vZero Neural ODE

From handwritten ideas to a working mental model

1. Learn a rate of change
2. Integrate it into a trajectory
3. Check physics, accuracy and cost

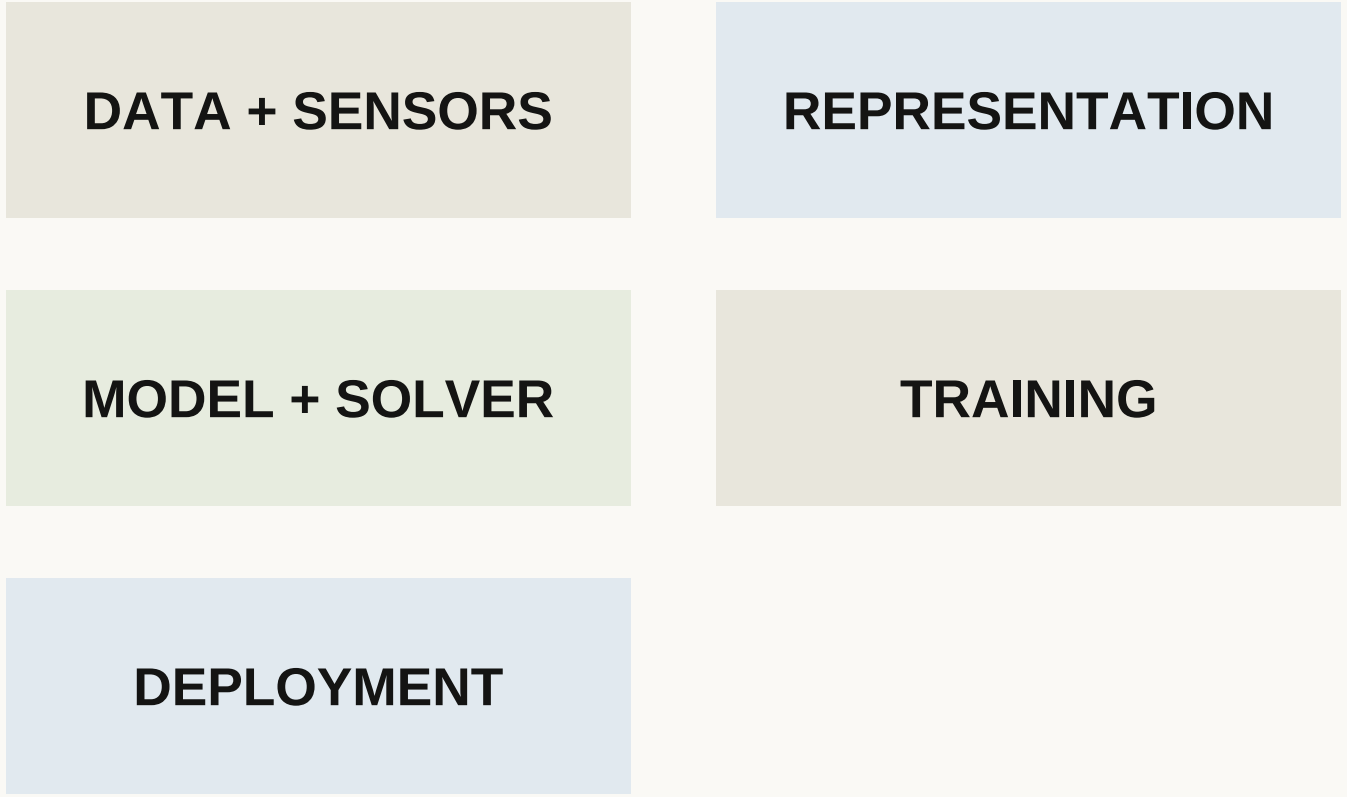


A separate explanation of the vZero Neural ODE ideas

Five jobs, five different questions

A useful method belongs somewhere in this pipeline.

1. Data: what was observed?
2. Dynamics + solver: how does it evolve?
3. Training + deployment: how do we fit and use it?



A state is a snapshot

A trajectory is a sequence of states indexed by time.

1. Chemistry: species amounts and temperature
2. Robot: joint angles and velocities
3. Keep time and units attached

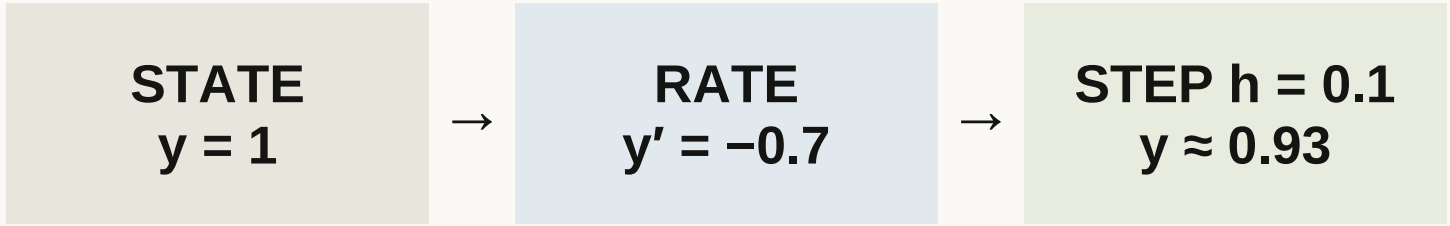


Snapshot ≠ rate ≠ trajectory

A rate is not the next state

The network supplies a slope; the solver accumulates changes.

1. State: where am I now?
2. Rate: how fast am I moving?
3. Solver: where will I be later?



The network supplies a slope; the solver accumulates changes.

What does the neural network learn?

An adjustable function maps the current state to its rate.

1. Weights are adjustable knobs
2. Training compares predicted trajectories with targets
3. Arrhenius is a familiar nonlinear rate law



The same learned function is evaluated many times

Where this appears in vZero

The inspected implementation exposes these decisions explicitly.

1. Normalize the initial state
2. Integrate ChemistryODEFunc
3. Return normalized or decoded output



The inspected implementation exposes these decisions explicitly.

A complete small learning loop

Generate a trajectory, measure its error, update the rate model.

Known target: $y(t) = y_0 \exp(-0.7t)$

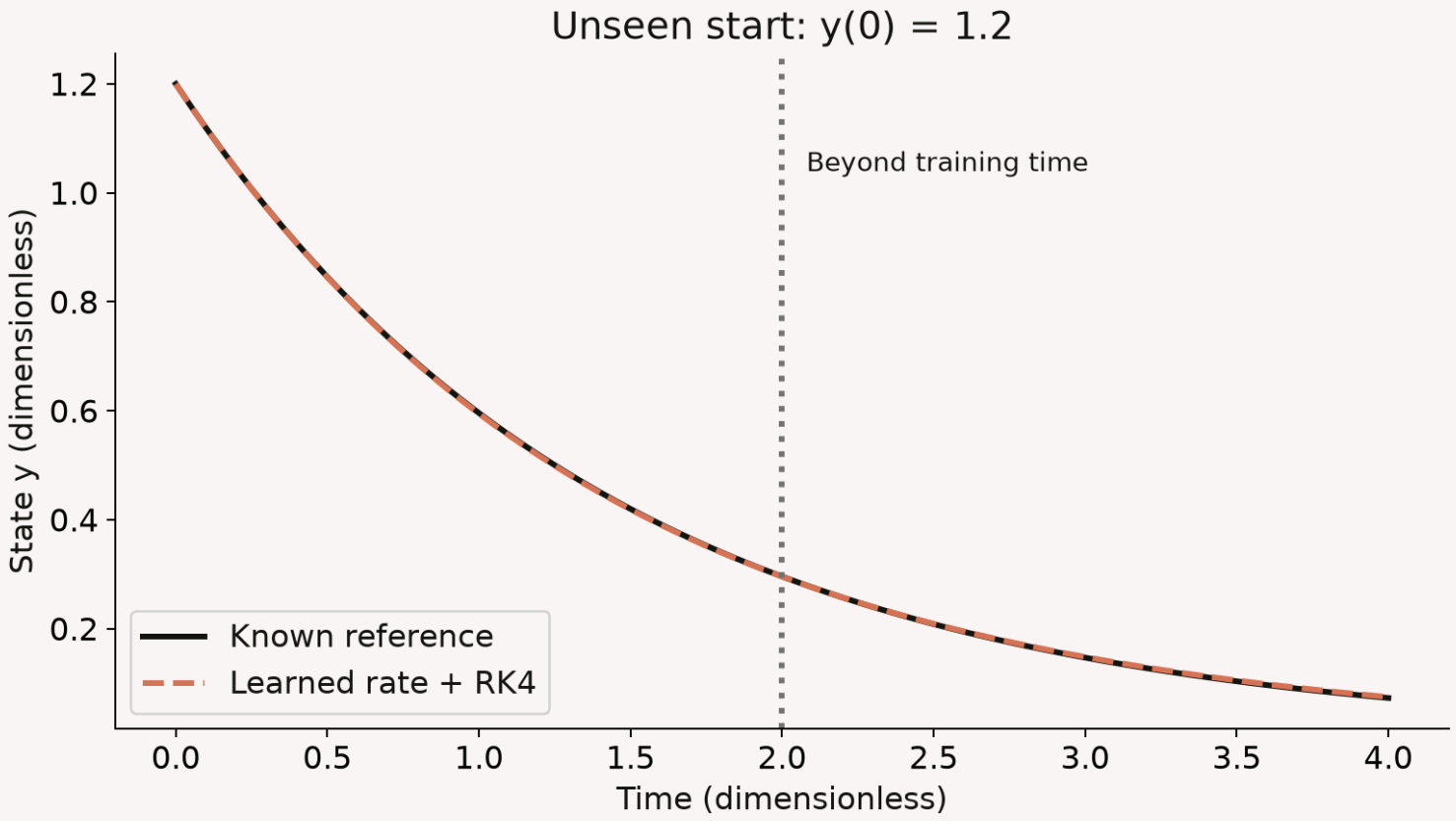
```
pred = odeint(
rate, y0, t, method="rk4")
loss = ((pred - target)**2).mean()
optimizer.zero_grad()
loss.backward()
optimizer.step()
```



Test a rollout beyond the training window

A good training curve is only the first check.

