Rigorous Polynomial Approximations and Applications

Mioara Joldeș

under the supervision of:
Nicolas Brisebarre and Jean-Michel Muller

École Normale Supérieure de Lyon, Arénaire Team, Laboratoire de l'Informatique du Parallélisme

Joint works with:

Alexandre Benoit, Nicolas Brisebarre, Sylvain Chevillard, Florent de Dinechin, Érik Martin Dorel, John Harrison, Peter Kornerup, Christoph Lauter, Marc Mezzarobba, Jean-Michel Muller, Bogdan Pașca, Guillaume Revy.

Predictions for Scientific Computing Fifty Years From Now*

– 12 predictions

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

5. Importance of floating-point arithmetic will be undiminished.

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

5. Importance of floating-point arithmetic will be undiminished.

> *"In truth, however, no amount of computer power will change the fact that most numerical problems cannot be solved symbolically. You have to make approximations, and floating-point arithmetic is the best general-purpose approximation idea ever devised."*

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

5. Importance of floating-point arithmetic will be undiminished.
   Most numerical problems cannot be solved symbolically
   $\rightarrow$ make approximations.

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

3. Numerical computing will be adaptive, iterative, exploratory.

5. Importance of floating-point arithmetic will be undiminished.
   Most numerical problems cannot be solved symbolically
   $\rightarrow$ make approximations.

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

3. Numerical computing will be adaptive, iterative, exploratory.

5. Importance of floatin
Most numerical probl
→ make approximati

> *"We will tell the machine what we want, and the machine, an intelligent control system sitting atop an encyclopedia of numerical methods, will juggle computational possibilities at incomprehensible speed until it has solved the problem to the accuracy required. Then it will give us the answer; and if we insist, it may even tell us something of how it got there."*

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

3. Numerical computing will be adaptive, iterative, exploratory.
   Solve problems with the required accuracy
   $\rightarrow$ guarantees/proofs about results.
5. Importance of floating-point arithmetic will be undiminished.
   Most numerical problems cannot be solved symbolically
   $\rightarrow$ make approximations.

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

3. Numerical computing will be adaptive, iterative, exploratory.
   Solve problems with the required accuracy
   $\rightarrow$ guarantees/proofs about results.
5. Importance of floating-point arithmetic will be undiminished.
   Most numerical problems cannot be solved symbolically
   $\rightarrow$ make approximations.
8. Breakthroughs will have occurred in spectral methods.

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

3. Numerical computing will be adaptive, iterative, exploratory.
   Solve problems with the required accuracy
   $\rightarrow$ guarantees/proofs about results.
5. Importance of floating-point arithmetic will be undiminished.
   Most numerical problems cannot be solved symbolically
   $\rightarrow$ make approximations.
8. Breakthroughs will have occurred in spectral methods.
   $\rightarrow$Use orthogonal series.

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

Predictions for Scientific Computing Fifty Years From Now*

1. "We" may not be here.
3. Numerical computing will be adaptive, iterative, exploratory.
   Solve problems with the required accuracy
   → guarantees/proofs about results.
5. Importance of floating-point arithmetic will be undiminished.
   Most numerical problems cannot be solved symbolically
   → make approximations.
8. Breakthroughs will have occurred in spectral methods.
   →Use orthogonal series.

*L.N. Trefethen, "Numerical Analysis and Computers - 50 Years of Progress", June 17, 1998

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

Floating point (FP) Arithmetic II

✓ 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.
  - ✓ FP programs with only these operations are deterministic
  - ✓ Accuracy and portability are improved

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.
  - ✓ FP programs with only these operations are deterministic
  - ✓ Accuracy and portability are improved
- What about standard functions ($\sin$, $\cos$, $\log$, etc.)?

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

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.
  - ✓ FP programs with only these operations are deterministic
  - ✓ Accuracy and portability are improved
- What about standard functions ($\sin$, $\cos$, $\log$, etc.)?
  - Most Mathematical Libraries (libms) do not provide correctly rounded functions, although IEEE-754-2008 recommends it.

---

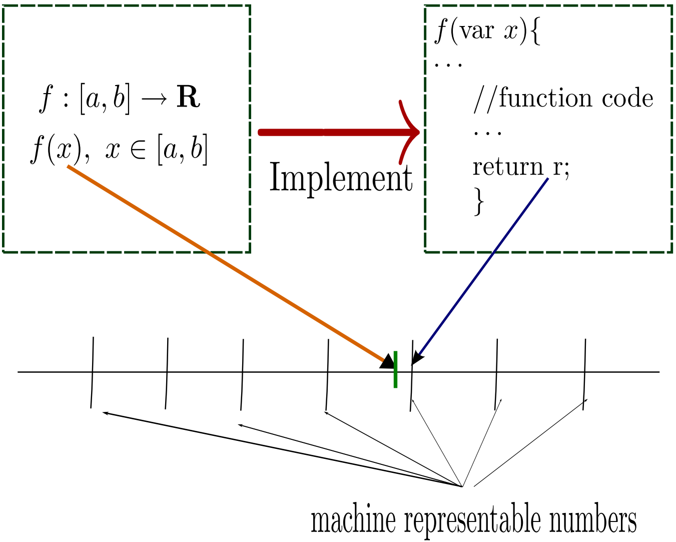
*http://lipforge.ens-lyon.fr/www/crlibm/

Floating point (FP) Arithmetic II

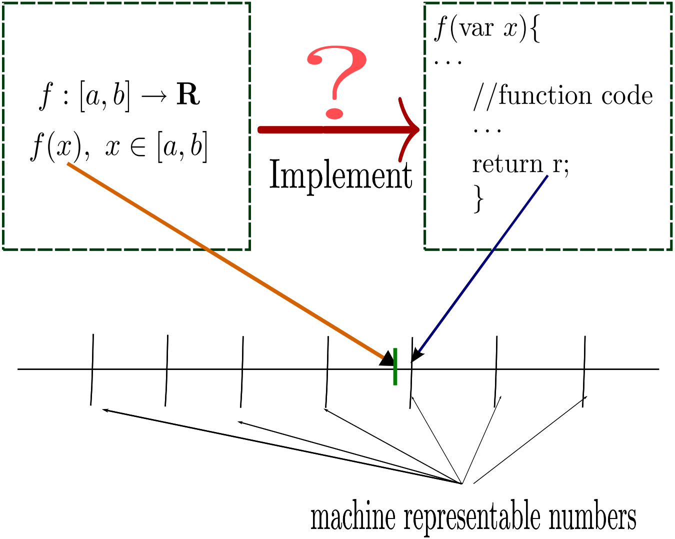
- ✓ Since 1985, IEEE-754 standard for FP arithmetic requests correct rounding for : $+, -, \times, \div, \sqrt{}$.
- ✓ Correct Rounding: An operation whose entries are FP numbers must return what we would get by infinitely precise operation followed by rounding.
  - ✓ FP programs with only these operations are deterministic
  - ✓ Accuracy and portability are improved
- What about standard functions ($\sin$, $\cos$, $\log$, etc.)?
  - Most Mathematical Libraries (libms) do not provide correctly rounded functions, although IEEE-754-2008 recommends it.
- Arénaire team develops the Correctly Rounded Libm (CRLibm*).

