Introduction to Physics Informed Neural Networks

CASML 2024 Preconference Workshop – Day 1

Divij Ghose

AIREX Lab and Zenteiq.ai

AIREX LAB Convenor: Prof. Sashikumaar Ganesan

SciML and Computational Research at AIREX Lab

- We have two broad verticals of computational research at AIREX, IISc

Scientific Machine Learning

- Physics-Informed Neural Networks for Solving PDEs
- Efficient PINNs for solving fluid flow problems
- Artificial Neural Networks augmented FEM (for parameter predictions)
- MLOps, CI/CD, DataOps for deployment of ML Applications

Computational Mathematics

- Turbulence Modelling for Fluid Flows using FEM
- High Performance Computing - OpenMP, MPI (Distributed Computing) & CUDA(GPU) applications of FEM
- Modelling wide variety of real-world applications such as infectious diseases modelling, particle deposition in human air pathways

AIREX Lab

The Team

Session Overview

Introduction to PINNs

An overview of this session

What are Physics Informed Neural Networks?

- What are neural networks?
- How can we train neural networks to solve partial differential equations?

How do we train Physics Informed Neural Networks?

- What are collocation points and how do we sample them?
- What is a "physics loss" and how do we implement it?
- How do we ensure that our solution is unique?

Introduction to PINNs

An overview of this session

What are the types of problems we can solve with PINNs?

- Forward Problems
- Inverse Problems

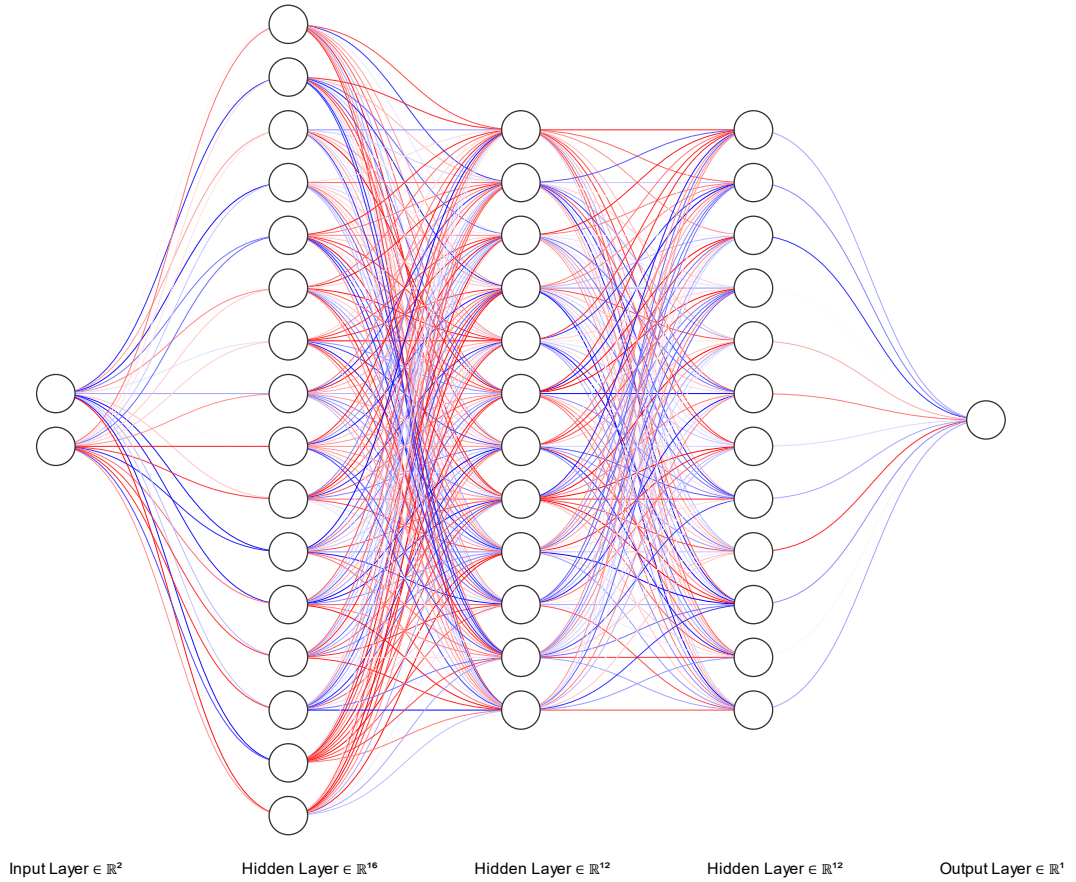
Advantages and Challenges with PINNs

- Where do PINNs shine?
- Where do PINNs fail?

What are Physics Informed Neural Networks?

Introduction to Neural Networks

What is a Neural Network?



Introduction to Neural Networks

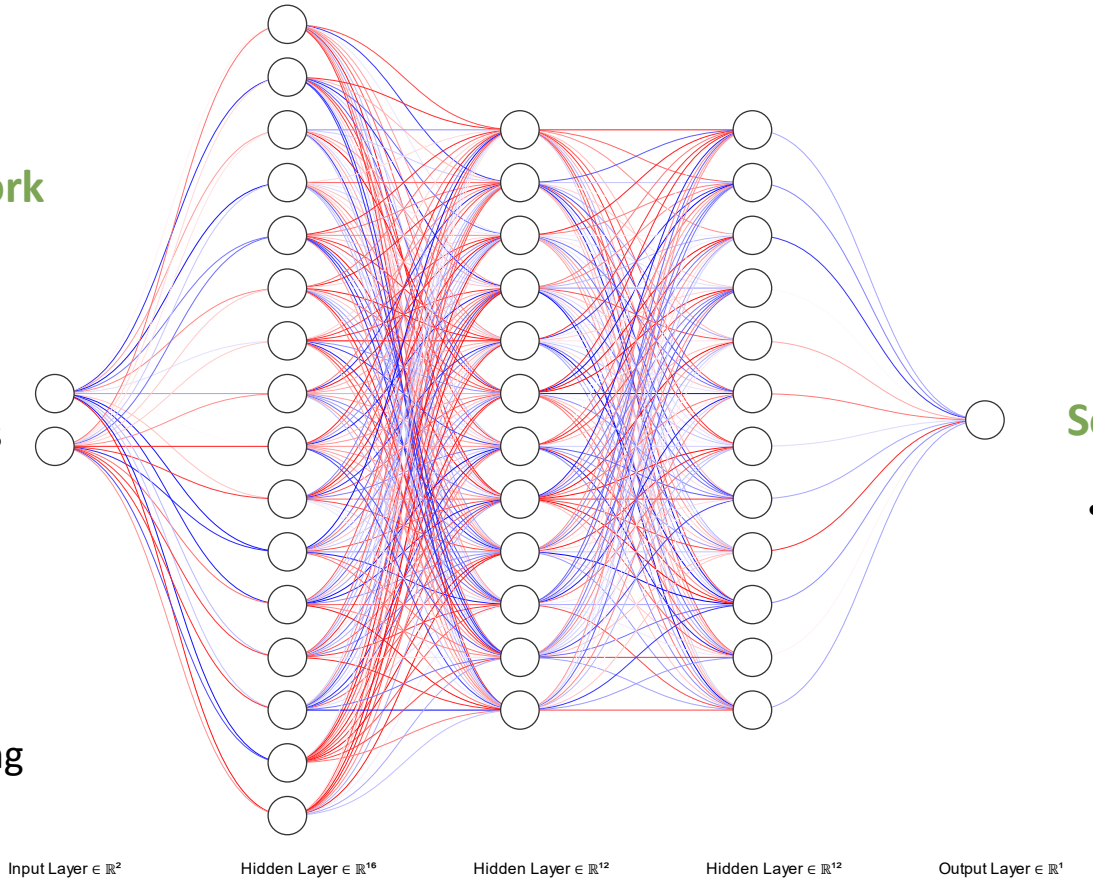
What is a Neural Network?

