1. Invitation to numerical analysis

What is the course about?

- Alternative names:
  - Numerical methods
  - Numerical computation
  - Technical computing
  - Scientific computing
  - Computational science and engineering
- **Numerical**: Calculate numbers (with a computer)
- Analyse / calculate in a numerical way
- Or: analyse a numerical computation

So what is the course actually about?

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

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

Problems

- Given input data x, calculate output data y
- I.e. problem is defined by a **function**

$$y = f(x)$$

- Our goal: “solve the problem”
- I.e. given input data x, **calculate** output y
- Mathematics does not always (or usually) tell us **how to calculate**
- How **sensitive** is output to variations in input? **Conditioning** of a problem

Example: Eigenvalues

- Problem: Given a matrix M (the input x), find the eigenvalues of M (the output y)
- Mathematics: “just find roots of characteristic polynomial of M”
- How can we actually *find* those roots? Is that even possible?
- Is that even the right approach to *calculate*?

How do we solve problems numerically?

- Need to **represent** input data in computer
- How deal with **real numbers**?
- Must (usually) **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 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

Example: Modelling a fluid

- Suppose we want to model a fluid (**simulation**)
- Simplest naive model: Colliding hard spheres (3D) or discs (2D)
- Want to **simulate** this system, i.e. reproduce its behaviour in the computer
- Which (mathematical) problems arise?

Collision of two discs

- Main component of simulation: collision of two discs
- Input:
  - positions x_1, x_2
  - velocities v_1, v_2
- Problem: Calculate **collision time**
- How can we solve this problem?

Time stepping

- One possible algorithm: Take small time steps δt
- Advance $x_i \leftarrow x_i + \delta t\, v_i$
- After each step, check for **overlap**
- Overlap condition: distance$(x_i, x_j) \leq 2r$
- How calculate distance?
- $f(x_1, x_2, y_1, y_2) = (x_1 - x_2)^2 + (y_1 - y_2)^2 \leq (2r)^2$
- If overlap, backtrack to find t when $f = 0$

Event-driven algorithm

- Alternative **event-driven** algorithm:
- Find the **event** of interest: collision
- Find collision time t^* directly by solving an equation – problem set 1
- Find that we need to **solve a quadratic equation** $at^2 + bt + c = 0$.

Solving a quadratic equation

- Suppose that $f(t) := at^2 + bt + c = 0$.
- How can we **solve** this equation, i.e. find the values of t for which $f(t) = 0$?
- In the particular case of quadratic equations, we are lucky and happy: there is an **analytical solution** that we can use:

$$t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- NB: Usually we do *not* know an analytical solution!

Questions about the quadratic formula

- Can we plug in "machine numbers" and get correct answer?
- Suppose can calculate **discriminant** $d := b^2 - 4ac$
- How calculate $\sqrt{d}$?
- Need an **algorithm** for square root

Can we calculate square root

- Problem: "Find the positive square root of d"
- Input: $d = 3$
- **Impossible** to calculate $y = \sqrt{d}$ exactly using $+$, $-$, $*$, $/$ starting from integers
- Since $\sqrt{3}$ is **irrational** number
- Infinite number of digits ($\sqrt{3} = 1.732050...$); will take infinite time!
- So what **can** we do?

Approximation algorithms for square root

- **Approximate** $\sqrt{d}$ up to some **tolerance** (distance from true value)
- Use a variant of “guess and check”:
- Often easier to **verify** a solution than **find** the solution
- How check that candidate s is close to $\sqrt{d}$?

Verification for square root

- $x = \sqrt{d}$ satisfies the equation $f(x) := x^2 - d = 0$
- Given a candidate $\tilde{y}$, find **residual** $f(\tilde{x}) = \tilde{x}^2 - d$
- Different from **error** $x - \tilde{x}$ – usually unknown
- If residual “small enough” then $\tilde{x}$ is “good” approximation

Finding a square root

- We know more information: not only *distance* but also *sign* of residual
- So can know if $\tilde{x}$ is too low or too high
- In this way we can **bracket** the root
- Then **search** for solutions
- $1 < \sqrt{3} < 2$ since $1^2 < 3 < 2^2$
- Now try 1.1^2 and 1.9^2 etc.
- What is efficient way to do this search?

Bisection

- Use **bisection search**
- Input: a and b such that $f(a) < 0 < f(b)$
- Find **midpoint** $m := \frac{1}{2}(a + b)$
- Check sign$(f(m))$ and choose correct half of interval
- Repeat with new interval
- **Code**

Babylonian algorithm

- A particular way to do this is as follows. Suppose x_0 is a candidate for $\sqrt{d}$.
- How can we find another, different candidate? We are looking for something such that $x_0 y_0 = d$, i.e. $y_0 = d/x_0$
- This is on the opposite side of x_0.
- Now take the mean

$$x_1 := \frac{1}{2}\left(x_0 + \frac{d}{x_0}\right)$$

Comparing algorithms

- We have described two approximation algorithms.
- In principle they can produce a result that is **as good as we would like**, i.e. such that it is as close as desired to $\sqrt{3}$
- Which of these two approximation algorithms is **better**?
- What does better mean? We would like to produce the answer as quickly as possible. So we need to know **how fast** each method approaches the true solution.
- **Rate of convergence**.

Summary

- We can only solve problems approximately
- We want to know how good the approximate solution is: Sensitivity = conditioning
- Different algorithms can have different rates of convergence