A Short Introduction to Interval Analysis

Raazesh Sainudiin* and Warwick Tucker†

*Centre for Computable and Constructive Mathematics,
School of Mathematics and Statistics, University of Canterbury,
Christchurch, New Zealand

†Computer-Aided Proofs in Analysis Group,
Department of Mathematics, Uppsala University,
Uppsala, Sweden

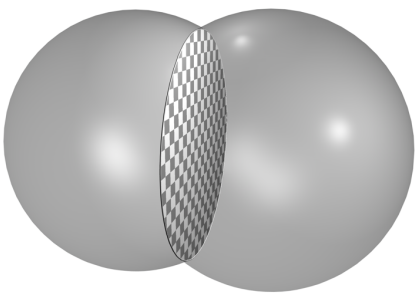
These are mostly extracts from Tucker's "Validated Numerics Talk"

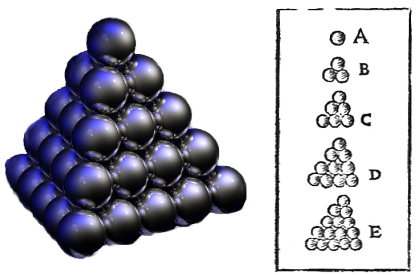
November 20, 2014,

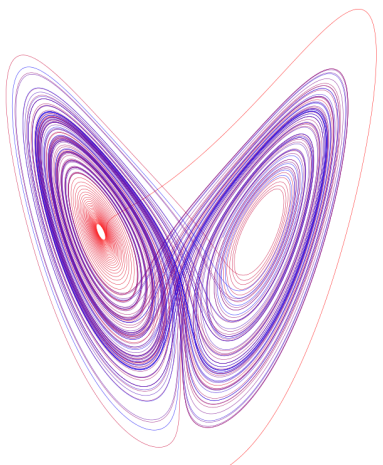
CCNY Mathematics Colloquium, New York, NY, USA

Open Problems Solved Using Interval Analysis

- Double Bubble Conjecture (J. Hass & R. Schlafly, 1995)
- Kepler conjecture about the densest arrangement of spheres in space (T. Hales, 2000).
- Smale's 14th problem on the existence of the Lorentz attractor (W. Tucker, 2002)







Are floating point computations reliable?

Computing with the C/C++ `single` format

UPPSALA
UNIVERSITET

Are floating point computations reliable?

Computing with the `C/C++ single` format

Example 1: Repeated addition

$$\sum_{i=1}^{10^3} \langle 10^{-3} \rangle = 0.999990701675415,$$

$$\sum_{i=1}^{10^4} \langle 10^{-4} \rangle = 1.000053524971008.$$

UPPSALA
UNIVERSITET

Are floating point computations reliable?

Computing with the `C/C++ single` format

Example 1: Repeated addition

$$\sum_{i=1}^{10^3} \langle 10^{-3} \rangle = 0.999990701675415,$$

$$\sum_{i=1}^{10^4} \langle 10^{-4} \rangle = 1.000053524971008.$$

Example 2: Order of summation

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{10^6} = 14.357357,$$

$$\frac{1}{10^6} + \cdots + \frac{1}{3} + \frac{1}{2} + 1 = 14.392651.$$

Are floating point computations reliable?

Given the point $(x, y) = (77617, 33096)$, evaluate the function

$$f(x, y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

Are floating point computations reliable?

Given the point $(x, y) = (77617, 33096)$, evaluate the function

$$f(x, y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

IBM S/370 ($\beta = 16$) with FORTRAN:

type	p	$f(x, y)$
`REAL*4`	24	$1.172603\ldots$
`REAL*8`	53	$1.1726039400531\ldots$
`REAL*10`	64	$1.172603940053178\ldots$

Are floating point computations reliable?

Given the point $(x, y) = (77617, 33096)$, evaluate the function

$$f(x, y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

IBM S/370 ($\beta = 16$) with FORTRAN:

type	p	$f(x, y)$
`REAL*4`	24	$1.172603\ldots$
`REAL*8`	53	$1.1726039400531\ldots$
`REAL*10`	64	$1.172603940053178\ldots$

Pentium III ($\beta = 2$) with C/C++ (gcc/g++):

type	p	$f(x, y)$
`float`	24	$1.7870283321406128e+20$
`double`	53	$1.7870283321406128e+20$
`long double`	64	$1.7870283321406128e+20$

UPPSALA UNIVERSITET

Are floating point computations reliable?

Given the point $(x, y) = (77617, 33096)$, evaluate the function

$$f(x, y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

IBM S/370 ($\beta = 16$) with FORTRAN:

type	p	$f(x, y)$
`REAL*4`	24	$1.172603\ldots$
`REAL*8`	53	$1.1726039400531\ldots$
`REAL*10`	64	$1.172603940053178\ldots$

Pentium III ($\beta = 2$) with C/C++ (gcc/g++):

type	p	$f(x, y)$
`float`	24	$1.7870283321406128 \times 10^{20}$
`double`	53	$1.7870283321406128 \times 10^{20}$
`long double`	64	$1.7870283321406128 \times 10^{20}$

Correct answer: $-0.8273960599\ldots$

Are integer computations reliable?

Example 3: The harmonic series

If we define

$$S_N = \sum_{k=1}^{N} \frac{1}{k},$$

then $\lim_{N\to\infty} S_N = +\infty$. The computer also gets this result, but for the entirely wrong reason (integer wrapping).

$$I_{max} + 1 = -I_{max} = I_{min}$$

Are integer computations reliable?

Example 3: The harmonic series

If we define

$$S_N = \sum_{k=1}^{N} \frac{1}{k},$$

then $\lim_{N\to\infty} S_N = +\infty$. The computer also gets this result, but for the entirely wrong reason (integer wrapping).

$$I_{max} + 1 = -I_{max} = I_{min}$$

Example 4: Elementary Taylor series

Now define

$$S_N = \sum_{k=0}^{N} \frac{1}{k!},$$

then $\lim_{N\to\infty} S_N = e \approx 2.7182818$. The integer wrapping produces a sequence that is *not* strictly increasing:

Are integer computations reliable?

```
S_0  = 1.000000000000000        S_18 = 2.718281826004540  S
S_1  = 2.000000000000000        S_19 = 2.718281835125155
S_2  = 2.500000000000000        S_20 = 2.718281834649448  S
S_3  = 2.666666666666667        S_21 = 2.718281833812708  S
S_4  = 2.708333333333333        S_22 = 2.718281831899620  S
S_5  = 2.716666666666666        S_23 = 2.718281833059102
S_6  = 2.718055555555555        S_24 = 2.718281831770353  S
S_7  = 2.718253968253968        S_25 = 2.718281832252007
S_8  = 2.718278769841270        S_26 = 2.718281831712599  S
S_9  = 2.718281525573192        S_27 = 2.718281832386097
S_10 = 2.718281801146385        S_28 = 2.718281831659211  S
S_11 = 2.718281826198493        S_29 = 2.718281830853743  S
S_12 = 2.718281828286169        S_30 = 2.718281831563322
S_13 = 2.718281828803753        S_31 = 2.718281832917973
S_14 = 2.718281829585647        S_32 = 2.718281832452312  S
S_15 = 2.718281830084572        S_33 = 2.718281831986650  S
S_16 = 2.718281830583527        S_34 =              inf
S_17 = 2.718281827117590  S
```

UPPSALA
UNIVERSITET

How do we control rounding errors?

Round each partial result both ways

If $x, y \in \mathbb{F}$ and $\star \in \{+, -, \times, \div\}$, we can enclose the exact result in an *interval*:

$$x \star y \in [\bigtriangledown(x \star y), \bigtriangleup(x \star y)].$$

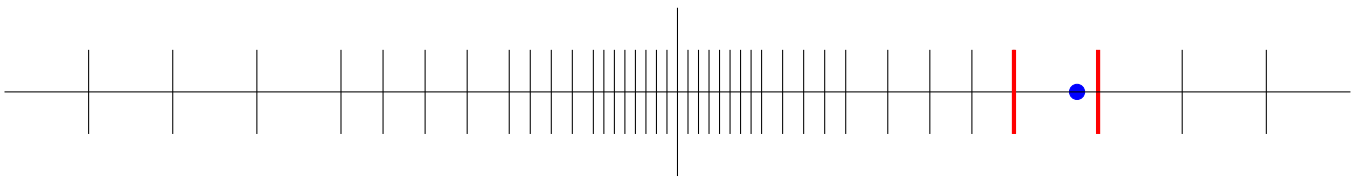
How do we control rounding errors?

Round each partial result both ways

If $x, y \in \mathbb{F}$ and $\star \in \{+, -, \times, \div\}$, we can enclose the exact result in an *interval*:

$$x \star y \in [\bigtriangledown(x \star y), \bigtriangleup(x \star y)].$$

Since all (modern) computers round with *maximal quality*, the interval is the smallest one that contains the exact result.



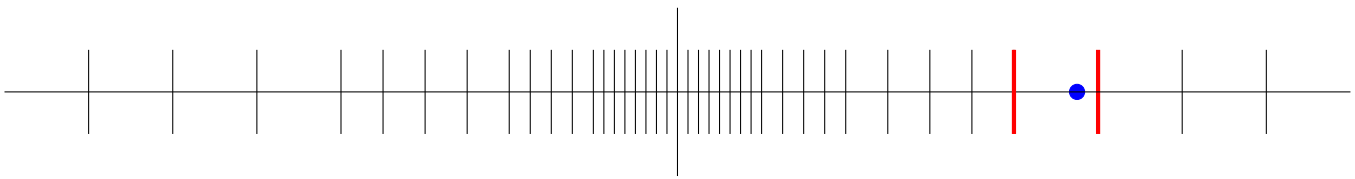
How do we control rounding errors?

Round each partial result both ways

If $x, y \in \mathbb{F}$ and $\star \in \{+, -, \times, \div\}$, we can enclose the exact result in an *interval*:

$$x \star y \in [\bigtriangledown(x \star y), \bigtriangleup(x \star y)].$$

Since all (modern) computers round with *maximal quality*, the interval is the smallest one that contains the exact result.



Question

How do we compute with intervals? And why, really?

Arithmetic over $\mathbb{R}$

Definition

If $\star$ is one of the operators $+, -, \times, \div$, and if $\mathbb{A}, \mathbb{B} \in \mathbb{R}$, then

$$\mathbb{A} \star \mathbb{B} = \{a \star b \colon a \in \mathbb{A}, b \in \mathbb{B}\},$$

except that $\mathbb{A} \div \mathbb{B}$ is undefined if $0 \in \mathbb{B}$.

Arithmetic over $\mathbb{R}$

Definition

If $\star$ is one of the operators $+, -, \times, \div$, and if $\mathbb{A}, \mathbb{B} \in \mathbb{R}$, then

$$\mathbb{A} \star \mathbb{B} = \{a \star b \colon a \in \mathbb{A}, b \in \mathbb{B}\},$$

except that $\mathbb{A} \div \mathbb{B}$ is undefined if $0 \in \mathbb{B}$.

Simple arithmetic

$$\begin{aligned}
\mathbb{A} + \mathbb{B} &= [\underline{\mathbb{A}} + \underline{\mathbb{B}}, \overline{\mathbb{A}} + \overline{\mathbb{B}}] \\
\mathbb{A} - \mathbb{B} &= [\underline{\mathbb{A}} - \overline{\mathbb{B}}, \overline{\mathbb{A}} - \underline{\mathbb{B}}] \\
\mathbb{A} \times \mathbb{B} &= [\min\{\underline{\mathbb{A}\mathbb{B}}, \underline{\mathbb{A}}\overline{\mathbb{B}}, \overline{\mathbb{A}}\underline{\mathbb{B}}, \overline{\mathbb{A}\mathbb{B}}\}, \max\{\underline{\mathbb{A}\mathbb{B}}, \underline{\mathbb{A}}\overline{\mathbb{B}}, \overline{\mathbb{A}}\underline{\mathbb{B}}, \overline{\mathbb{A}\mathbb{B}}\}] \\
\mathbb{A} \div \mathbb{B} &= \mathbb{A} \times [1/\overline{\mathbb{B}}, 1/\underline{\mathbb{B}}], \quad \text{if } 0 \notin \mathbb{B}.
\end{aligned}$$

Arithmetic over $\mathbb{R}$

Definition

If $\star$ is one of the operators $+, -, \times, \div$, and if $\mathbb{A}, \mathbb{B} \in \mathbb{R}$, then

$$\mathbb{A} \star \mathbb{B} = \{a \star b \colon a \in \mathbb{A}, b \in \mathbb{B}\},$$

except that $\mathbb{A} \div \mathbb{B}$ is undefined if $0 \in \mathbb{B}$.

