U0.T5 — Numerical ODE Solvers

Flow-Based Generative Models · UFRJ · 2026.2

01 Why solvers, why now

From U2 on, sampling *is* solving

$$\frac{\mathrm{d}x}{\mathrm{d}t} = u_t(x), \qquad x(0) = x_0 \sim \mathcal{N}(0, I_d)$$

Draw noise at $t = 0$. Follow the field. Whatever you reach at $t = 1$ is your sample.

The network is the **right-hand side**. Today is about everything to the left of the equals sign.

The word for the whole session

DEFINITION (NFE)

NFE = number of function evaluations: how many times a solver calls f while integrating.

$$\text{cost of one sample} \;=\; \text{NFE} \times \text{(one forward pass)}$$

Every evaluation of u_t^θ is a forward pass through a neural network.

By December you will discuss papers whose entire contribution is reducing NFE.

02 Euler and the anatomy of error

The problem, and the convention

DEFINITION (INITIAL VALUE PROBLEM)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(t, x(t)), \qquad x(t_0) = x_0$$

CODE CONTRACT (BINDING COURSE-WIDE)

Right-hand sides are **`f(t, x)` — time first.** Every solver **returns its NFE.**

Existence and uniqueness need f Lipschitz in x — that is Picard–Lindelöf, stated properly in U2.T1.

Euler: follow the tangent

At x_n the equation gives you the velocity. Pretend it holds for a time h.

DEFINITION (EXPLICIT EULER)

$$x_{n+1} = x_n + h\,f(t_n, x_n) \qquad \textbf{1 NFE per step}$$

A polygon inscribed in a curve.

Two errors, not one

DEFINITION (LOCAL TRUNCATION ERROR)

The error of **one** step, started from the **exact** point:

$$\tau_{n+1} = x(t_{n+1}) - \left[x(t_n) + hf(t_n, x(t_n)) \right]$$

DEFINITION (GLOBAL ERROR)

The error that actually reaches you, after all N steps:

$$e_N = x(t_N) - x_N$$

Order p: local $O(h^{p+1})$, global $O(h^p)$. **One power is lost to accumulation.**

The proof, part 1: local error is $O(h^2)$

Taylor-expand the exact solution about t_n:

$$x(t_{n+1}) = x(t_n) + h\,\dot{x}(t_n) + \frac{h^2}{2}\ddot{x}(\xi)$$

The ODE says $\dot{x}(t_n) = f(t_n, x(t_n))$ — the first two terms **are** the Euler step.

$$\Longrightarrow \quad \tau_{n+1} = \frac{h^2}{2}\,\ddot{x}(\xi) \;=\; O(h^2)$$

The proof, part 2: accumulation costs one power

Let $e_n = x(t_n) - x_n$. Subtract the two updates and take norms:

$$\|e_{n+1}\| \;\le\; (1 + hL)\,\|e_n\| \;+\; \tfrac{1}{2}Mh^2$$

L = Lipschitz constant of f in x. Unroll from $e_0 = 0$:

$$\|e_N\| \;\le\; \frac{Mh}{2L}\Big[\,e^{L(t_1 - t_0)} - 1\Big] \;=\; O(h)$$

Read the constant: it grows like $e^{L(t_1 - t_0)}$. Order 1 is a statement about a *slope*, not a promise of a small number.

The picture to keep

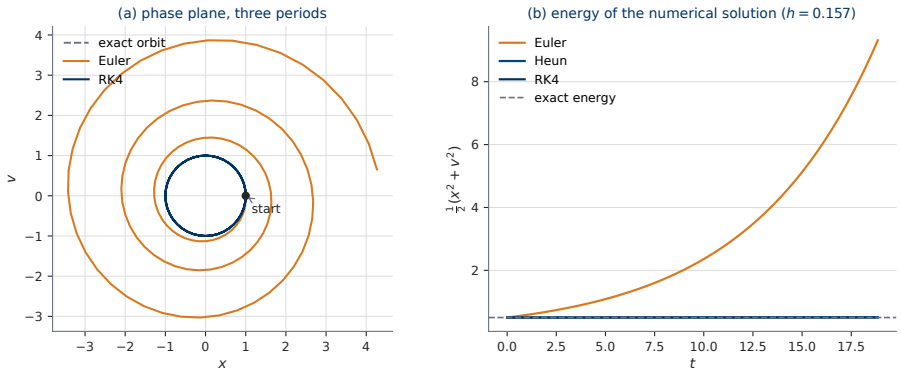
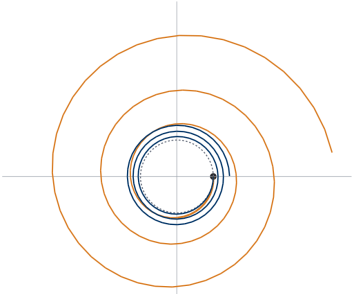


Figure 1: Explicit Euler on the harmonic oscillator, $h = 2\pi/40$. Left: three periods in the phase plane — the exact orbit is the unit circle; Euler ends at radius 4.32. Right: energy against time.

