

U2.T1 — Neural ODEs and the Adjoint Method

Flow-Based Generative Models · UFRJ · 2026.2

01 | From residual steps to a flow

The question on the board

U1.T2 ended on a residual stack with a step size, and on one question:

What does $\log|\det J|$ become in continuous time?

Today pays the **first half** of that debt, and leaves the question where it is:

Today	Not today
residual steps $\rightarrow$ an ODE, honestly the flow exists, is unique, is invertible how to differentiate through a solver	the log-determinant any density any divergence

The two costs, version 2

THE TWO COSTS (V_2) – $U_1.T_2$

Family	Density	Sampling	Trick, and its price
NICE / RealNVP	exact, triangular	direct	coupling; depth
Glow	exact	direct	mixing, multi-scale
MAF	parallel	sequential	AR, data side
IAF	sequential	parallel	AR, source side
Residual flow	direct; series log-det	iterative	contraction, traces

Every row pays with an **architectural constraint**: a coupling, a mixing layer, an autoregressive (AR) ordering, a contraction. What if depth $\rightarrow \infty$?

A residual stack with a step size

Put $h = 1/K$ in front of each residual branch, and let the branch depend on $t_k = kh$:

$$x_{k+1} = x_k + h f(t_k, x_k), \quad k = 0, \dots, K - 1.$$

This is **forward Euler** for $\dot{x} = f(t, x)$, $x(0) = x_0$, $t \in [0, 1]$ — the course's $f(t, x)$, time first. U1.T2 measured that the stack's output converges as $h \rightarrow 0$ (error halves when h halves).

What U1.T2 did not say: what the limit *is*; when it exists; whether it is unique; whether $x_0 \mapsto x(1)$ is invertible; how to differentiate through it. That is today.

The model

DEFINITION (NEURAL ODE; FLOW MAP)

Let $f_\theta: [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ be a network with parameters $\theta \in \mathbb{R}^p$, and suppose

$$\dot{x}(t) = f_\theta(t, x(t)), \quad x(0) = x_0$$

has a unique solution on $[0, 1]$, and so does the same equation started from any $x(t_0) = x$ with $t_0 \in [0, 1]$ (the well-posedness block says when). The **flow map** is $\psi_t(x_0) = x(t)$, with $\psi_0 = \text{id}$; the **Neural ODE** is $x_0 \mapsto \psi_1(x_0)$. For $t_0 < t_1$, $\psi_{t_0 \rightarrow t_1}$ carries the state at t_0 to the state at t_1 .

The model, read three ways

- Depth became the **integration variable**: no layer k , only a time t .
- **One** θ serves the whole interval.
- ψ_t is the course symbol table's symbol (L1 §4). *Chosen with the future in mind: in U3 it is generated by a different kind of field. Today it is only the solution map.*

Worked example: a flow map in closed form

EXAMPLE (THE SCALAR LINEAR FIELD)

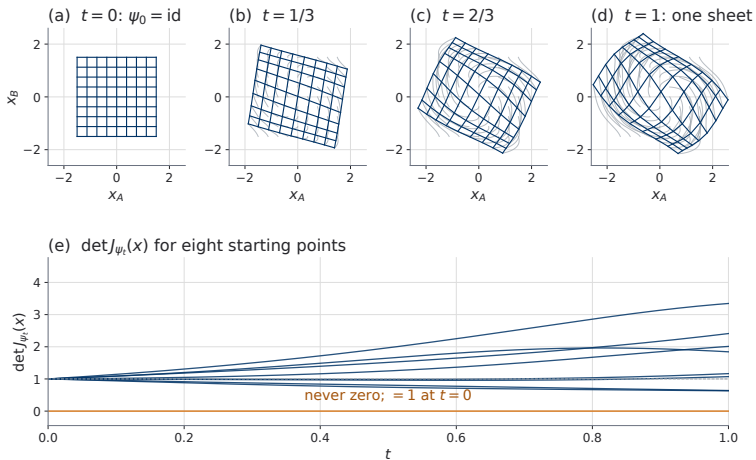
$d = 1$, $f_\theta(t, x) = -\theta x$, $\theta > 0$. Then $\dot{x} = -\theta x$, $x(0) = x_0$ gives $x(t) = x_0 e^{-\theta t}$:

$$\psi_t(x_0) = e^{-\theta t} x_0, \quad \psi_0 = \text{id}, \quad \psi_1(x_0) = e^{-\theta} x_0.$$

The K -block Euler stack computes $(1 - \theta/K)^K x_0 \rightarrow e^{-\theta} x_0$.

