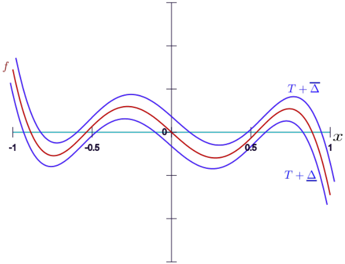
Some validated symbolic-numeric approximation algorithms

M. Joldes
joint works with D. Arzelier, F. Bréhard, N. Brisebarre, J.-M. Muller, J.-B. Lasserre, A. Rondepierre, B. Salvy

LAAS-CNRS, Toulouse, France



Winter Workshop on Dynamics, Topology and Computations, BEDLEWO, Poland

January 28 - February 3, 2018

Context

- Numerical Computing: floating-point arithmetic

→ High Performance Computing (MultiCores, GPUs, FPGAs):

  - **Fast numerical solutions**: global optimization, systems of differential equations, integration
  - Usually, solutions lack certification of the output accuracy

Context

- Numerical Computing: floating-point arithmetic

→ High Performance Computing (MultiCores, GPUs, FPGAs):

- **Fast numerical solutions**: global optimization, systems of differential equations, integration
- Usually, solutions lack certification of the output accuracy

A catastrophic cancellation example:
Evaluate

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

for $a = 77617.0$, $b = 33096.0$ (Rump '88)
Results of C program, gcc, Linux:
1.172603940053178 in binary64;
$1.17260394005317863185883490452018 38$ in binary128. Exact result is $-0.827396\ldots$.

Context

- Numerical Computing: floating-point arithmetic

$\rightarrow$ High Performance Computing (MultiCores, GPUs, FPGAs):

- **Fast numerical solutions**: global optimization, systems of differential equations, integration
- Usually, solutions lack certification of the output accuracy

A catastrophic cancellation example:
Evaluate

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

for $a = 77617.0$, $b = 33096.0$ (Rump '88)
Results of C program, gcc, Linux:
1.1726039400531787 in binary64;
$1.17260394005317863185883490452018380$ in binary128. Exact result is $-0.827396\ldots$.

$\rightarrow$ Computer Algebra Systems (eg. Maple):

- **Exact solution**, e.g. $-\frac{54767}{66192}$

Context

- Numerical Computing: floating-point arithmetic

→ High Performance Computing (MultiCores, GPUs, FPGAs):

- **Fast numerical solutions**: global optimization, systems of differential equations, integration
- Usually, solutions lack certification of the output accuracy

A catastrophic cancellation example:
Evaluate

$$\underbrace{(333.75 - a^2)b^6 + a^2\left(11a^2b^2 - 121b^4 - 2\right) + 5.5b^8}_{5.5b^8 - 2 - 5.5b^8 \rightsquigarrow \text{ eval to 0 by cancellation}} + \frac{a}{2b}$$

for $a = 77617.0$, $b = 33096.0$ (Rump '88)
Results of C program, gcc, Linux:
1.172603940053178 in binary64;
$1.17260394005317863185883490452018$ in binary128. Exact result is $-0.827396\ldots$.

→ Computer Algebra Systems (eg. Maple):

- **Exact solution**, e.g. $-\frac{54767}{66192}$

Constrained minimax polynomial approximation

Find $c_2, c_3 \in \mathbb{Z}$ such that

$$\max_{-2^{-12} \leq x \leq 2^{-12}} \left| \exp x - \left(1 + x + \frac{c_2}{2^{53}} x^2 + \frac{c_3}{2^{53}} x^3 \right) \right|$$

is minimal.

Constrained minimax polynomial approximation

Find $c_2, c_3 \in \mathbb{Z}$ such that

$$\max_{-2^{-12} \leq x \leq 2^{-12}} \left| \exp x - \left(1 + x + \frac{c_2}{2^{53}} x^2 + \frac{c_3}{2^{53}} x^3\right) \right|$$

is minimal.

Best truncated polynomial:

$$p^{\star}(x) = 1 + x + \frac{4503599645901977}{2^{53}}\, x^2 + \frac{4503599645901977}{2^{52}}\, x^3$$

Constrained minimax polynomial approximation

Find $c_2, c_3 \in \mathbb{Z}$ such that

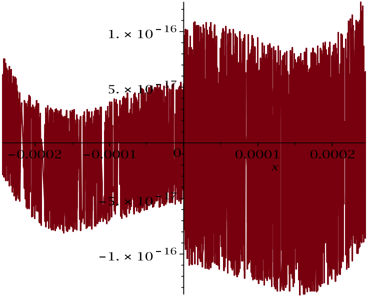
$$\max_{-2^{-12} \le x \le 2^{-12}} \left| \exp x - \left(1 + x + \frac{c_2}{2^{53}} x^2 + \frac{c_3}{2^{53}} x^3\right) \right|$$

is minimal.

Best truncated polynomial:

$$p^\star(x) = 1 + x + \frac{4503599645901977}{2^{53}} x^2 + \frac{4503599645901977}{2^{52}} x^3$$

Approx error $\varepsilon(x) := \exp x - p^\star(x)$ is (with Maple, 16 digits) :



Constrained minimax polynomial approximation

Find $c_2, c_3 \in \mathbb{Z}$ such that

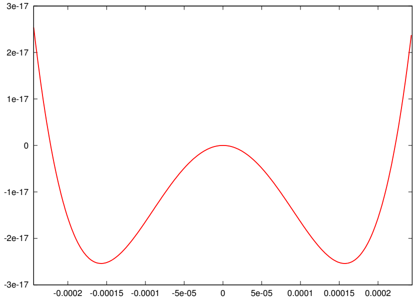
$$\max_{-2^{-12} \leq x \leq 2^{-12}} \left| \exp x - \left(1 + x + \frac{c_2}{2^{53}} x^2 + \frac{c_3}{2^{53}} x^3\right) \right|$$

is minimal.

Best truncated polynomial:

$$p^\star(x) = 1 + x + \frac{4503599645901977}{2^{53}}\, x^2 + \frac{4503599645901977}{2^{52}}\, x^3$$

Approx error $\varepsilon(x) := \exp x - p^\star(x)$ is (with Sollya):



Constrained minimax polynomial approximation

Find $c_2, c_3 \in \mathbb{Z}$ such that

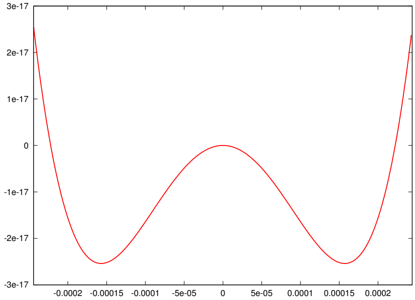
$$\max_{-2^{-12} \leq x \leq 2^{-12}} \left| \exp x - \left(1 + x + \frac{c_2}{2^{53}} x^2 + \frac{c_3}{2^{53}} x^3 \right) \right|$$

is minimal.

Best truncated polynomial:

$$p^\star(x) = 1 + x + \frac{4503599645901977}{2^{53}} x^2 + \frac{4503599645901977}{2^{52}} x^3$$

Approx error $\varepsilon(x) := \exp x - p^\star(x)$ is (with Sollya):



Prove that:

$$||\varepsilon||_{[-2^{-12};2^{-12}]} := \max_{-2^{-12} \leq x \leq 2^{-12}} |\varepsilon(x)| \leq 2.58 \cdot 10^{-17}$$

$\simeq 54$ bits accuracy.

Taylor series: $\exp = \sum \frac{1}{n!} x^n$

Recurrence for coefficients:

$$u(n+1) = \frac{u(n)}{n+1}$$

$u(0) = 1$	$1/0! = 1$
$u(1) = 1$	$1/1! = 1$
$u(2) = 0.5$	$1/2! = 0.5$
$\vdots$	$\vdots$
$u(50) \approx 3.28 \cdot 10^{-65}$	$1/50! \approx 3.28 \cdot 10^{-65}$

Taylor series: $\exp = \sum \frac{1}{n!} x^n$

Recurrence for coefficients:

$$u(n+1) = \frac{u(n)}{n+1}$$

$u(0) = 1$	$1/0! = 1$
$u(1) = 1$	$1/1! = 1$
$u(2) = 0.5$	$1/2! = 0.5$
$\vdots$	$\vdots$
$u(50) \approx 3.28 \cdot 10^{-65}$	$1/50! \approx 3.28 \cdot 10^{-65}$

2nd Case Study: Chebyshev Series of D-finite Functions

Taylor series: $\exp = \sum \frac{1}{n!} x^n$

Recurrence for coefficients:

$$u(n+1) = \frac{u(n)}{n+1}$$

$u(0) = 1$	$1/0! = 1$
$u(1) = 1$	$1/1! = 1$
$u(2) = 0.5$	$1/2! = 0.5$
$\vdots$	$\vdots$
$u(50) \approx 3.28 \cdot 10^{-65}$	$1/50! \approx 3.28 \cdot 10^{-65}$

Chebyshev series: $\exp = \sum I_n(1) T_n(x)$

Recurrence for coefficients:

$$u(n+1) = -2nu(n) + u(n-1)$$

$u(0) = 1.266$	$I_0(1) \approx 1.266$
$u(1) = 0.565$	$I_1(1) \approx 0.565$
$u(2) \approx 0.136$	$I_2(1) \approx 0.136$
$\vdots$	$\vdots$

2nd Case Study: Chebyshev Series of D-finite Functions

Taylor series: $\exp = \sum \frac{1}{n!} x^n$

Recurrence for coefficients:

$$u(n+1) = \frac{u(n)}{n+1}$$

$u(0) = 1$	$1/0! = 1$
$u(1) = 1$	$1/1! = 1$
$u(2) = 0.5$	$1/2! = 0.5$
$\vdots$	$\vdots$
$u(50) \approx 3.28 \cdot 10^{-65}$	$1/50! \approx 3.28 \cdot 10^{-65}$

Chebyshev series: $\exp = \sum I_n(1) T_n(x)$

Recurrence for coefficients:

$$u(n+1) = -2nu(n) + u(n-1)$$

$u(0) = 1.266$	$I_0(1) \approx 1.266$
$u(1) = 0.565$	$I_1(1) \approx 0.565$
$u(2) \approx 0.136$	$I_2(1) \approx 0.136$
$\vdots$	$\vdots$
$u(50) \approx 4.450 \cdot 10^{67}$	$I_{50}(1) \approx 2.934 \cdot 10^{-80}$

Taylor series: $\exp = \sum \frac{1}{n!} x^n$

Recurrence for coefficients:

$$u(n+1) = \frac{u(n)}{n+1}$$

$u(0) = 1$	$1/0! = 1$
$u(1) = 1$	$1/1! = 1$
$u(2) = 0.5$	$1/2! = 0.5$
$\vdots$	$\vdots$
$u(50) \approx 3.28 \cdot 10^{-65}$	$1/50! \approx 3.28 \cdot 10^{-65}$

Chebyshev series: $\exp = \sum I_n(1) T_n(x)$

Recurrence for coefficients:
$u(n+1) = -2nu(n) + u(n-1)$

$u(0) = 1.266$	$I_0(1) \approx 1.266$
$u(1) = 0.565$	$I_1(1) \approx 0.565$
$u(2) \approx 0.136$	$I_2(1) \approx 0.136$
$\vdots$	$\vdots$
$u(50) \approx 4.450 \cdot 10^{67}$	$I_{50}(1) \approx 2.934 \cdot 10^{-80}$

More subtle cause:

Convergent and Divergent Solutions of the Recurrence $u(n+1) = -2nu(n) + u(n-1)$:
If $u(n)$ is solution, then there exists another solution $v(n) \sim \frac{1}{u(n)}$