Simple arithmetic

$$\begin{aligned}
\mathbb{A} + \mathbb{B} &= [\underline{\mathbb{A}} + \underline{\mathbb{B}}, \overline{\mathbb{A}} + \overline{\mathbb{B}}] \\
\mathbb{A} - \mathbb{B} &= [\underline{\mathbb{A}} - \overline{\mathbb{B}}, \overline{\mathbb{A}} - \underline{\mathbb{B}}] \\
\mathbb{A} \times \mathbb{B} &= [\min\{\underline{\mathbb{A}\mathbb{B}}, \underline{\mathbb{A}}\overline{\mathbb{B}}, \overline{\mathbb{A}}\underline{\mathbb{B}}, \overline{\mathbb{A}\mathbb{B}}\}, \max\{\underline{\mathbb{A}\mathbb{B}}, \underline{\mathbb{A}}\overline{\mathbb{B}}, \overline{\mathbb{A}}\underline{\mathbb{B}}, \overline{\mathbb{A}\mathbb{B}}\}] \\
\mathbb{A} \div \mathbb{B} &= \mathbb{A} \times [1/\overline{\mathbb{B}}, 1/\underline{\mathbb{B}}], \quad \text{if } 0 \notin \mathbb{B}.
\end{aligned}$$

On a computer we use *directed rounding*, e.g.

$$\mathbb{A} + \mathbb{B} = [\bigtriangledown(\underline{\mathbb{A}} \oplus \underline{\mathbb{B}}), \bigtriangleup(\overline{\mathbb{A}} \oplus \overline{\mathbb{B}})].$$

UPPSALA UNIVERSITET

Interval-valued functions

Range enclosure

Extend a real-valued function f to an interval-valued F:

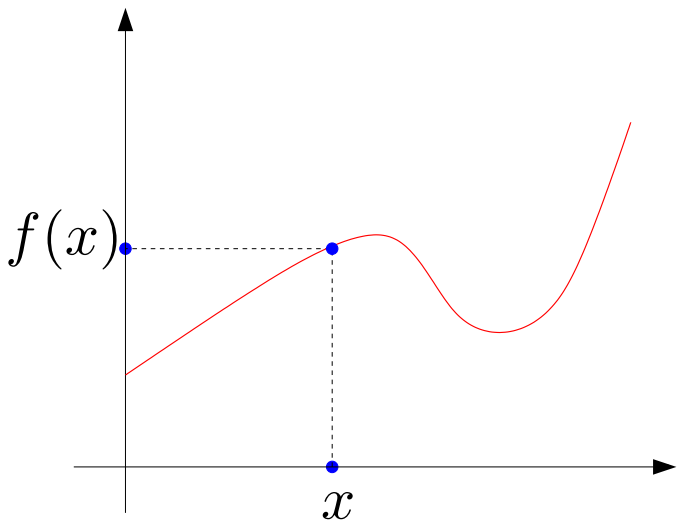
$$R(f;\mathbb{X}) = \{f(x) \colon x \in \mathbb{X}\} \subseteq F(\mathbb{X})$$

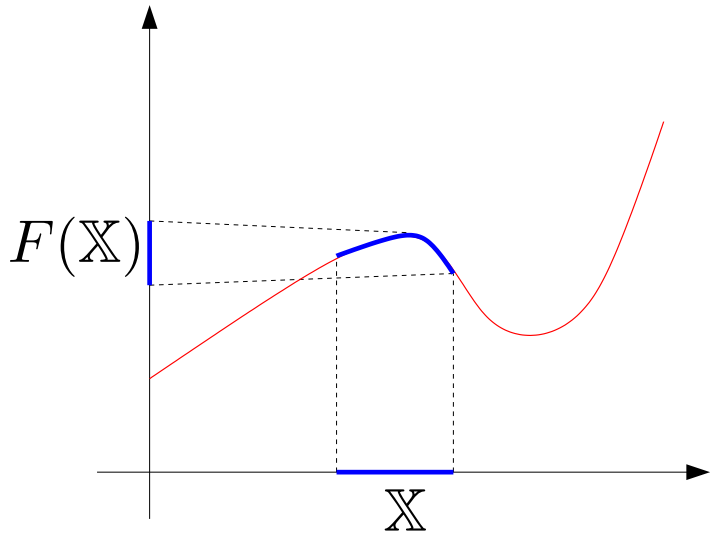
Interval-valued functions

Range enclosure

Extend a real-valued function f to an interval-valued F:

$$R(f;\mathbb{X}) = \{f(x) : x \in \mathbb{X}\} \subseteq F(\mathbb{X})$$



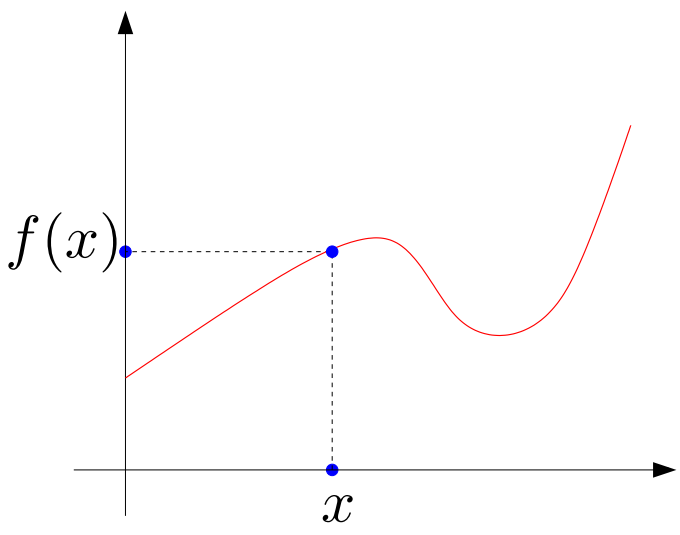


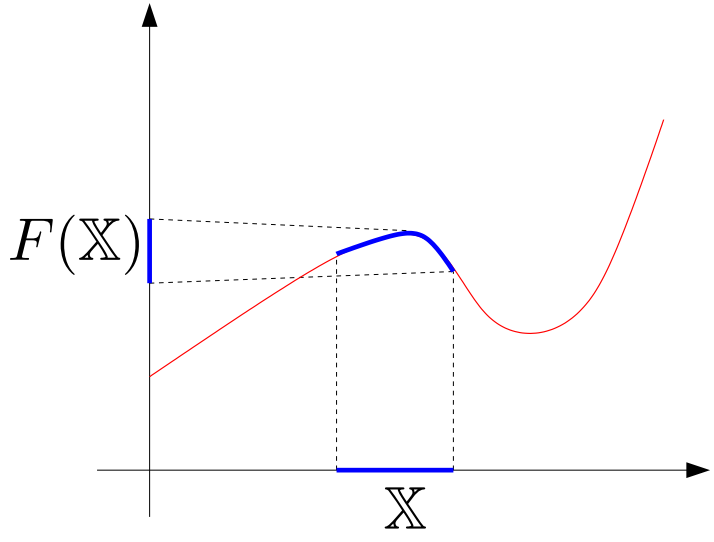
Interval-valued functions

Range enclosure

Extend a real-valued function f to an interval-valued F:

$$R(f;\mathbb{X}) = \{f(x) \colon x \in \mathbb{X}\} \subseteq F(\mathbb{X})$$





$y \notin F(\mathbb{X})$ implies that $f(x) \neq y$ for all $x \in \mathbb{X}$.

UPPSALA UNIVERSITET

Interval-valued functions

Some explicit formulas are given below:

UPPSALA
UNIVERSITET

Interval-valued functions

Some explicit formulas are given below:

$$
\begin{array}{lcll}
e^{\mathbb{X}} & = & [e^{\underline{\mathbb{X}}}, e^{\overline{\mathbb{X}}}] & \\
\sqrt{\mathbb{X}} & = & [\sqrt{\underline{\mathbb{X}}}, \sqrt{\overline{\mathbb{X}}}] & \text{if } 0 \leq \underline{\mathbb{X}} \\
\log \mathbb{X} & = & [\log \underline{\mathbb{X}}, \log \overline{\mathbb{X}}] & \text{if } 0 < \underline{\mathbb{X}} \\
\arctan \mathbb{X} & = & [\arctan \underline{\mathbb{X}}, \arctan \overline{\mathbb{X}}] & .
\end{array}
$$

Interval-valued functions

Some explicit formulas are given below:

$$\begin{array}{lcll} e^{\mathbb{X}} & = & [e^{\underline{\mathbb{X}}}, e^{\overline{\mathbb{X}}}] & \\ \sqrt{\mathbb{X}} & = & [\sqrt{\underline{\mathbb{X}}}, \sqrt{\overline{\mathbb{X}}}] & \text{if } 0 \leq \underline{\mathbb{X}} \\ \log \mathbb{X} & = & [\log \underline{\mathbb{X}}, \log \overline{\mathbb{X}}] & \text{if } 0 < \underline{\mathbb{X}} \\ \arctan \mathbb{X} & = & [\arctan \underline{\mathbb{X}}, \arctan \overline{\mathbb{X}}] & . \end{array}$$

Set $S^+ = \{2k\pi + \pi/2 \colon k \in \mathbb{Z}\}$ and $S^- = \{2k\pi - \pi/2 \colon k \in \mathbb{Z}\}$.
Then $\sin \mathbb{X}$ is given by

$$\begin{cases} [-1, 1] & : \text{ if } \mathbb{X} \cap S^- \neq \emptyset \text{ and } \mathbb{X} \cap S^+ \neq \emptyset, \\ [-1, \max\{\sin \underline{\mathbb{X}}, \sin \overline{\mathbb{X}}\}] & : \text{ if } \mathbb{X} \cap S^- \neq \emptyset \text{ and } \mathbb{X} \cap S^+ = \emptyset, \\ [\min\{\sin \underline{\mathbb{X}}, \sin \overline{\mathbb{X}}\}, 1] & : \text{ if } \mathbb{X} \cap S^- = \emptyset \text{ and } \mathbb{X} \cap S^+ \neq \emptyset, \\ [\min\{\sin \underline{\mathbb{X}}, \sin \overline{\mathbb{X}}\}, \max\{\sin \underline{\mathbb{X}}, \sin \overline{\mathbb{X}}\}] & : \text{ if } \mathbb{X} \cap S^- = \emptyset \text{ and } \mathbb{X} \cap S^+ = \emptyset. \end{cases}$$

UPPSALA
UNIVERSITET

Graph Enclosures

Example

Draw an accurate graph of the function $f(x) = \cos^3 x + \sin x$ over the domain $[-5, 5]$.

UPPSALA
UNIVERSITET

Graph Enclosures

Example

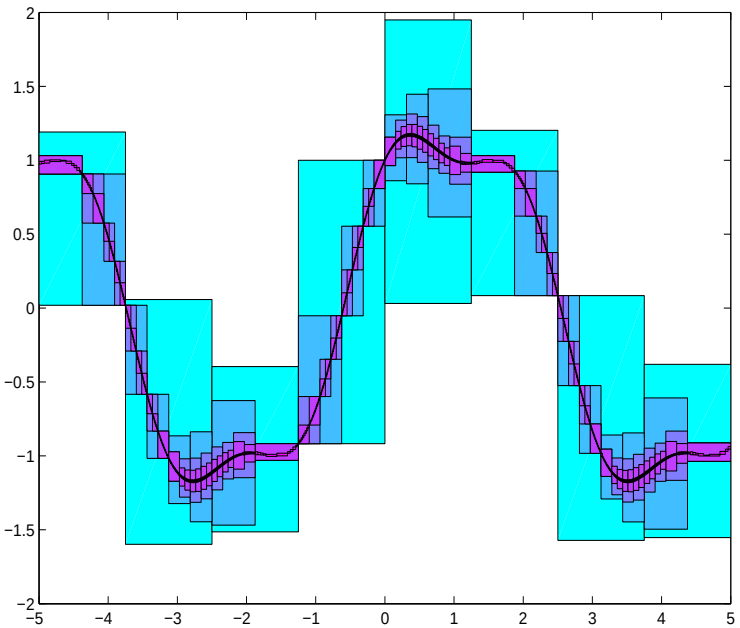
Draw an accurate graph of the function $f(x) = \cos^3 x + \sin x$ over the domain $[-5, 5]$.
Define $F(\mathbb{X}) = \cos^3 \mathbb{X} + \sin \mathbb{X}$, and adaptively bisect the domain $\mathbb{X}_0 = [-5, 5]$ into smaller pieces $\mathbb{X}_0 = \cup_{i=1}^{N} \mathbb{X}_i$ until we arrive at some desired accuracy, e.g. $\max_i \text{width}\big(F(\mathbb{X}_i)\big) \leq \texttt{TOL}$.

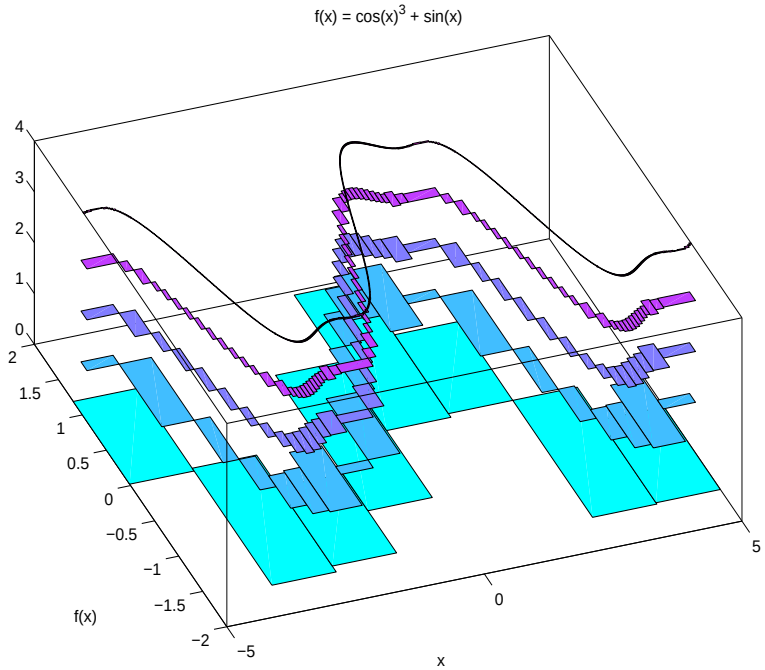
UPPSALA
UNIVERSITET

Graph Enclosures

Example

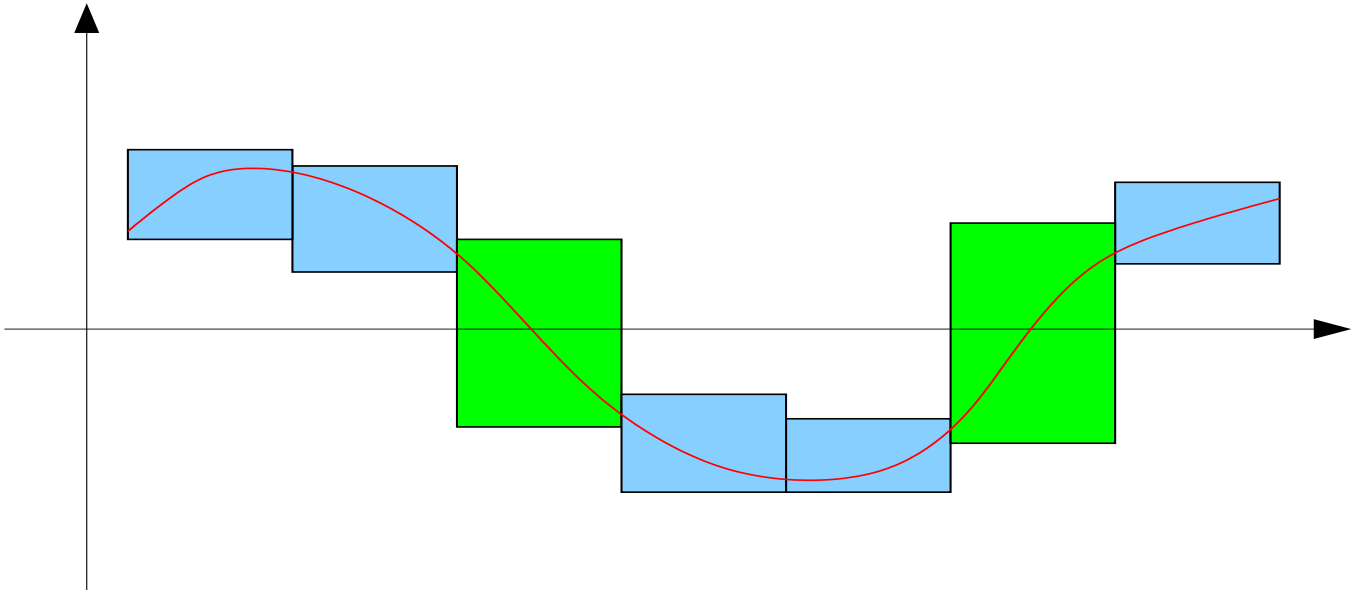
Draw an accurate graph of the function $f(x) = \cos^3 x + \sin x$ over the domain $[-5, 5]$.
Define $F(\mathbb{X}) = \cos^3 \mathbb{X} + \sin \mathbb{X}$, and adaptively bisect the domain $\mathbb{X}_0 = [-5, 5]$ into smaller pieces $\mathbb{X}_0 = \cup_{i=1}^{N} \mathbb{X}_i$ until we arrive at some desired accuracy, e.g. $\max_i \text{width}\big(F(\mathbb{X}_i)\big) \leq \texttt{TOL}$.



