Lecture 1: Introduction to the module

COMP5930M Scientific Computation

Today

- Resources
- Assessment
- Scientific Computing
- Nonlinearity
- Newton's method

Module resources

- Staff
  - Dr Toni Lassila, t.lassila@leeds.ac.uk (personal queries)
- Yammer
  - Announcements
  - Questions about lectures and coursework answered
- Minerva (virtual learning environment)
  - Module information
  - Virtual lectures (pre-recorded)
  - Tutorials (live on Collaborate Ultra)
  - Coursework submission (through Turnitin)
  - Final assessment (through Gradescope)

Weekly learning pattern

Learning activities:

- Two video lectures (watch on Minerva)
  - Theoretical aspects of scientific computing
  - Watch on your own time, but suggested to reserve a fixed slot on your timetable every week to get into a routine
  - Questions about material? Ask and discuss in Yammer.

- Live tutorial session (Wednesdays at 11:05 UK time)
  - Practical examples of using MATLAB
  - Coursework hints and tips
  - Participation through Collaborate Ultra
  - Sessions recorded and can be reviewed later

- Weekly multiple choice quiz (Fridays on Gradescope)
  - Measure your learning every week
  - Does not factor into final grade (formative assessment)

Assessment

- Coursework 1 (20%)
  - Released: October 20th
  - Deadline: November 12th, 10:00 a.m. UK time
  - Submission electronically through Minerva
- Coursework 2 (20%)
  - Released: Nov 17th
  - Deadline: December 7th, 10:00 am UK time
  - Submission electronically through Minerva
- Final assessment (60%)
  - Online assessment (through Gradescope) in January
  - 48-hours to complete

What is scientific computing?

Use of computer algorithms to solve mathematical problems arising from science and engineering:

- Computer algebra
  - solution of linear and nonlinear system of equations
  - computational linear algebra
  - vector and tensor analysis
  - computational geometry
  - optimisation algorithms
  - linear/non-linear programming
- Numerical analysis
  - numerical solution of differential equations
  - numerical integration
  - discrete Fourier/Laplace transform

What is scientific computing?

Use of computer algorithms to solve mathematical problems arising from science and engineering:

- Computer algebra (study of algebraic structures)
  - solution of linear and nonlinear system of equations
  - computational linear algebra
  - vector and tensor analysis
  - computational geometry
  - optimisation algorithms
  - linear/non-linear programming
- Numerical analysis (study of continuous functions)
  - numerical solution of differential equations
  - numerical integration
  - discrete Fourier/Laplace transform

The module syllabus

- Part 1: Solving nonlinear equations in one dimension
  - Solving one nonlinear equation: Newton's method
  - Alternatives: bisection method, secant method
  - Issues: convergence of Newton's method
- Part 2: Solving systems of nonlinear equations
  - Solving systems of equations: Gradient descent, Newton
  - Improvements: line-search, continuation methods
  - Partial differential equations: discretisation in space and time
  - Time-dependent problems: time-stepping algoritms

The module syllabus

- Part 1: Solving nonlinear equations in one dimension
  - Solving one nonlinear equation: Newton's method
  - Alternatives: bisection method, secant method
  - Issues: convergence of Newton's method
- Part 2: Solving systems of nonlinear equations
  - Solving systems of equations: Gradient descent, Newton
  - Improvements: line-search, continuation methods
  - Partial differential equations: discretisation in space and time
  - Time-dependent problems: time-stepping algoritms
- Part 3: Linear solvers and algorithms for large systems
  - Linear solvers: direct and indirect
  - Sparsity: computational complexity, pivoting, reordering
  - Direct methods: LU factorisation
  - Indirect methods: Gauss-Seidel, Jacobi, conjugate gradient

The module syllabus

- Part 1: Solving nonlinear equations in one dimension
  - Solving one nonlinear equation: Newton's method
  - Alternatives: bisection method, secant method
  - Issues: convergence of Newton's method
- Part 2: Solving systems of nonlinear equations
  - Solving systems of equations: Gradient descent, Newton
  - Improvements: line-search, continuation methods
  - Partial differential equations: discretisation in space and time
  - Time-dependent problems: time-stepping algoritms
- Part 3: Linear solvers and algorithms for large systems
  - Linear solvers: direct and indirect
  - Sparsity: computational complexity, pivoting, reordering
  - Direct methods: LU factorisation
  - Indirect methods: Gauss-Seidel, Jacobi, conjugate gradient

Books

Not essential, all material contained in lecture notes.
Based on textbook:

Solving nonlinear equations with Newton's method,
CT. Kelley,
SIAM Fundamentals of Algorithms FA01, SIAM 2003.

