Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing

Nicolas Brisebarre Mioara Joldes

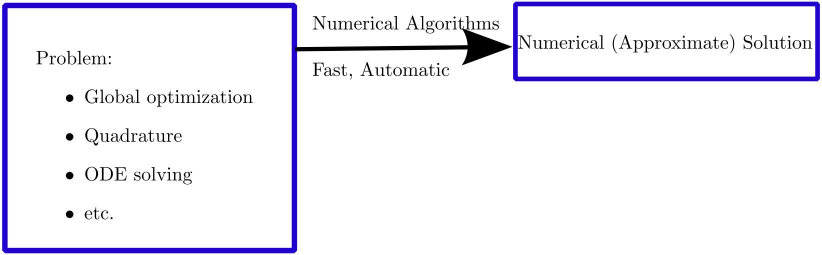
April 9, 2010

Motivation

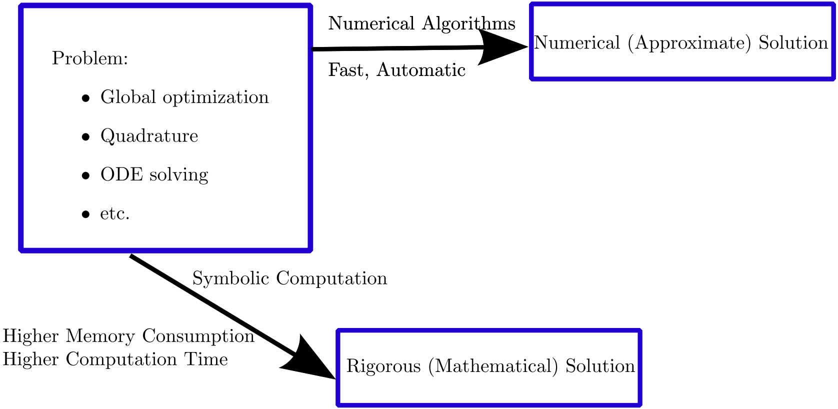
Problem:

- Global optimization
- Quadrature
- ODE solving
- etc.

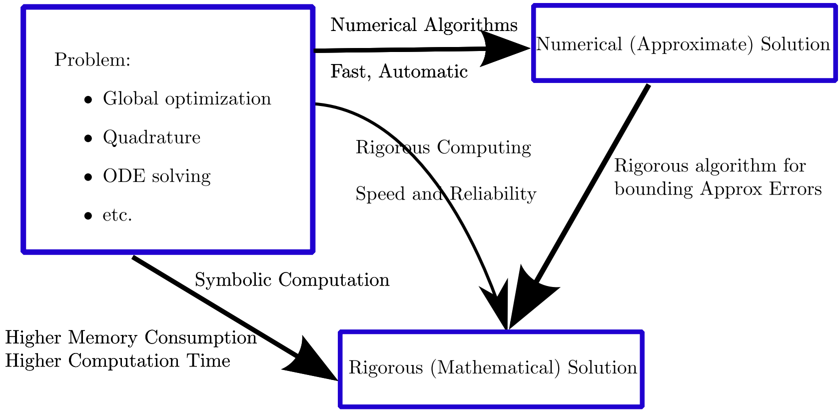
Motivation


Problem:
Global optimization
Quadrature
ODE solving
etc.
Numerical Algorithms
Fast, Automatic
Numerical (Approximate) Solution

Motivation


Problem:
- Global optimization
- Quadrature
- ODE solving
- etc.
Numerical Algorithms
Fast, Automatic
Numerical (Approximate) Solution
Symbolic Computation
Higher Memory Consumption
Higher Computation Time
Rigorous (Mathematical) Solution

Motivation



Problem:
Global optimization
Quadrature
ODE solving
etc.
Numerical Algorithms
Fast, Automatic
Numerical (Approximate) Solution
Rigorous Computing
Speed and Reliability
Rigorous algorithm for
bounding Approx Errors
Symbolic Computation
Higher Memory Consumption
Higher Computation Time
Rigorous (Mathematical) Solution

Rigorous computing tools

- Why?
  - Get the correct answer, not an "almost" correct one
  - Bridge the gap between scientific computing and pure mathematics - speed and reliability

Rigorous computing tools

- Why?
  - Get the correct answer, not an "almost" correct one
  - Bridge the gap between scientific computing and pure mathematics - speed and reliability
- How?
  - Use Floating-Point as support for computations (fast)
  - Bound roundoff, discretization, truncation errors in numerical algorithms
  - Compute enclosures instead of approximations

Rigorous computing tools

- Why?
  - Get the correct answer, not an ”almost” correct one
  - Bridge the gap between scientific computing and pure mathematics - speed and reliability
- How?
  - Use Floating-Point as support for computations (fast)
  - Bound roundoff, discretization, truncation errors in numerical algorithms
  - Compute enclosures instead of approximations
- What?
  1. Interval arithmetic (IA)
  2. Taylor models (TM)

- Where? Beam Physics (M. Berz, K. Makino), Lorentz attractor (W. Tucker), Flyspeck project (R. Zumkeller)

Rigorous computing tools

- Why?
  - Get the correct answer, not an ''almost'' correct one
  - Bridge the gap between scientific computing and pure mathematics - speed and reliability
- How?
  - Use Floating-Point as support for computations (fast)
  - Bound roundoff, discretization, truncation errors in numerical algorithms
  - Compute enclosures instead of approximations
- What?
  1. Interval arithmetic (IA)
  2. Taylor models (TM)
  3. Chebyshev models (CM)
- Where? Beam Physics (M. Berz, K. Makino), Lorentz attractor (W. Tucker), Flyspeck project (R. Zumkeller)

What kind of problems can we (CM) address ?

