EOSC 213: Computational Methods in Geological Engineering

Lecture 6: Stability analysis for Forward Euler / Explicit Midpoint / Backward Euler

Shadab Ahamed

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

Thu, Jan 22, 2026

Course Logistics: TA Office Hour scheduling

Action item (5 minutes):

We will schedule TA office hours using **when2meet**.
Link: https://www.when2meet.com/?34223342-8MfKx

- Please open the when2meet link posted on Canvas
- Mark **all times you are available** during the week
- TA office hours will be scheduled based on majority availability

Goal: choose a time that works for as many students as possible.

We will take 5 minutes now to complete this.

Course Logistics: Course Content and Timeline

The full course timeline has been posted on Canvas.

Link: https://canvas.ubc.ca/courses/176796/pages/course-timeline?module_item_id=8976899

The course is broadly organized into:

- Ordinary differential equations (ODEs)
- Numerical time-stepping methods
- Stability and stiffness
- Boundary value problems
- Parameter estimation and inverse problems

- Homework deadlines: Fridays, 11:59 pm
- Midterms: in-class (dates posted)
- Final exam: centrally scheduled

Note: The timeline is tentative and may be adjusted slightly as we go.

Course Logistics: Python Background

This course uses:

- Python
- NumPy / PyTorch
- Matplotlib

Question for you:

- Would you like an **additional class or review session**?
- focused specifically on Python basics and numerical coding?

Your feedback will help shape how we pace this course.

Course Logistics: How to do well in this course?

Main idea: Active engagement

Some practical advice:

- Keep an eye on **all course announcements**
- Get comfortable with **Python** (it will be very useful for homeworks)
- Go through the **lecture slides before class** (even briefly)
- Play with the **in-class notebooks** yourself and try coding from scratch
- When studying, try to **derive the mathematical steps on your own**
- Start working on **homework problems early**
- **Ask questions**: to classmates, TAs, or me (during class, office hours, or via email)
- **No use of AI Chatbots / AI Autocomplete features for completing your homework**. You can use AI Chatbots to teach yourself; but don't use it to generate content for any of the homework

Where we are in the course

- Lecture 3: First-order ODEs and analytic solutions
- Lecture 4: Forward Euler (finite differences & time stepping)
- Lecture 5: Forward Euler issues + Midpoint + Backward Euler glimpse
- **Lecture 6: Stability analysis (math)**

Goal today: predict when a method behaves well *before* you code.

Lecture 5 recap: what we observed

For the same decay ODE, changing the step size h led to:

- smooth decay (looks physical)
- sign-flip oscillations (alternating $\pm$ values)
- divergence (numbers blow up)

Today: derive step-size conditions that explain these outcomes.

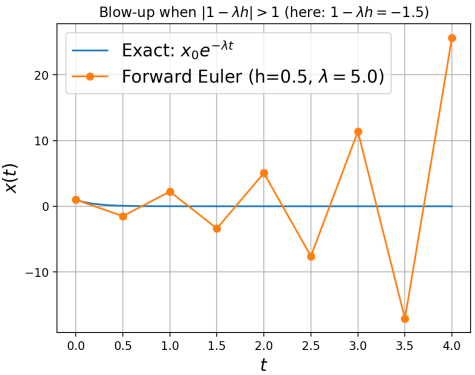


Figure: Forward Euler with $|1 - \lambda h| > 1$ (unstable and diverging)

Accuracy vs stability

Accuracy: how close is x_j to the true solution $x(t_j)$?

Stability: does the numerical update keep the solution bounded / qualitatively reasonable?

- A method can be accurate for small h but unstable for large h (like Forward Euler).
- Stability often creates a **hard limit** on h for explicit methods.

Roadmap for today

1. Use a simple **real** test equation for decay/growth
2. Write each method as $x_{j+1} = g(h)\, x_j$
3. Translate "stable" into conditions on $g(h)$
4. Analyze:
   - Forward Euler
   - Explicit Midpoint
   - Backward Euler
5. Connect the math to oscillation and blow-up

Our main focus: exponential decay

For many physical models (cooling, damping, relaxation), we have:

$$x'(t) = -\lambda x(t), \qquad \lambda > 0$$

Exact:

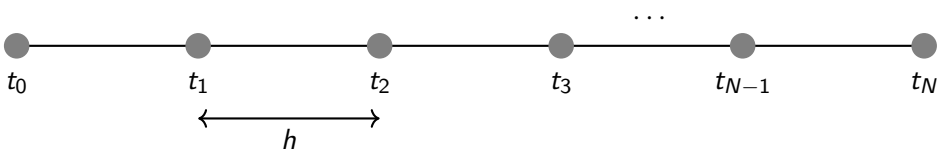
$$x(t) = x_0 e^{-\lambda t} \to 0$$

Stability question: does the numerical method also drive $x_N \to 0$ (for large enough N) without weird behaviors?

