24. Calculating with sets: Interval arithmetic

Summary of the previous class

- Chebyshev differentiation matrix
- Spectral methods for boundary-value problems
- Reduction to solving linear systems

Goals for today

- Calculating with sets: Interval arithmetic
- Intervals
- Extending functions to intervals
- Directed rounding
- Dependency problem
- Applications

Motivation

- How can we be *sure* that what we are computing is "correct"?

Motivation

- How can we be *sure* that what we are computing is "correct"?
- We introduce errors almost any time we perform a numerical calculation

Motivation

- How can we be *sure* that what we are computing is "correct"?
- We introduce errors almost any time we perform a numerical calculation
- We might always be left with a nagging feeling that we are not quite sure if something slipped through our fingers

Motivation

- How can we be *sure* that what we are computing is “correct”?
- We introduce errors almost any time we perform a numerical calculation
- We might always be left with a nagging feeling that we are not quite sure if something slipped through our fingers
- Mostly our calculations are fine: We can show that the results that we compute are “close to” the true result

Motivation

- How can we be *sure* that what we are computing is “correct”?
- We introduce errors almost any time we perform a numerical calculation
- We might always be left with a nagging feeling that we are not quite sure if something slipped through our fingers
- Mostly our calculations are fine: We can show that the results that we compute are “close to” the true result
- *How close*?

Motivation

- How can we be *sure* that what we are computing is "correct"?
- We introduce errors almost any time we perform a numerical calculation
- We might always be left with a nagging feeling that we are not quite sure if something slipped through our fingers
- Mostly our calculations are fine: We can show that the results that we compute are "close to" the true result
- *How close*?
- Can we get a guarantee of the form: Your result is definitely within this range?

Motivation I: Experimental error

- Suppose we measure a quantity x in an experiment
- If repeat experiment there is *variation* in outcome

Motivation I: Experimental error

- Suppose we measure a quantity x in an experiment
- If repeat experiment there is *variation* in outcome
- How can we model this **uncertainty**?

Motivation I: Experimental error

- Suppose we measure a quantity x in an experiment
- If repeat experiment there is *variation* in outcome
- How can we model this **uncertainty**?
- Maybe as a **probability distribution** of possible values

Motivation I: Experimental error

- Suppose we measure a quantity x in an experiment
- If repeat experiment there is *variation* in outcome
- How can we model this **uncertainty**?
- Maybe as a **probability distribution** of possible values
- Or **interval** of possible values
- If measurement is 1.35 and we think maximum error is 0.05 then $x \in 1.35 \pm 0.05$
- i.e. $x \in [1.3, 1.4]$

Motivation II: Bounding rounding errors

- We know that numerical calculations with floats have rounding errors

Motivation II: Bounding rounding errors

- We know that numerical calculations with floats have rounding errors
- Can we keep track of all possible errors?
- To obtain rigorous **bounds** on result of calculation?

Motivation II: Bounding rounding errors

- We know that numerical calculations with floats have rounding errors
- Can we keep track of all possible errors?
- To obtain rigorous **bounds** on result of calculation?
- Track bounds through calculation: at step i want

$$\ell_i \leq x_i \leq L_i$$

Motivation II: Bounding rounding errors

- We know that numerical calculations with floats have rounding errors
- Can we keep track of all possible errors?
- To obtain rigorous **bounds** on result of calculation?
- Track bounds through calculation: at step i want

$$\ell_i \leq x_i \leq L_i$$

- i.e. $x_i \in [\ell_i, L_i]$ – range (interval) of possible values of x_i

Motivation III: Another example

- Example by William Kahan: Consider

$$f(x) = \frac{1}{50} \log |3(1-x)+1| + x^2 + 1$$

- Looks uncomplicated if plot by sampling at many points

Motivation III: Another example

- Example by William Kahan: Consider

$$f(x) = \frac{1}{50} \log |3(1-x)+1| + x^2 + 1$$

- Looks uncomplicated if plot by sampling at many points
- Is it really that uncomplicated?
- What is happening near that dip?