Keep this example: in the adjoint block its gradient is computed twice, once directly and once by the adjoint.

The flow map is a family



The ODE limit, honestly

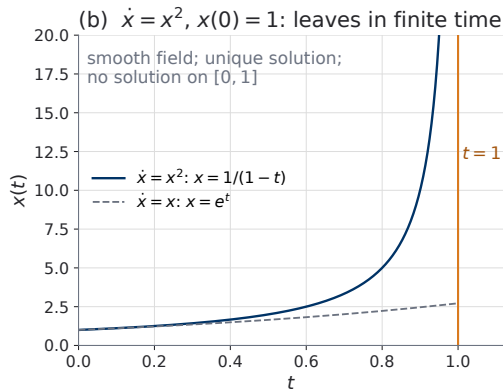
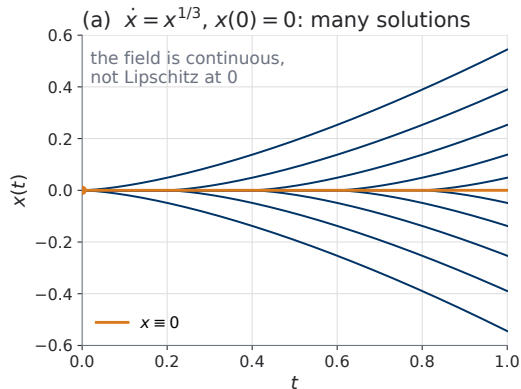
HONEST CAVEAT (WHAT THE LIMIT IS, AND IS NOT)

The Neural ODE with one depth-independent θ is the $h \rightarrow 0$ limit of the Euler stack **in which every block evaluates the same f_θ** . It is **not** the literal continuum limit of a generic ResNet: a ResNet has its own θ_k per block, so its natural continuum object carries a depth-dependent path $t \mapsto \theta(t)$, not a single θ (Massaroli et al. 2020, sec. 3).

The ODE limit is a **disciplined motivation** for the model, not an identity between arbitrary finite ResNets and autonomous-parameter ODEs.

02 | Well-posedness

What could go wrong



Picard–Lindelöf, at the right strength

THEOREM (EXISTENCE AND UNIQUENESS)

Let $f: [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ be continuous.

Local. If f is locally Lipschitz in x , uniformly in t — on every bounded set B , $\|f(t, x) - f(t, y)\| \leq L_B \|x - y\|$ for all t and $x, y \in B$ — then for every x_0 there is $\delta > 0$ and a unique C^1 solution on $[0, \delta]$.

Global — the course's hypothesis. If f is **globally Lipschitz in x , uniformly in t** , then for every x_0 the solution exists, is unique and is C^1 on all of $[0, 1]$. (Locally Lipschitz + linear growth also suffices.)

Panel (a) failed the Lipschitz hypothesis at 0; panel (b) failed linear growth. One hypothesis per failure.

The proof engine: PS1.5, on a different space

Write the problem as a fixed-point equation $x = Tx$ (the map T is in the table), and let $L = \text{Lip}(f)$ in x :

	There — PS1.5 (U1.T2)	Here — Picard–Lindelöf
space	$\mathbb{R}^d$, Euclidean norm	$C([0, 1]; \mathbb{R}^d)$, weighted norm $\sup_t e^{-2Lt} \ x(t)\ $
map	$T_z(x) = z - g(x)$	$(Tx)(t) = x_0 + \int_0^t f(\tau, x(\tau)) d\tau$
contraction	constant $\text{Lip}(g) < 1$	constant $\frac{1}{2}$, bought by the weight e^{-2Lt}
what buys it	an architectural condition	any finite L — nothing asked of its size

The line to remember: the residual block needed $L < 1$; the flow needs $L < \infty$.