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

What kind of problems can we (CM) address ?

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

Practical Examples:

- Computing supremum norms of approximation error functions:

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

  where g is a very good approximation of f.
- Rigorous quadrature:

$$\pi = \int_0^1 \frac{4}{1+x^2} \mathrm{d}x$$

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
- $\pi \in [3.1415, 3.1416]$

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1, 2] + [-3, 2] = [-2, 4]$

Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1,2] + [-3,2] = [-2,4]$
- Range bounding for functions
  Eg. $x \in [-1,2], f(x) = x^2 + x + 1$
  $F(X) = X^2 + X + 1$
  $F([-1,2]) = [-1,2]^2 + [-1,2] + [1,1]$
  $F([-1,2]) = [0,4] + [-1,2] + [1,1]$
  $F([-1,2]) = [0,7]$

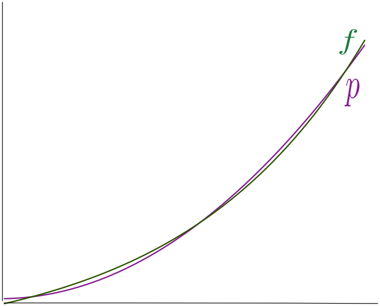
Interval Arithmetic (IA)

- Each interval = pair of floating-point numbers
- $\pi \in [3.1415, 3.1416]$
- Interval Arithmetic Operations
  Eg. $[1, 2] + [-3, 2] = [-2, 4]$
- Range bounding for functions
  Eg. $x \in [-1, 2], f(x) = x^2 + x + 1$
  $F(X) = X^2 + X + 1$
  $F([-1, 2]) = [-1, 2]^2 + [-1, 2] + [1, 1]$
  $F([-1, 2]) = [0, 4] + [-1, 2] + [1, 1]$
  $F([-1, 2]) = [0, 7]$
  $x \in [-1, 2], f(x) \in [0, 7]$, but $\mathtt{Im}(f) = [3/4, 7]$

When IA does not suffice: Computing supremum norms of approximation errors

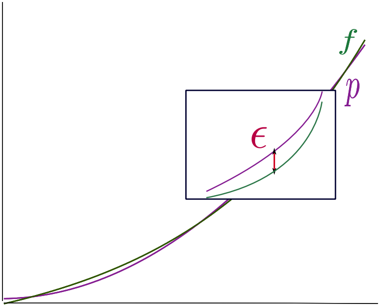
When IA does not suffice:
Computing supremum norms of approximation errors

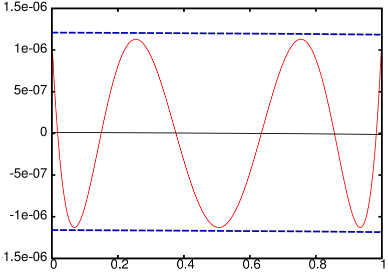
$$f(x) = e^x,\ x \in [0,1],\ \ p(x) = \sum_{i=0}^{5} c_i x^i,\ \ \varepsilon(x) = f(x) - p(x)$$



When IA does not suffice: Computing supremum norms of approximation errors

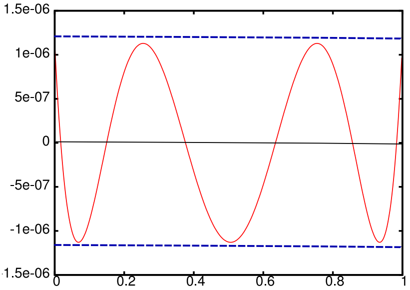
$f(x) = e^x,\ x \in [0,1],\ \ p(x) = \sum_{i=0}^{5} c_i x^i,\ \ \varepsilon(x) = f(x) - p(x)$
s.t. $\|\varepsilon\|_\infty = \sup_{x\in[a,b]}\{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)





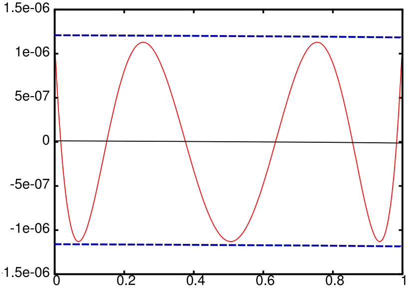
When IA does not suffice: Computing supremum norms of approximation errors

$f(x) = e^x, \; x \in [0, 1], \;\; p(x) = \sum_{i=0}^{5} c_i x^i, \; \varepsilon(x) = f(x) - p(x)$
s.t. $\|\varepsilon\|_\infty = \sup_{x \in [a, b]} \{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)