Motivation III: Another example

- Example by William Kahan: Consider

$$f(x) = \frac{1}{50}\,\text{log}\,|3(1-x)+1| + x^2 + 1$$

- Looks uncomplicated if plot by sampling at many points
- Is it really that uncomplicated?
- What is happening near that dip?
- In this case can understand by from expression of function
- But in general this may be very hidden

Motivation III: Another example

- Example by William Kahan: Consider

$$f(x) = \frac{1}{50} \log |3(1-x)+1| + x^2 + 1$$

- Looks uncomplicated if plot by sampling at many points
- Is it really that uncomplicated?
- What is happening near that dip?
- In this case can understand by from expression of function
- But in general this may be very hidden
- Can we find **guaranteed bounds** on the **range** of values a function takes over a set?

Motivation IV: Finding bounds

- In analysing many algorithms in the course, we needed **bounds**
- E.g. Lagrange form of the remainder for a Taylor series:
- How big is $|f'(\xi)|$ if $\xi \in [a, b]$?

Motivation IV: Finding bounds

- In analysing many algorithms in the course, we needed **bounds**
- E.g. Lagrange form of the remainder for a Taylor series:
- How big is $|f'(\xi)|$ if $\xi \in [a, b]$?

- Standard numerical methods provide *no* methods to compute bounds of a function f over an interval X!

Motivation IV: Finding bounds

- In analysing many algorithms in the course, we needed **bounds**
- E.g. Lagrange form of the remainder for a Taylor series:
- How big is $|f'(\xi)|$ if $\xi \in [a, b]$?

- Standard numerical methods provide *no* methods to compute bounds of a function f over an interval X!
- This is equivalent to **global optimisation**: find the maximum and minimum of f on X

Motivation V: Representing numbers

- How can we truly represent a real number in a computation?

Motivation V: Representing numbers

- How can we truly represent a real number in a computation?
- E.g. $\sqrt{3}$ or π, or even 0.1?

Motivation V: Representing numbers

- How can we truly represent a real number in a computation?
- E.g. $\sqrt{3}$ or π, or even 0.1?
- What does it actually mean when Julia tells you that $\sqrt{3}$ is 1.7320508075688772?

Motivation V: Representing numbers

- How can we truly represent a real number in a computation?
- E.g. $\sqrt{3}$ or π, or even 0.1?
- What does it actually mean when Julia tells you that $\sqrt{3}$ is 1.7320508075688772?
- It means that $\sqrt{3}$ is a real number *close to* $1.732\ldots$
- In fact, within $\epsilon(1.7320\ldots)$ (“1 ulp” – unit in last place)

Motivation V: Representing numbers

- How can we truly represent a real number in a computation?
- E.g. $\sqrt{3}$ or π, or even 0.1?
- What does it actually mean when Julia tells you that $\sqrt{3}$ is 1.7320508075688772?
- It means that $\sqrt{3}$ is a real number *close to* $1.732\ldots$
- In fact, within $\epsilon(1.7320\ldots)$ ("1 ulp" – unit in last place)
- I.e. It is in fact telling us that $\sqrt{(3)}$ is in certain **interval**

Collaboration I

Representing intervals

Suppose you want to represent a finite **interval** or **range** of real numbers.

Collaboration I

Representing intervals

Suppose you want to represent a finite **interval** or **range** of real numbers.

1. What is one way of representing that?
2. What is an alternative representation?
3. Suppose that you want to represent a **semi-infinite** range (i.e. one which is infinite only on one side, and finite on the other). Do both of the representations work?

Calculating with intervals: sets

- These examples suggest the following:

  *we need to calculate with **sets** of real numbers!*
- Instead of individual real numbers

Calculating with intervals: sets

- These examples suggest the following:

  *we need to calculate with **sets** of real numbers!*
- Instead of individual real numbers

- What does it mean to “calculate with a set”?
- What are basic questions about function f on set X?

Range of a function

- The basic question:

  *Calculate the **range** of a function f over the set X*

. . .

- range$(f; X) := \{f(x) : x \in X\}$

Range of a function

- The basic question:

  *Calculate the **range** of a function f over the set X*

. . .

- $\text{range}(f; X) := \{f(x) : x \in X\}$
- The set of all possible output values for all inputs in X

Range of a function

- The basic question:

  *Calculate the **range** of a function f over the set X*

. . .

- range$(f; X) := \{f(x) : x \in X\}$
- The set of all possible output values for all inputs in X
- Mathematics assumes that the range is accessible

Range of a function

- The basic question:

  *Calculate the **range** of a function f over the set X*

. . .