Discrete time grid and notation

We compute at equally spaced times:

$$t_j = t_0 + jh, \qquad n = 0, 1, 2, \ldots, N$$



and store:

$$x_j \approx x(t_j)$$

All three methods (FE, Midpoint, BE) are **one-step methods**:

$$x_{j+1} = \Phi(h, x_j)$$

For the test equation, each method becomes a simple *multiplication rule*.

Forward and Backward Euler for exponential decay

Consider:

$$x'(t) = -\lambda x(t) \quad \Rightarrow \quad f(t,x) = -\lambda x, \quad \lambda > 0$$

Forward Euler vs Backward Euler:

$$\text{Forward Euler: } x_{j+1} = (1-\lambda h)x_j, \qquad \text{Backward Euler: } x_{j+1} = \frac{1}{1+\lambda h}x_j$$

- For Forward Euler: $|1-\lambda h| \gtrless 1$, so the solution may be stable or unstable
- For Backward Euler: $0 < \left|\frac{1}{1+\lambda h}\right| < 1$ for any value of h, so the solution is always decaying (hence always stable)

Key idea: amplification factor

For the test equation, many methods reduce to:

$$x_{j+1} = g\, x_j$$

where the factor g depends on h and λ.

After N steps:

$$x_N = g^N x_0$$

So long-time behavior depends on $|g|$, as we will see next.

Stability criteria

For decay problems (true solution goes to 0), we want:

$$|g| < 1$$

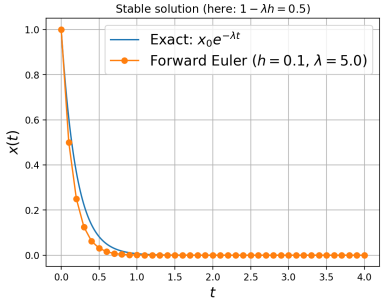
so that $|x_n| = |g|^n |x_0| \to 0$.

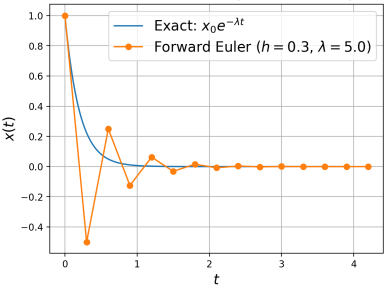
Two types of stable behavior:

- **Monotone decay:** $0 < g < 1$ (stays positive if $x_0 > 0$)
- **Oscillatory decay:** $-1 < g < 0$ (sign flips each step)

Unstable:

$$|g| > 1 \quad \Rightarrow \quad |x_N| \to \infty$$





Forward Euler update (review)

Forward Euler:

$$x_{j+1} = x_j + hf(t_j, x_j)$$

For $x' = -\lambda x$:

$$x_{j+1} = x_j - h\lambda x_j$$

Factor it to identify the amplification g.

Forward Euler amplification factor for exponential decay

$$x_{j+1} = (1 - h\lambda)x_j$$

So:

$$g_{\text{FE}} = 1 - h\lambda$$

Stability requires:

$$|1 - \lambda h| < 1$$

Now, we need to solve this inequality.

Forward Euler stability bound for exponential decay

We want:

$$|1 - \lambda h| < 1$$

Equivalent to:

$$-1 < 1 - \lambda h < 1$$

Subtract 1:

$$-2 < -\lambda h < 0$$

Multiply by -1 (flip inequalities):

$$0 < \lambda h < 2$$

Forward Euler stability condition for decay:

$$0 < h < \frac{2}{\lambda}$$

Forward Euler: three regimes for exponential decay

Recall:

$$x_{j+1} = (1 - \lambda h)x_j, \qquad g = 1 - \lambda h$$

- **Stable & monotone:** $0 < g < 1 \implies (0 < \lambda h < 1)$
- **Stable but sign-flipping:** $-1 < g < 0 \implies (1 < \lambda h < 2)$
- **Unstable (diverges):** $|g| > 1 \implies (\lambda h > 2)$

These explain the oscillation and blow-up seen in Lecture 5.

Why sign flips are "non-physical" in many models

If $g < 0$, then the numerical update forces:

$$x_1 = gx_0, \quad x_2 = g^2 x_0, \quad x_3 = g^3 x_0, \ \ldots$$

so the sign alternates at every step.
For quantities like:

- concentration
- mass
- population
- temperature above ambient (after shifting)

negative values are not physically meaningful, even if the magnitude decays.

Key point: numerical stability does not guarantee physical realism.

