Certified and fast computation of supremum norms of approximation errors

Sylvain Chevillard   Mioara Joldes   Christoph Lauter

October 24, 2008
Motivation

- Mathematical Libraries
- Correctly rounded elementary functions
- Supremum norm of error functions
- Previous approaches and difficulties
Outline

Motivation

- Mathematical Libraries
- Correctly rounded elementary functions
- Supremum norm of error functions
- Previous approaches and difficulties

Our approach

- Safe and fast supremum norm of approximation errors
- Automatic differentiation
- Isolation of roots of polynomials
Outline

Motivation
- Mathematical Libraries
- Correctly rounded elementary functions
- Supremum norm of error functions
- Previous approaches and difficulties

Our approach
- Safe and fast supremum norm of approximation errors
- Automatic differentiation
- Isolation of roots of polynomials

Results & Conclusion
Mathematical Libraries (libms)

- **What:** Compute sin, cos, exp and their inverses
Mathematical Libraries (libms)

- **What**: Compute sin, cos, exp and their inverses
- **Why**: Software systems: scientific computing, financial, embedded systems
Mathematical Libraries (libms)

- **What:** Compute sin, cos, exp and their inverses
- **Why:** Software systems: scientific computing, financial, embedded systems
- **How:** Finite precision, floating-point environment
Mathematical Libraries (libms)

- **What:** Compute sin, cos, exp and their inverses
- **Why:** Software systems: scientific computing, financial, embedded systems
- **How:** Finite precision, floating-point environment
- **Problem:** Most of them do not provide correctly rounded functions. This can ruin portability. IEEE-754 standard revision (June 2008) recommends correct rounding.

Arenaire team develops the Correctly Rounded Libm (CRLibm) - http://lipforge.ens-lyon.fr/www/crlibm/
Mathematical Libraries (libms)

- **What**: Compute sin, cos, exp and their inverses
- **Why**: Software systems: scientific computing, financial, embedded systems
- **How**: Finite precision, floating-point environment
- **Problem**: Most of them do not provide correctly rounded functions. This can ruin portability. IEEE-754 standard revision (June 2008) recommends correct rounding.
- **Arenaire team develops the Correctly Rounded Libm (CRLibm)**\(^1\).

\(^1\)http://lipforge.ens-lyon.fr/www/crlibm/
Need: Correctly rounded functions

Given \( f : [a, b] \rightarrow \mathbb{R} \) and \( x \in [a, b] \), the function code should always return the machine number closest to the exact value \( f(x) \).

Use an approximation polynomial \( p \) of \( f \) over \( [a, b] \), consider
\[
\varepsilon = f - p \quad \text{or} \quad \varepsilon = \frac{p}{f} - 1
\]

Given \( \mu \), basic condition to assure: \( \forall x \in [a, b], |\varepsilon(x)| \leq \mu \)
Supremum Norms of Error Functions

- Error $\varepsilon(x) = f(x) - p(x)$ or $\varepsilon(x) = \frac{p(x)}{f(x)} - 1, \quad x \in [a, b]$
- Define $\|\varepsilon\|_\infty = \sup_{x \in [a, b]} \{|\varepsilon(x)|\}$
- Purpose: Compute a certified bound for the supremum norm of an error function
- Given $p$ and $f$ find an interval $r$ (as thin as desired) such that $\|\varepsilon\|_\infty \in r$. 
Supremum Norms of Error Functions

- Error \( \varepsilon(x) = f(x) - p(x) \) or \( \varepsilon(x) = \frac{p(x)}{f(x)} - 1, \quad x \in [a, b] \)
- Define \( \|\varepsilon\|_\infty = \sup_{x \in [a, b]} \{|\varepsilon(x)|\} \)
- **Purpose:** Compute a certified bound for the supremum norm of an error function
- Given \( p \) and \( f \) find an interval \( r \) (as thin as desired) such that \( \|\varepsilon\|_\infty \in r \).