- range$(f; X) := \{f(x) : x \in X\}$
- The set of all possible output values for all inputs in X
- Mathematics assumes that the range is accessible
- We know that the range is a closed and bounded interval if f is continuous and X is closed and bounded

Range of a function

- The basic question:

  *Calculate the **range** of a function f over the set X*

. . .

- range$(f; X) := \{f(x) : x \in X\}$
- The set of all possible output values for all inputs in X
- Mathematics assumes that the range is accessible
- We know that the range is a closed and bounded interval if f is continuous and X is closed and bounded
- But can we **calculate** the range of a function?

Range II

- Conceptually easy: Find minimum and maximum over X

Range II

- Conceptually easy: Find minimum and maximum over X
- That is a difficult optimization problem!

Range II

- Conceptually easy: Find minimum and maximum over X
- That is a difficult optimization problem!
- Can we obtain *some* information about range more easily?

Range II

- Conceptually easy: Find minimum and maximum over X
- That is a difficult optimization problem!
- Can we obtain *some* information about range more easily?
- What would be most useful?

Range II

- Conceptually easy: Find minimum and maximum over X
- That is a difficult optimization problem!
- Can we obtain *some* information about range more easily?
- What would be most useful?
- What are the simplest sets to think about?

Intervals

- Range of real numbers
- Simplest: (closed) **interval** on real line:

$$X = [a..b] = \{a \leq x \leq b : x \in \mathbb{R}\}$$

- (Standard notation: $[a, b]$)

Intervals

- Range of real numbers
- Simplest: (closed) **interval** on real line:

$$X = [a..b] = \{a \leq x \leq b : x \in \mathbb{R}\}$$

- (Standard notation: $[a, b]$)
- Infinite (uncountable) number of elements x in set X

Intervals

- Range of real numbers
- Simplest: (closed) **interval** on real line:

$$X = [a..b] = \{a \leq x \leq b : x \in \mathbb{R}\}$$

- (Standard notation: $[a, b]$)
- Infinite (uncountable) number of elements x in set X

Defining intervals in Julia

- How can we represent an interval X in Julia?

Defining intervals in Julia

- How can we represent an interval X in Julia?
- Let's define a `SimpleInterval` type:

Defining intervals in Julia

- How can we represent an interval X in Julia?
- Let's define a `SimpleInterval` type:

~~~
```julia
struct SimpleInterval
    inf::Float64
    sup::Float64
end
```
~~~

. . .

- And set operations, e.g.

```julia
Base.in(a::Real, X::SimpleInterval) = X.inf ≤ a ≤ X.sup
```

Defining intervals in Julia

- How can we represent an interval X in Julia?
- Let's define a `SimpleInterval` type:

```julia
struct SimpleInterval
    inf::Float64
    sup::Float64
end
```

. . .

- And set operations, e.g.

  ```
  Base.in(a::Real, X::SimpleInterval) = X.inf ≤ a ≤ X.sup
  ```

- Can we define *functions* on these *sets*?

Defining intervals in Julia

- How can we represent an interval X in Julia?
- Let's define a `SimpleInterval` type:

```julia
struct SimpleInterval
    inf::Float64
    sup::Float64
end
```

. . .

- And set operations, e.g.

  ```
  Base.in(a::Real, X::SimpleInterval) = X.inf ≤ a ≤ X.sup
  ```

- Can we define *functions* on these *sets*?

Functions on intervals

- Given an interval X
- Suppose f is a function like $f(x) = x^2$

Functions on intervals

- Given an interval X
- Suppose f is a function like $f(x) = x^2$
- Can we define $f(X)$?
- What should this mean?

Functions on intervals

- Given an interval X
- Suppose f is a function like $f(x) = x^2$
- Can we define $f(X)$?
- What should this mean?
- How should we calculate it?

Functions on intervals

- Given an interval X
- Suppose f is a function like $f(x) = x^2$
- Can we define $f(X)$?
- What should this mean?
- How should we calculate it?
- Goal: Find **range** of f over X, i.e. set of possible values

Functions on intervals II

- Apply f to X by applying f to *each element of* X
- Output is a new set

Functions on intervals II

- Apply f to X by applying f to *each element of* X
- Output is a new set
- Obviously *impossible* to do this since too many elements

Functions on intervals II

- Apply f to X by applying f to *each element of* X
- Output is a new set
- Obviously *impossible* to do this since too many elements
- Can we calculate the result by hand instead?

