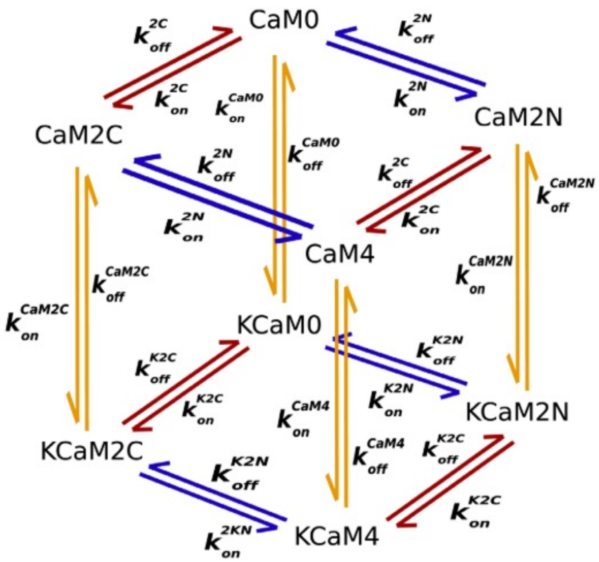
Scientific machine learning: methods to bridge scientific spatial and temporal modeling with machine learning

Michael W. Mahoney
LBNL, ICSI, and Dept of Statistics, UC Berkeley

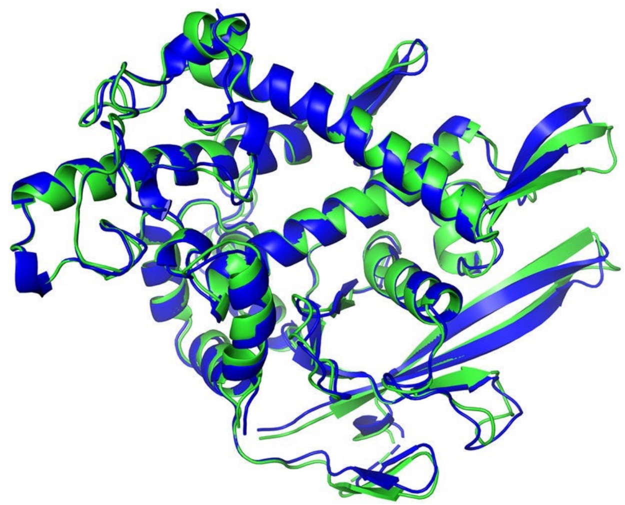
With: Aditi Krishnapriyan and many others

Modeling spatial and temporal behavior is crucial for progress in science and engineering

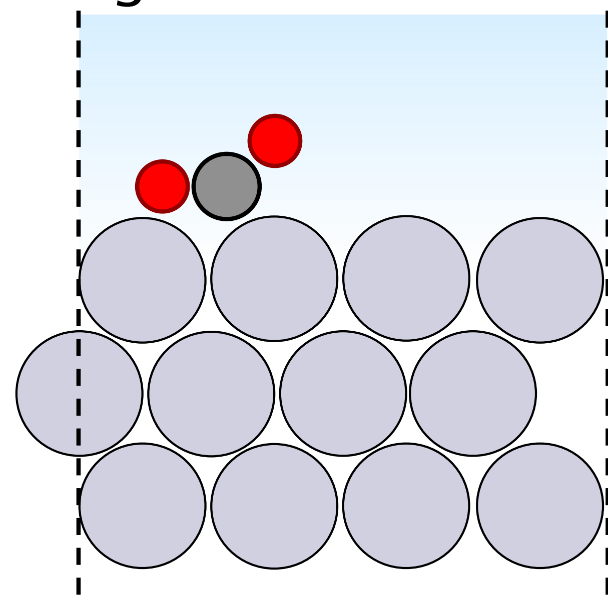


Zeman, Amundson. Continuous models for polymerization, AIChe Journal (1963).

Aris, Gavalas. On the theory of reactions in continuous mixtures, Phil Transactions of the Royal Society A (1966).



Suh, Arnold. A mathematical model for metal affinity protein portioning, Biotechnology and Bioengineering (1990).



Boudart. Electronic chemical potential in chemisorption and catalysis, J. Am. Chem. Soc (1952).

Gunter, Niemantsverdriet, Ribeiro, Somorjai. Surface science approach to modeling supported catalysts, Catalysis Reviews (1997).

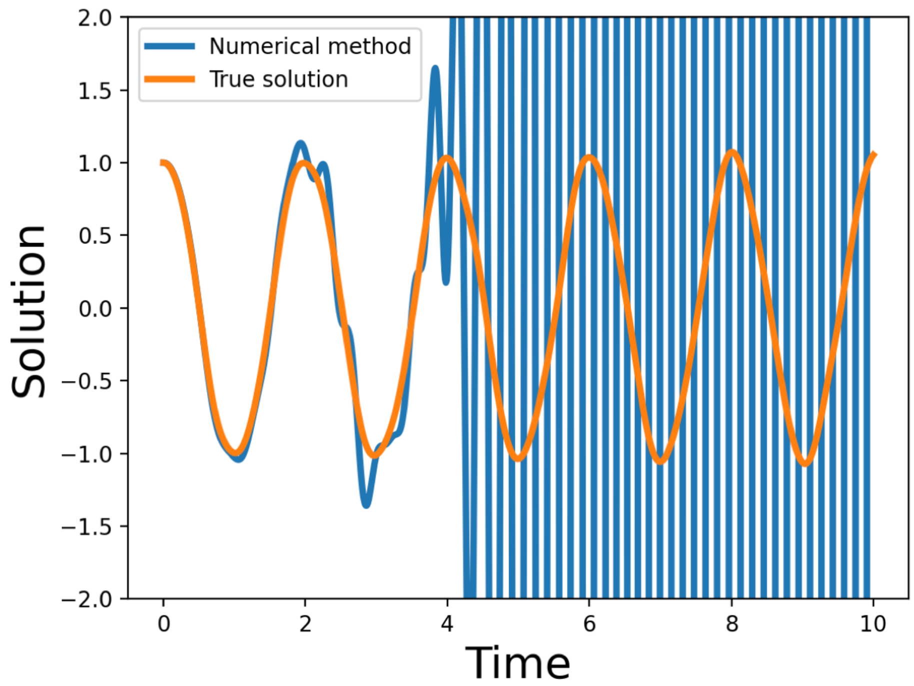
Challenge: Differential equations describing scientific phenomena often rely on a fine spatial/temporal discretization

$$\frac{dx}{dt} = f(x,t), x(t_0) = x_0$$

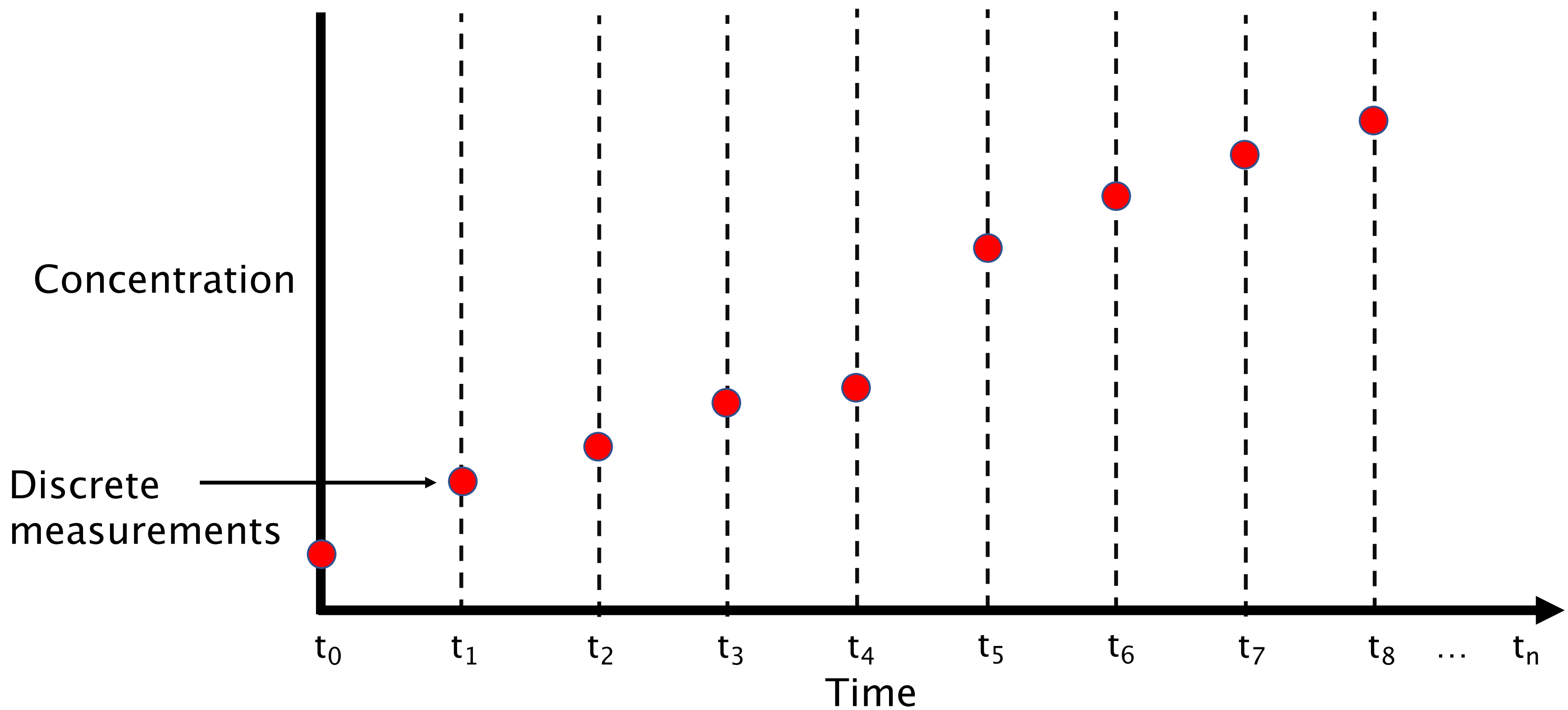
$$t_{i+1} = t_i + \Delta t$$

$$x_{i+1} = x_i + f(x_i, t_i)\Delta t$$

To converge: need smaller step sizes
→ **fundamentally limits modeling**

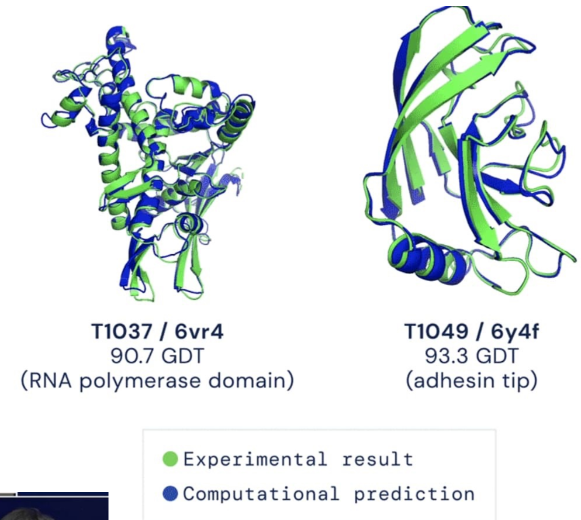


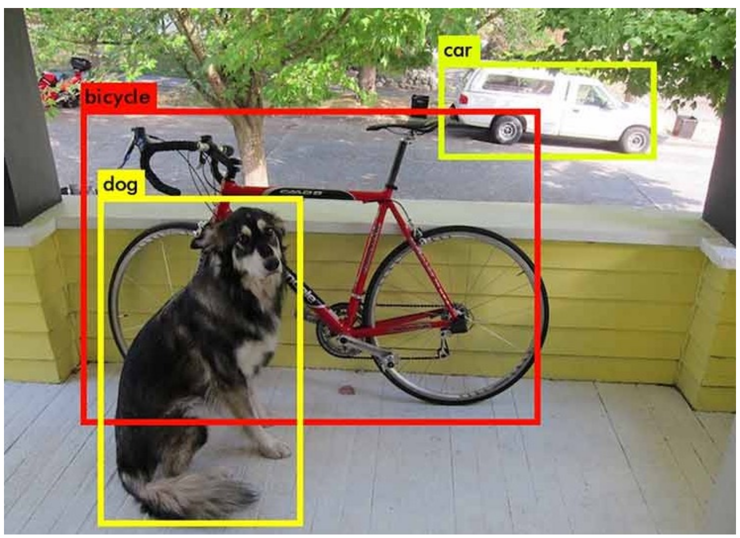
Challenge: Some phenomena are difficult to derive with differential equations

