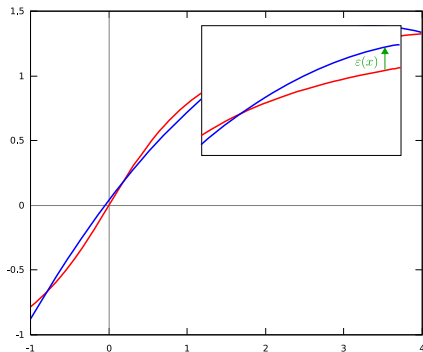
About the error of approximation



(n : degree of the polynomial)

- Consider a closed interval $[a, b]$. Replacing f by a polynomial p leads at each point x to :

About the error of approximation



(n : degree of the polynomial)

- ▶ Consider a closed interval $[a, b]$. Replacing f by a polynomial p leads at each point x to :

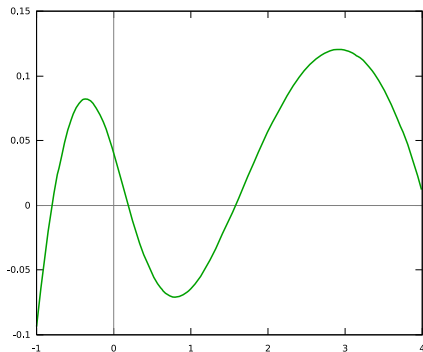
- ▶ an absolute error

$$\varepsilon(x) = f(x) - p(x);$$

- ▶ a relative error

$$\delta(x) = \varepsilon(x)/f(x).$$

About the error of approximation



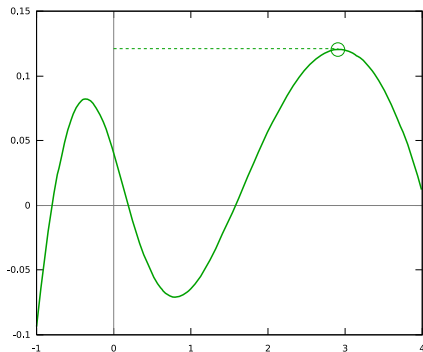
(n : degree of the polynomial)

- ▶ Consider a closed interval $[a, b]$. Replacing f by a polynomial p leads at each point x to :
 - ▶ an absolute error

$$\varepsilon(x) = f(x) - p(x);$$
 - ▶ a relative error

$$\delta(x) = \varepsilon(x)/f(x).$$
- ▶ The worst approximation is reached when $|\varepsilon(x)|$ has its maximal value.

About the error of approximation



(n : degree of the polynomial)

- ▶ Consider a closed interval $[a, b]$. Replacing f by a polynomial p leads at each point x to :
 - ▶ an absolute error

$$\varepsilon(x) = f(x) - p(x);$$
 - ▶ a relative error

$$\delta(x) = \varepsilon(x)/f(x).$$
- ▶ The worst approximation is reached when $|\varepsilon(x)|$ has its maximal value.

$$\|\varepsilon\|_{\infty} = \max_{x \in [a, b]} \{|\varepsilon(x)|\}$$

Focus on polynomial approximation

- ▶ The definition often gives a natural way to compute approximations of f . For instance : a power series and a formally computed bound on the error.

Focus on polynomial approximation

- ▶ The definition often gives a natural way to compute approximations of f . For instance : a power series and a formally computed bound on the error.
- ▶ Remark : a truncated power series is a polynomial
↔ especially nice to evaluate : it requires only additions and multiplications (fast on modern processors).

Focus on polynomial approximation

- ▶ The definition often gives a natural way to compute approximations of f . For instance : a power series and a formally computed bound on the error.
- ▶ Remark : a truncated power series is a polynomial
↔ especially nice to evaluate : it requires only additions and multiplications (fast on modern processors).
- ▶ Truncated power series are useful but. . .

Focus on polynomial approximation

- ▶ The definition often gives a natural way to compute approximations of f . For instance : a power series and a formally computed bound on the error.
- ▶ Remark : a truncated power series is a polynomial
↪ especially nice to evaluate : it requires only additions and multiplications (fast on modern processors).
- ▶ Truncated power series are useful but. . .
... usually inefficient in term of number of operations.
↪ $\exp(x)$ on $[-1; 2]$ with an absolute error ≤ 0.01 :
7 terms of the series / a degree 4 polynomial is sufficient.

Focus on polynomial approximation

- ▶ The definition often gives a natural way to compute approximations of f . For instance : a power series and a formally computed bound on the error.
- ▶ Remark : a truncated power series is a polynomial
↪ especially nice to evaluate : it requires only additions and multiplications (fast on modern processors).
- ▶ Truncated power series are useful but. . .
. . . usually inefficient in term of number of operations.
↪ $\exp(x)$ on $[-1; 2]$ with an absolute error ≤ 0.01 :
7 terms of the series / a degree **4** polynomial is sufficient.
- ▶ Natural question : what degree should have a polynomial to give a suitable approximation ?

Reminder of approximation theory

- ▶ Polynomial approximation theory has been deeply studied since the XIXth century.

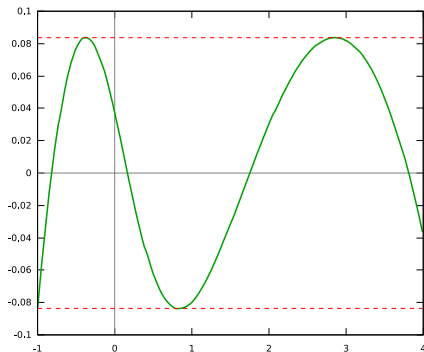
Reminder of approximation theory

- ▶ Polynomial approximation theory has been deeply studied since the XIXth century.
- ▶ Th. (Weierstrass) : the set $\mathbb{R}[X]$ is dense in $\mathcal{C}([a, b])$.
Bernstein gave an effective polynomial sequence.