Collaboration II

Squaring a set

Suppose $f(x) = x^2$. What does it mean to square a set? We mean that we want to *square every element in the set*.

1. If $X = [1..2]$, what is the range of f over X?

Collaboration II

Squaring a set

Suppose $f(x) = x^2$. What does it mean to square a set? We mean that we want to *square every element in the set*.

1. If $X = [1..2]$, what is the range of f over X?
2. How could we calculate this automatically?
3. What is the range over $X = [-1..1]$?
4. What is the general solution?
5. What about for other functions?

Example: Squaring

- Let's think about $f(x) = x^2$
- With $X = [1..2]$

Example: Squaring

- Let's think about $f(x) = x^2$
- With $X = [1..2]$
- What is result of squaring every element $x \in X$?

Example: Squaring

- Let's think about $f(x) = x^2$
- With $X = [1..2]$
- What is result of squaring every element $x \in X$?
- What about $[-1..2]^2$?

Squaring II

- We can write down a general definition for X^2:

$$
\begin{aligned}
[a..b] &:= [a^2..b^2] && \text{if } a \geq 0 \\
&:= [0..\,\mathsf{max}(a^2, b^2)] && \text{if } a < 0 \text{ and } b > 0 \\
&:= [b^2..a^2] && \text{if } a < b < 0
\end{aligned}
$$

Example: Addition

- How should we define $X + Y$ for intervals X and Y?

Example: Addition

- How should we define $X + Y$ for intervals X and Y?
- We want to find $x + y$ for *all* $x \in X$ and $y \in Y$

Example: Addition

- How should we define $X + Y$ for intervals X and Y?
- We want to find $x + y$ for *all* $x \in X$ and $y \in Y$
- Operational:

$$[a..b] + [c..d] := [(a + c)..(b + d)]$$

Example: Addition

- How should we define $X + Y$ for intervals X and Y?
- We want to find $x + y$ for *all* $x \in X$ and $y \in Y$
- Operational:

$$[a..b] + [c..d] := [(a + c)..(b + d)]$$

- Problem: What is $[0..1] - [0..1]$?

Correct rounding

- Think about an operation like $\exp(x)$ on a float x

Correct rounding

- Think about an operation like exp(x) on a float x
- Recall: a float x is a special (dyadic) *rational* number

Correct rounding

- Think about an operation like $\exp(x)$ on a float x
- Recall: a float x is a special (dyadic) *rational* number
- $\exp(x)$ will produce a non-float real

Correct rounding

- Think about an operation like $\exp(x)$ on a float x
- Recall: a float x is a special (dyadic) *rational* number
- $\exp(x)$ will produce a non-float real
- **Correct rounding**: Return the *closest* float

Correct rounding

- Think about an operation like $\exp(x)$ on a float x
- Recall: a float x is a special (dyadic) *rational* number
- $\exp(x)$ will produce a non-float real
- **Correct rounding**: Return the *closest* float
- This is *difficult* to do in general; we can use the CRlibm library

Correct rounding

- Think about an operation like $\exp(x)$ on a float x
- Recall: a float x is a special (dyadic) *rational* number
- $\exp(x)$ will produce a non-float real
- **Correct rounding**: Return the *closest* float
- This is *difficult* to do in general; we can use the CRlibm library
- **Faithful rounding**: Return *one of the nearest two floats* (not necessarily the closest one)
- Faithful rounding is much easier; Julia tries to do this

Correct rounding

- Think about an operation like exp(x) on a float x
- Recall: a float x is a special (dyadic) *rational* number
- exp(x) will produce a non-float real
- **Correct rounding**: Return the *closest* float
- This is *difficult* to do in general; we can use the CRlibm library
- **Faithful rounding**: Return *one of the nearest two floats* (not necessarily the closest one)
- Faithful rounding is much easier; Julia tries to do this
- The IEEE-754 standard for floating-point arithmetic mandates correct rounding for `+`, `-`, `*`, `/`, `sqrt`