**Need for a fast and certified algorithm:**

- Correctly rounded elementary functions
- For computing the minimum error between a function and thousands of polynomials with floating-point coefficients
Each interval = pair of floating-point numbers
Interval Arithmetic

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

- Each interval = pair of floating-point numbers
- \( \pi \in [3.1415, 3.1416] \)
- Extend to intervals
Interval Arithmetic

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

Arithmetic operations

Eg. $[1, 2] + [-3, 2] = [-2, 4]$
Interval Arithmetic

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

**Arithmetic operations**

- Functions: **Range Bounding**

Eg. $[1, 2] + [-3, 2] = [-2, 4]$  
$f(x) = x^2 + x + 1, x \in [-1, 2]$
Interval Arithmetic

- Each interval = pair of floating-point numbers
- \( \pi \in [3.1415, 3.1416] \)
- Extend to intervals

Arithmetic operations

Eg. \([1, 2] + [-3, 2] = [-2, 4]\)

Functions: Range Bounding

\( f(x) = x^2 + x + 1, x \in [-1, 2] \)

All variables are replaced by intervals

\( F(X) = X^2 + X + 1 \)
Interval Arithmetic

- Each interval = pair of floating-point numbers
- \( \pi \in [3.1415, 3.1416] \)
- Extend to intervals

Arithmetic operations

Functions: Range Bounding

All variables are replaced by intervals

Evaluate using interval arithmetic

Eg. \([1, 2] + [-3, 2] = [-2, 4]\)

\[ f(x) = x^2 + x + 1, \; x \in [-1, 2] \]

\[ F(X) = X^2 + X + 1 \]

\[ F([-1, 2]) = [-1, 2]^2 + [-1, 2] + [1, 1] \]

\[ F([-1, 2]) = [0, 4] + [-1, 2] + [1, 1] \]

\[ F([-1, 2]) = [0, 7] \]
Specific problem: supremum norm of error function

Let \( f(x) = e^x, x \in [-1, 1] \)

We are given \( p \) with real coefficients, \( \deg p \leq 5 \) s.t. \( \| f - p \|_\infty \) is as small as possible (Remez algorithm)
Specific problem: supremum norm of error function
Motivation

- Specific problem: supremum norm of error function

- Degenerate problem:
  - Cancellation
Motivation

- **Specific problem:** supremum norm of error function

- **Degenerate problem:**
  - Cancellation
  - Dependence
Motivation - Difficulties

- Specific problem: supremum norm of error function
- Dependence -
Specific problem: supremum norm of error function

Dependence - Interval arithmetic is “blind“!
$x \in [-2, 2], f(x) = x - x$, computing using interval extension,
$F(X) = [-2, 2] - [-2, 2] = [-4, 4] \neq [0, 0]$
Specific problem: supremum norm of error function

Dependence - Interval arithmetic is “blind”!

$x \in [-2, 2], f(x) = x - x$, computing using interval extension, $F(X) = [-2, 2] - [-2, 2] = [-4, 4] \neq [0, 0]$

Can be reduced by using very small intervals
Previous Approaches

- Floating-point techniques (Brent) - not “safe“
Previous Approaches

- Floating-point techniques (Brent) - not “safe“
- Global optimization software (eg. Globsol) - not tailored for our specific problem

Cheillard and Lauter’s technique: interval arithmetic, tight bounding of the zeros of the derivative of the error function, removes false singularities (like $x = 0$ for $\sin(x)/x$). High computation time for degree $(p) > 10$.

- Techniques based on a high order Taylor expansion of the error function and a sufficiently close bounding of the remainder

Sum of squares algorithms (Harrison)

The remainder is computed manually.
Previous Approaches

- Floating-point techniques (Brent) - not “safe“
- Global optimization software (eg. Globsol) - not tailored for our specific problem
- Chevillard and Lauter’s technique: interval arithmetic, tight bounding of the zeros of the derivative of the error function, removes false singularities (like $x = 0$ for $\sin(x)/x$). High computation time for $\deg(p) > 10$. 

