MACHINE LEARNING

MEI/1



University of Beira Interior,
Department of Informatics

Hugo Pedro Proença,
hugomcp@di.ubi.pt, 2026/27

Machine Learning

Week 12

Syllabus

Continuous function representations

Physics-informed neural networks

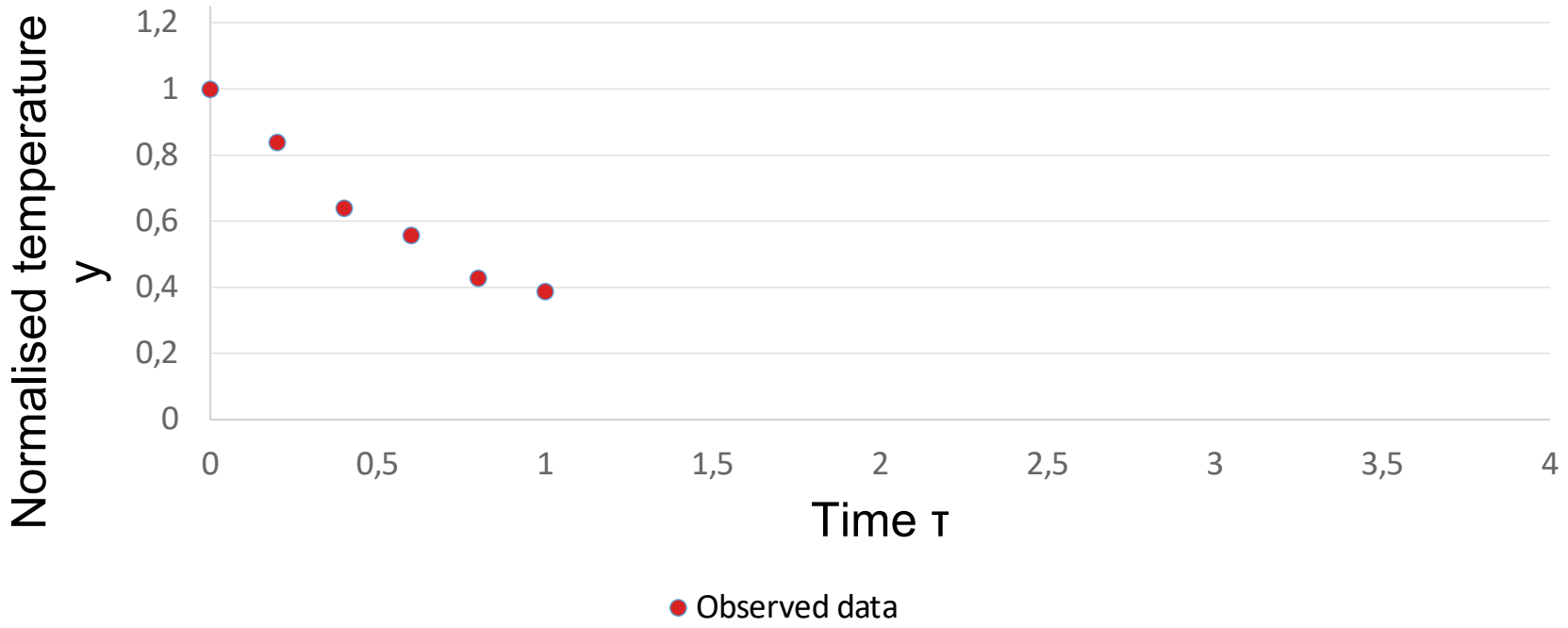
Forward and inverse problems

Practical cases and limitations

A cooling-system experiment

A cooling system with few sensors

Six readings from a warm device. What happens after the final reading?



Data alone leave many possible continuations.

Physics-inspired machine learning

Domain knowledge can enter the model in different places.

Loss: penalise violations of a differential equation → PINN

Architecture: build conservation or symmetry into the model

Hybrid model: learn the part that the simulator does not explain

Today's focus: a continuous solution constrained by a known law.

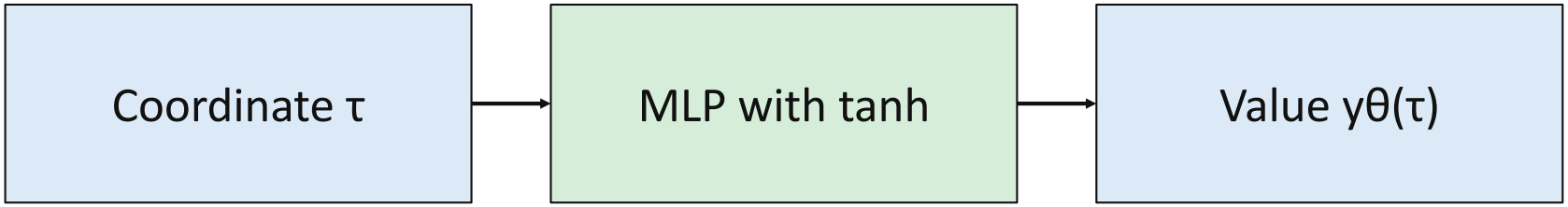
Samples and a function representation

The training locations and the representation are different objects.

Stored samples $(\tau_1, y_1), \ldots, (\tau_n, y_n)$	Shared weights θ $y\theta(\tau)$ = neural network(τ)
At a new τ: choose an interpolation rule.	At a new τ: evaluate the same network.

Finite weights can represent a function over a continuous domain.