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

Correctly rounded functions



Correctly rounded functions



Implementation of Standard Functions

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

Goal: evaluation of φ to a given accuracy η.

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, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.

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, Daumas *et al*): evaluation of a function φ over $\mathbb{R}$ or a subset of $\mathbb{R}$ is reduced to the evaluation of a function f over $[a, b]$.
- Step 2. Computation of $p^\star$, a "machine-efficient" polynomial approximation of f (Arénaire, implementation in Sollya).

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

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

Implementation of Standard Functions

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

Goal: evaluation of φ to a given accuracy η.

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

Example of Numerical Computing Today

Compute:

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

Example of Numerical Computing Today

Compute:

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

- Maple15: 0.7499743685;

Example of Numerical Computing Today

Compute:

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

- Maple15: 0.7499743685;
- Pari/GP: $0.7927730971479080755500978354$;

Example of Numerical Computing Today

Compute:

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

- Maple15: 0.7499743685;
- Pari/GP: $0.7927730971479080755500978354$;
- Mathematica and Chebfun fail to answer;

The Mathematica Book: "the only information it has about your integrand is a sequence of numerical values for it. If you give a sufficiently pathological integrand, Mathematica may simply give you the wrong answer for the integral."

Example of Numerical Computing Today

Compute:

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

- Maple15: 0.7499743685;
- Pari/GP: $0.7927730971479080755500978354$;
- Mathematica and Chebfun fail to answer;
- Chen, '06: 0.7578918118.

Example of Numerical Computing Today

Compute:

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

- Maple15: 0.7499743685;
- Pari/GP: $0.7927730971479080755500978354$;
- Mathematica and Chebfun fail to answer;
- Chen, '06: 0.7578918118.

What is the correct answer, then ?

Computer Assisted Proofs

Formal proof of Kepler's conjecture

Prove the inequality:
$2\pi - 2x \arcsin((\cos 0.797) \sin(\pi/x)) > 2.097 - 0.0331x$,
where $3 < x < 64$.

Computer Assisted Proofs

Formal proof of Kepler's conjecture

Prove the inequality:
$2\pi - 2x \arcsin((\cos 0.797) \sin(\pi/x)) > 2.097 - 0.0331x$,
where $3 < x < 64$.

Chebfun: "It is well known that the problem of determining the sign of a difference of real numbers with guaranteed accuracy poses difficulties. However, the chebfun system makes no claim to overcome these difficulties"

Computer Assisted Proofs

Formal proof of Kepler's conjecture

Prove the inequality:
$2\pi - 2x \arcsin((\cos 0.797)\sin(\pi/x)) > 2.097 - 0.0331x$,
where $3 < x < 64$.

Chebfun: "It is well known that the problem of determining the sign of a difference of real numbers with guaranteed accuracy poses difficulties. However, the chebfun system makes no claim to overcome these difficulties"

Mathematica book: "It is therefore worthwhile to go as far as you can symbolically, and then resort to numerical methods only at the very end."

Rigorous Computing Setting

- Use Floating-Point as support for computations (fast)

Rigorous Computing Setting

- Use Floating-Point as support for computations (fast)
- Bound roundoff, discretization, truncation errors in numerical algorithms

Rigorous Computing Setting

- Use Floating-Point as support for computations (fast)
- Bound roundoff, discretization, truncation errors in numerical algorithms
- Compute enclosures instead of approximations

Rigorous Computing Setting

- Use Floating-Point as support for computations (fast)
- Bound roundoff, discretization, truncation errors in numerical algorithms
- Compute enclosures instead of approximations
- Symbolic-numeric approach to get the correct answer

Rigorous Computing Setting

- Use Floating-Point as support for computations (fast)
- Bound roundoff, discretization, truncation errors in numerical algorithms
- Compute enclosures instead of approximations
- Symbolic-numeric approach to get the correct answer

Tools:

$\rightsquigarrow$ Interval arithmetic (IA)

Interval Arithmetic (IA)

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

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

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/

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/`

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/

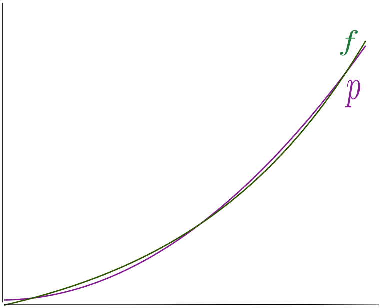
Interval Arithmetic (IA)

- 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 $\mathtt{Im}(f) = [3/4, 3]$

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

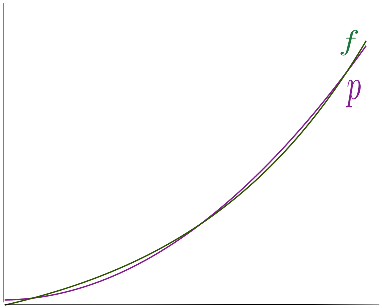
When Interval Arithmetic does not suffice: Computing supremum norms of approximation errors

$f(x) = e^{1/\cos(x)},\ x \in [0,1],\ p(x) = \sum_{i=0}^{10} c_i x^i,$



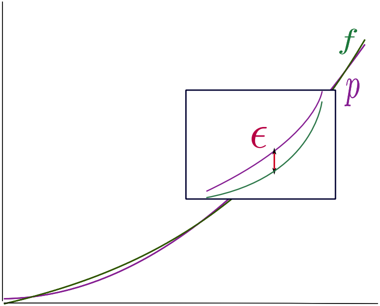
When Interval Arithmetic does not suffice: Computing supremum norms of approximation errors

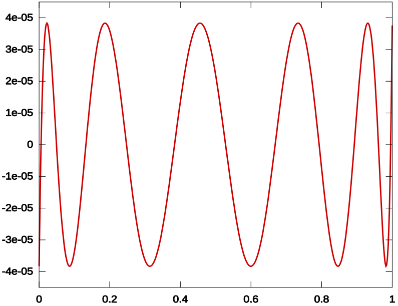
$$f(x) = e^{1/\cos(x)},\ x \in [0,1],\ p(x) = \sum_{i=0}^{10} c_i x^i,\ \varepsilon(x) = f(x) - p(x)$$



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

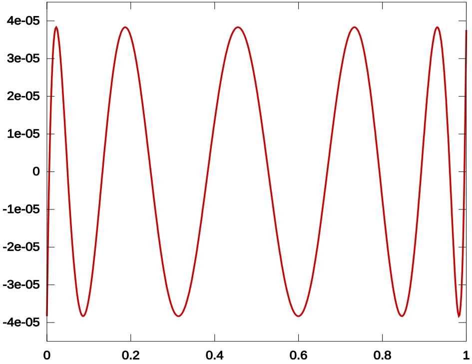
$f(x) = e^{1/\cos(x)}$, $x \in [0, 1]$, $p(x) = \sum_{i=0}^{10} c_i x^i$, $\varepsilon(x) = f(x) - p(x)$ s.t. $\|\varepsilon\|_\infty = \sup_{x \in [a,\, b]}\{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)