Techniques based on a high order Taylor expansion of the error function and a sufficiently close bounding of the remainder

Sum of squares algorithms (Harrison)

The remainder is computed manually
Previous Approaches

- Floating-point techniques (Brent) - not “safe“
- Global optimization software (eg. Globsol) - not tailored for our specific problem
- Chevillard and Lauter’s technique: interval arithmetic, tight bounding of the zeros of the derivative of the error function, removes false singularities (like $x = 0$ for $\sin(x)/x$). High computation time for $\deg(p) > 10$.
- Techniques based on a high order Taylor expansion of the error function and a sufficiently close bounding of the remainder
  - Sum of squares algorithms (Harrison)
  - The remainder is computed manually
Our Approach

- **Compute:** $\|\epsilon\|_{\infty} = \|f - p\|_{\infty}$
Our Approach

- **Compute:** $\|\varepsilon\|_\infty = \|f - p\|_\infty$
- **Example:**

Consider:

\[
f(x) = \exp(x) \text{ over } [0, 1] \\
p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707)
\]
Our Approach

- **Compute:** \( \| \varepsilon \|_\infty = \| f - p \|_\infty \)
- **Example:**

<table>
<thead>
<tr>
<th>Consider:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x) = \exp(x) ) over ([0, 1])</td>
</tr>
<tr>
<td>( p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707) )</td>
</tr>
</tbody>
</table>

- Using interval arithmetic we obtain: \( \| f - p \|_\infty \leq 0.2836 \)
Our Approach

- **Compute:** \( \| \varepsilon \|_\infty = \| f - p \|_\infty \)
- **Example:**

Consider:

\[
f(x) = \exp(x) \text{ over } [0, 1]
\]
\[
p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707)
\]

- Using interval arithmetic we obtain: \( \| f - p \|_\infty \leq 0.2836 \)
- By sampling we obtain: \( \| f - p \|_\infty \approx 0.008756064 \)
Our Approach

- **Compute:** \[ \| \varepsilon \|_\infty = \| f - p \|_\infty \]
- **Example:**

**Consider:**

\[ f(x) = \exp(x) \text{ over } [0, 1] \]
\[ p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707) \]

- Using interval arithmetic we obtain: \[ \| f - p \|_\infty \leq 0.2836 \]
- By sampling we obtain: \[ \| f - p \|_\infty \approx 0.008756064 \]
Our Approach

- **Compute:** \( \| \varepsilon \|_\infty = \| f - p \|_\infty \)**
Our Approach

- **Compute:** $\| \varepsilon \|_\infty = \| f - p \|_\infty$

- **Idea:** Use a higher degree approximation polynomial $T$. 
Our Approach

- **Compute:** $\|\varepsilon\|_\infty = \|f - p\|_\infty$
- **Idea:** Use a higher degree approximation polynomial $T$.  

$$\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty$$

1. Compute $T$
Our Approach

- **Compute**: $\| \varepsilon \|_\infty = \| f - p \|_\infty$
- **Idea**: Use a higher degree approximation polynomial $T$.

\[
\| f - p \|_\infty \leq \| f - T \|_\infty + \| T - p \|_\infty
\]

**Bounding a remainder**

1. Compute $T$
Our Approach

- **Compute:** $\|\varepsilon\|_\infty = \|f - p\|_\infty$
- **Idea:** Use a higher degree approximation polynomial $T$.

\[
\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty
\]

- **Bounding a remainder**
- **Bounding a polynomial**

1. **Compute** $T$
Our Approach

- **Compute:** \( \| \varepsilon \|_\infty = \| f - p \|_\infty \)
- **Idea:** Use a higher degree approximation polynomial \( T \).

\[
\| f - p \|_\infty \leq \underbrace{\| f - T \|_\infty}_{\text{bounding a remainder}} + \underbrace{\| T - p \|_\infty}_{\text{bounding a polynomial}}
\]

1. **Compute** \( T \)
2. **Automatic Differentiation**
Our Approach

- **Compute:** $\|\varepsilon\|_\infty = \|f - p\|_\infty$
- **Idea:** Use a higher degree approximation polynomial $T$.

\[ \|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty \]

bounding a remainder binding a polynomial

1. **Compute** $T$
2. **Automatic Differentiation**
3. **Bounding the polynomial difference** - Roots isolation and refinement techniques
Our Approach - (1) Introducing a higher degree approximation polynomial