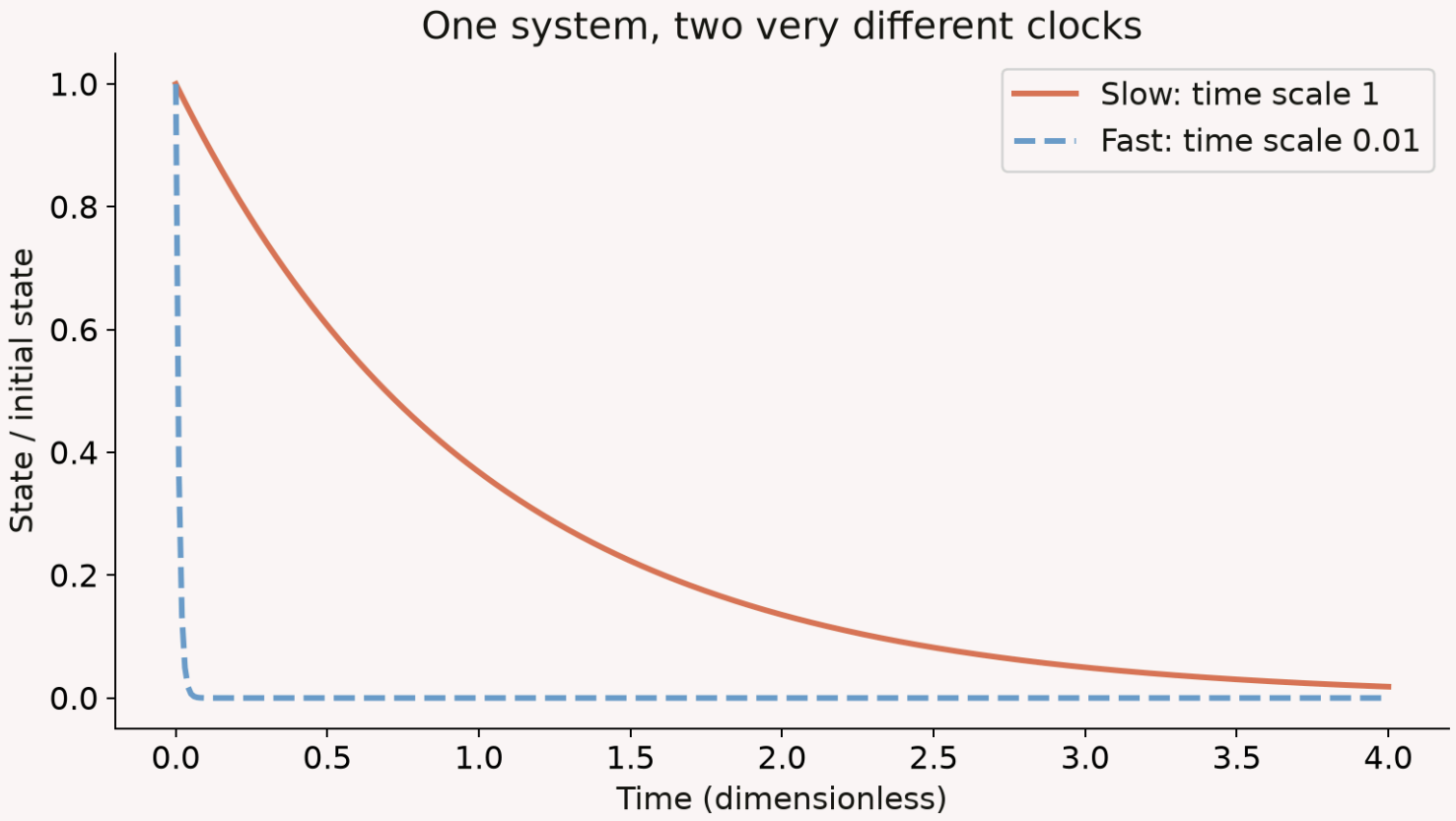
1. Unseen initial value: 1.2
2. Training ends at time 2
3. Evaluate the whole trajectory to time 4



Stiffness means competing clocks

A very fast component can restrict steps over a slow process.

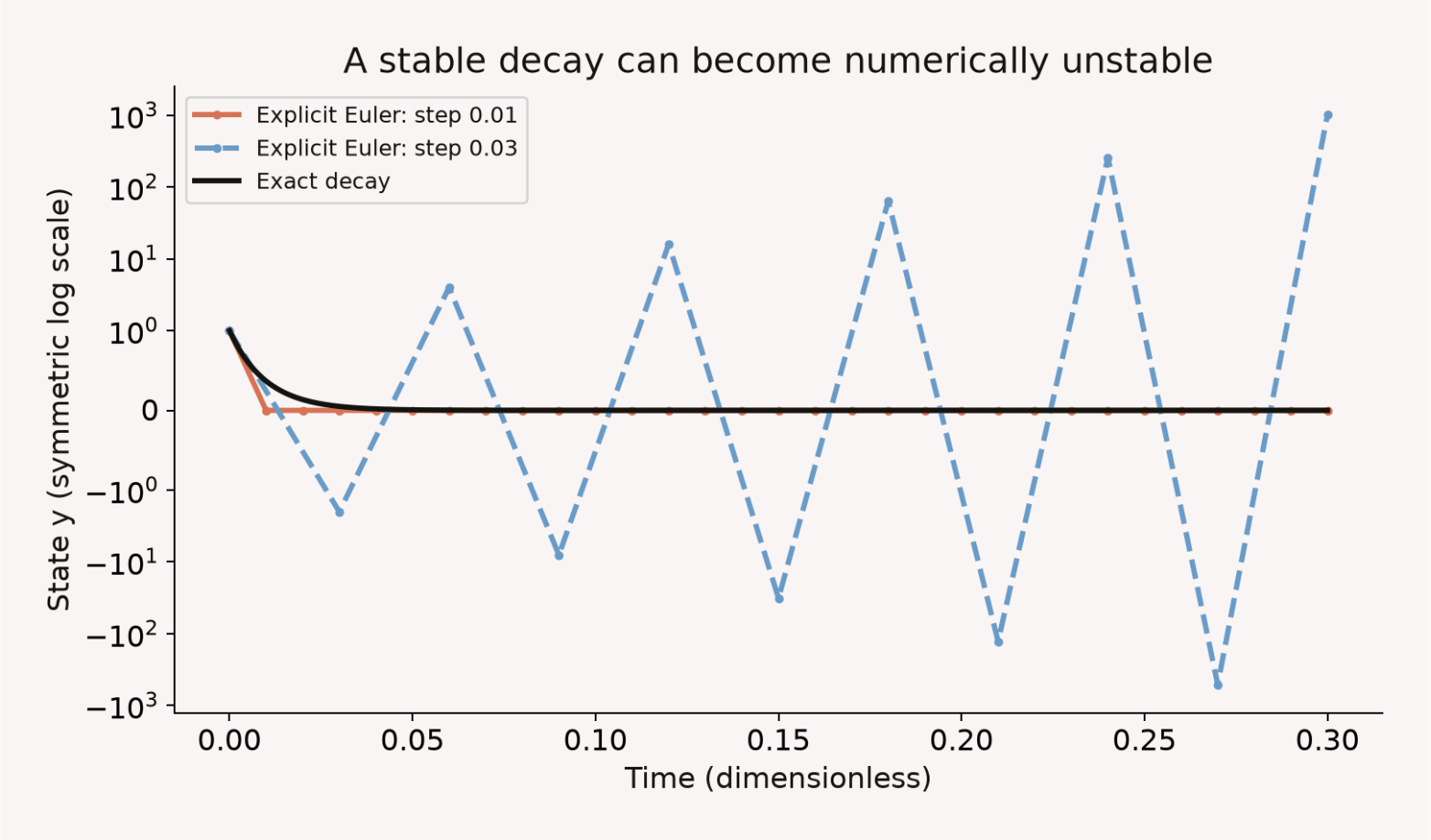
1. Slow decay: time scale 1
2. Fast decay: time scale 0.01
3. Small amplitude does not imply an easy solve



The solver is part of the model you run

Stable physics can still produce unstable numerical predictions.