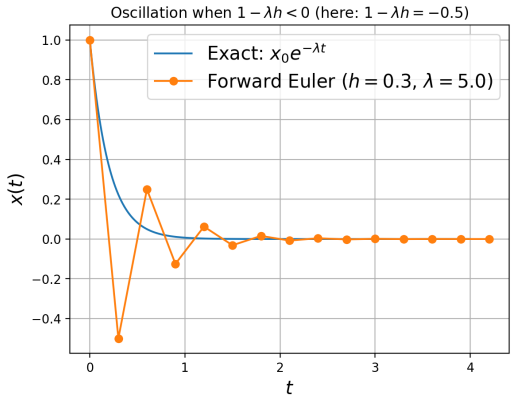


Figure: Forward Euler with $-1 < g < 0$: stable but oscillatory

Why divergence happens (numerical instability)

If $|g| > 1$, then:

$$|x_{j+1}| = |g|\,|x_j| \quad \Rightarrow \quad |x_N| = |g|^N |x_0| \to \infty$$

Even though the true solution decays, the numerical method grows (which is undesirable).

Interpretation: the method amplifies errors or modes instead of damping them.

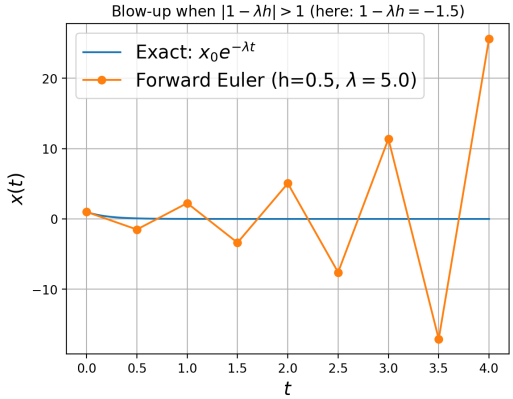


Figure: Forward Euler with $|g| > 1$: oscillatory and unstable

Explicit Midpoint update

Explicit Midpoint:

$$x_{j+1} = x_j + h\, f\left(t_j + \frac{h}{2},\ x_{j+\frac{1}{2}}\right)$$

where

$$x_{j+\frac{1}{2}} = x_j + \frac{h}{2} f(t_j, x_j)$$

Interpretation:

- take a half-step using slope at (t_j, x_j)
- evaluate slope at the midpoint
- use that midpoint slope for the full update

Midpoint method on $x' = -\lambda x$: compute the factor

For $f(t, x) = -\lambda x$,

Half-step predictor:

$$x_{j+\frac{1}{2}} = x_j - \frac{h}{2}\lambda x_j = \left(1 - \frac{h\lambda}{2}\right) x_j$$

Full step:

$$x_{j+1} = x_j - h\lambda x_{j+\frac{1}{2}}$$

Substitute the midpoint value and simplify.

Explicit Midpoint amplification factor for exponential decay

Substitute:

$$x_{j+1} = x_j - h\lambda \left(1 - \frac{h\lambda}{2}\right) x_j$$

So:

$$x_{j+1} = \left(1 - h\lambda + \frac{(h\lambda)^2}{2}\right) x_j$$

Therefore the amplification factor is:

$$g_{\mathrm{EM}} = 1 - h\lambda + \frac{(h\lambda)^2}{2}$$

Explicit Midpoint stability condition for exponential decay

For decay:

$$x_{j+1} = g_{\text{EM}} x_j, \qquad g_{\text{EM}} = 1 - \lambda h + \frac{(\lambda h)^2}{2}$$

We require:

$$|g_{\text{EM}}| < 1 \quad \Leftrightarrow \quad -1 < 1 - \lambda h + \frac{(\lambda h)^2}{2} < 1$$

This produces an interval of stable step sizes (still a step-size restriction).

Key message: Explicit Midpoint is more accurate (2nd order) than Forward/Backward Euler, but it is still conditionally stable.

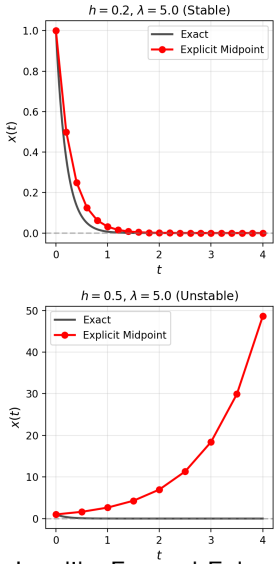


Figure: Just like Forward Euler, Explicit Midpoint is also **conditionally stable**.

Midpoint gives better accuracy per step (intuition)

Midpoint often tracks the exact decay curve better for moderate h because it uses a slope closer to the middle of the interval.

However:

- if h is too large, explicit midpoint can still become unstable
- "more accurate" does not mean "always stable"

Takeaway: order (accuracy) and stability are different properties.