Why IA does not suffice: Overestimation

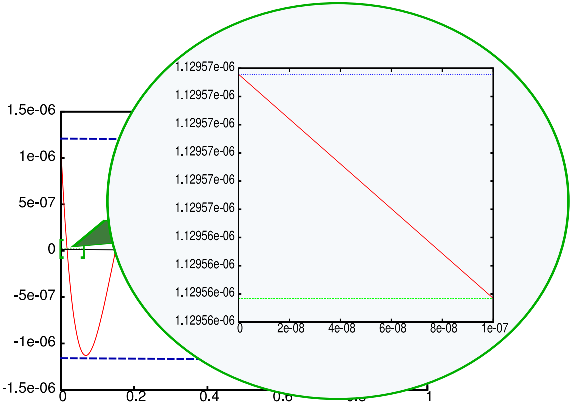
$f(x) = e^x,\ x \in [0,1],\ \ p(x) = \sum_{i=0}^{5} c_i x^i,\ \varepsilon(x) = f(x) - p(x)$
s.t. $\|\varepsilon\|_\infty = \sup_{x\in[a,b]}\{|\varepsilon(x)|\}$ is as small as possible (Remez algorithm)



Using IA, $\varepsilon(x) \in [-0.4, 0.4]$, but $\|\varepsilon(x)\|_\infty \simeq 1.1295 \cdot 10^{-6}$:

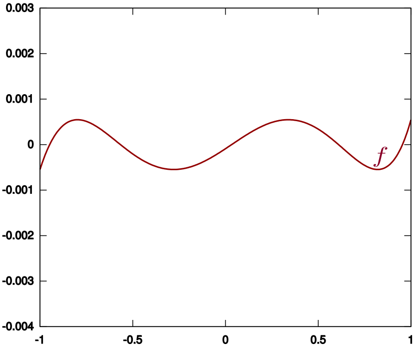
Why IA does not suffice: Overestimation

Overestimation can be reduced by using intervals of smaller width.



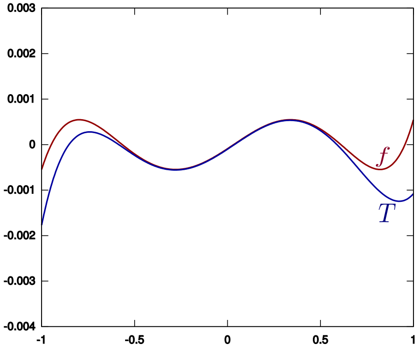
In this case, over $[0, 1]$ we need 10^7 intervals!

Rigorous polynomial approximations



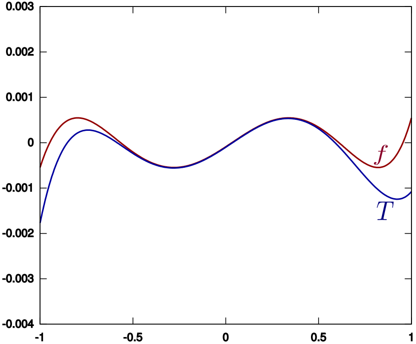
Rigorous polynomial approximations

f replaced with
- polynomial approximation T of degree n



Rigorous polynomial approximations

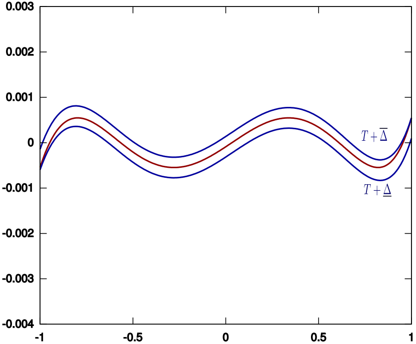
f replaced with
- polynomial approximation T of degree n
- interval $\mathbf{\Delta}$ s. t. $f(x) - T(x) \in \mathbf{\Delta}, \forall x \in [a, b]$



Rigorous polynomial approximations

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

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



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

Taylor Models

Idea: Consider Taylor approximations

Taylor Models - How do we obtain them?

Idea: Consider Taylor approximations

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

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

Taylor Models - How do we obtain them?

Idea: Consider Taylor approximations

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

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

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

Automatic Differentiation - Point intervals

Compute $f^{(i)}(x_0)$ - f represented as an expression tree

Automatic Differentiation - Point intervals

Compute $f^{(i)}(x_0)$ - f represented as an expression tree

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}(0)$

Automatic Differentiation - Point intervals

Compute $f^{(i)}(x_0)$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}(0)$

$$\sin(x) \rightarrow u = [\sin(0), \cos(0), -\sin(0), -\cos(0), \sin(0)]$$

$$\cos(x) \rightarrow v = [\cos(0), -\sin(0), -\cos(0), \sin(0), \cos(0)]$$

Automatic Differentiation - Point intervals

Compute $f^{(i)}(x_0)$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}(x_0) = \sum_{k=0}^{i} u_k \, v_{i-k}$

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}(0)$

$\sin(x) \to u = [\sin(0), \cos(0), -\sin(0), -\cos(0), \sin(0)]$

$\cos(x) \to v = [\cos(0), -\sin(0), -\cos(0), \sin(0), \cos(0)]$

Automatic Differentiation - Point intervals

Compute $f^{(i)}(x_0)$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}(x_0) = \sum_{k=0}^{i} u_k \, v_{i-k}$

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}(0)$

$$\sin(x) \rightarrow u = [\sin(0), \cos(0), -\sin(0), -\cos(0), \sin(0)]$$

$$\cos(x) \rightarrow v = [\cos(0), -\sin(0), -\cos(0), \sin(0), \cos(0)]$$

$$f(x) \rightarrow [u_0\, v_0, u_0\, v_1 + u_1\, v_0, \dots, u_0\, v_4 + u_1\, v_3 + u_2\, v_2 + u_3\, v_1 + u_4\, v_0]$$