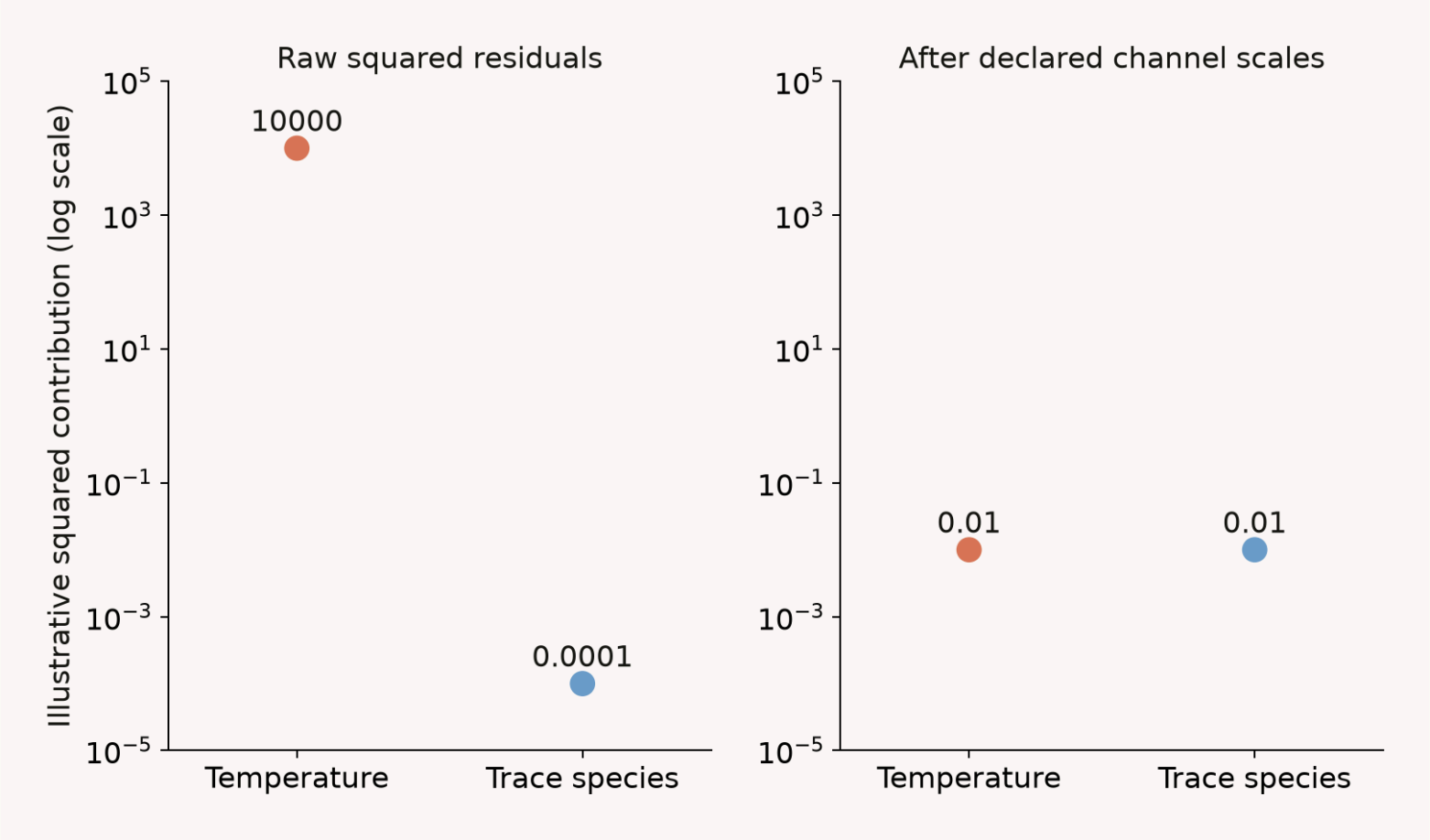
1. Euler multiplies by $1 - 100h$ in this toy
2. $h = 0.01$ is stable; $h = 0.03$ is unstable
3. Choose and verify a solver for the dynamics



Normalize the microphones

Declared channel scales change which residuals the loss hears.

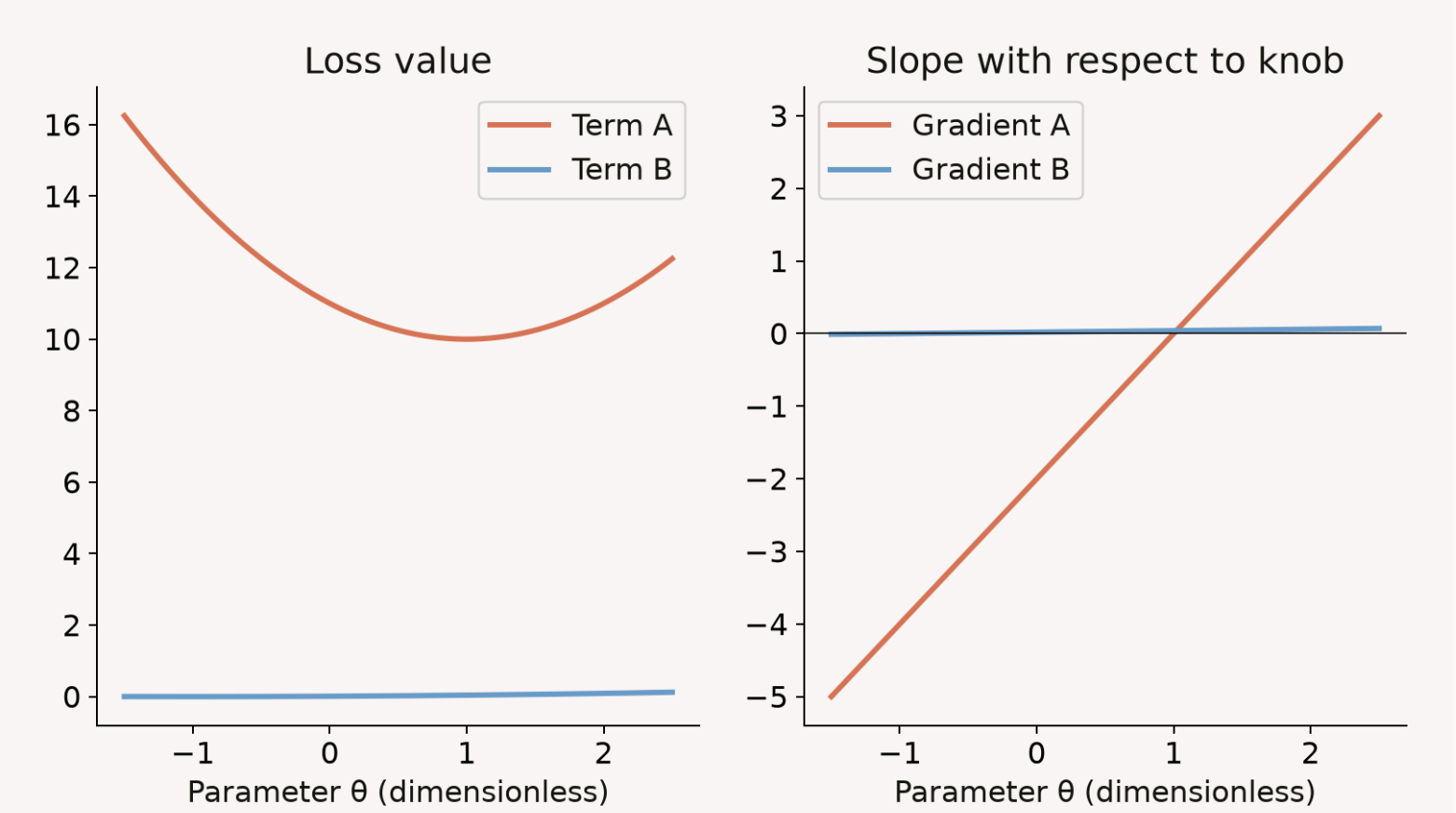
1. Raw units can drown out trace-species errors
2. Scale using training data or physical tolerances
3. Always report errors again in physical units



Loud loss is not loud gradient

Measure the push on the weights, not only the score value.