Reminder of approximation theory

- ▶ Polynomial approximation theory has been deeply studied since the XIXth century.
- ▶ Th. (Weierstrass) : the set $\mathbb{R}[X]$ is dense in $\mathcal{C}([a, b])$.
Bernstein gave an effective polynomial sequence.
- ▶ Th. (Chebyshev) : given n and f there is a unique polynomial p of degree $\leq n$ minimizing $\|f - p\|_\infty$.

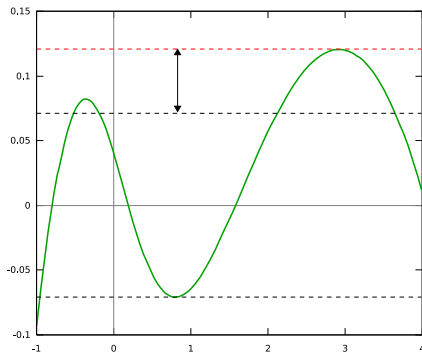
Reminder of approximation theory (2)



$n + 2$ oscillations

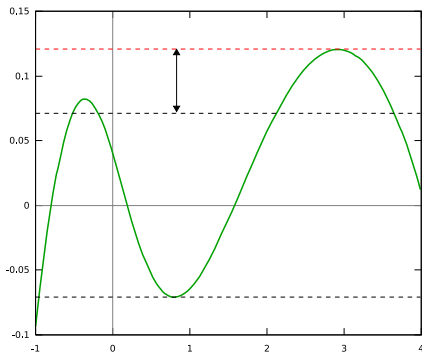
- ▶ Th. (Chebyshev) : characterization of the optimal error.

Reminder of approximation theory (2)



- ▶ Th. (Chebyshev) : characterization of the optimal error.
- ▶ Th. (La Vallée Poussin) : links the quality of an approximation with its error function.

Reminder of approximation theory (2)



- ▶ Th. (Chebyshev) : characterization of the optimal error.
- ▶ Th. (La Vallée Poussin) : links the quality of an approximation with its error function.
- ▶ Remez' algorithm : given n , computes the optimal polynomial of degree $\leq n$ (called **minimax**).

Representing real numbers in computers

- ▶ In general a real number is not finitely representable.
 - ↪ one has to choose a subset S and approximate the real line by the elements of S .

Representing real numbers in computers

- ▶ In general a real number is not finitely representable.
↪ one has to choose a subset S and approximate the real line by the elements of S .
- ▶ A usual choice : floating-point numbers (IEEE-754 standard).

Representing real numbers in computers

- ▶ In general a real number is not finitely representable.
↪ one has to choose a subset S and approximate the real line by the elements of S .
- ▶ A usual choice : floating-point numbers (IEEE-754 standard).
- ▶ A floating-point number with **radix β** and **precision t** is a number of the form

$$m \cdot \beta^e$$

where :

- ▶ $m \in \mathbb{Z}$ is the **mantissa** and is written with exactly t digits ;
- ▶ $e \in \mathbb{Z}$ is the **exponent**. It is usually bounded in a range $[e_{\min}, e_{\max}]$.

Representing real numbers in computers

- ▶ In general a real number is not finitely representable.
↪ one has to choose a subset S and approximate the real line by the elements of S .
- ▶ A usual choice : floating-point numbers (IEEE-754 standard).
- ▶ A floating-point number with **radix** β and **precision** t is a number of the form

$$m \cdot \beta^e$$

where :

- ▶ $m \in \mathbb{Z}$ is the **mantissa** and is written with exactly t digits ;
- ▶ $e \in \mathbb{Z}$ is the **exponent**. It is usually bounded in a range $[e_{\min}, e_{\max}]$.
- ▶ IEEE **double** format : $\beta = 2$, $t = 53$, and $e \in \llbracket -1074, 971 \rrbracket$.

Representing real numbers in computers

- ▶ In general a real number is not finitely representable.
↪ one has to choose a subset S and approximate the real line by the elements of S .
- ▶ A usual choice : floating-point numbers (IEEE-754 standard).
- ▶ A floating-point number with **radix β** and **precision t** is a number of the form

$$m \cdot \beta^e$$

where :

- ▶ $m \in \mathbb{Z}$ is the **mantissa** and is written with exactly t digits ;
- ▶ $e \in \mathbb{Z}$ is the **exponent**. It is usually bounded in a range $[e_{\min}, e_{\max}]$.
- ▶ IEEE **double** format : $\beta = 2$, $t = 53$, and $e \in \llbracket -1074, 971 \rrbracket$.
- ▶ From now on, we will assume that $[e_{\min}, e_{\max}] = [-\infty, +\infty]$.

Polynomials with floating-point coefficients

- ▶ Each coefficient of a polynomial is represented by a floating-point number.

Polynomials with floating-point coefficients

- ▶ Each coefficient of a polynomial is represented by a floating-point number.
- ▶ Naive method to obtain a polynomial approximation of f :
 - ▶ compute the real minimax p^* ;
 - ▶ replace each coefficient a_i of p^* by the nearest floating-point number \hat{a}_i ;
 - ▶ use $\hat{p} = \hat{a}_0 + \hat{a}_1 X + \cdots + \hat{a}_n X^n$.