The standing regularity contract for U2

STANDING REGULARITY CONTRACT (U2)

Tier 1 — well-posedness. f_θ continuous in (t, x) , globally Lipschitz in x uniformly in t . Enough for existence, uniqueness, invertibility of ψ_t .

Tier 2 — differential identities. In addition, f_θ is C^1 in x and in θ with continuous partials. Enough for the adjoint (today) and for the divergence identities (U2.T2). Then ψ_t is C^1 in x_0 with invertible Jacobian, and $t \mapsto J_{\psi_t}(x_0)$ is continuous.

Course models use tanh / softplus and satisfy both tiers. **ReLU is Lipschitz and not C^1 :** Tier 1 only. The two roles are never conflated in this unit.

Invertibility by well-posedness, not for free

PROPOSITION (THE FLOW MAP IS INVERTIBLE)

Under Tier 1, for $t_0 < t_1$ the map $\psi_{t_0 \rightarrow t_1}$ is a bijection of $\mathbb{R}^d$, and its inverse is obtained by integrating **the same equation backward in time** from t_1 to t_0 ; both directions are continuous (Grönwall), so it is a homeomorphism. Under Tier 2 it is a diffeomorphism.

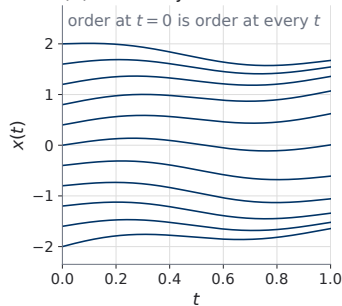
Proof in one line each. Injective: two solutions meeting at time t_1 coincide (uniqueness, run backward). Surjective: solve backward from any y at t_1 . Continuous: Grönwall bounds the spread of two solutions by e^L times their initial spread, in both directions.

The inverse: words, and cost

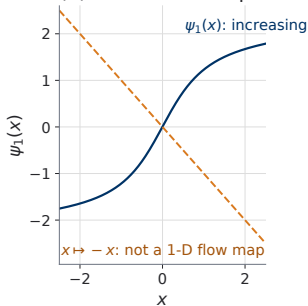
- **Words.** “The flow of $-f$ ” is right only after time reversal: $y(\tau) = x(t_1 - \tau)$ solves $\dot{y} = -f(t_1 - \tau, y)$, a *different* non-autonomous field. “Integrate the same equation backward” is always right.
- **Cost.** No coupling pattern, no triangular Jacobian, no contraction per block — **structurally automatic**. And **not free**: the inverse is an ODE solve, priced in **NFE** (the number of times the solver evaluates f_θ). Both directions of the map pay in that currency.

What a regular flow cannot do in one dimension

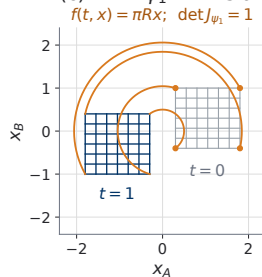
(a) 1-D: trajectories never cross



(b) the time-1 map in 1-D



(c) 2-D: $\psi_1 = -I$ is a rotation



What a regular flow map cannot be

PROPOSITION (CONTINUOUS-FLOW-MAP RESTRICTION)

Let f satisfy Tier 1 and ψ_t be its flow.

(i) $d = 1$: order is preserved. $x_0 < y_0 \Rightarrow \psi_t(x_0) < \psi_t(y_0)$ for all t . So ψ_1 is strictly increasing, and $x \mapsto -x$ is not a 1-D regular flow map.