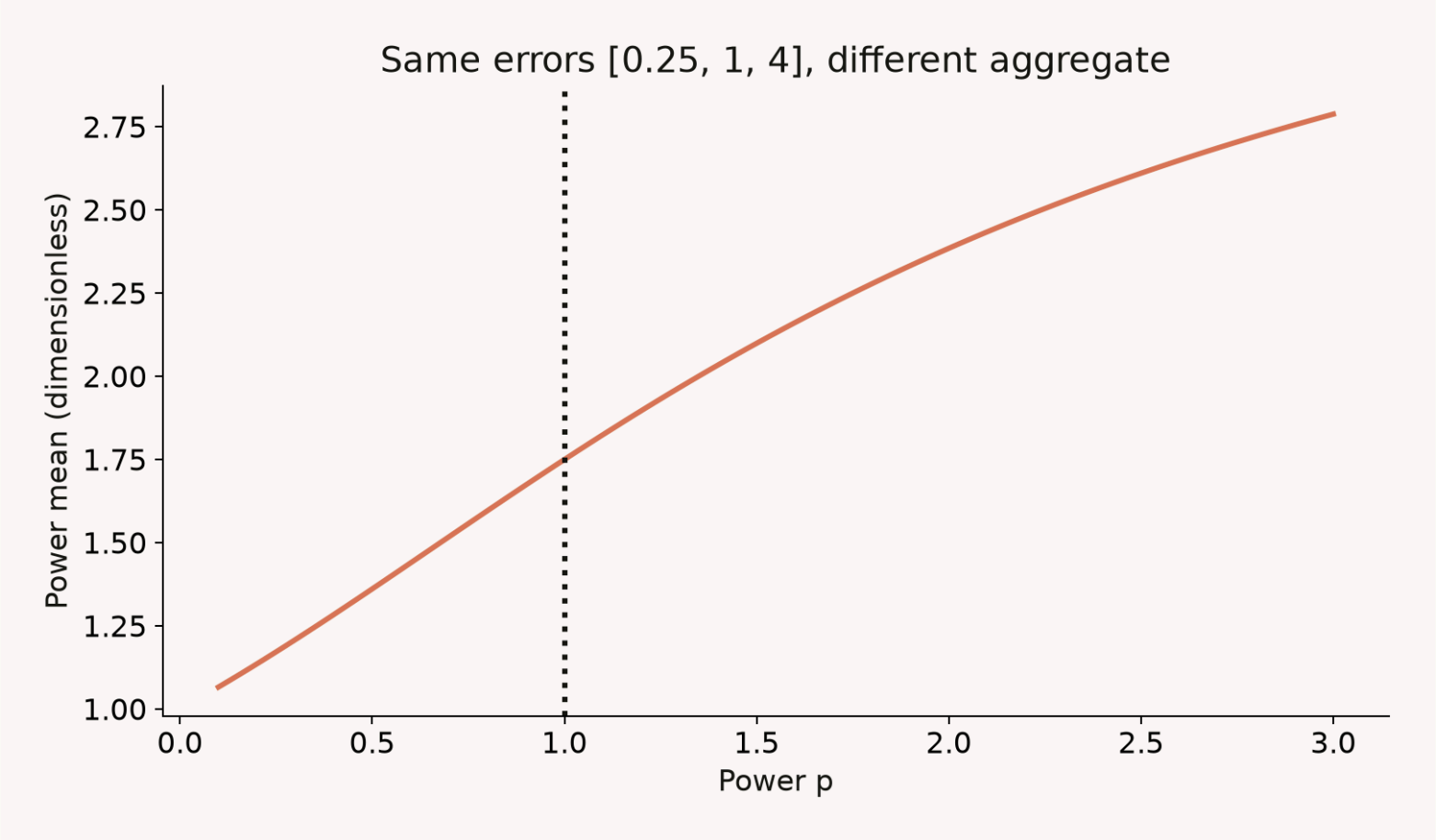
1. A constant adds loss but adds no gradient
2. Different terms can dominate at different epochs
3. Check both magnitude and direction



The power mean is an emphasis knob

It combines positive channel scores; it does not change chemistry.

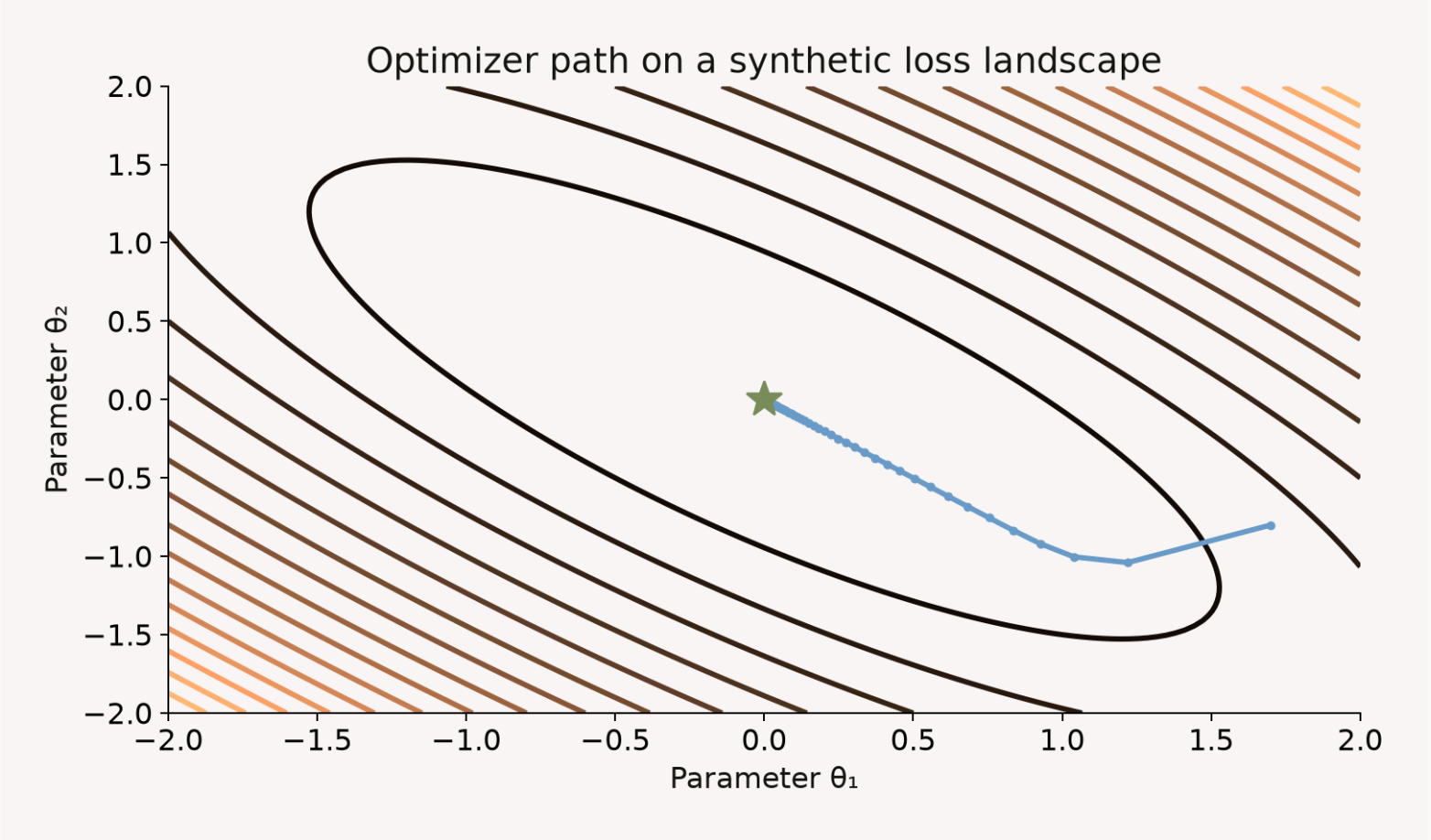
1. Calibrate against a fixed reference if intended
2. p = 1 is an arithmetic mean of those inputs
3. Smaller p does not guarantee every channel improves



The landscape is over network weights

The optimizer moves the knobs; the solver moves the state.

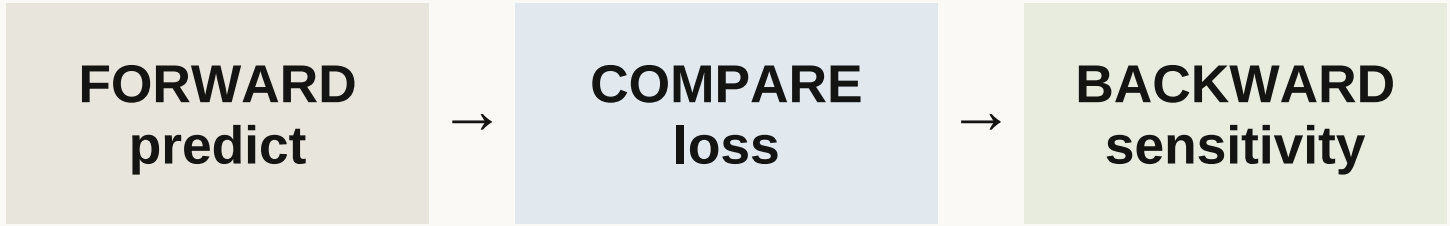
1. Horizontal axes: two illustrative parameters
2. Height: the training score
3. A descent path is not a chemical trajectory



Forward prediction and backward credit

Training asks which earlier decisions caused the final mismatch.

1. Forward: initial state → predicted trajectory
2. Compare: prediction → loss
3. Backward: loss → parameter sensitivities



Backward credit is not reversed chemistry

Check the gradient on a problem you can solve

Direct and adjoint results should be tested against an independent answer.