Polynomials with floating-point coefficients

- ▶ Each coefficient of a polynomial is represented by a floating-point number.
- ▶ Naive method to obtain a polynomial approximation of f :
 - ▶ compute the real minimax p^* ;
 - ▶ replace each coefficient a_i of p^* by the nearest floating-point number \hat{a}_i ;
 - ▶ use $\hat{p} = \hat{a}_0 + \hat{a}_1 X + \dots + \hat{a}_n X^n$.
- ▶ \hat{p} may be far from being optimal.

Polynomials with floating-point coefficients

- ▶ Each coefficient of a polynomial is represented by a floating-point number.
- ▶ Naive method to obtain a polynomial approximation of f :
 - ▶ compute the real minimax p^* ;
 - ▶ replace each coefficient a_i of p^* by the nearest floating-point number \hat{a}_i ;
 - ▶ use $\hat{p} = \hat{a}_0 + \hat{a}_1 X + \dots + \hat{a}_n X^n$.
- ▶ \hat{p} may be far from being optimal.
- ▶ Example with $f(x) = \log_2(1 + 2^{-x})$, $n = 6$, on $[0; 1]$ with single precision coefficients (24 bits).

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

Previous works

- ▶ W. Kahan claims to have studied the question and proposed an efficient method. No published work, no draft.

Previous works

- ▶ W. Kahan claims to have studied the question and proposed an efficient method. No published work, no draft.
- ▶ D. Kodek has studied a similar problem in signal processing. Limited to small precision and degree (typically $t < 10$, $n < 20$).

Previous works

- ▶ W. Kahan claims to have studied the question and proposed an efficient method. No published work, no draft.
- ▶ D. Kodek has studied a similar problem in signal processing. Limited to small precision and degree (typically $t < 10$, $n < 20$).
- ▶ N. Brisebarre, J.-M. Muller and A. Tisserand have proposed an approach by linear programming (the implementation relies on P. Feautrier's tool PIP).

Method of Brisebarre, Muller and Tisserand

- ▶ Idea : they reduce the initial problem to the problem of finding the points with integer coordinates in a polytope of \mathbb{R}^{n+1} .

Method of Brisebarre, Muller and Tisserand

- ▶ Idea : they reduce the initial problem to the problem of finding the points with integer coordinates in a polytope of \mathbb{R}^{n+1} .
- ▶ This approach is certified. . .

Method of Brisebarre, Muller and Tisserand

- ▶ Idea : they reduce the initial problem to the problem of finding the points with integer coordinates in a polytope of \mathbb{R}^{n+1} .
- ▶ This approach is certified. . .
- ▶ . . . and flexible (may be used to find real minimax, constrained real minimax, polynomial with floating-point coefficients, odd polynomials, etc.).

Method of Brisebarre, Muller and Tisserand

- ▶ Idea : they reduce the initial problem to the problem of finding the points with integer coordinates in a polytope of \mathbb{R}^{n+1} .
- ▶ This approach is certified. . .
- ▶ . . . and flexible (may be used to find real minimax, constrained real minimax, polynomial with floating-point coefficients, odd polynomials, etc.).
- ▶ But :
 - ▶ its time is exponential ;

Method of Brisebarre, Muller and Tisserand

- ▶ Idea : they reduce the initial problem to the problem of finding the points with integer coordinates in a polytope of \mathbb{R}^{n+1} .
- ▶ This approach is certified. . .
- ▶ . . . and flexible (may be used to find real minimax, constrained real minimax, polynomial with floating-point coefficients, odd polynomials, etc.).
- ▶ But :
 - ▶ its time is exponential ;
 - ▶ it is very sensitive to some parameters.

Method of Brisebarre, Muller and Tisserand

- ▶ Idea : they reduce the initial problem to the problem of finding the points with integer coordinates in a polytope of \mathbb{R}^{n+1} .
- ▶ This approach is certified. . .
- ▶ . . . and flexible (may be used to find real minimax, constrained real minimax, polynomial with floating-point coefficients, odd polynomials, etc.).
- ▶ But :
 - ▶ its time is exponential ;
 - ▶ it is very sensitive to some parameters.
- ▶ We developed a new method :
 - ▶ fast (it is proven to run in polynomial time) ;
 - ▶ heuristic (there is no proof that the result is always tight) ;
 - ▶ with good practical results.

Formalization of the problem

- ▶ Problem : given n and a floating-point format, find (one of) the polynomial(s) p of degree $\leq n$ with floating-point coefficients minimizing $\|p - f\|_\infty$.

Formalization of the problem

- ▶ Problem : given n and a floating-point format, find (one of) the polynomial(s) p of degree $\leq n$ with floating-point coefficients minimizing $\|p - f\|_\infty$.
- ▶ Remark : the existence is still ensured. The unicity may be lost.

Formalization of the problem

- ▶ Problem : given n and a floating-point format, find (one of) the polynomial(s) p of degree $\leq n$ with floating-point coefficients minimizing $\|p - f\|_\infty$.
- ▶ Remark : the existence is still ensured. The unicity may be lost.
- ▶ A simplification : we may try to guess the value of each e_i (assuming that the coefficients of p and p^* have the same order of magnitude)
 - ↪ if e_i is correctly guessed, we are reduced to find $m_i \in \mathbb{Z}$ such that

$$\left\| f(x) - \sum_{i=0}^n m_i \cdot \beta^{e_i} x^i \right\|_\infty$$

is minimal.