Continuity alone does not make a PINN

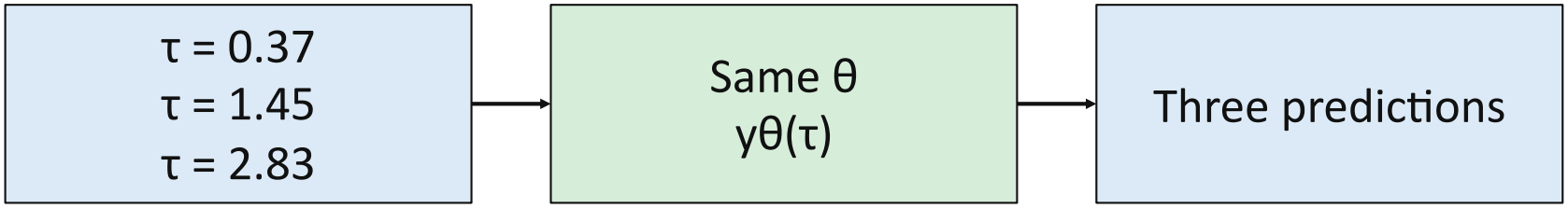


Data-only loss: match measured values.

PINN loss: also test the equation and conditions.

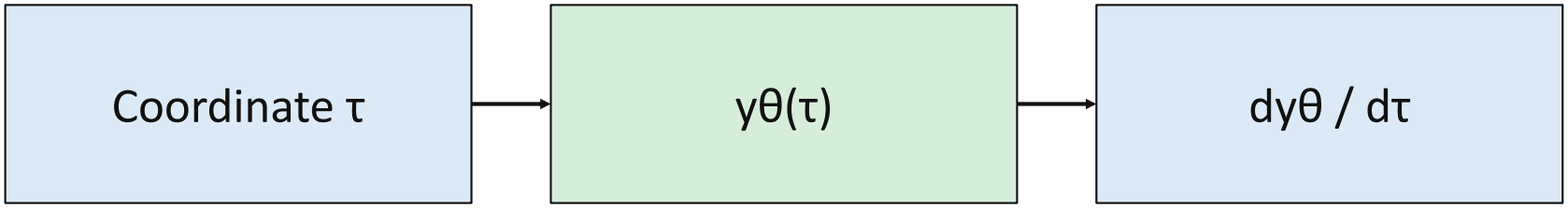
The additional constraint, rather than continuity itself, defines the PINN.

A query is not a new parameter



Changing the query resolution does not retrain the function.

Values, slopes and curvature



Value

Rate of change

With two spatial derivatives: $\partial^2 u\theta / \partial x^2$ describes curvature.

Physical laws often relate values to their derivatives.

Three kinds of information

Data	Equation	Initial condition
$y(\tau_i) \approx$ observed y_i	$y'(\tau) + y(\tau) = 0$	$y(0) = 1$
A few measured values	A relation across time	The starting state

A useful model reconciles all available information.

Newton's law of cooling

A well-mixed object loses heat to a fixed-temperature environment.

$$\frac{dT}{dt} = -k\,(T - T_a)$$

T: object temperature T_a: ambient temperature

$k > 0$: cooling rate, measured in time^{-1}

Initial condition: $T(0) = T_o$

A larger temperature difference gives a steeper cooling slope.

A dimensionless problem

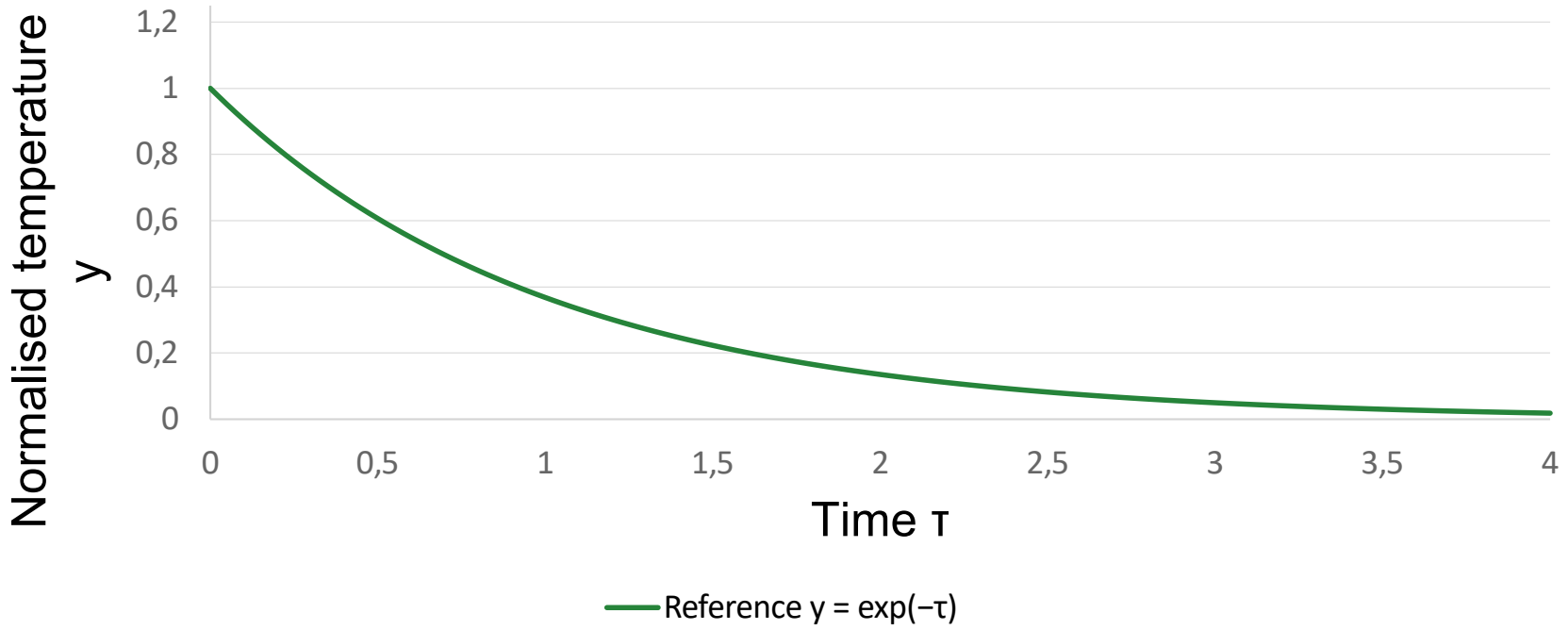
$$y = (T - T_a) / (T_0 - T_a) \qquad \tau = kt$$