(ii) Any d : orientation is preserved. Under Tier 2, $\det J_{\psi_t}(x) > 0$ for all x, t : continuous in t , equal to 1 at $t = 0$, never zero. So $x \mapsto -x$ (determinant $(-1)^d$) is not a regular flow map in **odd** d ; in $d = 2$ it is a rotation.

(iii) Orientation is necessary, not a characterization; finer claims are dimension-specific and not made here.

What this restriction is not

- It is **not** U1.T2's topology-obstruction-argument. That one is about supports under a homeomorphism — still true here, unchanged, because ψ_1 is a homeomorphism. Today's is about which maps arise as *flows* at all.
- U3 cites the two **side by side**, never one inside the other.

U3 changes what density path we prescribe, but any regular ODE flow map is still a regular flow map. Hold the thought.

The two costs, version 2, annotated

THE TWO COSTS (V_2 , ANNOTATED – $U_2.T_1$)

Family	Density	Sampling	Trick, and its price
NICE / RealNVP	exact, triangular	direct	coupling; depth
Glow	exact	direct	mixing, multi-scale
MAF	parallel	sequential	AR, data side
IAF	sequential	parallel	AR, source side
Residual flow	direct; series log-det	iterative	contraction, traces
Neural ODE	inverse: ODE solve / NFE; $\log \det J$???	ODE solve / NFE	no inverse layer; Lipschitz field

03 | The adjoint method

The setup, and one letter

Fix x_0 ; x solves $\dot{x} = f_\theta(t, x)$ under Tier 2; the loss reads the endpoint:

$$\ell(\theta) = L(x(1)), \quad L \in C^1. \quad \text{Wanted: } \frac{d\ell}{d\theta} \in \mathbb{R}^{1 \times p}.$$

- Cotangents — the row vectors that multiply Jacobians from the left in reverse mode — are **rows**: the adjoint state is $a^\top(t) \in \mathbb{R}^{1 \times d}$.
- **Goal:** $d\ell/d\theta$ *without storing every solver-internal state*.
- **One letter.** Today $\mathcal{L}$ is **not** a loss. It is the *perturbed functional* of the derivation; the loss is ℓ . They agree in value along trajectories — that is the trick — and are different functions. The roman L here is the loss function, not the Lipschitz constant of the well-posedness block.

Step 0 — the discrete ancestor

RESTATED (VJP COMPOSITION) — PROVED IN THE BOOTCAMP

For $F = f_K \circ \dots \circ f_1$ (K maps), $x^k = f_k(x^{k-1})$, $x^0 = x$, and a cotangent $v^\top$ on the output:

$$\begin{aligned} v_K^\top &= v^\top, & v_{k-1}^\top &= v_k^\top J_{f_k}(x^{k-1}) \quad (k = K, \dots, 1), \\ v_0^\top &= v^\top J_F(x). \end{aligned}$$

Each step is a **vector-Jacobian product (VJP)**: about one f_k -evaluation, no Jacobian formed.

Backprop pushes a cotangent backward through **composed maps**. The adjoint pushes a cotangent backward through **continuously composed maps**.

Step 0, continued — the layers as Euler steps

Let the layers be Euler steps, $f_k(x) = x + hf(t_{k-1}, x)$, so $J_{f_k} = I + h \partial f / \partial x$:

$$v_{k-1}^\top = v_k^\top + h v_k^\top \frac{\partial f}{\partial x}(t_{k-1}, x_{k-1}) \quad \xrightarrow{h \rightarrow 0} \quad \dot{a}^\top = -a^\top \frac{\partial f}{\partial x}.$$

The adjoint state is the running VJP. Whether the recursion is *exactly* the discrete adjoint is PS2.1(g).

Step 1 — the perturbed functional (G2, first half)

Fix θ_0 . For **any** C^1 function $a: [0, 1] \rightarrow \mathbb{R}^d$, chosen once as a function of t alone:

$$\mathcal{L}(\theta) = \ell(\theta) - \int_0^1 a^\top(t) [\dot{x}(t) - f_\theta(t, x(t))] dt.$$

The bracket is **identically zero**, because x solves the equation — for every θ .