Description of our method

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

$$m_0 \cdot \beta^{e_0} + m_1 \cdot \beta^{e_1} X + \dots + m_n \cdot \beta^{e_n} X^n$$

Description of our method

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

$$m_0 \cdot \beta^{e_0} + m_1 \cdot \beta^{e_1} X + \dots + m_n \cdot \beta^{e_n} X^n$$

- ▶ We use the idea of interpolation :

Description of our method

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

$$m_0 \cdot \beta^{e_0} + m_1 \cdot \beta^{e_1} X + \dots + m_n \cdot \beta^{e_n} X^n$$

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

Description of our method

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

$$m_0 \cdot \beta^{e_0} + m_1 \cdot \beta^{e_1} X + \dots + m_n \cdot \beta^{e_n} X^n$$

► We use the idea of interpolation :

- we choose $n + 1$ points x_0, \dots, x_n in $[a, b]$;
- we search m_0, \dots, m_n such that for all i

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

Description of our method

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

$$m_0 \cdot \beta^{e_0} + m_1 \cdot \beta^{e_1} X + \dots + m_n \cdot \beta^{e_n} X^n$$

► We use the idea of interpolation :

- we choose $n + 1$ points x_0, \dots, x_n in $[a, b]$;
- we search m_0, \dots, m_n such that for all i

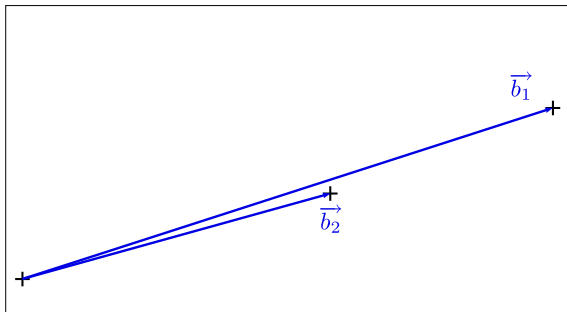
$$p(x_i) = m_0 \cdot \beta^{e_0} + m_1 \cdot \beta^{e_1} x_i + \dots + m_n \cdot \beta^{e_n} x_i^n \simeq f(x_i) \quad .$$

► Rewritten with vectors :

$$\underbrace{m_0 \begin{pmatrix} \beta^{e_0} \\ \beta^{e_0} \\ \vdots \\ \beta^{e_0} \end{pmatrix} + \dots + m_n \begin{pmatrix} \beta^{e_n} \cdot x_0^n \\ \beta^{e_n} \cdot x_1^n \\ \vdots \\ \beta^{e_n} \cdot x_n^n \end{pmatrix}}_{\Gamma \text{ of the form } \mathbb{Z}\vec{b}_0 + \mathbb{Z}\vec{b}_1 + \dots + \mathbb{Z}\vec{b}_n} \simeq \underbrace{\begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}}_{\vec{t} \in \mathbb{R}^{n+1}} \quad .$$

Notions about lattices

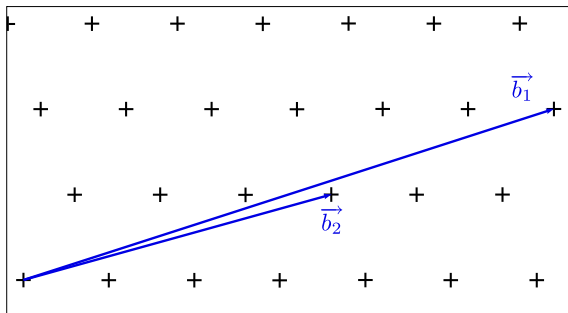
Let $(\vec{b}_1, \dots, \vec{b}_n)$ be a basis of a real vector space.



Notions about lattices

Let $(\vec{b}_1, \dots, \vec{b}_n)$ be a basis of a real vector space. The set of all integer combinations of the \vec{b}_i is called a **lattice** :

$$\Gamma = \mathbb{Z}\vec{b}_1 + \mathbb{Z}\vec{b}_2 + \dots + \mathbb{Z}\vec{b}_n \quad .$$

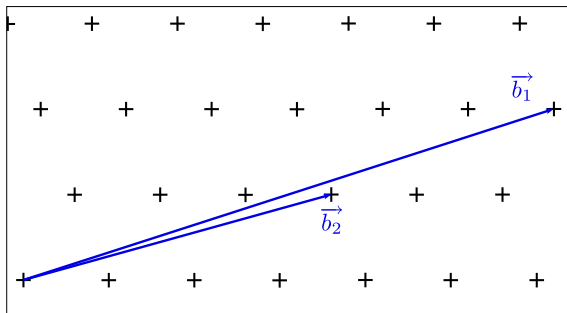


Notions about lattices

Let $(\vec{b}_1, \dots, \vec{b}_n)$ be a basis of a real vector space. The set of all integer combinations of the \vec{b}_i is called a **lattice** :

$$\Gamma = \mathbb{Z}\vec{b}_1 + \mathbb{Z}\vec{b}_2 + \dots + \mathbb{Z}\vec{b}_n \quad .$$

In general, a lattice has infinitely many bases.

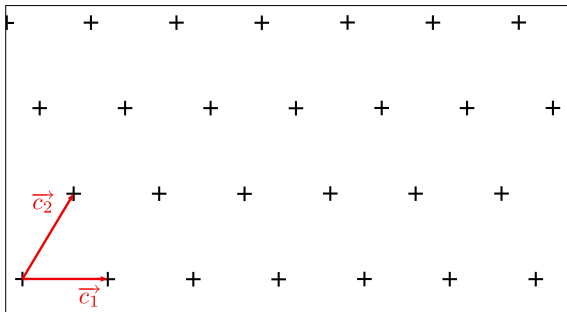


Notions about lattices

Let $(\vec{b}_1, \dots, \vec{b}_n)$ be a basis of a real vector space. The set of all integer combinations of the \vec{b}_i is called a **lattice** :

$$\Gamma = \mathbb{Z}\vec{b}_1 + \mathbb{Z}\vec{b}_2 + \dots + \mathbb{Z}\vec{b}_n .$$

In general, a lattice has infinitely many bases.