There are a lot of *numerical methods/analysis* books which cover single nonlinear equations and some that briefly consider systems.

Numerical analysis,
RL. Burden and JD. Faires,
Brooks/Cole 2005 (8th edn).

Nonlinear equations

- What is nonlinearity?
- Why do we need numerical methods to solve nonlinear equations?

Nonlinear equations

- What is nonlinearity?

Prototype scalar equation: Find $x^* \in \mathbb{R}$ such that

$$F(x^*) = 0$$

for a given function f that is **differentiable** with derivative $F'(x) \neq \textit{constant}$ (non-constant derivative)

Prototype system of equations: Find $\vec{x}^* \in \mathbb{R}^n$ such that

$$\vec{F}(\vec{x}^*) = \vec{0}$$

with Jacobian matrix $\vec{F}'(\vec{x}) \neq \textit{constant}$ (non-constant Jacobian)

Nonlinear equations

- What is nonlinearity?

Prototype scalar equation: Find $x^* \in \mathbb{R}$ such that

$$F(x^*) = 0$$

for a given function f that is **differentiable** with derivative $F'(x) \neq \mathit{constant}$ (non-constant derivative)

Prototype system of equations: Find $\vec{x}^* \in \mathbb{R}^n$ such that

$$\vec{F}(\vec{x}^*) = \vec{0}$$

with Jacobian matrix $\vec{F}'(\vec{x}) \neq \mathit{constant}$ (non-constant Jacobian)

Examples of nonlinear equations

- Polynomial equations: $a_p x^p + a_{p-1} x^{p-1} + \ldots + a_1 x + a_0 = 0$ (common approximations for other nonlinear equations)
- Trigonometric equations $\sin(x) + 2\cos(x) = 0$ *etc.*

Examples of nonlinear equations

- Polynomial equations: $a_p x^p + a_{p-1} x^{p-1} + \ldots + a_1 x + a_0 = 0$ (common approximations for other nonlinear equations)
- Trigonometric equations $\sin(x) + 2\cos(x) = 0$ *etc.*
- Many physics equations: $n_1 \sin\theta_1 = n_2 \sin\theta_2$, $E = mc^2$ *etc.*

Examples of nonlinear equations

- Polynomial equations: $a_p x^p + a_{p-1} x^{p-1} + \ldots + a_1 x + a_0 = 0$ (common approximations for other nonlinear equations)
- Trigonometric equations $\sin(x) + 2\cos(x) = 0$ *etc.*
- Many physics equations: $n_1 \sin\theta_1 = n_2 \sin\theta_2$, $E = mc^2$ *etc.*
- Backpropagation for finding weights in a neural network:

$$\frac{\partial J}{\partial W} = \frac{1}{n} \sum_i^n x^{(i)} \left(W x^{(i)} + b - y^{(i)} \right) = 0,$$

where J is the loss function, $x^{(i)}$ the input and $y^{(i)}$ the output

Examples of nonlinear equations

- Polynomial equations: $a_p x^p + a_{p-1} x^{p-1} + \ldots + a_1 x + a_0 = 0$ (common approximations for other nonlinear equations)
- Trigonometric equations $\sin(x) + 2\cos(x) = 0$ *etc.*
- Many physics equations: $n_1 \sin\theta_1 = n_2 \sin\theta_2$, $E = mc^2$ *etc.*
- Backpropagation for finding weights in a neural network:

$$\frac{\partial J}{\partial W} = \frac{1}{n} \sum_{i}^{n} x^{(i)} \left(W x^{(i)} + b - y^{(i)} \right) = 0,$$

where J is the loss function, $x^{(i)}$ the input and $y^{(i)}$ the output

Closed form solutions of nonlinear equations

- It is rare for nonlinear equations to have a **closed form solution** that can be found through algebraic manipulation

  General solution for $ax^2 + bx + c = 0$:

  $$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

  (assuming we know how take square roots of a negative number $b^2 - 4ac < 0$...)