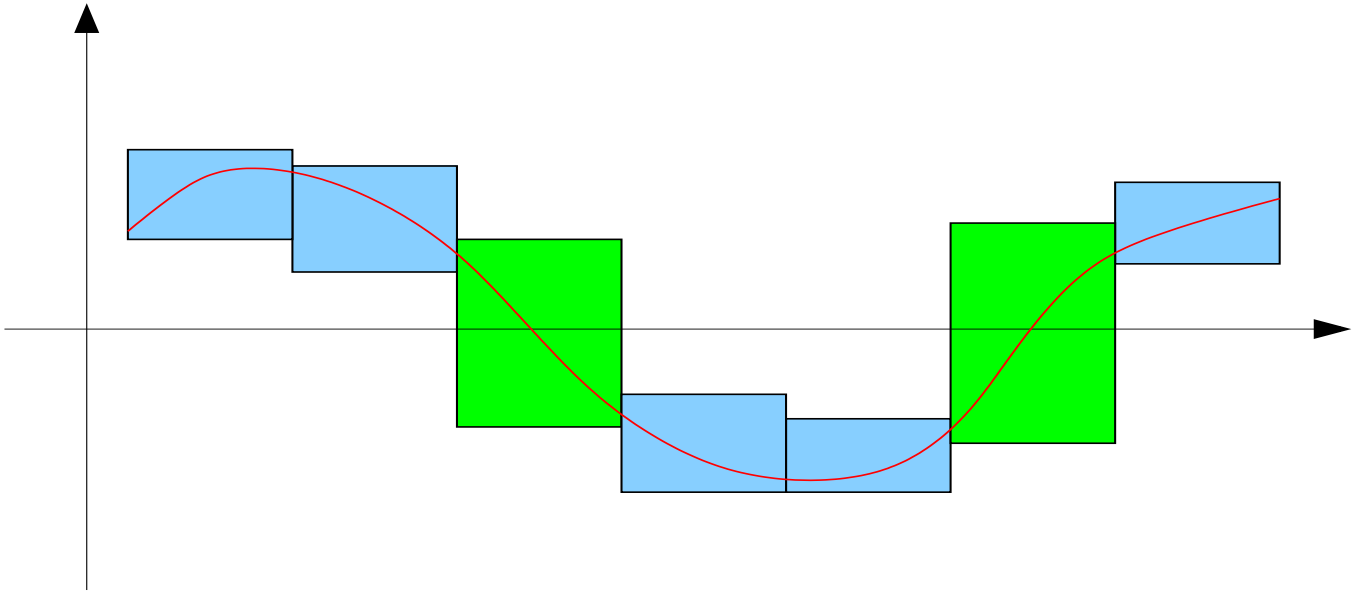


UPPSALA UNIVERSITET

Solving non-linear equations

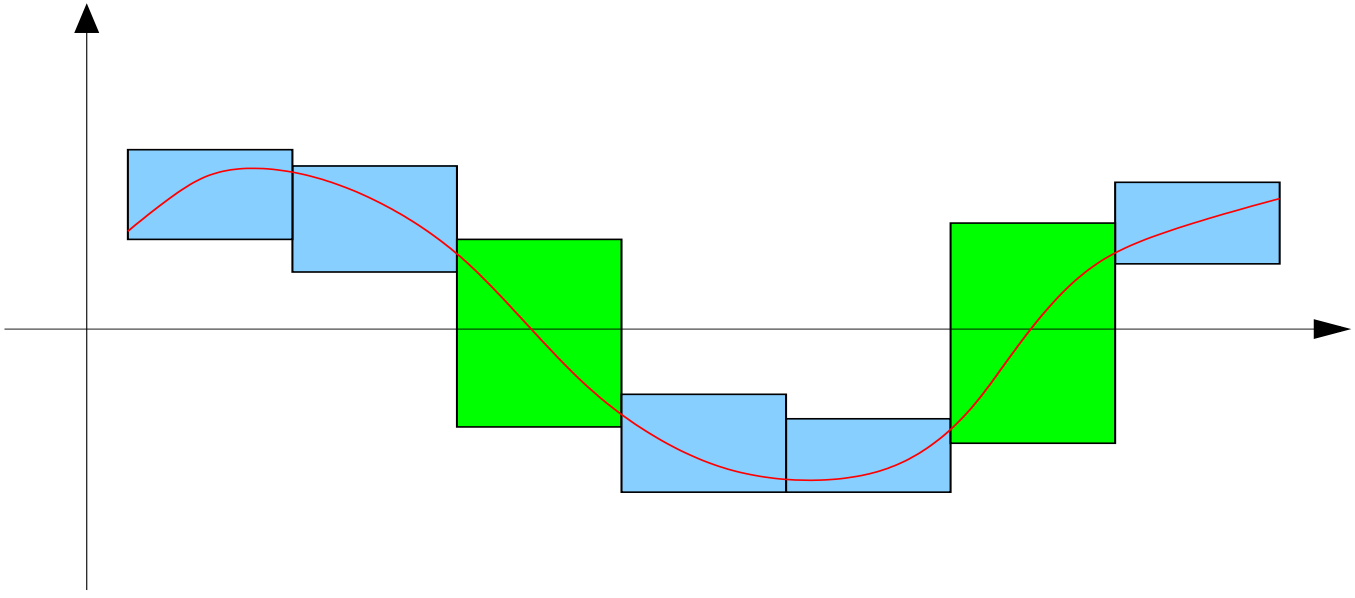


Solving non-linear equations



Consider everything. Keep what is good.
Avoid evil whenever you recognize it.
St. Paul, ca. 50 A.D. (The Bible, 1 Thess. 5:21-22)

Solving non-linear equations



Consider everything. Keep what is good.
Avoid evil whenever you recognize it.
St. Paul, ca. 50 A.D. (The Bible, 1 Thess. 5:21-22)

No solutions can be missed!

UPPSALA
UNIVERSITET

Solving non-linear equation

The code is transparent and natural

```
01 function bisect(fcnName, X, tol)
02 f = inline(fcnName);
03 if ( 0 <= f(X) )          % If f(X) contains zero...
04     if Diam(X) < tol      % and the tolerance is met...
05         X                 % print the interval X.
06     else                  % Otherwise, divide and conquer.
07         bisect(fcnName, interval(Inf(X), Mid(X)), tol);
08         bisect(fcnName, interval(Mid(X), Sup(X)), tol);
09     end
10 end
```

UPPSALA
UNIVERSITET

Solving non-linear equation

The code is transparent and natural

```
01 function bisect(fcnName, X, tol)
02 f = inline(fcnName);
03 if ( 0 <= f(X) )          % If f(X) contains zero...
04     if Diam(X) < tol      % and the tolerance is met...
05         X                 % print the interval X.
06     else                  % Otherwise, divide and conquer.
07         bisect(fcnName, interval(Inf(X), Mid(X)), tol);
08         bisect(fcnName, interval(Mid(X), Sup(X)), tol);
09     end
10 end
```

Nice property

If F is well-defined on the domain, the algorithm produces an enclosure of *all* zeros of f. [No existence is established, however.]

UPPSALA
UNIVERSITET

Existence and uniqueness

Existence and uniqueness require *fixed point* theorems.

Existence and uniqueness

Existence and uniqueness require *fixed point* theorems.

Brouwer's fixed point theorem

Let B be homeomorhpic to the closed unit ball in $\mathbb{R}^n$. Then given any continuous mapping $f : B \to B$ there exists $x \in B$ such that $f(x) = x$.

Existence and uniqueness

Existence and uniqueness require *fixed point* theorems.

Brouwer's fixed point theorem

Let B be homeomorhpic to the closed unit ball in $\mathbb{R}^n$. Then given any continuous mapping $f : B \to B$ there exists $x \in B$ such that $f(x) = x$.

Schauder's fixed point theorem

Let X be a normed vector space, and let $K \subset X$ be a non-empty, compact, and convex set. Then given any continuous mapping $f : K \to K$ there exists $x \in K$ such that $f(x) = x$.

Existence and uniqueness

Existence and uniqueness require *fixed point* theorems.

Brouwer's fixed point theorem

Let B be homeomorhpic to the closed unit ball in $\mathbb{R}^n$. Then given any continuous mapping $f : B \to B$ there exists $x \in B$ such that $f(x) = x$.

Schauder's fixed point theorem

Let X be a normed vector space, and let $K \subset X$ be a non-empty, compact, and convex set. Then given any continuous mapping $f : K \to K$ there exists $x \in K$ such that $f(x) = x$.

Banach's fixed point theorem

If f is a contraction defined on a complete metric space X, then there exists a unique $x \in X$ such that $f(x) = x$.

UPPSALA UNIVERSITET

Newton's method in $\mathbb{R}$

Theorem

Let $f \in C^1(\mathbb{R}, \mathbb{R})$, and set $\check{x} = \mathrm{mid}(\mathbb{X})$. We define

$$N_f(\mathbb{X}) \stackrel{def}{=} N_f(\mathbb{X}, \check{x}) = \check{x} - f(\check{x})/F'(\mathbb{X}).$$

If $N_f(\mathbb{X})$ is well-defined, then the following statements hold:

Newton's method in $\mathbb{R}$

Theorem

Let $f \in C^1(\mathbb{R}, \mathbb{R})$, and set $\check{x} = \mathrm{mid}(\mathbb{X})$. We define

$$N_f(\mathbb{X}) \overset{def}{=} N_f(\mathbb{X}, \check{x}) = \check{x} - f(\check{x})/F'(\mathbb{X}).$$

If $N_f(\mathbb{X})$ is well-defined, then the following statements hold:

(1) *if $\mathbb{X}$ contains a zero x^* of f, then so does $N_f(\mathbb{X}) \cap \mathbb{X}$;*

Newton's method in $\mathbb{R}$

Theorem

Let $f \in C^1(\mathbb{R}, \mathbb{R})$, and set $\check{x} = \mathrm{mid}(\mathbb{X})$. We define

$$N_f(\mathbb{X}) \overset{def}{=} N_f(\mathbb{X}, \check{x}) = \check{x} - f(\check{x})/F'(\mathbb{X}).$$

If $N_f(\mathbb{X})$ is well-defined, then the following statements hold:

(1) *if $\mathbb{X}$ contains a zero x^* of f, then so does $N_f(\mathbb{X}) \cap \mathbb{X}$;*

(2) *if $N_f(\mathbb{X}) \cap \mathbb{X} = \emptyset$, then $\mathbb{X}$ contains no zeros of f;*

UPPSALA
UNIVERSITET

Newton's method in $\mathbb{R}$

Theorem

Let $f \in C^1(\mathbb{R}, \mathbb{R})$, and set $\check{x} = \mathrm{mid}(\mathbb{X})$. We define

$$N_f(\mathbb{X}) \overset{def}{=} N_f(\mathbb{X}, \check{x}) = \check{x} - f(\check{x})/F'(\mathbb{X}).$$

If $N_f(\mathbb{X})$ is well-defined, then the following statements hold:

(1) *if $\mathbb{X}$ contains a zero x^* of f, then so does $N_f(\mathbb{X}) \cap \mathbb{X}$;*

(2) *if $N_f(\mathbb{X}) \cap \mathbb{X} = \emptyset$, then $\mathbb{X}$ contains no zeros of f;*

(3) *if $N_f(\mathbb{X}) \subseteq \mathbb{X}$, then $\mathbb{X}$ contains a unique zero of f.*

UPPSALA
UNIVERSITET

Newton's method in $\mathbb{R}$

Theorem

Let $f \in C^1(\mathbb{R}, \mathbb{R})$, and set $\check{x} = \mathrm{mid}(\mathbb{X})$. We define

$$N_f(\mathbb{X}) \overset{def}{=} N_f(\mathbb{X}, \check{x}) = \check{x} - f(\check{x})/F'(\mathbb{X}).$$

If $N_f(\mathbb{X})$ is well-defined, then the following statements hold:

(1) *if $\mathbb{X}$ contains a zero x^* of f, then so does $N_f(\mathbb{X}) \cap \mathbb{X}$;*

(2) *if $N_f(\mathbb{X}) \cap \mathbb{X} = \emptyset$, then $\mathbb{X}$ contains no zeros of f;*

(3) *if $N_f(\mathbb{X}) \subseteq \mathbb{X}$, then $\mathbb{X}$ contains a unique zero of f.*

Proof.

(1) Follows from the MVT;

(2) The contra-positive statement of (1);

(3) Existence from Brouwer's fixed point theorem;
Uniqueness from non-vanishing f'.