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

$$f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)$$
Our Approach - (1) Introducing a higher degree approximation polynomial

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

$$f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)$$

where $T(x)$ is the Taylor polynomial of degree $n$.
Our Approach - (1) Introducing a higher degree approximation polynomial

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

$$f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x-x_0)^i}{i!} + \Delta_n(x, \xi)$$

$$T(x)$$

The remainder

$$\Delta_n(x, \xi) = \frac{f^{(n)}(\xi)(x-x_0)^n}{n!}, \ x \in [a, b], \ \xi \text{ lies strictly between } x \text{ and } x_0$$
Our Approach - (1) Introducing a higher degree approximation polynomial

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

$$f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)$$

Issues:
Our Approach - (1) Introducing a higher degree approximation polynomial

Let \( n \in \mathbb{N} \), \( n \) times differentiable function \( f \) over \( [a, b] \) around \( x_0 \):

\[
f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)
\]

Issues:

- Compute the coeffs of \( T(x) : \frac{f^{(i)}(x_0)}{i!}, i = \{0, \cdots, n - 1\} \)
Our Approach - (1) Introducing a higher degree approximation polynomial

Let \( n \in \mathbb{N} \), \( n \) times differentiable function \( f \) over \([a, b]\) around \( x_0 \):

\[
f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)
\]

where \( T(x) \) is the remainder.

Issues:

- Compute the coeffs of \( T(x) : \frac{f^{(i)}(x_0)}{i!}, i = \{0, \cdots, n - 1\} \)
- Bound the remainder \( \Delta_n(x, \xi) : \frac{f^{(n)}(\xi)}{n!}(x - x_0)^n, \xi \) between \( x \) and \( x_0 \)
Our Approach - (1) Introducing a higher degree approximation polynomial

Let \( n \in \mathbb{N} \), \( n \) times differentiable function \( f \) over \([a, b]\) around \( x_0 \):

\[
f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)
\]

Issues:

- Compute the coeffs of \( T(x) : \frac{f^{(i)}(x_0)}{i!} \), \( i = \{0, \cdots, n-1\} \)
- Bound the remainder \( \Delta_n(x, \xi) : \frac{f^{(n)}(\xi)}{n!}(x - x_0)^n \), \( \xi \) between \( x \) and \( x_0 \)
- Compute high order derivatives!
Our Approach - (1) Introducing a higher degree approximation polynomial

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

$$f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)$$

$T(x)$ remainder

Issues:

- Compute the coeffs of $T(x)$: $\frac{f^{(i)}(x_0)}{i!}, i = \{0, \cdots, n - 1\}$

- Bound the remainder $\Delta_n(x, \xi)$: $\frac{f^{(n)}(\xi)}{n!}(x - x_0)^n$, $\xi$ between $x$ and $x_0$

- Compute high order derivatives!
  - Symbolic differentiation - inefficient
Our Approach - (1) Introducing a higher degree approximation polynomial

Let \( n \in \mathbb{N} \), \( n \) times differentiable function \( f \) over \([a, b]\) around \( x_0 \):

\[
f(x) = \sum_{i=0}^{n-1} \frac{f^{(i)}(x_0)(x - x_0)^i}{i!} + \Delta_n(x, \xi)
\]

Issues:

- Compute the coeffs of \( T(x) : \frac{f^{(i)}(x_0)}{i!}, i = \{0, \cdots, n - 1\} \)
- Bound the remainder \( \Delta_n(x, \xi) : \frac{f^{(n)}(\xi)}{n!}(x - x_0)^n, \xi \) between \( x \) and \( x_0 \)
- Compute high order derivatives!
  - Symbolic differentiation - inefficient
  - Automatic Differentiation (AD)
Our Approach - (2) Automatic differentiation (AD)

- Purpose: Compute values of high order derivatives of $f$

- Automatic differentiation (AD): usually developed by program code transformation, operators overloading