$$dy / d\tau + y = 0$$

$$y(0) = 1$$

Normalisation makes the variables and loss terms easier to balance.

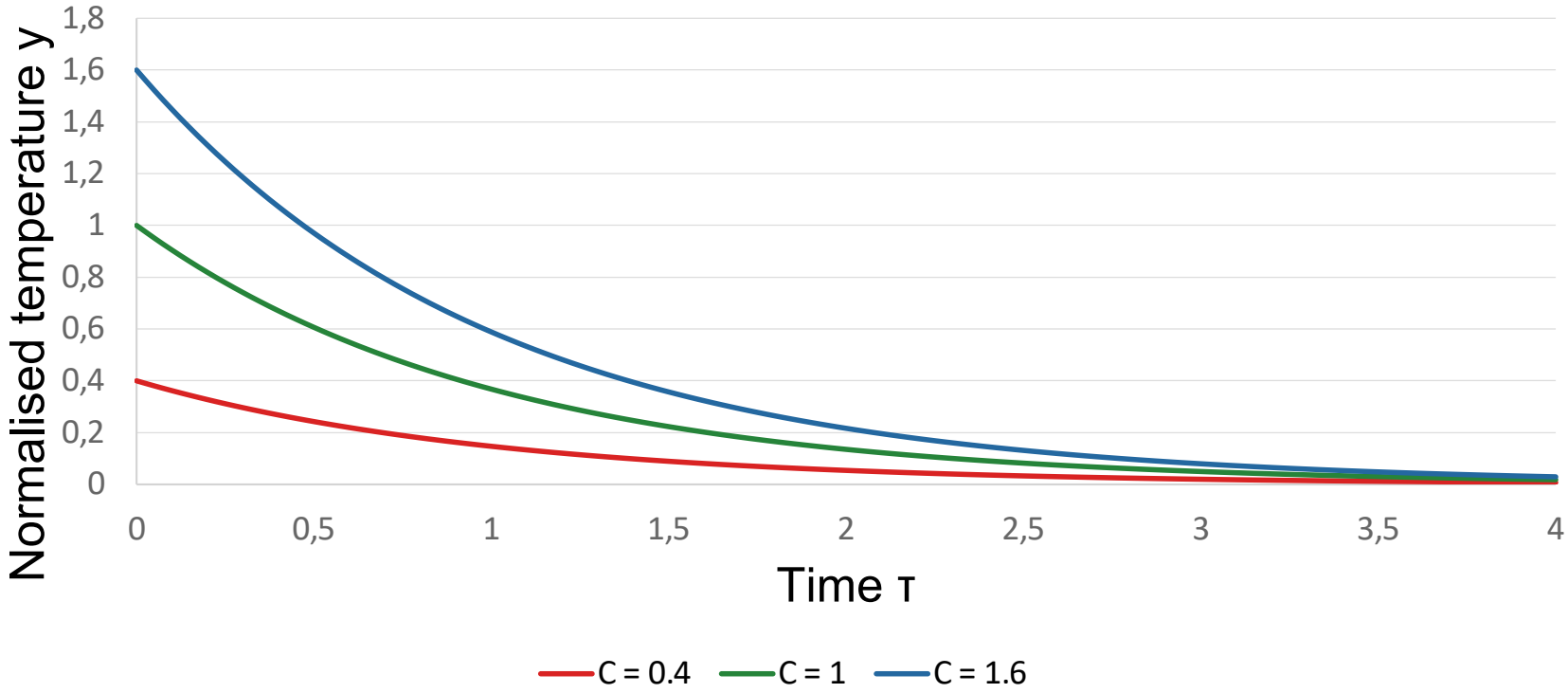
The equation is a local slope rule



At y = 1: y′ = −1 At y = 0.5: y′ = −0.5