The same picture, one step at a time

Explicit Euler on the harmonic oscillator, step by step



```
final radius, measured
  h          4.3166
  h / 4      1.4475
  exact      1.0000
```

measured growth factor: 1.024674 (h = 0.15708)

$$E_{n+1} = (1 + h^2)\, E_n, \qquad 1 + h^2 > 1 \ \text{ for every } h > 0$$

Animated in the web deck — final frame shown.

03 Heun and RK4

Heun: predict, then correct

Euler uses the velocity at the *start* for the whole step. Average the two ends instead — and guess the far end with Euler.

DEFINITION (HEUN'S METHOD)

$$k_1 = f(t_n, x_n), \qquad k_2 = f(t_n + h,\ x_n + hk_1)$$

$$x_{n+1} = x_n + \tfrac{h}{2}(k_1 + k_2) \qquad \textbf{2 NFE per step, order 2}$$

The first NFE-vs-accuracy trade of the course: one extra evaluation, one extra order.

RK4: the workhorse default

DEFINITION (CLASSICAL RUNGE–KUTTA)

$$k_1 = f(t_n,\ x_n), \qquad k_2 = f\left(t_n + \tfrac{h}{2},\ x_n + \tfrac{h}{2}k_1\right),$$
$$k_3 = f\left(t_n + \tfrac{h}{2},\ x_n + \tfrac{h}{2}k_2\right), \qquad k_4 = f(t_n + h,\ x_n + hk_3),$$
$$x_{n+1} = x_n + \tfrac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right). \qquad \textbf{4 NFE, order 4}$$

The weights $\frac{1}{6}, \frac{2}{6}, \frac{2}{6}, \frac{1}{6}$ are **Simpson's rule** — and for an f that does not depend on x, RK4 *is* Simpson's rule.

The Runge–Kutta idea, and where we stop

The idea, in one sentence: sample the field at trial points *inside* the step, then combine the samples with fixed weights so the Taylor terms cancel.

The coefficients live in a **Butcher tableau**. The theory of which tableaux reach which order is deep, well developed, and **explicitly declined here**.

> *We are consumers of tableaux, not designers of them.*

You need three methods and three words: *stage*, *order*, *embedded pair*.

Order and cost

Method	NFE / step	Order p	Global error
Euler	1	1	$O(h)$
Heun	2	2	$O(h^2)$
RK4	4	4	$O(h^4)$

Order alone settles nothing: an RK4 step costs **four** Euler steps.

The honest axis is NFE, not step count.

Error against h, and against cost

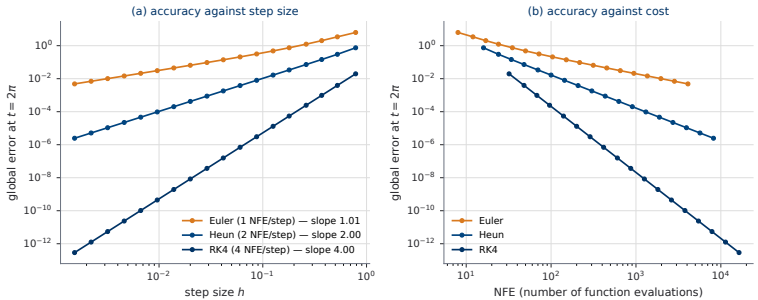


Figure 2: Global error at $t = 2\pi$. Left: against step size, log-log — fitted slopes 1.01, 2.00, 4.00. Right: the same errors against NFE, the honest cost axis.

NFE to reach a global error of 10^{-6}**:** Euler 1.9×10^7 · Heun 1.3×10^4 · RK4 3.8×10^2

Why this matters later

WHY THIS MATTERS LATER

The names on this slide come back as the names of **samplers**.
"Euler sampling" in a diffusion or Flow Matching paper means exactly $x_{n+1} = x_n + h\, u^{\theta}_{t_n}(x_n)$. The EDM sampler is a **Heun** variant.
When U3.T3 discusses them, you will recognise old friends — and you will already know their NFE per step.