- Can be easily extended to work with interval arithmetic

- Compute $f(i)(x_0)$, $i \in \{0, \ldots, n\}$ or an enclosure for $f(i)(\xi)$, $i \in \{0, \ldots, n\}$, when $\xi \in [a, b]$. 
Our Approach - (2) Automatic differentiation (AD)

- Purpose: Compute values of high order derivatives of $f$
- $f$ usually represented as an expression tree
Our Approach - (2) Automatic differentiation (AD)

- **Purpose:** Compute values of high order derivatives of \( f \)
- \( f \) usually represented as an expression tree
- **Problem:** For large values of \( n \), high memory usage for symbolic differentiation.
Our Approach - (2) Automatic differentiation (AD)

- **Purpose:** Compute values of high order derivatives of $f$
- $f$ usually represented as an expression tree
- **Problem:** For large values of $n$, high memory usage for symbolic differentiation.
- **Solution:** do the same operations like when evaluating the expression of the derivative, but don’t effectively write this expression
Our Approach - (2) Automatic differentiation (AD)

- Purpose: Compute values of high order derivatives of $f$
- $f$ usually represented as an expression tree
- Problem: For large values of $n$, high memory usage for symbolic differentiation.
- Solution: do the same operations like when evaluating the expression of the derivative, but don’t effectively write this expression
- AD: usually developed by program code transformation, operators overloading

Compute $f(i)(x_0)$, $i \in \{0, \ldots, n\}$ or an enclosure for $f(i)(\xi)$, $i \in \{0, \ldots, n\}$, when $\xi \in [a, b]$. 
Purpose: Compute values of high order derivatives of \( f \)

\( f \) usually represented as an expression tree

Problem: For large values of \( n \), high memory usage for symbolic differentiation.

Solution: do the same operations like when evaluating the expression of the derivative, but don’t effectively write this expression

AD: usually developed by program code transformation, operators overloading

Can be easily extended to work with interval arithmetic
Our Approach - (2) Automatic differentiation (AD)

- **Purpose:** Compute values of high order derivatives of \( f \)
- **\( f \) usually represented as an expression tree**
- **Problem:** For large values of \( n \), high memory usage for symbolic differentiation.
- **Solution:** do the same operations like when evaluating the expression of the derivative, but don’t effectively write this expression
- **AD:** usually developed by program code transformation, operators overloading
- **Can be easily extended to work with interval arithmetic**
- **Compute** \[ \frac{f^{(i)}(x_0)}{i!}, \quad i = \{0, \cdots, n\} \] or an enclosure for \[ \frac{f^{(i)}(\xi)}{i!}, \quad i = \{0, \cdots, n\}, \text{ when } \xi \in [a, b]. \]
Our Approach - (2) Automatic differentiation (AD)

Example:

\[ f(x) = \exp(x) \text{ over } [0, 1] \]
\[ p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707) \]
Our Approach - (2) Automatic differentiation (AD)

- Example:

\[ f(x) = \exp(x) \text{ over } [0, 1] \]
\[ p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707) \]

- Introduce a higher degree polynomial \( T \): Use AD

\[
T(x) = \sum_{i=0}^{6} \frac{f^{(i)}(1/2)}{i!} (x - 1/2)^i = \sum_{i=0}^{6} \frac{\exp(1/2)}{i!} (x - 1/2)^i
\]
Our Approach - (2) Automatic differentiation (AD)

- Example:

\[ f(x) = \exp(x) \text{ over } [0, 1] \]
\[ p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707) \]

- Introduce a higher degree polynomial \( T \): Use AD

\[ T(x) = \sum_{i=0}^{6} \frac{f^{(i)}(1/2)}{i!} (x - 1/2)^i = \sum_{i=0}^{6} \frac{\exp(1/2)}{i!} (x - 1/2)^i \]

- Compute an enclosure of the remainder: Use AD

\[ \Delta_7(x, \xi) = \frac{f^{(7)}(\xi)}{7!} \times (x - 1/2)^7 \]
\[ \leq (1/2)^{(-7)} \]
\[ \exp([0, 1]) \]
\[ \in \frac{7!}{7!} \]
\[ |\Delta_7(x, \xi)| \leq 1.46305781422e - 8 \]
Main idea:

\[ \| f - p \|_\infty \leq \| f - T \|_\infty + \| T - p \|_\infty \]

bounding a remainder  
bounding a polynomial
Our Approach - (2) Automatic differentiation (AD)