Elements of a Neural Network

- Neurons with non-linear activation functions
- Affine transformation of inputs to a neuron (weights and biases)

Training a Neural Network

- Minimize a loss function with an optimizer
- Gradients are obtained using **backpropagation**



So, what is a Neural Network?

- A parametric function mapping from the inputs to the outputs.

What are PINNs?

Can Neural Networks be used to solve PDEs?

Governing equation

- Consider the Poisson equation in two dimensions,

$$\begin{aligned} -\Delta u(x) &= f(x), \quad \text{in} \quad \Omega \subseteq \mathbb{R}^2, \\ u(x) &= g(x), \quad \text{on} \quad \partial\Omega. \end{aligned}$$

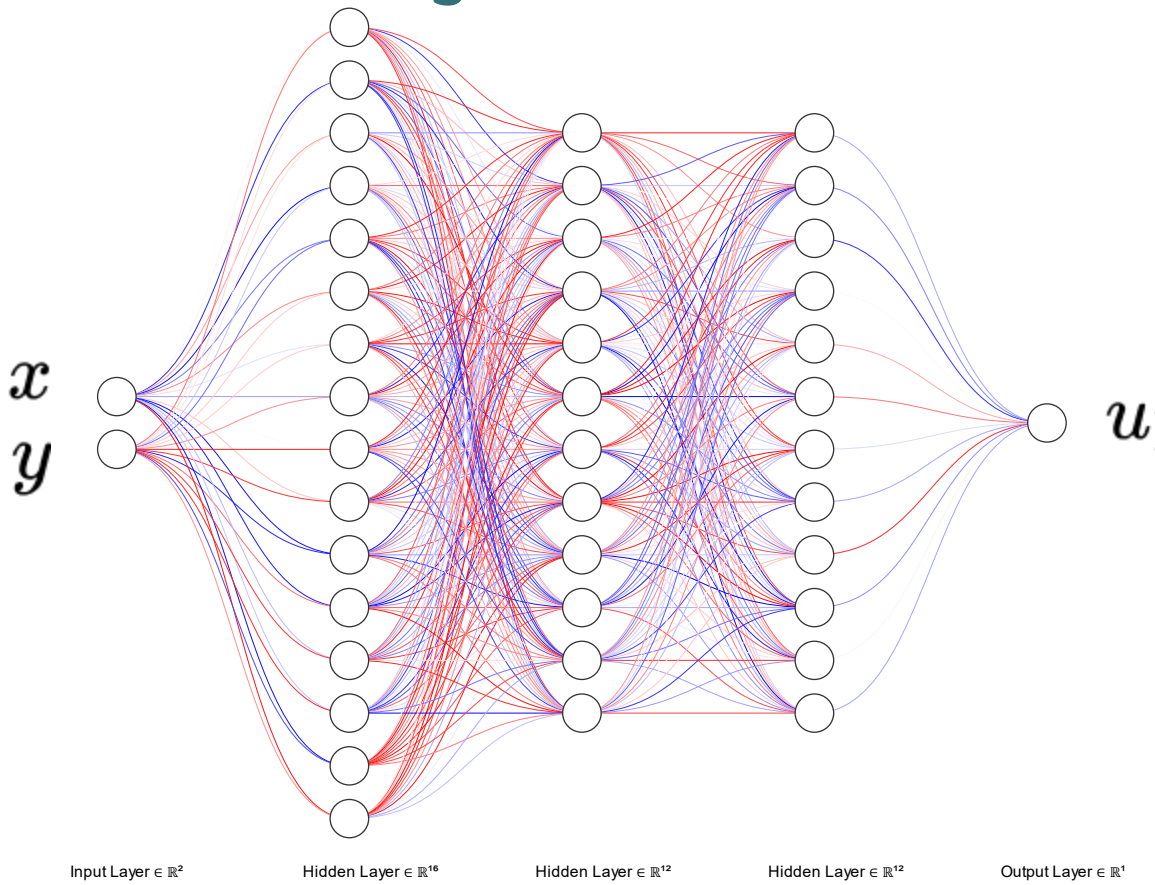
Neural Network function

- Consider the neural network as a parametric function,

$$u_{\mathrm{NN}}(x; W, b) = l \circ \mathrm{T}^{(h)} \circ \mathrm{T}^{(h-1)} \ldots \circ \mathrm{T}^{1}(x)$$

PINNs: the key idea

Approximate the solution using a neural network



$$u_{\mathrm{NN}}(x) \approx u(x)$$

How do we learn solutions using neural networks?

How do PINNs "learn" the solution?

Putting the physics in PINNs

PDE residual

- The PDE residual is evaluated on sample points in the domain using the Neural Network approximation

$$\mathcal{P}(x; W, b) \quad = -\Delta u_{\mathrm{NN}}(x; W, b) - f(x), \qquad \text{in } \Omega,$$

Total PDE Loss

- The PDE loss is defined over all interior points in a mean-squared sense

$$L_p(W, b) \quad = \frac{1}{N_T} \sum_{t=1}^{N_T} |\mathcal{P}(x; W, b)|^2,$$

How do PINNs "learn" the solution?

How can we learn a unique solution?