3rd Case Study: Cancellation in finite precision power series evaluation

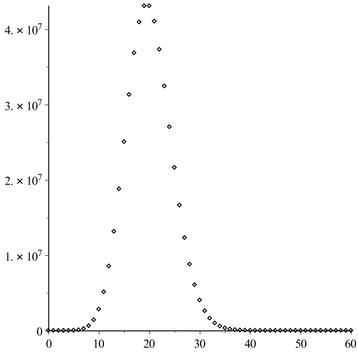
Example: $\exp(-x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^i}{i!}$

$$\exp(-20) = 1 - 20 \ldots + 1.66 \cdot 10^7 - 1.23 \cdot 10^7 + \ldots + 1.19 \cdot 10^{-8} - 3.45 \cdot 10^{-9} \ldots$$

Example: $\exp(-x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^i}{i!}$

$$\exp(-20) = 1 - 20 \ldots + 1.66 \cdot 10^7 - 1.23 \cdot 10^7 + \ldots + 1.19 \cdot 10^{-8} - 3.45 \cdot 10^{-9} \ldots$$

Values of $\left|\frac{(-1)^i 20^i}{i!}\right|$, compared to $\exp(-20) \simeq 2.06 \cdot 10^{-9}$:

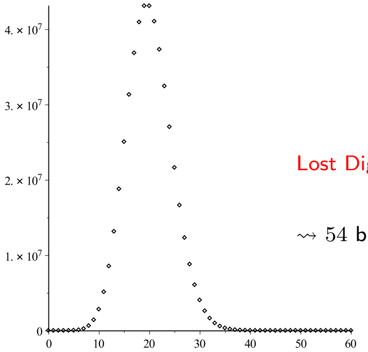


3rd Case Study: Cancellation in finite precision power series evaluation

Example: $\exp(-x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^i}{i!}$

$$\exp(-20) = 1 - 20 \ldots + 1.66 \cdot 10^7 - 1.23 \cdot 10^7 + \ldots + 1.19 \cdot 10^{-8} - 3.45 \cdot 10^{-9} \ldots$$

Values of $\left|\frac{(-1)^i 20^i}{i!}\right|$, compared to $\exp(-20) \simeq 2.06 \cdot 10^{-9}$:



Lost Digits: $\simeq \log \frac{\max_i \frac{20^i}{i!}}{\exp(-20)}$

$\rightsquigarrow$ 54 bits lost, hence binary64 result: $0.01583705682\ldots$

Safety-critical space applications

- 2009, Feb. 10: collision between Iridium 33 and Cosmos 2251, although predicted minimum distance of close approach was of 584m.

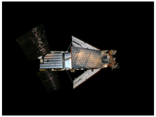


Figure: Animation of Iridium 33 and Kosmos 2251's collision; **GNU Free Documentation**, **Wikipedia**

- Collision probabilities estimated by reliable and efficient integral computations...

Safety-critical space applications

- 2009, Feb. 10: collision between Iridium 33 and Cosmos 2251, although predicted minimum distance of close approach was of 584m.

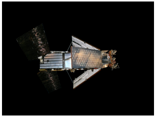


Figure: Animation of Iridium 33 and Kosmos 2251's collision; **GNU Free Documentation**, **Wikipedia**

- Collision probabilities estimated by reliable and efficient integral computations...

- 2009, Feb. 10: collision between Iridium 33 and Cosmos 2251, although predicted minimum distance of close approach was of 584m.

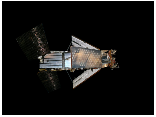


Figure: Animation of Iridium 33 and Kosmos 2251's collision; **GNU Free Documentation**, **Wikipedia**

- Collision probabilities estimated by reliable and efficient integral computations...

Computational methods (ultimate efficiency required) are a basic building brick

(courtesy 9gag.com)

Floating point (FP) Arithmetic I

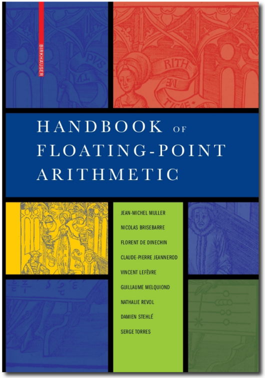
A real number is approximated in machine by a rational x:

$$x = (-1)^s \times m \times \beta^e$$

- β is the radix (usually $\beta = 2$)
- s is a sign bit
- m is the mantissa, a rational number of n_m digits in radix β:

$$m = d_0, d_1 d_2 ... d_{n_m - 1}$$

- e is the exponent, a signed integer on n_e bits



IEEE 754-2008 standard

Most common formats

- Single (*binary32*) precision format ($p = 24$):

1	8	23
s	e	m

- Double (*binary64*) precision format ($p = 53$):

1	11	52
s	e	m

$\longrightarrow$ Implicit bit that is not stored.

IEEE 754-2008 standard

Most common formats

- Single (*binary32*) precision format ($p = 24$):

1	8	23
s	e	m

- Double (*binary64*) precision format ($p = 53$):

1	11	52
s	e	m

$\longrightarrow$ Implicit bit that is not stored.

Rounding modes

- 4 rounding modes: RD, RU, RZ, RN
- Correct rounding for: $+, -, \times, \div, \sqrt{}$ (return what we would get by infinitely precise operations followed by rounding).
- Portability, determinism.

Multiple vs. standard precision

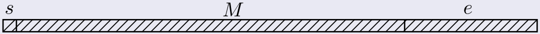
Standard precision $\rightsquigarrow$ hardware $\rightsquigarrow$ fast
Multiple precision $\rightsquigarrow$ software $\rightsquigarrow$ 100x slower (typically)

Multiple vs. standard precision

Standard precision $\rightsquigarrow$ hardware $\rightsquigarrow$ fast
Multiple precision $\rightsquigarrow$ software $\rightsquigarrow$ 100x slower (typically)

Two ways of representing numbers in extended precision

- *multiple-digit representation* - a number is represented by a sequence of digits coupled with a single exponent (Ex. GNU MPFR, ARPREC);

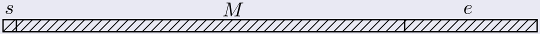


Multiple vs. standard precision

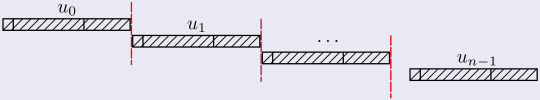
Standard precision $\rightsquigarrow$ hardware $\rightsquigarrow$ fast
Multiple precision $\rightsquigarrow$ software $\rightsquigarrow$ 100x slower (typically)

Two ways of representing numbers in extended precision

- *multiple-digit representation* - a number is represented by a sequence of digits coupled with a single exponent (Ex. GNU MPFR, ARPREC);



- *multiple-term representation* - a number is expressed as the unevaluated sum of several FP numbers (also called a FP *expansion*) (Ex. QD, CAMPARY).



Example: π in double-double

$$p_0 = 11.001001000011111101101010100010001000010110100011000_2,$$

and

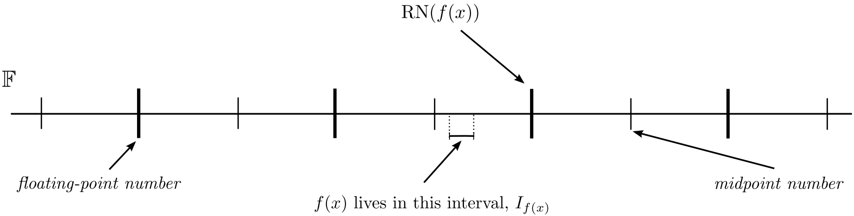
$$p_1 = 1.0001101001100010011000110011000101000101110000000111_2 \times 2^{-53}.$$

$p_0 + p_1 \leftrightarrow 107$ bits FP approx.

- ✓ Since 1985, IEEE-754 standard for FP arithmetic requests correct rounding for : $+, -, \times, \div, \sqrt{}$.

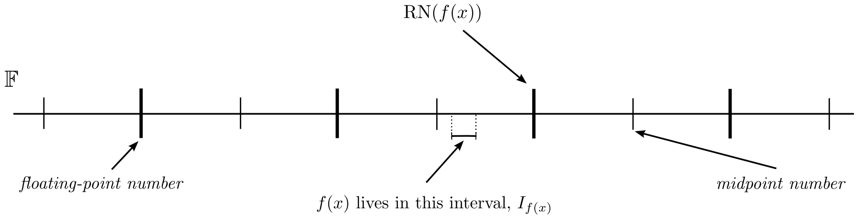
Floating point (FP) Arithmetic II

- ✓ Since 1985, IEEE-754 standard for FP arithmetic requests correct rounding for : $+, -, \times, \div, \sqrt{}$.
- ✓ Correct Rounding: An operation whose entries are FP numbers must return what we would get by infinitely precise operation followed by rounding.



Floating point (FP) Arithmetic II

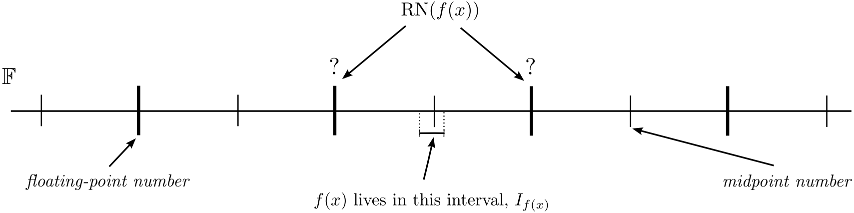
- ✓ Since 1985, IEEE-754 standard for FP arithmetic requests correct rounding for : $+, -, \times, \div, \sqrt{}$.
- ✓ Correct Rounding: An operation whose entries are FP numbers must return what we would get by infinitely precise operation followed by rounding.



- What about standard functions ($\sin$, $\cos$, $\log$, etc.)?

Floating point (FP) Arithmetic II

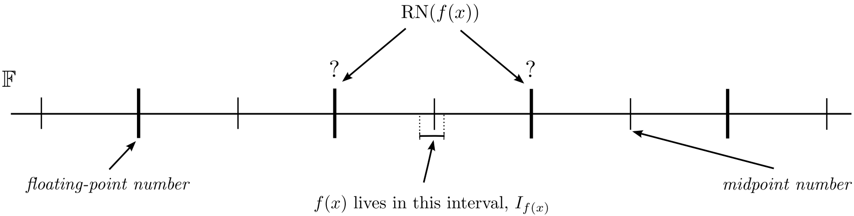
- ✓ Since 1985, IEEE-754 standard for FP arithmetic requests correct rounding for : $+, -, \times, \div, \sqrt{}$.
- ✓ Correct Rounding: An operation whose entries are FP numbers must return what we would get by infinitely precise operation followed by rounding.



- What about standard functions (sin, cos, log, etc.)?
  - Most Mathematical Libraries (libms) do not provide correctly rounded functions, although IEEE-754-2008 recommends it.