□

Newton's method in $\mathbb{R}$

Algorithm

Starting from an initial search region $\mathbb{X}_0$, we form the sequence

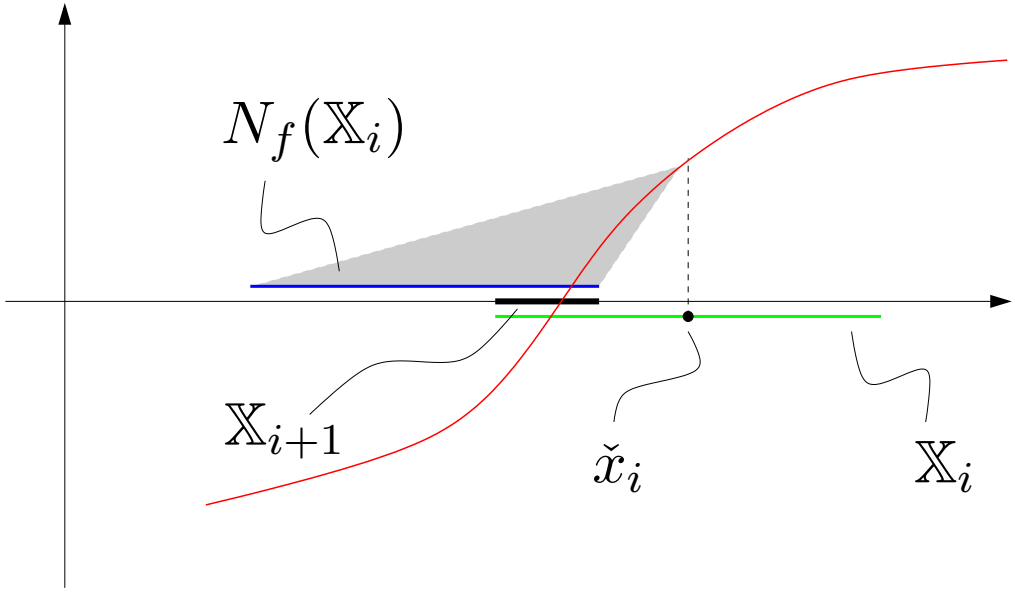
$$\mathbb{X}_{i+1} = N_f(\mathbb{X}_i) \cap \mathbb{X}_i \qquad i = 0, 1, \ldots .$$

Newton's method in $\mathbb{R}$

Algorithm

Starting from an initial search region $\mathbb{X}_0$, we form the sequence

$$\mathbb{X}_{i+1} = N_f(\mathbb{X}_i) \cap \mathbb{X}_i \qquad i = 0, 1, \ldots .$$

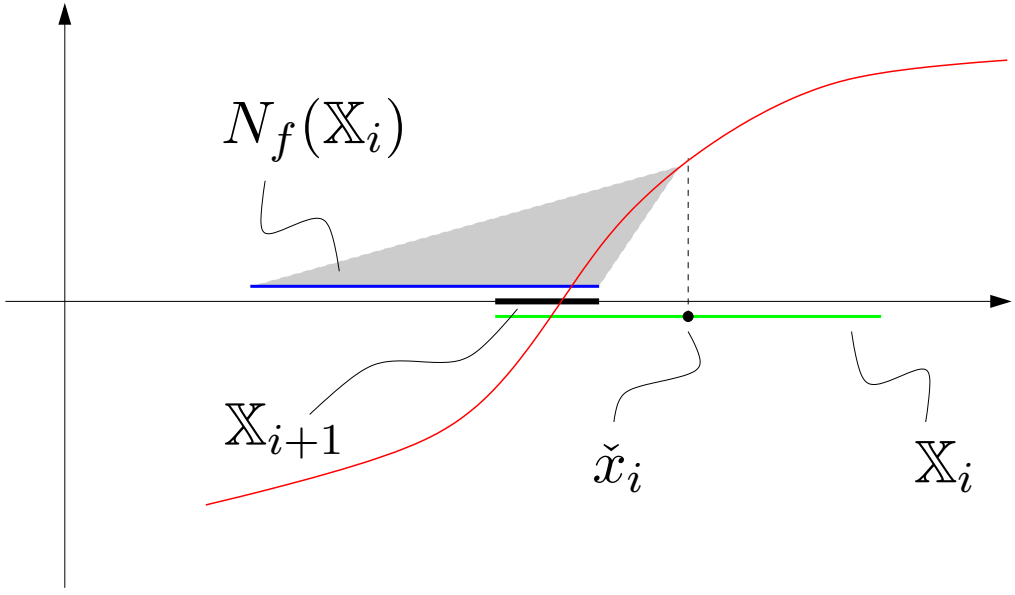


Newton's method in $\mathbb{R}$

Algorithm

Starting from an initial search region $\mathbb{X}_0$, we form the sequence

$$\mathbb{X}_{i+1} = N_f(\mathbb{X}_i) \cap \mathbb{X}_i \qquad i = 0, 1, \ldots .$$



Performance

If well-defined, this method is never worse than bisection, and it converges quadratically fast under mild conditions.

Newton's method in $\mathbb{R}$

Example

Let $f(x) = -2.001 + 3x - x^3$, and $\mathbb{X}_0 = [-3, -3/2]$. Then $F'(\mathbb{X}_0) = [-24, -15/4]$, so $N_f(\mathbb{X}_0)$ is well-defined, and the above theorem holds.

Newton's method in $\mathbb{R}$

Example

Let $f(x) = -2.001 + 3x - x^3$, and $\mathbb{X}_0 = [-3, -3/2]$. Then $F'(\mathbb{X}_0) = [-24, -15/4]$, so $N_f(\mathbb{X}_0)$ is well-defined, and the above theorem holds.

```
X(0) = [-3.00000000000000,-1.50000000000000]; rad = 7.50000e-01
X(1) = [-2.14001562500001,-1.54609999999996]; rad = 2.96958e-01
X(2) = [-2.14001562500001,-1.96127739828410]; rad = 8.93691e-02
X(3) = [-2.00684923964035,-1.99557058024720]; rad = 5.63933e-03
X(4) = [-2.00012010448627,-2.00010360853027]; rad = 8.24798e-06
X(5) = [-2.00011110289039,-2.00011110287381]; rad = 8.28893e-12
X(6) = [-2.00011110288173,-2.00011110288172]; rad = 1.55431e-15
X(7) = [-2.00011110288173,-2.00011110288172]; rad = 1.55431e-15
Finite convergence!
Unique root in -2.00011110288172 +- 1.555e-15
```

Newton's method in $\mathbb{R}$

Example

Let $f(x) = -2.001 + 3x - x^3$, and $\mathbb{X}_0 = [-3, -3/2]$. Then $F'(\mathbb{X}_0) = [-24, -15/4]$, so $N_f(\mathbb{X}_0)$ is well-defined, and the above theorem holds.

```
X(0) = [-3.00000000000000,-1.50000000000000]; rad = 7.50000e-01
X(1) = [-2.14001562500001,-1.54609999999996]; rad = 2.96958e-01
X(2) = [-2.14001562500001,-1.96127739828410]; rad = 8.93691e-02
X(3) = [-2.00684923964035,-1.99557058024720]; rad = 5.63933e-03
X(4) = [-2.00012010448627,-2.00010360853027]; rad = 8.24798e-06
X(5) = [-2.00011110289039,-2.00011110287381]; rad = 8.28893e-12
X(6) = [-2.00011110288173,-2.00011110288172]; rad = 1.55431e-15
X(7) = [-2.00011110288173,-2.00011110288172]; rad = 1.55431e-15
Finite convergence!
Unique root in -2.00011110288172 +- 1.555e-15
```

Question:

What do we do when $\mathbb{X}_0$ contains several zeros?

The Krawczyk method

If f has a zero x^* in $\mathbb{X}$, then for any $x \in \mathbb{X}$, we can enclose the zero via

$$x^* \in x - Cf(x) - \big(1 - CF'(\mathbb{X})\big)(x - \mathbb{X}) \stackrel{\text{def}}{=} K_f(\mathbb{X}, x, C).$$

The Krawczyk method

If f has a zero x^* in $\mathbb{X}$, then for any $x \in \mathbb{X}$, we can enclose the zero via

$$x^* \in x - Cf(x) - \big(1 - CF'(\mathbb{X})\big)(x - \mathbb{X}) \stackrel{\text{def}}{=} K_f(\mathbb{X}, x, C).$$

Good choices are $x = \check{x}$ and $C = 1/f'(\check{x})$, yielding the *Krawczyk operator*

$$K_f(\mathbb{X}) \stackrel{\text{def}}{=} \check{x} - \frac{f(\check{x})}{f'(\check{x})} - \left(1 - \frac{F'(\mathbb{X})}{f'(\check{x})}\right)[-r, r],$$

where we use the notation $r = \mathrm{rad}(\mathbb{X})$.

The Krawczyk method

If f has a zero x^* in $\mathbb{X}$, then for any $x \in \mathbb{X}$, we can enclose the zero via

$$x^* \in x - Cf(x) - \big(1 - CF'(\mathbb{X})\big)(x - \mathbb{X}) \stackrel{\text{def}}{=} K_f(\mathbb{X}, x, C).$$

Good choices are $x = \check{x}$ and $C = 1/f'(\check{x})$, yielding the *Krawczyk operator*

$$K_f(\mathbb{X}) \stackrel{\text{def}}{=} \check{x} - \frac{f(\check{x})}{f'(\check{x})} - \left(1 - \frac{F'(\mathbb{X})}{f'(\check{x})}\right)[-r, r],$$

where we use the notation $r = \mathrm{rad}(\mathbb{X})$.

Theorem

Assume that $K_f(\mathbb{X})$ is well-defined. Then the following statements hold:

(1) if $\mathbb{X}$ contains a zero x^* of f, then so does $K_f(\mathbb{X}) \cap \mathbb{X}$;

(2) if $K_f(\mathbb{X}) \cap \mathbb{X} = \emptyset$, then $\mathbb{X}$ contains no zeros of f;

(3) if $K_f(\mathbb{X}) \subseteq \mathrm{int}(\mathbb{X})$, then $\mathbb{X}$ contains a unique zero of f.

The Krawczyk method with bisection

Example

Let $f(x) = \sin(e^x + 1)$, and $\mathbb{X}_0 = [0, 3]$.

The Krawczyk method with bisection

Example

Let $f(x) = \sin(e^x + 1)$, and $\mathbb{X}_0 = [0, 3]$.

```
Domain           : [0, 3]
Tolerance        : 1e-10
Function calls   : 71
 Unique zero in the interval 0.761549782880[8890,9006]
 Unique zero in the interval 1.664529193[6825445,7060436]
 Unique zero in the interval 2.131177121086[2673,3558]
 Unique zero in the interval 2.4481018026567[773,801]
 Unique zero in the interval 2.68838906601606[36,68]
 Unique zero in the interval 2.8819786295709[728,1555]
```

UPPSALA
UNIVERSITET

The Krawczyk method with bisection

Example

Let $f(x) = \sin(e^x + 1)$, and $\mathbb{X}_0 = [0, 3]$.

```
Domain           : [0, 3]
Tolerance        : 1e-10
Function calls   : 71
 Unique zero in the interval 0.761549782880[8890,9006]
 Unique zero in the interval 1.664529193[6825445,7060436]
 Unique zero in the interval 2.131177121086[2673,3558]
 Unique zero in the interval 2.4481018026567[773,801]
 Unique zero in the interval 2.68838906601606[36,68]
 Unique zero in the interval 2.8819786295709[728,1555]
```

Applications

Counting short periodic orbits for ODEs [Z. Galias]

Measuring the stable regions of the quadratic map [D. Wilczak]

Implicit curves

Example

Draw the level-set defined by

$$f(x, y) = \sin\left(\cos x^2 + 10 \sin y^2\right) - y \cos x = 0;$$

restricted to the domain $[-5, 5] \times [-5, 5]$.

Implicit curves

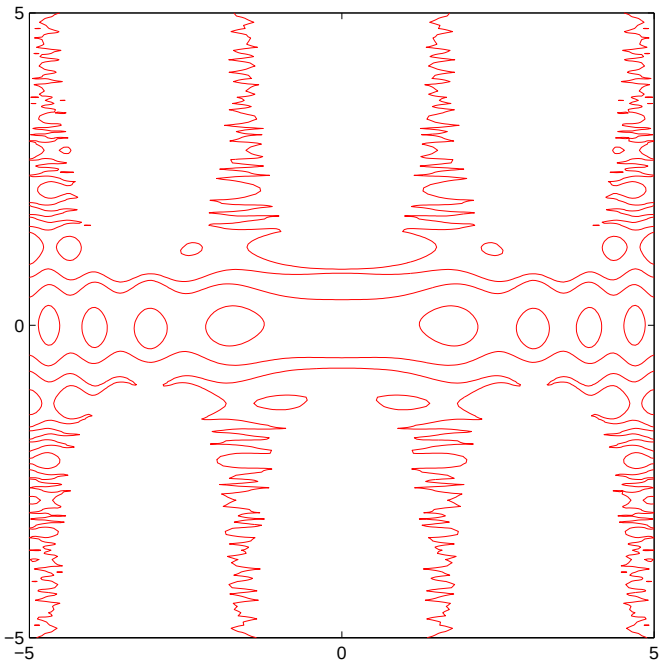
Example

Draw the level-set defined by

$$f(x, y) = \sin\left(\cos x^2 + 10 \sin y^2\right) - y \cos x = 0;$$

restricted to the domain $[-5, 5] \times [-5, 5]$.

