Physics Informed Neural Networks

Lecture for UCSD PHYS 139/239: Machine Learning in Physics

Presented by **Amir Gholami**

University of California Berkeley, ICSI

Mar 16, 2023

In collaboration with:

S. Subramanian, S. Zhe, M. Kirby, A. Krishnapriyan, K. Keutzer, M. Mahoney

Outline

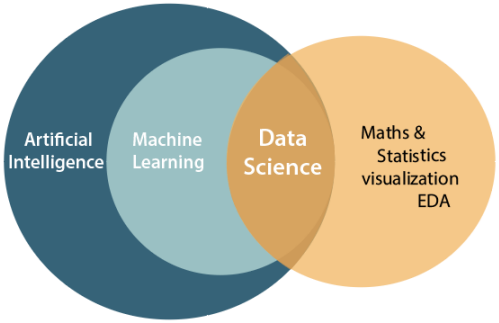
- **Introduction**
- Physics Informed Neural Networks
- Challenges associated with PINNs
- Conclusions and Future Work

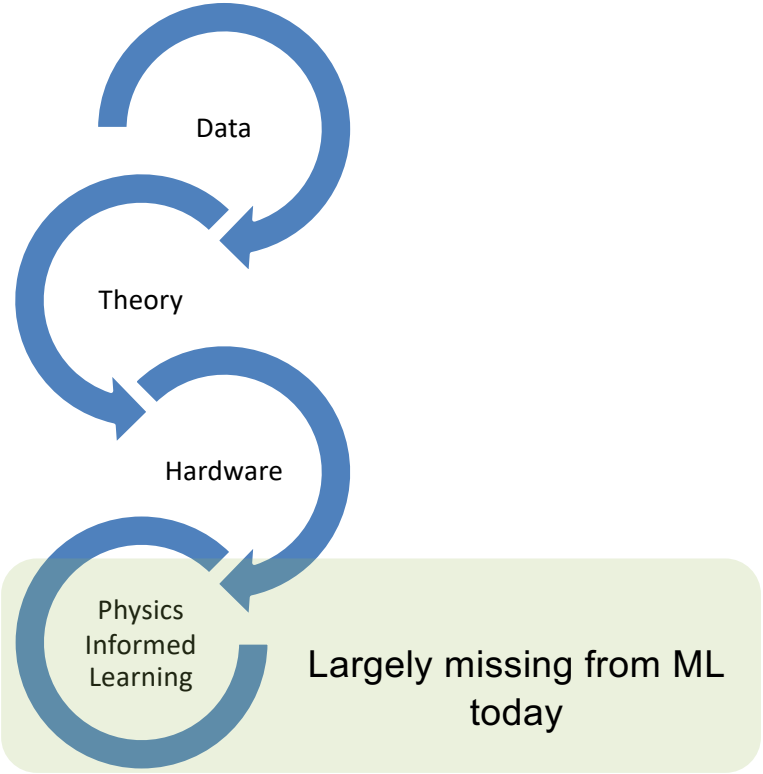
Physics Informed Learning

Computational Science is an important tool that we can use to incorporate physical invariances into learning, but until recently it was missing from mainstream ML.

"**Computational Science** can **analyze past events** and **look into the future**. It can explore the effects of thousands of scenarios for or in lieu of actual experiment and be used to study events beyond the reach of expanding the boundaries of experimental science"
–Tinsley Oden, 2013

To make further progress in ML it is crucial that we incorporate computational science into learning.





Dr. J. Tinsley Oden's Commemorative Speech: "THE THIRD PILLAR: The Computational Revolution of Science and Engineering", Honda Prize, 2013.

Physics Informed Neural Network

- Neural Networks **require a lot of data** to train
- Collecting large scale data is not always possible, for many applications, especially in medical and scientific domain
- However, an important source of data are the **Physical Laws** that govern our world which have been largely ignored in exchange for observed data
- Physical Laws include:
  - Conservation of Mass, Momentum, Energy, etc.

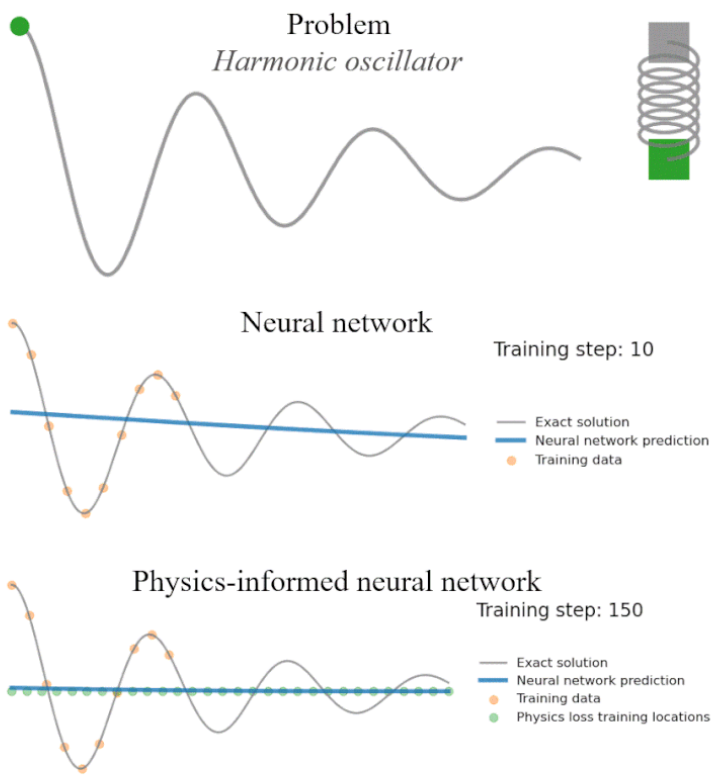
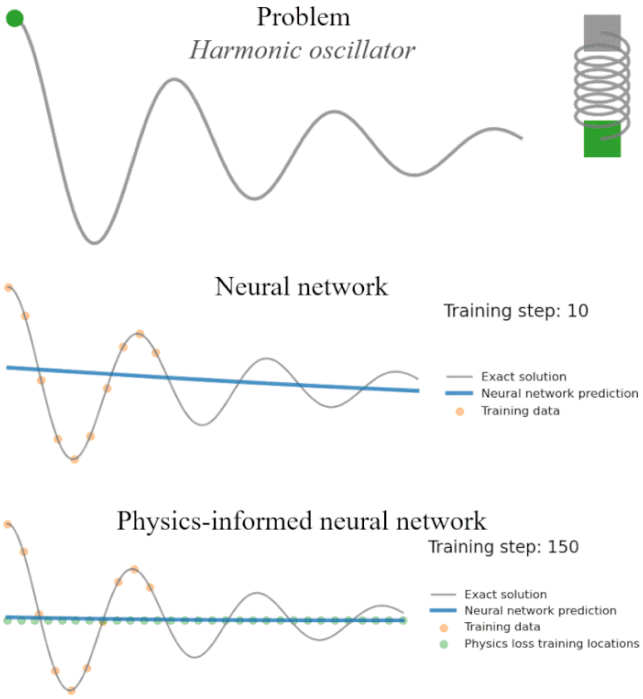


Illustration from Ben Moseley

Physics Informed Neural Network



$$m\frac{\partial^2 u}{\partial t^2} + \mu\frac{\partial u}{\partial t} + ku = 0$$

$$\min \frac{1}{N}\sum_{i}^{N}(\hat{u} - u_{true})^2 +$$

$$\frac{1}{M}\sum_{j}^{M}(m\frac{\partial^2 u_j}{\partial t^2} + \mu\frac{\partial u_j}{\partial t} + ku_j)^2$$

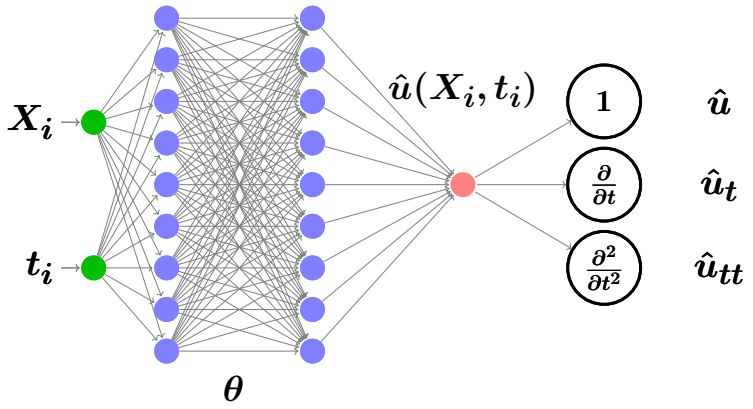


Illustration from Ben Moseley

Physical Laws as Additional Sources of Data

- Other than observed data, we know the invariances that govern physical phenomena
  - Conservation of mass, momentum, energy
  - In many cases, we also have approximate models that can predict the system behavior

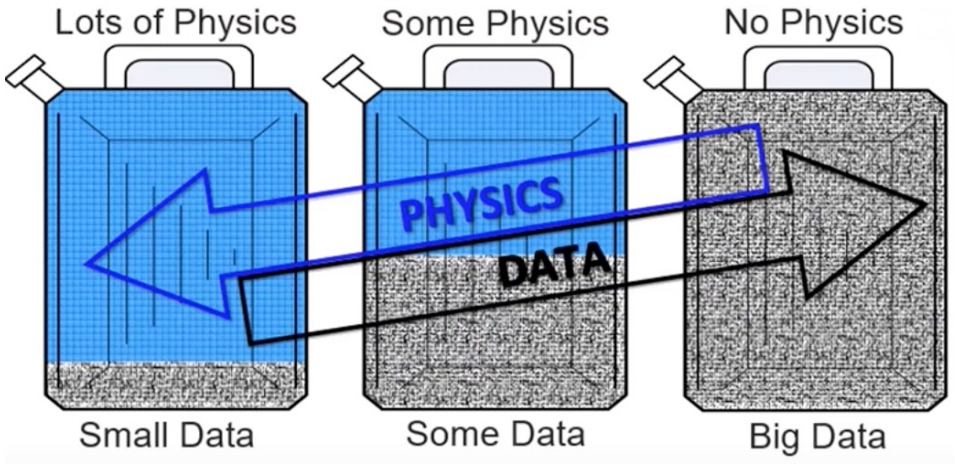
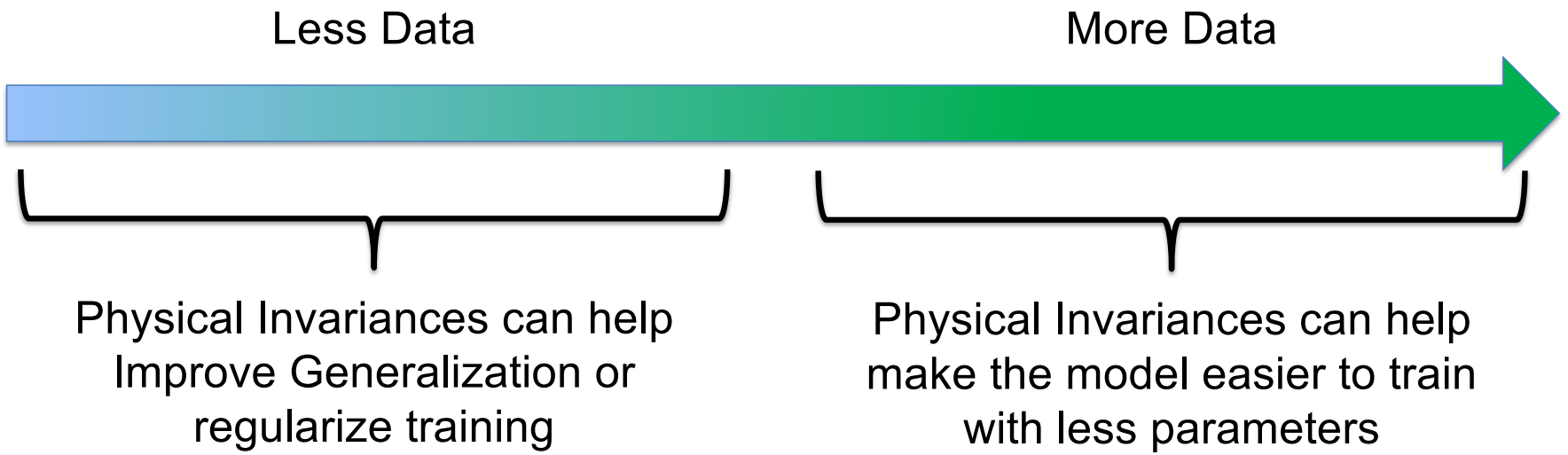


Illustration Credit: Prof. Karniadakis

Physical Laws as Additional Sources of Data

- Other than observed data, we know the invariances that govern physical phenomena
  - Conservation of mass, momentum, energy
  - In many cases, we also have approximate models that can predict the system behavior



The main question is how can we incorporate these invariances into learning?