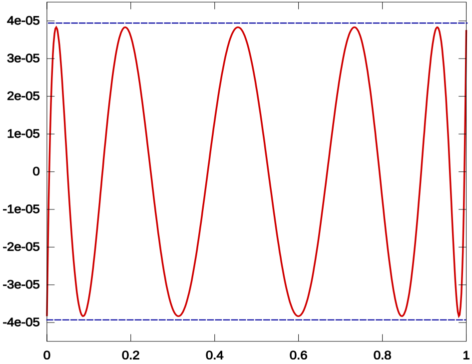
When Interval Arithmetic does not suffice: Computing supremum norms of approximation errors

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



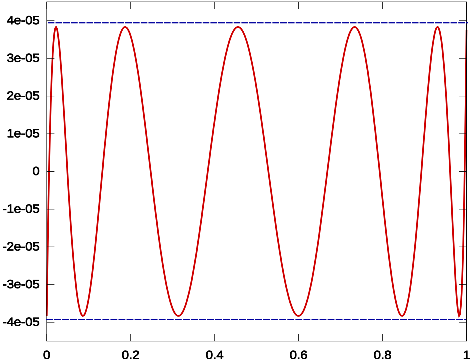
When Interval Arithmetic does not suffice: Computing supremum norms of approximation errors

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



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

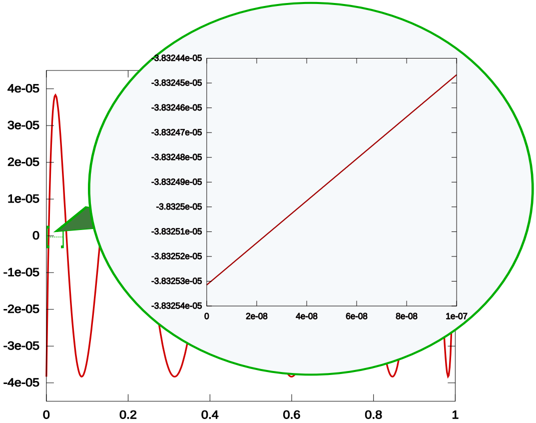
$f(x) = e^{1/\cos(x)}$, $x \in [0, 1]$, $p(x) = \sum_{i=0}^{10} c_i x^i$, $\varepsilon(x) = f(x) - p(x)$ s.t. $\|\varepsilon\|_\infty = \sup_{x \in [a,\, b]} \{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)



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.

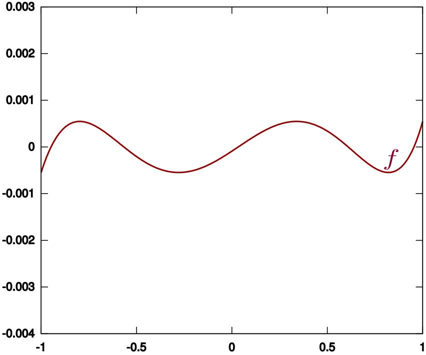


In this case, over $[0, 1]$ we need 10^7 intervals!

Why IA does not suffice: Overestimation

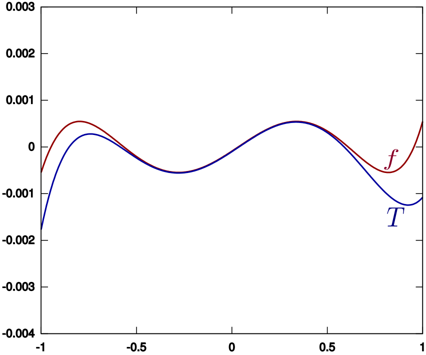
Use the appropriate tools!

Rigorous polynomial approximations (RPAs)



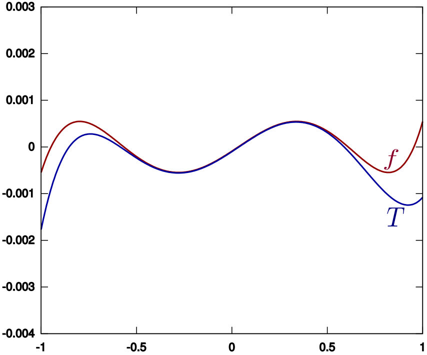
Rigorous polynomial approximations (RPAs)

f replaced with
- polynomial approximation T



Rigorous polynomial approximations (RPAs)

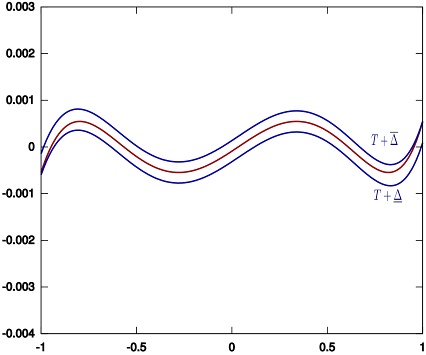
$\color{red}{f}$ replaced with
- polynomial approximation $\color{blue}{T}$
- interval $\color{blue}{\boldsymbol{\Delta}}$ s. t. $f(x) - \color{blue}{T(x)} \in \color{blue}{\boldsymbol{\Delta}}, \forall x \in [a, b]$



Rigorous polynomial approximations (RPAs)

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

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



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

Rigorous polynomial approximations (RPAs) Contributions of this thesis:

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

(1). Analyse and refine existing RPAs based on Taylor series $\rightsquigarrow$ Taylor Models (TMs).

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

(3). Bring forth 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.

Rigorous polynomial approximations (RPAs) Contributions of this thesis:

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

(1). Analyse and refine existing RPAs based on Taylor series $\rightsquigarrow$ Taylor Models (TMs).

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

(3). Bring forth 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.

Rigorous polynomial approximations (RPAs)
Contributions of this thesis:

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

(1). Analyse and refine existing RPAs based on Taylor series $\rightsquigarrow$ Taylor Models (TMs).

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

(3). Bring forth 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.

Rigorous polynomial approximations (RPAs) Contributions of this thesis:

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

(1). Analyse and refine existing RPAs based on Taylor series $\rightsquigarrow$ Taylor Models (TMs).

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

(3). Bring forth 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.

Taylor Models

- Consider Taylor approximations

Taylor Models

- Consider Taylor approximations

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

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

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

Taylor Models

- Consider Taylor approximations

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

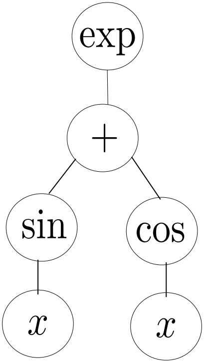
$$f(x) = \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!}}_{T(x)} + \underbrace{\Delta_n(x)}_{\boldsymbol{\Delta}}$$

- 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.