- **Main idea:**

\[
\| f - p \|_\infty \leq \| f - T \|_\infty + \| T - p \|_\infty
\]

bounding a remainder

bounding a polynomial

- **Achieved so far:**

\[
\| \exp - p \|_\infty \leq \| \exp - T \|_\infty + \| T - p \|_\infty
\]

\[\leq 1.46305781422e-8\]

bounding a polynomial
Purpose: Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$
Our Approach - (3) Bounding the polynomial difference

Purpose: Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$

Example:

$p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707)$

$T(x) = \sum_{i=0}^{6} \frac{f^{(i)}(1/2)}{i!}(x - 1/2)^i$
Our Approach - (3) Bounding the polynomial difference

**Purpose:** Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$

**Example:**

$p(x) = 1.008756 + x \times (0.85474264 + x \times 0.84602707)$

$T(x) = \sum_{i=0}^{6} \frac{f^{(i)}(1/2)}{i!}(x - 1/2)^i$

$T(x) - p(x) \Rightarrow$
Our Approach - (3) Bounding the polynomial difference

**Purpose:** Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$

$T(x) - p(x) \implies$
Our Approach - (3) Bounding the polynomial difference

Purpose: Tightly bound $\| T - p \|_{\infty}$ over the interval $[a, b]$
Our Approach - (3) Bounding the polynomial difference

Purpose: Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$

$T'(x) - p'(x) \implies$
Our Approach - (3) Bounding the polynomial difference

Purpose: Tightly bound \( \| T - p \|_\infty \) over the interval \([a, b]\)

\[ T'(x) - p'(x) \implies \]

- Tightly bound the roots of the derivative.
Our Approach - (3) Bounding the polynomial difference

Purpose: Tightly bound $\|T - p\|_\infty$ over the interval $[a, b]$

$T'(x) - p'(x)$ →

- Tightly bound the roots of the derivative
- Evaluate using interval arithmetic
Our Approach - (3) Isolation and refinement of roots of polynomials

- Techniques based on counting the number of roots inside an interval considered
Our Approach - (3) Isolation and refinement of roots of polynomials

- Techniques based on counting the number of roots inside an interval considered
  - Sturm Theorem based strategies
  - Descartes’ Rule of Signs based strategies
Our Approach - (3) Isolation and refinement of roots of polynomials

- Techniques based on counting the number of roots inside an interval considered
  - Sturm Theorem based strategies
  - Descartes’ Rule of Signs based strategies
- Use a bisection strategy for isolating the roots
Our Approach - (3) Isolation and refinement of roots of polynomials

- Techniques based on counting the number of roots inside an interval considered
  - Sturm Theorem based strategies
  - Descartes’ Rule of Signs based strategies
- Use a bisection strategy for isolating the roots
- Use dichotomy or Newton iteration process
Our Approach - (3) Bounding the polynomial difference

**Purpose:** Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$

$T(x) - p(x) \implies$

![Graph showing polynomial difference](image)
Our Approach - (3) Bounding the polynomial difference

Purpose: Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$.

$T(x) - p(x) \implies$

- Tightly bound the roots of the derivative

$r_1 \in [0.2657, 0.2659]$  
$r_2 \in [0.7652, 0.7654]$
Our Approach - (3) Bounding the polynomial difference

**Purpose**: Tightly bound $\| T - p \|_\infty$ over the interval $[a, b]$

- **Tightly bound the roots of the derivative**
  - $r_1 \in [0.2657, 0.2659]$  
  - $r_2 \in [0.7652, 0.7654]$  

- **Evaluate using interval arithmetic**
  - $\| T - p \|_\infty \leq 0.0087566$
Our Approach - (3) Bounding the polynomial difference

**Purpose:** Tightly bound $\|T - p\|_\infty$ over the interval $[a, b]$.

$T(x) - p(x)$

- Tightly bound the roots of the derivative
  - $r_1 \in [0.2657, 0.2659]$  
  - $r_2 \in [0.7652, 0.7654]$  