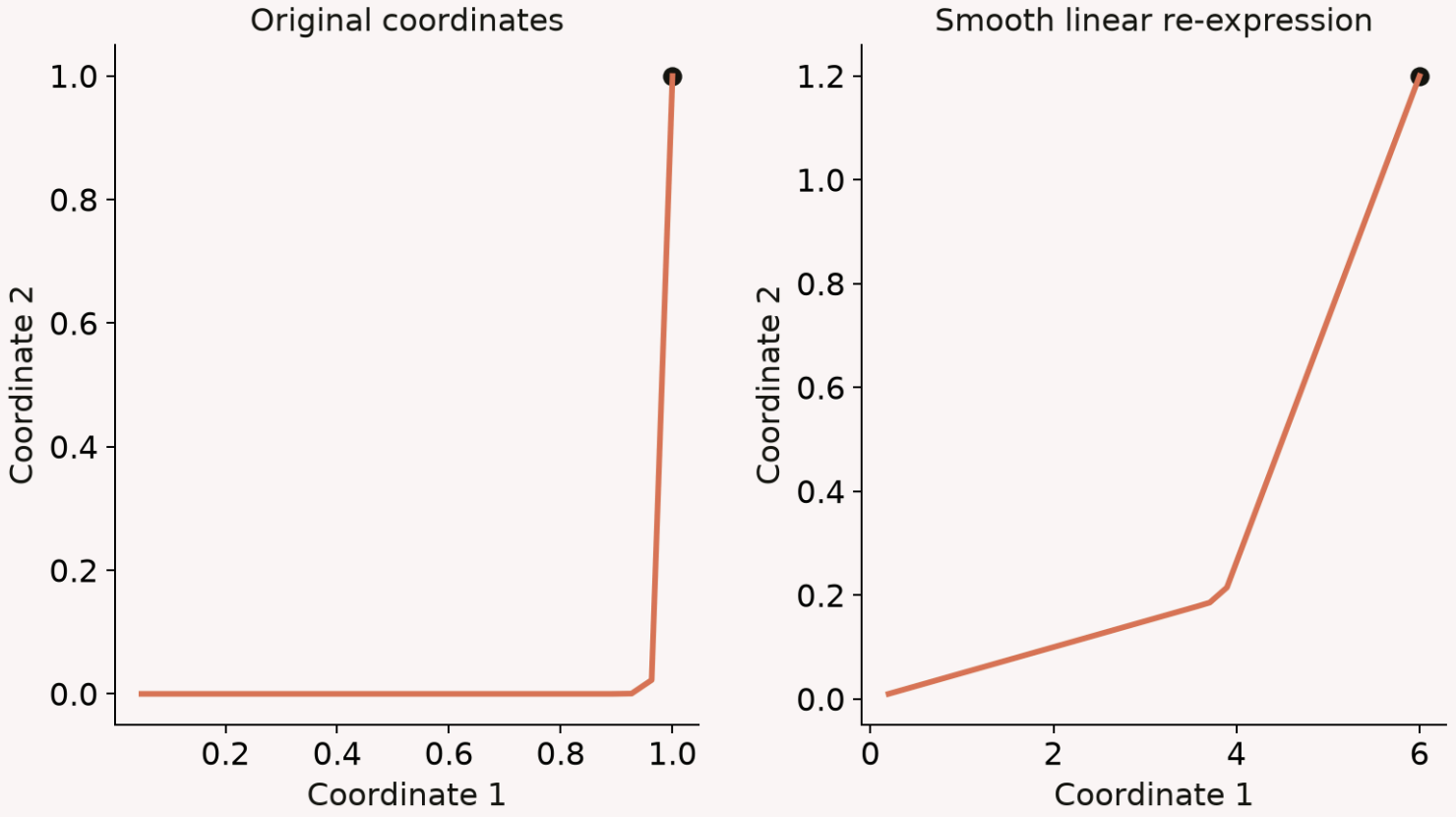
1. Toy field: $y' = \theta y$
2. Exact endpoint and exact derivative are available
3. This run checks both methods and finite differences

Analytic	**0.095925685**
Finite difference	**0.095925685**
Direct autograd	**0.095925685**
Adjoint	**0.095925685**

Change the coordinates, keep track of the meaning

A coordinate map changes the description of a state.

1. Raw state y → transformed state z
2. Learn or transform the rate consistently
3. Decode before physical interpretation



A new map does not erase every fast clock

At equilibrium, smooth invertible coordinates preserve eigenvalues.

1. Toy eigenvalues: -1 and -100
2. After a nonsingular linear map: still -1 and -100
3. Better scaling can help without proving stiffness removal

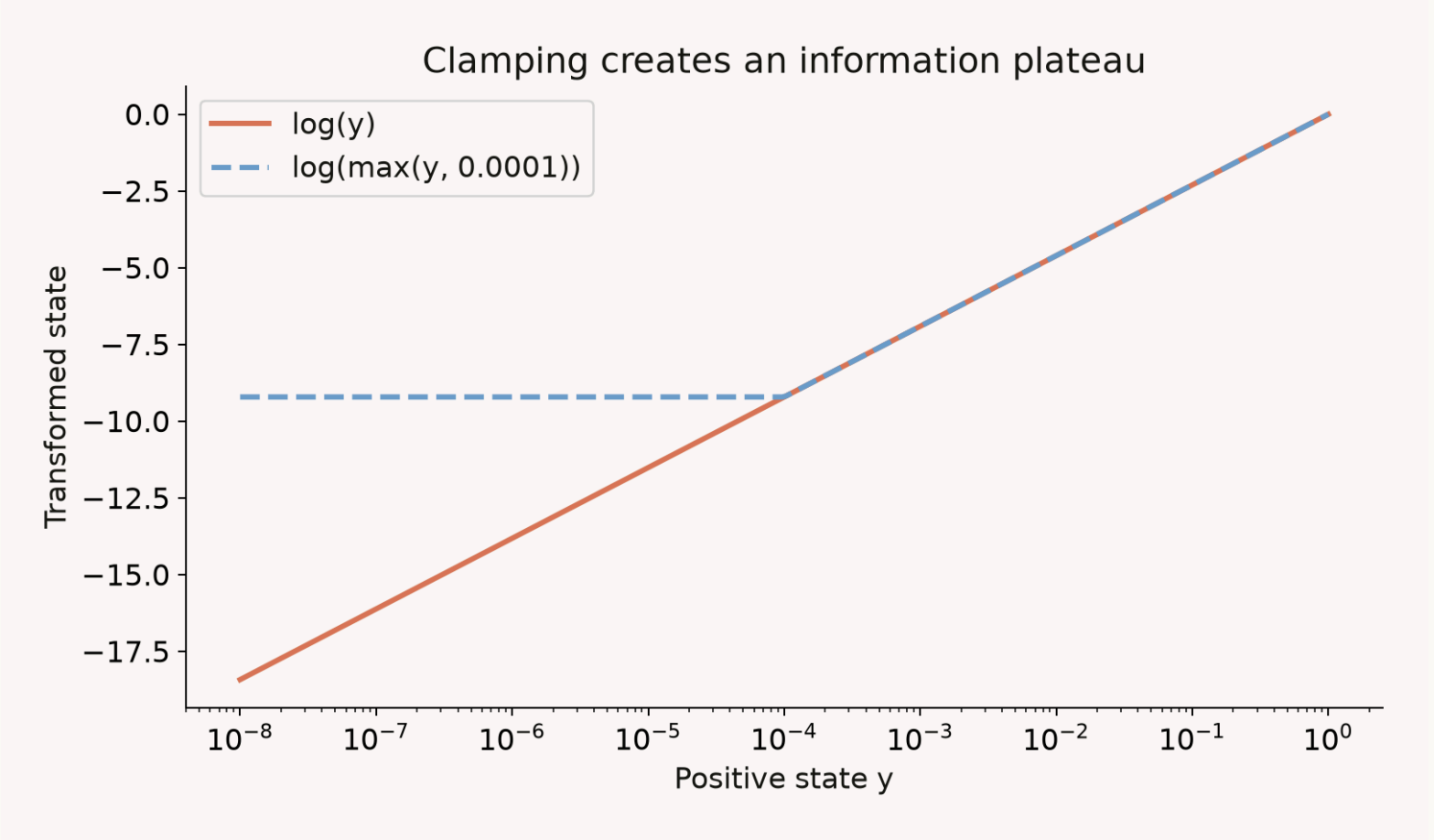


Different coordinates.
Same equilibrium eigenvalues.

Log transforms need a boundary policy

Small concentrations and exact zeros are different cases.