Correct rounding

- Think about an operation like exp(x) on a float x
- Recall: a float x is a special (dyadic) *rational* number
- exp(x) will produce a non-float real
- **Correct rounding**: Return the *closest* float
- This is *difficult* to do in general; we can use the CRlibm library
- **Faithful rounding**: Return *one of the nearest two floats* (not necessarily the closest one)
- Faithful rounding is much easier; Julia tries to do this
- The IEEE-754 standard for floating-point arithmetic mandates correct rounding for `+`, `-`, `*`, `/`, `sqrt`
- Calculating this requires using (?) additional bits of

Directed rounding

- In practice we do not know in *which* direction rounding occurred

Directed rounding

- In practice we do not know in *which* direction rounding occurred
- In context of interval arithmetic, we need to **bound** this **rounding error**

Directed rounding

- In practice we do not know in *which* direction rounding occurred
- In context of interval arithmetic, we need to **bound** this **rounding error**
- The result provided to the user should be an interval that is **guaranteed** to contain the **true** result

Directed rounding

- In practice we do not know in *which* direction rounding occurred
- In context of interval arithmetic, we need to **bound** this **rounding error**
- The result provided to the user should be an interval that is **guaranteed** to contain the **true** result
- This is often referred to as an **enclosure** of the true result

Directed rounding II

- There are various possible techniques to do this

Directed rounding II

- There are various possible techniques to do this
- The simplest solution to implement is to first do a calculation using faithful rounding

Directed rounding II

- There are various possible techniques to do this
- The simplest solution to implement is to first do a calculation using faithful rounding
- Then artificially force the result **outwards**:

```
- move the left endpoint down (towards $-\infty$)
- move the right endpoint up (towards $+\infty$)
```

Directed rounding II

- There are various possible techniques to do this
- The simplest solution to implement is to first do a calculation using faithful rounding
- Then artificially force the result **outwards**:

```
- move the left endpoint down (towards $-\infty$)
- move the right endpoint up (towards $+\infty$)
```

- In Julia we can accomplish this using the `prevfloat` and `nextfloat` functions
- Note that this gives a result that is 2ulps wide instead of 1ulp (unit in last place)

Simple Julia implementation

- We can implement this easily in Julia:

```julia
struct SimpleInterval
    inf::Float64
    sup::Float64
end

import Base: +
+(x::SimpleInterval, y::SimpleInterval) =
    SimpleInterval( prevfloat(x.inf + y.inf),
                    nextfloat(x.sup + y.sup) )

x = SimpleInterval(0.1, 0.3)
y = SimpleInterval(0.2, 0.4)

x + y
```

Interval extensions

- We call an interval-valued function $F(X)$ an **interval extension** of $f(x)$ if

$$F([x, x]) = [f(x)]$$

i.e. the function applied to an interval containing a single point gives the same value as the original function f

Interval extensions

- We call an interval-valued function $F(X)$ an **interval extension** of $f(x)$ if

$$F([x, x]) = [f(x)]$$

i.e. the function applied to an interval containing a single point gives the same value as the original function f

- Unfortunately, different mathematically-equivalent expressions for functions can give *different* results for non-point intervals

Interval extensions

- We call an interval-valued function $F(X)$ an **interval extension** of $f(x)$ if

$$F([x, x]) = [f(x)]$$

i.e. the function applied to an interval containing a single point gives the same value as the original function f

- Unfortunately, different mathematically-equivalent expressions for functions can give *different* results for non-point intervals
- E.g. $f_1(x) = x^2 - 2x$ vs $f_2(x) = x(x-2)$ vs $f_3(x) = (x-1)^2 - 1$

Interval extensions

- We call an interval-valued function $F(X)$ an **interval extension** of $f(x)$ if

$$F([x, x]) = [f(x)]$$

i.e. the function applied to an interval containing a single point gives the same value as the original function f

- Unfortunately, different mathematically-equivalent expressions for functions can give *different* results for non-point intervals
- E.g. $f_1(x) = x^2 - 2x$ vs $f_2(x) = x(x - 2)$ vs $f_3(x) = (x - 1)^2 - 1$

Applications of interval arithmetic

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$
- Recall that x^* is a root of f if $f(x^*) = 0$

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$
- Recall that x^* is a root of f if $f(x^*) = 0$

- Let's calculate the image of f over $X = [3..4]$

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$
- Recall that x^* is a root of f if $f(x^*) = 0$