MLPs are Universal Function Approximators

Now let's take a step back and see what are Neural Networks?

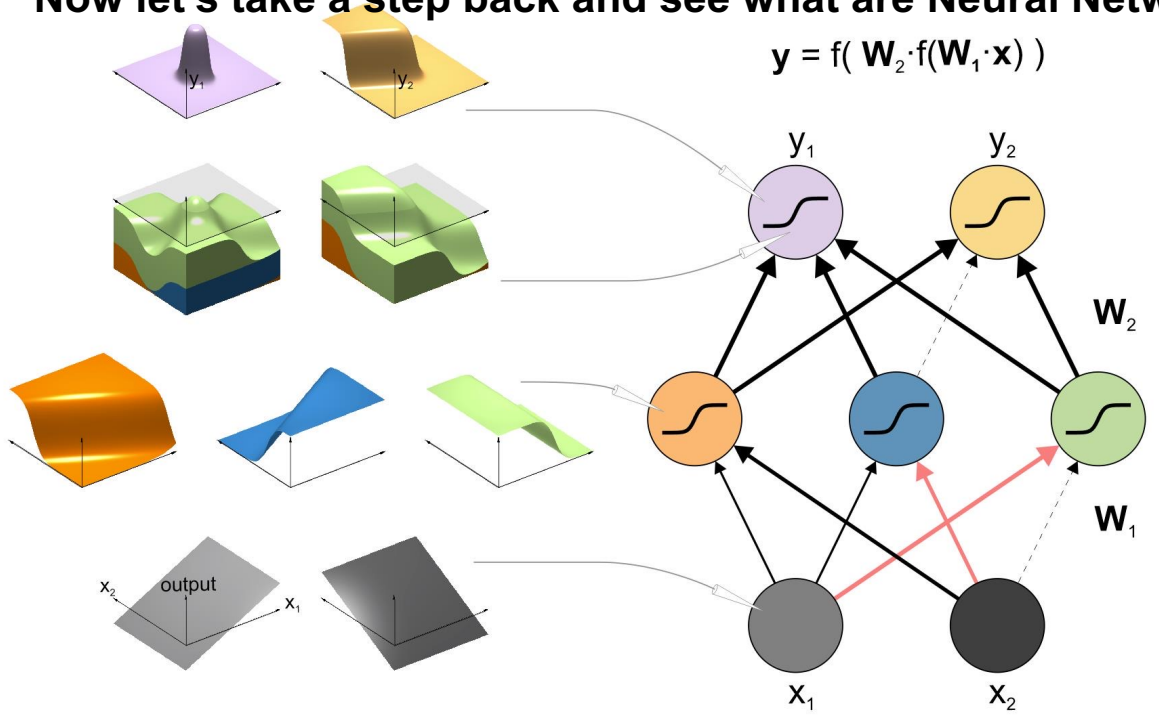
$$y = f(W_2 \cdot f(W_1 \cdot x))$$

Research Question: Given that NNs can approximate general functions, can we apply them to predict scientific phenomena?

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems, 2(4):303–314, 1989.*
K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, 1989.
Kriegeskorte N, Golan T. Neural network models and deep learning-a primer for biologists. arXiv preprint arXiv. 1902.

Why do we expect NNs to be helpful?

Now let's take a step back and see what are Neural Networks?



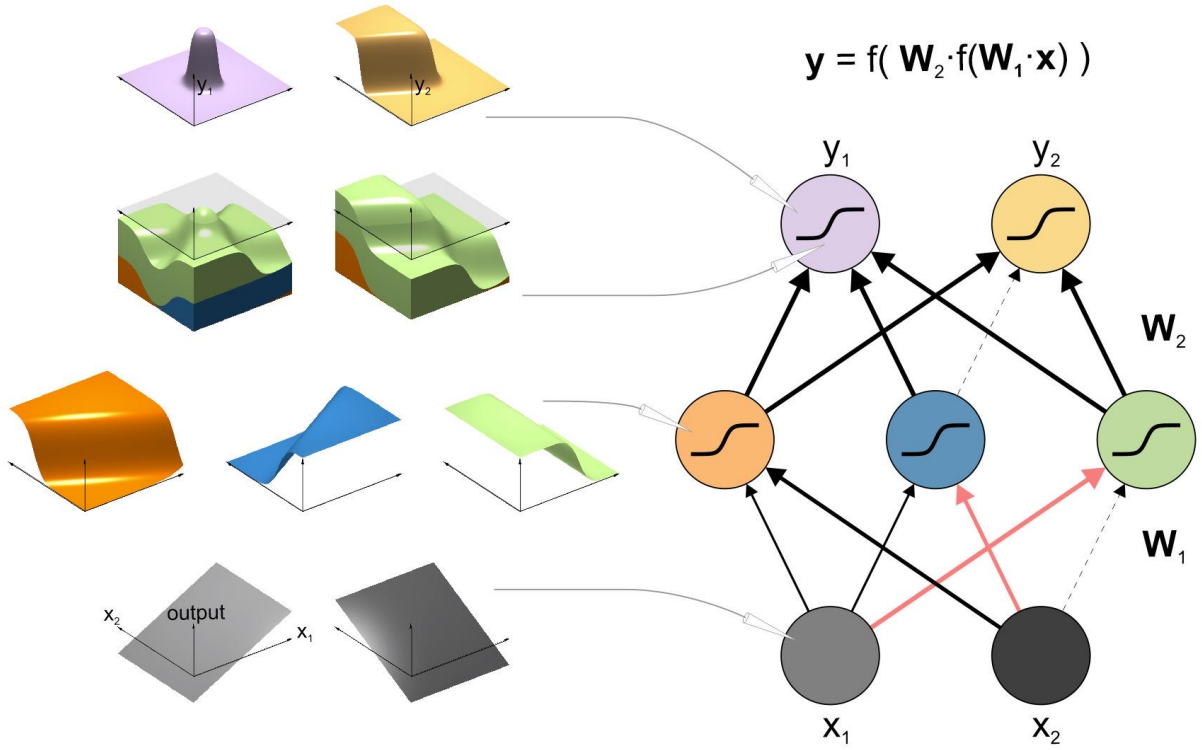
MLPs with non-linear activations are universal function approximators

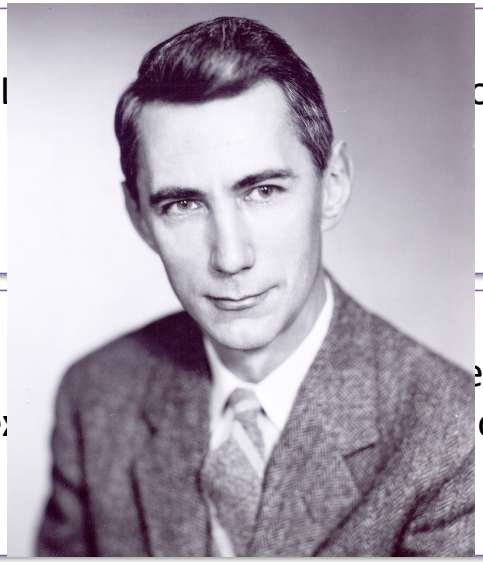
However, they may require **exponentially large** number of neurons

Theorem: for n>2, there is a Boolean function of n variables that requires at least $2^n/n$ Boolean gates, regardless of depth!

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems, 2(4):303–314, 1989.*
K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, 1989.
Kriegeskorte N, Golan T. Neural network models and deep learning-a primer for biologists. arXiv preprint arXiv. 1902.

MLPs are Universal Function Approximators





MI ons

ex of

Theorem: for n>2, there is a Boolean function of n variables that requires at least $2^n/n$ Boolean gates, regardless of depth!

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems, 2(4):303–314, 1989.*
K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, 1989.
Kriegeskorte N, Golan T. Neural network models and deep learning-a primer for biologists. arXiv preprint arXiv. 1902.

Universal Function *Approximation*

The missing piece in universal approximation theorem is that it only considers **approximation error,** and not trainability and/or generalizability of the NN.

We can broadly characterize the accuracy of a NN into three main types:

- **Approximation error to ground truth function**
- **Generalization to unseen data**
- **Trainability or optimization difficulty of the model**

Universal approximation theorem only considers the first one.

Moreover, it provides no method to train a model to get that approximation (naïve method using the basis function in the previous slide can require exponentially large number of neurons even for simple functions)

Summary So Far

- Neural Networks are universal function approximators but current methods require a lot of data to train them
- For many Scientific applications, we cannot obtain large amounts of data
- A solution for this is to incorporate physical laws into learning

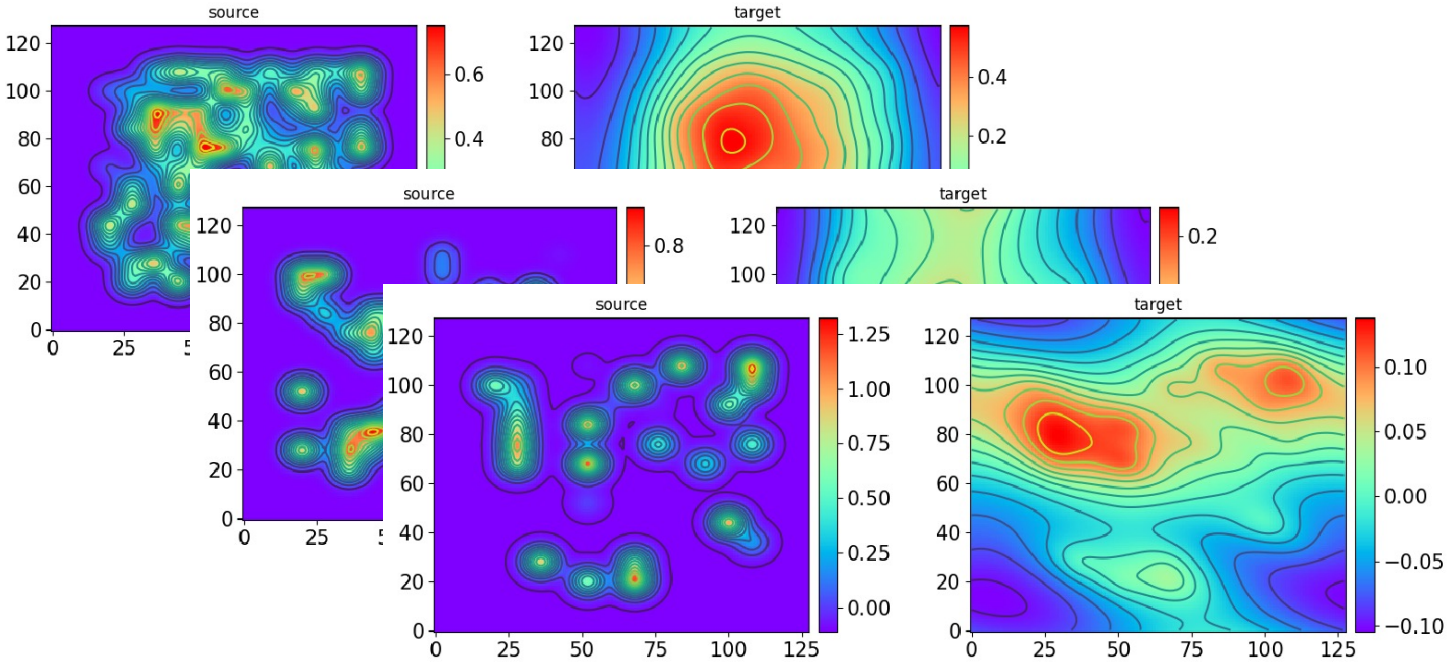
Next let's go over methods proposed so far for doing so!