MATLAB produces the following picture:



UPPSALA UNIVERSITET

Implicit curves

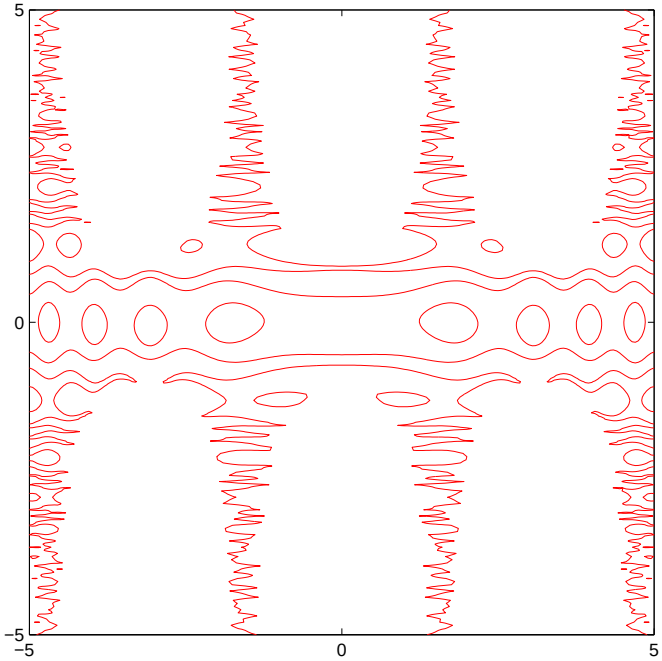
Example

Draw the level-set defined by

$$f(x, y) = \sin\left(\cos x^2 + 10 \sin y^2\right) - y \cos x = 0;$$

restricted to the domain $[-5, 5] \times [-5, 5]$.

MATLAB produces the following picture:

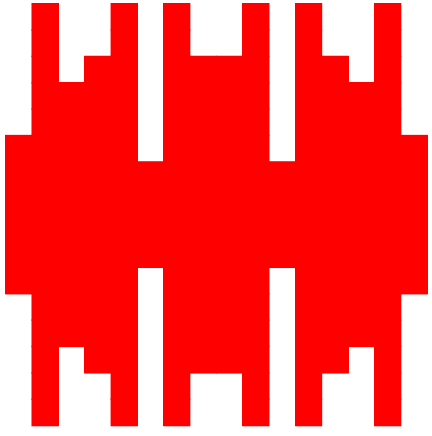
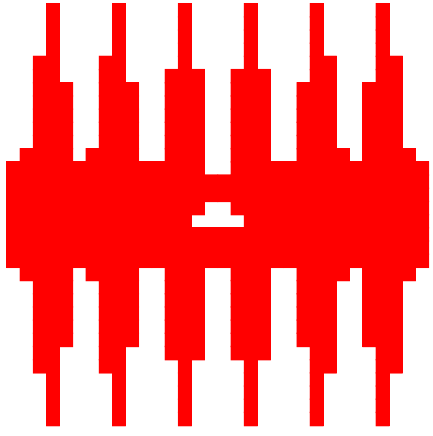


According to the same m-file, the level set defined by $|f(x, y)| = 0$, however, appears to be empty.

But this is the same set!!!

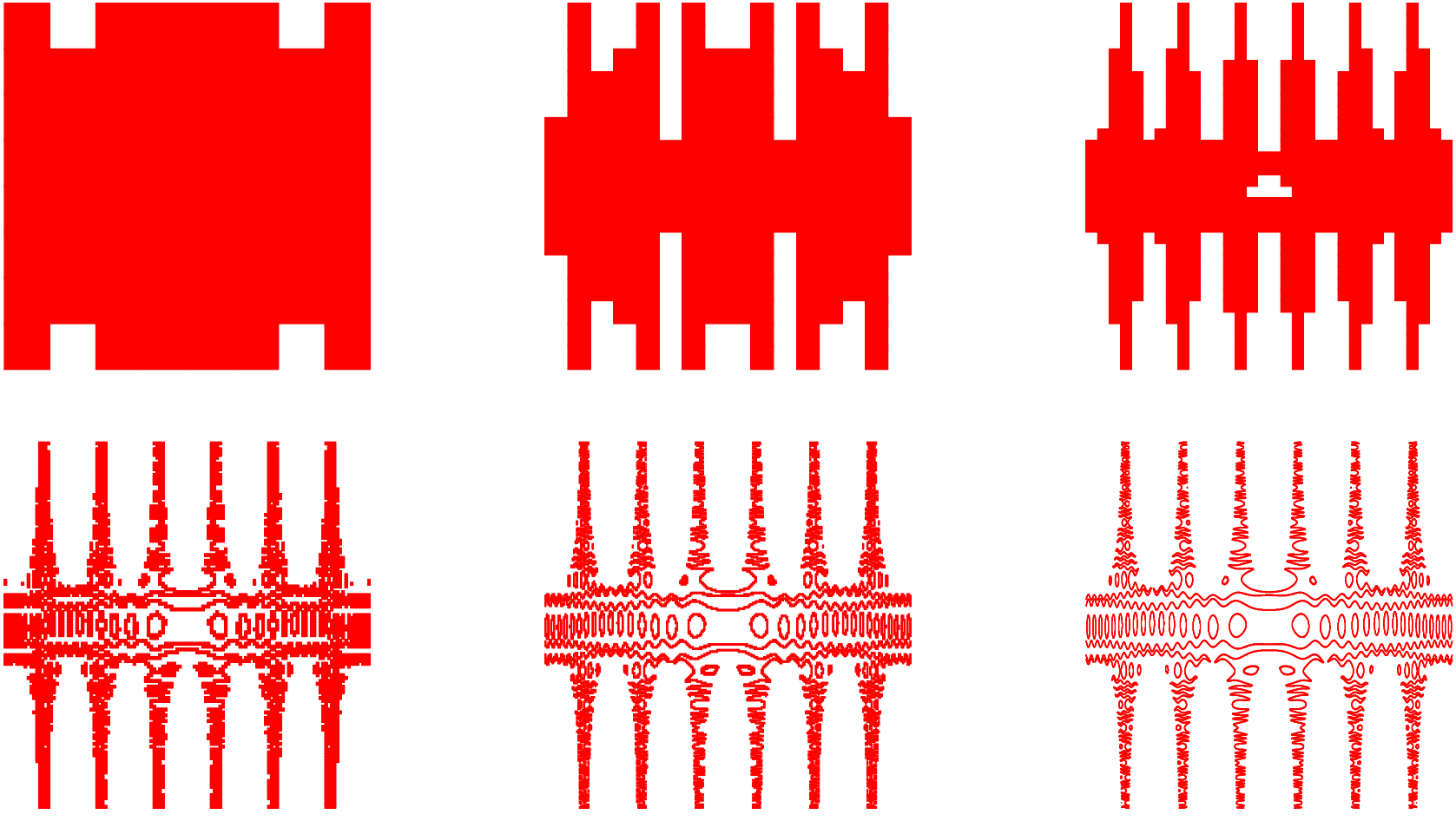
Implicit curves

The (increasingly tight) set-valued enclosures in both cases are

Implicit curves

The (increasingly tight) set-valued enclosures in both cases are



UPPSALA
UNIVERSITET

Set inversion

Given a search space $\mathbb{X}$, we can produce both inner and outer enclosures of $f^{-1}(\mathbb{Y})$: given a discretisation $\mathbb{X} = \cup_{i=1}^{N} \mathbb{X}_i$, we have

$$\{\mathbb{X}_i : F(\mathbb{X}_i) \subseteq \mathbb{Y}\} \subseteq f^{-1}(\mathbb{Y}) \cap \mathbb{X} \subseteq \{\mathbb{X}_i : F(\mathbb{X}_i) \cap \mathbb{Y} \neq \emptyset\}.$$

UPPSALA
UNIVERSITET

Set inversion

Given a search space $\mathbb{X}$, we can produce both inner and outer enclosures of $f^{-1}(\mathbb{Y})$: given a discretisation $\mathbb{X} = \cup_{i=1}^{N} \mathbb{X}_i$, we have

$$\{\mathbb{X}_i : F(\mathbb{X}_i) \subseteq \mathbb{Y}\} \subseteq f^{-1}(\mathbb{Y}) \cap \mathbb{X} \subseteq \{\mathbb{X}_i : F(\mathbb{X}_i) \cap \mathbb{Y} \neq \emptyset\}.$$

Example

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by $f(x_1, x_2) = x_1^4 - x_1^2 + 4x_2^2$, and compute the inverse image of $\mathbb{Y} = [-0.05, 0.05]$ over the domain $\mathbb{X} = [-5, 5] \times [-5, 5]$. Let us write $S = f^{-1}(\mathbb{Y}) \cap \mathbb{X}$.

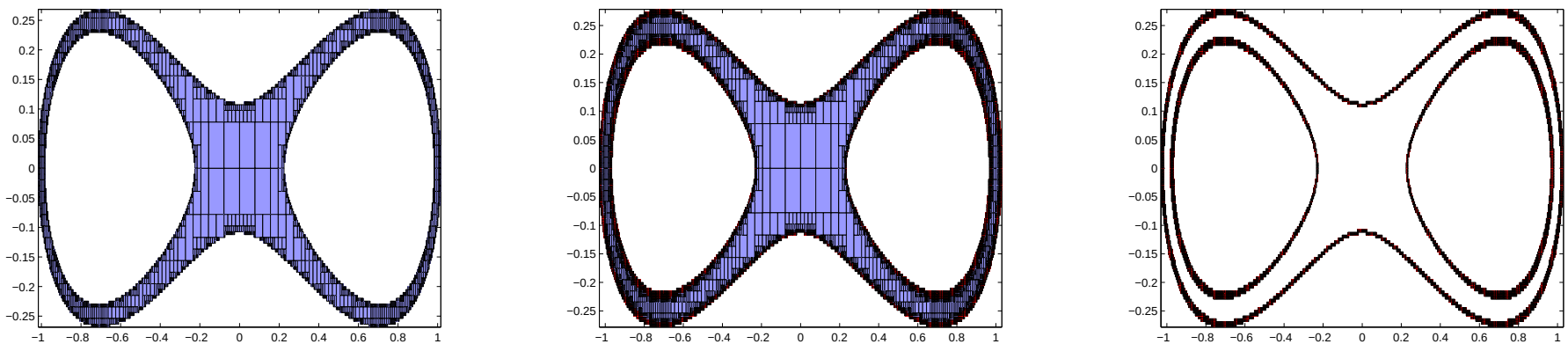
Set inversion

Given a search space $\mathbb{X}$, we can produce both inner and outer enclosures of $f^{-1}(\mathbb{Y})$: given a discretisation $\mathbb{X} = \cup_{i=1}^{N} \mathbb{X}_i$, we have

$$\{\mathbb{X}_i : F(\mathbb{X}_i) \subseteq \mathbb{Y}\} \subseteq f^{-1}(\mathbb{Y}) \cap \mathbb{X} \subseteq \{\mathbb{X}_i : F(\mathbb{X}_i) \cap \mathbb{Y} \neq \emptyset\}.$$

Example

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by $f(x_1, x_2) = x_1^4 - x_1^2 + 4x_2^2$, and compute the inverse image of $\mathbb{Y} = [-0.05, 0.05]$ over the domain $\mathbb{X} = [-5, 5] \times [-5, 5]$. Let us write $S = f^{-1}(\mathbb{Y}) \cap \mathbb{X}$.



UPPSALA UNIVERSITET

Figure: The (a) inner, (b) outer, and (c) boundary enclosures of S.

Parameter reconstruction

Question

Given a model function $y = f(x; p)$ together with (noisy) samples $\{(x_i, y_i + \varepsilon_i)\}_{i=0}^{N}$, is it possible to find a "best" parameter $p^\star$?

Parameter reconstruction

Question

Given a model function $y = f(x; p)$ together with (noisy) samples $\{(x_i, y_i + \varepsilon_i)\}_{i=0}^{N}$, is it possible to find a "best" parameter $p^\star$?

View the data set as being interval-valued: $\{(x_i, \mathbb{Y}_i)\}_{i=0}^{N}$.

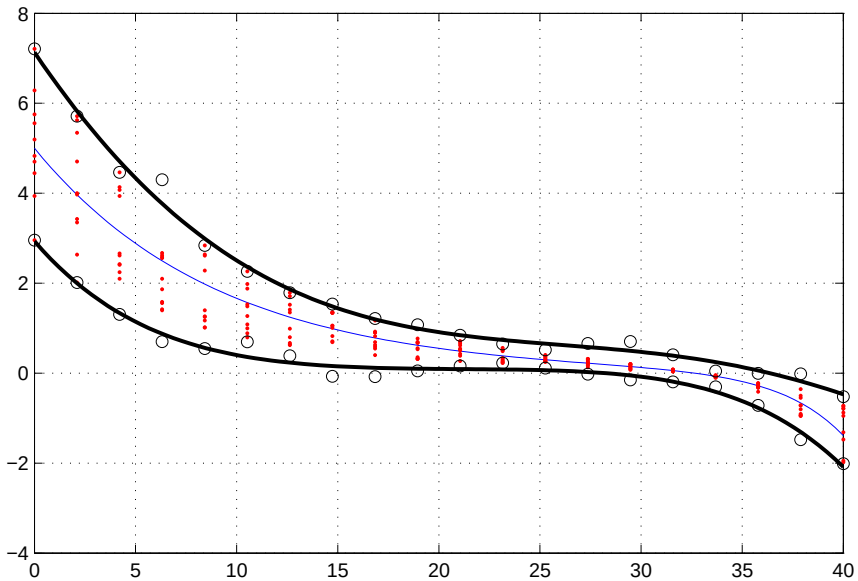
Parameter reconstruction

Question

Given a model function $y = f(x; p)$ together with (noisy) samples $\{(x_i, y_i + \varepsilon_i)\}_{i=0}^{N}$, is it possible to find a "best" parameter $p^\star$?

View the data set as being interval-valued: $\{(x_i, \mathbb{Y}_i)\}_{i=0}^{N}$.

Population: 10
Data points: 20
Noise type: normal
Noise level: 50%



UPPSALA UNIVERSITET

Parameter reconstruction

Strategy

Adaptively bisect the parameter space into sub-boxes: $\mathbb{P} = \cup_{j=1}^{M} \mathbb{Q}_j$ and examine each $\mathbb{Q}_j$ separately.

UPPSALA
UNIVERSITET

Parameter reconstruction

Strategy

Adaptively bisect the parameter space into sub-boxes: $\mathbb{P} = \cup_{j=1}^{M} \mathbb{Q}_j$ and examine each $\mathbb{Q}_j$ separately.

We will associate each sub-box $\mathbb{Q}_j$ of the parameter space to one of three categories:

Parameter reconstruction

Strategy

Adaptively bisect the parameter space into sub-boxes: $\mathbb{P} = \cup_{j=1}^{M} \mathbb{Q}_j$ and examine each $\mathbb{Q}_j$ separately.