Backward Euler update (implicit)

Backward Euler:

$$x_{j+1} = x_j + hf(t_{j+1}, x_{j+1})$$

For $x' = -\lambda x$:

$$x_{j+1} = x_j - h\lambda x_{j+1}$$

Because x_{j+1} appears on both sides, we solve for it.

Backward Euler amplification factor for exponential decay

Start with:

$$x_{j+1} = x_j - h\lambda x_{j+1}$$

Rearrange:

$$(1 + h\lambda)x_{j+1} = x_j$$

So:

$$x_{j+1} = \frac{1}{1 + h\lambda} x_j$$

Thus:

$$g_{\text{BE}} = \frac{1}{1 + h\lambda}$$

Backward Euler stability for decay: always stable

For $\lambda > 0$ and any $h > 0$:

$$g_{\mathrm{BE}} = \frac{1}{1 + \lambda h}$$

Then:

$$0 < g_{\mathrm{BE}} < 1$$

So:

- always decays (no blow-up)
- never changes sign (no sign flips)

Conclusion: Backward Euler is stable for decay for any step size $h > 0$.

But stability is not accuracy (important)

Backward Euler remains stable even for very large h.

But if h is huge, the numerical curve can be a poor approximation of the true solution.

Stability: does not explode.

Accuracy: close to the exact solution.

We usually want both: pick h small enough for accuracy (even if BE is stable).

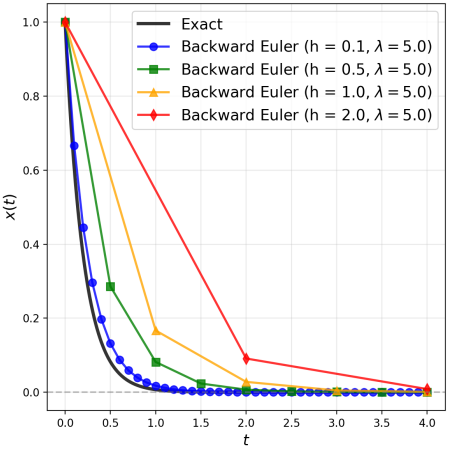


Figure: Backward Euler is **unconditionally stable**, but may not be accurate for large values of h

Summary: Amplification factors for $x' = -\lambda x$

For the decay model $x' = -\lambda x$:

$$x_{j+1} = g\, x_j$$

with,

$$\text{Forward Euler: } g_{\text{FE}} = 1 - \lambda h$$

$$\text{Explicit Midpoint: } g_{\text{EM}} = 1 - \lambda h + \frac{(\lambda h)^2}{2}$$

$$\text{Backward Euler: } g_{\text{BE}} = \frac{1}{1 + \lambda h}$$

Stability conclusions for exponential decay

- **Forward Euler:** stable only if $0 < \lambda h < 2$
- **Explicit Midpoint:** stable only for a bounded range of h (still conditional, **try solving this to find the exact bound on the value of h for stability**)
- **Backward Euler:** stable for all $h > 0$

Main point: these methods have step-size limits that depend on the decay rate λ.

If λ is large (fast decay), then h must be very small for these methods.

A practical interpretation of λh

The quantity λh compares:

- h: your step size
- $1/\lambda$: the natural time scale of decay

Rule of thumb for exponential decay:

$$\lambda h \ll 1$$

usually gives:

- stable updates (for explicit methods)
- good accuracy

If λh is not small, expect artifacts (oscillation/blow-up for Forward Euler/Explicit Midpoint or large errors for all methods).

A simple "engineering checklist"

For a decay ODE that behaves like $x' = -\lambda x$:

- compute λh
- for Forward Euler, ensure $\lambda h < 2$ (and preferably $\alpha h \ll 1$)
- if λ is large, either reduce h or consider an implicit method (we will cover more implicit methods in future)

Takeaway: stability constraints often come from the fastest time scale in the model.

Misconception: "stable" means "correct"

A stable method can still be inaccurate.

Example:

- Backward Euler is stable for any h
- but a very large h can severely distort the true curve

So we choose h based on:

- stability (hard constraint for explicit methods)
- accuracy (quality requirement for all methods)

Misconception: higher order always means more stable

Explicit Midpoint is 2nd order, but it is still explicit.

So it still has a stability restriction (a limited range of h).

Key point: higher order usually improves accuracy per step, but stability depends on the update structure, not just the order.

Big takeaway (what to remember)

- Use the test equation $x' = \lambda x$ to analyze stability
- For decay $x' = -\alpha x$, methods become $x_{n+1} = g x_n$
- Stability depends on $|g|$:
  - $|g| < 1$ stable, $|g| > 1$ unstable
  - $g < 0$ implies sign-flip oscillations
- Forward Euler and Midpoint are **conditionally stable**
- Backward Euler is **stable for decay for any** $h > 0$