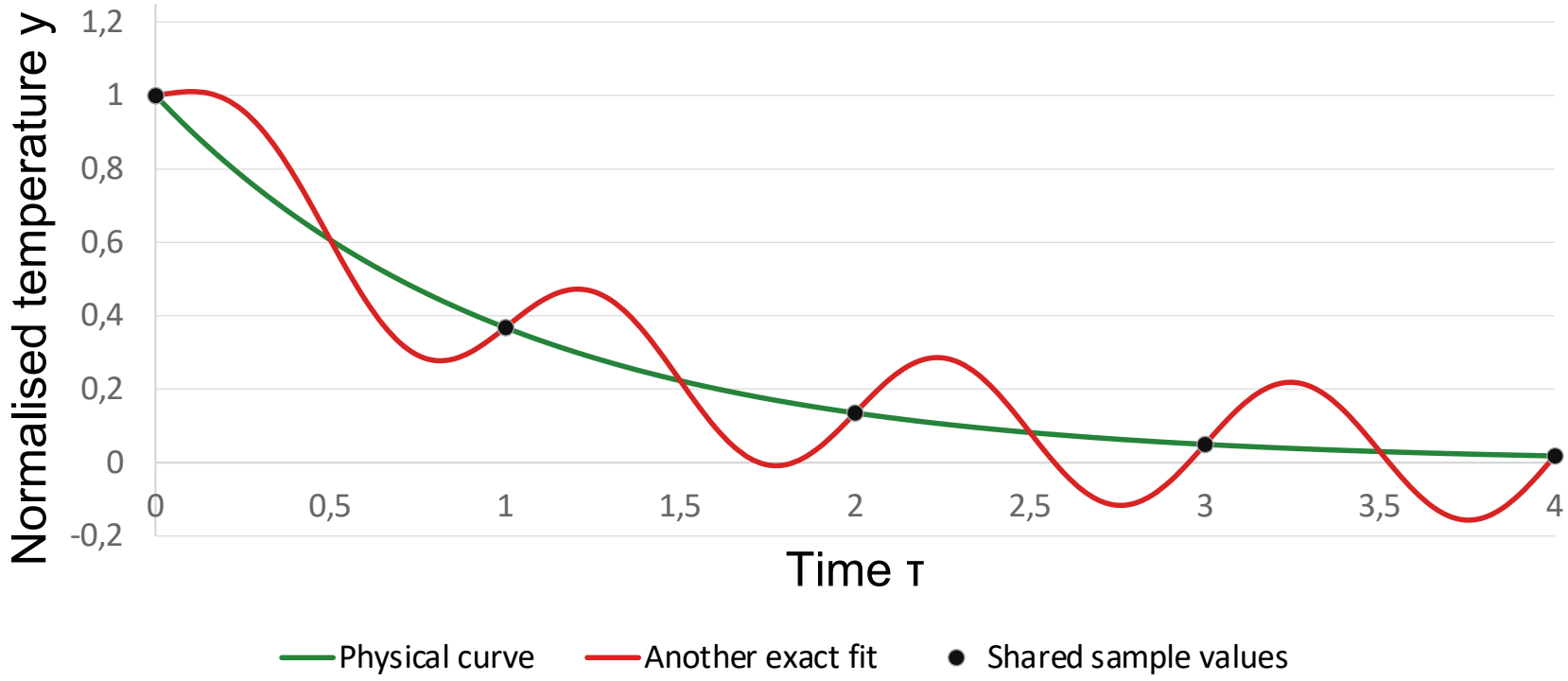
Cooling slows as the object approaches the environment.

The equation needs an initial condition



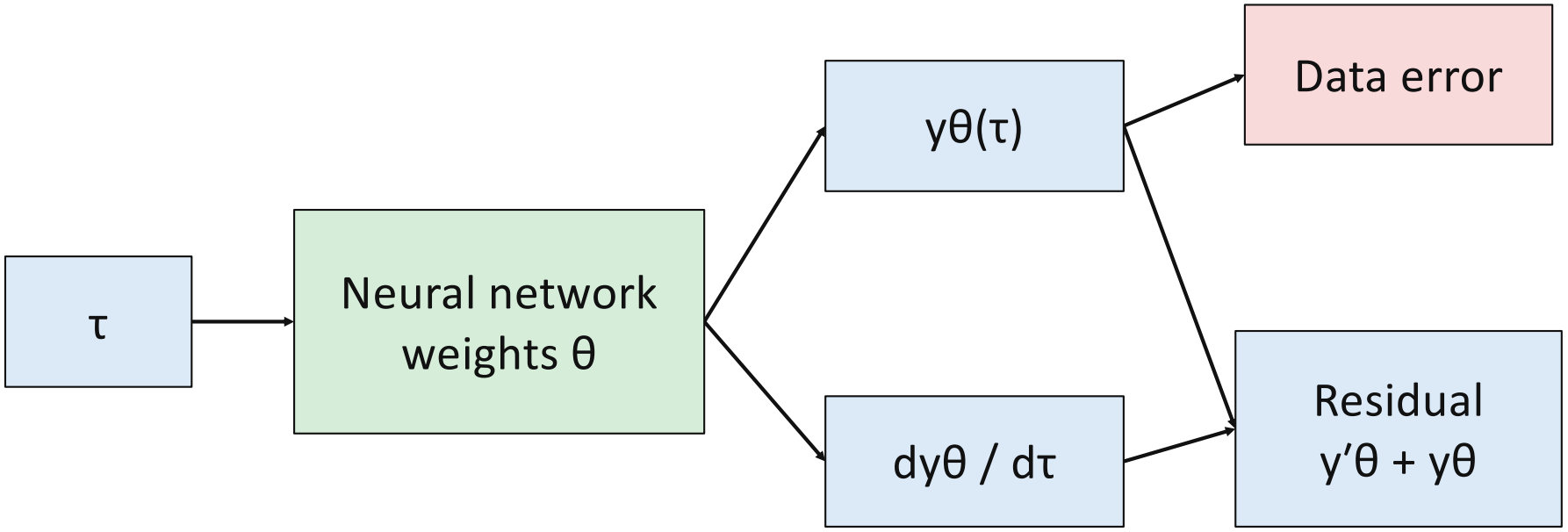
The law selects a family. y(0) = 1 selects one trajectory.