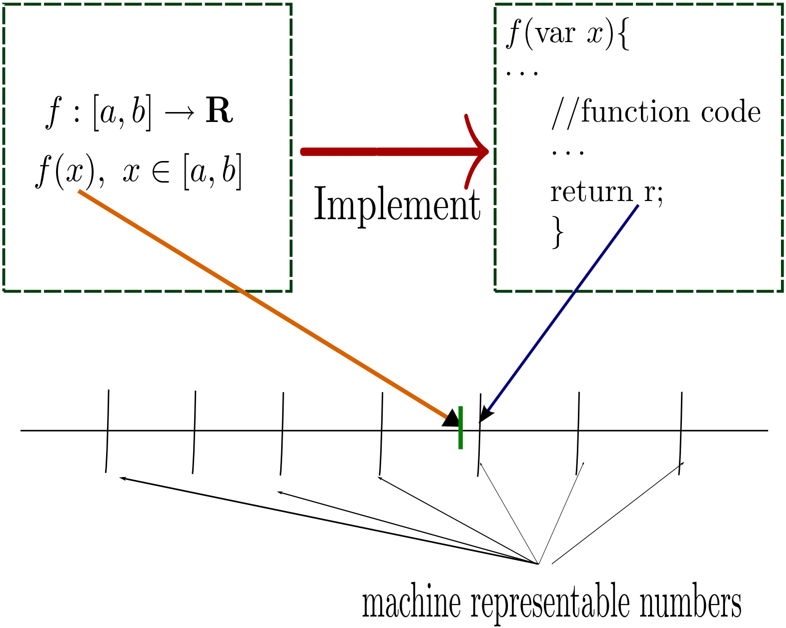
✓ Since 1985, IEEE-754 standard for FP arithmetic requests correct rounding for : $+, -, \times, \div, \sqrt{}$.

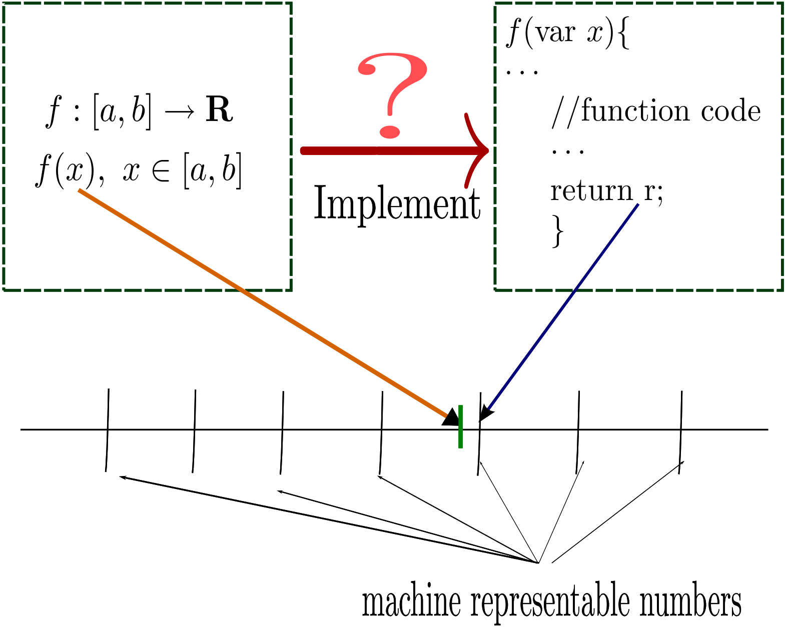
✓ Correct Rounding: An operation whose entries are FP numbers must return what we would get by infinitely precise operation followed by rounding.



- What about standard functions (sin, cos, log, etc.)?
  - Most Mathematical Libraries (libms) do not provide correctly rounded functions, although IEEE-754-2008 recommends it.
- Correctly Rounded Libm (CRLibm*) was developed by the Arénaire/AriC team, Lyon, France.

* https://gforge.inria.fr/scm/browser.php?group_id=5929&extra=crlibm


$f : [a, b] \to \mathbf{R}$
$f(x),\ x \in [a, b]$
Implement
f(var x){
...
//function code
...
return r;
}
machine representable numbers


$f : [a, b] \to \mathbf{R}$
$f(x), \ x \in [a, b]$
?
Implement
f(var x){
...
//function code
...
return r;
}
machine representable numbers

Sollya

- Tool & library for safe floating-point code development
- Targeted for automatized implementation of libms
- `http://sollya.gforge.inria.fr/`
- Developed by C. Lauter and S. Chevillard, M.J., N. Jourdan

Used for demos in this course.

Implementation of Standard Functions

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

Goal: evaluation of φ to a given accuracy η.

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

Goal: evaluation of φ to a given accuracy η.

- Step 1. Argument reduction (Payne & Hanek, Ng, Tang, etc.):
  $x \in \mathbb{R}$, $\varphi(x) \simeq f(y)$, $y \in [a, b]$.

Example

$$e^x = 2^{\frac{x}{\ln 2}} = 2^{\lceil \frac{x}{\ln 2} \rfloor} \cdot 2^{\frac{x}{\ln 2} - \lceil \frac{x}{\ln 2} \rfloor} = 2^E \cdot e^{x - E \ln(2)} = 2^E \cdot e^r, \ |r| \leq \ln 2.$$

$$= \ldots$$

$$= 2^{M+E} \cdot t_1 \cdot t_2 \cdot e^y, |y| \leq 2^{-\ell}.$$

- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f (AriC, implementation in Sollya)*.

Example

Find $c_2, c_3 \in \mathbb{Z}$ such that

$$\max_{-2^{-12} \leq x \leq 2^{-12}} \left| \exp x - \left(1 + x + \frac{c_2}{2^{53}} x^2 + \frac{c_3}{2^{53}} x^3 \right) \right|$$

is minimal.

*S. Chevillard, N. Brisebarre, A. Tisserand, S. Torres

- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f (AriC, implementation in Sollya)*.

Example

Find $c_2, c_3 \in \mathbb{Z}$ such that

$$\max_{-2^{-12} \le x \le 2^{-12}} \left| \exp x - \left(1 + x + \frac{c_2}{2^{53}} x^2 + \frac{c_3}{2^{53}} x^3 \right) \right|$$

is minimal.

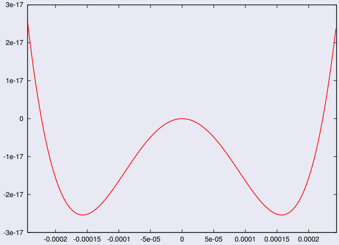
[fpminimax Sollya routine, BrisebarreChevillard2007] $\rightsquigarrow$
$p^\star(x) = 1 + x + \dfrac{4503599645901977}{2^{53}} x^2 + \dfrac{4503599645901977}{2^{52}} x^3$

*S. Chevillard, N. Brisebarre, A. Tisserand, S. Torres

- Step 3. Computation of a rigorous approximation error bound $||f - p^\star(x)||^*$

Example

$$\varepsilon(x) := \exp x - \left(1 + x + \frac{4503599645901977}{2^{53}}\,x^2 + \frac{4503599645901977}{2^{52}}\,x^3\right)$$



Prove that:

$$||\varepsilon||_{[-2^{-12};2^{-12}]} := \max_{-2^{-12} \le x \le 2^{-12}} |\varepsilon(x)| \le 2.58 \cdot 10^{-17}$$

*Sollya (S. Chevillard, M. Joldes, C. Lauter)

Implementation of Standard Functions

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

Goal: evaluation of φ to a given accuracy η.

- Step 1. Argument reduction (Payne & Hanek, Ng, Tang, etc.): $x \in \mathbb{R}$, $\varphi(x) \simeq f(y)$, $y \in [a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f (AriC, implementation in Sollya).[*].
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.[†].

[*] S. Chevillard, N. Brisebarre, A. Tisserand, S. Torres

[†] Sollya (S. Chevillard, M. Joldes, C. Lauter)

Implementation of Standard Functions

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

Goal: evaluation of φ to a given accuracy η.

- Step 1. Argument reduction (Payne & Hanek, Ng, Tang, etc.): $x \in \mathbb{R}$, $\varphi(x) \simeq f(y)$, $y \in [a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f (AriC, implementation in Sollya).[*].
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.[†].
- Step 4. Computation of a certified evaluation error of $p^\star$: GAPPA (G. Melquiond).

[*] S. Chevillard, N. Brisebarre, A. Tisserand, S. Torres

[†] Sollya (S. Chevillard, M. Joldes, C. Lauter)

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

Goal: evaluation of φ to a given accuracy η.

- Step 0. Computation of hardest-to-round cases (*binary32* done, *binary64* ongoing projects, AriC).
- Step 1. Argument reduction (Payne & Hanek, Ng, Tang, etc.): $x \in \mathbb{R}$, $\varphi(x) \simeq f(y)$, $y \in [a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f (AriC, implementation in Sollya).*.
- Step 3. Computation of a rigorous approximation error $||f - p^\star||$.†.
- Step 4. Computation of a certified evaluation error of $p^\star$: GAPPA (G. Melquiond).

---

* S. Chevillard, N. Brisebarre, A. Tisserand, S. Torres

† Sollya (S. Chevillard, M. Joldes, C. Lauter)

Framework of function evaluation, two norms over $\mathcal{C}([a,b])$:

- L^2 norm: given a nonnegative weight function $w \in \mathcal{C}([a,b])$, $\mathrm{d}x$ denotes the Lebesgue measure:

  $$g \in L^2([a,b], w, \mathrm{d}x)$$

  if

  $$\int_a^b w(x)|g(x)|^2 \mathrm{d}x < \infty,$$

  then define

  $$\|g\|_{2,w} = \sqrt{\int_a^b w(x)|g(x)|^2 \mathrm{d}x};$$

Framework of function evaluation, two norms over $\mathcal{C}([a,b])$:

- L^2 norm: given a nonnegative weight function $w \in \mathcal{C}([a,b])$, $\mathrm{d}x$ denotes the Lebesgue measure:

  $$g \in L^2([a,b], w, \mathrm{d}x)$$

  if

  $$\int_a^b w(x)|g(x)|^2 \mathrm{d}x < \infty,$$

  then define

  $$\|g\|_{2,w} = \sqrt{\int_a^b w(x)|g(x)|^2 \mathrm{d}x};$$

- L^∞ norm (aka Chebyshev, supremum norm): if g is bounded on $[a,b]$:

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

  (for continuous g, $\|g\|_\infty = \max_{x \in [a,b]} |g(x)|$).

Polynomial Approximation

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

Problem

Given $f \in \mathcal{C}([a, b])$, $n \in \mathbb{N}$, find $p \in \mathbb{R}_n[X]$ s.t.

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

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

Problem

Given $f \in \mathcal{C}([a,b])$, $n \in \mathbb{N}$, find $p \in \mathbb{R}_n[X]$ s.t.

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

- $\mathcal{C}([a,b]) \subset L^2([a,b], w, \mathrm{d}x)$, which is a complete Hilbert space with $\|\cdot\|_2$ and

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x)\mathrm{d}x,$$

  Hence, $p := \mathrm{pr}^\perp(f)$ onto $\mathbb{R}_n[x]$.

Polynomial Approximation

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

Problem

Given $f \in \mathcal{C}([a,b])$, $n \in \mathbb{N}$, find $p \in \mathbb{R}_n[X]$ s.t.

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

- $\mathcal{C}([a,b]) \subset L^2([a,b], w, \mathrm{d}x)$, which is a complete Hilbert space with $\| \cdot \|_2$ and

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x)\mathrm{d}x,$$

  Hence, $p := \mathrm{pr}^{\perp}(f)$ onto $\mathbb{R}_n[x]$.
- Weierstraß Thm. (1885) Polynomials are dense in $(\mathcal{C}([a,b]), \| \cdot \|_\infty)$

$$\inf_{q \in \mathbb{R}_n[x]} \|q - f\|_\infty \to 0 \text{ as } n \to \infty.$$

Polynomial Approximation

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

Problem

Given $f \in \mathcal{C}([a,b])$, $n \in \mathbb{N}$, find $p \in \mathbb{R}_n[X]$ s.t.