Boundary residual

- The Boundary residual is evaluated on sample points on the boundary using the Neural Network approximation

$$\mathcal{B}(x; W, b) = u_{\mathrm{NN}}(x; W, b) - g(x), \quad \text{on } \partial\Omega$$

Total Boundary Loss

- The Boundary loss is defined over all boundary points in a mean-squared sense

$$L_b(W, b) = \frac{1}{N_D} \sum_{d=1}^{N_D} |\mathcal{B}(x; W, b)|^2,$$

How do PINNs "learn" the solution?

Combining the two losses

Total PINNs Loss

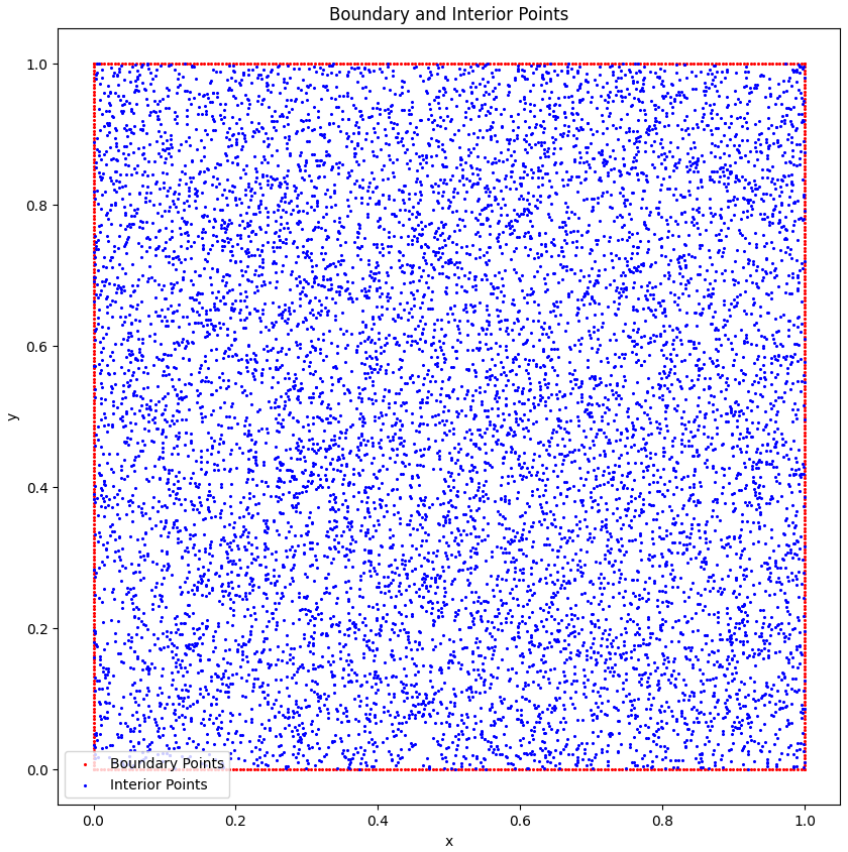
- The two losses are combined to get the final training loss. A scaling hyperparameter is usually used to balance the two components

$$L_{\text{PINN}}(W, b) = L_p + \tau L_b$$

How do we train PINNs?

Sampling points for residuals

How do we sample residual points

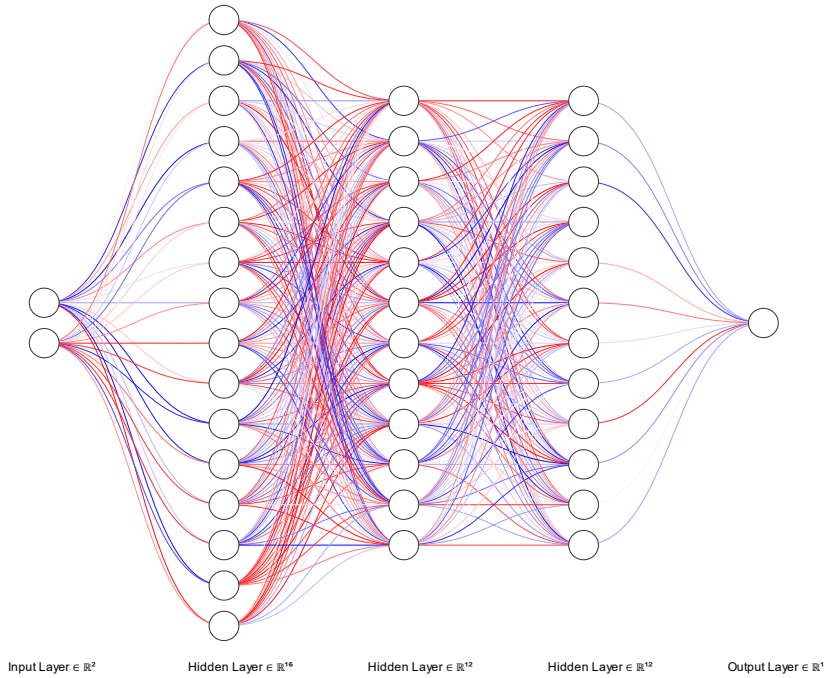


Sampling techniques used in PINNs

- Latin Hypercube Sampling technique is the most popular technique used to generate collocation points
- Uniform sampling generally works well for sampling on the boundary
- Adaptive sampling techniques use more collocation points in areas where the residuals are high
- Variants like hp-VPINNs calculate the residual at quadrature points