- Evaluate using interval arithmetic
  - $\|T - p\|_\infty \leq 0.0087566$  
  - $\|f - p\|_\infty \approx 0.00875606$
1 Purpose: fast and safely compute the supremum norm 
\[ \|f - p\|_\infty \] over an interval \([a, b]\)
Our Approach - Summary

1. Purpose: fast and safely compute the supremum norm \( \|f - p\|_\infty \) over an interval \([a, b]\)

2. Introduce a higher degree approximation polynomial:
\[
\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty \quad \text{(AD)}
\]
Our Approach - Summary

1. Purpose: fast and safely compute the supremum norm $\|f - p\|_\infty$ over an interval $[a, b]$

2. Introduce a higher degree approximation polynomial:

   $\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty$ (AD)

3. Bound the remainder (AD)

Our example:

$\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty \leq 1.46305781422$  

By comparison, using interval arithmetic, we obtain

$\|f - p\|_\infty \leq 0.2836$. 

-23-
Our Approach - Summary

1. Purpose: fast and safely compute the supremum norm
   \[\|f - p\|_\infty\] over an interval \([a, b]\)

2. Introduce a higher degree approximation polynomial:
   \[\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty\] (AD)

3. Bound the remainder (AD)

4. Bound the polynomial difference (polynomial roots isolation)
Our Approach - Summary

1 Purpose: fast and safely compute the supremum norm $\|f - p\|_\infty$ over an interval $[a, b]$

2 Introduce a higher degree approximation polynomial:

   $\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty$ (AD)

3 Bound the remainder (AD)

4 Bound the polynomial difference (polynomial roots isolation)

5 Add the two bounds to obtain a tight and safe bound of the approximation error
Our Approach - Summary

1. Purpose: fast and safely compute the supremum norm \( \| f - p \|_\infty \) over an interval \([a, b]\)

2. Introduce a higher degree approximation polynomial:
   \[ \| f - p \|_\infty \leq \| f - T \|_\infty + \| T - p \|_\infty \] (AD)

3. Bound the remainder (AD)

4. Bound the polynomial difference (polynomial roots isolation)

5. Add the two bounds to obtain a tight and safe bound of the approximation error

Our example:

\[
\| f - p \|_\infty \leq \left\| f - T \right\|_\infty + \left\| T - p \right\|_\infty \\
\leq 1.46305781422e-8 + 0.0087566
\]
Our Approach - Summary

1. **Purpose**: fast and safely compute the supremum norm $\|f - p\|_\infty$ over an interval $[a, b]$
2. Introduce a higher degree approximation polynomial:
   $\|f - p\|_\infty \leq \|f - T\|_\infty + \|T - p\|_\infty$ (AD)
3. Bound the remainder (AD)
4. Bound the polynomial difference (polynomial roots isolation)
5. Add the two bounds to obtain a tight and safe bound of the approximation error

**Our example:**

\[
\|f - p\|_\infty \leq \underbrace{\|f - T\|_\infty}_{\leq 1.46305781422e-8} + \underbrace{\|T - p\|_\infty}_{\leq 0.0087566}
\]

By comparison, using interval arithmetic, we obtain $\|f - p\|_\infty \leq 0.2836.$
Experiments were made on an Intel Pentium D 3.00GHz with a 2GB RAM.