Automatic Differentiation - Point intervals

Compute $f^{(i)}(x_0)$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}(x_0) = \sum_{k=0}^{i} u_k \, v_{i-k}$
- For composite functions, recursively apply operations (addition, multiplication, composition)

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}(0)$

$$\sin(x) \to u = [\sin(0), \cos(0), -\sin(0), -\cos(0), \sin(0)]$$

$$\cos(x) \to v = [\cos(0), -\sin(0), -\cos(0), \sin(0), \cos(0)]$$

$$f(x) \to [u_0\, v_0, u_0\, v_1 + u_1\, v_0, \dots, u_0\, v_4 + u_1\, v_3 + u_2\, v_2 + u_3\, v_1 + u_4\, v_0]$$

Automatic Differentiation - Point intervals

Compute $f^{(i)}(x_0)$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}(x_0) = \sum_{k=0}^{i} u_k\, v_{i-k}$
- For composite functions, recursively apply operations (addition, multiplication, composition)

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}(0)$

$$\sin(x) \to u = [\sin(0), \cos(0), -\sin(0), -\cos(0), \sin(0)]$$

$$\cos(x) \to v = [\cos(0), -\sin(0), -\cos(0), \sin(0), \cos(0)]$$

$$f(x) \to [u_0\, v_0, u_0\, v_1 + u_1\, v_0, \ldots, u_0\, v_4 + u_1\, v_3 + u_2\, v_2 + u_3\, v_1 + u_4\, v_0]$$

Automatic Differentiation - Larger intervals

Compute $f^{(i)}([a,b])$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}(x_0) = \sum_{k=0}^{i} u_k\, v_{i-k}$
- For composite functions, recursively apply operations (addition, multiplication, composition)

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}(0)$

$$\sin(x) \to u = [\sin(0), \cos(0), -\sin(0), -\cos(0), \sin(0)]$$

$$\cos(x) \to v = [\cos(0), -\sin(0), -\cos(0), \sin(0), \cos(0)]$$

$$f(x) \to [u_0\, v_0, u_0\, v_1 + u_1\, v_0, \ldots, u_0\, v_4 + u_1\, v_3 + u_2\, v_2 + u_3\, v_1 + u_4\, v_0]$$

Automatic Differentiation - Larger intervals

Compute $f^{(i)}([a,b])$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}([a,b]) = \sum_{k=0}^{i} u_k\, v_{i-k}$
- For composite functions, recursively apply operations (addition, multiplication, composition)

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}([0,1])$

$$\sin(x) \rightarrow u = [\sin(0), \cos(0), -\sin(0), -\cos(0), \sin(0)]$$

$$\cos(x) \rightarrow v = [\cos(0), -\sin(0), -\cos(0), \sin(0), \cos(0)]$$

$$f(x) \rightarrow [u_0\, v_0, u_0\, v_1 + u_1\, v_0, \ldots, u_0\, v_4 + u_1\, v_3 + u_2\, v_2 + u_3\, v_1 + u_4\, v_0]$$

Automatic Differentiation - Larger intervals

Compute $f^{(i)}([a, b])$ - f represented as an expression tree
- Simple formulas for derivatives of "basic functions": $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}([a, b]) = \sum_{k=0}^{i} u_k \, v_{i-k}$
- For composite functions, recursively apply operations (addition, multiplication, composition)

Example:

Given $f(x) = \sin(x)\cos(x), \text{compute } f^{(4)}([0, 1])$

$$\sin(x) \to U = [[0, 0.85], [0.54, 1], [-0.85, 0], [-1, -0.54], [0, 0.85]]$$

$$\cos(x) \to U = [[0.54, 1], [-0.85, 0], [-1, -0.55], [0, 0.85], [0.54, 1]]$$

$$f(x) \to [u_0\, v_0, u_0\, v_1 + u_1\, v_0, \ldots, u_0\, v_4 + u_1\, v_3 + u_2\, v_2 + u_3\, v_1 + u_4\, v_0]$$

Automatic Differentiation - Larger intervals

Compute $f^{(i)}([a, b])$ - f represented as an expression tree
- Simple formulas for derivatives of “basic functions”: $\exp$, $\sin$, etc.
- Leibnitz formula: $f^{(i)}([a, b]) = \sum_{k=0}^{i} u_k \, v_{i-k}$
- For composite functions, recursively apply operations (addition, multiplication, composition)

Example:

Given $f(x) = \sin(x)\cos(x)$, compute $f^{(4)}([0, 1])$

$\sin(x) \to U = [[0, 0.85], [0.54, 1], [-0.85, 0], [-1, -0.54], [0, 0.85]]$

$\cos(x) \to U = [[0.54, 1], [-0.85, 0], [-1, -0.55], [0, 0.85], [0.54, 1]]$

$f(x) \to [u_0 \, v_0, u_0 \, v_1 + u_1 \, v_0, \ldots, [0, 13.5]]$ But $f^{(4)}([0, 1]) = [0, 8]$

What happens when f is a composite function?

$$\color{blue}{f(x)} = \underbrace{\sum_{i=0}^{n} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!}}_{\color{blue}{T(x)}} + \underbrace{\Delta_n(x, \xi)}_{\color{blue}{\text{remainder}}}$$

The interval bound $\mathbf{\Delta}$ for $\Delta_n(x, \xi) = \dfrac{f^{(n+1)}(\xi)(x - x_0)^{n+1}}{(n+1)!}$,

$\xi \in [a, b]$ can be largely overestimated.