Calculating gradients for residuals

The convenience of autodiff in neural networks

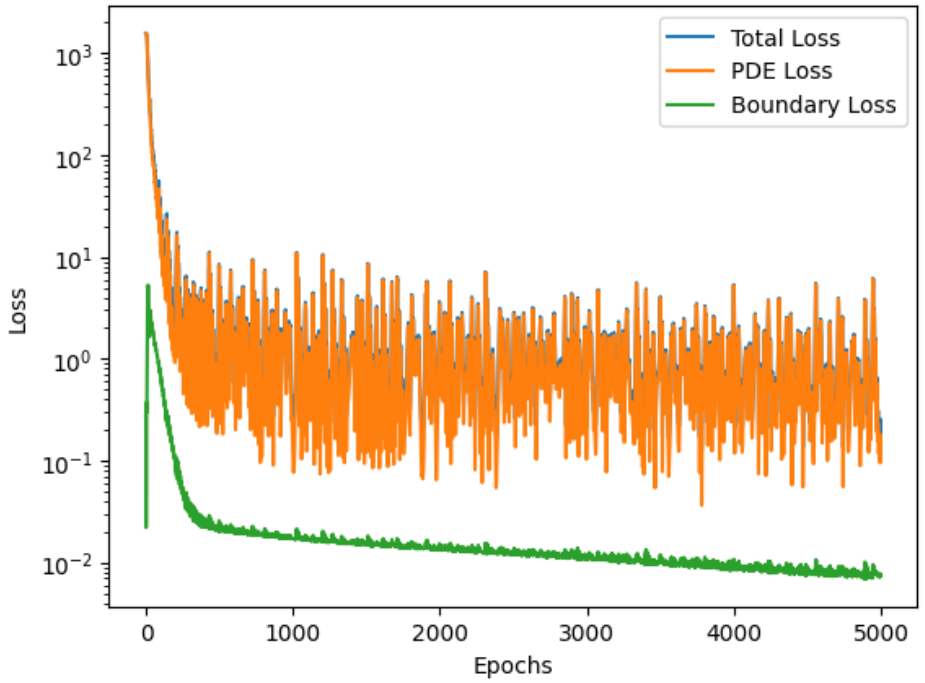


Computing gradients of the solution

- PINNs leverage the automatic differentiation capability of neural networks to calculate the gradients of the solution with respect to the input
- Evaluate the neural network from inputs to outputs, storing intermediate values and operations
- Propagate gradients backward through the network using the chain rule, layer by layer.
- Computes gradients for all parameters in a single backward pass, regardless of network size.
- Recursively apply autodiff to obtain higher derivatives

Training the network with the PINNs loss

Bringing it all together



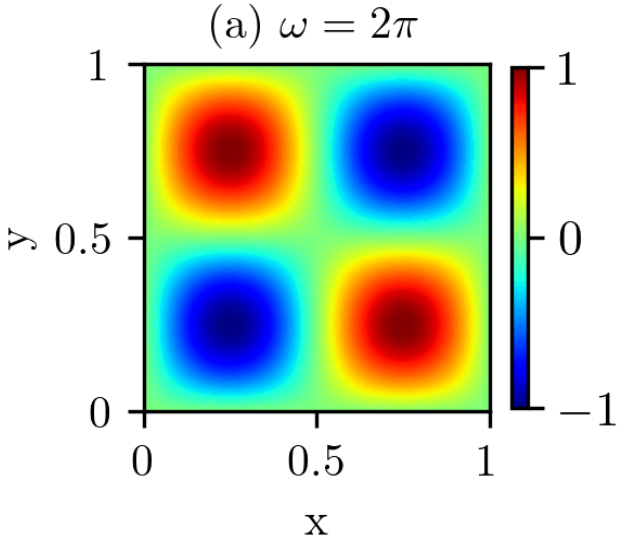
Training the network parameters

- Calculate the PINNs loss with a suitable weighting between the two components
- Compute the gradient of this loss with respect to the network parameters
- Perform a gradient descent step
- Use a stopping criterion (threshold on error or maximum number of iterations)

Solving PDEs with PINNs

Solving forward problems

Finding the unknown solution of a PDE



An example problem

Consider the Poisson equation with the given forcing function and boundary conditions

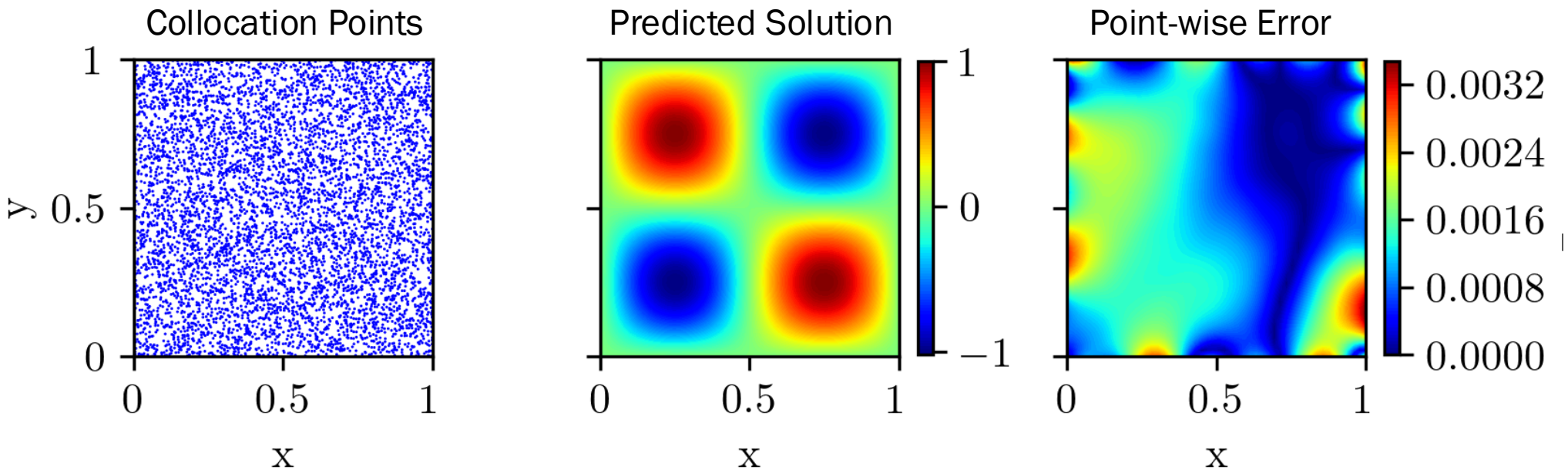
$$\begin{aligned} -\Delta u(x,y) &= -2\omega^2 \sin(\omega x)\sin(\omega y) \quad \text{in} \in \Omega = (0,1)^2, \\ u(0,\cdot) &= u(\cdot,0) = 0. \end{aligned}$$

This problem has the exact solution

$$u(x,y) = -\sin(\omega x)\sin(\omega y).$$

The PINNs solution

Solving the Poisson equation



Solving inverse problems

Estimating unknown parameters

Incorporating data in the PINNs loss

To estimate an unknown parameter in the PDE, we can sample the solution at "sensor points" and have a sensor residual

$$\mathcal{D}(x; W, b, \varepsilon) \quad = u_{\mathrm{NN}}(x; W, b, \varepsilon) - u(x), \qquad \text{in } \Omega.$$