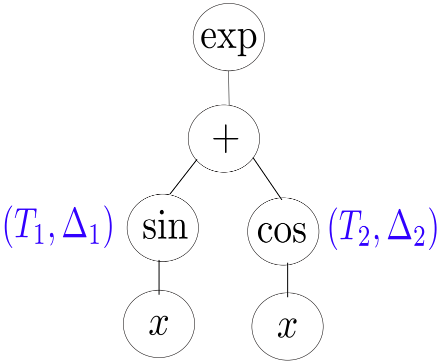
Taylor Models $\rightsquigarrow$ Algebra of RPAs

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



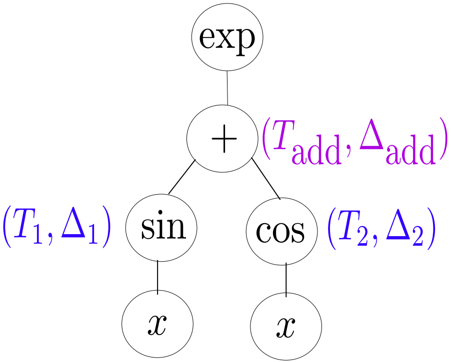
Taylor Models $\rightsquigarrow$ Algebra of RPAs

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



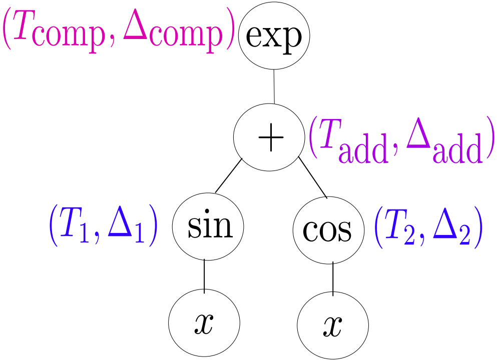
Taylor Models $\rightsquigarrow$ Algebra of RPAs

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



Taylor Models $\rightsquigarrow$ Algebra of RPAs

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

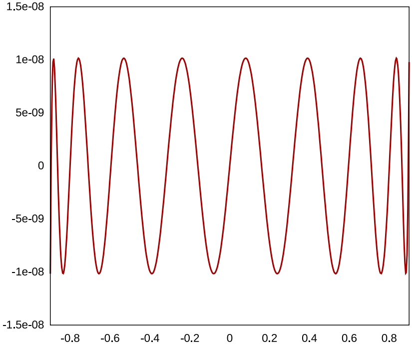
- Automatic differentiation and Lagrange formula:
  $\boldsymbol{\Delta} = [-1.93 \cdot 10^{2}, 1.35 \cdot 10^{3}]$
- Cauchy's Estimate
  $\boldsymbol{\Delta} = [-9.17 \cdot 10^{-2}, 9.17 \cdot 10^{-2}]$
- Taylor Models
  $\boldsymbol{\Delta} = [-9.04 \cdot 10^{-3}, 9.06 \cdot 10^{-3}]$

Supremum norm example

Example:

$f(x) = \arctan(x)$ over $[-0.9, 0.9]$
$p(x)$ - minimax, degree 15
$\varepsilon(x) = p(x) - f(x)$

$\|\varepsilon\|_\infty \simeq 10^{-8}$



Supremum norm 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: we need theoretically a TM of degree 120.

Supremum norm 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: we need theoretically a TM of degree 120. Practically, the computed interval error bound can not be made sufficiently small due to overestimation.

Improvement?

- Use a polynomial approximation better than Taylor:
  - Why?
    – better convergence domains
    – compact approximations needed in formal proofs

Improvement?

- Use a polynomial approximation better than Taylor:
  - Why?
    - better convergence domains
    - compact approximations needed in formal proofs
  - How?
    - fast coefficients computation
    - closed truncation error formula

Our Approach - Chebyshev Models

Basic idea:

- Use a polynomial approximation better than Taylor:

- Chebyshev Interpolants.
- Chebyshev Truncated Series.

.

Our Approach - Chebyshev Models

Basic idea:

- Use a polynomial approximation better than Taylor:
  - Chebyshev Interpolants.
  - Chebyshev Truncated Series.
- Use the two step approach as Taylor Models:
  - compute models $(P, \mathbf{\Delta})$ for basic functions;
  - apply algebraic rules with these models, instead of operations with the corresponding functions.

.

Our Approach - Chebyshev Models

Basic idea:

- Use a polynomial approximation better than Taylor:
  - Chebyshev Interpolants.
  - Chebyshev Truncated Series.
- Use the two step approach as Taylor Models:
  - compute models $(P, \mathbf{\Delta})$ for basic functions;
  - apply algebraic rules with these models, instead of operations with the corresponding functions.

Note: Chebfun - "Computing Numerically with Functions Instead of Numbers" (N. Trefethen et al.): Chebyshev interpolation polynomials are already used, but the approach is not rigorous.

Our Approach - Chebyshev Models

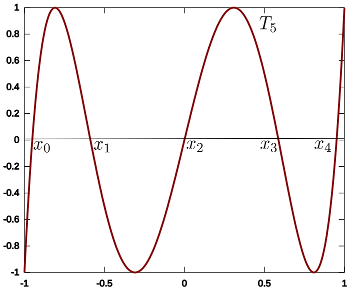
Basic idea:

- Use a polynomial approximation better than Taylor:
  - Chebyshev Interpolants.
  - Chebyshev Truncated Series.
- Use the two step approach as Taylor Models:
  - compute models $(P, \boldsymbol{\Delta})$ for basic functions;
  - apply algebraic rules with these models, instead of operations with the corresponding functions.

Note: Chebfun - "Computing Numerically with Functions Instead of Numbers" (N. Trefethen et al.): Chebyshev interpolation polynomials are already used, but the approach is not rigorous.

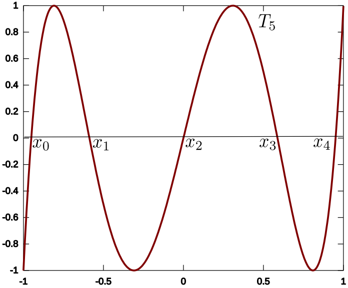
Quick Reminder: Chebyshev Polynomials

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



Quick Reminder: Chebyshev Polynomials

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

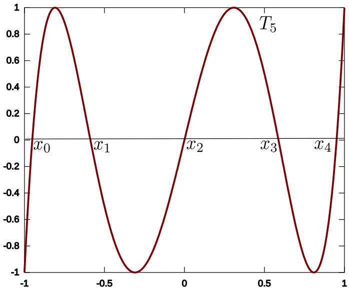


Chebyshev nodes: n distinct real roots in $[-1, 1]$ of T_n

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

Quick Reminder: Chebyshev Polynomials

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



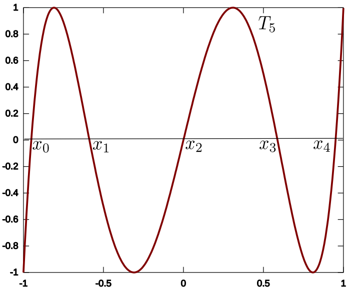
$$T_{i+1} = 2xT_i - T_{i-1}, T_0(x) = 1, T_1(x) = x$$

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:

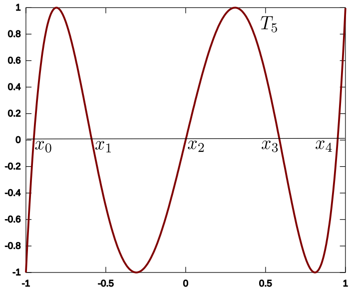
$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & \text{if } i \neq j \\ \pi & \text{if } i = 0 \\ \frac{\pi}{2} & \text{otherwise} \end{cases}$$

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

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

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:

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

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

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 I

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 I

Two approximations of f:

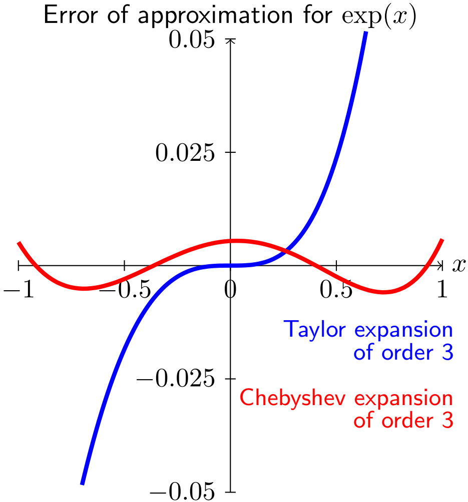
- by Taylor series

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

- or by Chebyshev series

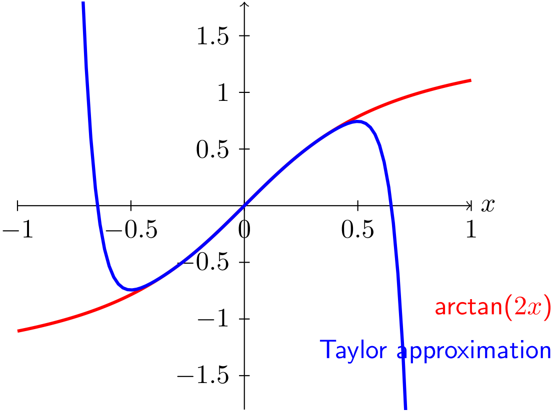
$$f = \sum_{n=-\infty}^{+\infty} t_n T_n(x),$$

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

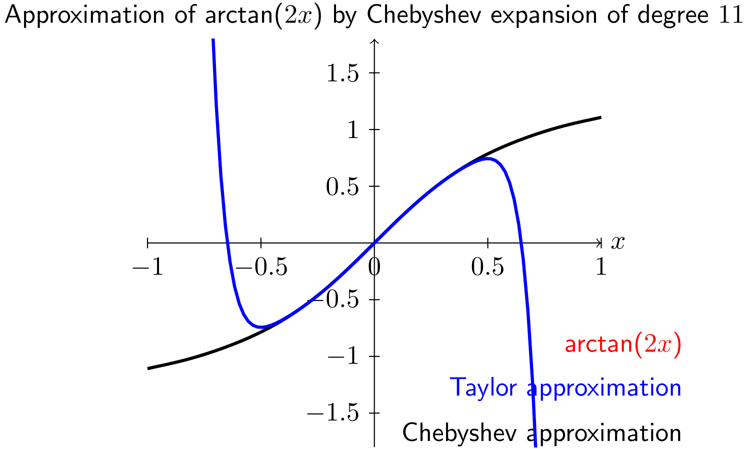


Chebyshev Series vs Taylor Series II

Bad approximation outside its circle of convergence



Chebyshev Series vs Taylor Series II

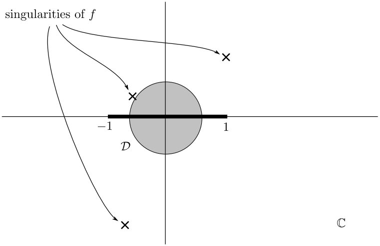


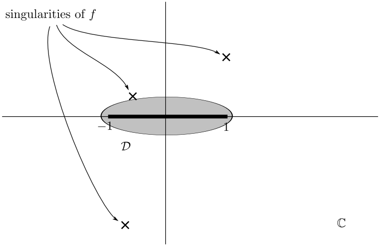
Chebyshev Series vs Taylor Series III

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

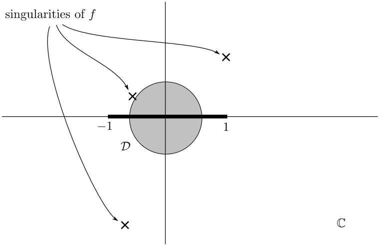




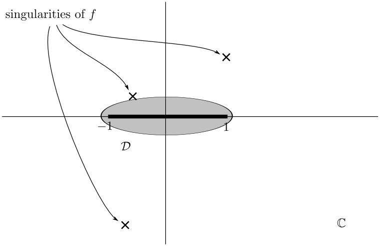
Chebyshev Series vs Taylor Series III

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

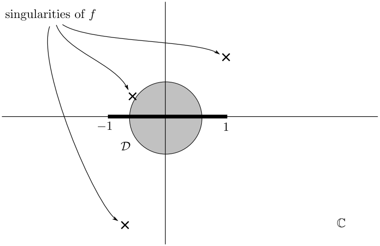


- Taylor series can not converge over entire $[-1, 1]$ unless all singularities lie outside the unit circle.

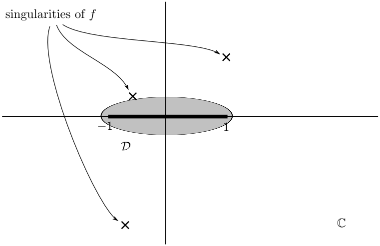
Chebyshev Series vs Taylor Series III

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



- 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 $[-1, 1]$.

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 $\|f - p\|_\infty = \sup_{-1 \leq x \leq 1} |f(x) - p(x)|$.

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

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 $\|f - p\|_\infty = \sup_{-1 \leq x \leq 1} |f(x) - p(x)|$.

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

- $\Lambda_{10} = 2.22... \rightarrow$ 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 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:

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

Chebyshev series, Bernstein-like formula:

$$\forall x \in [-1,1],\ \exists \xi \in [-1,1] \ \text{ s.t.}$$
$$f(x) - P(x) = \frac{f^{(n+1)}(\xi)}{2^n(n+1)!}.$$

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

Computing the coefficients

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

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

Computing the coefficients

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

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

– Discrete orthogonality $\rightsquigarrow \widetilde{t_i} = \sum_{k=0}^{n} \frac{1}{n+1} f(x_k) T_i(x_k)$

Computing the coefficients

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

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

– Discrete orthogonality $\rightsquigarrow$ $\widetilde{t_i} = \sum_{k=0}^{n} \frac{1}{n+1} f(x_k) T_i(x_k)$

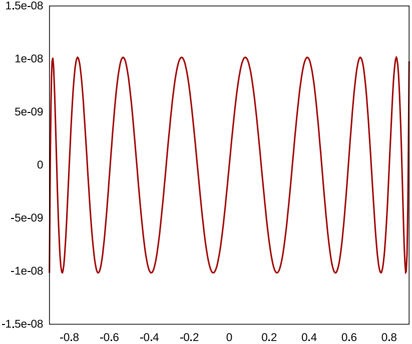
$\rightsquigarrow$ Chebyshev Interpolant (CI)

Chebyshev Models with CI: Supremum norm example

Example:

$f(x) = \arctan(x)$ over $[-0.9, 0.9]$
$p(x)$ - minimax, degree 15
$\varepsilon(x) = p(x) - f(x)$

$\|\varepsilon\|_\infty \simeq 10^{-8}$



Chebyshev Models with CI: Supremum norm 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: we need theoretically a TM of degree 120. Practically, the computed interval error bound can not be made sufficiently small due to overestimation.

Chebyshev Models with CI: Supremum norm 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: we need theoretically a TM of degree 120. Practically, the computed interval error bound can not be made sufficiently small due to overestimation.