- Let's calculate the image of f over $X = [3..4]$
- We get $Y = f(X) = [7..14]$

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$
- Recall that x^* is a root of f if $f(x^*) = 0$

- Let's calculate the image of f over $X = [3..4]$
- We get $Y = f(X) = [7..14]$
- This does not contain 0

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$
- Recall that x^* is a root of f if $f(x^*) = 0$

- Let's calculate the image of f over $X = [3..4]$
- We get $Y = f(X) = [7..14]$
- This does not contain 0
- *Hence* $0 \notin \text{range}(f; X)$

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$
- Recall that x^* is a root of f if $f(x^*) = 0$

- Let's calculate the image of f over $X = [3..4]$
- We get $Y = f(X) = [7..14]$
- This does not contain 0
- *Hence* $0 \notin \text{range}(f; X)$
- So *there is no root of* f *in* X

Application: Finding roots

- Let's consider the problem of calculating roots of $f(x) = x^2 - 2$
- Recall that x^* is a root of f if $f(x^*) = 0$

- Let's calculate the image of f over $X = [3..4]$
- We get $Y = f(X) = [7..14]$
- This does not contain 0
- *Hence* $0 \notin \text{range}(f; X)$
- So *there is no root of f in X*
- We have *proved* this using floating-point computations!

IntervalArithmetic.jl

- Julia package tuned for efficiency
- Almost compliant with the IEEE-1788 standard for interval packages

IntervalArithmetic.jl

- Julia package tuned for efficiency
- Almost compliant with the IEEE-1788 standard for interval packages

```
using IntervalArithmetic

x = 0.1..0.3   # shorthand for `interval(0.1, 0.3)`
y = 0.2..0.4

x + y
```

IntervalArithmetic.jl

- Julia package tuned for efficiency
- Almost compliant with the IEEE-1788 standard for interval packages

```
using IntervalArithmetic

x = 0.1..0.3   # shorthand for `interval(0.1, 0.3)`
y = 0.2..0.4

x + y
```

- Compare

```
x = SimpleInterval(0.1, 0.3)
x + x
```

IntervalArithmetic.jl

- Recall the example of excluding roots
- Let’s see how to do this with `IntervalArithmetic.jl`

IntervalArithmetic.jl

- Recall the example of excluding roots
- Let's see how to do this with `IntervalArithmetic.jl`

~~~~
```julia
X = 3..4
f(x) = x^2 - 2

0 ∈ f(X)  # returns false   # type \in<TAB>
```
~~~~

IntervalArithmetic.jl

- Recall the example of excluding roots
- Let’s see how to do this with `IntervalArithmetic.jl`

```julia
X = 3..4
f(x) = x^2 - 2

0 ∈ f(X)  # returns false   # type \in<TAB>
```

- The interval function $f(X)$ obtained by substituting X instead of x everywhere in the definition is called the **natural interval extension**

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?
- It should be $\{x - x : x \in X\} = 0$

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?
- It should be $\{x - x : x \in X\} = 0$
- But we actually get $\{x - y : x, y \in X\} = [-1..1]$

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?
- It should be $\{x - x : x \in X\} = 0$
- But we actually get $\{x - y : x, y \in X\} = [-1..1]$
- We “cannot tell” that it is the same X both times

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?
- It should be $\{x - x : x \in X\} = 0$
- But we actually get $\{x - y : x, y \in X\} = [-1..1]$
- We “cannot tell” that it is the same X both times
- This is the **dependency problem** of interval arithmetic

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?
- It should be $\{x - x : x \in X\} = 0$
- But we actually get $\{x - y : x, y \in X\} = [-1..1]$
- We “cannot tell” that it is the same X both times
- This is the **dependency problem** of interval arithmetic
- It often leads to an **over-estimation** of ranges

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?
- It should be $\{x - x : x \in X\} = 0$
- But we actually get $\{x - y : x, y \in X\} = [-1..1]$
- We “cannot tell” that it is the same X both times
- This is the **dependency problem** of interval arithmetic
- It often leads to an **over-estimation** of ranges
- This is an obstruction to using interval arithmetic more widely

Dependency problem

- What is $X - X$ for the interval $X := [0..1]$?
- It should be $\{x - x : x \in X\} = 0$
- But we actually get $\{x - y : x, y \in X\} = [-1..1]$
- We “cannot tell” that it is the same X both times
- This is the **dependency problem** of interval arithmetic
- It often leads to an **over-estimation** of ranges
- This is an obstruction to using interval arithmetic more widely
- A partial solution is a more complicated extension called **affine arithmetic**, which tracks linear dependencies

Finding a single root?