04 Adaptive stepping and tolerances

A fixed step is the wrong tool

A trajectory can be almost straight for most of its length and turn sharply in one short window.

- A step small enough for the sharp part **wastes** evaluations everywhere else.
- A step efficient elsewhere is wrong **exactly where it matters**.

So let the solver choose h as it goes — which means it must estimate its own error, without knowing the answer.

Embedded pairs: two orders, one extra evaluation

Heun already computes k_1 and k_2. But k_1 alone **is** the Euler step:

$$\hat{x}_{n+1} = x_n + hk_1 \ \text{(order 1)}, \qquad x_{n+1} = x_n + \tfrac{h}{2}(k_1 + k_2) \ \text{(order 2)}$$

$$\varepsilon_{n+1} = \left\| x_{n+1} - \hat{x}_{n+1} \right\| \ = \ \text{local error estimate}$$

Written **Heun(2)/Euler(1)**. Propagate the *higher* order — you paid for it.

What `rtol` and `atol` actually mean

DEFINITION (THE ERROR SCALE)

$$\mathrm{sc}_i = \texttt{atol} + \texttt{rtol} \cdot \max(|x_{n,i}|,\ |x_{n+1,i}|), \qquad \mathrm{err} = \sqrt{\frac{1}{d}\sum_i \left(\frac{\varepsilon_{n+1,i}}{\mathrm{sc}_i}\right)^2}$$

Accept the step when $\mathrm{err} \le 1$. Otherwise throw it away and retry smaller.

- **`rtol`** dominates where x is **large** — it asks for significant digits.
- **`atol`** dominates where x is **near zero** — it says how small is “don’t care”.

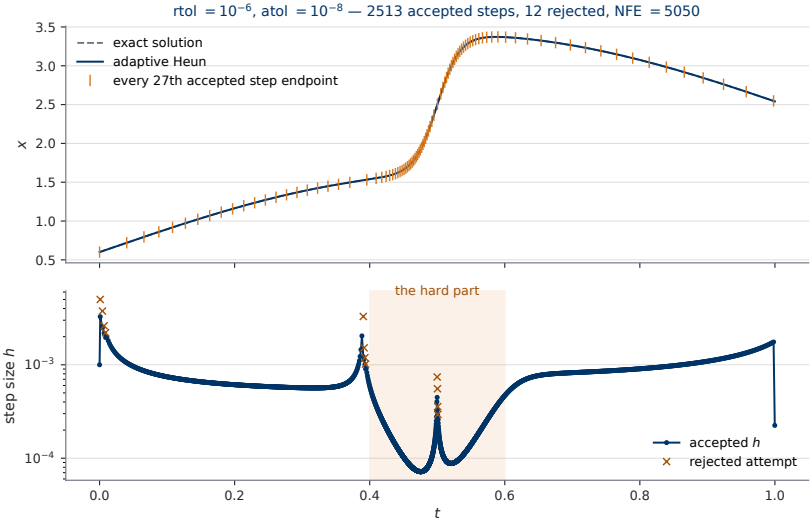
The step-size controller

$$h_{\text{new}} \;=\; h \cdot \min\Big(f_{\max},\; \max\big(f_{\min},\; \eta \cdot \mathrm{err}^{-1/(p+1)}\big)\Big)$$

- The exponent $-1/(p+1)$ is just "solve $\mathrm{err} = 1$" for the local order p.
- $\eta \approx 0.9$ — a **safety factor**, so a marginal step is not immediately rejected.
- $f_{\min} \approx 0.2$, $f_{\max} \approx 5$ — **clips**, so one freak estimate cannot change h by orders of magnitude.

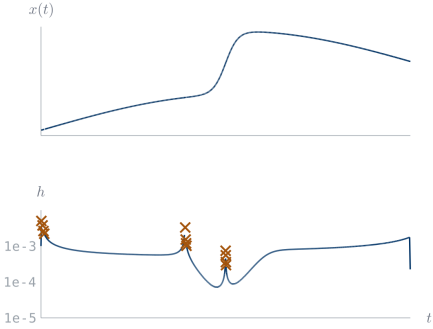
A rejected step is the controller working, not failing. It still costs NFE.

The solver finds the hard part by itself



The hunt, played from the attempt log

The step controller, played from its own attempt log




```
accepted       2513
rejected         12
NFE            5050

h from     7.2e-05
  to       3.3e-03
```

1047 of 2525 attempts — 41% of the work — inside t in [0.45, 0.55]

rejections are rare: 12 in 2525 attempts

Animated in the web deck — final frame shown.