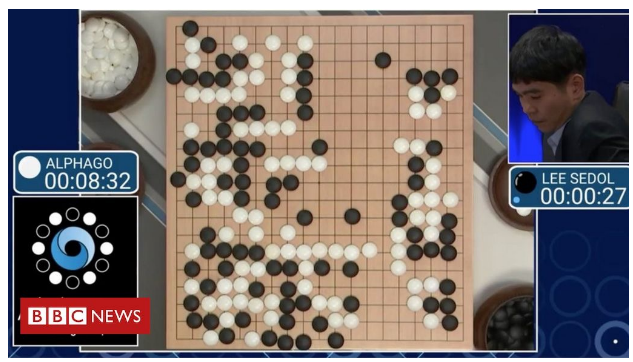


Machine learning has been very successful at solving challenging problems across fields

Great success across many areas at prediction tasks





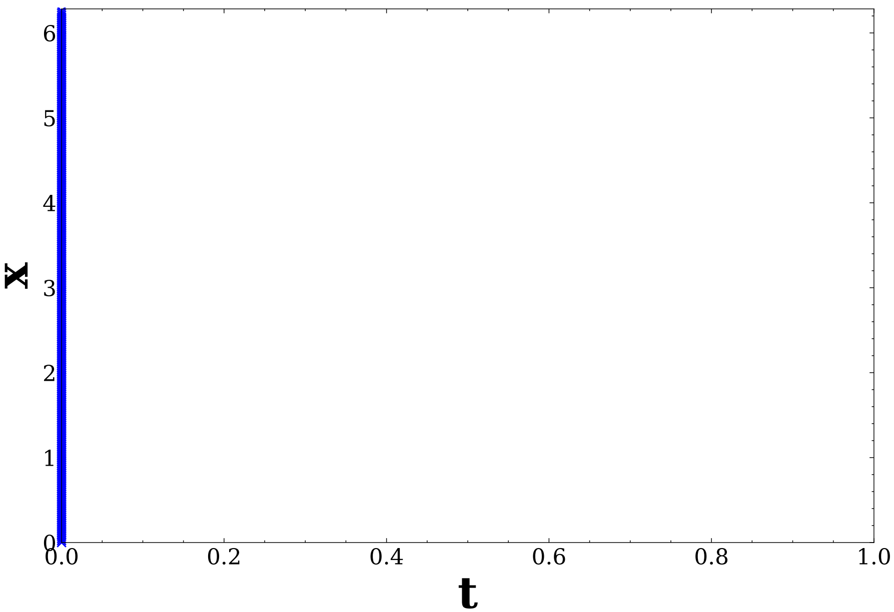


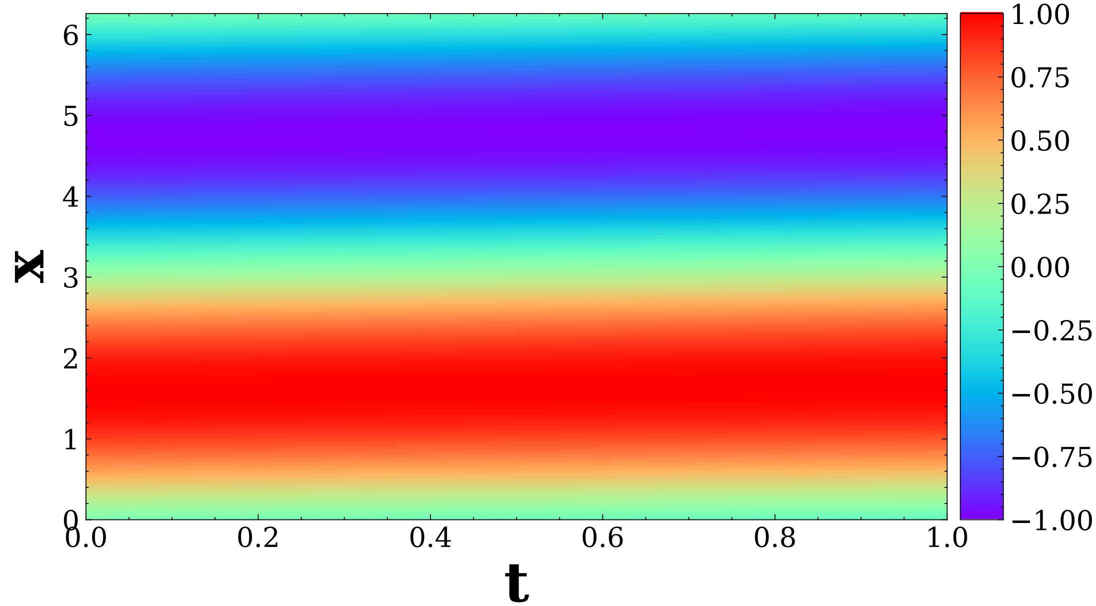
"Neural networks; universal approximation theorem": Cybenko (1989), Hornik (1991), Leshno (1993), etc.

Talk outline

- Incorporating differential equation structure into neural networks
- End-to-end differentiable simulations
- Other areas of scientific ML: uncertainty quantification, geometric deep learning

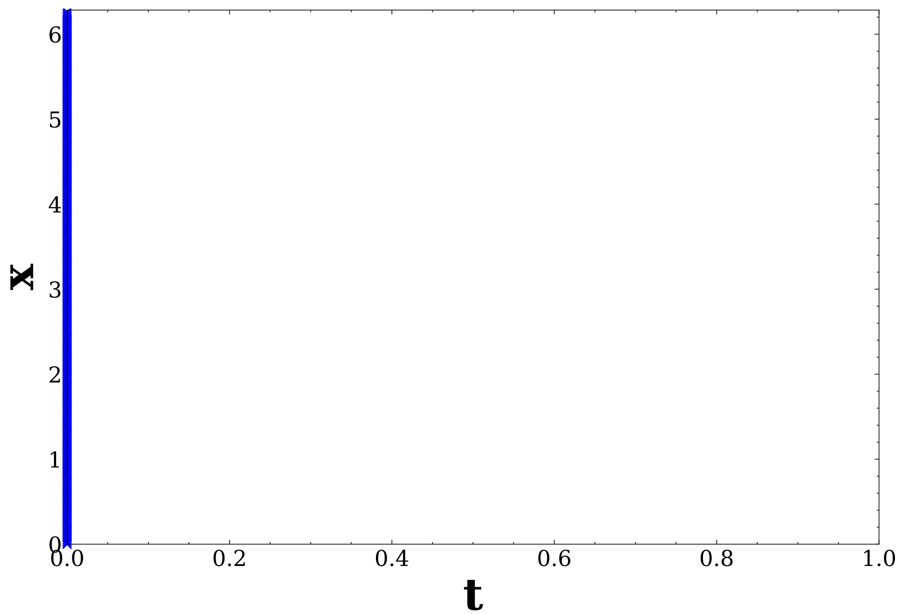
Standard problem: Train a model to extrapolate trajectories, with preliminary training data



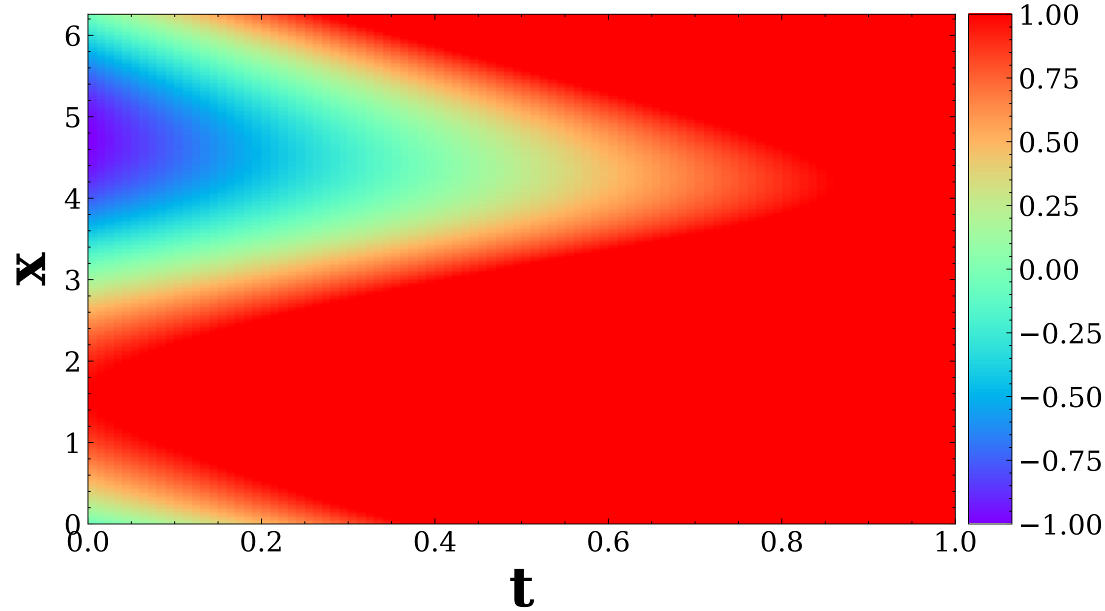


$$\mathcal{L} = min\frac{1}{N}\sum_{j}^{N} u_{pred}(\theta, x_j) - u_{true}(x_j))^2$$

The trained neural network (NN) can't generalize



Training data (initial condition)

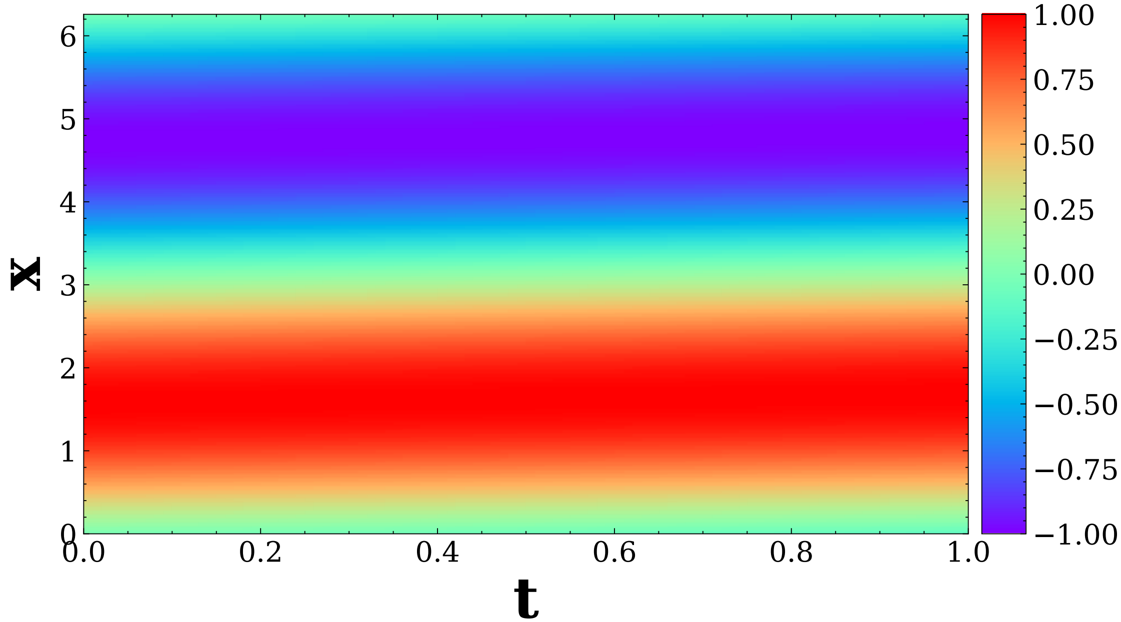


Predicted solution

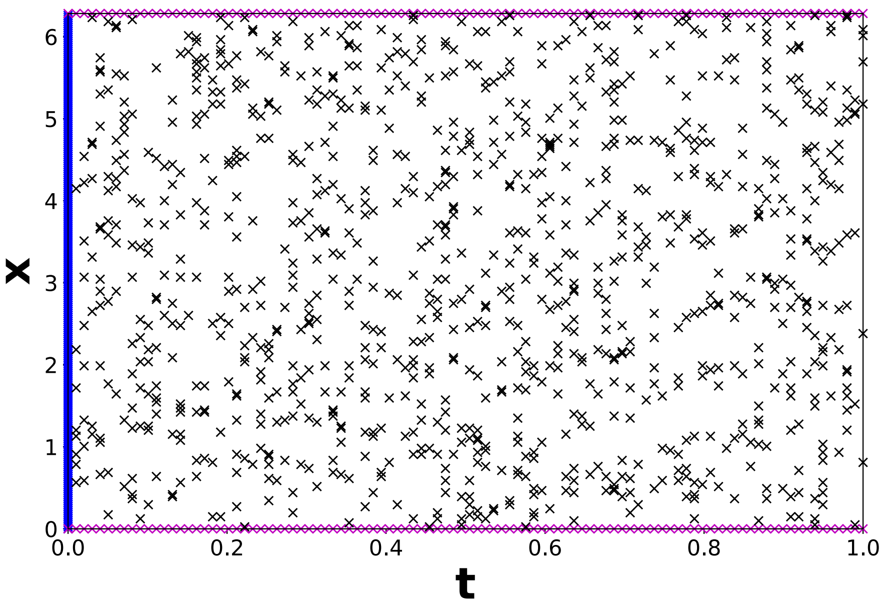
$$\mathcal{L} = \frac{1}{N}\sum_{j}^{N} u_{pred}(\theta, x_j) - u_{true}(x_j))^2$$