Closed form solutions of nonlinear equations

- General solution for $ax^3 + bx^2 + cx + d = 0$:

$$
\begin{aligned}
x_1 = & -\frac{b}{3a} \\
& -\frac{1}{3a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d + \sqrt{\left(2b^3 - 9abc + 27a^2d\right)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
& -\frac{1}{3a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d - \sqrt{\left(2b^3 - 9abc + 27a^2d\right)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
x_2 = & -\frac{b}{3a} \\
& +\frac{1 + i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d + \sqrt{\left(2b^3 - 9abc + 27a^2d\right)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
& +\frac{1 - i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d - \sqrt{\left(2b^3 - 9abc + 27a^2d\right)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
x_3 = & -\frac{b}{3a} \\
& +\frac{1 - i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d + \sqrt{\left(2b^3 - 9abc + 27a^2d\right)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
& +\frac{1 + i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d - \sqrt{\left(2b^3 - 9abc + 27a^2d\right)^2 - 4\left(b^2 - 3ac\right)^3}\right]}
\end{aligned}
$$

Numerical methods

- It is rare for nonlinear equations to have a **closed form solution** that can be found through algebraic manipulation
- We require numerical (approximate) solution
- This may still be tricky for large systems of nonlinear equations

Newton's Method for finding the zeros of $F(x)$

- Assume we have access to the **differentiable** function $F(x)$ and its derivative $F'(x)$.
- Assume we have one initial point x_0 that is close to the unknown zero x^* such that $F(x^*) = 0$.
- We look for a sequence of iterates $x_0, x_1, \dots$ such that

$$\lim_{n\to\infty} F(x_n) \to 0.$$

i.e. the sequence converges to x^*. In practice, we only take finitely many **iterations**. This is called an **iterative method**.

Derivation of Newton' Method

- Since F is differentiable, we can write its tangent at x_0 as:

$$T_0(x) = F(x_0) + F'(x_0)(x - x_0).$$

- Find the intersection between the tangent and the x-axis as:

$$F(x_0) + F'(x_0)(x_1 - x_0) = 0$$

or after manipulation, $x_1 = x_0 - \frac{F(x_0)}{F'(x_0)}$.

Derivation of Newton' Method

- Since F is differentiable, we can write its tangent at x_0 as:

$$T_0(x) = F(x_0) + F'(x_0)(x - x_0).$$

- Find the intersection between the tangent and the x-axis as:

$$F(x_0) + F'(x_0)(x_1 - x_0) = 0$$

or after manipulation, $x_1 = x_0 - \frac{F(x_0)}{F'(x_0)}$.

- By repeating the tangent approximation at every step we get:

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}, \quad n = 1, 2, \ldots$$

a sequence of iterates x_n that converges to x^* **under certain conditions** (to be discussed in detail later).

Derivation of Newton' Method

- Since F is differentiable, we can write its tangent at x_0 as:

$$T_0(x) = F(x_0) + F'(x_0)(x - x_0).$$

- Find the intersection between the tangent and the x-axis as:

$$F(x_0) + F'(x_0)(x_1 - x_0) = 0$$

or after manipulation, $x_1 = x_0 - \frac{F(x_0)}{F'(x_0)}$.

- By repeating the tangent approximation at every step we get:

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}, \quad n = 1, 2, \ldots$$

a sequence of iterates x_n that converges to x^* **under certain conditions** (to be discussed in detail later).

Pros and cons

Performance

- Fast (quadratic convergence rate)
- Not robust

Other issues

- Requires the derivative function
- Requires a "*good*" initial guess