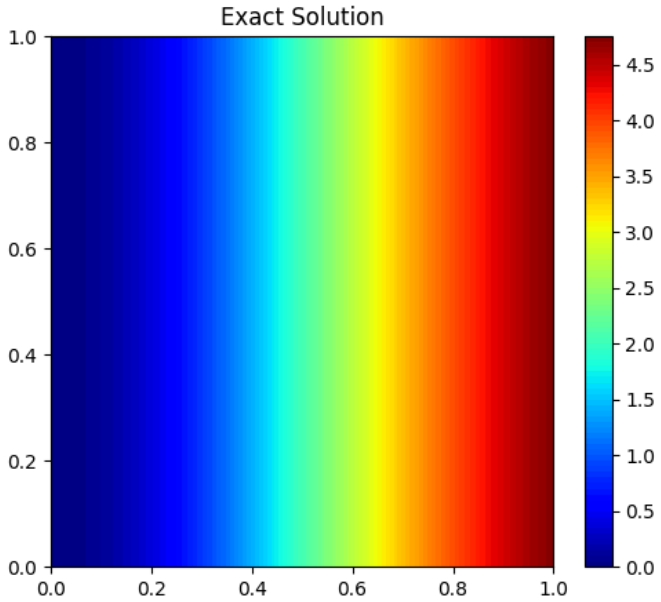
Where ε is the unknown parameter in the PDE, considered as a "trainable" variable

The total sensor loss is obtained across all sensor points

$$L_d(W, b, \varepsilon) \quad = \frac{1}{N_L} \sum_{l=1}^{N_L} \left| \mathcal{D}(x; W, b, \varepsilon) \right|^2,$$

Solving inverse problems

Finding the unknown solution of a PDE and an unknown parameter



An example problem

Consider the following equation with the given exact solution and an unknown parameter

$$-\varepsilon \Delta u(x) = f(x), \quad x \in (-1,1)^2$$

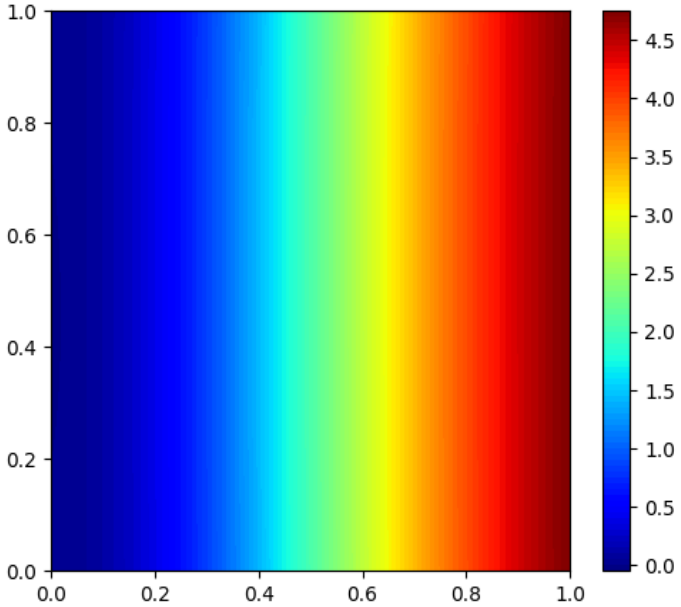
$$u(x) = 10 \sin(x) \tanh(x) e^{-\varepsilon x^2}$$