- Recall that the **inclusion test** $0 \in f(X)$ allows us to *exclude* a root:

Finding a single root?

- Recall that the **inclusion test** $0 \in f(X)$ allows us to *exclude* a root:
- If $0 \notin f(X)$ then there is no root – theorem

Finding a single root?

- Recall that the **inclusion test** $0 \in f(X)$ allows us to *exclude* a root:
- If $0 \notin f(X)$ then there is no root – theorem
- However, if $0 \in f(X)$ we *cannot conclude anything*

Finding a single root?

- Recall that the **inclusion test** $0 \in f(X)$ allows us to *exclude* a root:
- If $0 \notin f(X)$ then there is no root – theorem
- However, if $0 \in f(X)$ we *cannot conclude anything*
- Since overestimation from the dependency problem may lead to

  $0 \notin \text{range}(f; X)$

  but $0 \in f(X)$

Finding *all* roots

- Given an interval X, how can we find *all* roots in X

Finding *all* roots

- Given an interval X, how can we find *all* roots in X
- So far: know how to *exclude* roots from single interval

Finding *all* roots

- Given an interval X, how can we find *all* roots in X
- So far: know how to *exclude* roots from single interval
- **Idea**: Split interval into **pieces**

Finding *all* roots

- Given an interval X, how can we find *all* roots in X
- So far: know how to *exclude* roots from single interval
- **Idea**: Split interval into **pieces**
- Simplest: Equal-sized pieces – **mincing**

Finding *all* roots

- Given an interval X, how can we find *all* roots in X
- So far: know how to *exclude* roots from single interval
- **Idea**: Split interval into **pieces**
- Simplest: Equal-sized pieces – **mincing**
- Theorem: Over-estimation of range decreases as $\mathcal{O}(w)$
- w is width of each piece

Branch and prune

- How can we improve on this?

Branch and prune

- How can we improve on this?
- **Idea**: (Spatial) branch and prune

Branch and prune

- How can we improve on this?
- **Idea**: (Spatial) branch and prune
- Branch: Bisect
- Prune: Check each piece and throw away if no root

Branch and prune

- How can we improve on this?
- **Idea**: (Spatial) branch and prune
- Branch: Bisect
- Prune: Check each piece and throw away if no root
- Effectively builds a **binary tree** in an efficient way

Branch and prune

- How can we improve on this?
- **Idea**: (Spatial) branch and prune
- Branch: Bisect
- Prune: Check each piece and throw away if no root
- Effectively builds a **binary tree** in an efficient way
- Exhaustive search of the space up to some tolerance

Proving existence and uniqueness of roots

- Start from initial box X_0
- What does branch and prune produce?

Proving existence and uniqueness of roots

- Start from initial box X_0
- What does branch and prune produce?
- List of intervals X^i whose **union** contains all roots in X_0

Proving existence and uniqueness of roots

- Start from initial box X_0
- What does branch and prune produce?
- List of intervals X^i whose **union** contains all roots in X_0
- i.e. if x is a root of f then x is in some X^i

Proving existence and uniqueness of roots

- Start from initial box X_0
- What does branch and prune produce?
- List of intervals X^i whose **union** contains all roots in X_0
- i.e. if x is a root of f then x is in some X^i
- But still don't know if there *are* roots or how many

Collaboration III

Proving that there is a root

Suppose that f is a differentiable function $f : \mathbb{R} \to \mathbb{R}$

1. What is a sufficient condition for there to *exist* a root in an interval $[a, b]$? (There may be more than one root.)
2. What is a sufficient condition to show that it is *unique*?

First solution

- Suppose we have reached a small interval X^i where we they may be a root
- So $0 \in f(X)$

First solution

- Suppose we have reached a small interval X^i where we they may be a root
- So $0 \in f(X)$
- From calculus we know that a sufficient condition for a unique root to exist for a differentiable function f is:

  $|f'(x)| > 0$ for all $x \in X$

First solution

- Suppose we have reached a small interval X^i where we they may be a root
- So $0 \in f(X)$
- From calculus we know that a sufficient condition for a unique root to exist for a differentiable function f is:

  $|f'(x)| > 0$ for all $x \in X$
- How can we check this?

