1. What is numerical analysis?

Outline for today's class

- Course logistics
- Introductions
- What is numerical analysis?
- Working through an example problem: Calculating $\sqrt{}$
- Introduction to the Julia language
- What does it mean to solve a problem?

What is *numerical analysis*?

What is *numerical analysis*?

- “Numerical” modifies “analysis”

What is *numerical analysis*?

- “Numerical” modifies “analysis”
- “Analysis” = “mathematical analysis” = **calculus**
- What is special about calculus?

What is *numerical analysis*?

- "Numerical" modifies "analysis"
- "Analysis" = "mathematical analysis" = **calculus**
- What is special about calculus?

- Used for models in science and engineering

What is *numerical analysis*?

- "Numerical" modifies "analysis"
- "Analysis" = "mathematical analysis" = **calculus**
- What is special about calculus?

- Used for models in science and engineering

- How does it differ from computer science?

What is *numerical analysis*?

- "Numerical" modifies "analysis"
- "Analysis" = "mathematical analysis" = **calculus**
- What is special about calculus?

- Used for models in science and engineering

- How does it differ from computer science?
- We will use many concepts from computer science

Alternative / related subject titles

- Numerical methods
- Numerical computation

- Technical computing
- Scientific computing
- Computational science and engineering

- Numerical mathematics
- Experimental mathematics

So what is the course actually about?

- Solving **problems**
- That arise in science, engineering, economics, mathematics, …
- *Problem*: mathematical description of something to calculate

So what is the course actually about?

- Solving **problems**
- That arise in science, engineering, economics, mathematics, …
- *Problem*: mathematical description of something to calculate

- Using the **computer** to solve problems in science, engineering, mathematics…

How is this different from computer science?

How is this different from computer science?

- What kind of objects will *we* be dealing with?

How is this different from computer science?

- What kind of objects will *we* be dealing with?

- Functions $\mathbb{R} \to \mathbb{R}$
- $\mathbb{R}$ is the set of all **real numbers**

How is this different from computer science?

- What kind of objects will *we* be dealing with?

- Functions $\mathbb{R} \to \mathbb{R}$
- $\mathbb{R}$ is the set of all **real numbers**

- Calculus: differentation, integration
- Solution of a differential equation $\dot{x} = f(x)$ is a continuous function $t \mapsto x(t)$ for $t \in \mathbb{R}$

How is this different from computer science?

- What kind of objects will *we* be dealing with?
- Functions $\mathbb{R} \to \mathbb{R}$
- $\mathbb{R}$ is the set of all **real numbers**
- Calculus: differentation, integration
- Solution of a differential equation $\dot{x} = f(x)$ is a continuous function $t \mapsto x(t)$ for $t \in \mathbb{R}$
- Computer science: Discrete objects

Collaborative exercise 1: Is $\sqrt{x}$ an integer?

- Let's look at a basic example that you might see in a computer science course:

Is $\sqrt{x}$ of a non-negative integer x an integer?

- In a breakout room, write a program (algorithm) to find whether the square root of an integer x is also an integer.
  - Input: `x` – an integer
  - Output: `y` – a Boolean (`true` or `false`)

Collaborative exercise 1: Is $\sqrt{x}$ an integer?

- Let's look at a basic example that you might see in a computer science course:

Is $\sqrt{x}$ of a non-negative integer x an integer?

- In a breakout room, write a program (algorithm) to find whether the square root of an integer x is also an integer.
  - Input: `x` – an integer
  - Output: `y` – a Boolean (`true` or `false`)

Rules

- Write, in *words*, the simplest version of this algorithm.
- Use only `+`, `-` `*` `/` (division)
- You may *not* use the `sqrt` function!
- Variables take only integer values

Coding the algorithm in Julia

- Live coding in Julia with the Pluto notebook

Collaborative exercise 2: *Calculate* $y = \sqrt{x}$

Now let's go *beyond* "computer science"

Collaborative exercise 2: *Calculate* $y = \sqrt{x}$

Now let's go *beyond* "computer science"

Calculating the square root

1. Modify your algorithm to return information about the **value** of the square root, whether or not it is an integer.
2. Does your new algorithm work if the input is a real number like 17.35?
3. How can you extend your method to get a *more accurate* value for $\sqrt{x}$?
4. Can you think of a better way to get a **first approximation** for the $\sqrt{x}$, using mathematical properties of the $\sqrt{}$ function?

Coding the new problem in Julia

What problem have we solved?

- What kind of problem have we solved by calculating $\sqrt{x}$?
- What does $y = \sqrt{x}$ satisfy?

What problem have we solved?

- What kind of problem have we solved by calculating $\sqrt{x}$?
- What does $y = \sqrt{x}$ satisfy?

- Since $y^2 = x$, we have **solved an equation** numerically!

What problem have we solved?

- What kind of problem have we solved by calculating $\sqrt{x}$?
- What does $y = \sqrt{x}$ satisfy?

- Since $y^2 = x$, we have **solved an equation** numerically!

- Can we solve $y^3 = x$?
- Can we solve $\cos(x) = x$?
- Can we solve $g(y) = x$?

(Computational) problems

- Given **input** data x, calculate an **output** y
- A problem is defined by a **function**

$$y = f(x)$$

- Our goal is to “solve the problem”
- I.e. given input data x, **calculate** output y

(Computational) problems

- Given **input** data x, calculate an **output** y
- A problem is defined by a **function**

$$y = f(x)$$

- Our goal is to "solve the problem"
- I.e. given input data x, **calculate** output y

- Mathematics often tells us that a solution *exists*, but not **how to calculate it**!
- We will be concerned with how **sensitive** the output is to variations in the input?

How do we solve problems numerically?

- We must first **represent** the input data in the computer somehow
- How should we deal with **real numbers**?

How do we solve problems numerically?

- We must first **represent** the input data in the computer somehow
- How should we deal with **real numbers**?
- Usually must **approximate** them

How do we solve problems numerically?

- We must first **represent** the input data in the computer somehow
- How should we deal with **real numbers**?
- Usually must **approximate** them
- Need to find a **numerical method** or **algorithm** corresponding to the problem f
- Can only calculate **approximation** to output

How good is our solution?

- Approximate input x by $\tilde{x}$
- Approximate problem f by algorithm $\tilde{f}$
- Get approximate output $\tilde{y}$

How good is our solution?

- Approximate input x by $\tilde{x}$
- Approximate problem f by algorithm $\tilde{f}$
- Get approximate output $\tilde{y}$

- How close is $\tilde{y}$ to the true solution y?
- We need to **analyse** the approximations to understand how good the solution might be.
- Note that usually we do not have access to the true solution to compare to

Key feature of approximation algorithms

- We are given a **tolerance** ϵ
- We must find an approximation of the true solution within a distance ϵ

Key feature of approximation algorithms

- We are given a **tolerance** ϵ
- We must find an approximation of the true solution within a distance ϵ

- We must be able to do this for *any* ϵ!
- We will need to do more work as $\epsilon \to 0$. **How much?**

Summary

- We can only solve problems approximately to within some tolerance
- But we need to be able to do so for *any* tolerance
- Developing approximation algorithms for solving continuous problems is the heart of **numerical analysis**