The value of the parameter is 0.3, and we start with an initial guess of 2

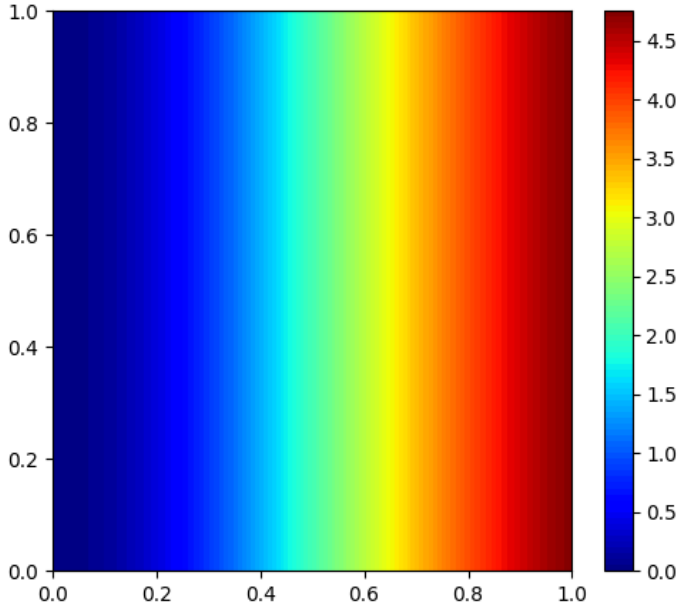
Inverse Solution

Finding the solution and unknown parameter

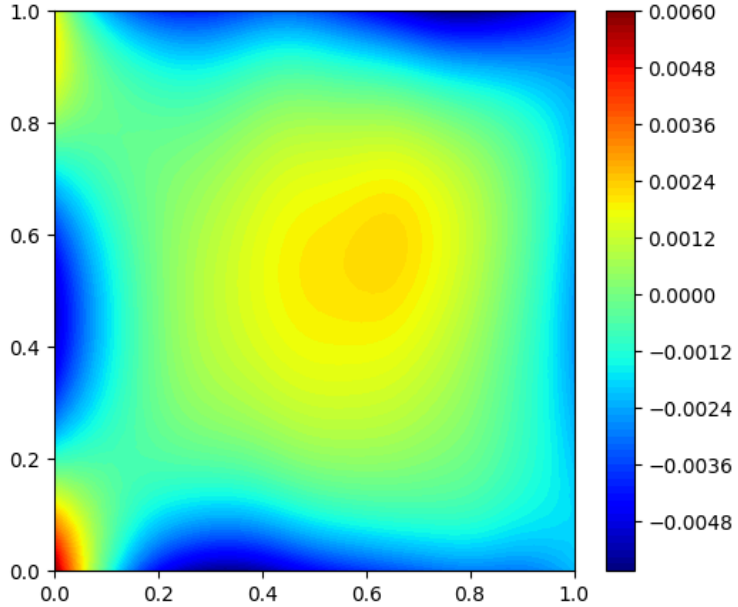
Predicted Solution



Exact Solution

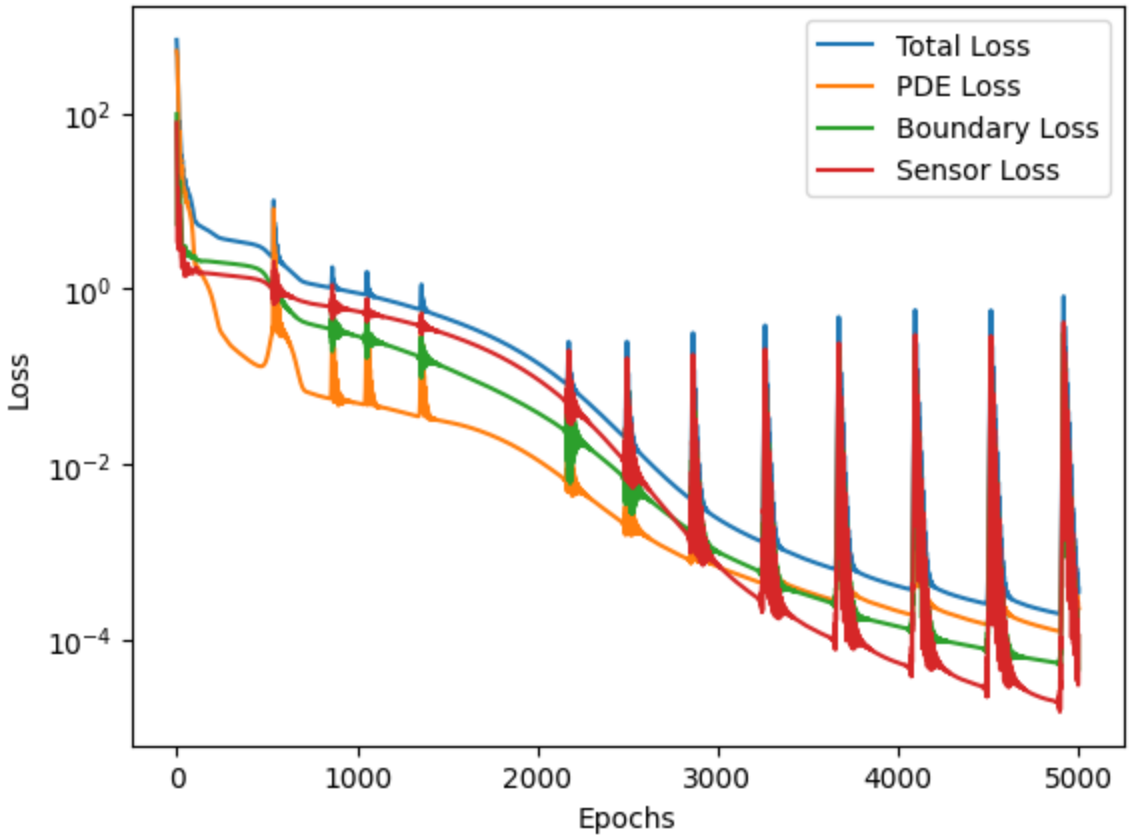


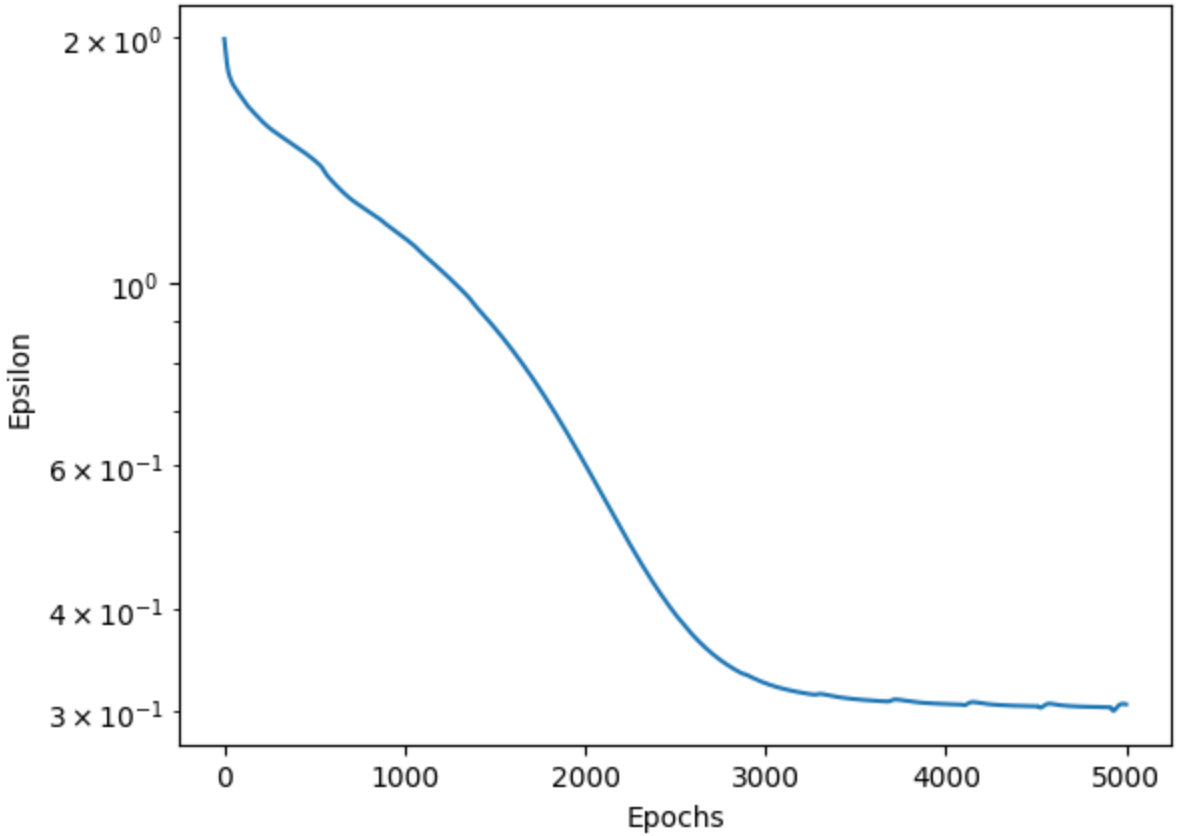
Point-wise Error



Inverse Solution

Finding the solution and unknown parameter





Advantages and Challenges with PINNs

PINNs: A summary

Are PINNs supervised or unsupervised?