What happens when f is a composite function?

The interval bound $\boldsymbol{\Delta}$ for $\Delta_n(x,\xi) = \dfrac{f^{(n+1)}(\xi)(x-x_0)^{n+1}}{(n+1)!}$,
$\xi \in [a,b]$ can be largely overestimated.

Example: $f(x) = e^{1/\cos x}$, over $[0,1]$, $n = 13$, $x_0 = 0.5$.

Using AD: $\boldsymbol{\Delta} = [-1.93 \cdot 10^2,\ 1.35 \cdot 10^3]$

In fact, $f(x) - T(x) \in [0, 4.56 \cdot 10^{-3}]$

What happens when f is a composite function?

The interval bound $\mathbf{\Delta}$ for $\Delta_n(x, \xi) = \dfrac{f^{(n+1)}(\xi)(x - x_0)^{n+1}}{(n+1)!}$, $\xi \in [a, b]$ can be largely overestimated.

Q: What does influence the width of the interval bound for the remainder?

What happens when f is a composite function?

The interval bound $\mathbf{\Delta}$ for $\Delta_n(x,\xi) = \dfrac{f^{(n+1)}(\xi)(x-x_0)^{n+1}}{(n+1)!}$, $\xi \in [a,b]$ can be largely overestimated.

Q: What does influence the width of the interval bound for the remainder?

- The way we compute an interval enclosure for the remainder using simply this formula for any function

Taylor Models Philosophy

For bounding the remainders:

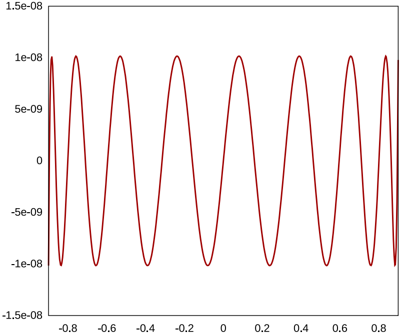
- For “basic functions” use AD.
- For “composite functions”use a two-step procedure:
  - compute models (T, I) for all basic functions;
  - apply algebraic rules with these models, instead of operations with the corresponding functions.

Taylor Models Issues

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 Models Issues

Example:

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

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

In this case Taylor approximations are not good, we need theoretically a TM of degree 120.

Practically, the computed interval remainder can not be made sufficiently small due to overestimation

Consequence: Remainder bounds are unsatisfactory in our case.

Our Approach - Chebyshev Models

Basic idea:

- Use a polynomial approximation better than Taylor: Chebyshev interpolation polynomial.
- Use the two step approach as Taylor Models:

- compute models (P, I) 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 interpolation polynomial.
- Use the two step approach as Taylor Models:

- compute models (P, I) 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

Compute models (P, I) for basic functions f, where P is the Chebyshev interpolation polynomial

- How to compute the coefficients of P?
- How to bound the remainder?
- What basis for representing P?
- What are the algebraic rules with these models ?

Chebyshev Models - Choice of Basis Polynomials

We used:

- Newton Basis
- Chebyshev Basis - discussed in what follows

Chebyshev Models - Choice of Basis Polynomials

We used:

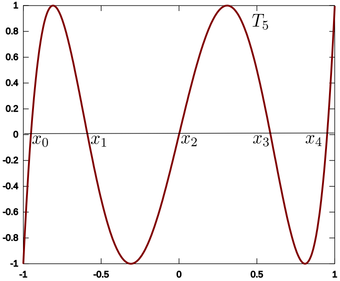
- Newton Basis
- Chebyshev Basis - discussed in what follows

Note: choice of other bases is not detailed in this talk
"Moral principle: Unless you're really, really sure that another set of basis functions is better, use Chebyshev polynomials.",J. P. Boyd[1]

[1] University of Michigan, http://www-personal.umich.edu/~jpboyd/

Chebyshev Polynomials

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



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

Interpolation polynomials - "Basic" Functions Step

Let $\{y_i, i = 0, \ldots, n\}$ be $n+1$ points in $[-1, 1]$. There exists a unique polynomial P of degree $\leq n$ s.t.
$P(y_i) = f(y_i), \forall i = 0, \ldots, n$, or if y_i is repeated k times,
$P^{(j)}(y_i) = f^{(j)}(y_i), \forall j = 0, \ldots, k-1.$

- all y_i equal, P is the Taylor polynomial of f
- all y_i distinct: Lagrange interpolation

Interpolation polynomials - "Basic" Functions Step

Let $\{y_i, i = 0, \ldots, n\}$ be $n+1$ points in $[-1, 1]$. There exists a unique polynomial P of degree $\leq n$ s.t.
$P(y_i) = f(y_i), \forall i = 0, \ldots, n$, or if y_i is repeated k times,
$P^{(j)}(y_i) = f^{(j)}(y_i), \forall j = 0, \ldots, k-1$.