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

- $\mathcal{C}([a,b]) \subset L^2([a,b], w, \mathrm{d}x)$, which is a complete Hilbert space with $\|\cdot\|_2$ and

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x)\mathrm{d}x,$$

  Hence, $p := \mathrm{pr}^{\perp}(f)$ onto $\mathbb{R}_n[x]$.
- Weierstraß Thm. (1885) Polynomials are dense in $(\mathcal{C}([a,b]), \|\cdot\|_\infty)$

$$\inf_{q \in \mathbb{R}_n[x]} \|q - f\|_\infty \to 0 \text{ as } n \to \infty.$$

  The infimum is reached:

Let $(E, \|\cdot\|)$ be a normed $\mathbb{R}$-vector space, let F be a finite dimensional subspace of $(E, \|\cdot\|)$. For all $f \in E$, there exists $p \in F$ such that $\|p - f\| = \min_{q \in F} \|q - f\|$. Moreover, the set of best approximations to a given $f \in E$ is convex.

The best L^2 approximation is unique, which is not always the case in the L^∞ setting.

Example

Consider the interval $[-1,1]$, f be the constant function 1 and $F = \mathbb{R}g$ where $g : x \to x^2$. Determine the set of best L^∞ approximations to f.

Note that

$$\min_{c\in\mathbb{R}} \max_{x\in[-1,1]} |1 - cx^2| \geq 1,$$

attained for all $c \in [0,2]$.

In the case of L^∞, it is necessary to introduce an additional condition known as the Haar condition.

Haar Condition

Consider $n+1$ functions $\varphi_0, \ldots, \varphi_n$ defined over $[a,b]$. We say that $\varphi_0, \ldots, \varphi_n$ satisfy the Haar condition iff

1. φ_i are continuous;
2. and the following equivalent statements hold:
   - (φ_i) are $\mathbb{R}$-linearly independent and any $p = \sum_{k=0}^{n} \alpha_k \varphi_k \neq 0$ has at most n distinct zeros in $[a,b]$.
   - for all $x_0, x_1, \ldots, x_n \in [a,b]$,

$$\begin{vmatrix} \varphi_0(x_0) & \cdots & \varphi_n(x_0) \\ \vdots & & \vdots \\ \varphi_0(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} = 0 \quad \Leftrightarrow \quad \exists i \neq j, x_i = x_j;$$

A set of functions that satisfy the Haar condition is called a Chebyshev system. The prototype example is $\varphi_i(x) = x^i$, for which we have

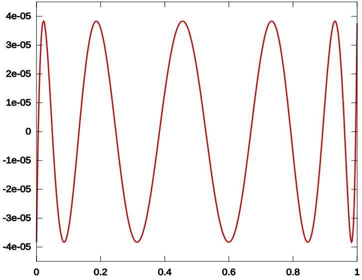
$$\begin{vmatrix} \varphi_0(x_0) & \cdots & \varphi_n(x_0) \\ \vdots & & \vdots \\ \varphi_0(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} = \begin{vmatrix} 1 & \cdots & x_0^n \\ \vdots & & \vdots \\ 1 & \cdots & x_n^n \end{vmatrix} = V_n = \prod_{0 \leqslant i < j \leqslant n} (x_j - x_i).$$

Alternation Theorem. Kirchberger (1902)

Let $\{\varphi_0, \dots, \varphi_n\}$ be a Chebyshev system over $[a, b]$. Let $f \in \mathcal{C}([a, b])$.
A generalized polynomial $p = \sum_{k=0}^{n} \alpha_k \varphi_k$ is the best approximation to f iff there exist $n + 2$ points $a \leqslant x_0 < x_1 < \dots < x_{n+1} \leqslant b$ such that, for all k,

$$f(x_k) - p(x_k) = (-1)^k (f(x_0) - p(x_0)) = \pm \|f - p\|_\infty .$$

Example error plot:



best approximation $p \Leftrightarrow$ error $f - p$ has at least $n + 2$ extrema, all global and with alternating signs.

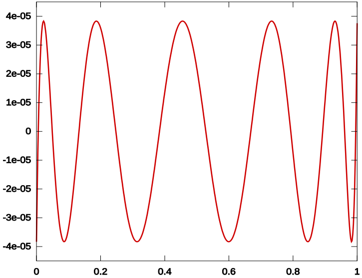
Best L^∞ (Minimax) Approximation. Uniqueness

Alternation Theorem. Kirchberger (1902)

Let $\{\varphi_0, \ldots, \varphi_n\}$ be a Chebyshev system over $[a, b]$. Let $f \in \mathcal{C}([a, b])$.
A generalized polynomial $p = \sum_{k=0}^{n} \alpha_k \varphi_k$ is the best approximation to f iff there exist $n+2$ points $a \leqslant x_0 < x_1 < \cdots < x_{n+1} \leqslant b$ such that, for all k,

$$f(x_k) - p(x_k) = (-1)^k (f(x_0) - p(x_0)) = \pm \|f - p\|_\infty.$$

Example error plot:



best approximation $p \Leftrightarrow$ error $f - p$ has at least $n+2$ extrema, all global and with alternating signs.

$\rightsquigarrow$ An iterative algorithm due to Remez(1934) approximates p.

Algorithm

Input: An interval $[a,b]$, a function $f \in \mathcal{C}([a,b])$, a natural integer n, a Chebyshev system $\{\varphi_k\}_{0 \leqslant k \leqslant n}$, a tolerance Δ.
Output: An approx of degree n-minimax polynomial of f on the system $\{\varphi_k\}_{0 \leq k \leq n}$.

- Choose $n+2$ points $x_0 < x_1 < \cdots < x_{n+1}$ in $[a,b]$, $\delta \leftarrow 1, \varepsilon \leftarrow 0$.
- WHILE $\delta \geqslant \Delta|\varepsilon|$
  - Determine the solutions $a_0, \ldots, a_n$ and ε of the linear system

    $$\sum_{k=0}^{n} a_k \varphi_k(x_j) - f(x_j) = (-1)^j \varepsilon, \; j = 0, \ldots, n+1.$$

  - Choose $x_{\text{new}} \in [a,b]$ such that

    $$\|p - f\|_\infty = |p(x_{\text{new}}) - f(x_{\text{new}})|, \text{with } p = \sum_{k=0}^{n} a_k \varphi_k.$$

  - Replace one of the x_i with x_{new}, in such a way that the sign of $p - f$ alternates at the points of the resulting discretization $x_{0,\text{new}}, \ldots, x_{n+1,\text{new}}$.
  - $\delta \leftarrow |p(x_{\text{new}}) - f(x_{\text{new}})| - |\varepsilon|$.
- Return p.

Keep calm and (don't) read, a step-by-step demo follows!

Truncated Polynomials

Standard Functions Implementation $\rightsquigarrow$ Coefficients encoded on finite (constrained) format.

Truncated Polynomials

Standard Functions Implementation $\rightsquigarrow$ Coefficients encoded on finite (constrained) format.

Let $m = (m_i)_{0 \leq i \leq n}$ a finite sequence of rational integers. Let

$$\mathcal{P}_n^m = \{q = q_0 + q_1 x + \cdots + q_n x^n \in \mathbb{R}_n[X]; q_i \text{ integer multiple of } 2^{-m_i}, \forall i\}.$$

Standard Functions Implementation $\rightsquigarrow$ Coefficients encoded on finite (constrained) format.

Let $m = (m_i)_{0 \le i \le n}$ a finite sequence of rational integers. Let

$$\mathcal{P}_n^m = \{q = q_0 + q_1 x + \cdots + q_n x^n \in \mathbb{R}_n[X]; q_i \text{ integer multiple of } 2^{-m_i}, \forall i\}.$$

Question: find $p^\star \in \mathcal{P}_n^m$ which minimizes $||f - q||$, $q \in \mathcal{P}_n^m$.

Standard Functions Implementation $\rightsquigarrow$ Coefficients encoded on finite (constrained) format.

Let $m = (m_i)_{0 \leq i \leq n}$ a finite sequence of rational integers. Let

$$\mathcal{P}_n^m = \{q = q_0 + q_1 x + \cdots + q_n x^n \in \mathbb{R}_n[X]; q_i \text{ integer multiple of } 2^{-m_i}, \forall i\}.$$

Question: find $p^\star \in \mathcal{P}_n^m$ which minimizes $||f - q||$, $q \in \mathcal{P}_n^m$.

First idea. Remez $\rightarrow p(x) = p_0 + p_1 x + \cdots + p_n x^n$. Every p_i rounded to $\hat{a}_i/2^{m_i}$, the nearest integer multiple of 2^{-m_i} $\rightarrow \hat{p}(x) = \dfrac{\hat{a}_0}{2^{m_0}} + \dfrac{\hat{a}_1}{2^{m_1}} x + \cdots + \dfrac{\hat{a}_n}{2^{m_n}} x^n$.

Truncated Polynomials

Standard Functions Implementation $\rightsquigarrow$ Coefficients encoded on finite (constrained) format.

Let $m = (m_i)_{0 \leq i \leq n}$ a finite sequence of rational integers. Let

$$\mathcal{P}_n^m = \{q = q_0 + q_1 x + \cdots + q_n x^n \in \mathbb{R}_n[X]; q_i \text{ integer multiple of } 2^{-m_i}, \forall i\}.$$

Question: find $p^\star \in \mathcal{P}_n^m$ which minimizes $||f - q||$, $q \in \mathcal{P}_n^m$.

First idea. Remez $\rightarrow p(x) = p_0 + p_1 x + \cdots + p_n x^n$. Every p_i rounded to $\hat{a}_i / 2^{m_i}$, the nearest integer multiple of $2^{-m_i} \rightarrow \hat{p}(x) = \dfrac{\hat{a}_0}{2^{m_0}} + \dfrac{\hat{a}_1}{2^{m_1}} x + \cdots + \dfrac{\hat{a}_n}{2^{m_n}} x^n$.

Problem: $\hat{p}$ not necessarily a minimax approx. of f among the polynomials of $\mathcal{P}_n^m$.

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

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

is minimal.

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

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

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

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

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.

Approaches for best "truncated" approximants

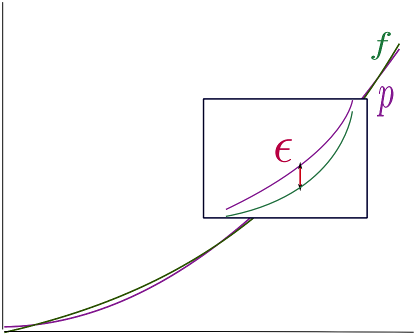
- Linear programming: tackle degree-8 or 10 polynomials: good for hardware-oriented applications, not satisfying for software-oriented.
- Lattice Basis Reduction: much faster and more efficient, gives a very good approximant (e.g. provides practical gains of 16 bits in double precision implementation of $\arcsin$ function).
- Works of N. Brisebarre, S. Chevillard, A. Tisserand, S. Torres.
- Nice implementation in Sollya

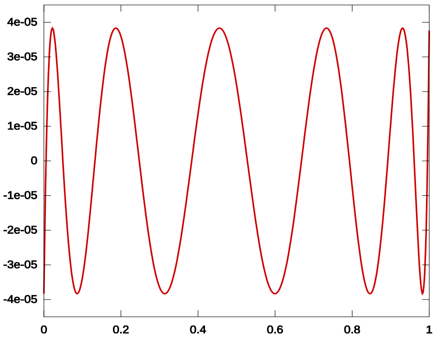
- Step 0. Computation of hardest-to-round cases.
- Step 1. Argument reduction $\rightsquigarrow f(y)$, $y \in [a, b]$.
- Step 2. Computation of p, a "machine-efficient" polynomial approximation of f.
- Step 3. Computation of a rigorous approximation error bound $||f - p||_\infty$