Outline

- Introduction
- **Physics Informed Neural Networks**
- Challenges associated with PINNs
- Conclusions and Future Work

Methods for Incorporating Physics into Learning

- **Method 1 (Neural Operator): Train on large amount of data and let the NN learn the physics based operators**

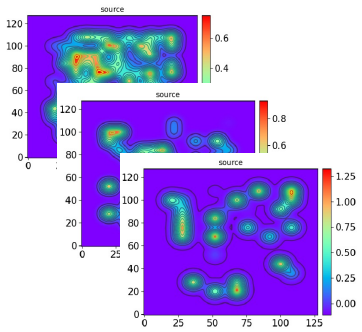


[Lu et al., 2019; Bhattacharya et al., 2020; Nelsen & Stuart, 2020; Li et al. 2020, Patel et al., 2021]

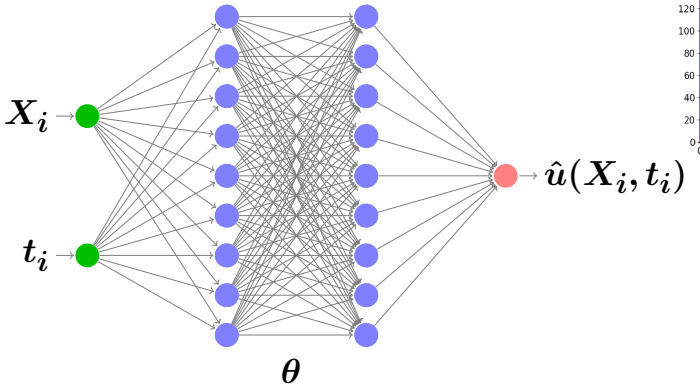
Methods for Incorporating Physics into Learning

- Train on large amount of data (and hope) that the NN learns the constraints

Obtain/Simulate a lot of data



Train the NN on this dataset



$$\min_{\theta} \mathcal{L} = \sum_{i} \|\hat{u}_i - u_i\|_2$$

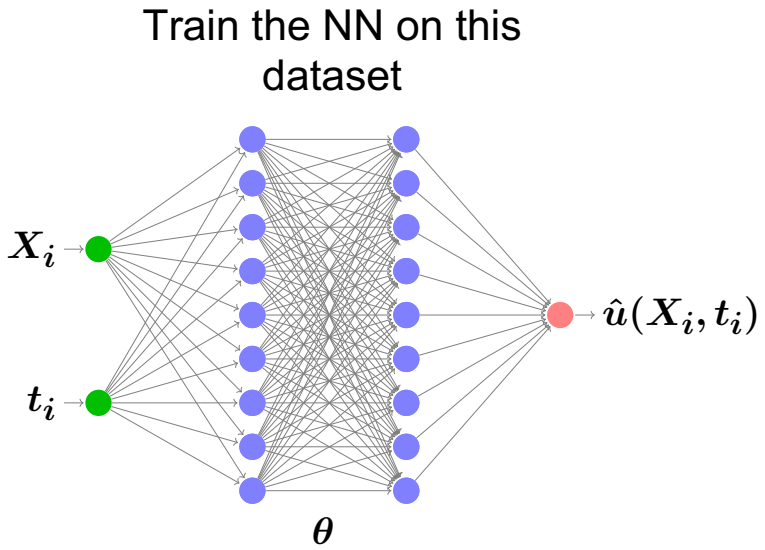
Main problems:

- No guarantee that the model obeys the physics laws (e.g. conservation laws)
- May require a lot of training data and obtaining/simulating these data is not always feasible

[Lu et al., 2019; Bhattacharya et al., 2020; Nelsen & Stuart, 2020; Li et al. 2020, Patel et al., 2021]

Methods for Incorporating Physics into Learning

- **Method 2:** Enforce physical laws as hard constraints either in:
  - NN Architecture: This is an open problem
  - Optimization: Very difficult to train the NN with such constraints



$$\min_{\theta} \mathcal{L} = \sum_{i} \|\hat{u}_i - u_i\|_2,$$

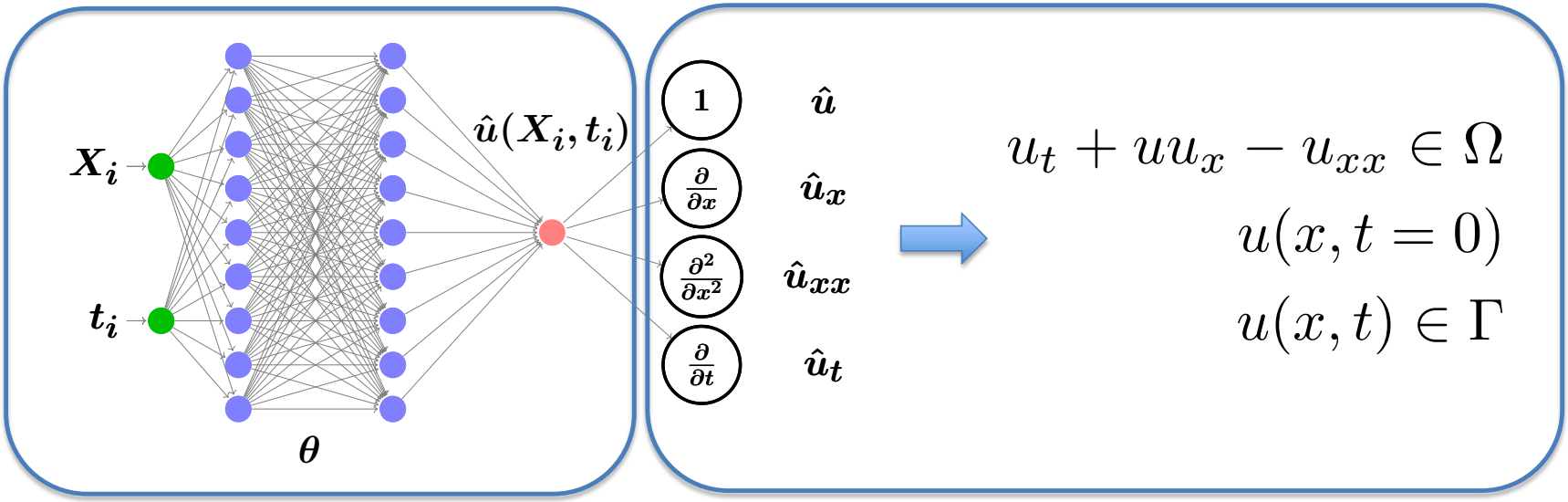
$$s.t. \quad u_t + uu_x - u_{xx} = 0.$$

Xu K, Darve E. Physics constrained learning for data-driven inverse modeling from sparse observations. arXiv preprint arXiv:2002.10521. 2020 Feb 24.

Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics. 2019 Feb 1;378:686-707.

Methods for Incorporating Physics into Learning

- **Method 3:** Use penalty methods and add the **physics model** as a residual to the loss.
- Below we consider Burgers' equation which can be modeled with a PDE (but this approach is applicable even if your model is different than a PDE)



Data Loss Function:

$$\mathcal{L}_u = \|\hat{u} - u\|_2^2$$

Physics Loss Function:

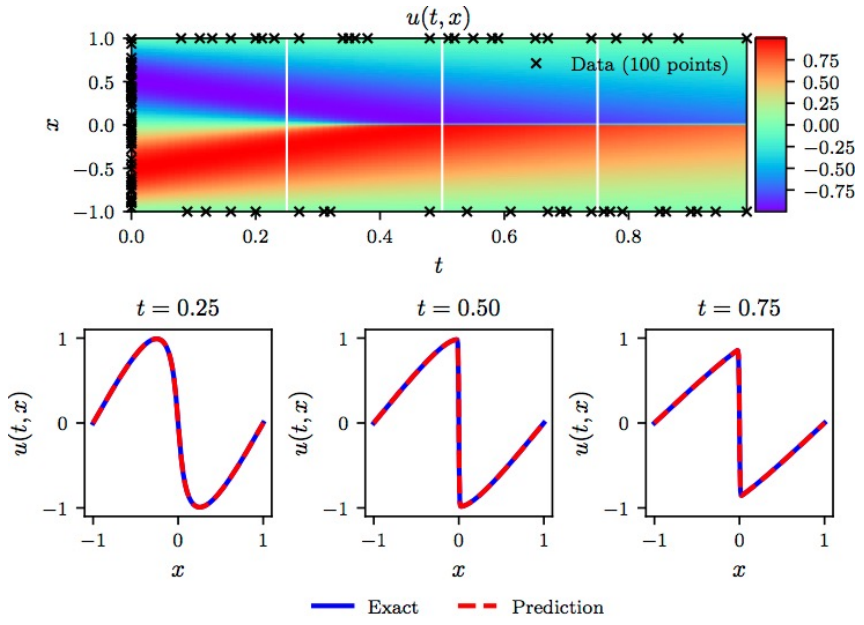
$$\mathcal{L}_\mathcal{F} = \|\hat{u}_t + \hat{u}\hat{u}_x - \hat{u}_{xx}\|_2^2$$

$$\min_\theta \mathcal{L} = \mathcal{L}_u + \lambda_\mathcal{F} \mathcal{L}_\mathcal{F}$$

Physics Informed Neural Networks

- **Method 3:** Use penalty methods and add the PDE residual to the loss.
  - Very easy to implement, and works with any NN architecture
  - Does not require a mesh or a numerical solver for the PDE
  - Can (in theory) work for high dimensional problems, and complex PDEs
    - For example, PDEs containing integral operators which are difficult to solve with finite difference methods.

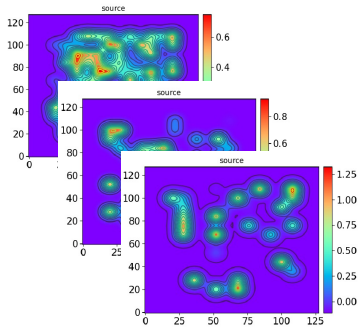
$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

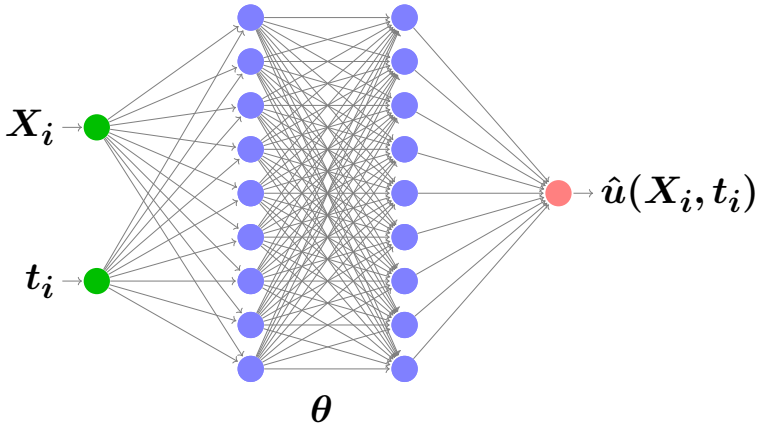


*[Weinan et al. 2017; **Raissi et al. 2019**; Rackauckas et al. 2020; Hennigh et al. 2021; Lu et al. 2021]*

PINO: Physics Informed Neural Operators

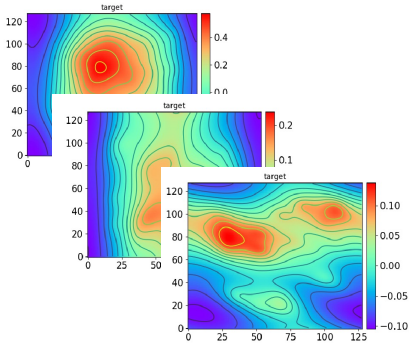
- **Method 4:** Use a combination of Neural Operator and PINNs
  - Uses a combination of observation data points as well as physical constraints added as soft penalty terms to the loss function





Data Loss Function:

$$\mathcal{L}_u = \|\hat{u} - u\|_2^2$$



Physics Loss Function:

$$\mathcal{L}_{\mathcal{F}} = \|\hat{u}_t + \hat{u}\hat{u}_x - \hat{u}_{xx}\|_2^2$$

$$\min_{\theta} \mathcal{L} = \mathcal{L}_u + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}$$

Li et al. 2021]

But this is not the entire story

- There are a lot of subtleties in adding a soft-constraint and PINNs actually do not work as well, even for simple problems.
- To study this, we chose three families of PDEs:
  - Advection (aka wave equation)
  - Reaction
  - Reaction-Diffusion

For all of these cases we observed that PINNs fail to learn the relevant physics, since there are many moving parts in this problem

Summary So Far

Broadly there are four popular ways of incorporating physical laws into learning:

❖ **Neural Operators**: Have the NN learn the operator by providing it with enough training data points

Pros:

- Can learn the operator and apply it without specific constraints on mesh/discretization (often only true if given ample data)
- Only needs to be trained once -> Can be applied for different configurations of the operator
- Does not need explicit knowledge of the underlying Physics
- Very easy to implement as it does not need any special back propagation for computing PDEs and derivatives

Cons:

- It often needs a lot of training data
- There is no knob to tune to penalize the model if it violates the physical constraints of the problem

Summary So Far

Broadly there are four popular ways of incorporating physical laws into learning:

❖ **Constrained Learning**: Constrain the NN to always satisfy the physical laws

Pros:

- Guarantee that the model outputs will always obey the physical laws of the problem

Cons:

- Very difficult to constrain the architecture to obey the laws
- Alternative methods of incorporating them into discretization methods (such as FEM) either creates optimization difficulty, or requires designing a mesh and a solver as done in FEM methods.