Fitting the points can hide wrong behaviour



The derivative exposes behaviour that pointwise errors miss.

Physics-informed neural network



Initial condition: compare yθ(0) with 1

One function. Several complementary tests.

The physics residual

$$r\theta(\tau) = dy\theta(\tau) / d\tau + y\theta(\tau)$$

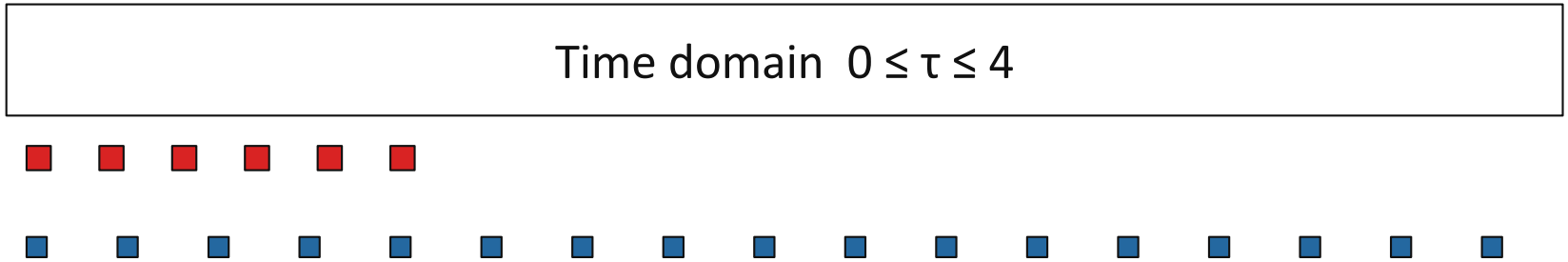
$$r\theta(\tau) = 0$$

The predicted slope matches the physical slope at τ.

Residuals measure equation violations without needing a target y.

Collocation points

A collocation point asks: does the predicted function obey the law here?



Observations: values available only in $0 \le \tau \le 1$

Collocation: equation checks throughout $0 \le \tau \le 4$

Collocation points are checks, not control values of the solution.

The training objective

$$L = \lambda_d L_{data} + \lambda_0 L_{initial} + \lambda_r L_{physics}$$

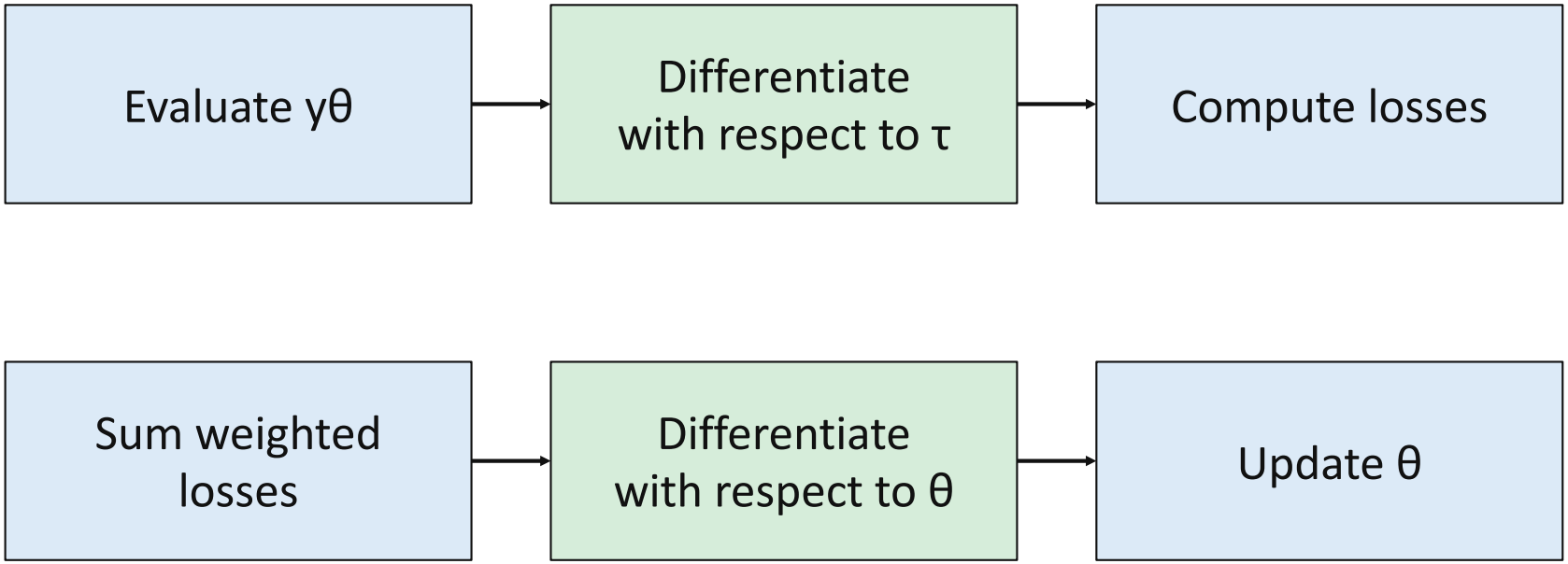
$$L_{data} = \text{mean}_i \left[y_\theta(\tau_i) - y_i \right]^2$$

$$L_{initial} = \left[y_\theta(0) - 1 \right]^2$$

$$L_{physics} = \text{mean}_j \left[r_\theta(\tau_j) \right]^2$$

Optimisation balances fitting observations and satisfying the model.