A perspective on PINNs

- PINNs can be seen as supervised learning because of the boundary (and sensor) losses
- We add an unsupervised physics **regularizer** to have a physically consistent solution

$$L_p(W,b) = \frac{1}{N_T}\sum_{t=1}^{N_T}|\mathcal{P}(x;W,b)|^2,$$

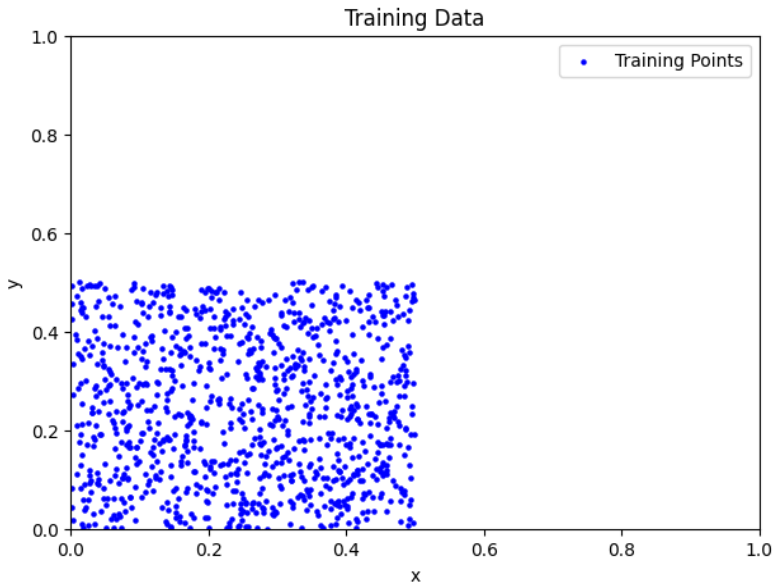
$$L_b(W,b) = \frac{1}{N_D}\sum_{d=1}^{N_D}|\mathcal{B}(x;W,b)|^2,$$

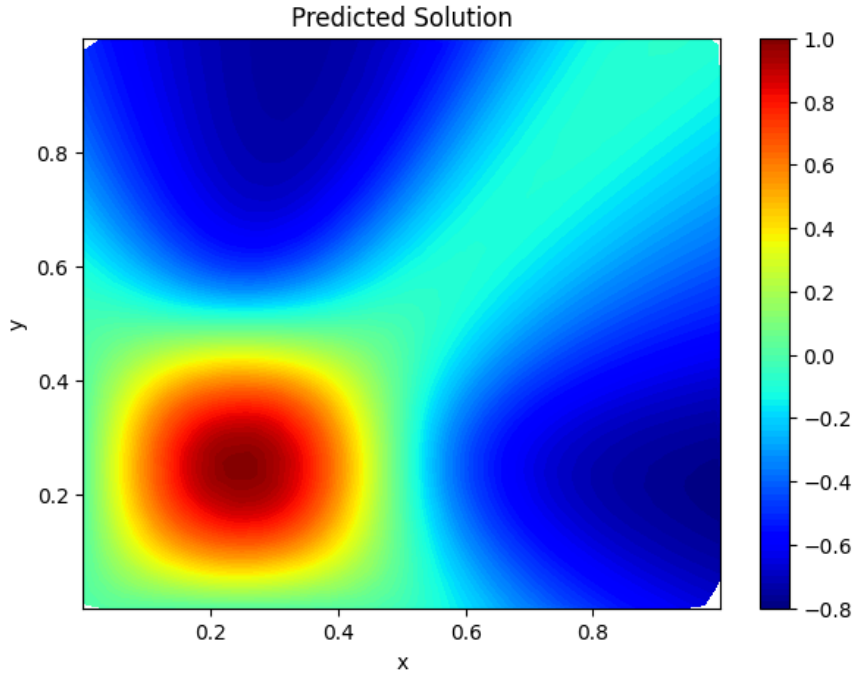
$$L_d(W,b,\varepsilon) = \frac{1}{N_L}\sum_{l=1}^{N_L}|\mathcal{D}(x;W,b,\varepsilon)|^2,$$

PINNs: A summary

The effect of incorporating physics

Why is a physics loss useful?

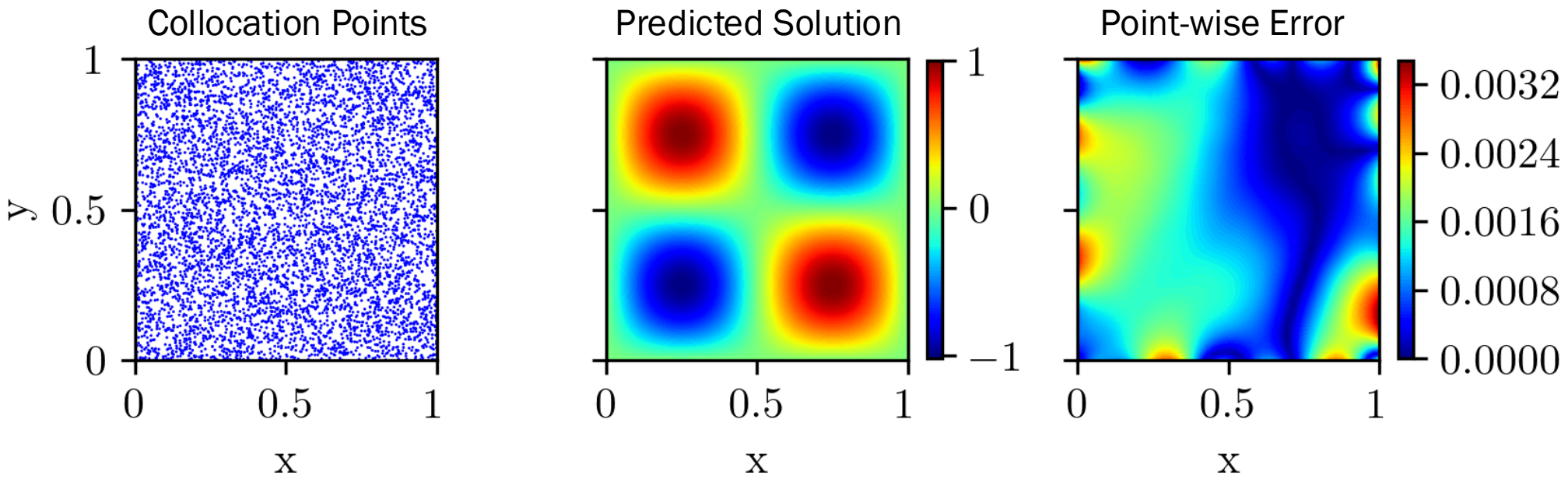




PINNs: A summary

The effect of incorporating physics

Why is a physics loss useful?