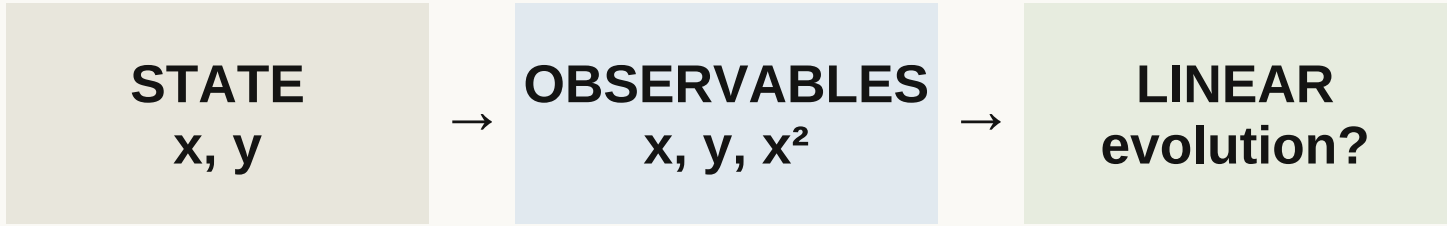
1. log(y) requires positive y
2. Clamping merges distinct small values
3. An epsilon changes the representation



Koopman changes what we observe

Nonlinear state dynamics can act linearly on functions of state.

1. An observable could be x, x^2 or a sensor signal
2. The exact operator acts on a function space
3. A small closed representation is a special achievement



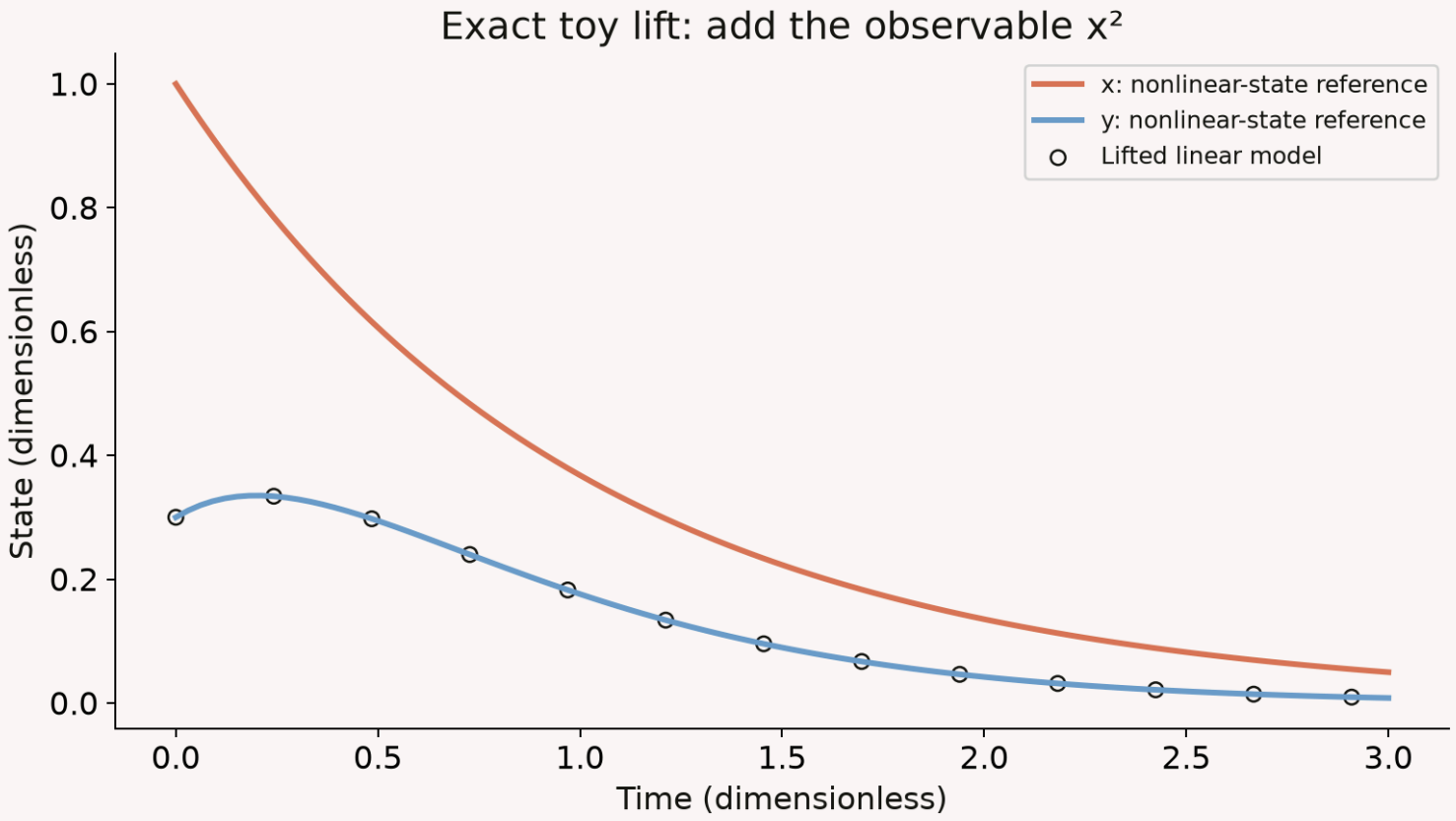
Finite closure must be demonstrated

One extra observable closes this toy

Add x^2 and a nonlinear system becomes a three-variable linear system.

$$x' = -x;\ y' = -2y + x^2$$

```
K = [[-1, 0, 0],
[ 0,-2, 1],
[ 0, 0,-2]]
lifted_t = expm(K*t) @ [1, .3, 1]
```

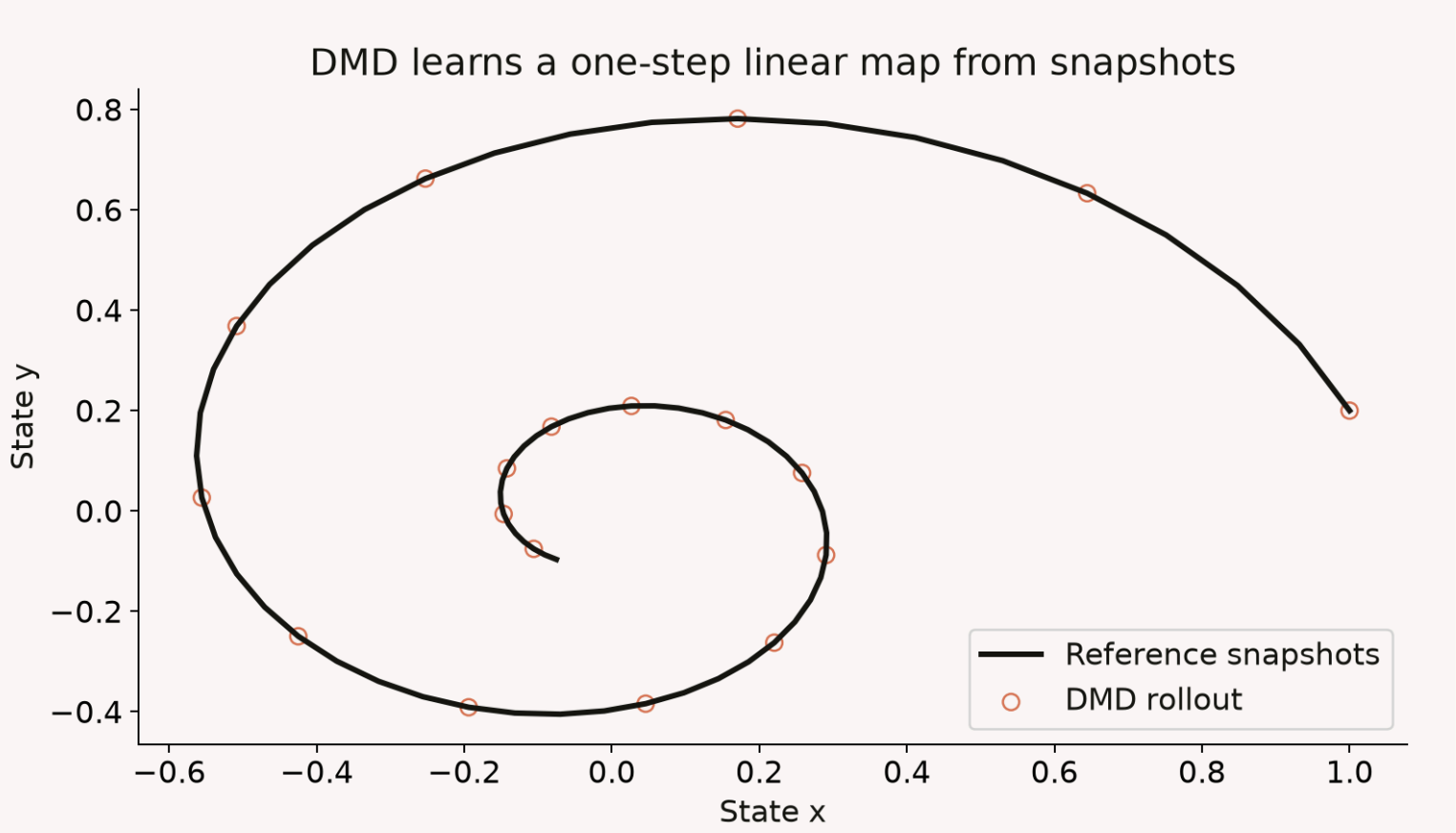


DMD fits a linear step from snapshots

Pair each measured snapshot with the one that follows it.