<table>
<thead>
<tr>
<th>$f$</th>
<th>$[a, b]$</th>
<th>$d_p$</th>
<th>$m^2$</th>
<th>acc</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exp(x) - 1$</td>
<td>$[-0.25, 0.25]$</td>
<td>5</td>
<td>r</td>
<td>37.6</td>
<td>412</td>
</tr>
<tr>
<td>$\log_2(1 + x)$</td>
<td>$[-2^{-9}, 2^{-9}]$</td>
<td>7</td>
<td>r</td>
<td>83.3</td>
<td>2, 186</td>
</tr>
<tr>
<td>$\cos(x)$</td>
<td>$[-0.5, 0.25]$</td>
<td>15</td>
<td>r</td>
<td>19.5</td>
<td>2, 235</td>
</tr>
<tr>
<td>$\exp(x)$</td>
<td>$[-0.125, 0.125]$</td>
<td>25</td>
<td>r</td>
<td>42.3</td>
<td>7, 753</td>
</tr>
<tr>
<td>$\sin(x)$</td>
<td>$[-0.5, 0.5]$</td>
<td>9</td>
<td>a</td>
<td>21.5</td>
<td>520</td>
</tr>
<tr>
<td>$\exp(\cos(x)^2 + 1)$</td>
<td>$[1, 2]$</td>
<td>15</td>
<td>r</td>
<td>25.5</td>
<td>10, 984</td>
</tr>
<tr>
<td>$\tan(x)$</td>
<td>$[0.25, 0.5]$</td>
<td>10</td>
<td>r</td>
<td>26.0</td>
<td>1, 072</td>
</tr>
<tr>
<td>$x^{2.5}$</td>
<td>$[1, 2]$</td>
<td>7</td>
<td>r</td>
<td>15.5</td>
<td>1, 362</td>
</tr>
</tbody>
</table>

---

1 Degree of $p$
2 Error mode considered: a=absolute, r=relative
3 Accuracy
4 Timings in ms
Conclusion

- Safe and fast algorithm for bounding the supremum norm of the error functions
- Combination and reusal of various techniques (AD, polynomial roots isolation, interval arith)
- Absolute and Relative errors handled
- Faster and more accurate than other current approaches
- Future works:
  - Formal proof (AD, isolation of roots, multiple precision interval arithmetic are needed in the proof checker)
  - Replace "Taylor" polynomial with "Chebyshev-like interpolation polynomial"
Thank you for your attention!

Questions?
Experiments were made on an Intel Pentium D 3.00GHz with a 2GB RAM.

<table>
<thead>
<tr>
<th>$f$</th>
<th>$[a, b]$</th>
<th>$d_p^{1}$</th>
<th>$m^2$</th>
<th>$d_T^{3}$</th>
<th>acc$^4$</th>
<th>time$^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exp(x) - 1$</td>
<td>$[-0.25, 0.25]$</td>
<td>5</td>
<td>r</td>
<td>11</td>
<td>37.6</td>
<td>412</td>
</tr>
<tr>
<td>$\log_2(1 + x)$</td>
<td>$[-2^{-9}, 2^{-9}]$</td>
<td>7</td>
<td>r</td>
<td>23</td>
<td>83.3</td>
<td>2, 186</td>
</tr>
<tr>
<td>$\cos(x)$</td>
<td>$[-0.5, 0.25]$</td>
<td>15</td>
<td>r</td>
<td>28</td>
<td>19.5</td>
<td>2, 235</td>
</tr>
<tr>
<td>$\exp(x)$</td>
<td>$[-0.125, 0.125]$</td>
<td>25</td>
<td>r</td>
<td>41</td>
<td>42.3</td>
<td>7, 753</td>
</tr>
<tr>
<td>$\sin(x)$</td>
<td>$[-0.5, 0.5]$</td>
<td>9</td>
<td>a</td>
<td>14</td>
<td>21.5</td>
<td>520</td>
</tr>
<tr>
<td>$\exp(\cos(x)^2 + 1)$</td>
<td>$[1, 2]$</td>
<td>15</td>
<td>r</td>
<td>60</td>
<td>25.5</td>
<td>10, 984</td>
</tr>
<tr>
<td>$\tan(x)$</td>
<td>$[0.25, 0.5]$</td>
<td>10</td>
<td>r</td>
<td>21</td>
<td>26.0</td>
<td>1, 072</td>
</tr>
<tr>
<td>$x^{2.5}$</td>
<td>$[1, 2]$</td>
<td>7</td>
<td>r</td>
<td>26</td>
<td>15.5</td>
<td>1, 362</td>
</tr>
</tbody>
</table>

$^1$Degree of $p$
$^2$Error mode considered: a=absolute, r=relative
$^3$Degree of $T$
$^4$Accuracy
$^5$Timings in ms