EOSC 213: Computational Methods in Geological Engineering

Lecture 12: Midterm I logistics + recap (L10–L11) + in-class coding (systems of ODEs)

Shadab Ahamed

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

Thu, Feb 12, 2026

Midterm I: when, where, and format

Midterm I: Tuesday, Feb 24, 2026 (in class). Detailed instructions are posted here. **It is important that you go through these detailed instructions carefully**.

- **Time:** 12:40 pm – 1:50 pm (70 minutes)
- **Arrive by:** 12:30 pm (we start the exam at 12:40 pm sharp)
- **Format:** short-answer + long-answer
- **Pen-and-paper exam:** no electronic devices

Long-answer questions: show all steps (partial credit for showing detailed steps)

Practice questions for Midterm-I: will be posted by this weekend (mostly for the system of ODEs part)

HW02 reminder (important)

HW02 is posted. Submission deadline: Friday, 20th February, 2026 (11:59 pm)

- This homework is **good practice for Midterm I**
- Follow the submission instructions carefully

Submit:

- Written solutions as a **single PDF**
- Python notebook: **both** the `.ipynb` and the corresponding `.html` export

Reminder: make sure your submitted files are correct and complete.

Office hour

Office hour today: first floor of ESB, **2:00 pm - 3:00 pm**
Extra office hour during Midterm break: first floor of ESB, **Thursday, 19th February, 2026, 12:30 pm - 3:00 pm**

- Bring questions about HW02 or Midterm I preparation
- If you cannot make it, email me
- TA will also have slightly longer ($\sim$1.5 hours) office hours during Midterm break

Recap: why systems of ODEs?

Many real models involve **multiple interacting quantities**:

- position + velocity
- coupled chemical species
- coupled components in geoscience models

A system tracks a **state vector** $\mathbf{x}(t)$ with multiple components:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}.$$

Recap: vector/matrix form (the big simplification)

Linear systems often look like,

$$\mathbf{x}'(t) = A\mathbf{x}(t),$$

where:

- $\mathbf{x}(t)$ is the state (all variables stacked)
- A contains **self-effects** (diagonal) and **coupling** (off-diagonal)

Key point: once we have A, we can use linear algebra to predict behavior.

Recap: phase portraits

For 2D systems, we visualize trajectories in the **phase plane**:

$$(x(t), y(t)) \text{ as a curve in the } (x, y)\text{-plane.}$$

- axes are **state variables**, not time
- each curve corresponds to a different initial condition
- direction along the curve shows how the state evolves

Why useful: phase portraits show qualitative behavior at a glance.

Recap: eigenvalues of A predict qualitative behavior

For $\mathbf{x}'(t) = A\mathbf{x}(t)$, eigenvalues of A tell us what happens to modes.

Main cases (2D intuition):

- both eigenvalues $< 0 \Rightarrow$ stable (trajectories go to the origin)
- one > 0, one $< 0 \Rightarrow$ saddle (some directions grow, some decay)
- both $> 0 \Rightarrow$ unstable (trajectories move away)
- complex $a \pm ib$: spirals; $\Re(\lambda) = a$ sets decay/growth, $\Im(\lambda) = b$ sets oscillation
- purely imaginary: center (closed orbits)

Recap: time-stepping methods extend to systems

If $\mathbf{x}'(t) = \mathbf{F}(t, \mathbf{x})$, we apply the same ideas as the scalar case, but with vectors.

Forward Euler (system):

$$\mathbf{x}_{j+1} = \mathbf{x}_j + h\,\mathbf{F}(t_j, \mathbf{x}_j)$$

Explicit Midpoint (system):

$$\mathbf{x}_{j+\frac{1}{2}} = \mathbf{x}_j + \frac{h}{2}\mathbf{F}(t_j, \mathbf{x}_j), \qquad \mathbf{x}_{j+1} = \mathbf{x}_j + h\,\mathbf{F}\left(t_j + \frac{h}{2}, \mathbf{x}_{j+\frac{1}{2}}\right)$$