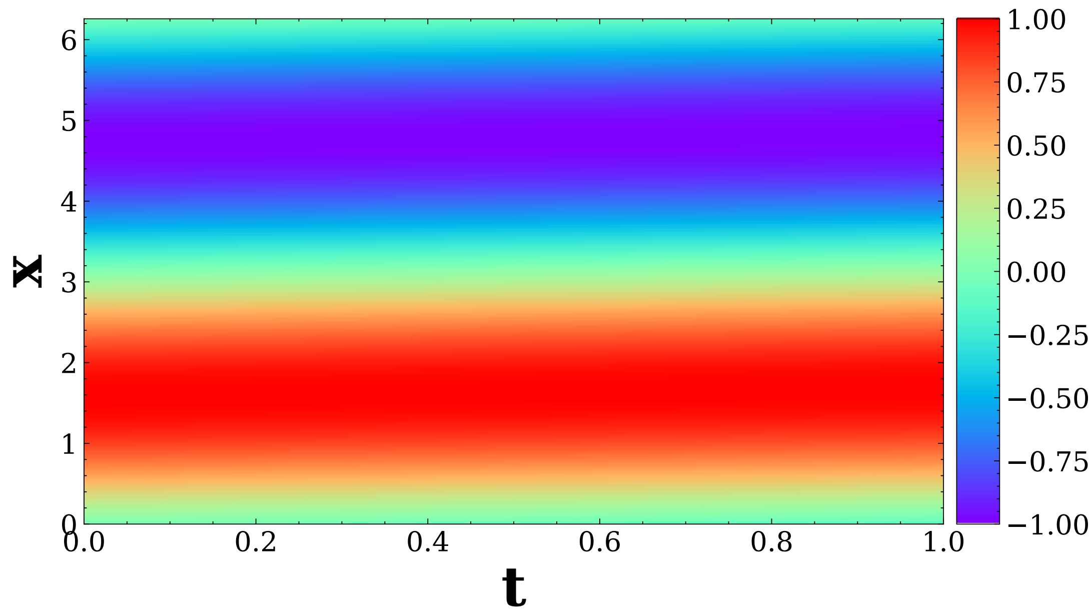
We have an idea of the "physics" of the process: it models transport phenomena

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0$$



Physical laws can be incorporated into the ML process through “physics-informed” neural networks





$$\mathcal{L} = \frac{1}{N}\sum_{j}^{N} u_{pred}(\theta, x_j) - u_{true}(x_j))^2 + \frac{1}{N_f}\sum_{k=1}^{N_f} \lambda_k \Big(\frac{\partial u_{pred}(\theta, x_k)}{\partial t} + \beta \frac{\partial u_{pred}(\theta, x_k)}{\partial x}\Big)^2 + \mathcal{L}_{\mathcal{B}}$$

Systematic investigation of scientific phenomena using physics-informed neural networks

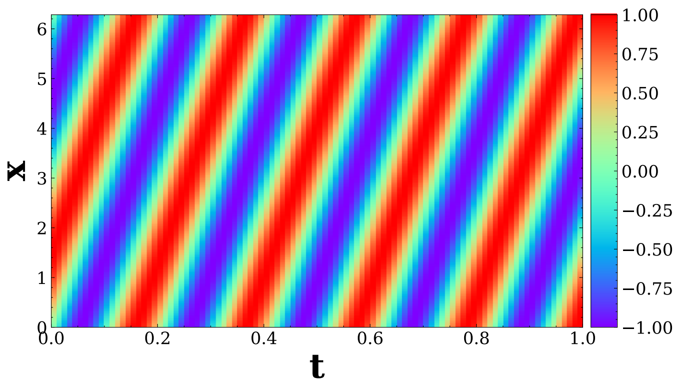
How well does this approach generalize
to relevant engineering problems?

- Convection (transport phenomena) $$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0$$
- Reaction $$\frac{\partial u}{\partial t} - \rho u(1-u) = 0$$
- Reaction-diffusion $$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0$$

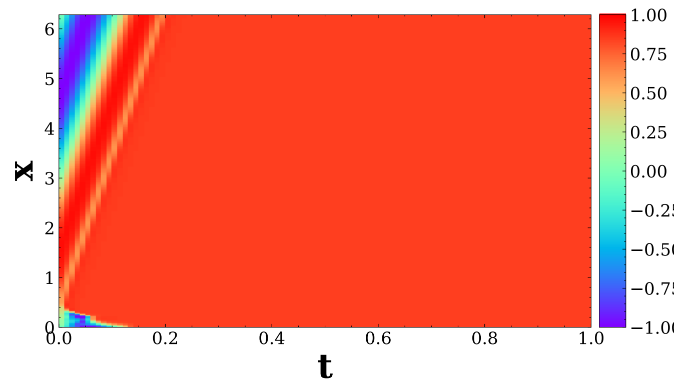
At certain ODE/PDE coefficients, the physics-informed NN approach fails to find an answer

Convection:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0$$



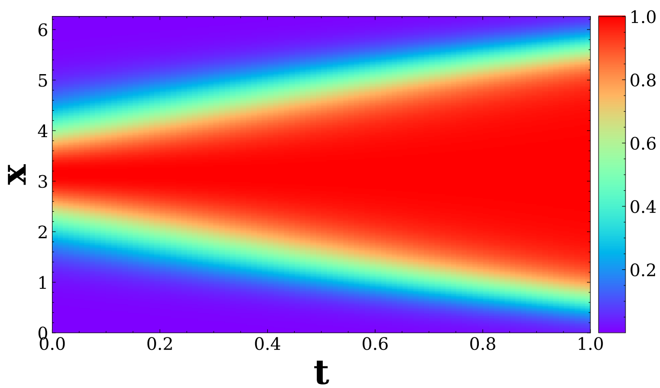
Exact solution, β = 30



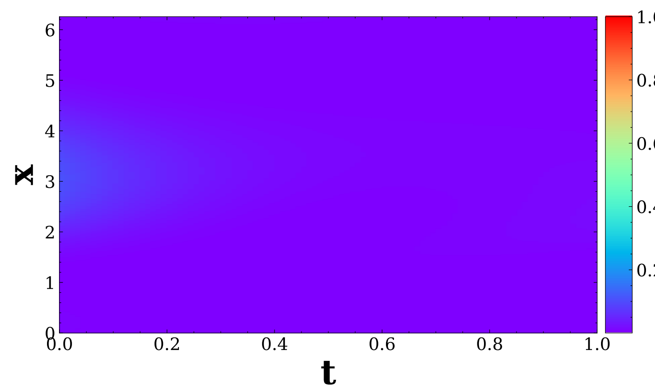
PINN predicted solution, β = 30

Reaction:

$$\frac{\partial u}{\partial t} - \rho u(1-u) = 0$$



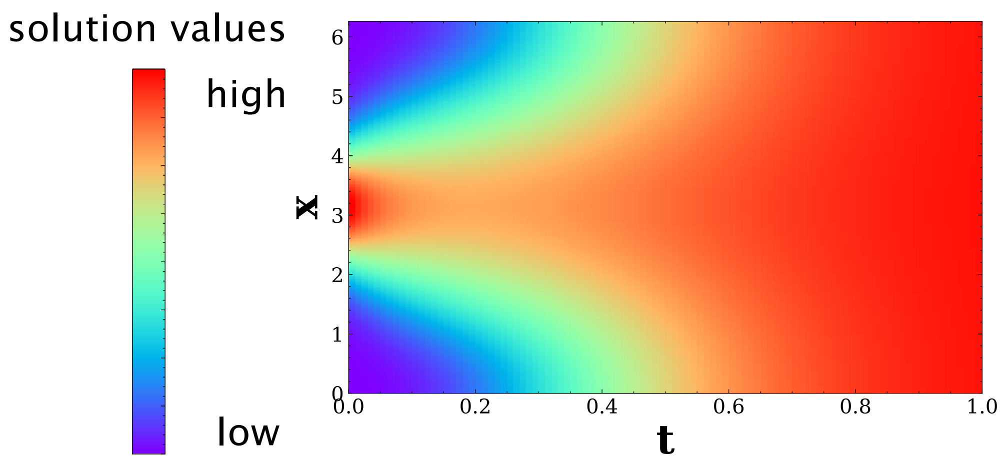
Exact solution, ρ = 5



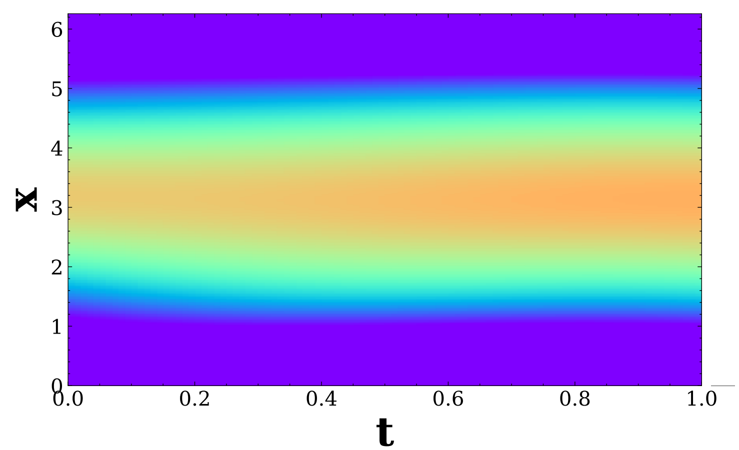
PINN predicted solution, ρ = 5

Example: At certain ODE/PDE coefficients, the physics-informed NN fails to find an answer