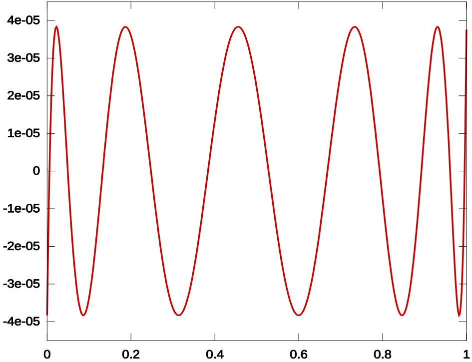
A CM of degree 60 works.

Chebyshev Models with CI: Supremum norm example

Example: $\varepsilon(x) = f(x) - p(x)$

$f(x) = e^{1/\cos x}$, over $[0, 1]$, $p(x)$ - minimax, degree 10

$\|\varepsilon(x)\|_\infty \simeq 3.8325 \cdot 10^{-5}$



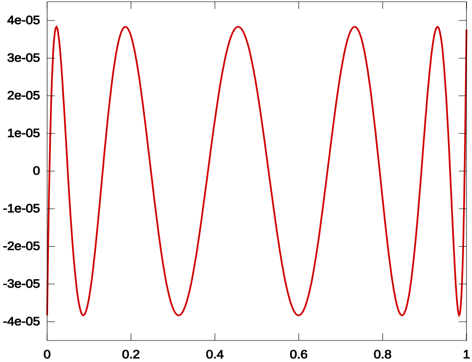
Chebyshev Models with CI: Supremum norm example

Example: $\varepsilon(x) = f(x) - p(x)$

$f(x) = e^{1/\cos x}$, over $[0, 1]$, $p(x)$ - minimax, degree 10

$\|\varepsilon(x)\|_\infty \simeq 3.8325 \cdot 10^{-5}$

Need: TM of degree 30.



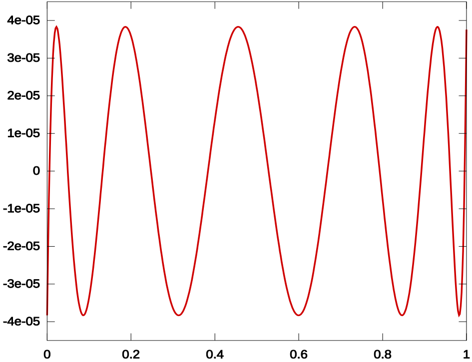
Chebyshev Models with CI: Supremum norm example

Example: $\varepsilon(x) = f(x) - p(x)$

$f(x) = e^{1/\cos x}$, over $[0, 1]$, $p(x)$ - minimax, degree 10

$\|\varepsilon(x)\|_\infty \simeq 3.8325 \cdot 10^{-5}$

Need: TM of degree 30.
CM of degree 13.



Example:

Compute:

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

- Maple15: 0.7499743685;
- Pari/GP: $0.7927730971479080755500978354$;
- Mathematica and Chebfun fail to answer;
- Chen, '06: 0.7578918118.

What is the correct answer, then ?
Using CMs: $0.749974368527[1,3]$.

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

Computing the coefficients

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

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

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

$\rightsquigarrow$ 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: Currently, this step is more costly than in the case of TMs. Use Truncated Chebyshev Series (TCS) instead: for D-finite functions.

D-finite Functions

A function $y : \mathbb{R} \to \mathbb{R}$ is D-finite if it is solution of a (homogeneous) linear differential equation with polynomial coefficients:

$$L \cdot y = a_r y^{(r)} + a_{r-1} y^{(r-1)} + \cdots + a_0 y = 0, \quad a_i \in \mathbb{Q}[x].$$

D-finite Functions

A function $y : \mathbb{R} \to \mathbb{R}$ is D-finite if it is solution of a (homogeneous) linear differential equation with polynomial coefficients:

$$L \cdot y = a_r y^{(r)} + a_{r-1} y^{(r-1)} + \cdots + a_0 y = 0, \quad a_i \in \mathbb{Q}[x].$$

Example 1

$$f(x) = \exp(x) \quad \leftrightarrow \quad \{f' - f = 0,\ f(0) = 1\}.$$

D-finite Functions

A function $y : \mathbb{R} \to \mathbb{R}$ is D-finite if it is solution of a (homogeneous) linear differential equation with polynomial coefficients:

$$L \cdot y = a_r y^{(r)} + a_{r-1} y^{(r-1)} + \cdots + a_0 y = 0, \quad a_i \in \mathbb{Q}[x].$$

Example 1

$f(x) = \exp(x) \quad \leftrightarrow \quad \{f' - f = 0,\ f(0) = 1\}.$
$\cos$, $\arccos$, Airy functions, Bessel functions, ...

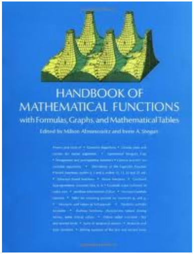
D-finite Functions

A function $y : \mathbb{R} \to \mathbb{R}$ is D-finite if it is solution of a (homogeneous) linear differential equation with polynomial coefficients:

$$L \cdot y = a_r y^{(r)} + a_{r-1} y^{(r-1)} + \cdots + a_0 y = 0, \quad a_i \in \mathbb{Q}[x].$$

Example 1

$f(x) = \exp(x) \quad \leftrightarrow \quad \{f' - f = 0,\ f(0) = 1\}.$
$\cos$, $\arccos$, Airy functions, Bessel functions, ...
About 60% of Abramowitz & Stegun



D-finite Functions

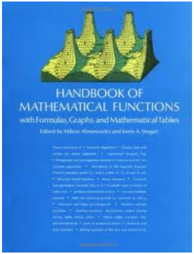
A function $y : \mathbb{R} \to \mathbb{R}$ is D-finite if it is solution of a (homogeneous) linear differential equation with polynomial coefficients:

$$L \cdot y = a_r y^{(r)} + a_{r-1} y^{(r-1)} + \cdots + a_0 y = 0, \quad a_i \in \mathbb{Q}[x].$$

Differential equation + initial conditions = Data Structure

Example 1

$f(x) = \exp(x) \quad \leftrightarrow \quad \{f' - f = 0,\ f(0) = 1\}$.
$\cos$, $\arccos$, Airy functions, Bessel functions, ...
About 60% of Abramowitz & Stegun



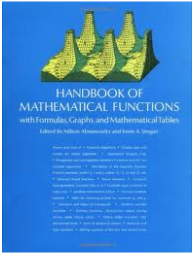
D-finite Functions

A function $y : \mathbb{R} \to \mathbb{R}$ is D-finite if it is solution of a (homogeneous) linear differential equation with polynomial coefficients:

$$L \cdot y = a_r y^{(r)} + a_{r-1} y^{(r-1)} + \cdots + a_0 y = 0, \quad a_i \in \mathbb{Q}[x].$$

Differential equation $+$ initial conditions $=$ Data Structure

How to obtain a near-minimax RPA of degree d for a D-finite function in $O(d)$?