Summary So Far

Broadly there are four popular ways of incorporating physical laws into learning:

❖ **Physics Informed Neural Networks**: Add the physical constraints as **soft penalty** terms

Pros:

- Gives us a knob for enforcing the physical constraints
- Relatively easy to formulate the loss and compute the derivatives

Cons:

- Adding the soft constraint often makes the loss landscape very difficult to optimize
- Only specializes for a given set of model constraints/configuration and needs to be retrained if the config is changed

Summary So Far

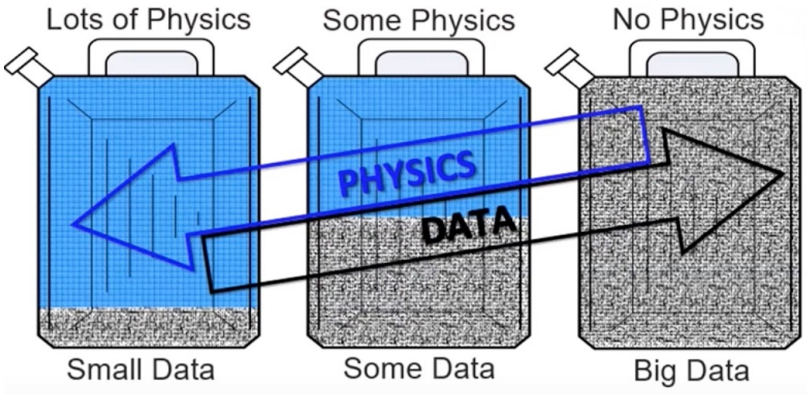
- PINO: Could be thought as combining Neural Operators and PINNs
- It uses both available data as well as given physical constraints to learn

Pros:

- It often results in better results since it learns from more data sources

Cons:

- Similar issues with Neural Operator and PINNs



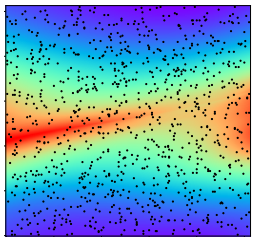
Outline

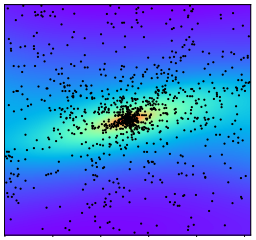
- Introduction
- Physics Informed Neural Networks
- **Challenges associated with PINNs**
  - **Optimization Difficulties**
  - Choice of collocation points
- Conclusions and Future Work

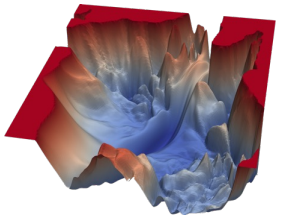
Challenges Associated with PINNs?

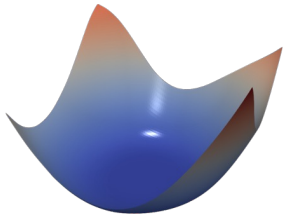
We need to rethink the design, training, and the input data for successful application of NNs to scientific problems.

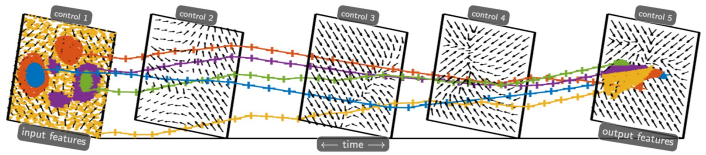
- PINN Failures: NeurIPS'21
- PyHessian IEEE BigData'20
- Flat/Sharp Minima: NeurIPS'18
- ANODEV2: NeurIPS'19
- ANODE: IJCAI'19



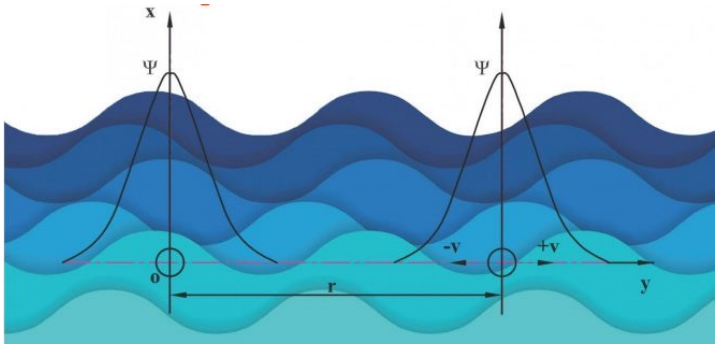








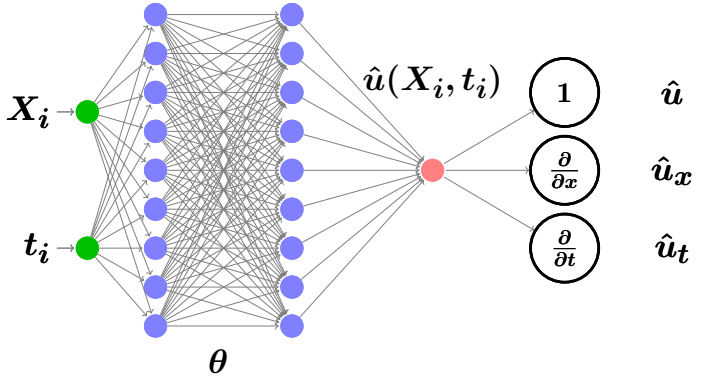
Advection Equation



$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega,\ t \in [0, T],$$

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

Periodic boundary conditions: $u(0, t) = u(2\pi, t)$



$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2 \qquad \text{PDE Residual}$$

$$+ \|\hat{u}(x, 0) - sin(x)\|_2^2 \qquad \text{Initial Condition}$$

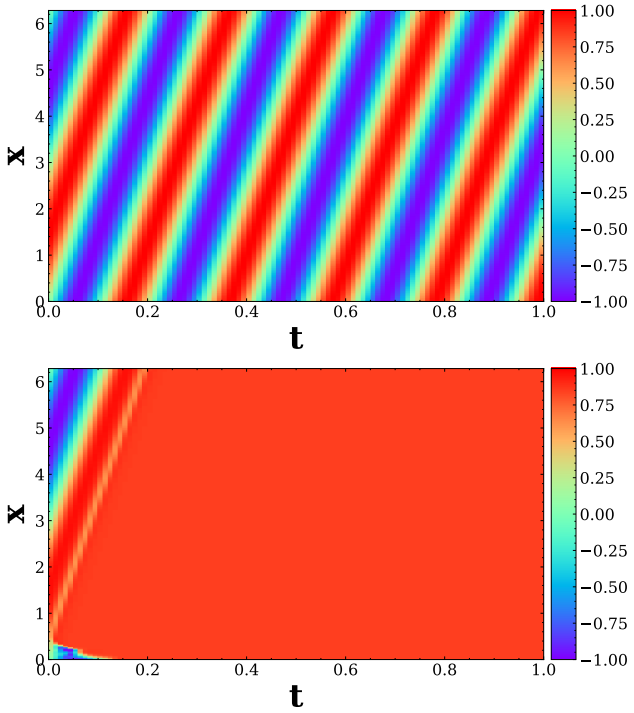
$$+ \|\hat{u}(x = 2\pi) - \hat{u}(x = 0)\|_2^2 \qquad \text{Boundary Condition}$$

