25. Review, software packages and outlook

Summary of the previous class

- Interval arithmetic
- Intervals $X = [a, b]$
- Define functions $f(X)$ to contain range$(f; X) := \{f(x) : x \in X\}$
- Can use to exclude roots and prove existence

Goals for today

- Review of the course
- Main ideas and techniques

- Relevant Julia software packages

- Future directions

Review of the course

Overview

1. Representing numbers and functions
2. Solving (nonlinear) equations
   - Root finding
   - Iterative methods
   - Differentiation: numerical and automatic
3. Linear algebra
   - Solving linear equations: Direct and iterative
   - Least squares
   - Matrix factorisations
4. Conditioning and stability
5. Interpolation
   - Numerical integration

So what *is* numerical analysis?

- "Solving problems numerically"
- Using **approximation algorithms** (usually)
- Where the solution converges to the true solution as $N \to \infty$ or $h \to 0$
- Calculating their rate of convergence
- Finding better algorithms with better convergence properties

Which problems do we solve?

- Many of the problems we have solved are stated in an *implicit* fashion
- " We know that x solves the following equation. Find x "
- Often we can prove mathematically that one (or more) solutions *exist*
- We need to develop algorithms to *calculate* x

Examples

- Find a root of $f : \mathbb{R}^n \to \mathbb{R}^n$, i.e. $\mathbf{f}(\mathbf{x}) = \mathbf{0}$
- Solve a linear system of equations: $A\mathbf{x} = \mathbf{b}$
- Solve an ODE $\dot{x} = \mathbf{f}(\mathbf{x})$

What tools do we have?

- Suppose we want to solve $\mathcal{F}(\mathcal{X}) = 0$
- Iterative methods: Find an f such that the sequence $x^{(n)}$ given by $x^{(n+1)} = f(x^{(n)})$ converges to the solution $\mathcal{X}$
- Reduce to problems we already know how to solve, e.g. linear systems $Ax = b$

Ideas

- Most problems are not solvable exactly, even in principle, so we (almost) always must look for some kind of iterative refinement method
- We need a criterion for when to stop
- Look for “special structure”
  - Solving $Ax = b$ is much more efficient if A is tridiagonal than for a general matrix
  - The Chebyshev interpolation problem is much more efficient than the general polynomial interpolation problem
- The only equations we can really solve are linear, so reduce everything to repeatedly solving linear systems
- The only functions we can really work with are polynomials, so we approximate everything with polynomials

Software

Some useful Julia packages for high-quality numerics

- Root finding: `Roots.jl` in 1D; `NonlinearSolve.jl` and `IntervalRootFinding.jl` for small dimensions
  - `Polynomials.jl`
  - `BifurcationKit.jl`: Sophisticated numerical continuation (following roots as a parameter changes)
- Linear algebra: `LinearAlgebra` standard library; `GenericLinearAlgebra.jl` (e.g. for eigenvalues of `BigFloat` matrices)
- Differentiation: https://juliadiff.org/ `ForwardDiff.jl`, `TaylorSeries.jl`
- ODEs: `DifferentialEquations.jl`

Type-based dispatch

- Often there are several (or hundreds, for example in the case of ODEs) of possible algorithms that you may want to try for a given problem
- It should be easy to switch between them and compare their performance (in terms both of accuracy and of speed) for your particular use case
- A common mechanism in Julia packages to do this selection is using **type-based dispatch**
- Each algorithm is represented by a type; passing an instance of the type to a function specifies which algorithm to use
- The implementation is via multiple dispatch

Optimization

- `Optim.jl`; `GalacticOptim.jl`
- Global optimization

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