One training iteration



Coordinate derivatives test the law. Parameter derivatives learn the function.

Residual check: think, then discuss

Which candidates satisfy both $y' + y = 0$ and $y(0) = 1$?

A. $y(\tau) = \exp(-\tau)$

B. $y(\tau) = 0$

C. $y(\tau) = \exp(-2\tau)$

Compute a derivative before judging the shape of the curve.

Pause

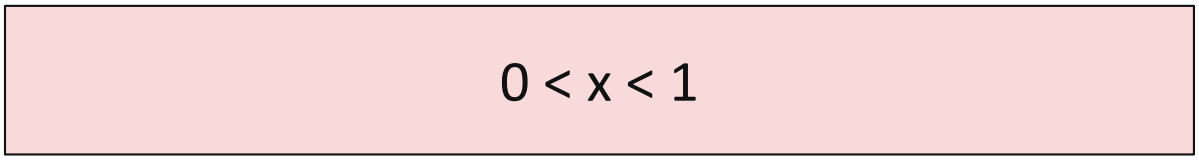
5 minutes

After the break: spatial fields and practical comparisons

A spatial example: heat diffusion

u(x,t) is the temperature along a rod.

$$\partial u / \partial t = \alpha \, \partial^2 u / \partial x^2$$



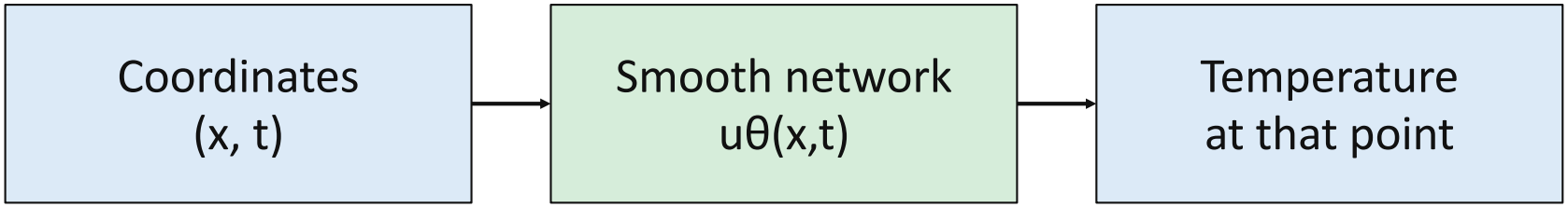
u(0,t) = 0

u(1,t) = 0

u(x,0) = sin(πx)

The initial field and endpoint conditions complete the problem.

One network represents a space–time field



$$r\theta(x,t) = \partial u\theta / \partial t - \alpha \, \partial^2 u\theta / \partial x^2$$

A rendered grid samples the learned field at a chosen resolution.

Forward and inverse problems

FORWARD

Known k and T_o

Learn $T\theta(t)$

INVERSE

Observed temperatures

Learn $T\theta(t)$ and k

The same differential equation can support different inference tasks.

An unknown parameter needs information

$$r_{\theta,\beta}(\tau) = y'_{\theta}(\tau) + \beta\, y_{\theta}(\tau)$$

$\beta = k\, t_s > 0$, with a fixed reference time t_s

Learn θ and β from measurements away from equilibrium.

At $y = 0$, every positive β gives the same equilibrium.

A learnable parameter is not necessarily an identifiable parameter.

Soft and hard constraints

Soft: add $[y\theta(0) - 1]^2$ to the loss

Hard: $y\theta(\tau) = 1 + \tau\, N\theta(\tau)$

At $\tau = 0$: $y\theta(0) = 1$ for every choice of θ

Hard constraints can guarantee a selected condition, not the entire equation.

Automatic differentiation in code

```
tau = tau.detach().clone().requires_grad_(True)
y = model(tau)
dy = torch.autograd.grad(
    y, tau, grad_outputs=torch.ones_like(y),
    create_graph=True
)[0]
r = dy + y
```

create_graph=True keeps the derivative available for training.