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

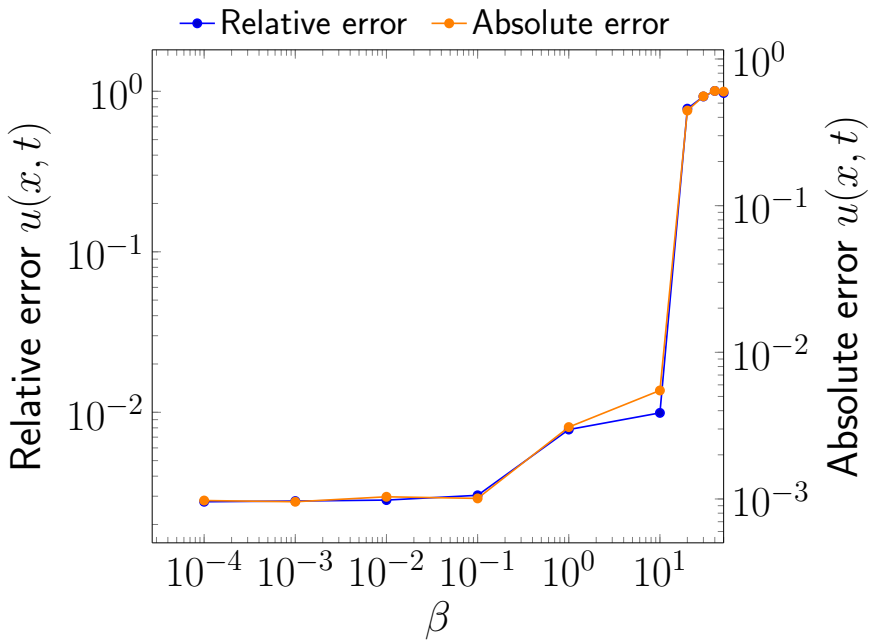
PINN can fail to learn Advection

Exact Solution

PINN Prediction

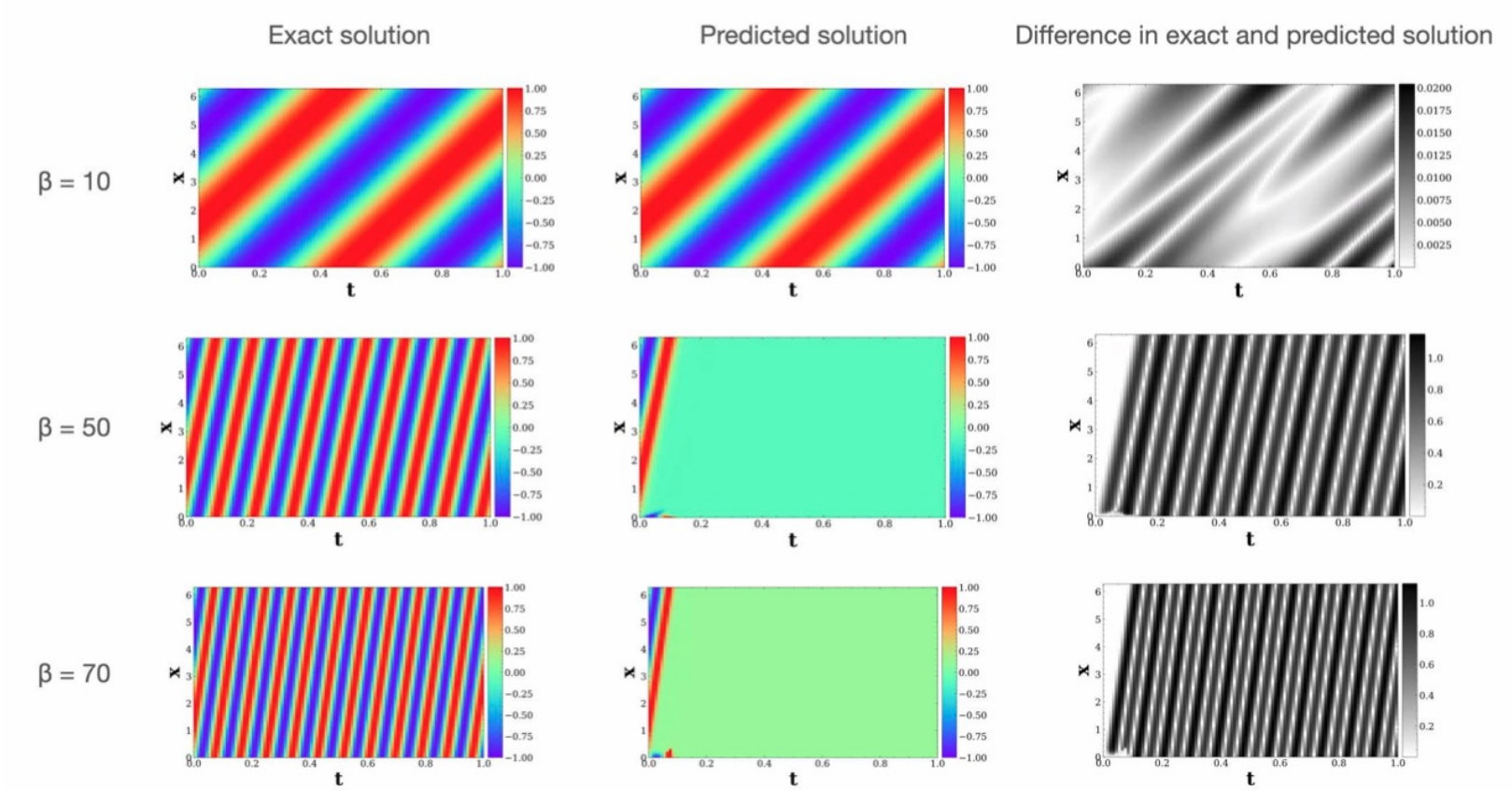


$\beta = 30$



Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

PINN can fail to learn Advection



PINN can fail to Learn Reaction Equation

Learning reaction with PINNs

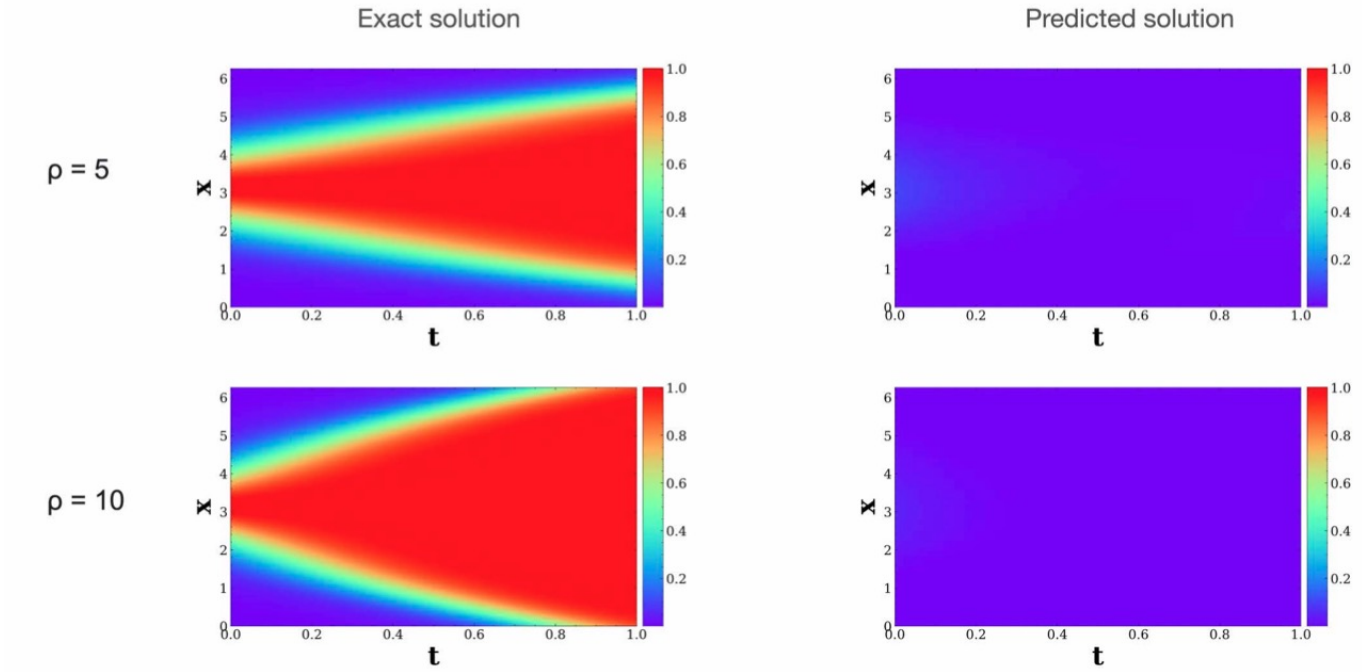
$$\frac{\partial u}{\partial t} - \rho u(1-u) = 0, x \in \Omega, t \in [0, T],$$

reaction coefficient

$$u(x, 0) = h(x), x \in \Omega$$

Initial condition: $u(x, 0) = e^{-\frac{(x-\pi)^2}{2(\pi/4)^2}}$,

Periodic boundary conditions: $u(0, t) = u(2\pi, t)$



Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

PINN can fail to Learn Fisher Equation

Learning reaction-diffusion with PINNs

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0, \quad x \in \Omega,\ t \in (0, T],$$

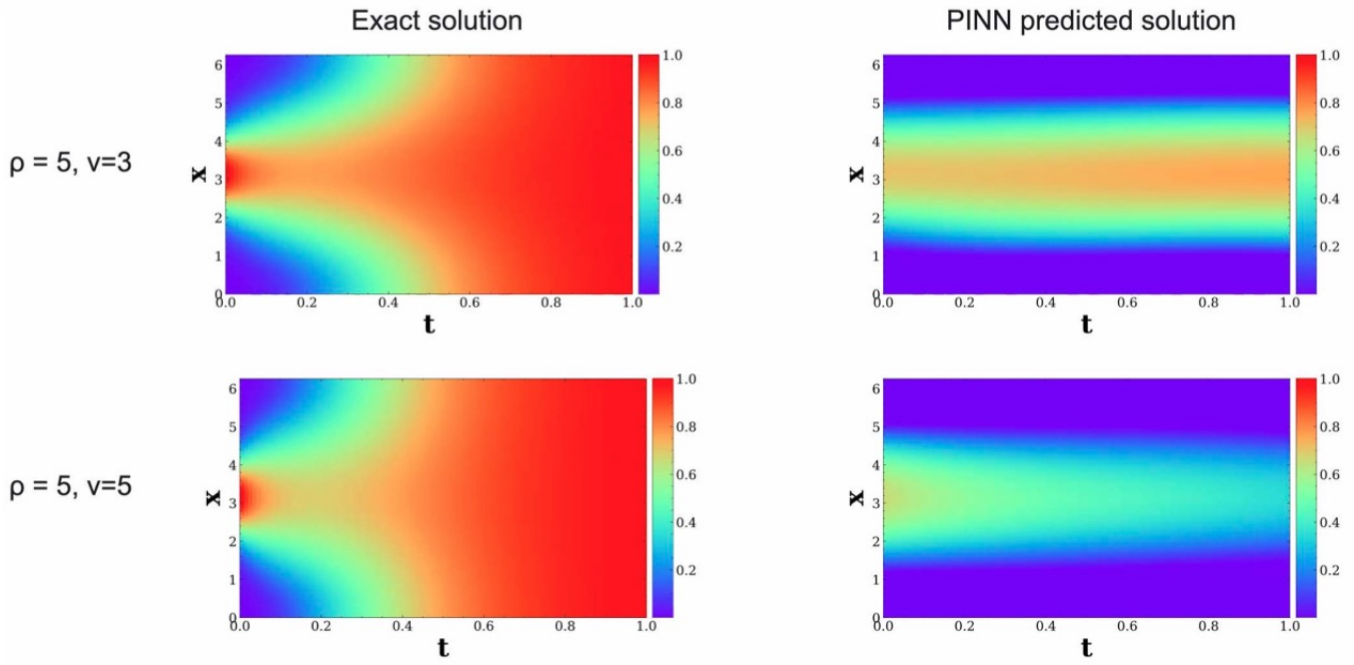
$$u(x, 0) = h(x), \quad x \in \Omega$$

diffusion coefficient

reaction coefficient

Initial condition: $u(x, 0) = e^{-\frac{(x-\pi)^2}{2(\pi/4)^2}}$,

Periodic boundary conditions: $u(0, t) = u(2\pi, t)$



Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

Training: Optimization Challenges with PINNs

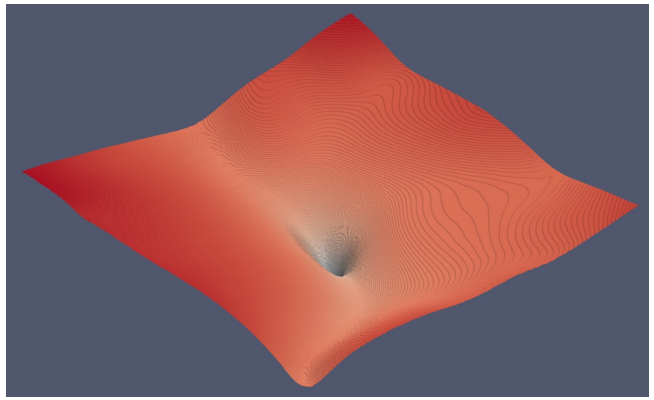
Data Loss Function:

$$\mathcal{L}_u = \|\hat{u} - u\|_2^2$$

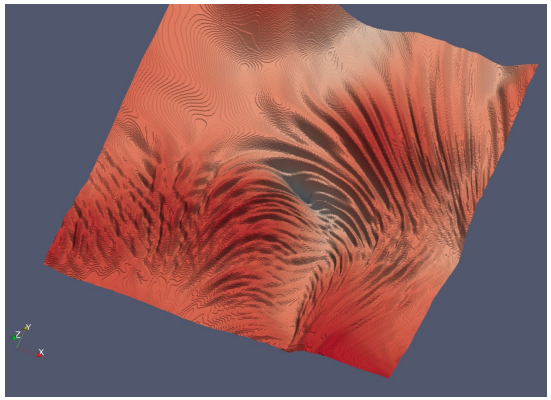
Physics Loss Function:

$$\mathcal{L}_{\mathcal{F}} = \|\hat{u}_t + \hat{u}\hat{u}_x - \hat{u}_{xx}\|_2^2$$

$$\min_{\theta} \mathcal{L} = \mathcal{L}_u + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}$$



Without Physics Loss



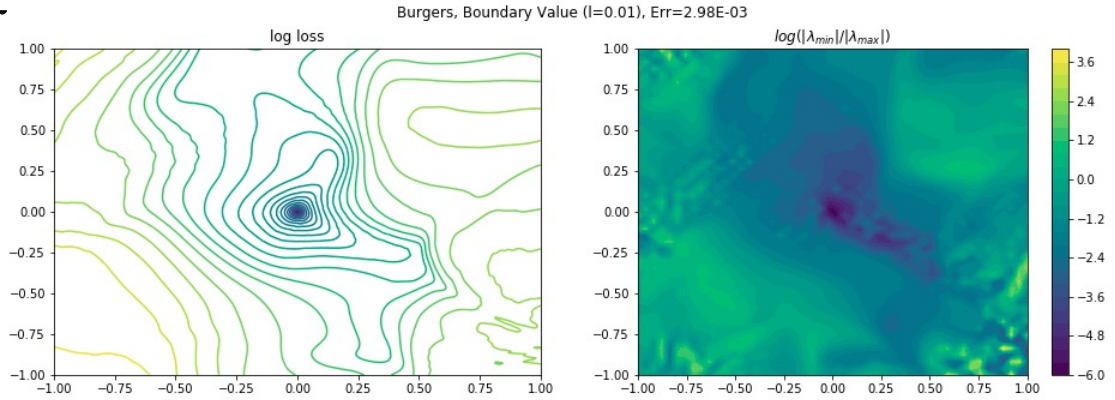
With Physics Loss

Illustration credit: Roman Amici, Mike Kirby

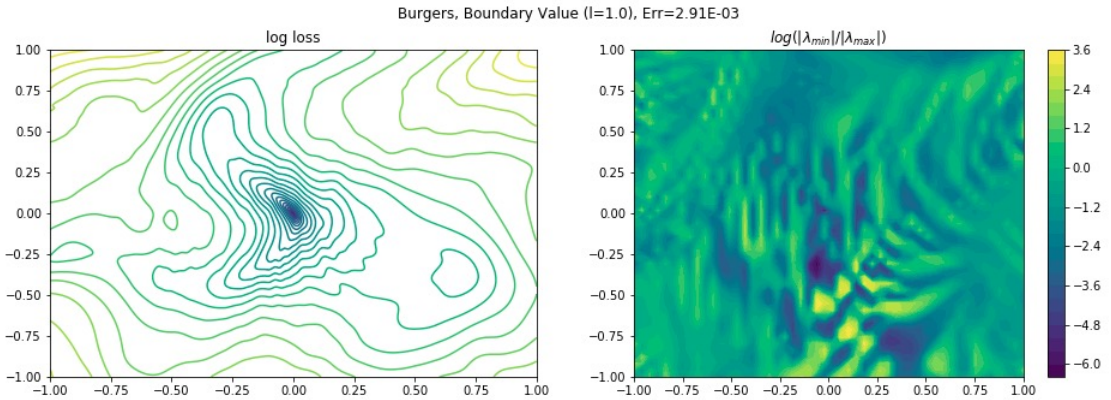
Training: Optimization Challenges with PINNs

$$\min_{\theta} \mathcal{L} = \mathcal{L}_u + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}$$

Loss function with 0.01 multiplier for PDE loss

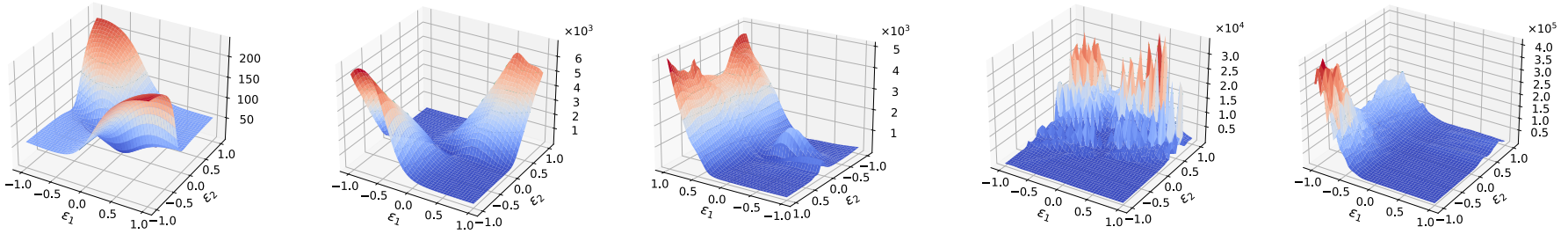


Loss function with 1.0 multiplier for PDE loss



Characterizing Failure Points: Loss Landscape

- To better understand the problem, we need to look at the optimization problem.
- The loss function that we are dealing with is quite complex and non-convex, so there is no guarantee that the optimizer can find a good solution.
- To diagnose the problem let's look at the loss landscape after training PINN