We will associate each sub-box $\mathbb{Q}_j$ of the parameter space to one of three categories:

(1) consistent

if $F(x_i; \mathbb{Q}_j) \subset \mathbb{Y}_i$ for all $i = 0, \ldots, N$. SAVE

Parameter reconstruction

Strategy

Adaptively bisect the parameter space into sub-boxes: $\mathbb{P} = \cup_{j=1}^{M} \mathbb{Q}_j$ and examine each $\mathbb{Q}_j$ separately.

We will associate each sub-box $\mathbb{Q}_j$ of the parameter space to one of three categories:

(1) consistent

if $F(x_i; \mathbb{Q}_j) \subset \mathbb{Y}_i$ for all $i = 0, \ldots, N$. SAVE

(2) inconsistent

if $F(x_i; \mathbb{Q}_j) \cap \mathbb{Y}_i = \emptyset$ for at least one i. DELETE

UPPSALA
UNIVERSITET

Parameter reconstruction

Strategy

Adaptively bisect the parameter space into sub-boxes: $\mathbb{P} = \cup_{j=1}^{M} \mathbb{Q}_j$ and examine each $\mathbb{Q}_j$ separately.

We will associate each sub-box $\mathbb{Q}_j$ of the parameter space to one of three categories:

(1) consistent

if $F(x_i; \mathbb{Q}_j) \subset \mathbb{Y}_i$ for all $i = 0, \ldots, N$. SAVE

(2) inconsistent

if $F(x_i; \mathbb{Q}_j) \cap \mathbb{Y}_i = \emptyset$ for at least one i. DELETE

(3) undetermined

not (1), but $F(x_i; \mathbb{Q}_j) \cap \mathbb{Y}_i \neq \emptyset$ for all $i = 0, \ldots, N$. SPLIT

Parameter reconstruction

Example

Consider the model function

$$f(x; p_1, p_2) = 5e^{-p_1 x} - 4 \times 10^{-6} e^{-p_2 x}$$

with samples taken at $x = 0, 5 \ldots, 40$ using $p^\star = (0.11, -0.32)$. With a relative noise level of 90%, we get the following set of consistent parameters:

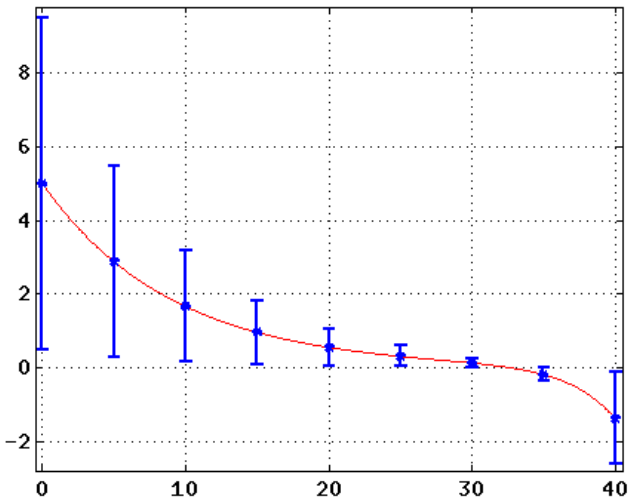
Parameter reconstruction

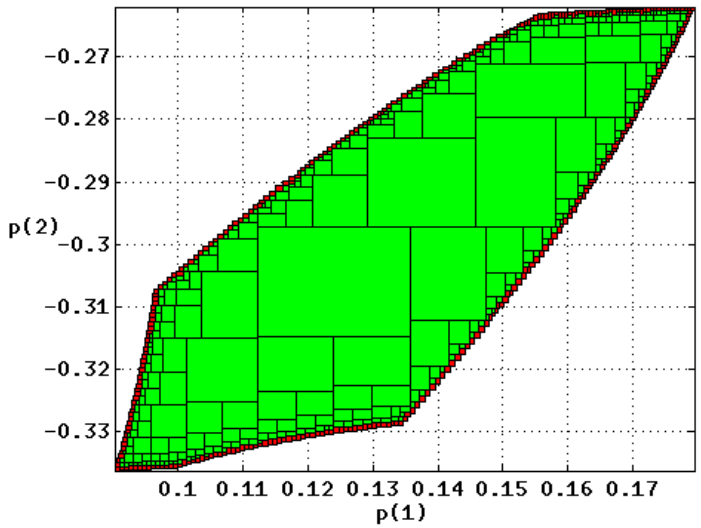
Example

Consider the model function

$$f(x; p_1, p_2) = 5e^{-p_1 x} - 4 \times 10^{-6} e^{-p_2 x}$$

with samples taken at $x = 0, 5 \ldots, 40$ using $p^\star = (0.11, -0.32)$. With a relative noise level of 90%, we get the following set of consistent parameters:

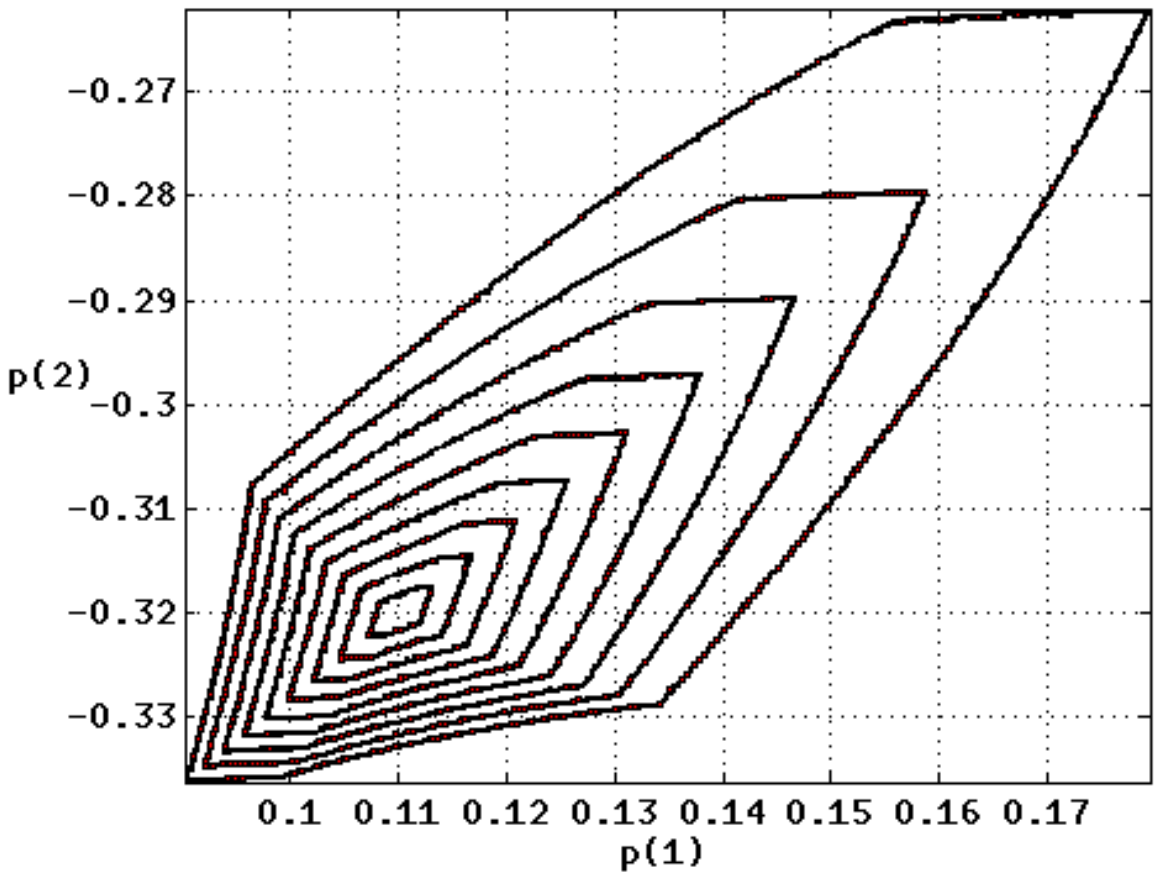




UPPSALA UNIVERSITET

Parameter reconstruction

Varying the relative noise levels between $10, 20 \ldots, 90\%$, we get the following indeterminate sets.



When do we use derivatives?

When do we use derivatives?

- (1:st order) Newton's method, monotonicity, implicit function theorem, stability.

When do we use derivatives?

- (1:st order) Newton's method, monotonicity, implicit function theorem, stability.
- (2:nd order) Optimization: convexity.

When do we use derivatives?

- (1:st order) Newton's method, monotonicity, implicit function theorem, stability.
- (2:nd order) Optimization: convexity.
- (n:th order) High-order approximations, quadrature, differential equations.

When do we use derivatives?

- (1:st order) Newton's method, monotonicity, implicit function theorem, stability.
- (2:nd order) Optimization: convexity.
- (n:th order) High-order approximations, quadrature, differential equations.

Example (A simple calculus task)

What is the value of $f^{(n)}(x_0)$, where

$$f(x) = e^{\sin e^{\cos x + 2x^5}}$$

When do we use derivatives?

- (1:st order) Newton's method, monotonicity, implicit function theorem, stability.
- (2:nd order) Optimization: convexity.
- (n:th order) High-order approximations, quadrature, differential equations.

Example (A simple calculus task)

What is the value of $f^{(n)}(x_0)$, where

$$f(x) = e^{\sin e^{\cos x + 2x^5}}$$

for $x_0 = +1$ and $n = 1$? [Undergraduate maths - but tedious]

When do we use derivatives?

- (1:st order) Newton's method, monotonicity, implicit function theorem, stability.
- (2:nd order) Optimization: convexity.
- (n:th order) High-order approximations, quadrature, differential equations.

Example (A simple calculus task)

What is the value of $f^{(n)}(x_0)$, where

$$f(x) = e^{\sin e^{\cos x + 2x^5}}$$

for $x_0 = +1$ and $n = 1$? [Undergraduate maths - but tedious]
for $x_0 = -2$ and $n = 40$? [Undergraduate maths - impossible?]

How do we compute derivatives in practice?

How do we compute derivatives in practice?

- (Symbolic representation) Generates exact formulas for $f, f', \ldots, f^{(n)}, \ldots$. This is very memory/time consuming. Produces enormous formulas. Actually too much information.

How do we compute derivatives in practice?

- (Symbolic representation) Generates exact formulas for $f, f', \ldots, f^{(n)}, \ldots$. This is very memory/time consuming. Produces enormous formulas. Actually too much information.
- (Finite differences) Generates numerical approximations of the value of a derivative, e.g. $f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}$, based on

$$f(x_0 + h) = f(x_0) + hf'(x_0) + h^2 f''(x_0) + \mathcal{O}(h^3).$$

Various errors: roundoff, cancellation, discretization. Which h is optimal? How does the error behave? Can't really handle high-order derivatives.

How do we compute derivatives in practice?

- (Symbolic representation) Generates exact formulas for $f, f', \ldots, f^{(n)}, \ldots$. This is very memory/time consuming. Produces enormous formulas. Actually too much information.
- (Finite differences) Generates numerical approximations of the value of a derivative, e.g. $f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}$, based on

$$f(x_0 + h) = f(x_0) + hf'(x_0) + h^2 f''(x_0) + \mathcal{O}(h^3).$$

  Various errors: roundoff, cancellation, discretization. Which h is optimal? How does the error behave? Can't really handle high-order derivatives.
- (Complex differentiation) A nice "trick" using complex extensions: $f'(x_0) \approx \frac{\Im(f(x_0+ih))}{h}$, where $\Im(x + iy) = y$. Avoids cancellation, and gives quadratic approximation, but requires a complex extension of the function.

Automatic differentiation

Generates evaluations (and not formulas) of the derivatives. Based on a strategy similar to symbolic differentiation, but does not use placeholders for constants or variables. All intermediate expressions are evaluated as soon as possible; this saves memory, and removes the need for later simplification.

Automatic differentiation

Generates evaluations (and not formulas) of the derivatives. Based on a strategy similar to symbolic differentiation, but does not use placeholders for constants or variables. All intermediate expressions are evaluated as soon as possible; this saves memory, and removes the need for later simplification.

Bonus properties

Automatic differentiation

Generates evaluations (and not formulas) of the derivatives. Based on a strategy similar to symbolic differentiation, but does not use placeholders for constants or variables. All intermediate expressions are evaluated as soon as possible; this saves memory, and removes the need for later simplification.

Bonus properties

- No discretization errors.

Automatic differentiation

Generates evaluations (and not formulas) of the derivatives. Based on a strategy similar to symbolic differentiation, but does not use placeholders for constants or variables. All intermediate expressions are evaluated as soon as possible; this saves memory, and removes the need for later simplification.

Bonus properties

- No discretization errors.
- No huge memory consumption.

Automatic differentiation

Generates evaluations (and not formulas) of the derivatives. Based on a strategy similar to symbolic differentiation, but does not use placeholders for constants or variables. All intermediate expressions are evaluated as soon as possible; this saves memory, and removes the need for later simplification.

Bonus properties

- No discretization errors.
- No huge memory consumption.
- No complex “tricks“.

Automatic differentiation

Generates evaluations (and not formulas) of the derivatives. Based on a strategy similar to symbolic differentiation, but does not use placeholders for constants or variables. All intermediate expressions are evaluated as soon as possible; this saves memory, and removes the need for later simplification.

Bonus properties

- No discretization errors.
- No huge memory consumption.
- No complex “tricks“.
- Very easy to understand.

Taylor series AD - definitions

An effective approach to high-order automatic differentiation is through the calculus of Taylor series:

$$f(x) = f_0 + f_1(x - x_0) + \cdots + f_k(x - x_0)^k + \ldots,$$

Here we use the notation $f_k = f_k(x_0) = f^{(k)}(x_0)/k!$

Taylor series AD - definitions

An effective approach to high-order automatic differentiation is through the calculus of Taylor series:

$$f(x) = f_0 + f_1(x - x_0) + \cdots + f_k(x - x_0)^k + \ldots,$$

Here we use the notation $f_k = f_k(x_0) = f^{(k)}(x_0)/k!$

Basic arithmetic

$$\begin{aligned}
(f + g)_k &= f_k + g_k \\
(f - g)_k &= f_k - g_k \\
(f \times g)_k &= \sum_{i=0}^{k} f_i g_{k-i} \\
(f \div g)_k &= \frac{1}{g_0}\left(f_k - \sum_{i=0}^{k-1} (f \div g)_i g_{k-i}\right).
\end{aligned}$$

Proof: (formula for division).