- all y_i equal, P is the Taylor polynomial of f
- Lagrange remainder:
  $\forall x \in [-1, 1],\ \exists \xi \in [-1, 1]\ \text{ s.t.}$
  $$f(x) - P(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - y_i)^{n+1}.$$

Interpolation polynomials - "Basic" Functions Step

Let $\{y_i, i = 0, \ldots, n\}$ be $n+1$ points in $[-1, 1]$. There exists a unique polynomial P of degree $\leq n$ s.t.
$P(y_i) = f(y_i), \forall i = 0, \ldots, n$, or if y_i is repeated k times,
$P^{(j)}(y_i) = f^{(j)}(y_i), \forall j = 0, \ldots, k-1.$

- all y_i distinct: Lagrange interpolation
- Lagrange remainder:
  $\forall x \in [-1, 1], \ \exists \xi \in [-1, 1] \ \text{ s.t.}$
  $f(x) - P(x) = \dfrac{f^{(n+1)}(\xi)}{(n+1)!} W_{\overline{y}}(x)$, with $W_{\overline{y}}(x) = \prod_{i=0}^{n}(x - y_i)$.

Interpolation Error - "Basic" Function Step

Lagrange remainder:

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

$f(x) - P(x) = \dfrac{f^{(n+1)}(\xi)}{(n+1)!} W_{\overline{y}}(x)$, with $W_{\overline{y}}(x) = \prod_{i=0}^{n}(x - y_i)$.

Note:

- Optimal choice of interpolation points is the Chebyshev nodes, $W_{\overline{x}}(x) = \frac{1}{2^n} T_{n+1}(x)$.

Interpolation Error - "Basic" Function Step

Lagrange remainder:

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

$f(x) - P(x) = \dfrac{f^{(n+1)}(\xi)}{(n+1)!} W_{\overline{y}}(x)$, with $W_{\overline{y}}(x) = \prod_{i=0}^{n}(x - y_i)$.

Note:

- Optimal choice of interpolation points is the Chebyshev nodes, $W_{\overline{x}}(x) = \frac{1}{2^n} T_{n+1}(x)$.

✓ We should have an improvement of 2^n in the width of the remainder, compared to Taylor remainder.

X We inherit all issues related to overestimation of $f^{(n+1)}$

Chebyshev interpolation polynomial - "Basic" Function Step

$P(x) = \sum_{i=0}^{n} p_i T_i(x)$ interpolates f at $x_k \in [-1, 1]$, Chebyshev nodes of order $n+1$.

Computation of the coefficients:

$p_i = \sum_{k=0}^{n} \frac{2}{n+1} f(x_k) T_i(x_k)$, $i = 0, \dots, n$

Chebyshev interpolation polynomial - "Basic" Function Step

$P(x) = \sum_{i=0}^{n} p_i T_i(x)$ interpolates f at $x_k \in [-1, 1]$, Chebyshev nodes of order $n+1$.

Computation of the coefficients:

$p_i = \sum_{k=0}^{n} \frac{2}{n+1} f(x_k) T_i(x_k),\ i = 0, \ldots, n$

Remark: Currently, this step is more costly than in the case of TMs.

Chebyshev Models - Operations

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

Chebyshev Models - Operations

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

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

Where $\divideontimes$ is:

- Addition
- Multiplication
- Composition

Chebyshev Models - Operations

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

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

Where $\divideontimes$ is:

- Addition
- Multiplication
- Composition

This is the "hidden difficult part" in designing such models: $\mathbf{\Delta}$ has to be kept tight, while $(P, \mathbf{\Delta})$ has to be computed fast.

Chebyshev Models - Operations: Addition

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

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

Chebyshev Models - Operations: Multiplication

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

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

Chebyshev Models - Operations: Multiplication

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

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

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

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

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

Chebyshev Models - Operations: Multiplication

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

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

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

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

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

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

Chebyshev Models - Operations: Composition

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

Chebyshev Models - Operations: Composition

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

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

Chebyshev Models - Operations: Composition

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

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

$f_1(y) \in P_1(y) + \mathbf{\Delta}_1$

Chebyshev Models - Operations: Composition

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

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

$f_1(f_2(x)) \in P_1(f_2(x)) + \mathbf{\Delta}_1$

Chebyshev Models - Operations: Composition

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

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

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

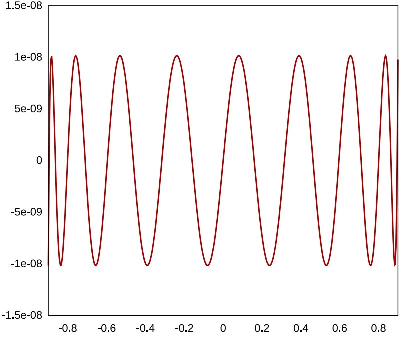
Extract polynomial and remainder: P_1 can be evaluated using only additions and multiplications: Clenshaw's algorithm

Chebyshev Models - 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 - Supremum norm example

Example:

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

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

In this case Taylor approximations are not good, we need theoretically a TM of degree 120.
Practically, the computed interval remainder can not be made sufficiently small due to overestimation.

.