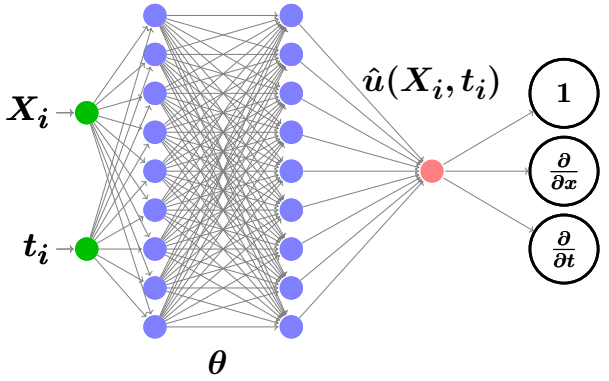


(a) $\beta = 1.0$ (b) $\beta = 10.0$ (c) $\beta = 20.0$ (d) $\beta = 30.0$ (e) $\beta = 40.0$

β	1	10	20	30	40
Relative error	7.84×10^{-3}	1.08×10^{-2}	7.50×10^{-1}	8.97×10^{-1}	9.61×10^{-1}
Absolute error	3.17×10^{-3}	6.03×10^{-3}	4.32×10^{-1}	5.42×10^{-1}	5.82×10^{-1}

Characterizing Failure Points: Loss Landscape

- Adding the PDE-based soft-regularization, while easy to implement, can result in optimization difficulties (and sometimes ill-conditioning behaviour).
- To study this let's do an experiment and change the weight of the PDE loss and analyze how the loss landscape changes

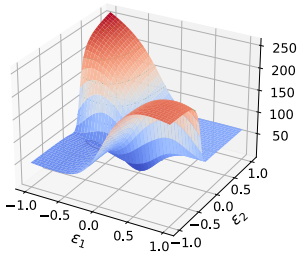


$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2 \quad \text{PDE Residual}$$

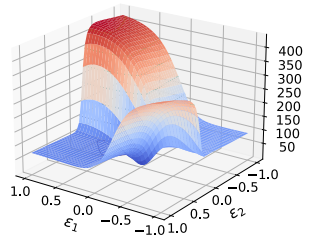
$$+ \|\hat{u}(x,0) - sin(x)\|_2^2 \quad \text{Initial Condition}$$

$$+ \|\hat{u}(x = 2\pi) - \hat{u}(x = 0)\|_2^2 \quad \text{Boundary Condition}$$

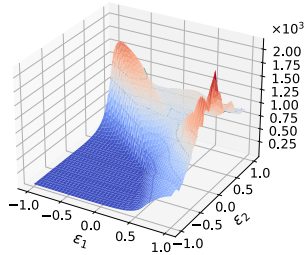
Characterizing Failure Points: Loss Landscape



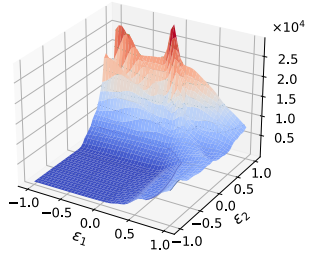
(a) $\lambda = 1 \times 10^{-6}$



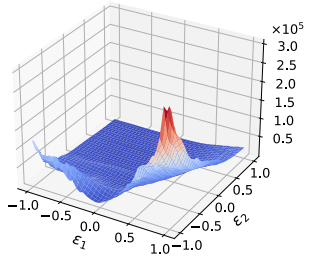
(b) $\lambda = 1 \times 10^{-5}$



(c) $\lambda = 1 \times 10^{-3}$



(d) $\lambda = 1 \times 10^{-1}$



(e) $\lambda = 1 \times 10^{1}$

λ	1×10^{-6}	1×10^{-5}	1×10^{-3}	1×10^{-1}	1×10^{1}
Relative error	1.69	1.65	1.00	1.08	0.982
Absolute error	0.987	0.987	0.623	0.647	0.595

$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2 \quad \text{PDE Residual}$$

$$+ \|\hat{u}(x,0) - sin(x)\|_2^2 \quad \text{Initial Condition}$$

$$+ \|\hat{u}(x = 2\pi) - \hat{u}(x = 0)\|_2^2 \quad \text{Boundary Condition}$$

As we reduce λ the optimization gets easier but PINN's solution has ~100% error

Summary So Far

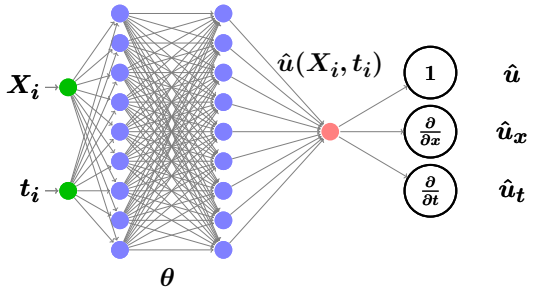
- While PINN formulation is easy to implement with auto-diff but it is often very difficult to train it:
- We saw several cases where PINNs may fail to converge to a reasonable solution
- One of the sources is that adding the soft constraint could make the loss landscape rather difficult to optimize

These are still open problems, but next I will introduce some recent approaches from our group which aim to alleviate these issues

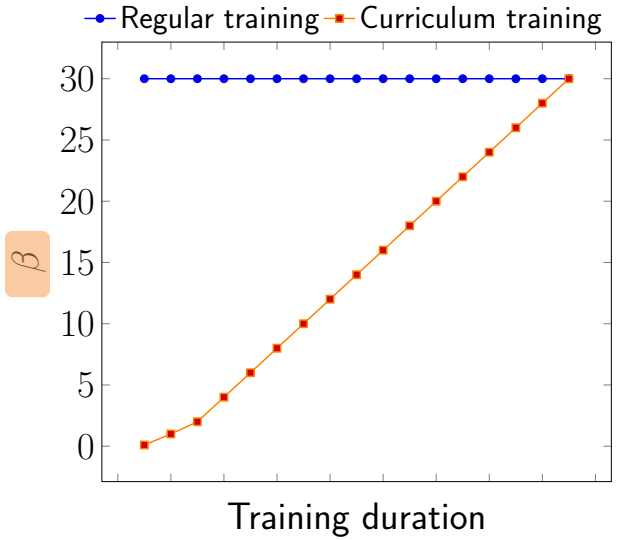
Rethinking PINNs: Curriculum Learning

- The main idea is to start the training with **simple physical constraints** and introduce the complexities iteratively throughout learning
- First let the NN learn the simple problems, before penalizing it for learning the exact PDE

Example: For the advection equation, we start to train the NN with very small velocities, and slowly increase the velocity to the target one

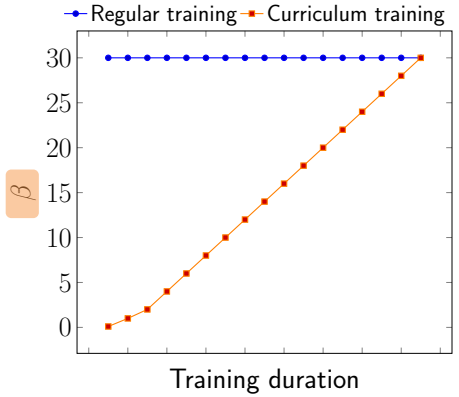


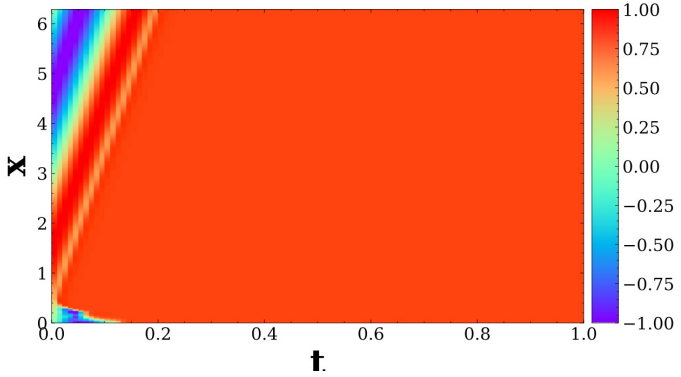
$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2$$
$$+ \|\hat{u}(x,0) - sin(x)\|_2^2$$
$$+ \|\hat{u}(x=2\pi) - \hat{u}(x=0)\|_2^2$$



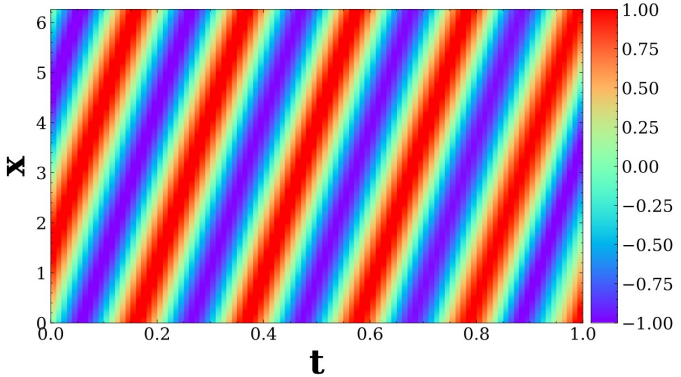
Rethinking PINNs: Curriculum Learning

$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2$$
$$+ \|\hat{u}(x,0) - sin(x)\|_2^2$$
$$+ \|\hat{u}(x=2\pi) - \hat{u}(x=0)\|_2^2$$



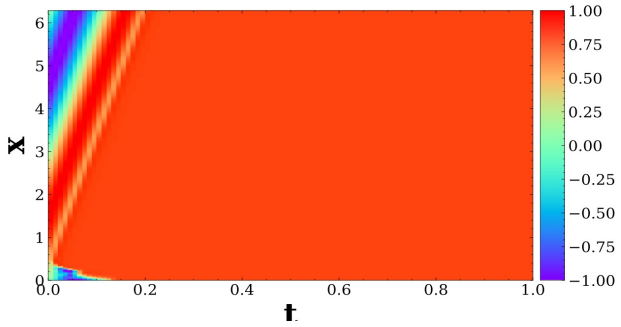


Regular training PINN solution for $\beta = 30$

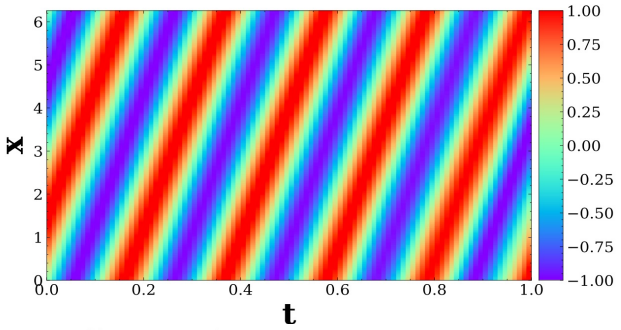


Curriculum training PINN solution for $\beta = 30$

Rethinking PINNs: Curriculum Learning



Regular training PINN solution for $\beta = 30$



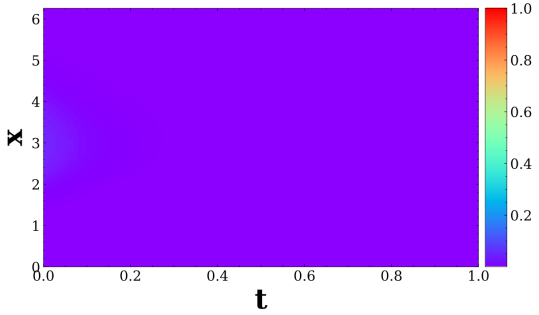
Curriculum training PINN solution for $\beta = 30$

		Regular PINN	Curriculum training
1D convection: $\beta = 20$	Relative error	7.50×10^{-1}	$\mathbf{9.84 \times 10^{-3}}$
	Absolute error	4.32×10^{-1}	$\mathbf{5.42 \times 10^{-3}}$
1D convection: $\beta = 30$	Relative error	8.97×10^{-1}	$\mathbf{2.02 \times 10^{-2}}$
	Absolute error	5.42×10^{-1}	$\mathbf{1.10 \times 10^{-2}}$
1D convection: $\beta = 40$	Relative error	9.61×10^{-1}	$\mathbf{5.33 \times 10^{-2}}$
	Absolute error	5.82×10^{-1}	$\mathbf{2.69 \times 10^{-2}}$