Chebyshev Series of D-finite Functions

Theorem 1 (60's, BenoitJoldesMezzarobba11)

$\sum u_n T_n(x)$ *is solution of a linear differential equation with polynomial coefficients iff the sequence* u_n *is cancelled by a linear recurrence with polynomial coefficients.*

Chebyshev Series of D-finite Functions

Theorem 1 (60's, BenoitJoldesMezzarobba11)

$\sum u_n T_n(x)$ is solution of a linear differential equation with polynomial coefficients iff the sequence u_n is cancelled by a linear recurrence with polynomial coefficients.

Recurrence relation + good initial conditions $\Rightarrow$ Fast numerical computation of the coefficients

Taylor: $\exp = \sum \frac{1}{n!} x^n$
Rec: $u(n+1) = \frac{u(n)}{n+1}$

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

Chebyshev Series of D-finite Functions

Theorem 1 (60's, BenoitJoldesMezzarobba11)

$\sum u_n T_n(x)$ is solution of a linear differential equation with polynomial coefficients iff the sequence u_n is cancelled by a linear recurrence with polynomial coefficients.

Recurrence relation + good initial conditions $\Rightarrow$ Fast numerical computation of the coefficients

Taylor: $\exp = \sum \frac{1}{n!} x^n$
Rec: $u(n+1) = \frac{u(n)}{n+1}$

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

Chebyshev: $\exp = \sum I_n(1) T_n(x)$
Rec: $u(n+1) = -2nu(n) + u(n-1)$

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

Chebyshev Series of D-finite Functions

Theorem 1 (60's, BenoitJoldesMezzarobba11)

$\sum u_n T_n(x)$ is solution of a linear differential equation with polynomial coefficients iff the sequence u_n is cancelled by a linear recurrence with polynomial coefficients.

Recurrence relation + good initial conditions $\Rightarrow$ Fast numerical computation of the coefficients

Taylor: $\exp = \sum \frac{1}{n!} x^n$
Rec: $u(n+1) = \frac{u(n)}{n+1}$

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

Chebyshev: $\exp = \sum I_n(1) T_n(x)$
Rec: $u(n+1) = -2nu(n) + u(n-1)$

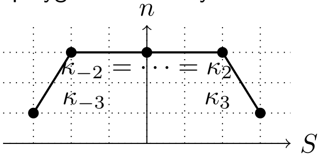
$u(0) = 1.266$	$I_0(1) \approx 1.266$
$u(1) = 0.565$	$I_1(1) \approx 0.565$
$u(2) \approx 0.136$	$I_2(1) \approx 0.136$
$\vdots$	$\vdots$
$u(50) \approx 4.450 \cdot 10^{67}$	$I_{50}(1) \approx 2.934 \cdot 10^{-80}$

Convergent and Divergent Solutions of the Recurrence

Study of the Chebyshev recurrence

If $u(n)$ is solution, then there exists another solution $v(n) \sim \frac{1}{u(n)}$

Newton polygon of a Chebyshev recurrence

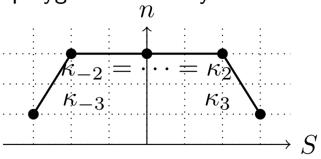


Convergent and Divergent Solutions of the Recurrence

Study of the Chebyshev recurrence

If $u(n)$ is solution, then there exists another solution $v(n) \sim \frac{1}{u(n)}$

Newton polygon of a Chebyshev recurrence



For the recurrence $u(n+1) + 2nu(n) - u(n-1)$

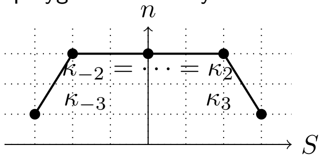
Two independent solutions are $I_n(1) \sim \frac{1}{(2n)!}$ and $K_n(1) \sim (2n)!$

Convergent and Divergent Solutions of the Recurrence

Study of the Chebyshev recurrence

If $u(n)$ is solution, then there exists another solution $v(n) \sim \frac{1}{u(n)}$

Newton polygon of a Chebyshev recurrence



For the recurrence $u(n+1) + 2nu(n) - u(n-1)$

Two independent solutions are $I_n(1) \sim \frac{1}{(2n)!}$ and $K_n(1) \sim (2n)!$

Miller's algorithm

Compute the first N coefficients of the most convergent solution of a recurrence relation of order 2

- Initialize $u(N) = 0$ and $u(N-1) = 1$ and compute the first coefficients using the recurrence backwards
- Normalize u with the initial condition of the recurrence

Example: Back to $\exp$

$$\begin{aligned} u(52) &= 0 \\ u(51) &= 1 \\ u(50) &= -102 \\ &\vdots \\ u(2) &\approx -4.72 \cdot 10^{80} \\ u(1) &\approx 1.96 \cdot 10^{81} \\ u(0) &\approx -4.4 \cdot 10^{81} \end{aligned}$$

$$\begin{aligned} I_{52}(1) &\approx 2.77 \cdot 10^{-84} \\ I_{51}(1) &\approx 2.88 \cdot 10^{-82} \\ I_{50}(1) &\approx 2.93 \cdot 10^{-80} \\ &\vdots \\ I_2(1) &\approx 0.14 \\ I_1(1) &\approx -0.57 \\ I_0(1) &\approx 1.27 \end{aligned}$$

Example: Back to $\exp$

$$
\begin{aligned}
u(52) &= 0 & I_{52}(1) &\approx 2.77 \cdot 10^{-84} \\
u(51) &= 1 & I_{51}(1) &\approx 2.88 \cdot 10^{-82} \\
u(50) &= -102 & I_{50}(1) &\approx 2.93 \cdot 10^{-80} \\
&\vdots & &\vdots \\
u(2) &\approx -4.72 \cdot 10^{80} & I_2(1) &\approx 0.14 \\
u(1) &\approx 1.96 \cdot 10^{81} & I_1(1) &\approx -0.57 \\
u(0) &\approx -4.4 \cdot 10^{81} & I_0(1) &\approx 1.27
\end{aligned}
$$

$$
C = \sum_{n=-50}^{50} u(n) T_n(0) \approx -3.48 \cdot 10^{81}
$$

Example: Back to $\exp$

$$\frac{u(52)}{C} = 0 \qquad\qquad I_{52}(1) \approx 2.77 \cdot 10^{-84}$$

$$\frac{u(51)}{C} \approx -2.88 \cdot 10^{-82} \qquad\qquad I_{51}(1) \approx 2.88 \cdot 10^{-82}$$

$$\frac{u(50)}{C} \approx 2.93 \cdot 10^{-80} \qquad\qquad I_{50}(1) \approx 2.93 \cdot 10^{-80}$$

$$\vdots \qquad\qquad \vdots$$

$$\frac{u(2)}{C} \approx 0.14 \qquad\qquad I_2(1) \approx 0.14$$

$$\frac{u(1)}{C} \approx -0.57 \qquad\qquad I_1(1) \approx -0.57$$

$$\frac{u(0)}{C} \approx 1.27 \qquad\qquad I_0(1) \approx 1.27$$

$$C = \sum_{n=-50}^{50} u(n) T_n(0) \approx -3.48 \cdot 10^{81}$$

Validation

Fixed Point Theorem Applied to a Differential Equation: f is solution of

$$y'(x) - a(x)y(x) = 0, \text{ with } y(0) = y_0,$$

if and only if f is a fixed point of τ defined by

$$\tau(y)(t) = y_0 + \int_0^t a(x)y(x)dx.$$

Validation

Fixed Point Theorem Applied to a Differential Equation: f is solution of

$$y'(x) - a(x)y(x) = 0, \text{ with } y(0) = y_0,$$

if and only if f is a fixed point of τ defined by

$$\tau(y)(t) = y_0 + \int_0^t a(x)y(x)dx.$$

For all rational functions $a(x)$, if $\dfrac{\|a\|_\infty^j}{j!} < 1$ then $\forall i \geqslant j$, τ^i is a contraction map.