A fair practical comparison

Same MLP: 1–32–32–1 with tanh

Same six noisy readings and y(0) = 1

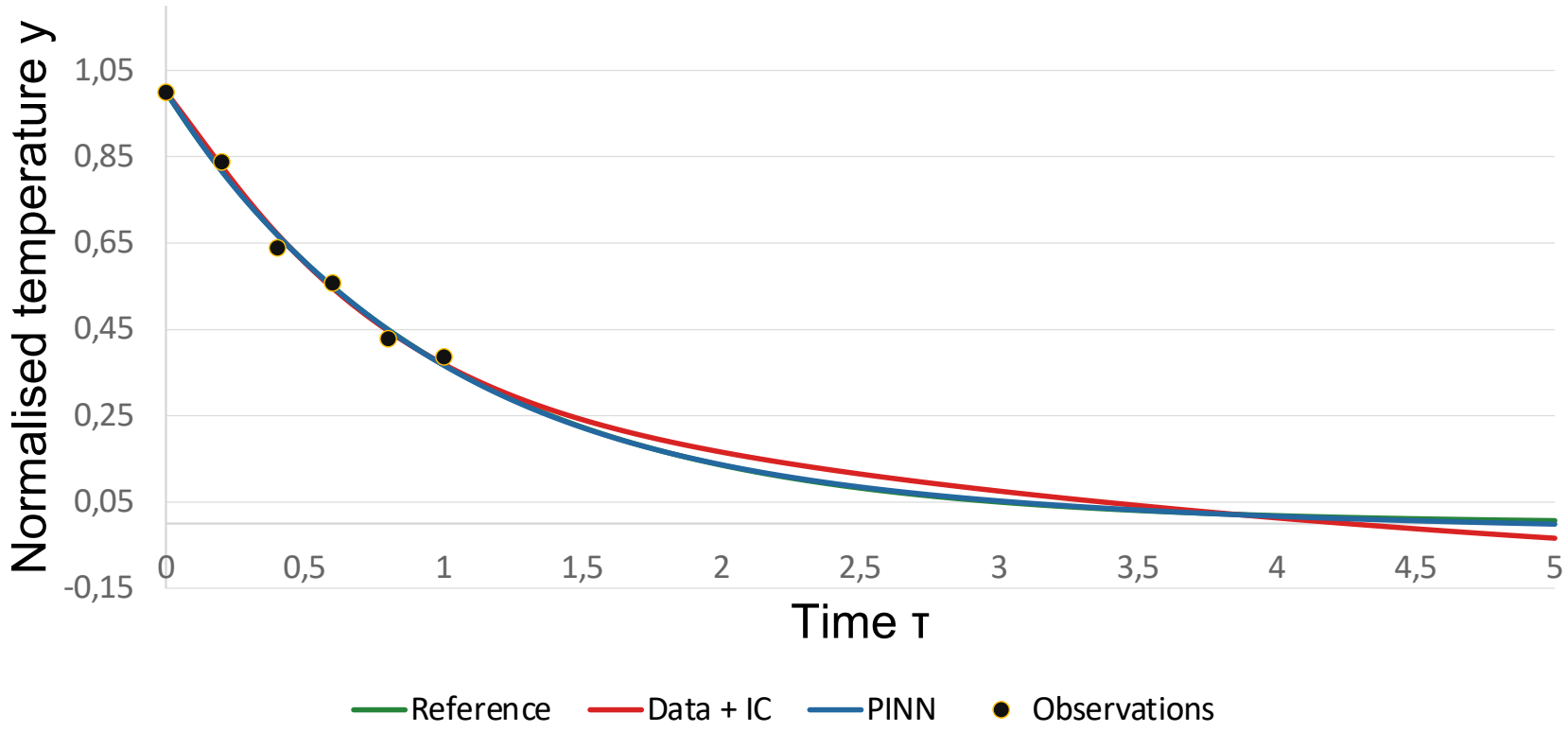
Same initial weights, Adam and 2,000 updates

Baseline: Ldata + 10 Linitial

PINN: Ldata + 10 Linitial + Lphysics

We compare the value of physical prior information under controlled settings.

Cooling experiment: an actual training run



Both models represent functions. Only one uses the equation during training.