Chebyshev Models - Supremum norm example

Example:

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

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

In this case Taylor approximations are not good, we need theoretically a TM of degree 120.
Practically, the computed interval remainder can not be made 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	Exact bound	TM	Exact bound
$\sin(x)$, $[3,\ 4]$, 10	$1.19 \cdot 10^{-14}$	$1.13 \cdot 10^{-14}$	$1.22 \cdot 10^{-11}$	$1.16 \cdot 10^{-11}$
$\arctan(x)$, $[-0.25,\ 0.25]$, 15	$7.89 \cdot 10^{-15}$	$7.95 \cdot 10^{-17}$	$2.58 \cdot 10^{-10}$	$3.24 \cdot 10^{-12}$
$\arctan(x)$, $[-0.9,\ 0.9]$, 15	$5.10 \cdot 10^{-3}$	$1.76 \cdot 10^{-8}$	$1.67 \cdot 10^{2}$	$5.70 \cdot 10^{-3}$
$\exp(1/\cos(x))$, $[0,\ 1]$, 14	$5.22 \cdot 10^{-7}$	$4.95 \cdot 10^{-7}$	$9.06 \cdot 10^{-3}$	$2.59 \cdot 10^{-3}$
$\frac{\exp(x)}{\log(2+x)\ \cos(x)}$, $[0,\ 1]$, 15	$9.11 \cdot 10^{-9}$	$2.21 \cdot 10^{-9}$	$1.18 \cdot 10^{-3}$	$3.38 \cdot 10^{-5}$
$\sin(\exp(x))$,$[-1,\ 1]$, 10	$9.47 \cdot 10^{-5}$	$3.72 \cdot 10^{-6}$	$2.96 \cdot 10^{-2}$	$1.55 \cdot 10^{-3}$

CMs vs. TMs

Operations complexity:

- ✓ Addition ($O(n)$), Multiplication($O(n^2)$) and Composition ($O(n^3)$) have similar complexity.
- X Initial computation of coefficients for "basic functions" is slower with CMs ($O(n^2)$) vs. TMs ($O(n)$)

Comparison between remainder bounds for several functions:

$f(x)$, I, n	CM	Exact bound	TM	Exact bound
$\sin(x)$, $[3, 4]$, 10	$1.19 \cdot 10^{-14}$	$1.13 \cdot 10^{-14}$	$1.22 \cdot 10^{-11}$	$1.16 \cdot 10^{-11}$
$\arctan(x)$, $[-0.25, 0.25]$, 15	$7.89 \cdot 10^{-15}$	$7.95 \cdot 10^{-17}$	$2.58 \cdot 10^{-10}$	$3.24 \cdot 10^{-12}$
$\arctan(x)$, $[-0.9, 0.9]$, 15	$5.10 \cdot 10^{-3}$	$1.76 \cdot 10^{-8}$	$1.67 \cdot 10^{2}$	$5.70 \cdot 10^{-3}$
$\exp(1/\cos(x))$, $[0, 1]$, 14	$5.22 \cdot 10^{-7}$	$4.95 \cdot 10^{-7}$	$9.06 \cdot 10^{-3}$	$2.59 \cdot 10^{-3}$
$\frac{\exp(x)}{\log(2+x)\cos(x)}$, $[0, 1]$, 15	$9.11 \cdot 10^{-9}$	$2.21 \cdot 10^{-9}$	$1.18 \cdot 10^{-3}$	$3.38 \cdot 10^{-5}$
$\sin(\exp(x))$,$[-1, 1]$, 10	$9.47 \cdot 10^{-5}$	$3.72 \cdot 10^{-6}$	$2.96 \cdot 10^{-2}$	$1.55 \cdot 10^{-3}$

What about other polynomial approximations?

- Remez (minimax):
  - X More costly to obtain (more complex numerical algorithm);
  - X Existent close formula for remainder has the same quality as the one we use.

Quality of approximation compared to minimax

Remark: It is known [Ehlich & Zeller, 1966] that Chebyshev interpolants are "near-best":

$$\|\varepsilon\|_\infty \leq \underbrace{\left(2 + (2/\pi)\log(n)\right)}_{\Lambda_n} \|\varepsilon_{\text{minimax}}\|_\infty$$

- $\Lambda_{15} = 3.72... \rightarrow$ we lose at most 2 bits
- $\Lambda_{30} = 4.16... \rightarrow$ we lose at most 3 bits
- $\Lambda_{100} = 4.93... \rightarrow$ we lose at most 3 bits
- $\Lambda_{100000} = 9.32... \rightarrow$ we lose at most 4 bits

Quality of approximation compared to minimax

No	$f(x)$, I, n	CM	Exact bound	Minimax
1	$\sin(x)$, $[3, 4]$, 10	$1.19 \cdot 10^{-14}$	$1.13 \cdot 10^{-14}$	$1.12 \cdot 10^{-14}$
2	$\arctan(x)$, $[-0.25, 0.25]$, 15	$7.89 \cdot 10^{-15}$	$7.95 \cdot 10^{-17}$	$4.03 \cdot 10^{-17}$
3	$\arctan(x)$, $[-0.9, 0.9]$, 15	$5.10 \cdot 10^{-3}$	$1.76 \cdot 10^{-8}$	$1.01 \cdot 10^{-8}$
4	$\exp(1/\cos(x))$, $[0, 1]$, 14	$5.22 \cdot 10^{-7}$	$4.95 \cdot 10^{-7}$	$3.57 \cdot 10^{-7}$
5	$\frac{\exp(x)}{\log(2+x)\ \cos(x)}$, $[0, 1]$, 15	$9.11 \cdot 10^{-9}$	$2.21 \cdot 10^{-9}$	$1.72 \cdot 10^{-9}$
6	$\sin(\exp(x))$,$[-1, 1]$, 10	$9.47 \cdot 10^{-5}$	$3.72 \cdot 10^{-6}$	$1.78 \cdot 10^{-6}$

What about other polynomial approximations?