So $\mathcal{L} = \ell$ as functions of θ , and $\frac{d\mathcal{L}}{d\theta} = \frac{d\ell}{d\theta}$, **whatever a is**. a is free. We will spend it.

Step 2 — integration by parts (G1)

$$\int_0^1 a^\top \dot{x} dt = a^\top(1)x(1) - a^\top(0)x(0) - \int_0^1 \dot{a}^\top x dt.$$

Stated, not performed: boundary terms plus a sign flip. **Check it — that is PS2.1(b).**
With it,

$$\mathcal{L}(\theta) = L(x(1)) - a^\top(1)x(1) + a^\top(0)x_0 + \int_0^1 [\dot{a}^\top x + a^\top f_\theta(t, x)] dt.$$

Step 3 — differentiate, and collect the sensitivity

Write $S(t) = dx(t)/d\theta \in \mathbb{R}^{d \times p}$ (exists and is continuous under Tier 2), with $S(0) = 0$ because x_0 does not depend on θ . Let $A = \partial f_\theta / \partial x$, $B = \partial f_\theta / \partial \theta$, both along the trajectory.

$$\begin{aligned} \frac{d\mathcal{L}}{d\theta} = & \underbrace{\left[\frac{\partial \mathcal{L}}{\partial x}(x(1)) - a^\top(1) \right] S(1)}_{\text{multiplies the unknown } S(1)} + \underbrace{\int_0^1 [\dot{a}^\top + a^\top A] S dt}_{\text{multiplies the unknown } S(t)} \\ & + \underbrace{\int_0^1 a^\top B dt}_{\text{no sensitivity at all}} \end{aligned}$$

Step 4 — spend the free choice (G2)

Why are we allowed to choose a ? Because Step 1 holds for **every** fixed a . So choose the one that annihilates both sensitivity groups:

$$\dot{a}^\top(t) = -a^\top(t) \frac{\partial f_\theta}{\partial x}(t, x(t)), \quad a^\top(1) = \frac{\partial L}{\partial x}(x(1)).$$

A **linear final-value problem**, solved backward from $t = 1$; it exists, is unique and is C^1 by Picard–Lindelöf (continuous coefficient matrix). With this a , both unknown groups vanish **whatever S is**.

A free choice, spent wisely — the same proof taste as the projection Lemma's session, and no formal connection claimed.

What survives: the adjoint sensitivity theorem

THEOREM (ADJOINT SENSITIVITY, TERMINAL LOSS) – BOX U2.T1-3

Under Tier 2, with x_0 independent of θ and $\ell = L(x(1))$, $L \in C^1$:

$$\frac{d\ell}{d\theta} = \int_0^1 a^\top(t) \frac{\partial f_\theta}{\partial \theta}(t, x(t)) dt,$$
$$\dot{a}^\top = -a^\top \frac{\partial f_\theta}{\partial x}, \quad a^\top(1) = \frac{\partial L}{\partial x}(x(1)).$$

Moreover $a^\top(t) = d\ell/dx(t)$: the sensitivity of the loss to the state at time t — the running VJP (Chen et al. 2018, App. B).

Named, boxed, and on the right board for the rest of the session. The full gap-free derivation is Box U2.T1-3 of the notes; the gaps are PS2.1.

Implementation: the augmented backward solve (G3)

Add one state, the **parameter adjoint** $a_\theta(t) \in \mathbb{R}^p$:

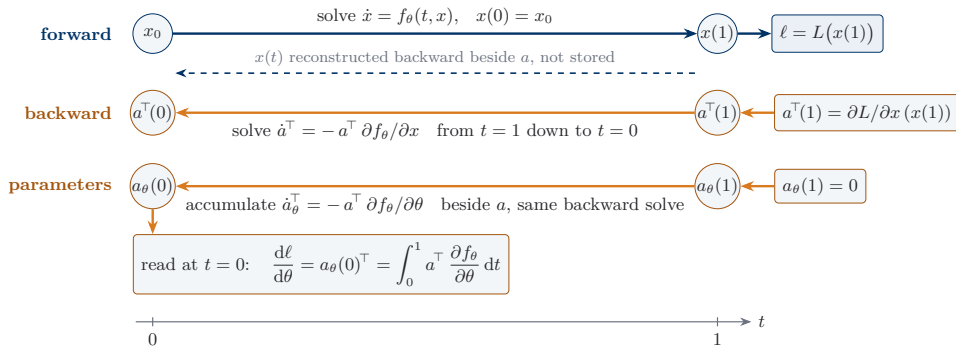
$$\dot{a}_\theta^\top = -a^\top \frac{\partial f_\theta}{\partial \theta}(t, x(t)), \quad a_\theta(1) = 0.$$

Recipe (Chen et al. 2018, Alg. 1): solve (x, a, a_θ) , dimension $2d + p$, **backward from** $t = 1$ **to** 0 , from $(x(1), \partial L / \partial x(x(1))^\top, 0)$ with dynamics $(f_\theta, -A^\top a, -B^\top a)$; read

$$\frac{d\ell}{d\theta} = a_\theta(0)^\top.$$

$x(t)$ is **reconstructed** backward beside a , not stored. Why $a_\theta(0)$ equals the boxed integral is PS2.1(e).

The two passes, on one time axis



Worked example: the scalar field, differentiated twice

EXAMPLE (THE SCALAR FIELD, BOTH WAYS)

$f_\theta(t, x) = -\theta x$ on $\mathbb{R}$, $x(0) = x_0$, $L(x) = \frac{1}{2}x^2$.

Directly. $x(1) = x_0 e^{-\theta}$, so $\ell = \frac{1}{2}x_0^2 e^{-2\theta}$ and $d\ell/d\theta = -x_0^2 e^{-2\theta}$.

By the adjoint. $\partial f/\partial x = -\theta$: $\dot{a} = \theta a$, $a(1) = x_0 e^{-\theta}$, so $a(t) = x_0 e^{\theta(t-2)}$. And $\partial f/\partial \theta = -x_0 e^{-\theta t}$:

$$\int_0^1 a \frac{\partial f}{\partial \theta} dt = -x_0^2 \int_0^1 e^{\theta(t-2)} e^{-\theta t} dt = -x_0^2 e^{-2\theta}. \quad \checkmark$$

And $a(t) = d\ell/dx(t)$: $\ell = \frac{1}{2}x(t)^2 e^{-2\theta(1-t)}$ has derivative $x(t)e^{-2\theta(1-t)} = x_0 e^{\theta(t-2)}$. The adjoint is the running sensitivity.

The punchline, said aloud

For a fixed number of requested output times, the continuous adjoint stores $x(1)$, a , a_θ and the solver's current step: $O(1)$ **state with respect to the number of solver steps**. Direct backprop stores every intermediate state the solver produced.

The price, both halves:

1. **another numerical solve**, of dimension $2d + p$, with its own NFE;
2. an **optimize-then-discretize gradient**: an approximation of the exact solution's gradient, not the exact gradient of the discretized forward map.

Bonus: the generalized adjoint (no proof)

REMARK (INTEGRAL LOSS) – PROOF IS PS2.1(F)

For $\ell = L(x(1)) + \int_0^1 l(t, x(t), \theta) dt$, $l \in C^1$:

$$\dot{a}^\top = -a^\top \frac{\partial f_\theta}{\partial x} - \frac{\partial l}{\partial x}, \quad \frac{d\ell}{d\theta} = \int_0^1 \left[a^\top \frac{\partial f_\theta}{\partial \theta} + \frac{\partial l}{\partial \theta} \right] dt.$$

Proposition 1 of Massaroli et al. (2020) proves the case $l = l(t, x)$, with no explicit θ ; the **direct** $\partial l / \partial \theta$ **term** is what appears when the integrand depends on the parameters — a regularizer on the learned field itself (U2.T2 meets one). “Integral loss” is not merely an extra state derivative.