Two failure smells

TOLERANCES TOO LOOSE — THE DANGEROUS ONE

> Fast, small NFE, a smooth plausible trajectory that is **wrong**. Nothing complains.
> **Test:** halve both tolerances and re-run. If the answer moves, it was never converged.

TOLERANCES TOO TIGHT — THE CHEAP ONE

> NFE explodes; the extra digits are below the noise of everything else. You notice immediately.

The solver you will actually call

RK45, also called **Dormand–Prince**: an embedded pair of orders 5 and 4, seven stages.

- the default of `scipy.integrate.solve_ivp`
- `dopri5` in `torchdiffeq` and in `diffrax`

You now know what every one of its arguments means.

With an adaptive solver, NFE becomes data-dependent. In U2 you will watch it grow during training, and it will hurt. In U3 it stops hurting. Understanding why is the course.

05 Stability and stiffness

The other question about h

Block B asked: how accurate is this **as** $h \to 0$?

Block E asks the question that decides whether a method is usable at all:
*For a **given** h, does the numerical solution stay bounded?*

Test equation — a scalar problem, because a linear system diagonalises into copies of it:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \lambda x, \quad \lambda \in \mathbb{C}, \qquad x(t) = x_0 e^{\lambda t}$$

Amplification, and the region

Euler on the test equation:

$$x_{n+1} = (1 + h\lambda)\, x_n \quad \Longrightarrow \quad x_n = (1 + h\lambda)^n x_0$$

DEFINITION (REGION OF ABSOLUTE STABILITY)

$$S = \{\, z = h\lambda \in \mathbb{C} \;:\; |R(z)| \le 1 \,\}$$

Euler $R(z) = 1 + z$; Heun $R(z) = 1 + z + \frac{z^2}{2}$; RK4 through $\frac{z^4}{24}$.

S constrains the **product** $h\lambda$ — so it is a **ceiling on your step size**.

The regions

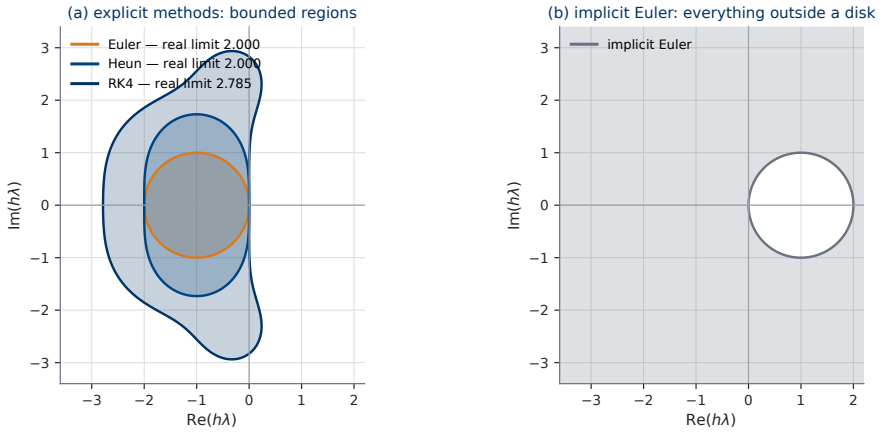


Figure 4: Shaded = stable. Left: the explicit methods, all bounded; real-axis limits 2.000, 2.000, 2.785, located by bisection. Right: implicit Euler — everything outside a disk.

The spiral, explained

The harmonic oscillator has eigenvalues $\lambda = \pm i$ — **purely imaginary**.

The Euler region touches the imaginary axis **only at the origin**. So:

$$|1 + ih| = \sqrt{1 + h^2} > 1 \quad \text{for every } h > 0$$

No positive step size is stable for that problem. The energy law and the stability region are **the same statement twice**.

Implicit methods, named and declined

$$x_{n+1} = x_n + h\,f(t_{n+1},\, x_{n+1}) \qquad \text{(unknown on both sides)}$$

$R(z) = 1/(1 - z)$ — the **whole left half-plane** is stable, at any step size.

That is bought, not given: each step **solves** an equation in x_{n+1}, which for a nonlinear f means an iterative solve with Jacobians.

Our fields are expensive neural networks. The course world is **explicit + adaptive**; implicit solvers are named here and never implemented.