X: current snapshots

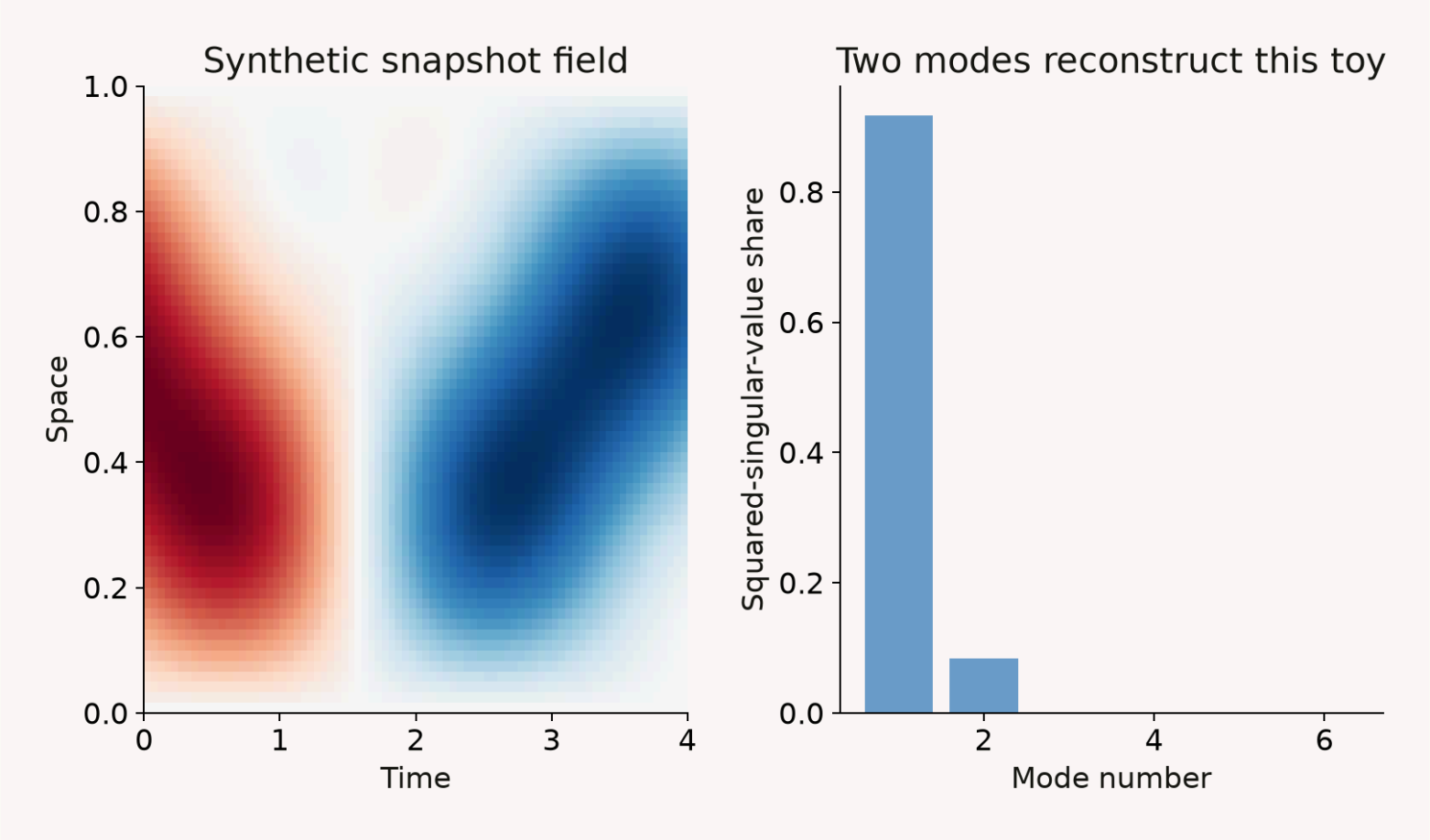
```
X = snapshots[:, :-1]
Y = snapshots[:, 1:]
A = Y @ np.linalg.pinv(X)
next_state = A @ current_state
```



POD compresses snapshots

Compact reconstruction does not yet give a predictive dynamics model.

1. Find dominant directions using an SVD
2. Retain a chosen number of modes
3. Validate rollout as well as reconstruction



A latent Neural ODE adds a learned evolution

Compress → evolve a hidden state → reconstruct observations.

1. Encoder defines latent coordinates
2. Neural ODE supplies latent dynamics
3. Decoder returns the quantities we can compare

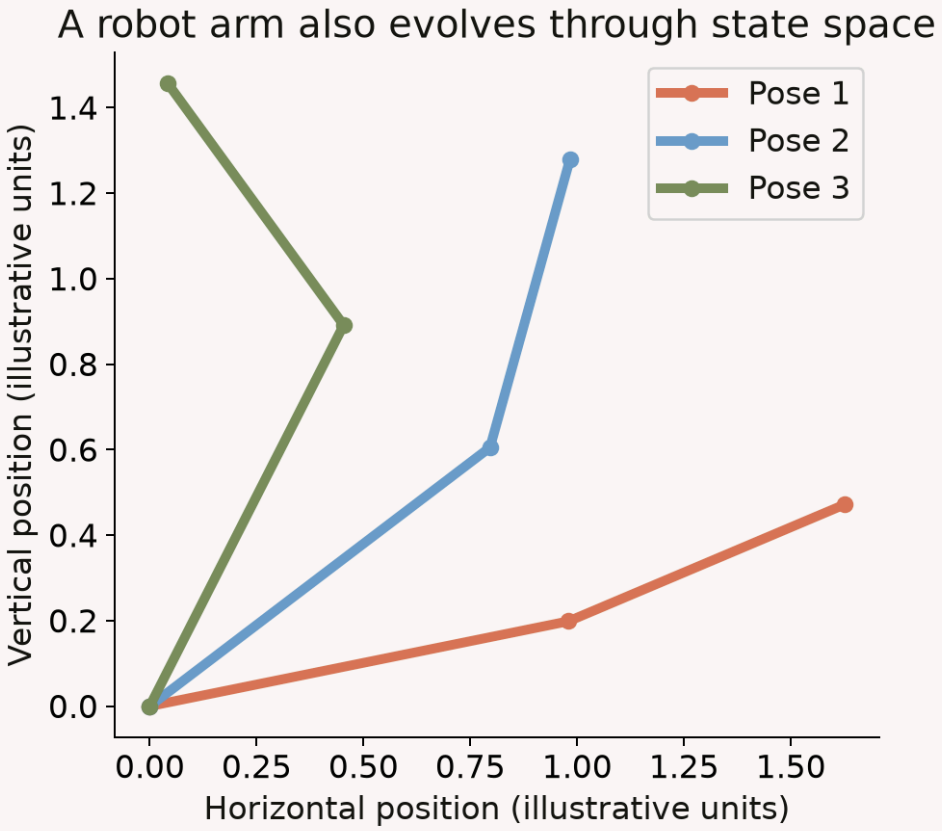


Compression must preserve predictive information

A robot arm makes state and rate tangible

Angles and velocities evolve; torques drive the motion.

1. Position alone does not specify future motion
2. An acceleration law completes the state derivative
3. Use the analogy to organize, not equate, physics



Build a valid mechanical mass matrix

Cholesky factors are structural knobs with a useful guarantee.

1. Choose a lower-triangular factor L
2. Construct $M = LL^T + \varepsilon I$, with $\varepsilon > 0$
3. Then kinetic energy is positive for nonzero velocity

$$M = LL^T + \varepsilon I$$

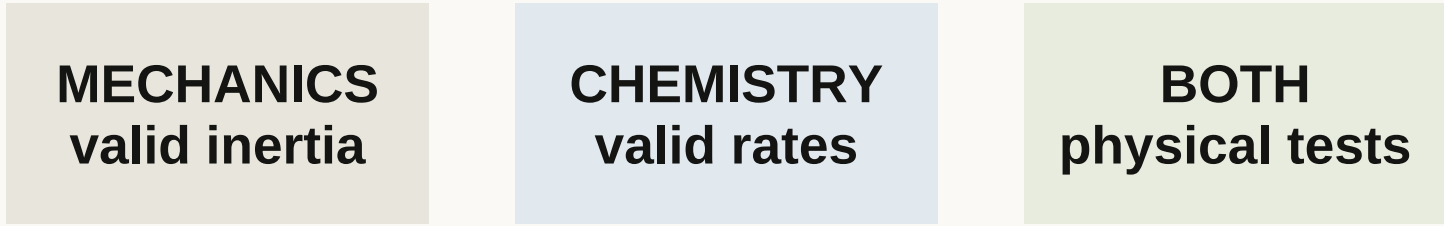
$\varepsilon > 0$ guarantees a positive-definite M

Structure helps. Accuracy still needs data.

Use constraints that match the domain

Good structure removes impossible behavior without inventing physics.

1. Mechanics: inertia, forces and energy accounting
2. Chemistry: species, elements and reactor energy balance
3. Check the complete state, including omitted species



Good structure removes impossible behavior without inventing physics.

A graph tells the model who interacts

Connectivity is prior information; it is not a complete rate law.