Backward Euler (system):

$$\mathbf{x}_{j+1} = \mathbf{x}_j + h\,\mathbf{F}(t_{j+1}, \mathbf{x}_{j+1})$$

Some important concepts – I: Even a single ODE is a system

Consider a scalar linear ODE,

$$x'(t) = \lambda x(t)$$

Some important concepts – I: Even a single ODE is a system

Consider a scalar linear ODE,

$$x'(t) = \lambda x(t)$$

We can define a **state vector** with one component,

$$\mathbf{x}(t) = \begin{bmatrix} x(t) \end{bmatrix}$$

Some important concepts – I: Even a single ODE is a system

Consider a scalar linear ODE,

$$x'(t) = \lambda x(t)$$

We can define a **state vector** with one component,

$$\mathbf{x}(t) = \left[x(t)\right]$$

Then the ODE becomes,

$$\mathbf{x}'(t) = \left[\lambda\right] \mathbf{x}(t)$$

Some important concepts – I: Even a single ODE is a system

Consider a scalar linear ODE,

$$x'(t) = \lambda x(t)$$

We can define a **state vector** with one component,

$$\mathbf{x}(t) = \left[x(t)\right]$$

Then the ODE becomes,

$$\mathbf{x}'(t) = \left[\lambda\right] \mathbf{x}(t)$$

So even a single ODE fits the form,

$$\mathbf{x}' = A\mathbf{x},$$

where

$$A = \left[\lambda\right].$$

Why rewrite a single ODE this way?

- The theory for systems applies to scalar equations
- Eigenvalues for a 1×1 matrix are just the number itself
- For real λ, stability analysis becomes uniform:

$$\lambda < 0 \Rightarrow \text{decay}$$

$$\lambda > 0 \Rightarrow \text{growth}$$

Key insight: Systems of ODEs are just a natural extension of scalar ODEs.

What if λ is complex or imaginary?

For the scalar ODE,

$$x'(t) = \lambda x(t),$$

What if λ is complex or imaginary?

For the scalar ODE,

$$x'(t) = \lambda x(t),$$

the solution is,

$$x(t) = x_0 e^{\lambda t}.$$

What if λ is complex or imaginary?

For the scalar ODE,

$$x'(t) = \lambda x(t),$$

the solution is,

$$x(t) = x_0 e^{\lambda t}.$$

If $\lambda = a + ib$ (complex), then

$$e^{\lambda t} = e^{at} e^{ibt}.$$

What if λ is complex or imaginary?

For the scalar ODE,

$$x'(t) = \lambda x(t),$$

the solution is,

$$x(t) = x_0 e^{\lambda t}.$$

If $\lambda = a + ib$ (complex), then

$$e^{\lambda t} = e^{at} e^{ibt}.$$

Using Euler's formula,

$$e^{ibt} = \cos(bt) + i \sin(bt).$$

What if λ is complex or imaginary?

For the scalar ODE,

$$x'(t) = \lambda x(t),$$

the solution is,

$$x(t) = x_0 e^{\lambda t}.$$

If $\lambda = a + ib$ (complex), then

$$e^{\lambda t} = e^{at} e^{ibt}.$$

Using Euler's formula,

$$e^{ibt} = \cos(bt) + i \sin(bt).$$

Therefore,

$$e^{\lambda t} = e^{at} \big(\cos(bt) + i \sin(bt) \big).$$

What if λ is complex or imaginary?

For the scalar ODE,

$$x'(t) = \lambda x(t),$$

the solution is,

$$x(t) = x_0 e^{\lambda t}.$$

If $\lambda = a + ib$ (complex), then

$$e^{\lambda t} = e^{at} e^{ibt}.$$

Using Euler's formula,

$$e^{ibt} = \cos(bt) + i \sin(bt).$$

Therefore,

$$e^{\lambda t} = e^{at} \big(\cos(bt) + i \sin(bt) \big).$$

Interpretation when $b \neq 0$:

- $a < 0 \Rightarrow$ oscillations decay
- $a = 0 \Rightarrow$ pure oscillation (constant amplitude)
- $a > 0 \Rightarrow$ oscillations grow

Some important concepts – II: Eigenvectors are not unique

If $\mathbf{v}$ is an eigenvector of A with eigenvalue λ:

$$A\mathbf{v} = \lambda\mathbf{v},$$

Some important concepts – II: Eigenvectors are not unique

If $\mathbf{v}$ is an eigenvector of A with eigenvalue λ:

$$A\mathbf{v} = \lambda\mathbf{v},$$

then for any nonzero scalar c, let us define,

$$\mathbf{w} := c\mathbf{v}.$$

Some important concepts – II: Eigenvectors are not unique

If $\mathbf{v}$ is an eigenvector of A with eigenvalue λ:

$$A\mathbf{v} = \lambda\mathbf{v},$$

then for any nonzero scalar c, let us define,

$$\mathbf{w} := c\mathbf{v}.$$

Then,

$$A\mathbf{w} = A(c\mathbf{v}) = cA\mathbf{v} = c\lambda\mathbf{v} = \lambda(c\mathbf{v}) = \lambda\mathbf{w}.$$

Some important concepts – II: Eigenvectors are not unique

If $\mathbf{v}$ is an eigenvector of A with eigenvalue λ:

$$A\mathbf{v} = \lambda\mathbf{v},$$

then for any nonzero scalar c, let us define,

$$\mathbf{w} := c\mathbf{v}.$$

Then,

$$A\mathbf{w} = A(c\mathbf{v}) = cA\mathbf{v} = c\lambda\mathbf{v} = \lambda(c\mathbf{v}) = \lambda\mathbf{w}.$$

So, $\mathbf{w} = c\mathbf{v}$ is also an eigenvector of the matrix A corresponding to the same eigenvalue λ.