What about other polynomial approximations?

- Truncated Chebyshev series:

$$P(x) = \sum_{k=0}^{n}{}' a_k T_k(x), \text{ where } a_k = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x) T_k(x)}{\sqrt{1-x^2}} \mathrm{d}x$$

✓ possible speed-up: recurrence formulae for computing polynomial coefficients for "basic functions"

?? Possible loss in the quality of remainder.

Rigorous quadrature

Example:

$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

- Compute a TM/CM $(P, \boldsymbol{I})$ for $f(x) = \dfrac{4}{1+x^2}$.

Rigorous quadrature

Example:

$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

- Compute a TM/CM $(P, \boldsymbol{I})$ for $f(x) = \dfrac{4}{1+x^2}$.

  $P(x) + \underline{\boldsymbol{I}} \leq f(x) \leq P(x) + \overline{\boldsymbol{I}}$

Rigorous quadrature

Example:

$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

- Compute a TM/CM $(P, \boldsymbol{I})$ for $f(x) = \dfrac{4}{1+x^2}$.

$$\int_a^b (P(x) + \underline{\boldsymbol{I}})\mathrm{d}x \leq \int_a^b f(x)\mathrm{d}x \leq \int_a^b (P(x) + \overline{\boldsymbol{I}})\mathrm{d}x$$

Rigorous quadrature

Example:

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x$$

Order	Subdiv.	Bound TM[2]	Bound CM
5	1	[3.0231893333333, 8.5807786666666]	[3.0986941190195, 3.1859962140742]
	4	[3.1415363229415, 3.1416629536292]	[3.1415907717769, 3.1415943610772]
	16	[3.1415926101614, 3.1415926980786]	[3.1415926531269, 3.1415926539131]
10	1	[-2.1984010266006, 3.2113963175267]	[3.1411981994969, 3.1419909934525]
	4	[3.1415926519535, 3.1415926546870]	[3.1415926535805, 3.1415926535990]
	16	[3.1415926535897, 3.1415926535897]	[3.1415926535897932, 3.1415926535897932]

[2] Results taken from M. Berz, K. Makino, "New Methods for High-Dimensional Verified Quadrature", Reliable Computing 5:13-22, 1999

Conclusion

- CMs are potentially useful in various rigorous computing applications: smaller remainders than TMs, but require more computing time.
- Current implementation: partially Sollya and Maple.
- Work in progress: use Chebyshev truncated series instead of Chebyshev interpolation polynomials.
- Future work: extend to multivariate functions

How to reduce the width of the remainder?

Monotonic properties of the remainder can be infered for "basic" functions [3]:

[3] Corollary in R. Zumkeller, "Global Optimization in Type Theory", PhD thesis, page 84

How to reduce the width of the remainder?

Monotonic properties of the remainder can be infered for "basic" functions [3]:

If $f^{(n+1)}([a,b]) \geq 0$, and T is Taylor polynomial of degree n for f , $\dfrac{f-T}{(x-x_0)^{n+1}}$ is monotonic over $[a,b]$: the remainder can be exactly bounded using two evaluations of $f-T$ in the end points of $[a,b]$.

[3] Corollary in R. Zumkeller, "Global Optimization in Type Theory", PhD

How to reduce the width of the remainder?

Monotonic properties of the remainder can be infered for "basic" functions [3]:

Example: $f(x) = \log(x)$, over $[0.001, 1.001]$, $n = 13$, $x_0 = 0.5$.

$$\Delta_n(x,\xi) = \frac{-1}{\xi^{14}} \cdot \frac{(x-0.5)^{14}}{14!}$$

$$\boldsymbol{\Delta} \subseteq [-2.66 \cdot 10^{31}, 2.63 \cdot 10^{-10}]$$

[3] Corollary in R. Zumkeller, "Global Optimization in Type Theory", PhD thesis, page 84

How to reduce the width of the remainder?

Monotonic properties of the remainder can be infered for “basic” functions [3]:

Example: $f(x) = \log(x)$, over $[0.001, 1.001]$, $n = 13$, $x_0 = 0.5$.

$$\Delta_n(x, \xi) = \frac{-1}{\xi^{14}} \cdot \frac{(x - 0.5)^{14}}{14!}$$

$$\boldsymbol{\Delta} \subseteq [-2.66 \cdot 10^{31}, 2.63 \cdot 10^{-10}]$$

With Z's remark, $f(x) - T(x) \in [-3.06, 7.89 \cdot 10^{-31}]$

[3] Corollary in R. Zumkeller, "Global Optimization in Type Theory", PhD thesis, page 84

Interpolation Error - "Basic" Function Step

We can prove (a generalization of Zumkeller's remark):

if $f^{(n+1)}$ is monotonic over I, then $\frac{f(x)-P(x)}{W_{\overline{y}}(x)}$ is monotonic over I.
The remainder can be exactly bounded using two evaluations in the end points of I