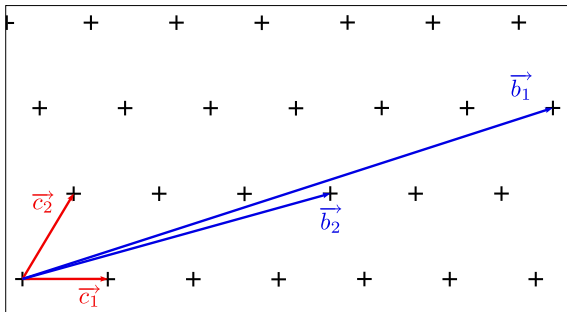


Notions about lattices

Let $(\vec{b}_1, \dots, \vec{b}_n)$ be a basis of a real vector space. The set of all integer combinations of the \vec{b}_i is called a **lattice** :

$$\Gamma = \mathbb{Z}\vec{b}_1 + \mathbb{Z}\vec{b}_2 + \dots + \mathbb{Z}\vec{b}_n \quad .$$

In general, a lattice has infinitely many bases.



Algorithmic problems

In the following we consider the euclidean norm on \mathbb{R}^n :

$$\|\vec{x}\|^2 = \sum_{i=1}^n x_i^2.$$

- ▶ Shortest vector problem (**SVP**).

Algorithmic problems

In the following we consider the euclidean norm on \mathbb{R}^n :

$$\|\vec{x}\|^2 = \sum_{i=1}^n x_i^2.$$

- ▶ Shortest vector problem (**SVP**).
 - ▶ Ajtai (1997) and Micciancio (1998) showed that SVP is NP-hard under probabilistic randomized reduction ; it is NP-hard to approximate SVP within a factor $\sqrt{2}$.

Algorithmic problems

In the following we consider the euclidean norm on \mathbb{R}^n :

$$\|\vec{x}\|^2 = \sum_{i=1}^n x_i^2.$$

- ▶ Shortest vector problem (**SVP**).
 - ▶ Ajtai (1997) and Micciancio (1998) showed that SVP is NP-hard under probabilistic randomized reduction ; it is NP-hard to approximate SVP within a factor $\sqrt{2}$.
 - ▶ There is no polynomial algorithm known to approximate SVP within a factor $f(n)$ where f is a polynomial.

Algorithmic problems

In the following we consider the euclidean norm on \mathbb{R}^n :

$$\|\vec{x}\|^2 = \sum_{i=1}^n x_i^2.$$

- ▶ Shortest vector problem (**SVP**).
 - ▶ Ajtai (1997) and Micciancio (1998) showed that SVP is NP-hard under probabilistic randomized reduction ; it is NP-hard to approximate SVP within a factor $\sqrt{2}$.
 - ▶ There is no polynomial algorithm known to approximate SVP within a factor $f(n)$ where f is a polynomial.
- ▶ Shortest basis problem (**SBP**).
 - ▶ Given a basis of a lattice L , find a basis (b_1, \dots, b_n) of L for which $\|b_1\| \cdot \|b_2\| \cdots \|b_n\|$ is minimal.

Algorithmic problems

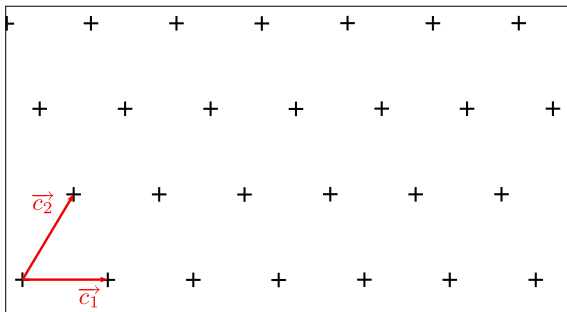
In the following we consider the euclidean norm on \mathbb{R}^n :

$$\|\vec{x}\|^2 = \sum_{i=1}^n x_i^2.$$

- ▶ Shortest vector problem (**SVP**).
 - ▶ Ajtai (1997) and Micciancio (1998) showed that SVP is NP-hard under probabilistic randomized reduction ; it is NP-hard to approximate SVP within a factor $\sqrt{2}$.
 - ▶ There is no polynomial algorithm known to approximate SVP within a factor $f(n)$ where f is a polynomial.
- ▶ Shortest basis problem (**SBP**).
 - ▶ Given a basis of a lattice L , find a basis (b_1, \dots, b_n) of L for which $\|b_1\| \cdot \|b_2\| \cdots \|b_n\|$ is minimal.
 - ▶ It is NP-hard.

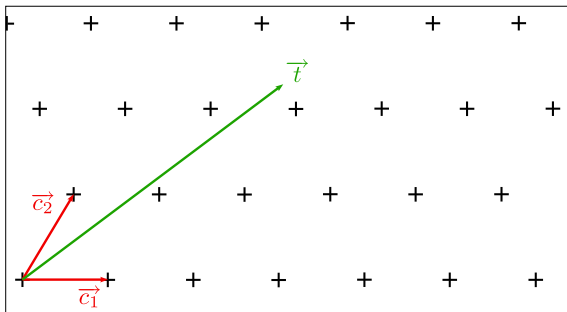
Algorithmic problems

- ▶ Closest vector problem (CVP).