- Step 3. Computation of a rigorous approximation error bound $||f - p||_\infty$

Example

$f(x) = e^{1/\cos(x)}$, $x \in [0, 1]$, $p(x) = \sum_{i=0}^{10} c_i x^i$, $\varepsilon(x) = f(x) - p(x)$ s.t. $\|\varepsilon\|_\infty = \sup_{x \in [a,\, b]}\{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)





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

* http://gforge.inria.fr/projects/mpfi/

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

* http://gforge.inria.fr/projects/mpfi/

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

*http://gforge.inria.fr/projects/mpfi/

1. Interval arithmetic (IA)

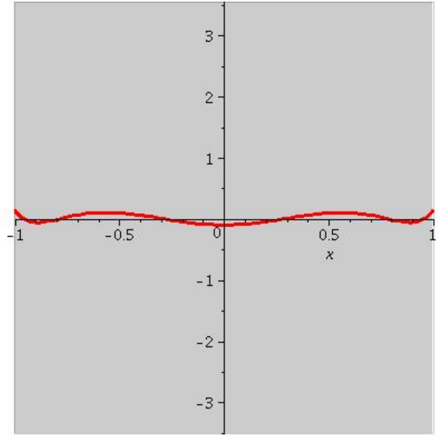
- Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI*)
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1,2] + [-3,2] = [-2,4]$
- Range bounding for functions
  Eg. $x \in [-1,2], f(x) = x^2 - x + 1$
  $F(X) = X^2 - X + 1$
  $F([-1,2])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

*http://gforge.inria.fr/projects/mpfi/

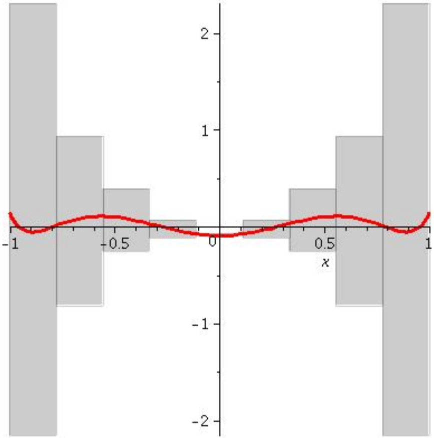
1. Interval arithmetic (IA)
   - Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI*)
   - $\pi \in [3.1415, 3.1416]$
   - Interval Arithmetic Operations
     Eg. $[1, 2] + [-3, 2] = [-2, 4]$
   - Range bounding for functions
     Eg. $x \in [-1, 2], f(x)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

*http://gforge.inria.fr/projects/mpfi/

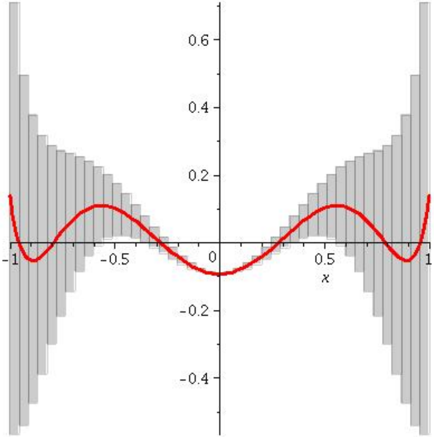
1. Interval arithmetic (IA)
   - Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI*)
   - $\pi \in [3.1415, 3.1416]$
   - Interval Arithmetic Operations
     Eg. $[1, 2] + [-3, 2] = [-2, 4]$
   - Range bounding for functions $\rightsquigarrow$ Overestimation
     Eg. $x \in [-1, 1], f(x) = e^{1/\cos(x)} - 2.8114 - 3.411x^4$



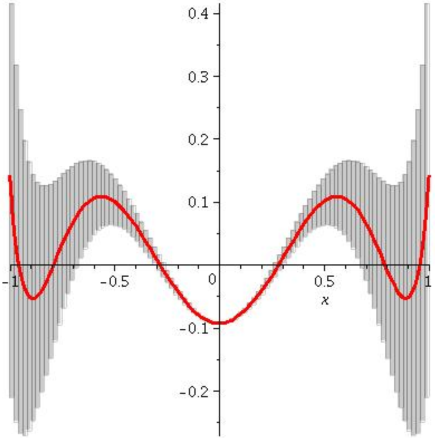
1. Interval arithmetic (IA)
   - Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI*)
   - $\pi \in [3.1415, 3.1416]$
   - Interval Arithmetic Operations
     Eg. $[1, 2] + [-3, 2] = [-2, 4]$
   - Range bounding for functions $\rightsquigarrow$ Overestimation
     Eg. $x \in [-1, 1], f(x) = e^{1/\cos(x)} - 2.8114 - 3.411x^4$



1. Interval arithmetic (IA)
   - Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI*)
   - $\pi \in [3.1415, 3.1416]$
   - Interval Arithmetic Operations
     Eg. $[1, 2] + [-3, 2] = [-2, 4]$
   - Range bounding for functions $\rightsquigarrow$ Overestimation
     Eg. $x \in [-1, 1], f(x) = e^{1/\cos(x)} - 2.8114 - 3.411x^4$

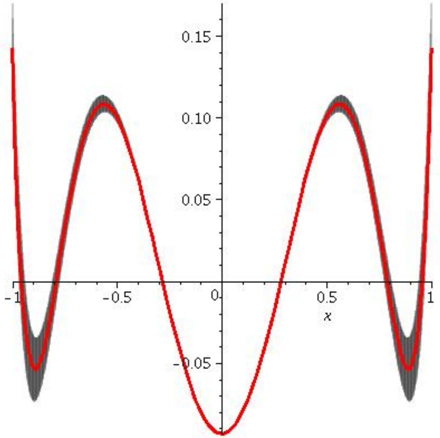


1. Interval arithmetic (IA)
   - Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI*)
   - $\pi \in [3.1415, 3.1416]$
   - Interval Arithmetic Operations
     Eg. $[1, 2] + [-3, 2] = [-2, 4]$
   - Range bounding for functions $\rightsquigarrow$ Overestimation
     Eg. $x \in [-1, 1], f(x) = e^{1/\cos(x)} - 2.8114 - 3.411x^4$



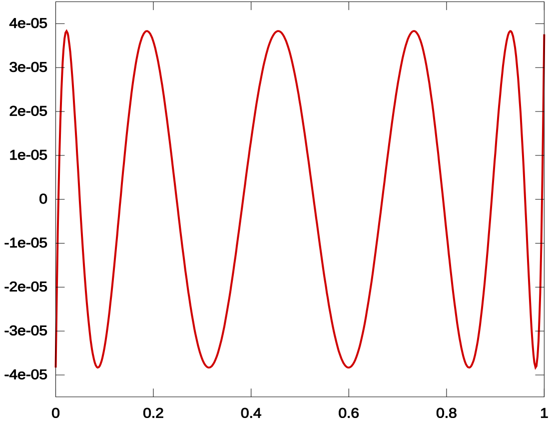
1. Interval arithmetic (IA)

- Each interval = pair of floating-point numbers (multiple precision IA libraries exist, e.g. MPFI*)
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1,2] + [-3,2] = [-2,4]$
- Range bounding for functions $\rightsquigarrow$ Overestimation
  Eg. $x \in [-1,1], f(x) = e^{1/\cos(x)} - 2.8114 - 3.411x^4$



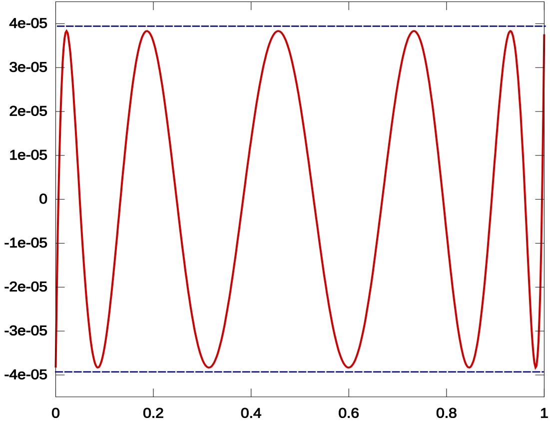
When Interval Arithmetic does not suffice: Computing supremum norms of approximation errors

$f(x) = e^{1/\cos(x)},\ x \in [0,1],\ \ p(x) = \sum_{i=0}^{10} c_i x^i$, $\varepsilon(x) = f(x) - p(x)$ s.t.
$\|\varepsilon\|_\infty = \sup_{x\in[a,\,b]}\{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)



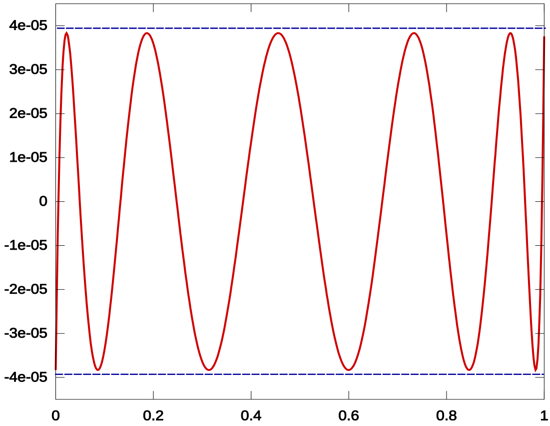
When Interval Arithmetic does not suffice: Computing supremum norms of approximation errors

$f(x) = e^{1/\cos(x)},\ x \in [0,1],\ \ p(x) = \sum_{i=0}^{10} c_i x^i,\ \varepsilon(x) = f(x) - p(x)$ s.t.
$\|\varepsilon\|_\infty = \sup_{x \in [a,\,b]} \{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)



When Interval Arithmetic does not suffice: Computing supremum norms of approximation errors

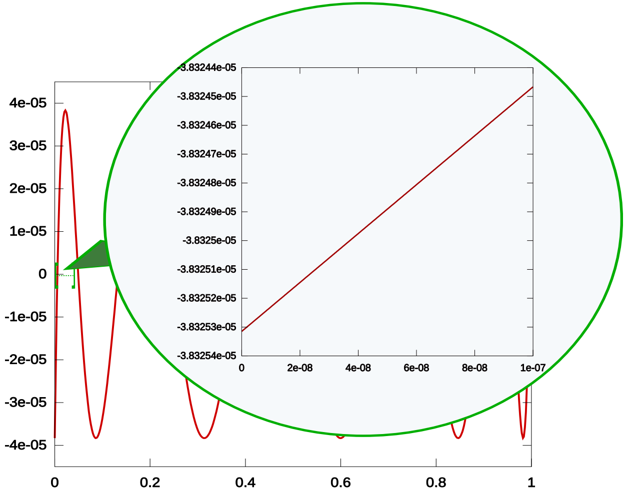
$f(x) = e^{1/\cos(x)}$, $x \in [0, 1]$, $p(x) = \sum_{i=0}^{10} c_i x^i$, $\varepsilon(x) = f(x) - p(x)$ s.t. $\|\varepsilon\|_\infty = \sup_{x \in [a,\, b]} \{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)



Using IA, $\varepsilon(x) \in [-233, 298]$, but $\|\varepsilon(x)\|_\infty \simeq 3.8325 \cdot 10^{-5}$

Why IA does not suffice: Overestimation

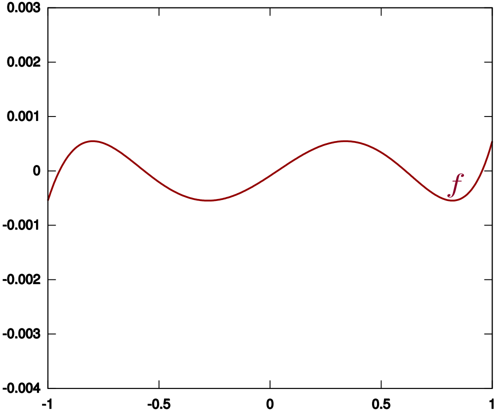
Overestimation can be reduced by using intervals of smaller width.