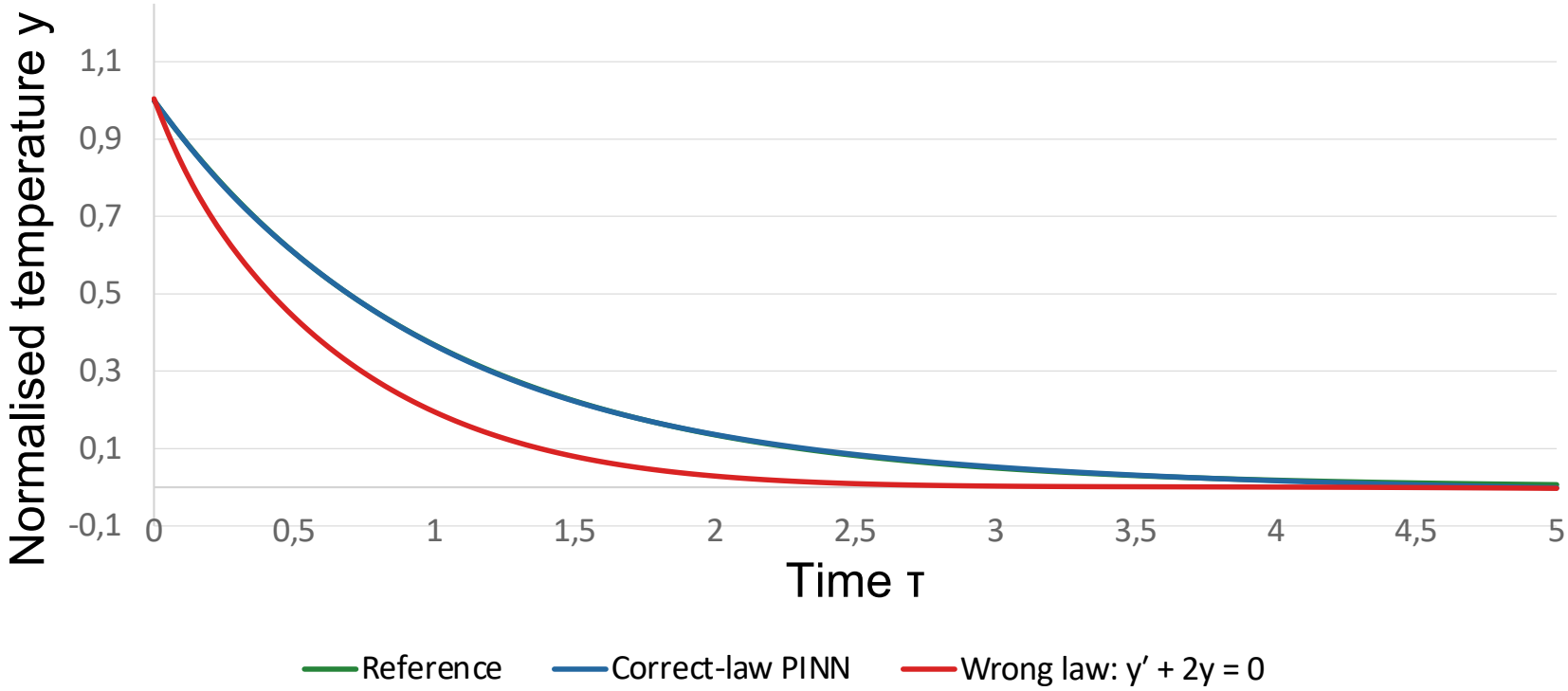
Three evaluation regions

$0 \le \tau \le 1$	$1 < \tau \le 4$	$4 < \tau \le 5$
Data region	Physics-only region	Outside training

Equation checks cover the first two regions.

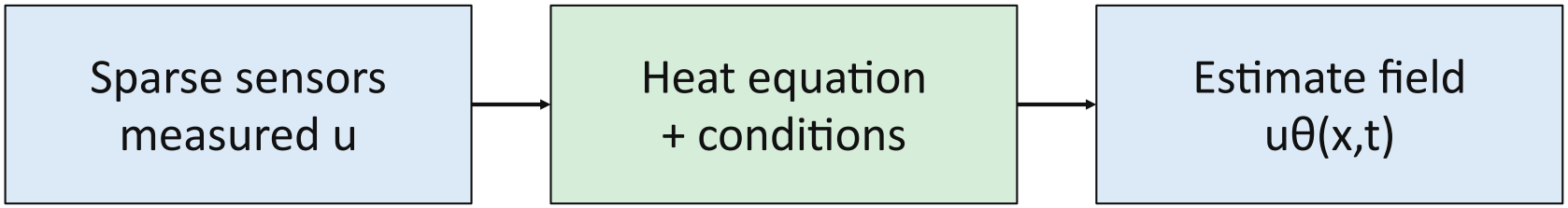
A prediction outside the data range is not always outside the training domain.

Incorrect physics can harm predictions



Physics helps when the chosen model describes the process adequately.

Sparse temperature sensors in a material



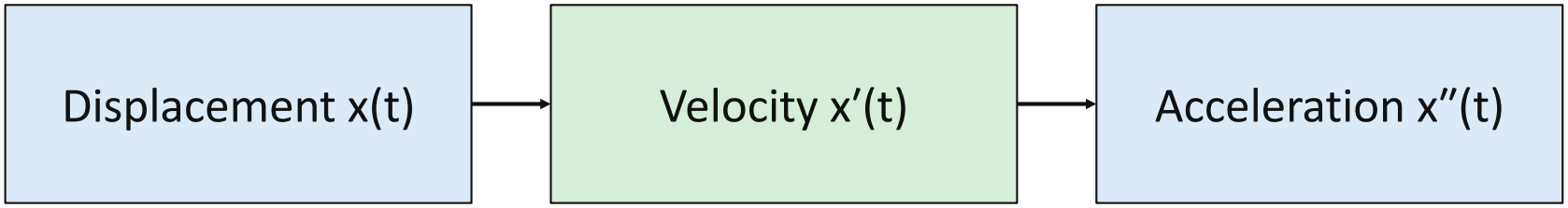
Supervised model: learns from labelled field examples.

PINN: adds equation checks where sensors are absent.

Unknown heat sources or incorrect boundary conditions can invalidate the estimate.

Mechanical motion and energy

$$m\,x'' + c\,x' + k\,x = 0$$



For c = 0: energy remains constant in the ideal model.

A physics penalty and a conservation-based architecture are different choices.

A numerical solver remains an essential baseline

Known equation, one forward problem: start with a solver.

Sparse data and unknown parameters: consider physics-informed inference.

Many repeated input fields: consider a trained surrogate or neural operator.

The right method depends on what is known and what must be inferred.

Why a PINN can fail

Wrong equation or boundary conditions

Poorly balanced losses and difficult optimisation

Too few checks near rapid changes

Insufficient smoothness for the required derivatives

A small training loss is not a certificate of a correct physical solution.

Validation beyond the training points

Prediction error on independent measurements or a trusted reference

Residuals at new coordinates

Initial and boundary-condition errors

Behaviour outside the imposed physics domain

Accuracy, constraint satisfaction and robustness answer different questions.

Design activity: a sensor goes offline

A device has temperature readings for one minute, then loses its sensor.

What should the network receive and predict?

Which physical assumptions must you check?

Which conditions and losses would you use?

How would you test the estimate after the sensor fails?

A good formulation starts with assumptions and available information.

Retrieval check

Does a continuous neural function automatically qualify as a PINN?

Does every collocation point need a measured target?

Does $y' + y = 0$ rule out $y = 0$?

Does a tiny residual guarantee an accurate physical prediction?

The function, the law and the conditions have different roles.

Practical class: cooling with a PINN

Compare a data-only model and a PINN with identical initial weights.

Implement the derivative, residual and combined loss.

Evaluate three regions and repeat a random seed.

Change the physics and explain what fails.

The goal is to explain the behaviour, not merely lower the loss.

Reading and implementation references

Raissi et al. (2019) — Physics-informed neural networks

Krishnapriyan et al. (2021) — PINN failure modes

Greydanus et al. (2019) — Hamiltonian neural networks

Li et al. (2021) — Fourier neural operator

PyTorch documentation — torch.autograd.grad

Full links and teaching guidance appear in the speaker notes.