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0$$

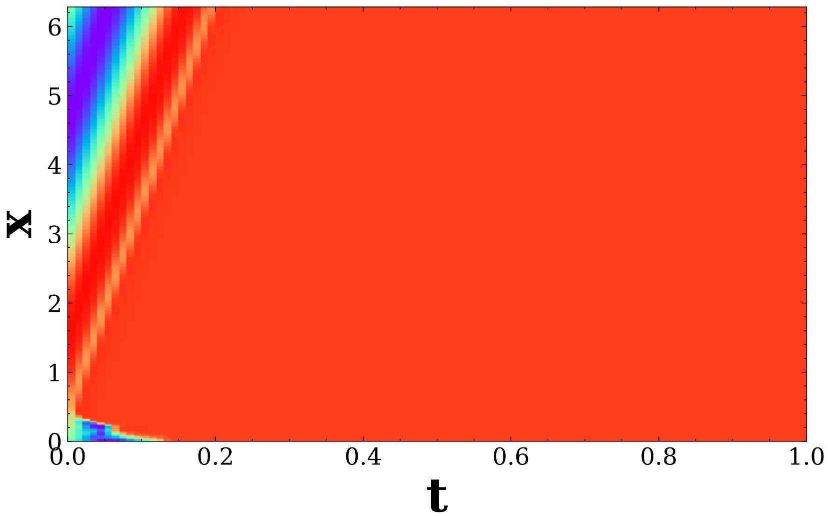


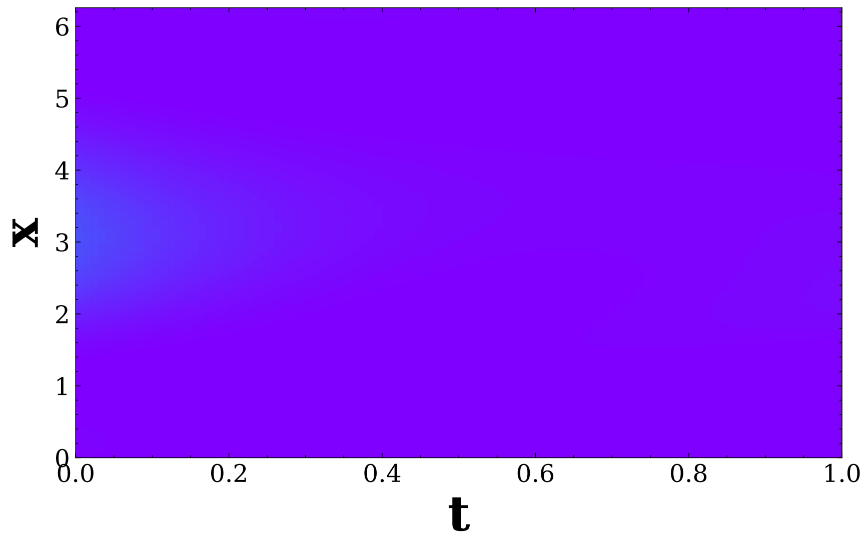
Actual solution, ρ = 5, ν=3



Physics-informed NN predicted solution, ρ = 5, ν=3

Now we understand the failures:
address them by changing the learning paradigm

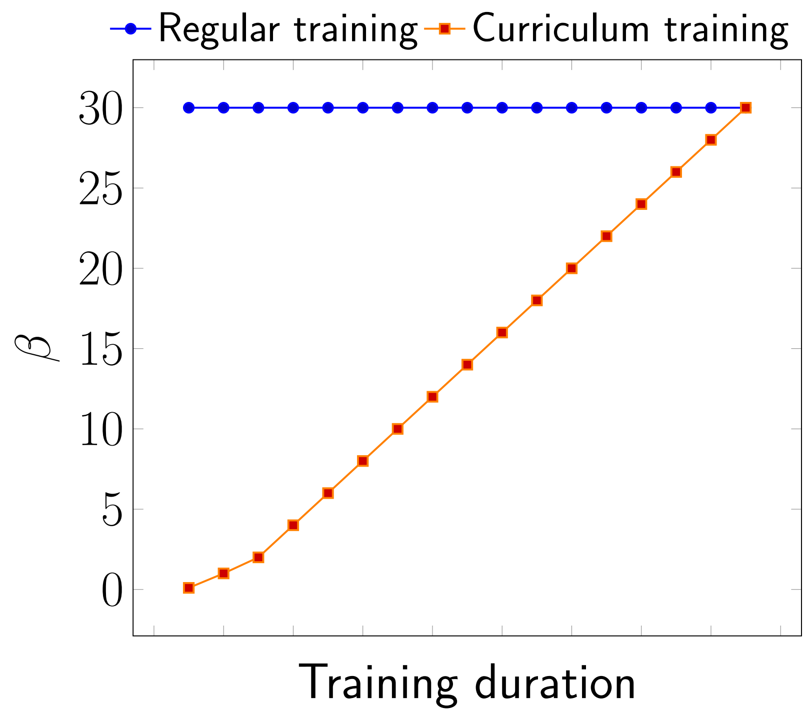




Curriculum regularization:
Start with simple ODE/PDEs, then make the problem harder

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x}$$

The physics-informed NN performed poorly at higher β coefficients



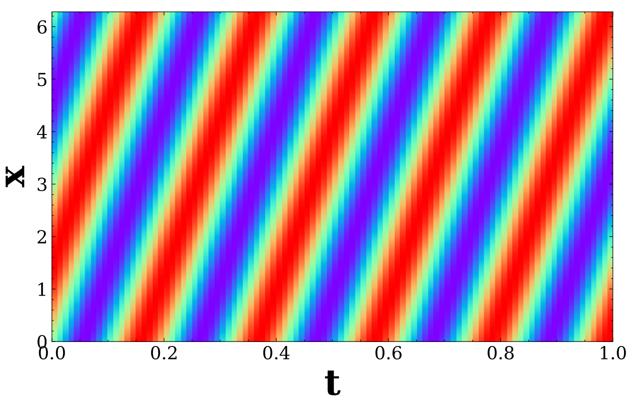
Curriculum regularization: decreases error by orders of magnitude, captures “sharp” features in the solution

Convection:

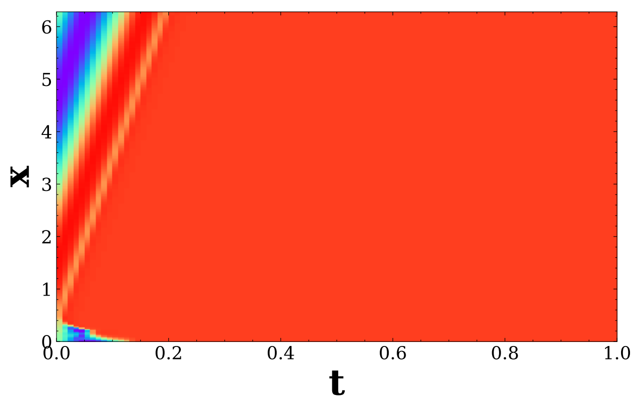
solution

$$\frac{\partial \boxed{u}}{\partial t} + \boxed{\beta}\frac{\partial u}{\partial x} = 0$$

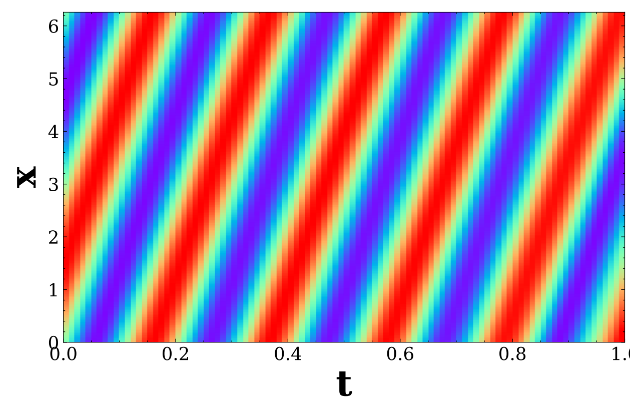
wave speed



Actual solution, β = 30



Regular training predicted solution, β = 30

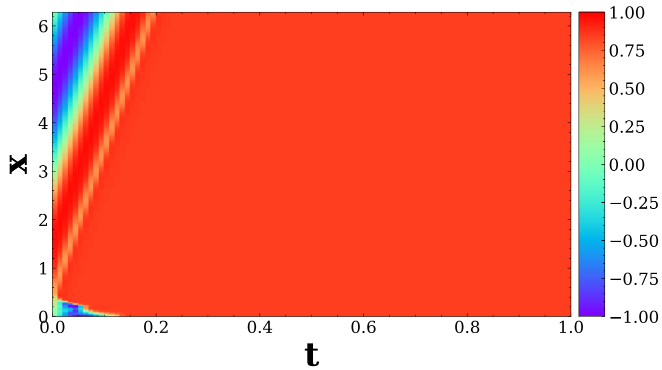


Curriculum regularization solution, β = 30

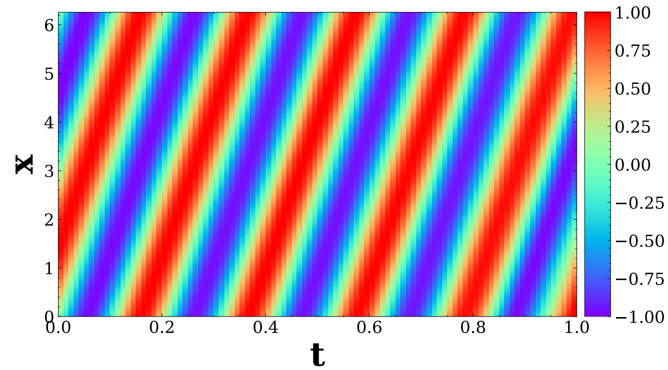
Curriculum regularization decreases error by orders of magnitude, captures “sharp” features in the solution

Convection:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0$$



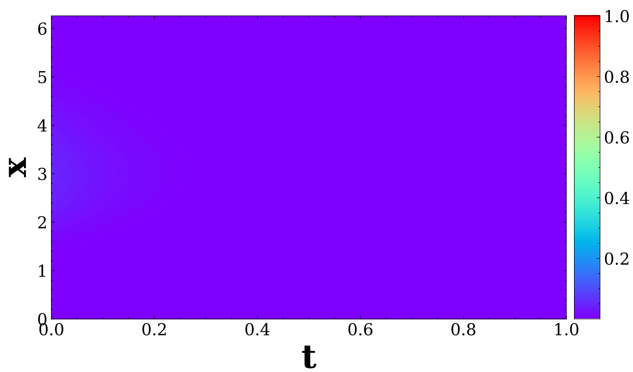
Regular PINN solution, β = 30



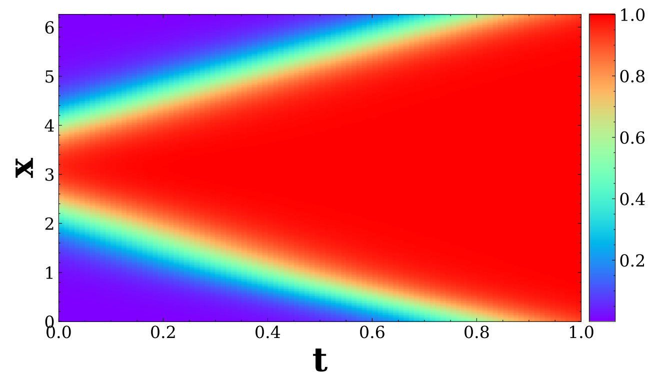
Curriculum regularization solution, β = 30

Reaction:

$$\frac{\partial u}{\partial t} - \rho u(1-u) = 0$$

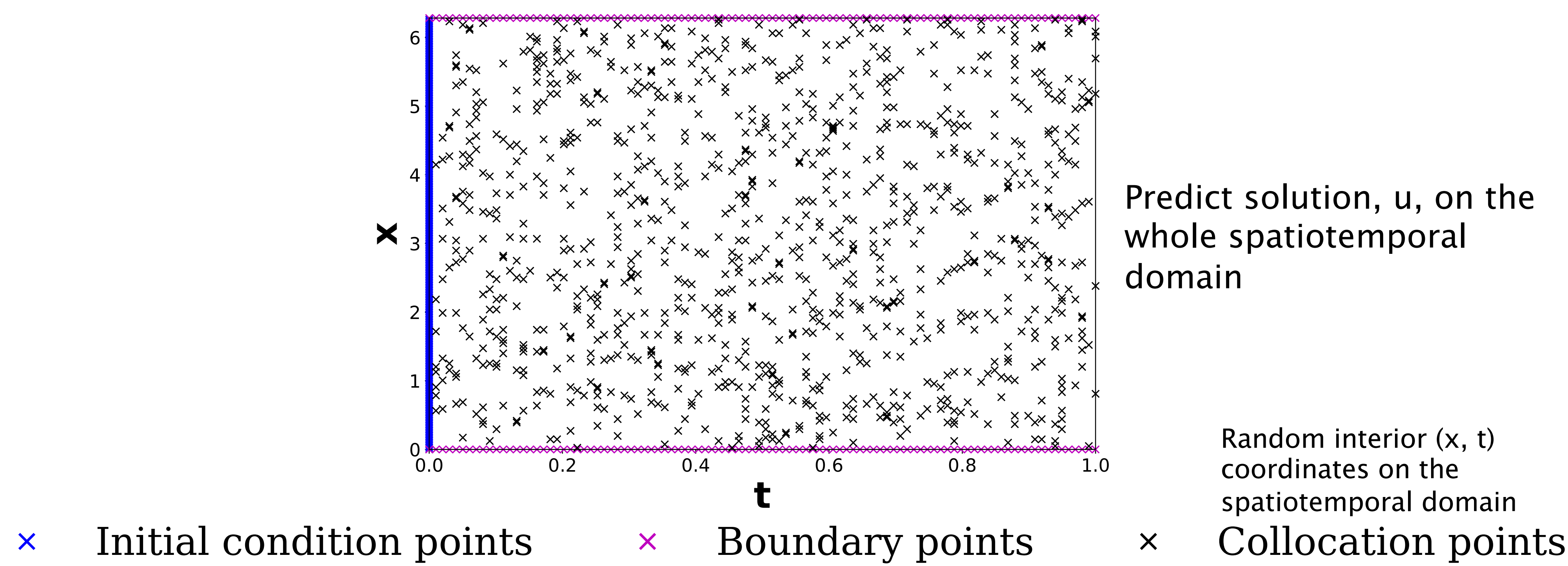


Regular PINN solution, ρ = 10



Curriculum regularization solution ρ = 10

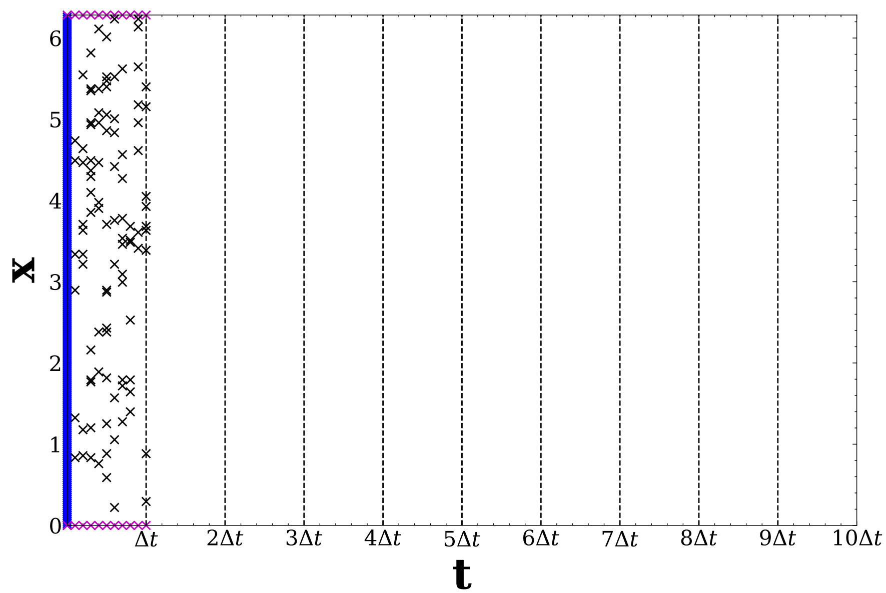
Regular physics-informed training tries to predict the solution for the whole state space



Predict solution, u, on the whole spatiotemporal domain

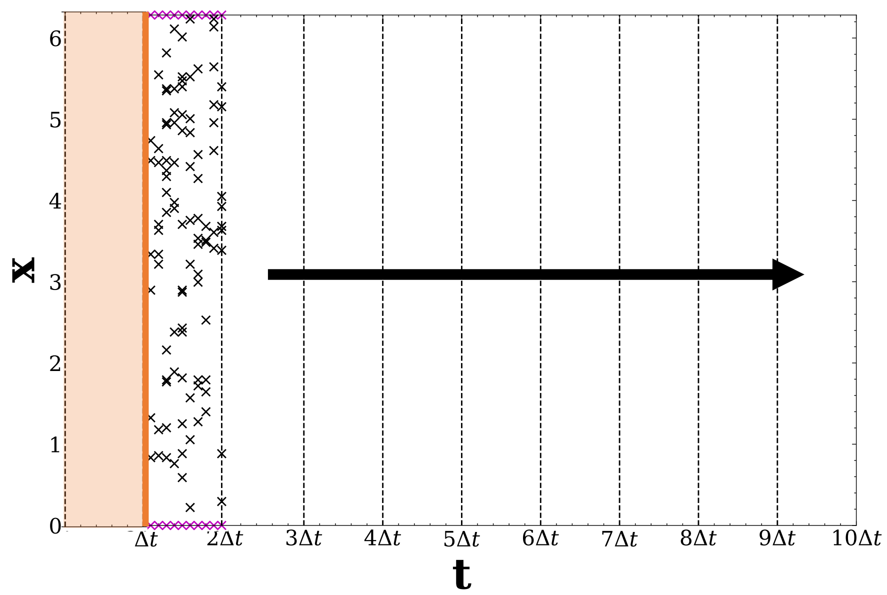
Random interior (x, t) coordinates on the spatiotemporal domain

Sequence-to-sequence learning: solve one "time segment" at a time



Exact initial condition

Sequence-to-sequence learning: solve one "time segment" at a time



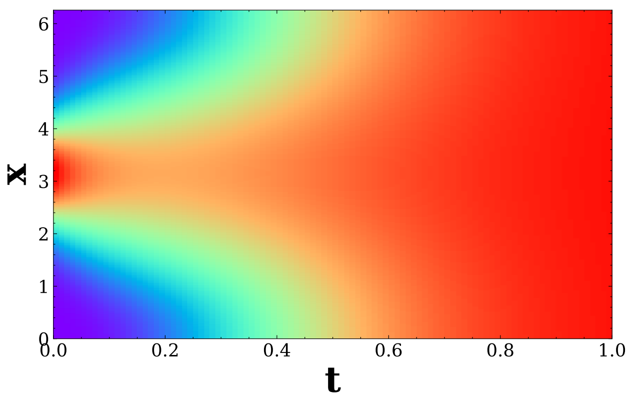
Predicted initial condition for the next time block

Already predicted solution

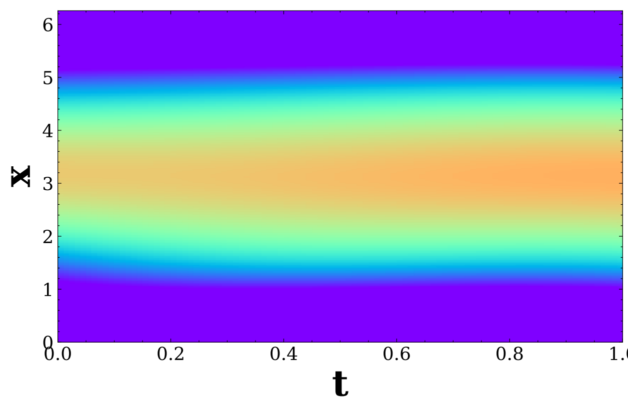
We still assume that we only have the exact solution at t=0 (initial condition).

Sequence-to-sequence learning greatly decreases error for all systems of study

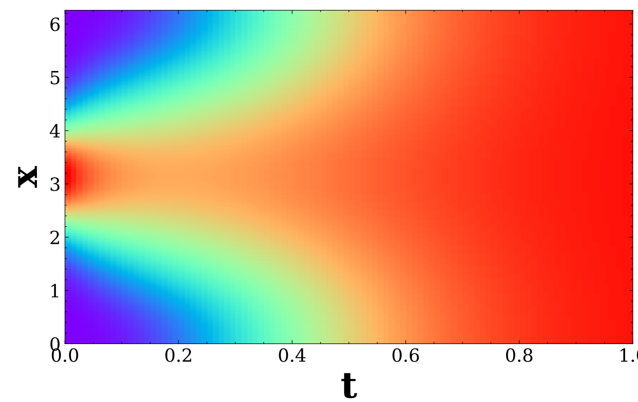
$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0$$



Exact solution, ρ = 5, ν=3



Regular training solution, ρ = 5, ν=3



Seq2seq solution, ρ = 5, ν=3

Summary on the challenges with incorporating physical laws into NNs, and changing the learning paradigm

- Physics-informed NNs is a promising method that works in certain cases, but can fail easily for common scientific problems

- The PDE constraint can make the loss landscape very hard to optimize

- Changing the learning paradigm can greatly decrease error: curriculum regularization, sequence-to-sequence learning

Solving a new parameterized differential equation each time is inefficient

Parameter 1 → Model $\mathcal{F}\left(a_1\frac{\partial u}{\partial t}, b_1\frac{\partial u}{\partial x}, c_1\frac{\partial^2 u}{\partial x^2}, \ldots\right)$ → Solution 1

Parameter 2 → Model $\mathcal{F}\left(a_2\frac{\partial u}{\partial t}, b_2\frac{\partial u}{\partial x}, c_2\frac{\partial^2 u}{\partial x^2}, \ldots\right)$ → Solution 2

...

Solution: solve the family of equations

Parameter 1	$\rightarrow$	Model	$\rightarrow$	Solution 1
Parameter 2	$\rightarrow$	$\mathcal{F}\left(a\frac{\partial u}{\partial t}, b\frac{\partial u}{\partial x}, c\frac{\partial^2 u}{\partial x^2}, \ldots\right)$	$\rightarrow$	Solution 2
Parameter 3	$\rightarrow$		$\rightarrow$	Solution 3
Parameter n	$\rightarrow$		$\rightarrow$	Solution n

Learn a set of basis functions that can map to all the solutions

Before:
physical laws were only enforced approximately ("soft" constraint)

$$\mathcal{L}(u_\theta) = \frac{1}{2}(\hat{u}(\theta, x, t) - u(x, t))^2$$

Loss between NN predicted solution and true, observed solution

$$\min_\theta \mathcal{L}(u_\theta) \text{ s.t. } \mathcal{F}(u_\theta) = 0$$

Fit to observation data such that the PDE constraint is respected

$$\min_\theta \mathcal{L}(u_\theta) + \lambda_{\mathcal{F}} \mid\mid \mathcal{F}(u_\theta) \mid\mid_2^2$$

"Soft" constraint: penalty if the PDE constraint is not satisfied

Solution: enforce the physical laws ***exactly*** ("hard" constraint)

$$\mathcal{L}(u_\theta) = \frac{1}{2}(\hat{u}(\theta, x, t) - u(x, t))^2$$

Loss between NN predicted solution and true, observed solution

$$\min_\theta \mathcal{L}(u_\theta) \ \text{ s.t. } \boxed{\mathcal{F}(u_\theta) = 0}$$

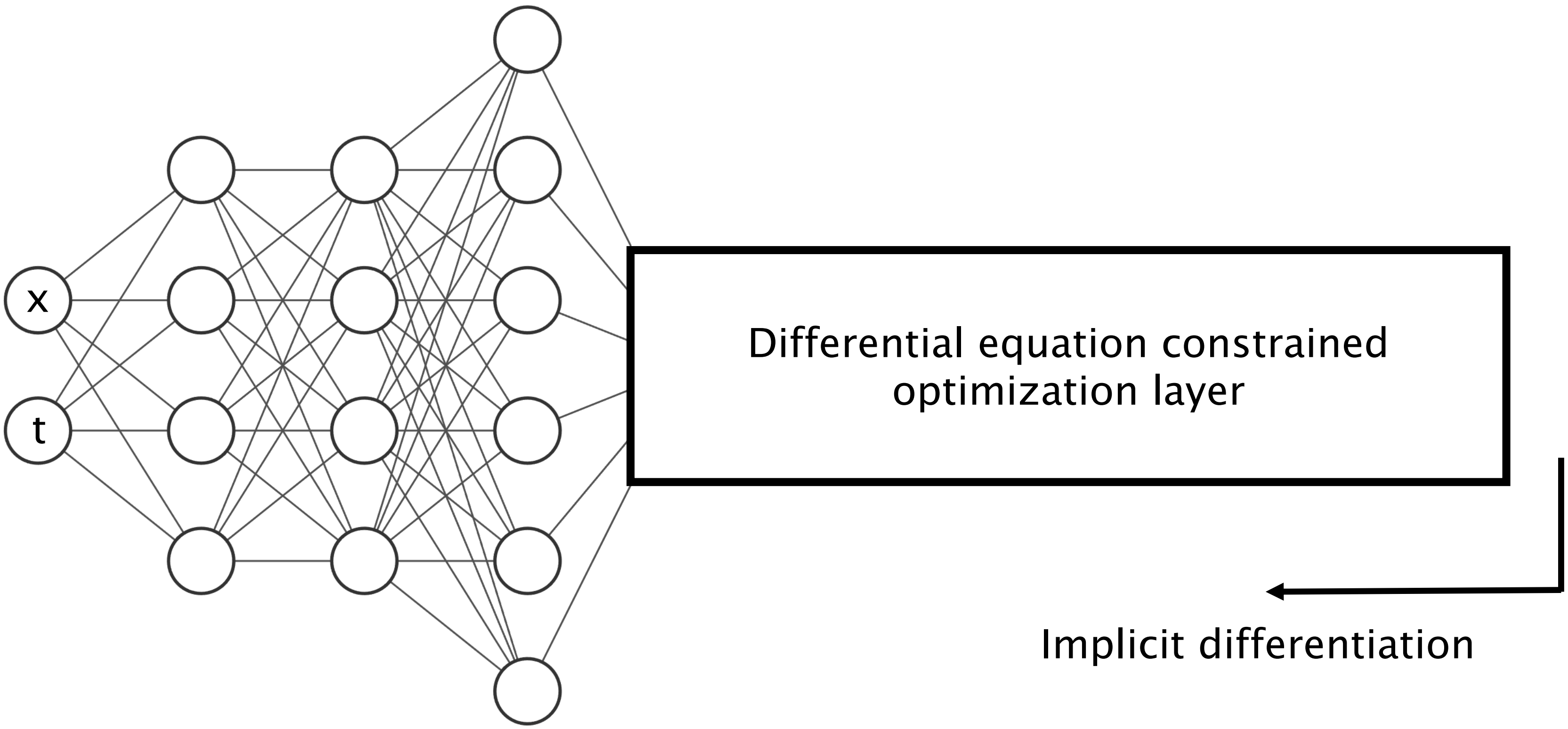
Fit to observation data such that the PDE constraint is respected

$$\cancel{\min_\theta \mathcal{L}(u_\theta) + \lambda_{\mathcal{F}} \, || \, \mathcal{F}(u_\theta) \, ||_2^2}$$