04 | Backprop through the solver, or the adjoint

The two strategies

	Memory	Compute	Correctness
Direct backprop through the solver (discretize → optimize)	every stored solver state; grows with the steps	forward solve + one reverse sweep of VJPs	for a fixed-step solver: the exact gradient of the executed discrete map
Continuous adjoint (optimize → discretize)	$O(1)$ in the steps, at a fixed number of requested output times	a second solve, dimension $2d + p$, each RHS call an f_θ plus VJPs	an approximation of the exact-solution gradient; two discretizations; reconstruction can drift

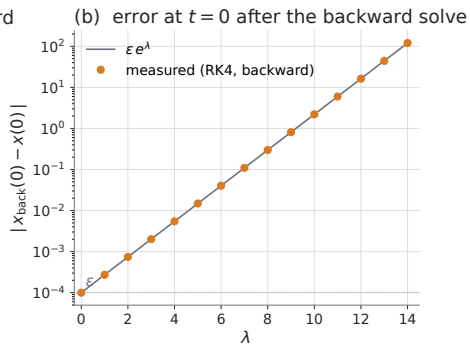
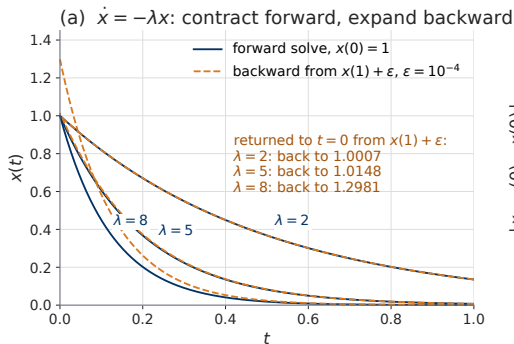
Correctness, read slowly

Discretize, then optimize. A fixed-step solver computes a specific composition of arithmetic; autodiff differentiates *that*. Exact for the function the computer evaluated — not for the exact solution. For an **adaptive** solver the step controller's decisions are treated as constants: a statement about the *executed trace*, locally valid, not globally smooth.

Optimize, then discretize. Derive the gradient of the *exact* solution's loss (the theorem), then discretize the adjoint equation. Close to the true gradient when both solves are accurate — and **not** the gradient of anything the forward solver computed.

Neither is “the correct one”: exact for the wrong function, or approximate for the right one. Which is better is empirical — U2.L1 asks it with a table.

The asterisk under invertibility



RK4 is the fourth-order Runge–Kutta solver; $\varepsilon = 10^{-4}$ is injected at $t = 1$.

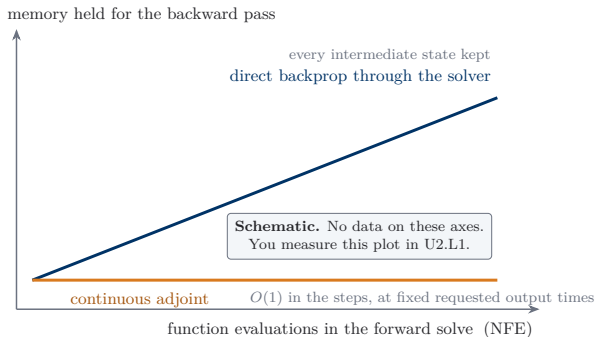
Mitigations, named and not oversold

- **Tighter tolerances** on the backward solve — costs NFE.
- **Direct backpropagation** when the problem is small enough to store.
- **Checkpointing / hybrid schemes**: store a few forward states, reconstruct only between them — bounded reconstruction length, in systems that support it.

Libraries, stated carefully: torchdiffeq (Chen 2018) offers the continuous adjoint as `odeint_adjoint`. Diffrax (Kidger 2021) makes **recursive checkpointing** its default and offers the continuous adjoint as a contrast — the contrast U2.L1 uses.