In this case, over $[0, 1]$ we need 10^7 intervals!

Rigorous polynomial approximations (RPAs)

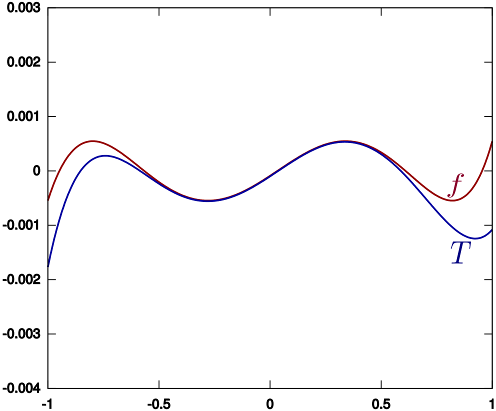
$$||f - p|| \leq$$



Rigorous polynomial approximations (RPAs)

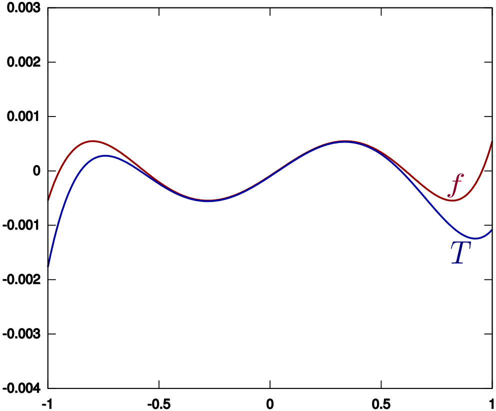
$$||f - p|| \leq \underbrace{||f - T||}_{\text{easier to compute}} + \underbrace{||T - p||}_{\text{reduced dependency}}$$

f replaced with
- polynomial approximation T



$$||f - p|| \leq \underbrace{||f - T||}_{\text{easier to compute}} + \underbrace{||T - p||}_{\text{reduced dependency}}$$

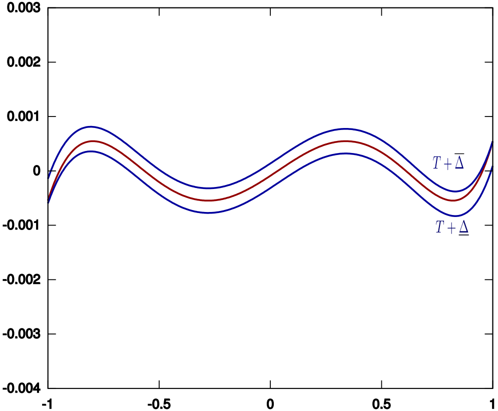
f replaced with
- polynomial approximation T
- interval $\boldsymbol{\Delta}$ s. t. $f(x) - T(x) \in \boldsymbol{\Delta}, \forall x \in [a, b]$



$$||f - p|| \leq \underbrace{||f - T||}_{\text{easier to compute}} + \underbrace{||T - p||}_{\text{reduced dependency}}$$

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

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



Rigorous polynomial approximations (RPAs)

– Consider "sufficiently smooth" univariate functions f over $[a, b]$.
– f replaced with a rigorous polynomial approximation : $(T, \boldsymbol{\Delta})$

(1). RPAs based on Taylor series
⇝ Taylor Models (TMs).

(2). ⇝ Certify RPAs based on best polynomial approximations: use intermediary RPAs obtained in (1), (3).

(3). Near-best RPAs: based on Chebyshev Series
⇝ Chebyshev Models (CMs).

- f is an elementary function, e.g. $\exp(1/\cos(x))$;
- f is a D-finite function, i.e. solution of an ordinary differential equation with polynomial coefficients, e.g. $\exp$, Airy, Bessel.

(4). Other orthogonal polynomials...

Rigorous polynomial approximations (RPAs)

– Consider "sufficiently smooth" univariate functions f over $[a, b]$.
– f replaced with a rigorous polynomial approximation : $(T, \boldsymbol{\Delta})$

(1). RPAs based on Taylor series
$\rightsquigarrow$ Taylor Models (TMs).

(2). $\rightsquigarrow$ Certify RPAs based on best polynomial approximations: use intermediary RPAs obtained in (1), (3).

(3). Near-best RPAs: based on Chebyshev Series
$\rightsquigarrow$ Chebyshev Models (CMs).

- f is an elementary function, e.g. $\exp(1/\cos(x))$;
- f is a D-finite function, i.e. solution of an ordinary differential equation with polynomial coefficients, e.g. $\exp$, Airy, Bessel.

(4). Other orthogonal polynomials...

Rigorous polynomial approximations (RPAs)

– Consider "sufficiently smooth" univariate functions f over $[a, b]$.
– f replaced with a rigorous polynomial approximation : $(T, \boldsymbol{\Delta})$

(1). RPAs based on Taylor series
$\rightsquigarrow$ Taylor Models (TMs).

(2). $\rightsquigarrow$ Certify RPAs based on best polynomial approximations: use intermediary RPAs obtained in (1), (3).

(3). Near-best RPAs: based on Chebyshev Series
$\rightsquigarrow$ Chebyshev Models (CMs).

- f is an elementary function, e.g. $\exp(1/\cos(x))$;
- f is a D-finite function, i.e. solution of an ordinary differential equation with polynomial coefficients, e.g. $\exp$, Airy, Bessel.

(4). Other orthogonal polynomials...

Rigorous polynomial approximations (RPAs)

– Consider "sufficiently smooth" univariate functions f over $[a, b]$.
– f replaced with a rigorous polynomial approximation : $(T, \mathbf{\Delta})$

(1). RPAs based on Taylor series
$\rightsquigarrow$ Taylor Models (TMs).

(2). $\rightsquigarrow$ Certify RPAs based on best polynomial approximations: use intermediary RPAs obtained in (1), (3).

(3). Near-best RPAs: based on Chebyshev Series
$\rightsquigarrow$ Chebyshev Models (CMs).

- f is an elementary function, e.g. $\exp(1/\cos(x))$;
- f is a D-finite function, i.e. solution of an ordinary differential equation with polynomial coefficients, e.g. $\exp$, Airy, Bessel.

(4). Other orthogonal polynomials...

Rigorous polynomial approximations (RPAs)

– Consider "sufficiently smooth" univariate functions f over $[a, b]$.
– f replaced with a rigorous polynomial approximation : $(T, \mathbf{\Delta})$

(1). RPAs based on Taylor series
$\rightsquigarrow$ Taylor Models (TMs).

(2). $\rightsquigarrow$ Certify RPAs based on best polynomial approximations: use intermediary RPAs obtained in (1), (3).

(3). Near-best RPAs: based on Chebyshev Series
$\rightsquigarrow$ Chebyshev Models (CMs).

- f is an elementary function, e.g. $\exp(1/\cos(x))$;
- f is a D-finite function, i.e. solution of an ordinary differential equation with polynomial coefficients, e.g. $\exp$, Airy, Bessel.

(4). Other orthogonal polynomials...

Taylor Models

- Consider Taylor approximations

- Consider Taylor approximations

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

$$f(x) = \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!}}_{T(x)} + \underbrace{\Delta_n(x)}_{\boldsymbol{\Delta}}$$

- Consider Taylor approximations

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

$$f(x) = \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!}}_{T(x)} + \underbrace{\Delta_n(x)}_{\boldsymbol{\Delta}}$$

- For obtaining $\boldsymbol{\Delta}$:

- For "basic functions" ($\sin$, $\cos$, etc.) use Lagrange formula

  $$\forall x \in [a,b],\ \exists \xi \in [a,b] \text{ s.t. } \Delta_n(x,\xi) = \frac{f^{(n+1)}(\xi)(x-x_0)^{n+1}}{(n+1)!}$$

Taylor Models

- Consider Taylor approximations

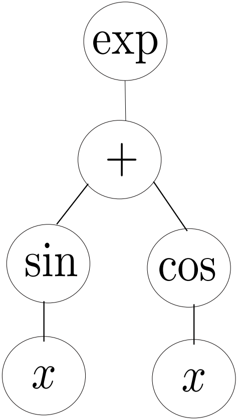
Let $n \in \mathbb{N}$, $n+1$ times differentiable function f over $[a,b]$ around x_0.

$$f(x) = \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!}}_{T(x)} + \underbrace{\Delta_n(x)}_{\boldsymbol{\Delta}}$$

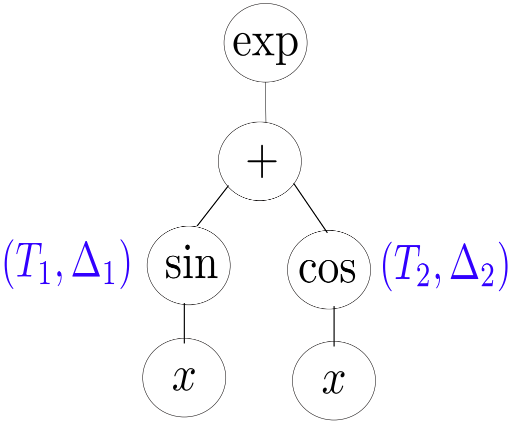
- For obtaining $\boldsymbol{\Delta}$:

- For "basic functions" $(\sin,\ \cos,$ etc.$)$ use Lagrange formula
- For "composite functions"use a two-step procedure:
  - compute models $(T, \boldsymbol{\Delta})$ for all basic functions;
  - apply algebraic rules with these models, instead of operations with the corresponding functions.

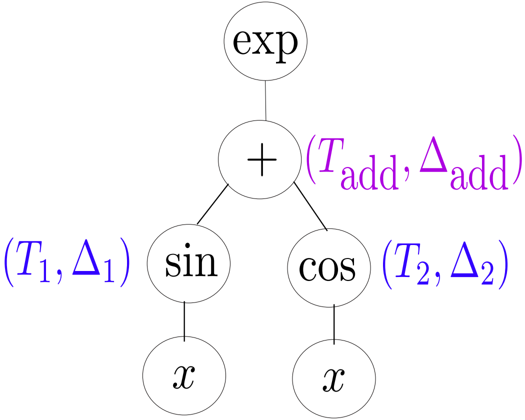
Example: $f_{\text{comp}}(x) = \exp(\sin(x) + \cos(x))$



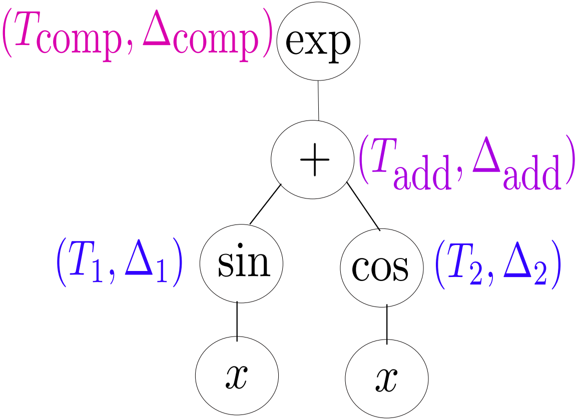
Example: $f_{\text{comp}}(x) = \exp(\sin(x) + \cos(x))$



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



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



Why use a two-step procedure for composite functions?

Otherwise $\boldsymbol{\Delta}$ can be largely overestimated.

Why use a two-step procedure for composite functions?

Otherwise $\boldsymbol{\Delta}$ can be largely overestimated.

Example:

$f(x) = e^{1/\cos x}$, over $[0, 1]$, $n = 13$, $x_0 = 0.5$. $f(x) - T(x) \in [0, 4.56 \cdot 10^{-3}]$

Why use a two-step procedure for composite functions?