By definition, we have

$$\sum_{k=0}^{\infty} f_k(x-x_0)^k / \sum_{k=0}^{\infty} g_k(x-x_0)^k = \sum_{k=0}^{\infty} (f \div g)_k (x-x_0)^k.$$

Proof: (formula for division).

By definition, we have

$$\sum_{k=0}^{\infty} f_k(x-x_0)^k / \sum_{k=0}^{\infty} g_k(x-x_0)^k = \sum_{k=0}^{\infty} (f \div g)_k (x-x_0)^k.$$

Multiplying both sides with the Taylor series for g produces

$$\sum_{k=0}^{\infty} f_k(x-x_0)^k = \sum_{k=0}^{\infty} (f \div g)_k (x-x_0)^k \sum_{k=0}^{\infty} g_k(x-x_0)^k,$$

Proof: (formula for division).

By definition, we have

$$\sum_{k=0}^{\infty} f_k(x-x_0)^k / \sum_{k=0}^{\infty} g_k(x-x_0)^k = \sum_{k=0}^{\infty} (f \div g)_k (x-x_0)^k.$$

Multiplying both sides with the Taylor series for g produces

$$\sum_{k=0}^{\infty} f_k(x-x_0)^k = \sum_{k=0}^{\infty} (f \div g)_k (x-x_0)^k \sum_{k=0}^{\infty} g_k (x-x_0)^k,$$

and, by the rule for multiplication, we have

$$f_k = \sum_{i=0}^{k} (f \div g)_i g_{k-i} = \sum_{i=0}^{k-1} (f \div g)_i g_{k-i} + (f \div g)_k g_0.$$

Proof: (formula for division).

By definition, we have

$$\sum_{k=0}^{\infty} f_k(x-x_0)^k / \sum_{k=0}^{\infty} g_k(x-x_0)^k = \sum_{k=0}^{\infty} (f \div g)_k (x-x_0)^k.$$

Multiplying both sides with the Taylor series for g produces

$$\sum_{k=0}^{\infty} f_k(x-x_0)^k = \sum_{k=0}^{\infty} (f \div g)_k (x-x_0)^k \sum_{k=0}^{\infty} g_k(x-x_0)^k,$$

and, by the rule for multiplication, we have

$$f_k = \sum_{i=0}^{k} (f \div g)_i g_{k-i} = \sum_{i=0}^{k-1} (f \div g)_i g_{k-i} + (f \div g)_k g_0.$$

Solving for $(f \div g)_k$ produces the desired result. □

Taylor series AD - definitions

Constants and the independent variable x are treated as expected: seen as functions, these have particularly simple Taylor expansions:

$$\begin{aligned} x &= x_0 + 1 \cdot (x - x_0) + 0 \cdot (x - x_0)^2 + \cdots + 0 \cdot (x - x_0)^k + \ldots, \\ c &= c + 0 \cdot (x - x_0) + 0 \cdot (x - x_0)^2 + \cdots + 0 \cdot (x - x_0)^k + \ldots. \end{aligned}$$

UPPSALA
UNIVERSITET

Taylor series AD - definitions

Constants and the independent variable x are treated as expected: seen as functions, these have particularly simple Taylor expansions:

$$\begin{aligned} x &= x_0 + 1 \cdot (x - x_0) + 0 \cdot (x - x_0)^2 + \cdots + 0 \cdot (x - x_0)^k + \ldots, \\ c &= c + 0 \cdot (x - x_0) + 0 \cdot (x - x_0)^2 + \cdots + 0 \cdot (x - x_0)^k + \ldots. \end{aligned}$$

We now represent a function as a, possibly infinite, string of its Taylor coefficients:

$$f(x_0) \sim (f_0, f_1, \ldots, f_k, \ldots) \qquad f_k = f^{(k)}(x_0)/k!$$

Taylor series AD - definitions

Constants and the independent variable x are treated as expected: seen as functions, these have particularly simple Taylor expansions:

$$\begin{aligned} x &= x_0 + 1 \cdot (x - x_0) + 0 \cdot (x - x_0)^2 + \cdots + 0 \cdot (x - x_0)^k + \ldots, \\ c &= c + 0 \cdot (x - x_0) + 0 \cdot (x - x_0)^2 + \cdots + 0 \cdot (x - x_0)^k + \ldots. \end{aligned}$$

We now represent a function as a, possibly infinite, string of its Taylor coefficients:

$$f(x_0) \sim (f_0, f_1, \ldots, f_k, \ldots) \qquad f_k = f^{(k)}(x_0)/k!$$

The base field

Of course, we can perform the underlying *scalar* computations with anything else that is suitable. In our applications, we always use interval arithmetic with directed rounding.

Taylor series AD - definitions

Taylor series AD for standard functions

Given a function g whose Taylor series is known, how do we compute the Taylor series for, say, e^g?

Taylor series AD - definitions

Taylor series AD for standard functions

Given a function g whose Taylor series is known, how do we compute the Taylor series for, say, e^g?

Let us formally write

$$g(x) = \sum_{k=0}^{\infty} g_k (x - x_0)^k \quad \text{and} \quad e^{g(x)} = \sum_{k=0}^{\infty} (e^g)_k (x - x_0)^k,$$

and use the fact that

$$\frac{d}{dx} e^{g(x)} = g'(x) e^{g(x)}. \tag{1}$$

Taylor series AD - definitions

Taylor series AD for standard functions

Given a function g whose Taylor series is known, how do we compute the Taylor series for, say, e^g?

Let us formally write

$$g(x) = \sum_{k=0}^{\infty} g_k (x - x_0)^k \quad \text{and} \quad e^{g(x)} = \sum_{k=0}^{\infty} (e^g)_k (x - x_0)^k,$$

and use the fact that

$$\frac{d}{dx} e^{g(x)} = g'(x) e^{g(x)}. \tag{1}$$

A formal computation gives

$$(e^g)_k = \begin{cases} e^{g_0} & \text{if } k = 0, \\ \frac{1}{k} \sum_{i=1}^{k} i g_i (e^g)_{k-i} & \text{if } k > 0. \end{cases}$$

More standard functions $(k > 0)$

$$
\begin{aligned}
(\ln g)_k &= \frac{1}{g_0}\left(g_k - \frac{1}{k}\sum_{i=1}^{k-1} i(\ln g)_i g_{k-i}\right) \\
(g^a)_k &= \frac{1}{g_0}\sum_{i=1}^{k}\left(\frac{(a+1)i}{k} - 1\right) g_i (g^a)_{k-i} \\
(\sin g)_k &= \frac{1}{k}\sum_{i=1}^{k} i g_i (\cos g)_{k-i} \\
(\cos g)_k &= -\frac{1}{k}\sum_{i=1}^{k} i g_i (\sin g)_{k-i}.
\end{aligned}
$$

Taylor series AD - definitions

More standard functions $(k > 0)$

$$
\begin{aligned}
(\ln g)_k &= \frac{1}{g_0}\left(g_k - \frac{1}{k}\sum_{i=1}^{k-1} i(\ln g)_i g_{k-i}\right) \\
(g^a)_k &= \frac{1}{g_0}\sum_{i=1}^{k}\left(\frac{(a+1)i}{k} - 1\right) g_i (g^a)_{k-i} \\
(\sin g)_k &= \frac{1}{k}\sum_{i=1}^{k} i g_i (\cos g)_{k-i} \\
(\cos g)_k &= -\frac{1}{k}\sum_{i=1}^{k} i g_i (\sin g)_{k-i}.
\end{aligned}
$$

Remember that we always have $(f \circ g)_0 = f(g(x_0))$.

Taylor series AD - implementations

A MATLAB implementation for arbitrary order differentiation:

```
01 function dx = computeDerivative(fcnName, x0, order)
02 f  = inline(fcnName);
03 x  = taylor(x0, order+1, 'variable');
04 dx = getDer(f(x), order);
```

Taylor series AD - implementations

A MATLAB implementation for arbitrary order differentiation:

```
01 function dx = computeDerivative(fcnName, x0, order)
02 f  = inline(fcnName);
03 x  = taylor(x0, order+1, 'variable');
04 dx = getDer(f(x), order);
```

Here, getDer converts a Taylor coefficient into a derivative value by multiplying it by the proper factorial.

Taylor series AD - implementations

A MATLAB implementation for arbitrary order differentiation:

```
01 function dx = computeDerivative(fcnName, x0, order)
02 f  = inline(fcnName);
03 x  = taylor(x0, order+1, 'variable');
04 dx = getDer(f(x), order);
```

Here, getDer converts a Taylor coefficient into a derivative value by multiplying it by the proper factorial.

A typical usage is

```
>> dfx = computeDerivative('exp(sin(exp(cos(x) + 2*x^5)))', 1, 1)
dfx =
    129.6681309181679
```

Taylor series AD - implementations

A MATLAB implementation for arbitrary order differentiation:

```
01 function dx = computeDerivative(fcnName, x0, order)
02 f  = inline(fcnName);
03 x  = taylor(x0, order+1, 'variable');
04 dx = getDer(f(x), order);
```

Here, getDer converts a Taylor coefficient into a derivative value by multiplying it by the proper factorial.

A typical usage is

```
>> dfx = computeDerivative('exp(sin(exp(cos(x) + 2*x^5)))', 1, 1)
dfx =
    129.6681309181679
```

Let us solve the harder part of our homework...

```
>> df40 = computeDerivative('exp(sin(exp(cos(x) + 2*x^5)))', -2, 40)
df40 =
        1.4961e+53
```

Double–checking using interval analysis gives the enclosure:

```
df40 inside 1.49[5513361918031623,6705656849363108 7]E+53
```

Taylor series AD - implementations

A more practical application is solving non-linear equations.

```
01 function y = newtonSearch(fcnName, x, tol)
02 f = inline(fcnName);
03 y = newtonStep(f, x);
04 while ( abs(x-y) > tol )
05     x = y;
06     y = newtonStep(f, x);
07 end
08 end
09
10 function Nx = newtonStep(f, x)
11 xx = taylor(x, 2, 'variable');
12 fx = f(xx);
13 Nx = x - getVal(fx)/getDer(fx, 1);
14 end
```

UPPSALA
UNIVERSITET

Taylor series AD - implementations

A more practical application is solving non-linear equations.

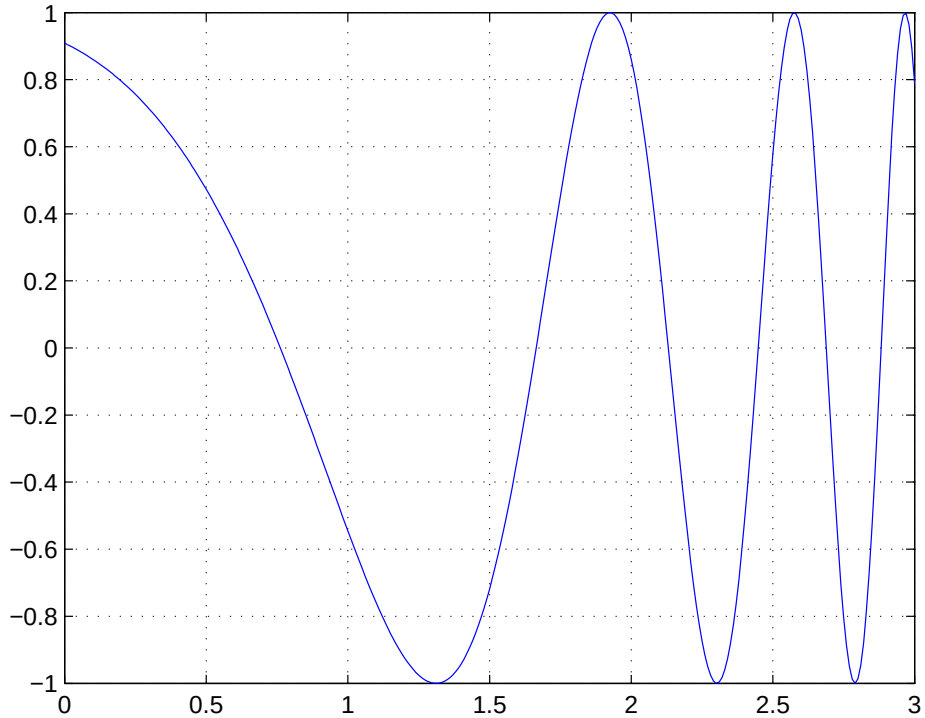
```
01 function y = newtonSearch(fcnName, x, tol)
02 f = inline(fcnName);
03 y = newtonStep(f, x);
04 while ( abs(x-y) > tol )
05     x = y;
06     y = newtonStep(f, x);
07 end
08 end
09
10 function Nx = newtonStep(f, x)
11 xx = taylor(x, 2, 'variable');
12 fx = f(xx);
13 Nx = x - getVal(fx)/getDer(fx, 1);
14 end
```

Note that this function "hides" the AD from the user: all input/output is scalar.

Taylor series AD - implementations

Some sample outputs:

```
>> x = newtonSearch('sin(exp(x) + 1)', 1, 1e-10)
x =
  0.761549782880894
>> x = newtonSearch('sin(exp(x) + 1)', 0, 1e-10)
x =
  2.131177121086310
```



Quadrature

Task:

Compute (approximate/enclose) the definite integral

$$I = \int_a^b f(x)dx.$$

Quadrature

Task:

Compute (approximate/enclose) the definite integral

$$I = \int_a^b f(x)dx.$$

We assume that I is well-defined, and that f is nice...

Quadrature

Task:

Compute (approximate/enclose) the definite integral

$$I = \int_a^b f(x)dx.$$

We assume that I is well-defined, and that f is nice...

Naive approach:

Split the domain of integration into N equally wide subintervals: we set $h = (b-a)/N$ and $x_i = a + ih$, $i = 0, \ldots, N$, and enclose the integrand via IA.

Quadrature

Task:

Compute (approximate/enclose) the definite integral

$$I = \int_a^b f(x)dx.$$

We assume that I is well-defined, and that f is nice...

Naive approach:

Split the domain of integration into N equally wide subintervals: we set $h = (b-a)/N$ and $x_i = a + ih$, $i = 0, \ldots, N$, and enclose the integrand via IA.