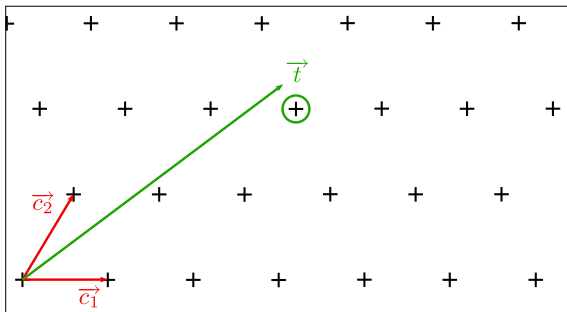
Algorithmic problems

- ▶ Closest vector problem (CVP).



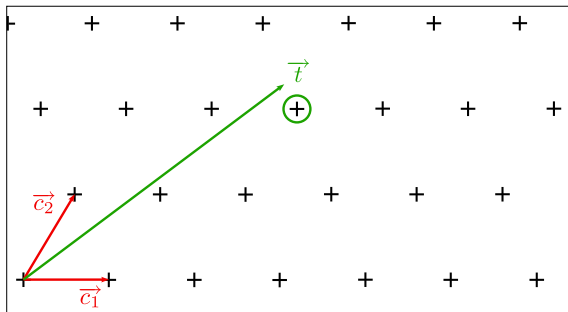
Algorithmic problems

- ▶ Closest vector problem (CVP).



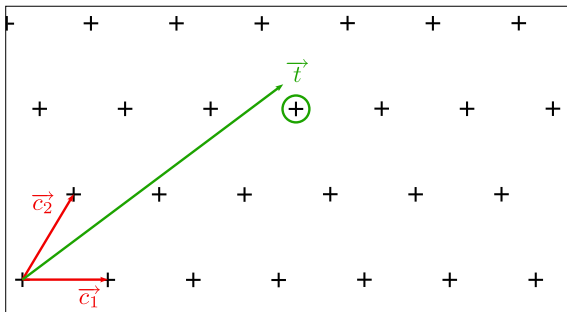
Algorithmic problems

- ▶ Closest vector problem (CVP).
 - ▶ Emde Boas (1981) : CVP is NP-hard.



Algorithmic problems

- ▶ Closest vector problem (**CVP**).
 - ▶ Emde Boas (1981) : CVP is NP-hard.
 - ▶ Goldreich and al. : CVP is not easier than SVP.



LLL algorithm

- ▶ Algorithm developed by A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász.

Factoring Polynomials with Rational Coefficients,
Math. Annalen **261**, 515-534, 1982.

LLL algorithm

- ▶ Algorithm developed by A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász.
Factoring Polynomials with Rational Coefficients,
Math. Annalen **261**, 515-534, 1982.
- ▶ Given a basis (b_1, \dots, b_n) of a lattice, the LLL algorithm gives a basis (c_1, \dots, c_n) composed of pretty short vectors.

LLL algorithm

- ▶ Algorithm developed by A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász.
Factoring Polynomials with Rational Coefficients,
Math. Annalen **261**, 515-534, 1982.
- ▶ Given a basis (b_1, \dots, b_n) of a lattice, the LLL algorithm gives a basis (c_1, \dots, c_n) composed of pretty short vectors.
 $\hookrightarrow \|c_1\| \leq 2^{(n-1)/2} \lambda_1(L)$ where $\lambda_1(L)$ denotes the norm of a shortest nonzero vector of L .
- ▶ LLL terminates in at most $O(n^6 \ln^3 B)$ operations with $B = \max \|b_i\|^2$.

LLL algorithm

- ▶ Algorithm developed by A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász.
Factoring Polynomials with Rational Coefficients,
Math. Annalen **261**, 515-534, 1982.
- ▶ Given a basis (b_1, \dots, b_n) of a lattice, the LLL algorithm gives a basis (c_1, \dots, c_n) composed of pretty short vectors.
 $\hookrightarrow \|c_1\| \leq 2^{(n-1)/2} \lambda_1(L)$ where $\lambda_1(L)$ denotes the norm of a shortest nonzero vector of L .
- ▶ LLL terminates in at most $O(n^6 \ln^3 B)$ operations with $B = \max \|b_i\|^2$.
- ▶ Very good practical results compared to the theoretical bounds.

LLL reduction

- ▶ Gram-Schmidt orthogonalization : to any basis (b_1, \dots, b_n) of a vector space is associated an orthogonal basis (b_1^*, \dots, b_n^*) such that $\text{Span}(b_1, \dots, b_j) = \text{Span}(b_1^*, \dots, b_j^*)$ for all j .

LLL reduction

- ▶ Gram-Schmidt orthogonalization : to any basis (b_1, \dots, b_n) of a vector space is associated an orthogonal basis (b_1^*, \dots, b_n^*) such that $\text{Span}(b_1, \dots, b_j) = \text{Span}(b_1^*, \dots, b_j^*)$ for all j .
Remark : one may choose it so that $b_1 = b_1^*$.

LLL reduction

- ▶ Gram-Schmidt orthogonalization : to any basis (b_1, \dots, b_n) of a vector space is associated an orthogonal basis (b_1^*, \dots, b_n^*) such that $\text{Span}(b_1, \dots, b_j) = \text{Span}(b_1^*, \dots, b_j^*)$ for all j .
Remark : one may choose it so that $b_1 = b_1^*$.
- ▶ Prop. : if (b_1, \dots, b_n) is the basis of a lattice L , $\lambda_1(L) \geq \min \|b_j^*\|$.

LLL reduction

- ▶ Gram-Schmidt orthogonalization : to any basis (b_1, \dots, b_n) of a vector space is associated an orthogonal basis (b_1^*, \dots, b_n^*) such that $\text{Span}(b_1, \dots, b_j) = \text{Span}(b_1^*, \dots, b_j^*)$ for all j .
Remark : one may choose it so that $b_1 = b_1^*$.
- ▶ Prop. : if (b_1, \dots, b_n) is the basis of a lattice L , $\lambda_1(L) \geq \min \|b_j^*\|$.
- ▶ Idea of LLL algorithm : control the Gram-Schmidt basis to make $b_1^* = b_1$ minimal among the vectors of the orthogonal basis.