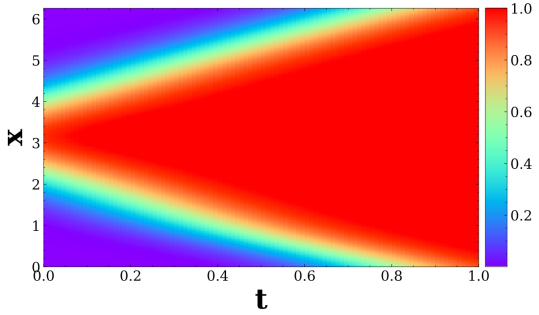
Rethinking PINNs: Curriculum Learning

- This approach works quite well for reaction problem as well

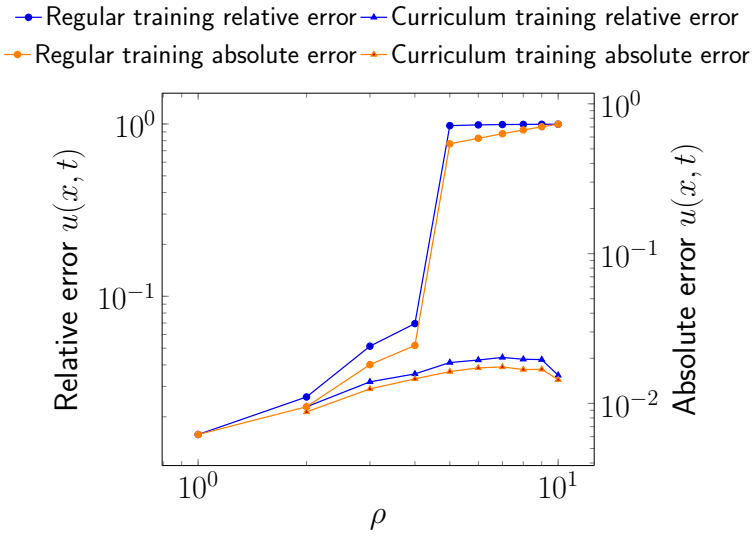
$$\frac{\partial u}{\partial t} - \rho u(1-u) = 0, \quad x \in \Omega,\ t \in (0, T],$$
$$u(x, 0) = h(x), \quad x \in \Omega.$$



Regular training PINN solution for $\rho = 10$

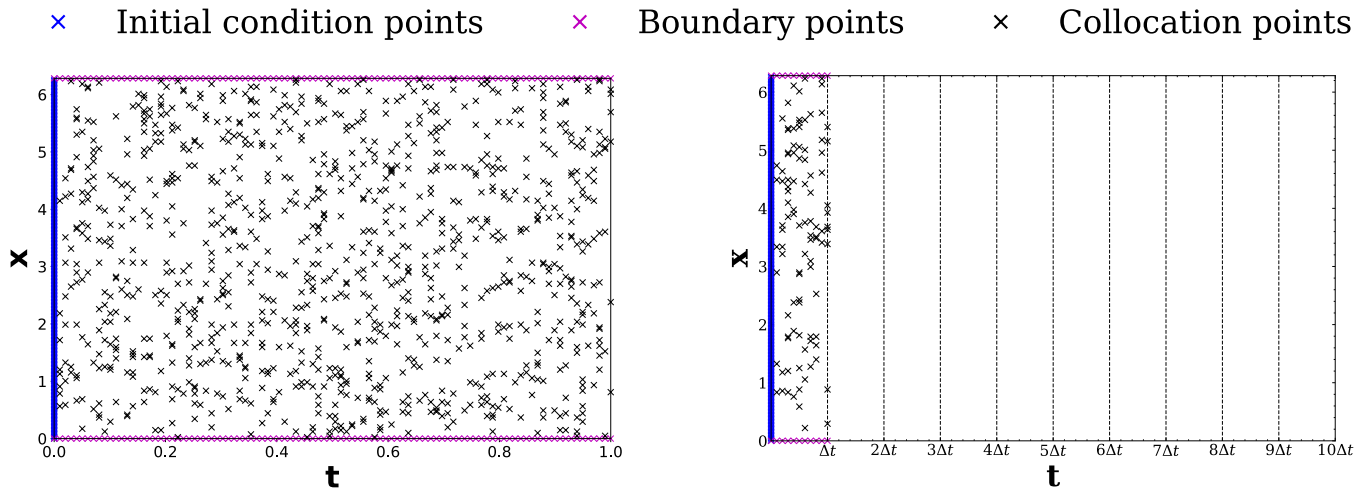


Curriculum training PINN solution for $\rho = 10$



Rethinking PINNs:
Pose the Problem as Sequence to Sequence Learning

- PINN formulation tries to predict the **entire space-time simultaneously**.
  - This is a very difficult task/function to approximate.
- An alternative is to pose the problem as sequence to sequence learning, where PINN learns to predict the solution in a finite time horizon, and iteratively predicts next time steps



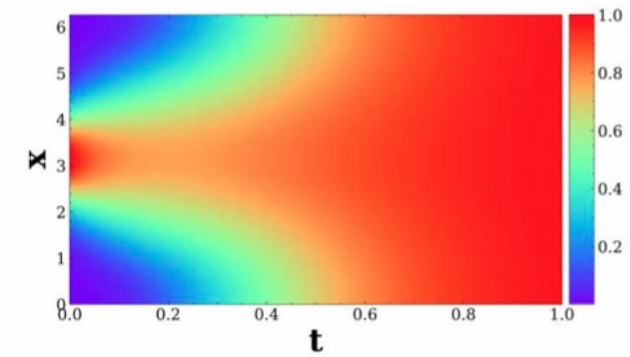
Regular PINN Training

Seq2Seq Training

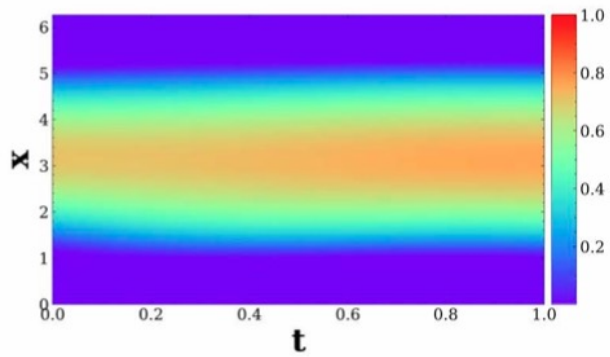
Rethinking PINNs: Seq2Seq Learning for Reaction-Diffusion Problem

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0, \quad x \in \Omega,\ t \in (0, T],$$

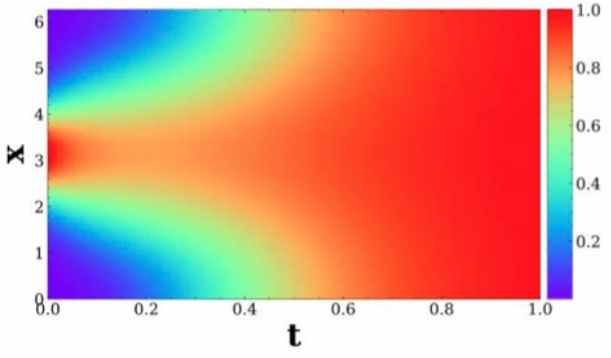
$$u(x, 0) = h(x), \quad x \in \Omega.$$



Exact solution for ρ = 5, v=3



Regular PINN solution for ρ = 5, v=3



seq2seq PINN solution for ρ = 5, v=3

Rethinking PINNs: Seq2Seq Learning for Reaction-Diffusion Problem

Seq2Seq approach can get significantly lower error than regular PINN which tries to predict the entire state space at once

		Entire state space	$\Delta t = 0.05$	$\Delta t = 0.1$
$\nu = 2,\ \rho = 5$	Relative error	5.07×10^{-1}	2.04×10^{-2}	$\mathbf{1.18 \times 10^{-2}}$
	Absolute error	2.70×10^{-1}	1.06×10^{-2}	$\mathbf{6.41 \times 10^{-3}}$
$\nu = 3,\ \rho = 5$	Relative error	7.98×10^{-1}	1.92×10^{-2}	$\mathbf{1.56 \times 10^{-2}}$
	Absolute error	4.79×10^{-1}	1.01×10^{-2}	$\mathbf{8.17 \times 10^{-3}}$
$\nu = 4,\ \rho = 5$	Relative error	8.84×10^{-1}	2.37×10^{-2}	$\mathbf{1.59 \times 10^{-2}}$
	Absolute error	5.74×10^{-1}	1.15×10^{-2}	$\mathbf{8.01 \times 10^{-3}}$
$\nu = 5,\ \rho = 5$	Relative error	9.35×10^{-1}	$\mathbf{2.36 \times 10^{-2}}$	2.39×10^{-2}
	Absolute error	6.46×10^{-1}	$\mathbf{1.09 \times 10^{-2}}$	1.15×10^{-2}
$\nu = 6,\ \rho = 5$	Relative error	9.60×10^{-1}	2.81×10^{-2}	$\mathbf{2.69 \times 10^{-2}}$
	Absolute error	6.84×10^{-1}	$\mathbf{1.17 \times 10^{-2}}$	1.28×10^{-2}

Outline

- Introduction
- Physics Informed Neural Networks
- **Challenges associated with PINNs**
  - Optimization Difficulties
  - **Choice of collocation points**
- Conclusions and Future Work

Where to enforce the physical laws?

- An important consideration in PINNs is which points should we enforce the physical constraints?

Question:

- What is the problem of enforcing the constraints on as many points as possible?
- What is a good strategy for choosing the points?

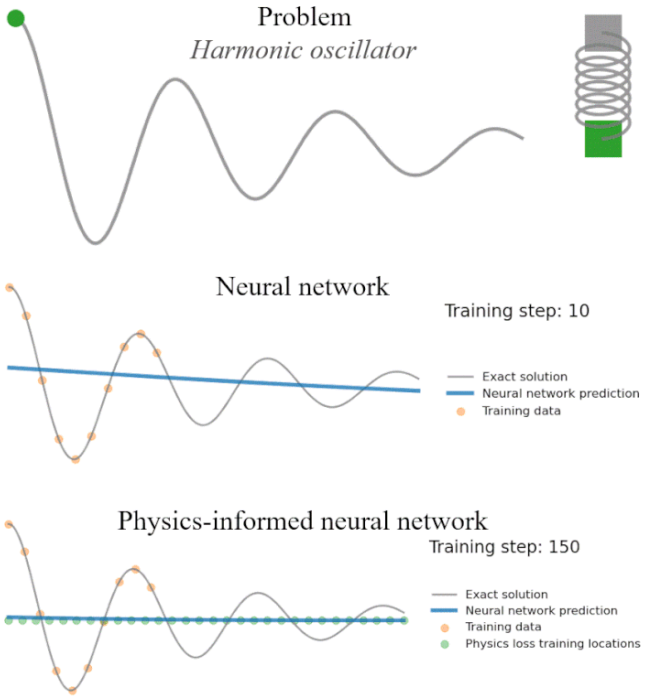
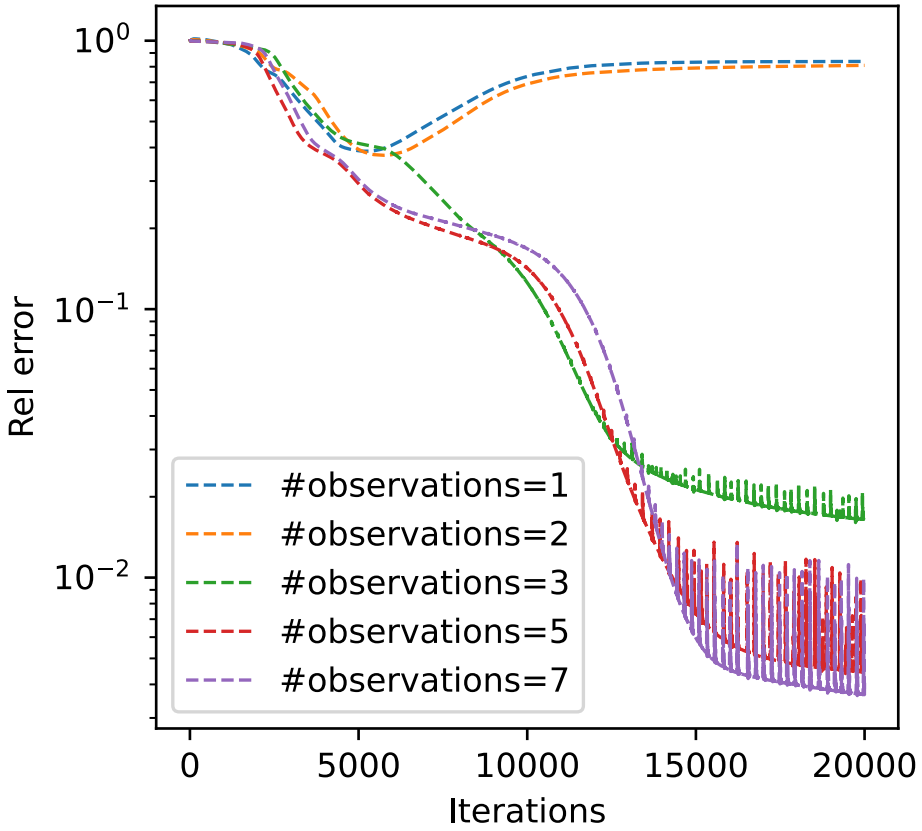
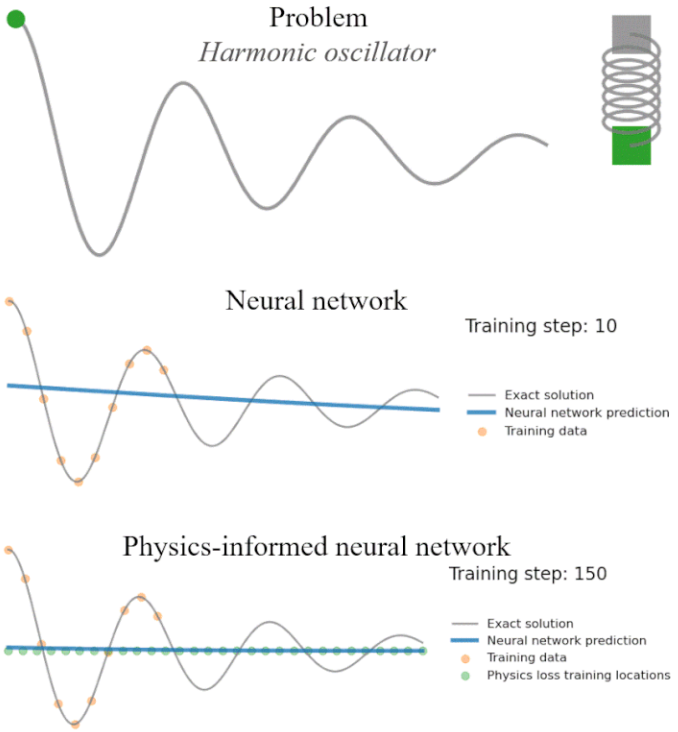