"Soft" constraint: penalty if the PDE constraint is not satisfied

Enforce differential equations ***exactly*** through constrained optimization

NN architecture incorporating constrained optimization



Testing:
Predicted solutions for **many** PDEs

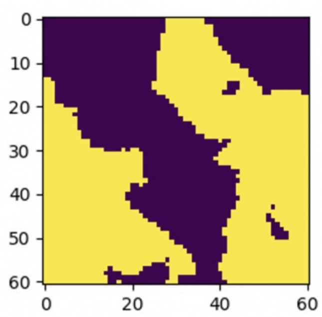
Goal: Predict solutions on the whole spatial domain without solution data

2D Darcy flow:

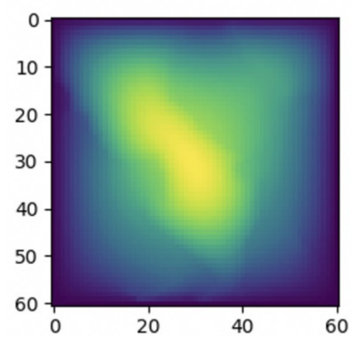
$$-\nabla \cdot (a(x)\nabla u(x)) = f(x) \;\; x \in (0,1)^2$$
$$u(x) = 0 \;\; x \in \partial(0,1)^2$$

Train: Set coefficients a(x) randomly, train by enforcing PDE constraint (no solution data)
Test: Predict solution for problems with different values of a(x) (not from the training set)

Problem setup: Given different a(x), predict the solutions of all PDEs in the test set, on the whole spatial domain.

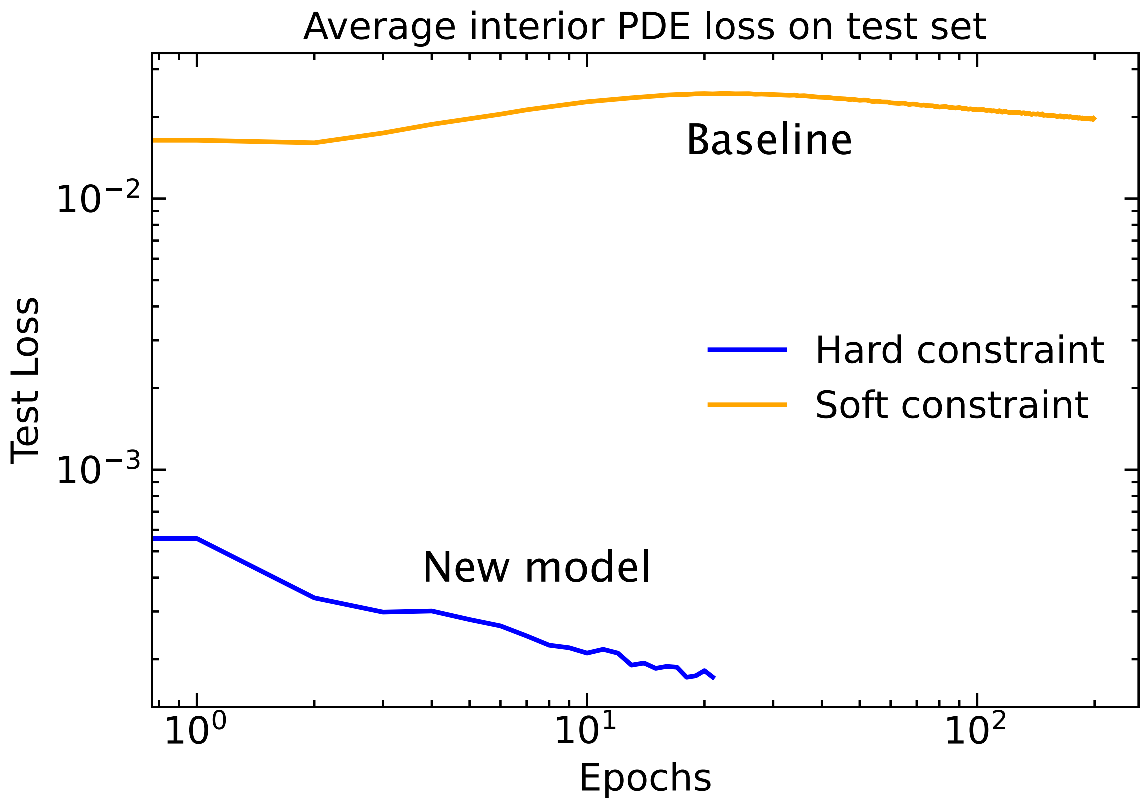


PDE parameters

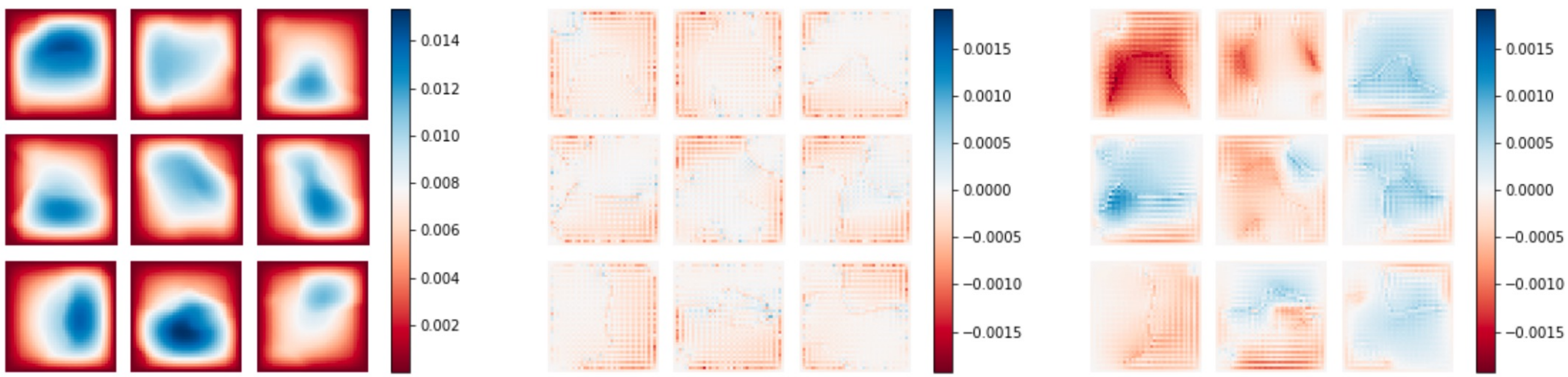


PDE solution

Within minutes of training: fast inference/low prediction error on full spatial domain for 50+ different PDEs



Hard-constrained network architecture is much closer to the numerical solution



(a) Target (b) Hard-constrained difference (c) Soft-constrained difference

NN constrained architecture applied to another problem

1D convection (transport phenomena):

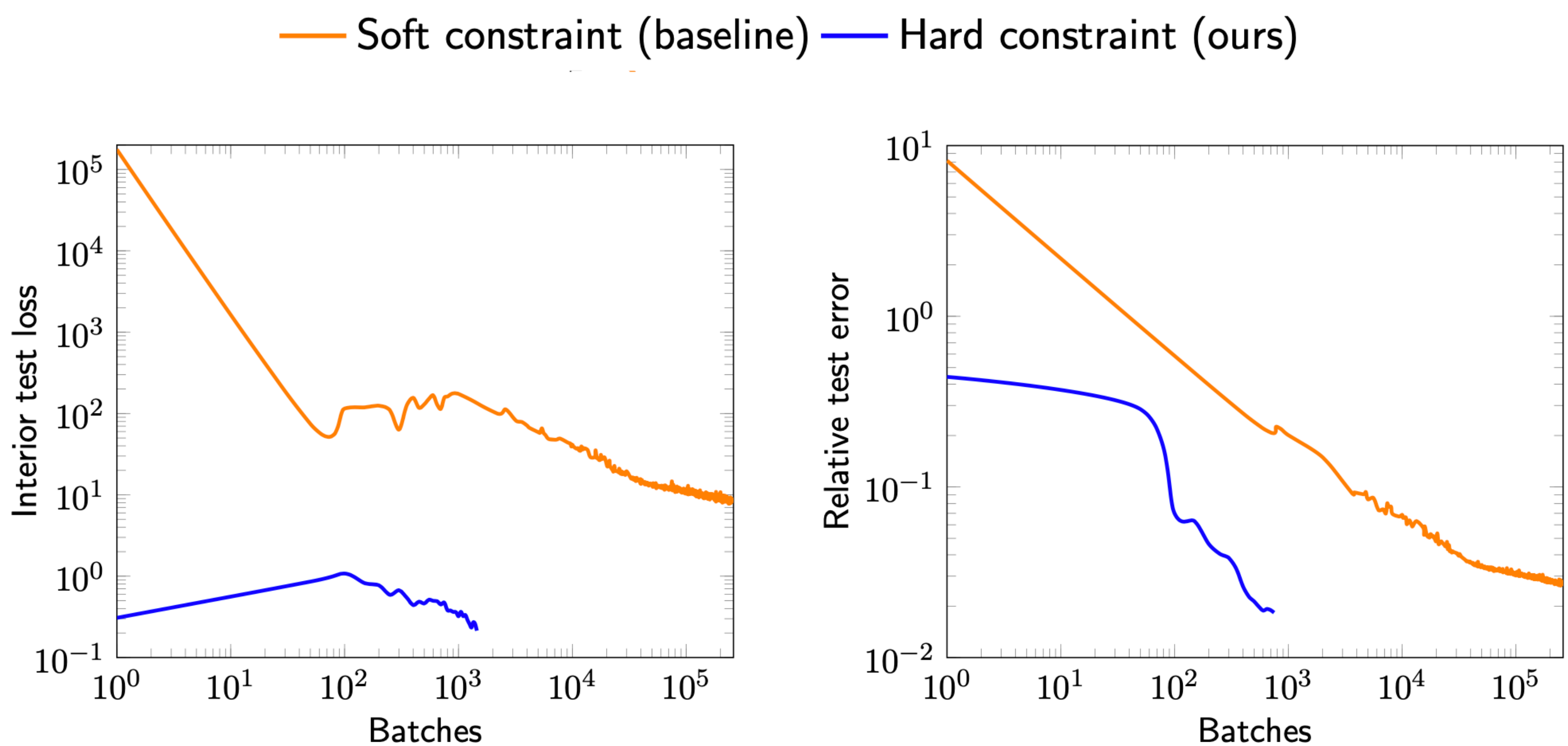
$$\frac{\partial u(x,t)}{\partial t} + \beta(x)\frac{\partial u(x,t)}{\partial x} = 0$$

Boundary condition: $u(x=0,t) = sin(\frac{\pi}{2}t),$

Initial condition: $u(x,t=0) = sin(\pi x)$

Problem setup: Given only boundary and initial conditions and random β(x), predict the solutions of all PDEs in the test set, on the whole spatiotemporal domain.