This produces the enclosure

$$\int_a^b f(x)dx \in I_a^b(f, N) \stackrel{\text{def}}{=} h \sum_{i=1}^{N} F([x_{i-1}, x_i]),$$

which satisfies $w(I_a^b(f, N)) = \mathcal{O}(1/N)$.

UPPSALA UNIVERSITET

Quadrature

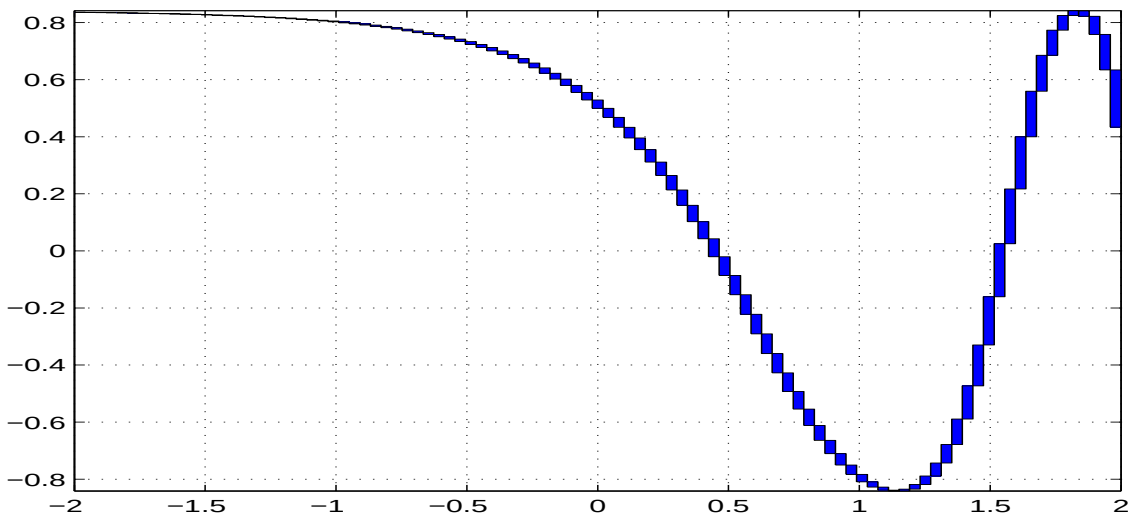
Example

Enclose the definite integral $\int_{-2}^{2} \sin\left(\cos\left(e^{x}\right)\right)dx$.

Example

Enclose the definite integral $\int_{-2}^{2} \sin\left(\cos\left(e^x\right)\right)dx$.

N	$I^2_{-2}(f,N)$	$w(I^2_{-2}(f,N))$
10^0	$[-3.36588, 3.36588]$	$6.73177 \cdot 10^0$
10^2	$[1.26250, 1.41323]$	$1.50729 \cdot 10^{-1}$
10^4	$[1.33791, 1.33942]$	$1.50756 \cdot 10^{-3}$
10^6	$[1.33866, 1.33868]$	$1.50758 \cdot 10^{-5}$



Quadrature

Taylor series approach:

Generate tighter bounds on the integrand via AD.

Quadrature

Taylor series approach:

Generate tighter bounds on the integrand via AD.

For all $x \in \mathbb{X}$, we have

$$\begin{aligned} f(x) &= \sum_{k=0}^{n-1} f_k(\check{x})(x-\check{x})^k + f_n(\zeta_x)(x-\check{x})^n \\ &\in \sum_{k=0}^{n} f_k(\check{x})(x-\check{x})^k + [-\varepsilon_n, \varepsilon_n]|x-\check{x}|^n, \end{aligned}$$

where $\varepsilon_n = \operatorname{mag}\big(F_n(\mathbb{X}) - f_n(\check{x})\big)$.

Quadrature

Taylor series approach:

Generate tighter bounds on the integrand via AD.

For all $x \in \mathbb{X}$, we have

$$\begin{aligned} f(x) &= \sum_{k=0}^{n-1} f_k(\check{x})(x-\check{x})^k + f_n(\zeta_x)(x-\check{x})^n \\ &\in \sum_{k=0}^{n} f_k(\check{x})(x-\check{x})^k + [-\varepsilon_n, \varepsilon_n]|x-\check{x}|^n, \end{aligned}$$

where $\varepsilon_n = \mathrm{mag}\big(F_n(\mathbb{X}) - f_n(\check{x})\big)$.
We are now prepared to compute the integral itself:

$$\begin{aligned} \int_{\check{x}-r}^{\check{x}+r} f(x)dx &\in \int_{\check{x}-r}^{\check{x}+r} \left(\sum_{k=0}^{n} f_k(\check{x})(x-\check{x})^k + [-\varepsilon_n, \varepsilon_n]|x-\check{x}|^n \right) dx \\ &= \sum_{k=0}^{n} f_k(\check{x}) \int_{-r}^{r} x^k dx + [-\varepsilon_n, \varepsilon_n] \int_{-r}^{r} |x|^n dx. \end{aligned}$$

Quadrature

Continuing the calculation, we see a lot of cancellation:

$$\begin{aligned}
\int_{\check{x}-r}^{\check{x}+r} f(x)dx &\in \sum_{k=0}^{n} f_k(\check{x}) \int_{-r}^{r} x^k dx + [-\varepsilon_n, \varepsilon_n] \int_{-r}^{r} |x|^n dx \\
&= \sum_{k=0}^{\lfloor n/2 \rfloor} f_{2k}(\check{x}) \int_{-r}^{r} x^{2k} dx + [-\varepsilon_n, \varepsilon_n] \int_{-r}^{r} |x|^n dx \\
&= 2 \left(\sum_{k=0}^{\lfloor n/2 \rfloor} f_{2k}(\check{x}) \frac{r^{2k+1}}{2k+1} + [-\varepsilon_n, \varepsilon_n] \frac{r^{n+1}}{n+1} \right) .
\end{aligned}$$

UPPSALA
UNIVERSITET

Quadrature

Continuing the calculation, we see a lot of cancellation:

$$\begin{aligned}\int_{\check{x}-r}^{\check{x}+r} f(x)dx &\in \sum_{k=0}^{n} f_k(\check{x}) \int_{-r}^{r} x^k dx + [-\varepsilon_n, \varepsilon_n] \int_{-r}^{r} |x|^n dx \\ &= \sum_{k=0}^{\lfloor n/2 \rfloor} f_{2k}(\check{x}) \int_{-r}^{r} x^{2k} dx + [-\varepsilon_n, \varepsilon_n] \int_{-r}^{r} |x|^n dx \\ &= 2\left(\sum_{k=0}^{\lfloor n/2 \rfloor} f_{2k}(\check{x}) \frac{r^{2k+1}}{2k+1} + [-\varepsilon_n, \varepsilon_n] \frac{r^{n+1}}{n+1} \right).\end{aligned}$$

Partition the domain of integration: $a = x_0 < x_1 < \cdots < x_N = b$.

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} f(x)dx = \sum_{i=1}^{N} \int_{\check{x}_i - r_i}^{\check{x}_i + r_i} f(x)dx \\ &\in 2\sum_{i=1}^{N} \left(\sum_{k=0}^{\lfloor n/2 \rfloor} f_{2k}(\check{x}_i) \frac{r_i^{2k+1}}{2k+1} + [-\varepsilon_{n,i}, \varepsilon_{n,i}] \frac{r_i^{n+1}}{n+1} \right).\end{aligned}$$

UPPSALA UNIVERSITET

Quadrature

Uniform partition:

N	$E^2_{-2}(f,6,N)$	$w(E^2_{-2}(f,6,N))$
9	$[0.86325178469, 1.81128961988]$	$9.4804 \cdot 10^{-1}$
12	$[1.28416304745, 1.39316025451]$	$1.0900 \cdot 10^{-1}$
21	$[1.33783795371, 1.33950680633]$	$1.6689 \cdot 10^{-3}$
75	$[1.33866863493, 1.33866878008]$	$1.4514 \cdot 10^{-7}$

UPPSALA
UNIVERSITET

Quadrature

Uniform partition:

N	$E^2_{-2}(f,6,N)$	$w(E^2_{-2}(f,6,N))$
9	$[0.86325178469, 1.81128961988]$	$9.4804 \cdot 10^{-1}$
12	$[1.28416304745, 1.39316025451]$	$1.0900 \cdot 10^{-1}$
21	$[1.33783795371, 1.33950680633]$	$1.6689 \cdot 10^{-3}$
75	$[1.33866863493, 1.33866878008]$	$1.4514 \cdot 10^{-7}$

Adaptive partition:

$\texttt{TOL}$	$A^2_{-2}(f,6,\texttt{TOL})$	$w(A^2_{-2}(f,6,\texttt{TOL}))$	$N_{\texttt{TOL}}$
10^{-1}	$[1.33229594606, 1.34500942603]$	$1.2713 \cdot 10^{-2}$	9
10^{-2}	$[1.33822575109, 1.33911045235]$	$8.8470 \cdot 10^{-4}$	12
10^{-4}	$[1.33866170207, 1.33867571626]$	$1.4014 \cdot 10^{-5}$	21
10^{-8}	$[1.33866870618, 1.33866870862]$	$2.4304 \cdot 10^{-9}$	75

UPPSALA
UNIVERSITET

Example (A bonus problem)

Compute the integral $\int_0^8 \sin\left(x + e^x\right) dx$.

Quadrature

Example (A bonus problem)

Compute the integral $\int_0^8 \sin\left(x + e^x\right) dx$.

A regular MATLAB session:

```
>> q = quad('sin(x + exp(x))', 0, 8)
q =
    0.251102722027180
```

UPPSALA UNIVERSITET

Quadrature

Example (A bonus problem)

Compute the integral $\int_0^8 \sin\left(x + e^x\right) dx$.

A regular MATLAB session:

```
>> q = quad('sin(x + exp(x))', 0, 8)
q =
    0.251102722027180
```

Using the adaptive validated integrator:

```
$$ ./adQuad 0 8 4 1e-4
Partitions: 8542
CPU time  : 0.52 seconds
Integral  : 0.347[3863144222905,4140198005782]
```

Quadrature

Example (A bonus problem)

Compute the integral $\int_0^8 \sin\left(x + e^x\right) dx$.

A regular MATLAB session:

```
>> q = quad('sin(x + exp(x))', 0, 8)
q =
    0.251102722027180
```

Using the adaptive validated integrator:

```
$$ ./adQuad 0 8 4 1e-4
Partitions: 8542
CPU time  : 0.52 seconds
Integral  : 0.347[3863144222905,4140198005782]

$$ ./adQuad 0 8 20 1e-10
Partitions: 874
CPU time  : 0.45 seconds
Integral  : 0.3474001726[492276,652638]
```

UPPSALA
UNIVERSITET

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers

Set-valued numerics

compute with sets

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers

produce an approximation

$x \approx 3.1415926535$

Set-valued numerics

compute with sets

produce an enclosure

$x \in [3.1415926535, 3.1415926536]$

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers

produce an approximation

$x \approx 3.1415926535$

discretisation/rounding errors?

Set-valued numerics

compute with sets

produce an enclosure

$x \in [3.1415926535, 3.1415926536]$

discretisation/rounding errors!

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers

produce an approximation

$x \approx 3.1415926535$

discretisation/rounding errors?

work required for error *estimates*

Set-valued numerics

compute with sets

produce an enclosure

$x \in [3.1415926535, 3.1415926536]$

discretisation/rounding errors!

rigorous error *bounds* for free

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers

produce an approximation

$x \approx 3.1415926535$

discretisation/rounding errors?

work required for error *estimates*

two results can be *compared*

Set-valued numerics

compute with sets

produce an enclosure

$x \in [3.1415926535, 3.1415926536]$

discretisation/rounding errors!

rigorous error *bounds* for free

two results can be *intersected*

UPPSALA
UNIVERSITET

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers

produce an approximation

$x \approx 3.1415926535$

discretisation/rounding errors?

work required for error *estimates*

two results can be *compared*

not suitable for proofs

Set-valued numerics

compute with sets

produce an enclosure

$x \in [3.1415926535, 3.1415926536]$

discretisation/rounding errors!

rigorous error *bounds* for free

two results can be *intersected*

can verify existence and uniqueness

UPPSALA
UNIVERSITET

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers

produce an approximation

$x \approx 3.1415926535$

discretisation/rounding errors?

work required for error *estimates*

two results can be *compared*

not suitable for proofs

can only examine finite sets

Set-valued numerics

compute with sets

produce an enclosure

$x \in [3.1415926535, 3.1415926536]$

discretisation/rounding errors!

rigorous error *bounds* for free

two results can be *intersected*

can verify existence and uniqueness

can examine subdomains

Classical numerics vs interval analysis based numerics

Classical numerics

compute with numbers
produce an approximation
$x \approx 3.1415926535$
discretisation/rounding errors?
work required for error *estimates*
two results can be *compared*
not suitable for proofs
can only examine finite sets
hardware support

Set-valued numerics

compute with sets
produce an enclosure
$x \in [3.1415926535, 3.1415926536]$
discretisation/rounding errors!
rigorous error *bounds* for free
two results can be *intersected*
can verify existence and uniqueness
can examine subdomains
coming soon?

Further study...

Interval Computations Web Page

`http://www.cs.utep.edu/interval-comp`

UPPSALA
UNIVERSITET

Further study...

Interval Computations Web Page

`http://www.cs.utep.edu/interval-comp`

INTLAB – INTerval LABoratory

`http://www.ti3.tu-harburg.de/~rump/intlab/`

UPPSALA
UNIVERSITET

Further study...

Interval Computations Web Page

http://www.cs.utep.edu/interval-comp

INTLAB – INTerval LABoratory

http://www.ti3.tu-harburg.de/~rump/intlab/

CXSC – C eXtensions for Scientific Computation

http://www.xsc.de/

UPPSALA
UNIVERSITET

Further study...

Interval Computations Web Page

http://www.cs.utep.edu/interval-comp

INTLAB – INTerval LABoratory

http://www.ti3.tu-harburg.de/~rump/intlab/

CXSC – C eXtensions for Scientific Computation

http://www.xsc.de/

CAPA – Computer–Aided Proofs in Analysis

http://www.math.uu.se/~warwick/CAPA/

UPPSALA
UNIVERSITET