Illustration from Ben Moseley

Balancing observations with physics can lead to better estimates





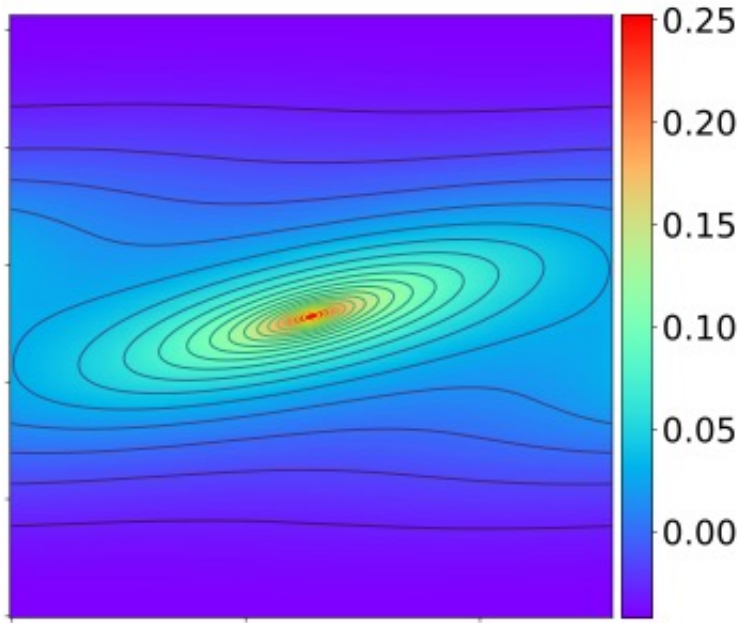
S. Subramanian, M. Kirby, M. Mahoney, A. Gholami: Rethinking the role of data in PINNs, arxiv:2207.04084, 2022.

Sampling bias in physics loss can be detrimental

- How you impose the physics loss matters

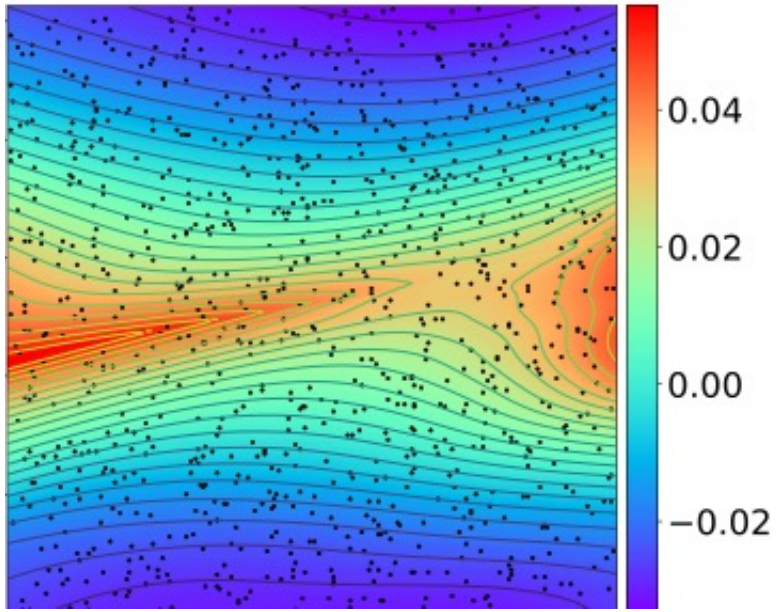
$$-\operatorname{div} \boldsymbol{K}(\boldsymbol{x})\nabla u = f(\boldsymbol{x})$$

$$\mathcal{L}_f = \|\operatorname{div} \boldsymbol{K}(\boldsymbol{x})\nabla u + f(\boldsymbol{x})\|^2$$

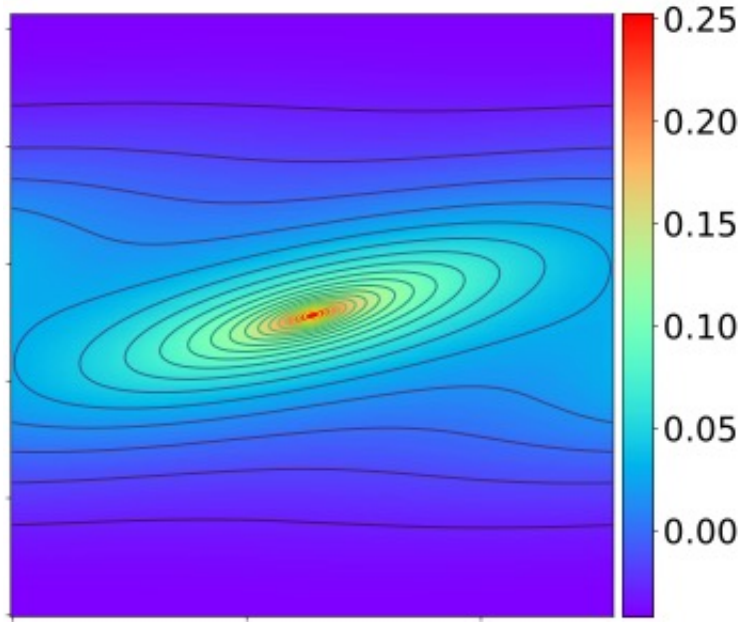


Sampling bias in physics loss can be detrimental

- Sampling uniformly is suboptimal and can result in large errors
  - Testing error ~ 60%



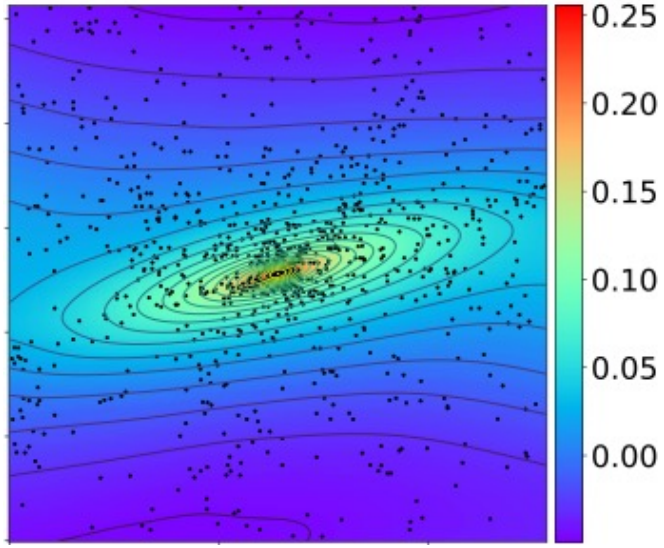
PINN Solution with
Uniform Sampling



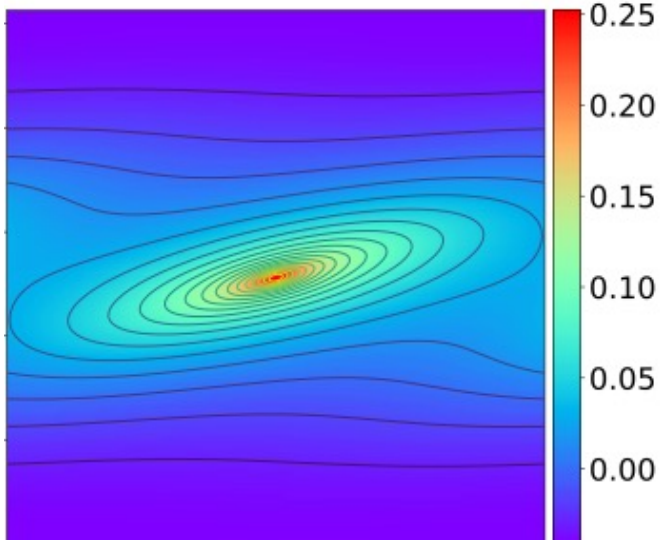
Ground Truth Poisson Solution

Rethinking the collocation points sampling helps

- Don’t add new points, just resample them based on a proxy function
- Resample in windows, regularize with uniform sampling
- Testing error improves from 60% to 10%



PINN Solution with
Adaptive Sampling



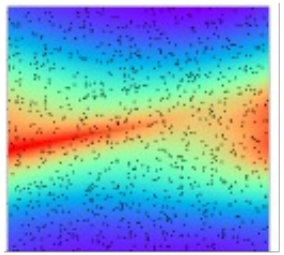
Ground Truth Poisson Solution

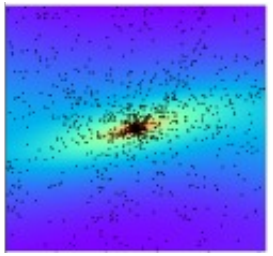
S. Subramanian, M. Kirby, M. Mahoney, A. Gholami: Rethinking the role of data in PINNs, arxiv:2207.04084, 2022.

Conclusions

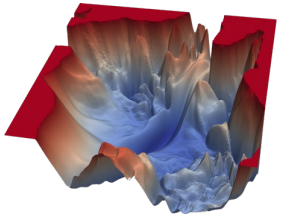
Rethinking the *design*, *training,* and *role of data* for the successful application of neural networks in scientific applications

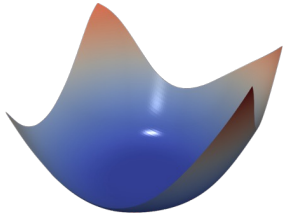
- Adaptive Collocation Points
- PINN Failures: NeurIPS'21
- PyHessian IEEE BigData'20
- Flat/Sharp Minima: NeurIPS'18



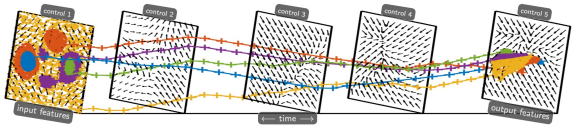


- ANODEV2: NeurIPS'19
- ANODE: IJCAI'19





- On going research on Large Models for Physical Systems



Open Problems

- There are many more open problems:
- **Optimization:**
  - Unlike all other classical ML tasks, PINNs cannot be optimized with mini-batch (SGD, ADAM, etc.) and only works with LBFGS with full batch size
  - This makes training PINNs very slow and hard to optimize

- **NN Architecture:**
  - Classical NN architecture may not be optimal for PINNs. Need to investigate alternative architectures that are more suited for the continuous nature of the problem.
  - Need to investigate how the architecture should be changed as the underlying dynamics change
    - Elliptical vs Hyperbolic vs Parabolic PDEs may need different architectures

Lecture Summary

- We briefly introduced different methods for incorporating physical laws into learning:
  - Neural Operators (NO)
  - Physical laws as hard constraints
  - Physics Informed Neural Networks (PINN)
  - Physics Informed Neural Operators (PINO)
- We specifically focused on PINN and discussed possible challenges in training them

- Physics Informed Neural Networks are easy to implement but there are many subtle issues associated with its soft-regularization method
- PINNs can fail to learn simple problems such as advection, reaction, and/or reaction-diffusion problems with non-trivial coefficients
- Analyzing the problem shows that while the NN has enough capacity to learn the solution, the optimization problem with PINN regularization becomes very difficult to solve