Algorithm for a Differential Equation of Order 1

Given p, compute a sharp bound B such that $|f(x) - p(x)| < B,\ x \in [-1, 1]$.

Find B

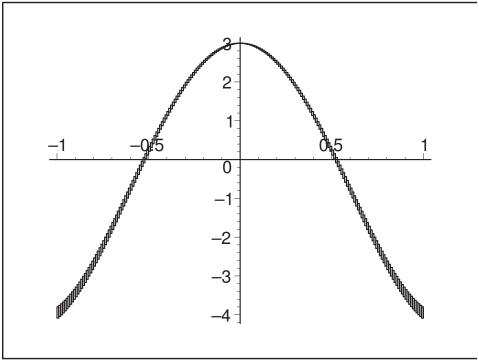
- $p_0 := p$
- while $i! < \|a\|_\infty^i$
  - Compute $p_i(t)$ a rigorous approximation of $\tau(p_{i-1}) = \int_0^t a(x) p_{i-1}(x) dx$ s.t. $\|\tau(p_{i-1}) - p_i\|_\infty < M$.
- Return

$$B = \frac{\|p_i - p\|_\infty + M \sum_{j=1}^{i} \frac{\|a\|_\infty^{j-1}}{j!}}{1 - \frac{\|a\|_\infty^i}{i!}}$$

Certified Plots

Random Example:

$(60x^2 + 75 + 9x^4)y(x) + (-4x^3 - 12x)y'(x) + (x^4 + 6x^2 + 9)y''(x)$,
$y(0) = 3,\ y'(0) = 0.$
Compute coefficients of polynomial of degree 30.



Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models

Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models
  - ✓ CMs for univariate composite functions are more efficient than TMs.

Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models
  - ✓ CMs for univariate composite functions are more efficient than TMs.
    $\rightsquigarrow$ What about multivariate functions?

Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models
  - ✓ CMs for univariate composite functions are more efficient than TMs.
    $\rightsquigarrow$ What about multivariate functions?
    $\rightsquigarrow$ Improving complexity of RPAs' Algebra: use and adaptation of Fast Fourier Transform to IA.

Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models
  - ✓ CMs for univariate composite functions are more efficient than TMs.
    $\rightsquigarrow$ What about multivariate functions?
    $\rightsquigarrow$ Improving complexity of RPAs' Algebra: use and adaptation of Fast Fourier Transform to IA.
- From rigorous to formal proofs:
  $\rightsquigarrow$ complete formalization of TMs and CMs in a formal proof checker.

Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models
  - ✓ CMs for univariate composite functions are more efficient than TMs.
    $\rightsquigarrow$ What about multivariate functions?
    $\rightsquigarrow$ Improving complexity of RPAs' Algebra: use and adaptation of Fast Fourier Transform to IA.
- From rigorous to formal proofs:
  $\rightsquigarrow$ complete formalization of TMs and CMs in a formal proof checker.
- We presented a linear complexity near-minimax RPAs for D-finite functions.
  $\rightsquigarrow$ Use that as a basic brick for rigorously solving non-linear ODEs.

Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models
  - ✓ CMs for univariate composite functions are more efficient than TMs.
    $\rightsquigarrow$ What about multivariate functions?
    $\rightsquigarrow$ Improving complexity of RPAs' Algebra: use and adaptation of Fast Fourier Transform to IA.
- From rigorous to formal proofs:
  $\rightsquigarrow$ complete formalization of TMs and CMs in a formal proof checker.
- We presented a linear complexity near-minimax RPAs for D-finite functions.
  $\rightsquigarrow$ Use that as a basic brick for rigorously solving non-linear ODEs.

$\rightsquigarrow$ What about other families of orthogonal polynomials?

Conclusion and Perspectives

- We presented a new rigorous computing tool: Chebyshev Models
  - ✓ CMs for univariate composite functions are more efficient than TMs.
    $\rightsquigarrow$ What about multivariate functions?
    $\rightsquigarrow$ Improving complexity of RPAs' Algebra: use and adaptation of Fast Fourier Transform to IA.
- From rigorous to formal proofs:
  $\rightsquigarrow$ complete formalization of TMs and CMs in a formal proof checker.
- We presented a linear complexity near-minimax RPAs for D-finite functions.
  $\rightsquigarrow$ Use that as a basic brick for rigorously solving non-linear ODEs.

$\rightsquigarrow$ What about other families of orthogonal polynomials?

$\rightsquigarrow$ What about the 12 predictions and beyond?

No	$f(x)$, I, n	CM bound	TM bound
1	$\sin(x)$, $[3,\ 4]$, 10	$1.19 \cdot 10^{-14}$	$\mathbf{5.95 \cdot 10^{-15}}$
2	$\arctan(x)$, $[-0.25,\ 0.25]$, 15	$7.89 \cdot 10^{-15}$	$\mathbf{1.06 \cdot 10^{-15}}$
3	$\arctan(x)$, $[-0.9,\ 0.9]$, 15	$5.10 \cdot 10^{-3}$	$\mathbf{5.81 \cdot 10^{-4}}$
4	$\exp(1/\cos(x))$, $[0,\ 1]$, 14	$\mathbf{5.22 \cdot 10^{-7}}$	$1.10 \cdot 10^{-5}$
5	$\frac{\exp(x)}{\log(2+x)\ \cos(x)}$, $[0,\ 1]$, 15	$\mathbf{4.86 \cdot 10^{-9}}$	$4.60 \cdot 10^{-8}$
6	$\sin(\exp(x))$,$[-1,\ 1]$, 10	$\mathbf{2.56 \cdot 10^{-5}}$	$1.01 \cdot 10^{-4}$
7	$\tanh(x + 0.5) - \tanh(x - 0.5)$, $[-1, 1]$, 10	$1.75 \cdot 10^{-3}$	$\mathbf{7.28 \cdot 10^{-4}}$
8	$\sqrt{x + 1.0001}$, $[-1, 0]$, 10	$\mathbf{3.64 \cdot 10^{-2}}$	0.11
9	$\sqrt{x + 1.0001} \cdot \sin(x)$, $[-1, 0]$, 10	$\mathbf{3.32 \cdot 10^{-2}}$	$7.06 \cdot 10^{-2}$
10	$\frac{1}{1+4x^2}$, $[-1, 1]$, 10	$\mathbf{1.13 \cdot 10^{-2}}$	$1.39 \cdot 10^{2}$
11	$\sin^2(x) + \cos^2(x)$, $[-1, 1]$, 10	$\mathbf{3.91 \cdot 10^{-9}}$	$2.23 \cdot 10^{-8}$

Table: Examples of bounds *1 CM vs. 2 TMs*.