34. Calculating with sets: Interval arithmetic

Last time

- Spectral methods
- Eigenvalues of operators
- Time-evolution PDEs

Goals for today

- Why do we need to calculate with sets?
- Intervals and functions on them
- Rounding

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

$$\ell_i \le x_i \le L_i$$

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

Motivation III: Standard floats are not always good enough

- Example by Siegried Rump: Calculate

$$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

at $a = 77617$ and $b = 33096$

Motivation III: Standard floats are not always good enough

- Example by Siegried Rump: Calculate

$$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

at $a = 77617$ and $b = 33096$

- Something strange: try `Float32`, `Float64` and `BigFloat`

Motivation III: Standard floats are not always good enough

- Example by Siegried Rump: Calculate

$$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

  at $a = 77617$ and $b = 33096$

- Something strange: try `Float32`, `Float64` and `BigFloat`
- Get totally different answers!

Motivation III: Standard floats are not always good enough

- Example by Siegried Rump: Calculate

$$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

at $a = 77617$ and $b = 33096$

- Something strange: try `Float32`, `Float64` and `BigFloat`
- Get totally different answers!
- Which is correct (if any)?

Motivation III: Standard floats are not always good enough

- Example by Siegried Rump: Calculate

$$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

  at $a = 77617$ and $b = 33096$

- Something strange: try `Float32`, `Float64` and `BigFloat`
- Get totally different answers!
- Which is correct (if any)?
- How **guarantee** that result of calculation using floats is

Motivation IV: 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 IV: 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 IV: 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

Motivation IV: 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 **range** of values a function takes over a set?

Motivation V: Representing numbers

- How can we truly represent a real number in the computer?

Motivation V: Representing numbers

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

Motivation V: Representing numbers

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

Motivation V: Representing numbers

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

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

- Basic question: calculate **range** of function f over set X

Range of a function

- Basic question: calculate **range** of function f over set X
- range$(f; X) := \{f(x) : x \in X\}$

Range of a function

- Basic question: calculate **range** of function f over set X
- range$(f; X) := \{f(x) : x \in X\}$
- Set of all possible output values for inputs in X

Range of a function

- Basic question: calculate **range** of function f over set X
- $\text{range}(f; X) := \{f(x) : x \in X\}$
- Set of all possible output values for inputs in X
- Mathematics assumes that the range is accessible

Range of a function

- Basic question: calculate **range** of function f over set X
- range$(f; X) := \{f(x) : x \in X\}$
- Set of all possible output values for inputs in X
- Mathematics assumes that the range is accessible
- 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 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
- How can we represent an interval X in Julia?

Intervals in Julia

- Define new `SimpleInterval` type:

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

Intervals in Julia

- Define new `SimpleInterval` type:

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

- And set operations, e.g.

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

Intervals in Julia

- Define new `SimpleInterval` type:

```
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 *arithmetic* 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?

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

- General definition for X^2:

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

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?
- Want to add *all* pairs x, y with $x \in X$ and $y \in Y$

Example: Addition

- How should we define $X + Y$ for intervals X and Y?
- Want to add *all* pairs x, y with $x \in X$ 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?
- Want to add *all* pairs x, y with $x \in X$ and $y \in Y$
- Operational:

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

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

Correct rounding

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

Correct rounding

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

Correct rounding

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

Correct rounding

- Think about operation like exp(x) on 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 operation like $\exp(x)$ on 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: CRlibm library

Correct rounding

- Think about operation like $\exp(x)$ on 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: CRlibm library
- IEEE-754 standard mandates correct rounding for `+`, `-`, `*`, `/`, `sqrt`

Correct rounding

- Think about operation like $\exp(x)$ on 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: CRlibm library
- IEEE-754 standard mandates correct rounding for `+`, `-`, `*`, `/`, `sqrt`
- Requires using additional bits of precision internally

Directed rounding

- In practice we cannot know in *which* direction rounding occurred

Directed rounding

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

Directed rounding

- In practice we cannot know in *which* direction rounding occurred
- In context of interval arithmetic, need to **bound** this **error**
- I.e. result provided to user should be an interval that is **guaranteed** to contain **true** value

Directed rounding

- In practice we cannot know in *which* direction rounding occurred
- In context of interval arithmetic, need to **bound** this **error**
- I.e. result provided to user should be an interval that is **guaranteed** to contain **true** value
- “**Enclosure**”

Directed rounding II

- Various possible techniques to do this:

  . . .

  1. Control processor's rounding direction

Directed rounding II

- Various possible techniques to do this:

  . . .

  1. Control processor's rounding direction
     - Possible but technically difficult and slow, not thread-safe

Directed rounding II

- Various possible techniques to do this:

  . . .

  1. Control processor's rounding direction
     - Possible but technically difficult and slow, not thread-safe
  2. Artificially move the result **outwards**:
     - move left endpoint down (towards $-\infty$)
     - move right endpoint up (towards $+\infty$)

Directed rounding II

- Various possible techniques to do this:

   . . .

  1. Control processor's rounding direction
     - Possible but technically difficult and slow, not thread-safe
  2. Artificially move the result **outwards**:
     - move left endpoint down (towards $-\infty$)
     - move right endpoint up (towards $+\infty$)
     - In Julia: `prevfloat` and `nextfloat`

Directed rounding II

- Various possible techniques to do this:

  . . .

  1. Control processor's rounding direction
     - Possible but technically difficult and slow, not thread-safe
  2. Artificially move the result **outwards**:
     - move left endpoint down (towards $-\infty$)
     - move right endpoint up (towards $+\infty$)
     - In Julia: `prevfloat` and `nextfloat`
     - Gives result that is 2ulps wide instead of 1ulp (unit in last place)

Application: Finding roots

- Define $f(x) = x^2 - 2$

Application: Finding roots

- Define $f(x) = x^2 - 2$
- Calculate image of f over $X = [3..4]$

Application: Finding roots

- Define $f(x) = x^2 - 2$
- Calculate image of f over $X = [3..4]$
- Get $Y = f(X) = [7..14]$

Application: Finding roots

- Define $f(x) = x^2 - 2$
- Calculate image of f over $X = [3..4]$
- Get $Y = f(X) = [7..14]$
- This does not contain 0

Application: Finding roots

- Define $f(x) = x^2 - 2$
- Calculate image of f over $X = [3..4]$
- Get $Y = f(X) = [7..14]$
- This does not contain 0
- *Hence* $0 \notin \text{range}(f; X)$

Application: Finding roots

- Define $f(x) = x^2 - 2$
- Calculate image of f over $X = [3..4]$
- 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

- Define $f(x) = x^2 - 2$
- Calculate image of f over $X = [3..4]$
- Get $Y = f(X) = [7..14]$
- This does not contain 0
- *Hence* $0 \notin \mathrm{range}(f; X)$
- So *there is no root of* f *in* X
- *Proved* using floating-point computations!

Summary

- Defined intervals as sets and functions / arithmetic on them
- Enable us to calculate **enclosure** of **range** of function
- Can prove results such as non-existence of roots using interval arithmetic