First solution

- Suppose we have reached a small interval X^i where we they may be a root
- So $0 \in f(X)$
- From calculus we know that a sufficient condition for a unique root to exist for a differentiable function f is:

  $|f'(x)| > 0$ for all $x \in X$
- How can we check this?
- **Idea**: Use algorithmic differentiation and interval arithmetic!

Second solution: Interval Newton operator

- It turns out that we can extend ther standard Newton method to the context of intervals

Second solution: Interval Newton operator

- It turns out that we can extend ther standard Newton method to the context of intervals
- Define $\tilde{m}(X)$:= midpoint of interval X

Second solution: Interval Newton operator

- It turns out that we can extend ther standard Newton method to the context of intervals
- Define $\tilde{m}(X)$:= midpoint of interval X
- Need thin interval version: $m(X) := [\tilde{m}(x)..\tilde{m}(x)]$

Second solution: Interval Newton operator

- It turns out that we can extend ther standard Newton method to the context of intervals
- Define $\tilde{m}(X)$:= midpoint of interval X
- Need thin interval version: $m(X) := [\tilde{m}(x)..\tilde{m}(x)]$
- Newton operator is
- $\mathcal{N}_f(X) := m(X) - \frac{f(m(X))}{f'(X)}$

Interval Newton II

- **Idea**: Through $(m, f(m))$ take *all* straight lines whose slope is some number in the interval $f'(X)$

Interval Newton II

- **Idea**: Through $(m, f(m))$ take *all* straight lines whose slope is some number in the interval $f'(X)$
- Here $f'(X)$ is the natural extension *of the derivative function* f'

Interval Newton II

- **Idea**: Through $(m, f(m))$ take *all* straight lines whose slope is some number in the interval $f'(X)$
- Here $f'(X)$ is the natural extension *of the derivative function* f'
- Can calculate using algorithmic differentiation!

Interval Newton II

- **Idea**: Through $(m, f(m))$ take *all* straight lines whose slope is some number in the interval $f'(X)$
- Here $f'(X)$ is the natural extension *of the derivative function* f'
- Can calculate using algorithmic differentiation!

Interval Newton III

- Theorem:
  - Any root in X lies in $\mathcal{N}_f(X)$
  - So if $\mathcal{N}_f(X) \cap X = \emptyset$ then there is no root
  - If $\mathcal{N}_f(X) \subseteq X$ then there is a unique root in X

Higher dimensions

- What are the equivalent simplest sets to intervals in higher dimensions?

Higher dimensions

- What are the equivalent simplest sets to intervals in higher dimensions?
- They are **Cartesian products** of intervals: $X_1 \times X_2$

Higher dimensions

- What are the equivalent simplest sets to intervals in higher dimensions?
- They are **Cartesian products** of intervals: $X_1 \times X_2$
- We can bound a function $f : \mathbb{R}^2 \to \mathbb{R}$ over a box $B = X \times Y$ by *just evaluating f using interval arithmetic!$
- $f(X, Y) \supseteq \{f(x, y) : x \in X, y \in Y\}$

Higher dimensions

- What are the equivalent simplest sets to intervals in higher dimensions?
- They are **Cartesian products** of intervals: $X_1 \times X_2$
- We can bound a function $f : \mathbb{R}^2 \to \mathbb{R}$ over a box $B = X \times Y$ by *just evaluating f using interval arithmetic!$
- $f(X, Y) \supseteq \{f(x, y) : x \in X, y \in Y\}$
- Branch and prune and interval Newton extend directly

Other applications

- Guaranteed global optimization: Branch and bound
- Constraint satisfaction: find the feasible set satisfied by several inequalities – interval constraint propagation
- Solve ODEs rigorously: Tube enclosures of solutions; Taylor models

Summary

- We can define an interval X as a set
- And functions on them such that $f(X)$ contains range$(f; X)$
- Interval arithmetic provides a computationally cheap method to *bound* a function over an input set
- It gives an **enclosure** of the **range**, but is in general an *over*-estimate
- We can prove results such as the non-existence of roots using interval arithmetic
- Branch and prune for excluding roots
- Interval Newton for proving existence and uniqueness
- Global optimization