Otherwise $\boldsymbol{\Delta}$ can be largely overestimated.

Example:

$f(x) = e^{1/\cos x}$, over $[0, 1]$, $n = 13$, $x_0 = 0.5$. $f(x) - T(x) \in [0, 4.56 \cdot 10^{-3}]$

- Automatic differentiation and Lagrange formula:
  $\boldsymbol{\Delta} = [-1.93 \cdot 10^2, \, 1.35 \cdot 10^3]$

Why use a two-step procedure for composite functions?

Otherwise $\boldsymbol{\Delta}$ can be largely overestimated.

Example:

$f(x) = e^{1/\cos x}$, over $[0, 1]$, $n = 13$, $x_0 = 0.5$. $f(x) - T(x) \in [0, 4.56 \cdot 10^{-3}]$

- Automatic differentiation and Lagrange formula:
  $\boldsymbol{\Delta} = [-1.93 \cdot 10^2, 1.35 \cdot 10^3]$
- Cauchy's Estimate
  $\boldsymbol{\Delta} = [-9.17 \cdot 10^{-2}, 9.17 \cdot 10^{-2}]$

Why use a two-step procedure for composite functions?

Otherwise $\boldsymbol{\Delta}$ can be largely overestimated.

Example:

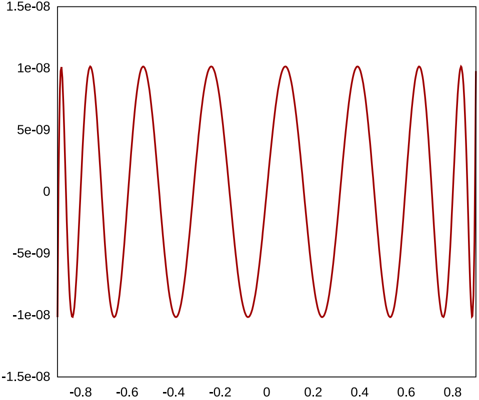
$f(x) = e^{1/\cos x}$, over $[0, 1]$, $n = 13$, $x_0 = 0.5$. $f(x) - T(x) \in [0, 4.56 \cdot 10^{-3}]$

- Automatic differentiation and Lagrange formula:
  $\boldsymbol{\Delta} = [-1.93 \cdot 10^2, 1.35 \cdot 10^3]$
- Cauchy's Estimate
  $\boldsymbol{\Delta} = [-9.17 \cdot 10^{-2}, 9.17 \cdot 10^{-2}]$
- Taylor Models
  $\boldsymbol{\Delta} = [-9.04 \cdot 10^{-3}, 9.06 \cdot 10^{-3}]$

Another L^∞ (Minimax) example

Example:

$f(x) = \arctan(x)$ over $[-0.9, 0.9]$, $p(x)$ - minimax, degree 15,
$\varepsilon(x) = p(x) - f(x)$, $\|\varepsilon\|_\infty \simeq 10^{-8}$



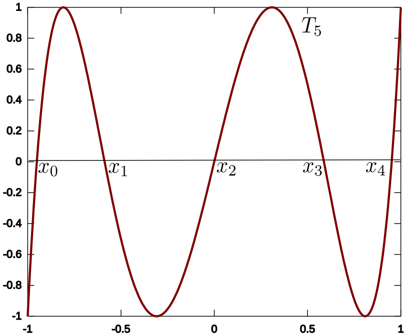
Taylor approximations:
need a TM of degree 120 (in theory)

In practice, computed interval error bound not sufficiently small due to overestimation.

Improvement?

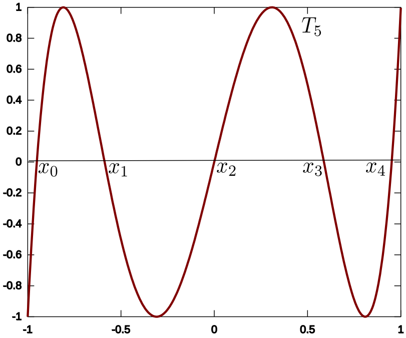
- Use a polynomial approximation better than Taylor:
  - Why?
    – better convergence domains
    – better compact approximations on larger domains

$$T_n(cos(\theta)) = \cos(n\theta)$$



Quick Reminder: Chebyshev Polynomials

$T_n(cos(\theta)) = \cos(n\theta)$

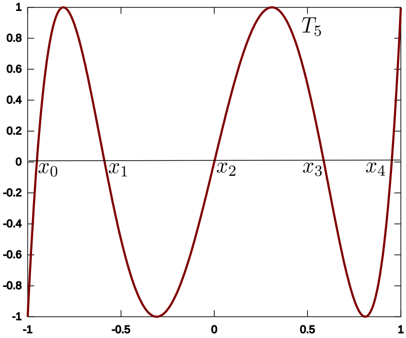


Chebyshev nodes: n distinct real roots in $[-1, 1]$ of T_n

$$x_k = \cos\left(\frac{(k+1/2)\,\pi}{n}\right), k = 0, \ldots, n-1.$$

Quick Reminder: Chebyshev Polynomials

$T_n(cos(\theta)) = \cos(n\theta)$



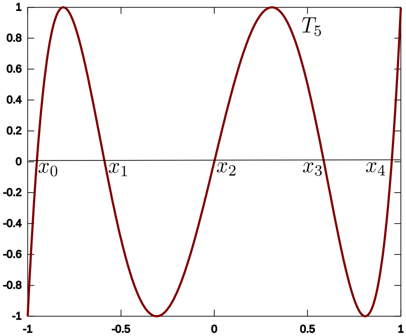
$T_{i+1} = 2xT_i - T_{i-1}, T_0(x) = 1, T_1(x) = x$

Chebyshev nodes: n distinct real roots in $[-1, 1]$ of T_n

$x_k = \cos\left(\frac{(k+1/2)\,\pi}{n}\right), k = 0, \ldots, n-1.$

Quick Reminder: Chebyshev Polynomials

$T_n(cos(\theta)) = \cos(n\theta)$



$T_{i+1} = 2xT_i - T_{i-1}, T_0(x) = 1, T_1(x) = x$

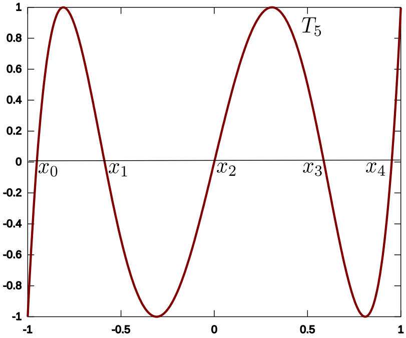
Orthogonality:

$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & \text{if } i \neq j \\ \pi & \text{if } i = 0 \\ \frac{\pi}{2} & \text{otherwise} \end{cases}$$

Chebyshev nodes: n distinct real roots in $[-1, 1]$ of T_n

$x_k = \cos\left(\frac{(k+1/2)\,\pi}{n}\right), k = 0, \ldots, n-1.$

$T_n(cos(\theta)) = \cos(n\theta)$



$$T_{i+1} = 2xT_i - T_{i-1}, T_0(x) = 1, T_1(x) = x$$

Orthogonality:

$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}}dx = \begin{cases} 0 & \text{if } i \neq j \\ \pi & \text{if } i = 0 \\ \frac{\pi}{2} & \text{otherwise} \end{cases}$$

$$\sum_{k=0}^{n-1} T_i(x_k)T_j(x_k) = \begin{cases} 0 & \text{if } i \neq j \\ n & \text{if } i = 0 \\ \frac{n}{2} & \text{otherwise} \end{cases}$$

Chebyshev nodes: n distinct real roots in $[-1, 1]$ of T_n

$$x_k = \cos\left(\frac{(k+1/2)\,\pi}{n}\right), k = 0, \ldots, n-1.$$

Chebyshev Series vs Taylor Series

Two approximations of f:

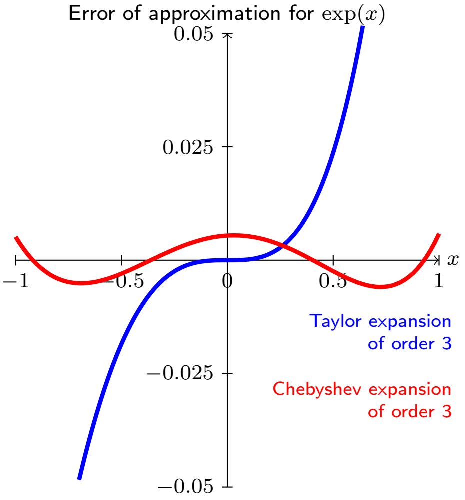
- by Taylor series

$$f = \sum_{n=0}^{+\infty} c_n x^n, \;\; c_n = \frac{f^{(n)}(0)}{n!},$$

- or by Chebyshev series

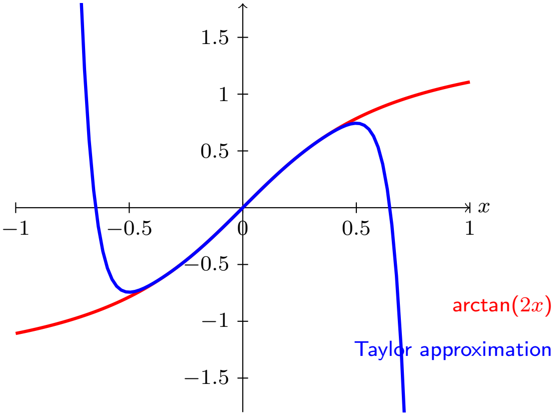
$$f = \sum_{n=-\infty}^{+\infty} t_n T_n(x),$$

$$t_n = \frac{1}{\pi} \int_{-1}^{1} T_n(t) \frac{f(t)}{\sqrt{1-t^2}} dt.$$

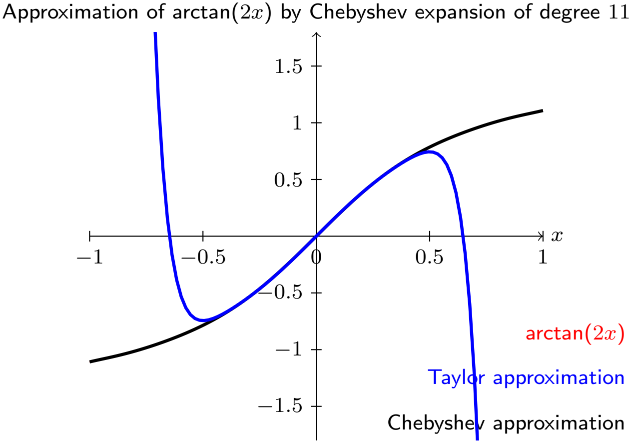


Chebyshev Series vs Taylor Series II

Bad approximation outside its circle of convergence



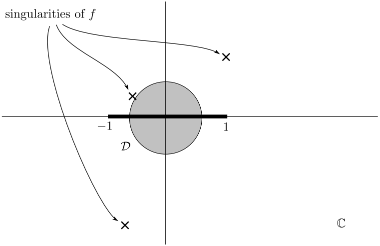
Chebyshev Series vs Taylor Series II



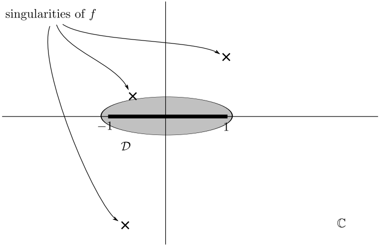
Chebyshev Series vs Taylor Series III

Convergence Domains :

For Taylor series:
disc centered at $x_0 = 0$ which avoids all the singularities of f

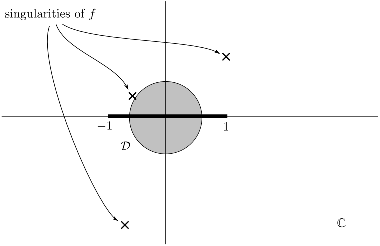


For Chebyshev series:
elliptic disc with foci at ± 1 which avoids all the singularities of f

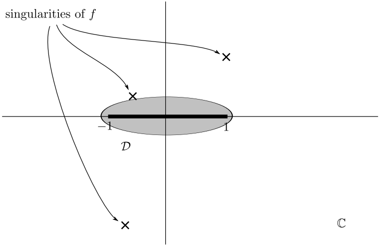


Convergence Domains :

For Taylor series:
disc centered at $x_0 = 0$ which avoids all the singularities of f



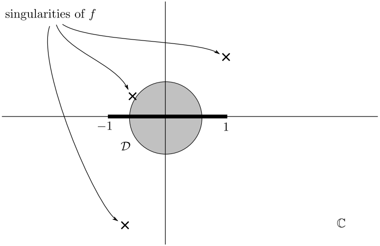
For Chebyshev series:
elliptic disc with foci at ± 1 which avoids all the singularities of f



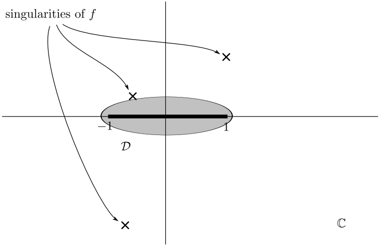
- Taylor series can not converge over entire $[-1, 1]$ unless all singularities lie outside the unit circle.