1. Reaction graph: species and their interactions
2. Molecular graph: atoms and bonds within a molecule
3. vZero includes a species-graph front layer

SPECIES
reaction graph

ATOMS
molecular graph

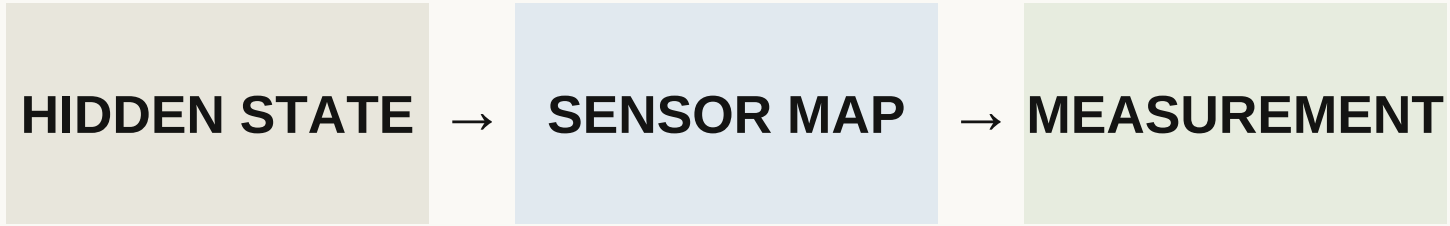
DIFFERENT
representations

Connectivity guides the model; it does not determine rates

A camera sees an observation of the state

Write the measurement process before fitting its output.

1. Dynamics predicts the hidden physical state
2. An observation map predicts sensor output
3. Tomography reconstructs from projections

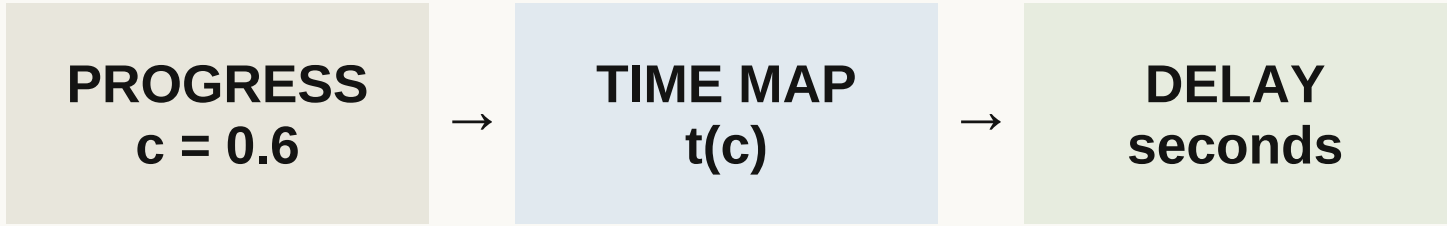


A measurement may hide much of the state

A progress coordinate is not a clock

Matching a curve’s shape can still miss ignition timing.

1. Progress might run from 0 to 1
2. Ignition delay is measured in physical time
3. Retain or learn the mapping back to seconds

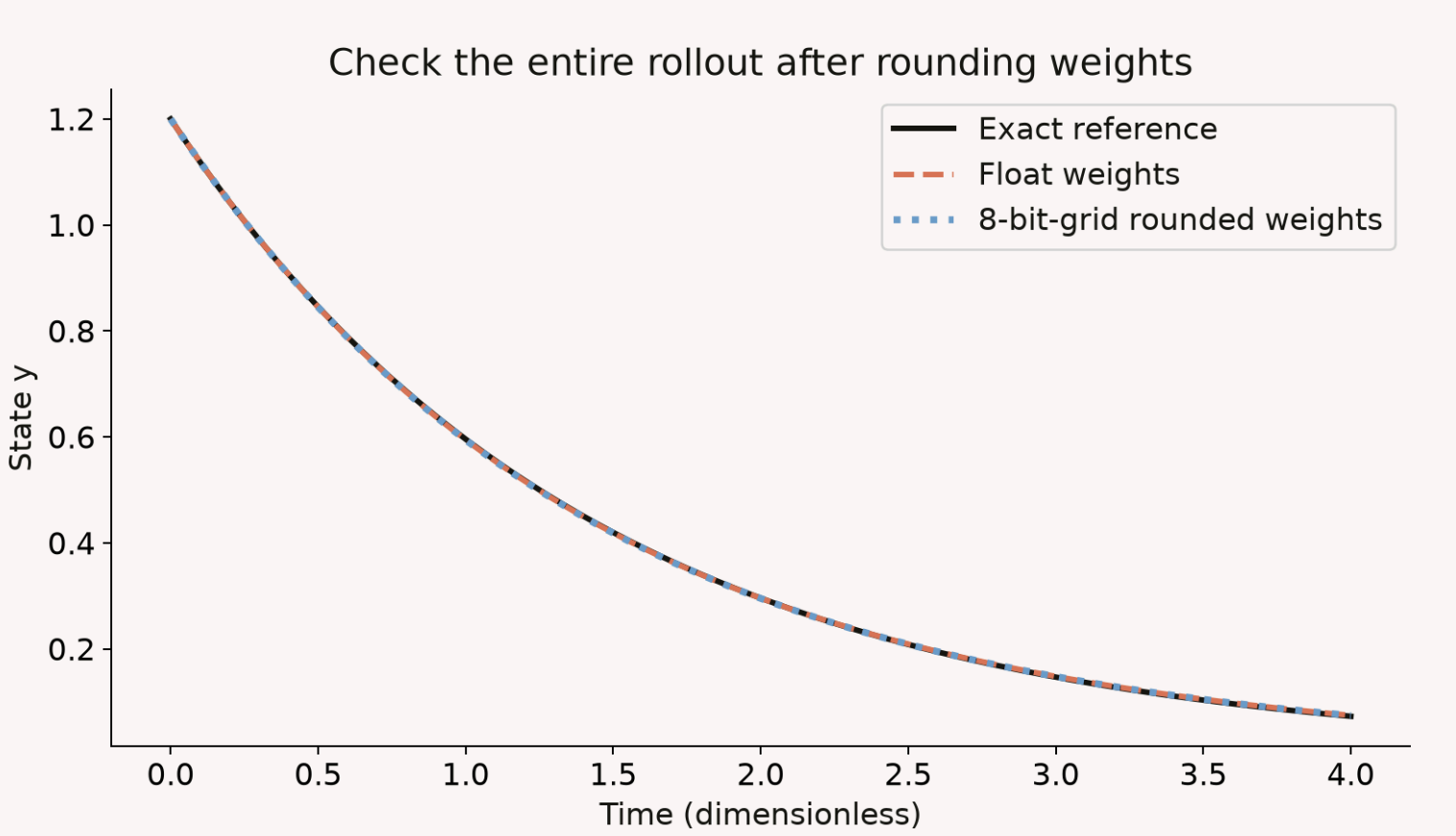


A peak’s progress location is not a time measurement

Quantization is controlled rounding

Check rollout error after reducing numerical precision.

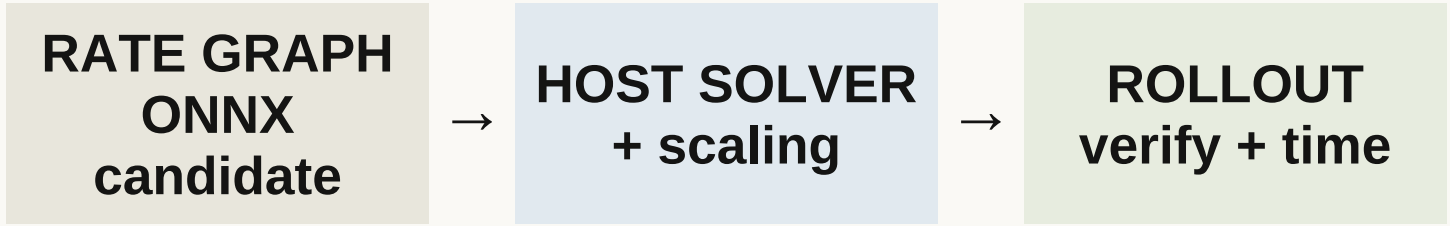
1. FP16 conversion and INT8 quantization are different
2. Our toy rounds weights to an 8-bit grid
3. A smaller file does not prove a faster solve



ONNX carries a computation graph

Export the right component and verify the complete inference path.

1. Candidate: export the rate network
2. Keep preprocessing and solver behavior explicit
3. Compare reference and deployed trajectories



Exporting the network alone does not export the solver

Change one layer, keep the comparison honest

A useful ablation answers a specific question.

1. Baseline → one controlled change → repeated seeds
2. Hold data, metrics and budgets comparable
3. Inspect per-channel and physical outcomes



Keep the physical metric and comparison budget explicit

Learn the rate. Respect the solver. Check the physics.

The notes describe connected tools with distinct responsibilities.

1. Microphones: which learning signals are heard?
2. Knobs: which part of the pipeline changes?
3. Landscape: how are the weights updated?

MICROPHONES
loss signals

KNOBS
design choices

LANDSCAPE
optimization

Which part changes—and what evidence shows it helps?