Geometric meaning of eigenvectors

Eigenvectors determine a **direction** in phase space.

- Any multiple of **v** lies on the same line
- The eigenvector defines a **line through the origin**

Key idea: Eigenvectors represent directions, not specific magnitudes.

Example: eigenvector and eigenvalue in a 3×3 matrix

Consider the matrix

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

Take the vector

$$\mathbf{v} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Compute:

$$A\mathbf{v} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} = 2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 2\mathbf{v}.$$

So $\mathbf{v}$ is an eigenvector with eigenvalue $\lambda = 2$.

Multiples of an eigenvector

Take any nonzero scalar c.

For example:

$$5\mathbf{v} = \begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix}, \qquad -2\mathbf{v} = \begin{pmatrix} -2 \\ 0 \\ 0 \end{pmatrix}.$$

Now compute:

$$A(5\mathbf{v}) = 5(A\mathbf{v}) = 5(2\mathbf{v}) = 2(5\mathbf{v}).$$

Therefore,

$$A(c\mathbf{v}) = \lambda(c\mathbf{v}).$$

Conclusion: Eigenvectors are defined up to scaling. They represent a **direction**, not a specific magnitude.

Do complex multiples of an eigenvector remain eigenvectors?

Suppose $\mathbf{v}$ is an eigenvector of A with eigenvalue λ,

$$A\mathbf{v} = \lambda\mathbf{v}.$$

Let c be any **nonzero scalar**. Then

$$A(c\mathbf{v}) = c\,A\mathbf{v} = c(\lambda\mathbf{v}) = \lambda(c\mathbf{v}).$$

Conclusion:

$c\mathbf{v}$ is also an eigenvector for the same eigenvalue λ,

for any $c \neq 0$ (real *or* complex).

Real vs complex scaling: what changes?

Important subtlety:

- If we work in $\mathbb{R}^n$, eigenvectors are usually taken as **real vectors** and they live in $\mathbb{R}^n$.
- If c is complex and $\mathbf{v}$ is real, then $c\mathbf{v}$ becomes a **complex vector**.
- It is still an eigenvector — but now it lives in $\mathbb{C}^n$.

Examples: complex multiples of an eigenvector

Let

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

Its eigenvalues are

$$\lambda = \pm i.$$

An eigenvector corresponding to $\lambda = i$ is

$$\mathbf{v} = \begin{bmatrix} 1 \\ -i \end{bmatrix}.$$

Now multiply $\mathbf{v}$ by complex scalars,

$$i\mathbf{v} = i \begin{bmatrix} 1 \\ -i \end{bmatrix} = \begin{bmatrix} i \\ -i^2 \end{bmatrix} = \begin{bmatrix} i \\ 1 \end{bmatrix}, \qquad (2+3i)\mathbf{v} = (2+3i) \begin{bmatrix} 1 \\ -i \end{bmatrix} = \begin{bmatrix} 2+3i \\ 3+2i \end{bmatrix}.$$

Since

$$A(c\mathbf{v}) = c(A\mathbf{v}) = c(i\mathbf{v}) = i(c\mathbf{v}),$$

any nonzero complex multiple remains an eigenvector for the same eigenvalue.

Today's plan

Today is mostly coding. The goal is to connect the math to computation.

1. Identify A from a linear system written in components
2. Compute eigenvalues (and eigenvectors) numerically
3. Use eigenvalues to predict behavior (decay/growth/oscillation)
4. Simulate the system using Euler / Midpoint / Backward Euler
5. Compare: time series + phase portraits

Coding checklist (what you should be able to do)

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

- rewrite a component-wise system into $\mathbf{x}'(t) = A\mathbf{x}(t)$
- use `torch.linalg.eig` (or `eigvals`) to compute eigenvalues
- classify eigenvalue cases (real negative/positive, complex, imaginary)
- implement FE / EM / BE updates for vector-valued states
- generate phase portraits from simulations (multiple initial conditions)

Takeaway

Big message:

- Systems of ODEs become manageable when written in vector/matrix form
- Eigenvalues give a "first prediction" of behavior
- Numerical methods for systems are the same ideas as scalar methods – just vectorized

Now we switch to the coding notebook.