LLL reduction

- ▶ Gram-Schmidt orthogonalization : to any basis (b_1, \dots, b_n) of a vector space is associated an orthogonal basis (b_1^*, \dots, b_n^*) such that $\text{Span}(b_1, \dots, b_j) = \text{Span}(b_1^*, \dots, b_j^*)$ for all j .
Remark : one may choose it so that $b_1 = b_1^*$.
- ▶ Prop. : if (b_1, \dots, b_n) is the basis of a lattice L , $\lambda_1(L) \geq \min \|b_j^*\|$.
- ▶ Idea of LLL algorithm : control the Gram-Schmidt basis to make $b_1^* = b_1$ minimal among the vectors of the orthogonal basis.
- ▶ Babai's algorithm uses the LLL algorithm to solve an approximation of CVP.

A concrete case

- ▶ Example coming from a collaboration with John Harrison from Intel.

A concrete case

- ▶ Example coming from a collaboration with John Harrison from Intel.
- ▶ He asked for a polynomial minimizing the absolute error
 - ▶ approximating $f : x \mapsto \frac{2^x - 1}{x}$
 - ▶ on $[-1/16, 1/16]$
 - ▶ with a degree 9 polynomial.

A concrete case

- ▶ Example coming from a collaboration with John Harrison from Intel.
- ▶ He asked for a polynomial minimizing the absolute error
 - ▶ approximating $f : x \mapsto \frac{2^x - 1}{x}$
 - ▶ on $[-1/16, 1/16]$
 - ▶ with a degree 9 polynomial.
 - ▶ a degree 0 coefficient of the form : $a_{0h} + a_{0l}$ where a_{0h} and a_{0l} are double extended numbers
 - ▶ other coefficients are double extended numbers.
- ▶ A double extended number has **64 bits of mantissa**.

A concrete case

- ▶ Example coming from a collaboration with John Harrison from Intel.
- ▶ He asked for a polynomial minimizing the absolute error
 - ▶ approximating $f : x \mapsto \frac{2^x - 1}{x}$
 - ▶ on $[-1/16, 1/16]$
 - ▶ with a degree 9 polynomial.
 - ▶ a degree 0 coefficient of the form $a_{0h} + a_{0l}$ where a_{0h} and a_{0l} are double extended numbers
 - ▶ other coefficients are double extended numbers.
- ▶ A double extended number has **64 bits of mantissa**.
- ▶ He actually wants to have approximately 74 correct bits.
(i.e. $\varepsilon \simeq 5.30e-23$)

First try

Target	Degree 8 minimax	Degree 9 minimax
$5.30e-23$	$40.1e-23$	$0.07897e-23$

↔ degree 9 should be a good choice.

First try

Target	Degree 8 minimax	Degree 9 minimax
$5.30e-23$	$40.1e-23$	$0.07897e-23$

↔ degree 9 should be a good choice.

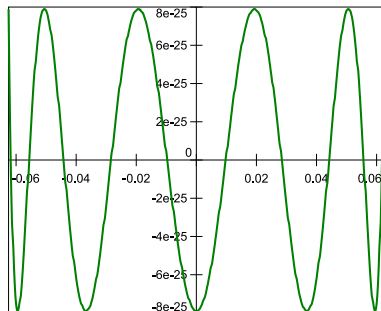
- ▶ How to choose the points ?

First try

Target	Degree 8 minimax	Degree 9 minimax
$5.30e-23$	$40.1e-23$	$0.07897e-23$

↪ degree 9 should be a good choice.

► How to choose the points ?



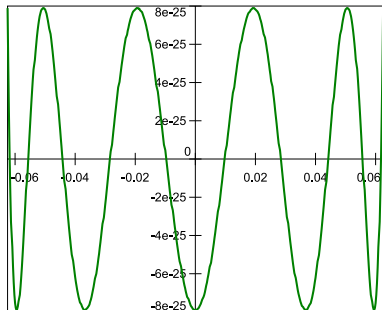
► We need $n + 1$ points.

First try

Target	Degree 8 minimax	Degree 9 minimax
$5.30e-23$	$40.1e-23$	$0.07897e-23$

↪ degree 9 should be a good choice.

► How to choose the points ?



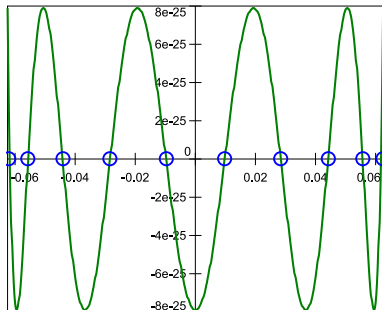
- We need $n + 1$ points.
- They should correspond to the interpolation intuition.

First try

Target	Degree 8 minimax	Degree 9 minimax
$5.30e-23$	$40.1e-23$	$0.07897e-23$

↪ degree 9 should be a good choice.

► 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.

First try : results

Target	Degree 9 minimax	our polynomial p_1	naive method
$5.30e-23$	$0.07897e-23$	$5.32e-23$	$40.35e-23$

First try : results

Target	Degree 9 minimax	our polynomial p_1	naive method
$5.30e-23$	$0.07897e-23$	$5.32e-23$	$40.35e-23$

↔ pretty good but...