None of these is a theorem. Each is an engineering choice with a bill. Beyond the course: (Kidger 2022, sec. 5.1).

The plot you will make



In U2.L1 you produce the memory-against-NFE plot yourselves, on a small classifier where nothing hurts yet, and compare it to this schematic. The data is yours to make.

05 | **close**

PS2.1, assigned at the board (1 of 2)

Setting: $\dot{x} = f_\theta(t, x)$, $x(0) = x_0$, $\ell = L(x(1))$, f_θ continuously differentiable, Lipschitz in x .

- **(a)** Define the perturbed functional $\mathcal{L}$ with multiplier $a(t)$ and justify $d\ell/d\theta = d\mathcal{L}/d\theta$ for every choice of a . [= G2, part 1]
- **(b)** Carry out the integration by parts, tracking boundary terms at $t = 0$ and $t = 1$ explicitly. [= G1]
- **(c)** Differentiate $\mathcal{L}$ w.r.t. θ and collect all terms containing $dx/d\theta$. Explain why $dx(0)/d\theta = 0$.
- **(d)** Exhibit the choice of final-value problem for $a(t)$ that annihilates every $dx/d\theta$ term; conclude the adjoint sensitivity theorem. [= G2, part 2]

PS2.1, assigned at the board (2 of 2)

- **(e)** Derive the augmented-state implementation: show that $a_\theta(o)$ as defined in lecture equals the boxed integral. [= G3]
- **(f)** *Bonus.* Extend (a)–(d) to $\ell = L(x(1)) + \int_0^1 l(t, x(t), \theta) dt$. Recover both the $-\partial l / \partial x$ term in the adjoint ODE and the direct $\partial l / \partial \theta$ contribution to $d\ell / d\theta$ (Massaroli et al. 2020, Prop. 1); attached PDF only after attempting.
- **(g)** *Bonus.* Euler-discretize the forward ODE with step h and show the adjoint recursion at the discrete points is exactly reverse-mode autodiff through the unrolled steps (the VJP identity from lecture). Conclude in one sentence in what precise sense “the adjoint is backprop’s continuum limit.”

The board at the close

Left board	Right board	Top right
the two costs, v_2 annotated: inverse resolved , runtime ODE solve / NFE , log-determinant ??? — circled	the adjoint sensitivity theorem, boxed; PS2.1 (a)–(g) with their G-numbers	What does $\log \det J$ become in continuous time? — not erased

We can now build continuous-depth invertible maps and differentiate them without storing every solver step. What we cannot yet do is account for their density change cheaply. The lab measures the numerical bill; U2.T2 finds out what $\log|\det J|$ became.

Reading the board

On the board	Status
Residual steps $\rightarrow$ an ODE Existence, uniqueness	Made precise , with the honest caveat Yes, under a Lipschitz field — two tiers recorded
Invertibility	Structurally automatic; computationally an ODE solve — orientation preserved
Differentiating through a solver	The adjoint sensitivity theorem — gaps G1–G3 = PS2.1
$\log \det J $ in continuous time	Open. U2.T2

Next session: the lab (U2.L1), solvers and adjoints in practice. The theory session after it (U2.T2) opens on the circled cell.

References

- Chen, Ricky T. Q. 2018. “torchdiffeq: ODE Solvers and Adjoint Sensitivity Analysis in PyTorch.” <https://github.com/rtqichen/torchdiffeq>.
- Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. “Neural Ordinary Differential Equations.” In *Advances in Neural Information Processing Systems*. <https://arxiv.org/abs/1806.07366>.
- Kidger, Patrick. 2021. “DiffraX: Numerical Differential Equation Solvers in JAX.” <https://github.com/patrick-kidger/diffrax>.
- . 2022. “On Neural Differential Equations.” PhD thesis, University of Oxford. <https://arxiv.org/abs/2202.02435>.
- Massaroli, Stefano, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. 2020. “Dissecting Neural ODEs.” In *Advances in Neural Information Processing Systems*. <https://arxiv.org/abs/2002.08071>.