EOSC 213: Computational Methods in Geological Engineering

Lecture 1: Course overview + motivation + computational tools setup

Shadab Ahamed

Department of Earth, Ocean and Atmospheric Sciences,
The University of British Columbia

Tue, Jan 6, 2026

Class time: Tue/Thu 12:30–2:00 pm **Location:** H.R. MacMillan Building (MCML), Floor 1, Room 158

Teaching team

Instructor: Shadab Ahamed
Current Appointment: Postdoctoral Researcher, EOAS, UBC
Previously: PhD (Physics), UBC
Research Interests: Deep learning for Generative AI
Email: shadab.ahamed@ubc.ca

Teaching Assistant: Md Shahriar Rahim Siddiqui
Current Appointment: PhD Candidate, EOAS, UBC
Research Interests: Graph Neural Networks and Generative AI, Astrophysics, etc.
Email: shahriar.siddiqui@ubc.ca

Introduction

Why this course?

Scientific computing (a.k.a. computational science) shows up everywhere:

- Simulations (forward modeling)
- Data fitting and analysis (inference from measurements)
- Optimization (calibration, inverse problems, design)
- Visualization (understanding and communicating results)

Goal: Gain understanding through analysis of mathematical models implemented on computers.

The basic workflow

Most computational science problems follow the same pipeline:

1. Observation / story (what do we see?)
2. Mathematical model (equations + assumptions)
3. Discretization (continuous $\rightarrow$ finite representation)
4. Solve / simulate (algorithms + linear algebra)
5. Parameter fitting (match data, quantify uncertainty)
6. Visualize + interpret (does it make physical sense?)

Ordinary Differential Equations (ODEs) classification (with examples)

1. **Linear, first order:** $\dfrac{dy}{dt} + a(t)\, y = b(t)$
2. **Linear, higher order:** $\dfrac{d^2y}{dt^2} + \omega^2 y = 0$
3. **Nonlinear, first order:** $\dfrac{dy}{dt} = \lambda y(1 - y/a)$
4. **Nonlinear, higher order:** $\dfrac{d^2y}{dt^2} + y^3 = 0$
5. **System, linear:** $\dot{\mathbf{y}} = A\mathbf{y}$
6. **System, nonlinear:** $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$

ODEs in vectorized/matrix notation: a *system* of ODEs

An ODE relates an unknown function to its derivatives:

$$\frac{dy}{dt} = f(t, y; p)$$

More generally, a system of ODEs:

$$\frac{d}{dt}\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} f_1(y_1, \ldots, y_n; p) \\ \vdots \\ f_n(y_1, \ldots, y_n; p) \end{pmatrix}$$

- Appears in particle flow, disease propagation, geochemistry, and more.

Some real life examples

Example 1: Tracer decay (ODE → discretization)

Observation: a radioactive tracer (or chemical concentration) decays over time.

Model: first order ODE

$$\frac{dx}{dt} = -\lambda x \qquad \text{(but what is } \lambda\text{?)}$$

Finite-difference discretization (first derivative):

$$\frac{x(t_{i+1}) - x(t_i)}{\Delta t} = -\lambda x(t_i)$$

Update (time-marching simulation):

$$x(t_{i+1}) = (1 - \lambda \Delta t)\, x(t_i)$$

Goal: Measure decay and find an approximation to λ

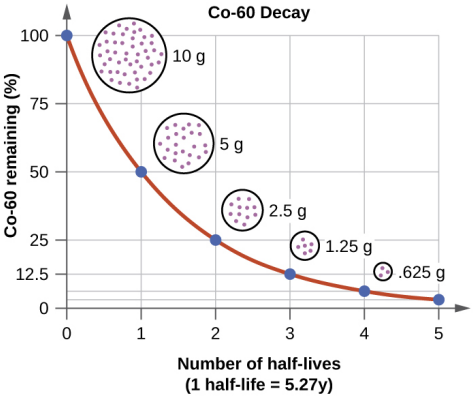

Figure: Radioactive decay curve

Example 1 cont'd: data assimilation (estimate parameters)

Given noisy observations:

$$t = [0, 1, 2, 3] \text{ s}, \quad x = [10.0, 6.7, 4.5, 3.0]$$

we can ask:

- Can the mathematical model reasonably explain the data?
- What is the **best** value of λ that explains the observations (parameter estimation)?

When the model **does not** exactly match the observations, we must adjust its parameters systematically. This will naturally lead to optimization-based approaches and data-driven model calibration later in the term, where we will deal with optimization (find the best-fit λ) and inverse problems (given observations, infer parameter λ of the ODE).

Example 2: Newton's apple (ODE $\rightarrow$ discretization)

Observation: an apple falls.
Model:

$$\frac{d^2x}{dt^2} = -g \quad \text{(but what is } g\text{?)}$$

Finite-difference discretization (second derivative):

$$\frac{x(t_{i+1}) - 2x(t_i) + x(t_{i-1})}{\Delta t^2} = -g$$

Update (time-marching simulation):

$$x(t_{i+1}) = -g\Delta t^2 + 2x(t_i) - x(t_{i-1})$$

Goal: Measure and find an approximation to g

Figure: Newton's apple tree folklore!

Example 2 cont'd: data assimilation (estimate parameters)

Given noisy observations:

$$t = [0, 1, 2, 3] \text{ s}, \quad x = [0, 4.4, 21.0, 54.2] \text{ m},$$

we can ask:

- Can the mathematical model reasonably explain the data?
- What is the **best** value of g that explains the observations (parameter estimation)?