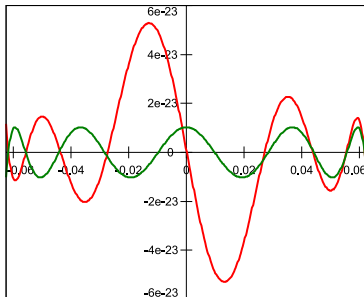
- ▶ Our polynomial does not respect the interpolation constraint.

First try : results

Target	Degree 9 minimax	our polynomial p_1	naive method
$5.30e-23$	$0.07897e-23$	$5.32e-23$	$40.35e-23$

↪ pretty good but...

- ▶ Our polynomial does not respect the interpolation constraint.



- ▶ degree 1 coefficient of p_1 :

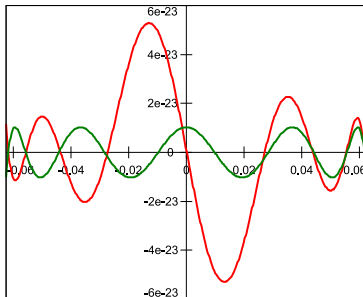
$$a_1 = o(\log(2)^2/2)$$

First try : results

Target	Degree 9 minimax	our polynomial p_1	naive method
$5.30e-23$	$0.07897e-23$	$5.32e-23$	$40.35e-23$

↪ pretty good but...

- ▶ Our polynomial does not respect the interpolation constraint.



- ▶ degree 1 coefficient of p_1 :

$$a_1 = o(\log(2)^2/2)$$

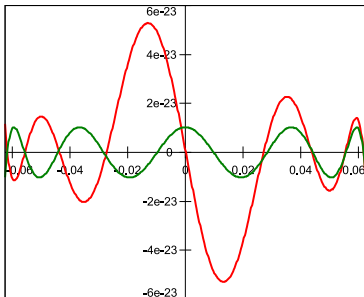
→ the slope at 0 is very constrained.

First try : results

Target	Degree 9 minimax	our polynomial p_1	naive method
$5.30e-23$	$0.07897e-23$	$5.32e-23$	$40.35e-23$

↪ pretty good but...

- ▶ Our polynomial does not respect the interpolation constraint.



- ▶ degree 1 coefficient of p_1 :

$$a_1 = o(\log(2)^2/2)$$

→ the slope at 0 is very constrained.

- ▶ we have to take it into account.

Second try

- ▶ The polytope approach confirms that a_1 has a constrained value.

Second try

- ▶ The polytope approach confirms that a_1 has a constrained value.
- ▶ We compute the best real polynomial of the form $a_0 + a_2 X^2 + \dots + a_9 X^9$ approximating $f - a_1 X$.

Second try

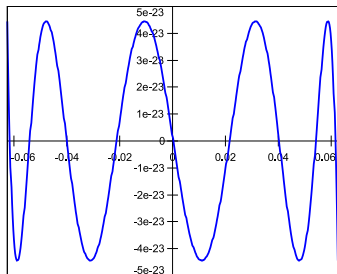
- ▶ The polytope approach confirms that a_1 has a constrained value.
- ▶ We compute the best real polynomial of the form $a_0 + a_2 X^2 + \dots + a_9 X^9$ approximating $f - a_1 X$.

Degree 9 minimax	Constrained optimum	p_1
0.07897e-23	4.44e-23	5.32e-23

Second try

- ▶ The polytope approach confirms that a_1 has a constrained value.
- ▶ We compute the best real polynomial of the form $a_0 + a_2 X^2 + \dots + a_9 X^9$ approximating $f - a_1 X$.

Degree 9 minimax	Constrained optimum	p_1
0.07897e-23	4.44e-23	5.32e-23

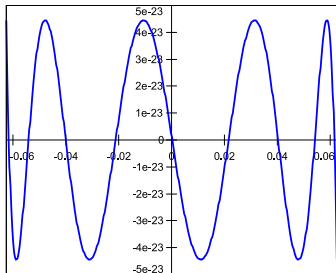


- ▶ We have only 9 points, but now only 9 unknowns : it is OK.

Second try

- ▶ The polytope approach confirms that a_1 has a constrained value.
- ▶ We compute the best real polynomial of the form $a_0 + a_2 X^2 + \dots + a_9 X^9$ approximating $f - a_1 X$.

Degree 9 minimax	Constrained optimum	p_1
0.07897e-23	4.44e-23	5.32e-23



- ▶ We have only 9 points, but now only 9 unknowns : it is OK.
- ▶ This time, our polynomial p_2 gives an error of $4.44e-23$ and is practically optimal.

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 not proven, but works well in practice and gives certified results with help of the polytope approach.

Conclusion

- ▶ We have developed an algorithm to find very good polynomial approximants with floating-point coefficients.
- ▶ The algorithm is not proven, but works well in practice and gives certified results with help of the polytope approach.
- ▶ The algorithm is flexible : each coefficient may use a different floating-point format, one may search polynomial with additional constraints.

Future work

- ▶ We need a good algorithm to find constrained minimax.

Future work

- ▶ We need a good algorithm to find constrained minimax.
↳ Remez' algorithm is not sufficient.

Future work

- ▶ We need a good algorithm to find constrained minimax.
↳ Remez' algorithm is not sufficient.
- ▶ Use similar methods to find other approximants :

Future work

- ▶ We need a good algorithm to find constrained minimax.
↳ Remez' algorithm is not sufficient.
- ▶ Use similar methods to find other approximants :
 - ▶ rational fractions;

Future work

- ▶ We need a good algorithm to find constrained minimax.
↔ Remez' algorithm is not sufficient.
- ▶ Use similar methods to find other approximants :
 - ▶ rational fractions ;
 - ▶ sums of cosines.