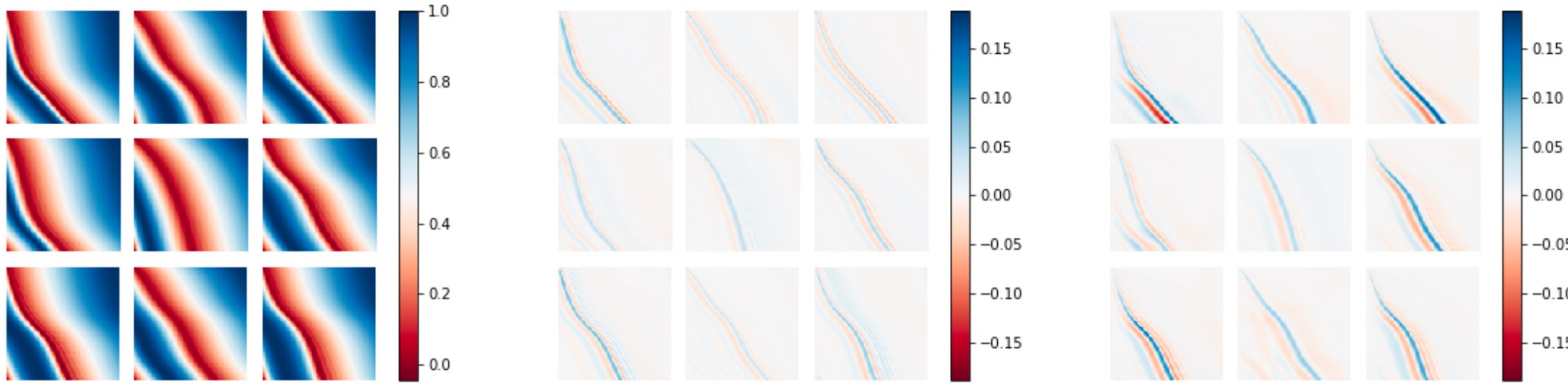
Within minutes of training: fast inference/low prediction error on full spatiotemporal domain for 50+ different PDEs



(a) PDE residual loss on test set

(b) Relative error on test set

Hard-constrained architecture is much closer to the numerical solution



(a) Target (b) Hard-constrained difference (c) Soft-constrained difference

Summary on NN architecture building in hard constraints

- **Previous limitations**:
  - Not generalizing to many different solutions
  - Physical laws aren't enforced exactly

- **Now**:
  - Generalize to many different solutions
  - Enforce physical **laws exactly**
  - Can solve new problems (with different initial conditions, different parameter coefficients, etc.) quickly and accurately

Learning discrete points with neural networks

$$x_t \to x_{t+1}$$

Input – output mapping

$$x_{t+1} = \mathcal{N}(\theta, x_t)$$

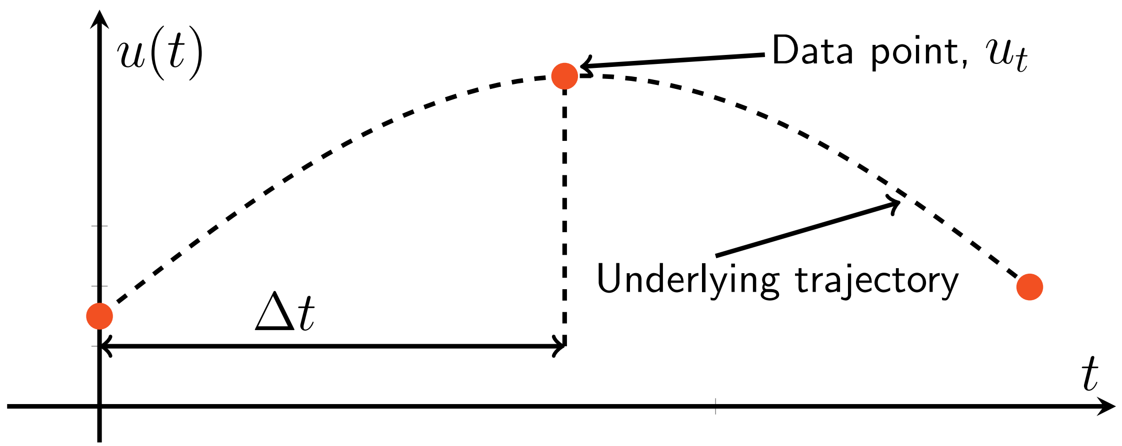
Use an NN to approximate the next step

Issues:

- Compounding errors over time

Measurements come from an underlying continuous trajectory

$$\frac{dx(t)}{dt} = F(x(t))$$

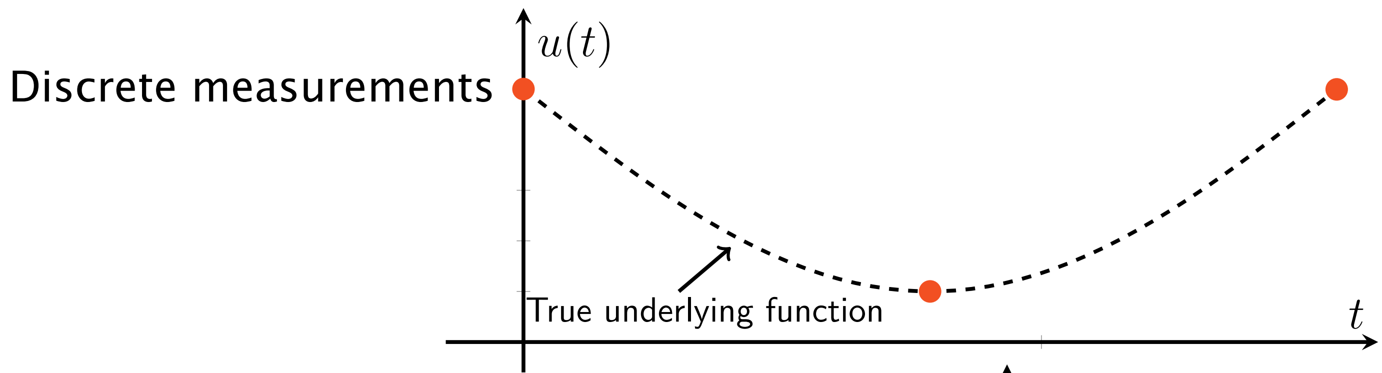


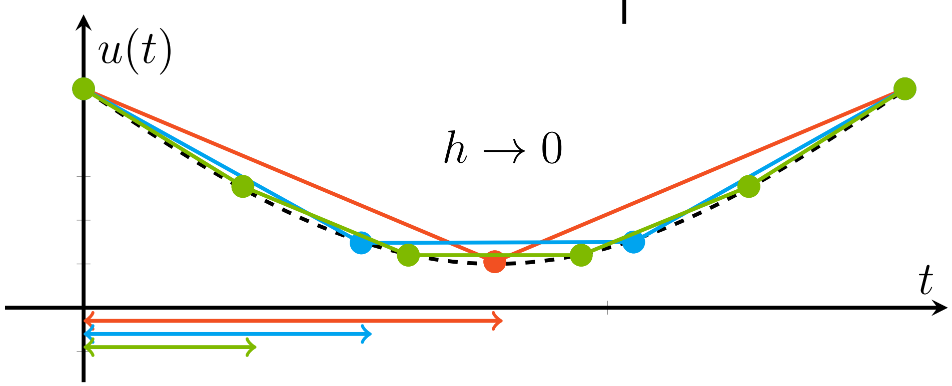
Training data

Can we learn continuous approximations from discrete points?

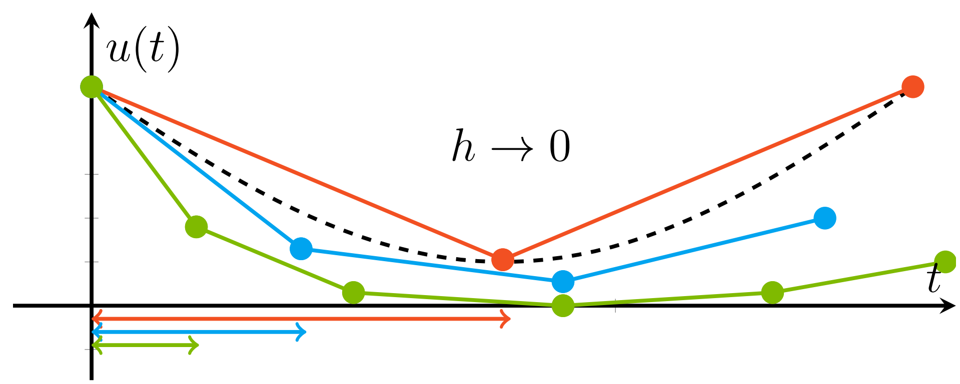
$$
\begin{aligned}
x_{t+1} &= x_t + \int_{t_0}^{t_0+h} \mathcal{N}(\theta, x(t))\mathrm{dt} \\
x_{t+1} &= x_t + \mathrm{ODESolver}[\mathcal{N}(\theta, x_t)] \\
x_{t+1} &= x_t + h\mathcal{N}(\theta, x_t)
\end{aligned}
$$

Learning discrete points versus continuous dynamics





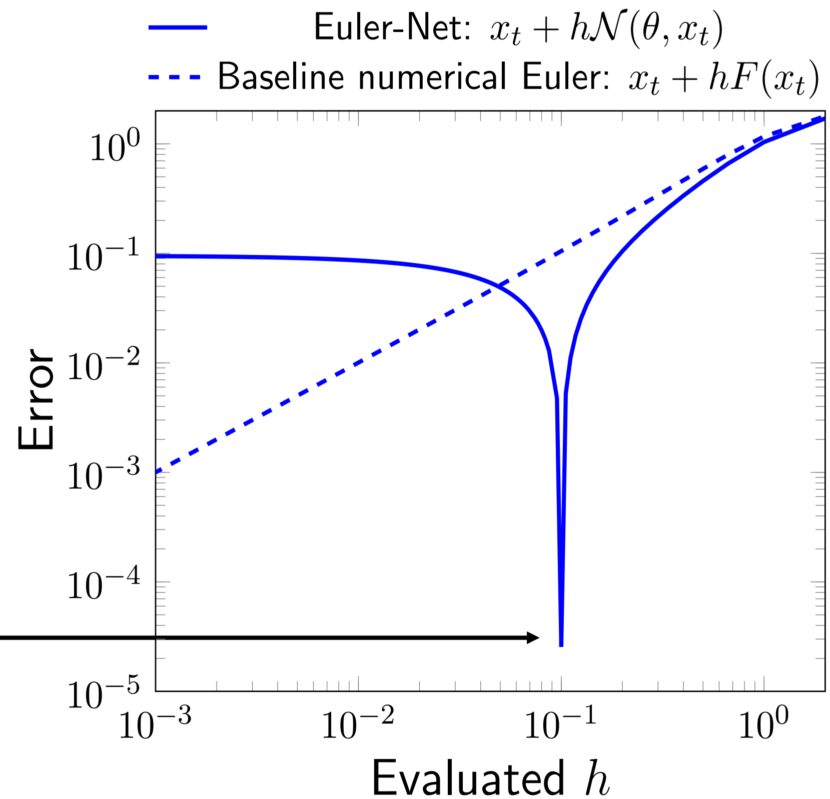
Continuous model: as h → 0, we approach the true underlying trajectory



Discrete-only model: on the correct trajectory when h = Δt, but wrong as h → 0

We can verify continuity through convergence tests

$$x_{t+1} = x_t + h\mathcal{N}(\theta, x_t)$$



Low error only at h = Δt = 0.1, the trained NN has **not** learned a continuous model (and fails the convergence test)

Train on Δt = 0.1, evaluate at **all** h step sizes and compute error at some final T

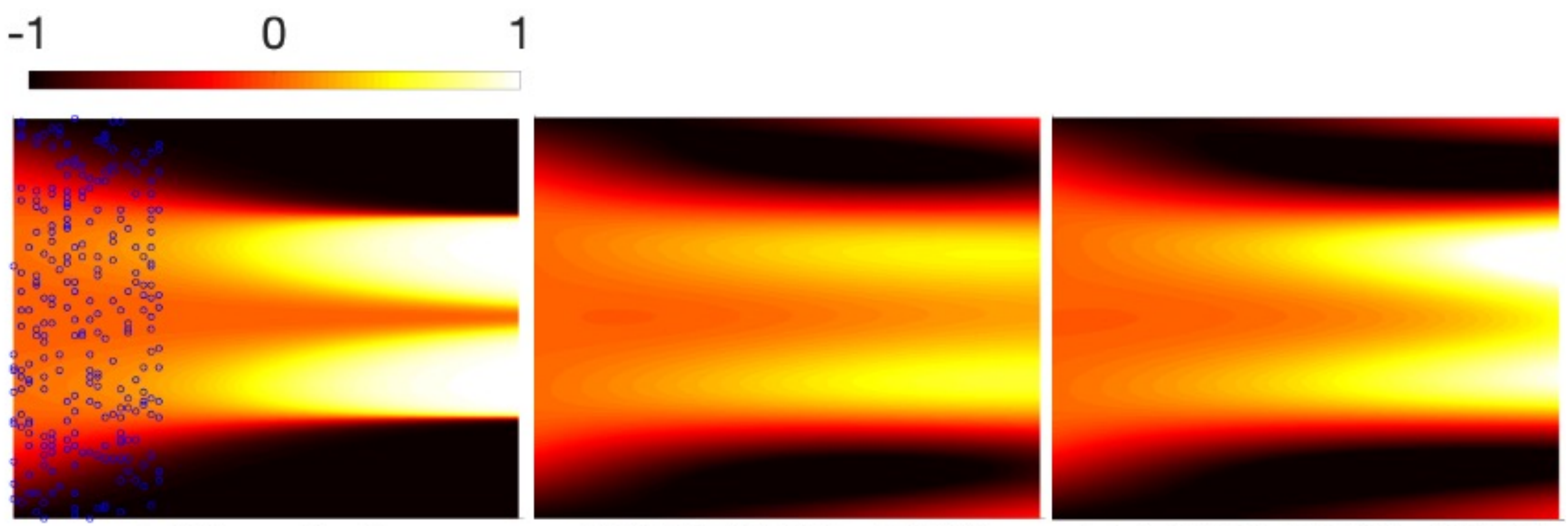
Looking ahead to other directions…

- **Uncertainty quantification**: What if we have noise in our data, and/or don’t know the exact form of our differential equations?
- **Exploiting symmetry**: What if we know that there are important symmetries that need to be respected in our scientific systems?