Convergence Domains :

For Taylor series:
disc centered at $x_0 = 0$ which avoids all the singularities of f



For Chebyshev series:
elliptic disc with foci at ± 1 which avoids all the singularities of f



- Taylor series can not converge over entire $[-1, 1]$ unless all singularities lie outside the unit circle.

✓ Chebyshev series converge over entire $[-1, 1]$ as soon as there are no real singularities in $[-1, 1]$.

Chebyshev Series vs Taylor Series IV

Truncation Error :

Taylor series, Lagrange formula:

$\forall x \in [-1,1], \ \exists \xi \in [-1,1] \ \text{ s.t.}$

$$f(x) - T(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}.$$

Chebyshev Series vs Taylor Series IV

Truncation Error :

Taylor series, Lagrange formula:

$\forall x \in [-1,1],\ \exists \xi \in [-1,1]\ \text{ s.t.}$

$$f(x) - T(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}.$$

Chebyshev series, Bernstein-like formula:

$\forall x \in [-1,1],\ \exists \xi \in [-1,1]\ \text{ s.t.}$

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi)}{2^n(n+1)!}.$$

Chebyshev Series vs Taylor Series IV

Truncation Error :

Taylor series, Lagrange formula:

$\forall x \in [-1,1],\ \exists \xi \in [-1,1]$ s.t.

$$f(x) - T(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}.$$

Chebyshev series, Bernstein-like formula:

$\forall x \in [-1,1],\ \exists \xi \in [-1,1]$ s.t.

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi)}{2^n (n+1)!}.$$

[✓] We should have an improvement of 2^n in the width of the Chebyshev truncation error.

Quality of approximation of truncated Chebyshev series compared to best polynomial approximation

It is well-known that truncated Chebyshev series $\pi_d(f)$ are *near-best* uniform approximations [Chap 5.5, Mason & Handscomb 2003].

Quality of approximation of truncated Chebyshev series compared to best polynomial approximation

It is well-known that truncated Chebyshev series $\pi_d(f)$ are *near-best* uniform approximations [Chap 5.5, Mason & Handscomb 2003].

Let p_d^* is the polynomial of degree at most d that minimizes $\|f - p\|_\infty = \sup_{-1 \leq x \leq 1} |f(x) - p(x)|$.

Quality of approximation of truncated Chebyshev series compared to best polynomial approximation

It is well-known that truncated Chebyshev series $\pi_d(f)$ are *near-best* uniform approximations [Chap 5.5, Mason & Handscomb 2003].

Let p_d^* is the polynomial of degree at most d that minimizes $\|f - p\|_\infty = \sup_{-1 \leq x \leq 1} |f(x) - p(x)|$.

$$\|f - \pi_d(f)\|_\infty \leqslant \underbrace{\left(\frac{4}{\pi^2} \log d + O(1)\right)}_{\Lambda_d} \|f - p_d^*\|_\infty \quad (1)$$

Quality of approximation of truncated Chebyshev series compared to best polynomial approximation

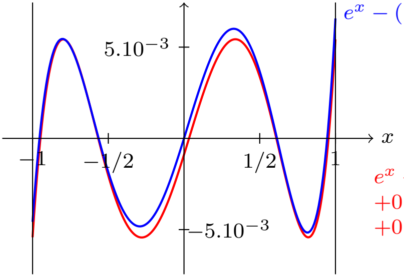
It is well-known that truncated Chebyshev series $\pi_d(f)$ are *near-best* uniform approximations [Chap 5.5, Mason & Handscomb 2003].

Let p_d^* is the polynomial of degree at most d that minimizes $\|f - p\|_\infty = \sup_{-1 \leq x \leq 1} |f(x) - p(x)|$.

$$\|f - \pi_d(f)\|_\infty \leqslant \underbrace{\left(\frac{4}{\pi^2} \log d + O(1)\right)}_{\Lambda_d} \|f - p_d^*\|_\infty \qquad (1)$$

- $\Lambda_{10} = 2.22... \rightarrow$ we lose at most 2 bits
- $\Lambda_{30} = 2.65... \rightarrow$ we lose at most 2 bits
- $\Lambda_{100} = 3.13... \rightarrow$ we lose at most 3 bits
- $\Lambda_{500} = 3.78... \rightarrow$ we lose at most 3 bits

Chebyshev truncations are near-best : Example



$e^x - (1,2660 T_0(x) + 1,1303 T_1(x) + 0,2714 T_2(x) + 0,0443 T_3(x) + 0,0054 T_4(x))$

$e^x - (1.0000 + 0.9973x + 0.4988x^2 + 0.1773x^3 + 0.0441x^4)$

Chebyshev truncation of degree 4

Best approximant of degree 4

Chebyshev Series vs Taylor Series (9gag version)

Computing the coefficients

Chebyshev series of $f = \sum_{i=-\infty}^{+\infty} t_i T_i(x)$:

– Orthogonality $\rightsquigarrow t_i = \frac{1}{\pi} \int_{-1}^{1} T_i(t) \frac{f(t)}{\sqrt{1-t^2}} dt \rightsquigarrow$ TCS

Chebyshev series of $f = \sum_{i=-\infty}^{+\infty} t_i T_i(x)$:

– Orthogonality $\rightsquigarrow t_i = \frac{1}{\pi} \int_{-1}^{1} T_i(t) \frac{f(t)}{\sqrt{1-t^2}} dt \rightsquigarrow$ TCS

– Discrete orthogonality $\rightsquigarrow \widetilde{t}_i = \sum_{k=0}^{n} \frac{1}{n+1} f(x_k) T_i(x_k) \rightsquigarrow$ Chebyshev Interpolant (CI)

Computing the coefficients

Chebyshev series of $f = \sum_{i=-\infty}^{+\infty} t_i T_i(x)$:

– Orthogonality $\rightsquigarrow t_i = \frac{1}{\pi} \int_{-1}^{1} T_i(t) \frac{f(t)}{\sqrt{1-t^2}} dt \rightsquigarrow$ TCS

– Discrete orthogonality $\rightsquigarrow \widetilde{t}_i = \sum_{k=0}^{n} \frac{1}{n+1} f(x_k) T_i(x_k) \rightsquigarrow$ Chebyshev Interpolant (CI)

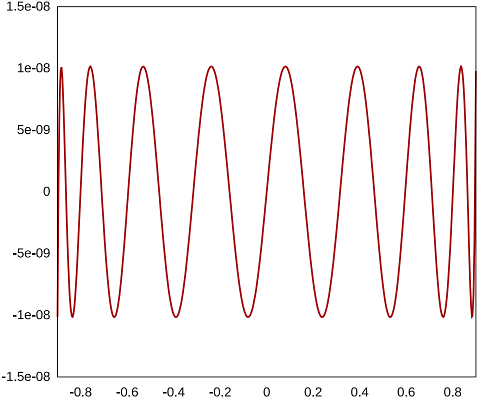
Remark: TCS or CI?

- CI: when f is elementary, evaluating f at Chebyshev nodes is easy
- TCS: when f is given by LODE

Another L^∞ (Minimax) example

Example:

$f(x) = \arctan(x)$ over $[-0.9, 0.9]$, $p(x)$ - minimax, degree 15,
$\varepsilon(x) = p(x) - f(x)$, $\|\varepsilon\|_\infty \simeq 10^{-8}$



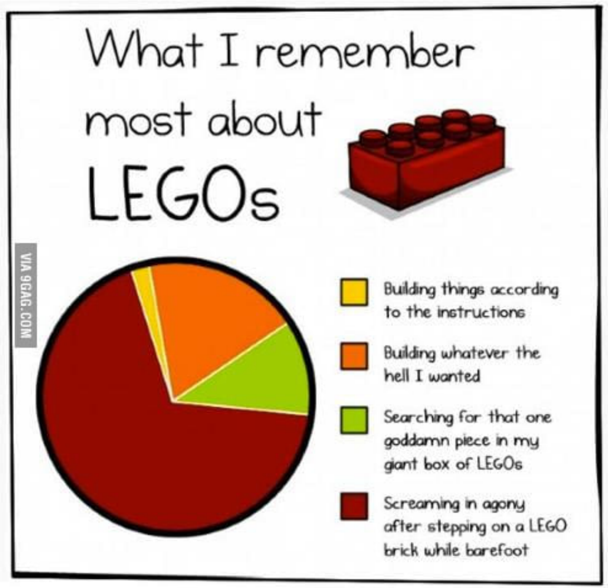
Taylor approximations:
need a TM of degree 120 (in theory)

In practice, computed interval error bound not sufficiently small due to overestimation.
A CM of degree 60 works.

CMs vs. TMs

Comparison between remainder bounds for several functions:

$f(x)$, I, n	CM	Timing (ms)	TM	Timing (ms)
$\sin(x)$, $[3,\ 4]$, 10	$1.19 \cdot 10^{-14}$	4	$1.22 \cdot 10^{-11}$	2
$\arctan(x)$, $[-0.25,\ 0.25]$, 15	$7.89 \cdot 10^{-15}$	10	$2.58 \cdot 10^{-10}$	4
$\arctan(x)$, $[-0.9,\ 0.9]$, 15	$5.10 \cdot 10^{-3}$	14	$1.67 \cdot 10^{2}$	7
$\exp(1/\cos(x))$, $[0,\ 1]$, 14	$5.22 \cdot 10^{-7}$	31	$9.06 \cdot 10^{-3}$	14
$\frac{\exp(x)}{\log(2+x)\ \cos(x)}$, $[0,\ 1]$, 15	$4.86 \cdot 10^{-9}$	38	$1.18 \cdot 10^{-3}$	19
$\sin(\exp(x))$,$[-1,\ 1]$, 10	$2.56 \cdot 10^{-5}$	7	$2.96 \cdot 10^{-2}$	4



LIBMs

IEEE 754-2008 standard

Automatic approach for many functions

Best FPMinimaxApprox

Certifying
Approx & Rounding Errors

Many thanks for N. Brisebarre and B. Salvy for useful sources and resources related to their course on approximation http://www.ens-lyon.fr/LIP/AriC/M2R/ASNA/