Stiffness, defined by the symptom

DEFINITION (STIFFNESS, OPERATIONAL)

> A problem is **stiff** when the step size is limited by **stability** rather than by **accuracy** — the solver is forced far below the step the answer needs.
> Usual cause: a fast decaying mode that dies quickly, stops affecting the answer, and keeps dictating h long afterwards.

Not a property you read off an equation. A symptom you observe in a solver.

Stiffness, measured

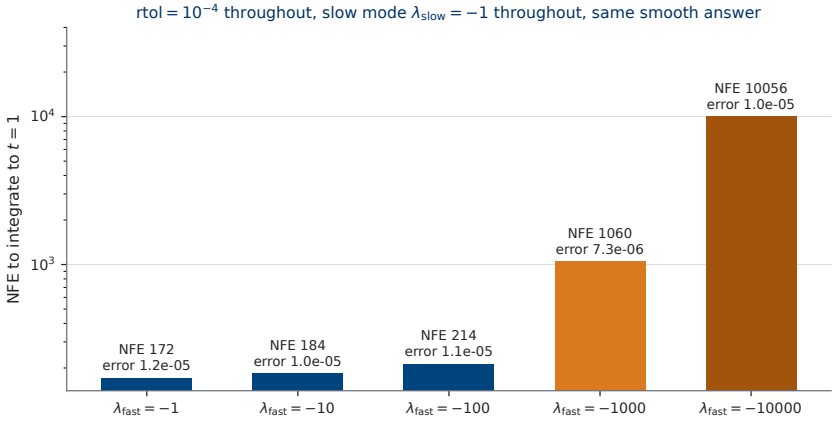


Figure 5: NFE to integrate a 2D linear system to $t = 1$ at rtol = 10^{-4}; slow mode fixed at -1, fast mode swept to -10^4. Each bar is annotated with the achieved error.

An honest scope note

Are learned velocity fields stiff? **It depends on the design**, and this course answers it empirically, not by assertion:

- **U2.L2** — tolerance-sensitivity study on a trained model
- **U3.T4** — *straightness*: making the field easy to integrate

A large part of recent progress in fast sampling is not better solvers. It is fields that are easier to integrate.

06 The equation that moves the density

One field, a whole cloud

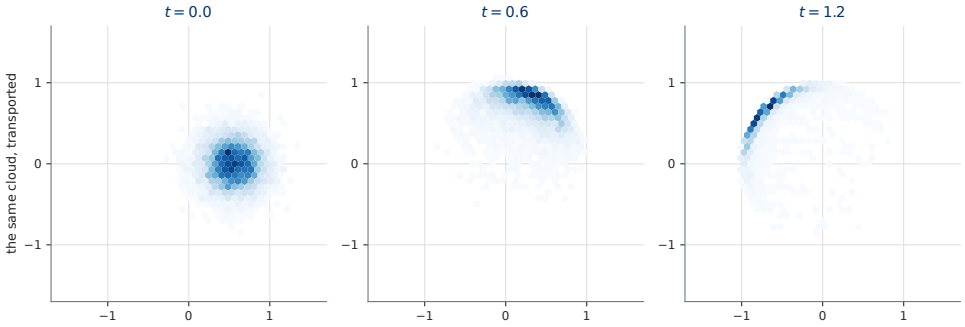


Figure 6: One cloud of initial conditions, transported by a single velocity field, at three times. Every point obeys the same ODE. Nobody moves the cloud as a whole — and yet the density changes shape.

The ODE moves a **point**. What moves the **density**?

Shown, not derived

$$\partial_t p_t + \nabla \cdot (p_t u_t) = 0$$

The continuity equation. Stated properly and derived in U2.T2. It becomes the foundation of everything in U3.T1.

Generative modelling in this course = choosing the density movie, and learning the field that plays it.

07 Close

What this session hands forward

From today	Reappears in
`f(t, x)` time first; NFE returned	every solver, from U0.L5 on
Euler, Heun, RK4	U0.L5 (by hand); U3.T3 (as samplers)
Adaptive stepping, rtol/atol	U2.L1 (assumed); U2.L2
Stability and stiffness	U2.L2; U3.T4 (straightness)
The continuity equation	U2.T2 (derived); U3.T1

These notes are the PESC packet

The lecture notes for this session are written **standalone-first**: complete prose, self-explanatory figures, exercises with answers, everything runs on CPU.

They are the core of the self-study packet PESC students receive at enrollment — which makes them **your best review document too**.

Next session (U0.L5): implement all four solvers from scratch, each returning its NFE; make the log-log plot yourself; then the same toy ODE in JAX.