This is the bridge to optimization (find best fit g) and inverse problems (given observations, infer parameter g of the ODE) later in the term.

Example 3: Identifying handwritten digits 0–9 (when physics is hard)

Sometimes the math model is unknown or too complex. Example: recognize digits in an image.

Example 3 cont'd: a simple Machine Learning (ML) model

One model family: convolutional neural networks (CNNs)

$$y = w^\top \tanh(K \star x + b)$$

- x: the image (data)
- K, b, w: parameters learned from examples
- y: probabilities for classes (digits 0–9): y is a vector of size 10, where the element at position i denotes the probability that the input image represents the digit i.
  Example: $y = [0, 0.65, 0, 0, 0, 0, 0, 0.35, 0, 0]$ implies that the input image x has a 65% probability of being the digit 1 and 35% probability of being 7!

Go home and Google these concepts

- **Forward modeling / simulation**
  What does it mean to simulate a physical model forward in time?
- **Discretization of differential equations**
  How do derivatives become finite differences on a computer?
- **Fitting a model to data**
  What does it mean to "fit" a mathematical model to observations?
- **Parameter estimation**
  How can unknown parameters (like λ or g) be inferred from data?
- **Inverse problems (forward vs inverse)**
  What is the difference between predicting data and inferring causes?
- **Numerical stability (time step choice)**
  Why can a numerical simulation fail even if the model is correct?

You **do not** need to understand these in detail **yet**. This is just to build awareness of the ideas we will revisit throughout the course.

Course Logistics

Course syllabus (will be finalized on Canvas)

- Week 1–2: Python refresher + intro to ODEs (including separable ODEs)
- Week 3–5: finite differences + initial value problems (IVPs)
- Week 6–8: nonlinear equations, implicit methods, systems of ODEs
- Week 9–10: discrete Laplacian + boundary value problems (BVPs)
- Week 11–14: parameter estimation / optimization / review of the course

Note

The official schedule is expected to change a bit; the most updated schedule will be on Canvas.

Tools we will use

Core Python stack:

- **Python** in Visual Studio Code IDE (please don't use IDEs like PyCharm or Spyder; we don't provide support for those)
- **NumPy** (arrays, vectors, matrices)
- **Matplotlib** (visualization)
- **PyTorch** (tensors, automatic differentiation, optimization, CPU/GPU)

Recommended online reference:

- Python Programming and Numerical Methods: A Guide for Engineers and Scientists
- Official documentation for NumPy / Matplotlib / PyTorch

Goals of the course

By the end of the term, you should be able to:

- Translate physical models into mathematical form
- Discretize ODE/PDE models into computable systems
- Implement methods cleanly in Python (vectorization, linear algebra)
- Diagnose accuracy/stability issues using plots and sanity checks
- Fit parameters to data (optimization / inverse problems)

Homework assignments (4 total)

Homework 1 (programming foundations):

- Python basics, vectors/matrices, loops vs vectorization, good plotting
- This homework will be posted by Sat, Jan 10, 2026. Keep a watch on Canvas
- **Due: Fri Jan 16, 2026 @ 11:59 pm** (before the drop deadline)

Homework 2–4 (numerical methods):

- Emphasis on numerical concepts + Python implementation
- Clear, well-structured code + brief explanations required

Submitted electronically via Canvas (details will be posted on Canvas).

Grading

Homework Assignments (4)	40%
Midterm I	15%
Midterm II	15%
Final Exam	30%

Dates

Midterm dates and final exam details will be confirmed and posted on Canvas.

Communication

- **Questions about course material:** post on the course discussion forum on Canvas (preferred)
- **Canvas Messages:** fine for general class communication
- **Email:** urgent or private matters only (academic concessions, personal circumstances)

Academic integrity (UBC policy)

Academic integrity is taken very seriously at UBC.

- **Homework:** work independently; you may discuss concepts/strategies, but all submitted work must be your own.
- Copying code/solutions (classmates, internet, previous years) is strictly prohibited.
- Do not post or share course materials publicly.

Generative AI tools (not permitted for homework)

Using tools like ChatGPT/Claude/Copilot to generate homework code or solutions is **not permitted** in this course.

Why we do not allow generative AI for homework

This course is about building **your** computational thinking:

- translating models into code
- debugging and validating results
- learning to reason about stability/accuracy

Research suggests that higher confidence in GenAI can correlate with *reduced critical engagement* and cognitive effort when evaluating outputs.

Evidence (example)

Lee et al. (2025) report *"self-reported reductions in cognitive effort"* when using GenAI in knowledge work.[a]

[a] H.P.H. Lee et al., The Impact of Generative AI on Critical Thinking (Microsoft Research, 2025)

Accommodations and missed work

- If you miss class due to illness: use lecture notes + discussion forum.
- If illness or extenuating circumstances prevent you from meeting a deadline or attending an exam, contact the instructor **as early as possible**.
- Requests for academic concessions must follow UBC Science guidelines.
- Deferred standing for missed final exams is arranged through Science Advising.

Next: Python environment setup (in class)

No slides for this part.

- We will switch to setting up Python environments on your laptops
- Confirm you can run: NumPy, Matplotlib, PyTorch
- We will post installation instructions and troubleshooting tips on Canvas here: `https://canvas.ubc.ca/courses/176796/pages/coding-tools-installation-instructions?module_item_id=8937801`

See you Thursday

- We will start separable ODEs and build intuition via examples.
- Bring your laptop to *every* class.