Allen-Cahn reaction-diffusion system

$$\frac{\partial u}{\partial t} - \nu \frac{\partial u^2}{\partial x^2} + g(x,t) = 0$$

$$g(x,t) = \rho u^2(u-1)$$



Actual solution (and sampled training data)

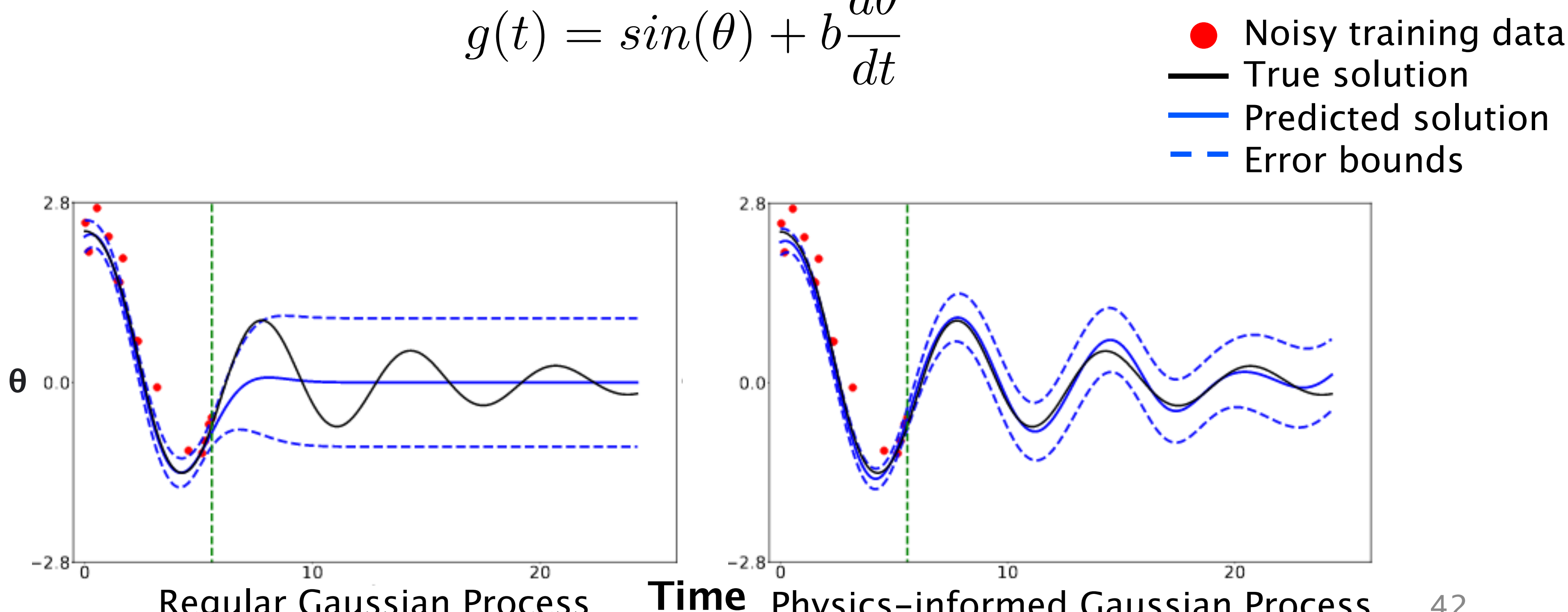
Regular Gaussian Process

Physics-informed Gaussian Process

Non-linear pendulum example

$$\frac{d^2\theta}{dt^2} + g(t) = 0$$

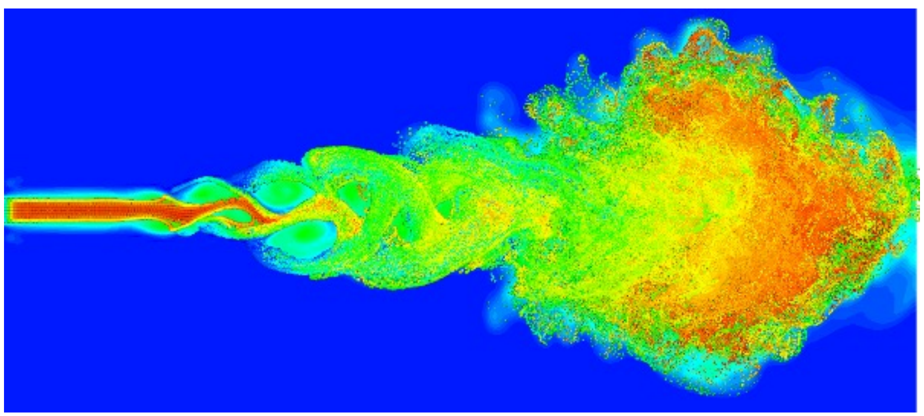
$$g(t) = sin(\theta) + b\frac{d\theta}{dt}$$

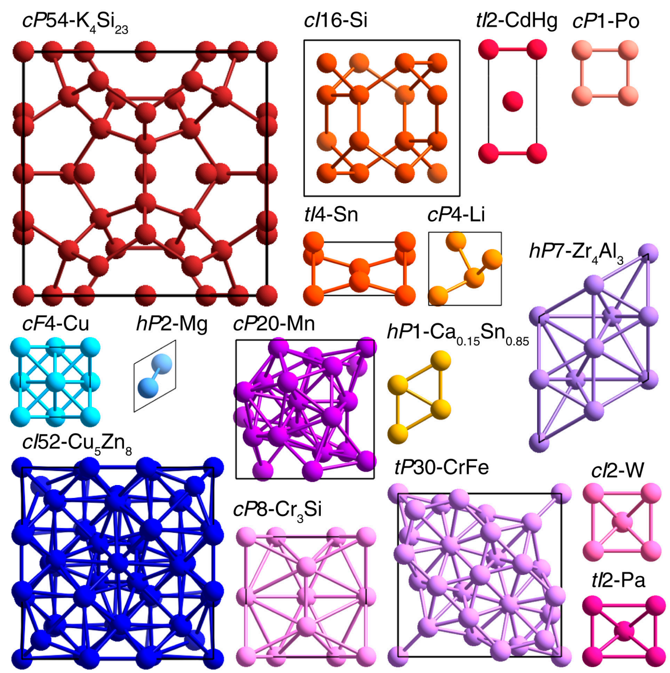


Regular Gaussian Process

Physics-informed Gaussian Process

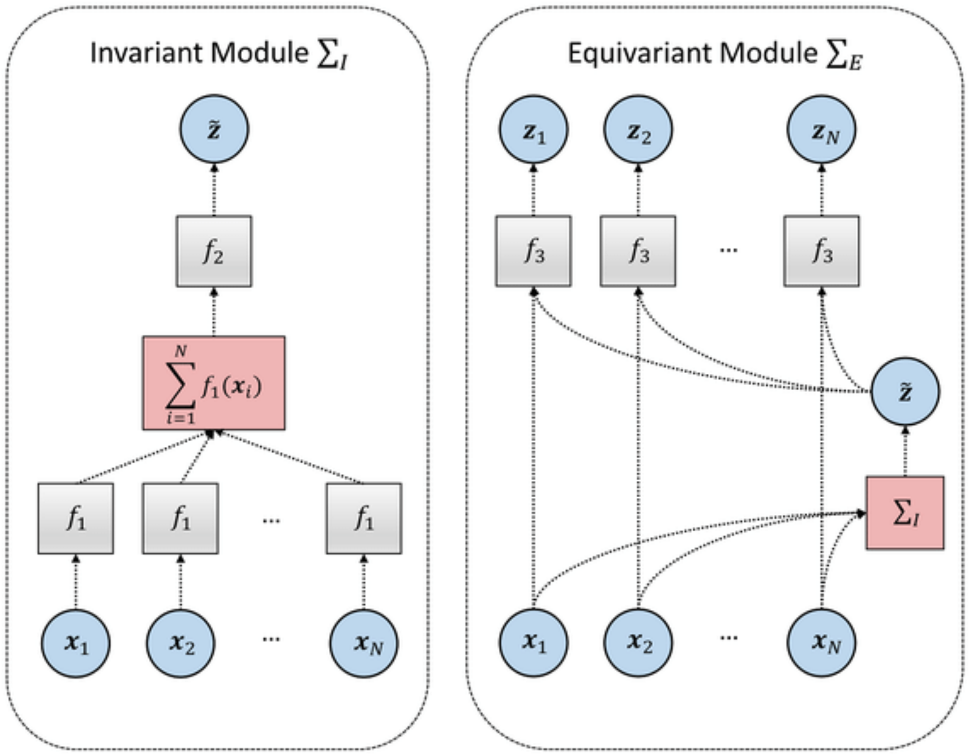
Symmetry is present in many scientific problems





Rotations, reflections, permutations, translations…

Other ways to add hard constraints: build NNs that respect given symmetries



f(gx) = f(x)

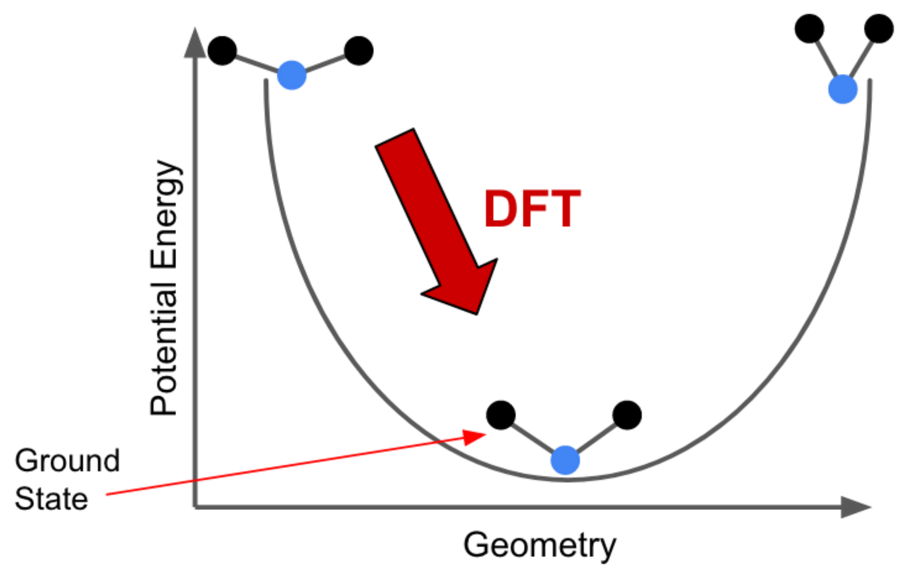
f(gx) = gf(x)

Atomistic systems have many symmetries

Equivariance and iterative calculations are important:

1) We use an equivariant GNN, though our relaxation method is not constrained to any particular GNN

2) Many individual steps (similar to numerical solvers) works better than one step



Acknowledgements

- **LBNL/UC Berkeley:** Aditi Krishnapriyan, Geoff Negiar, Amir Gholami, Andrew Rosen, Eric Taw, Daniel Rothchild
- **Google Research:** Alejandro Quieruga, JAXOpt team (Mathieu Blondel, Fabian Pedregosa)
- **Univ. of Pittsburgh:** Ben Erichson
- **Univ. of Utah:** Shandian Zhe, Mike Kirby