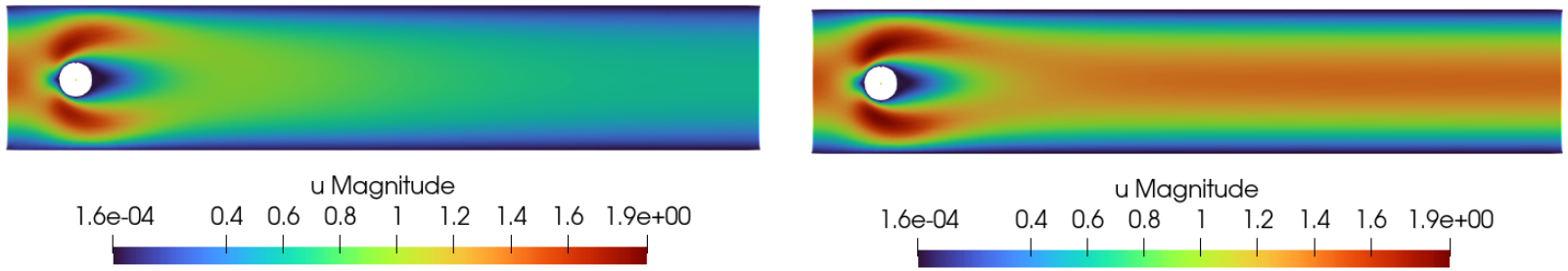


PINNs: Advantages

Why are PINNs useful?

Meshless and powerful for inverse problems

- PINNs are meshless – gradients are computed using autodiff
- Both forward and inverse problems can be solved in one pass
- Data assimilation and parameter estimation are far easier than in conventional methods
- Solutions can be obtained at unknown points at the speed of inference



PINNs: Challenges

Where do PINNs struggle?

Approximation and training errors

- PINNs inherit all problems of neural networks - approximation, estimation and optimization errors
- The model needs to be sufficiently *expressive* to mitigate approximation errors
- We need to have enough collocation points to mitigate estimation errors
- We need to train sufficiently with an appropriate optimizer to mitigate optimization errors
- Neural networks struggle to learn steep solutions and high frequency solutions (*spectral bias*)

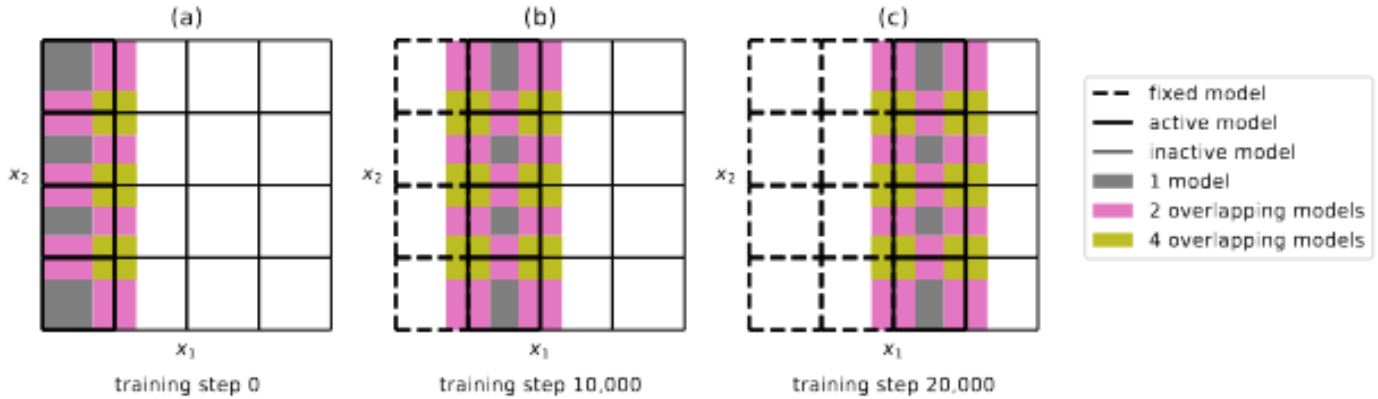
Addressing Challenges with PINNs – some case studies

Finite Basis PINNs

Domain Decomposition with PINNs

Mitigating spectral bias

- FBPINNs use overlapping subdomains, with a network assigned to each subdomain
- The networks share solutions and gradients across the overlapping region
- Window functions are used to restrict solutions within the subdomain
- Scheduling is used to propagation information from the boundaries

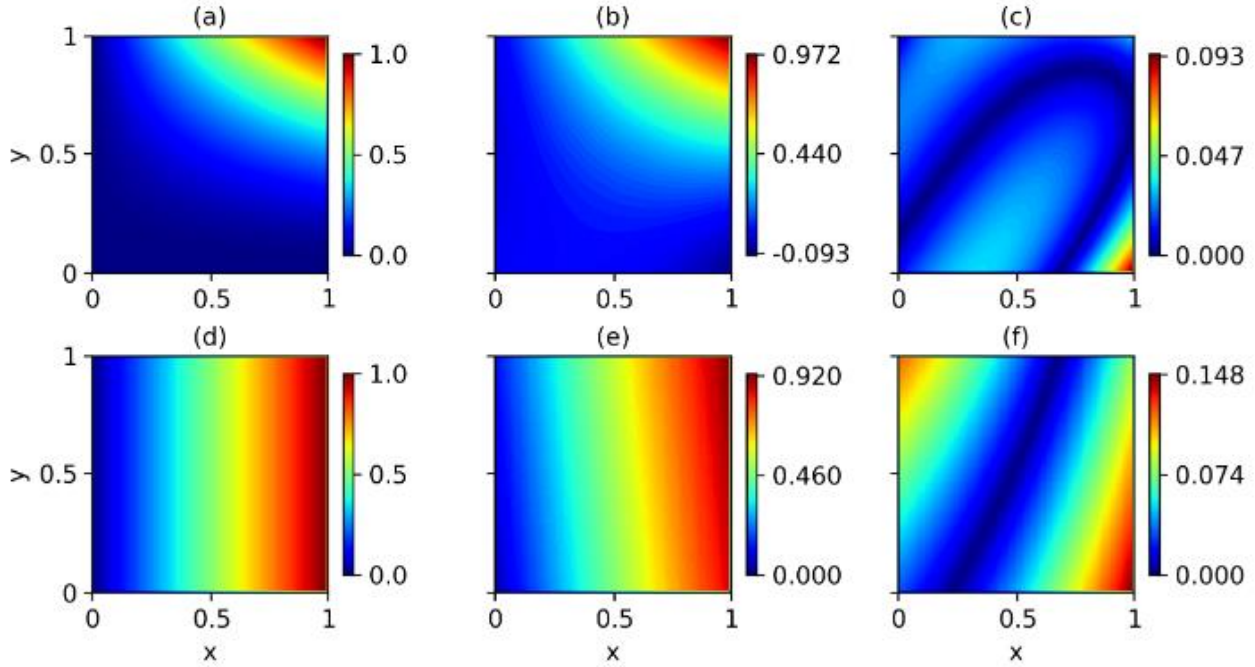


SUPG hp-VPINNs

Using FEM techniques in the context of PINNs

Failure of PINNs in singularly perturbed problems

- PINNs fail to capture steep solutions in the layer region for convection-dominated problems

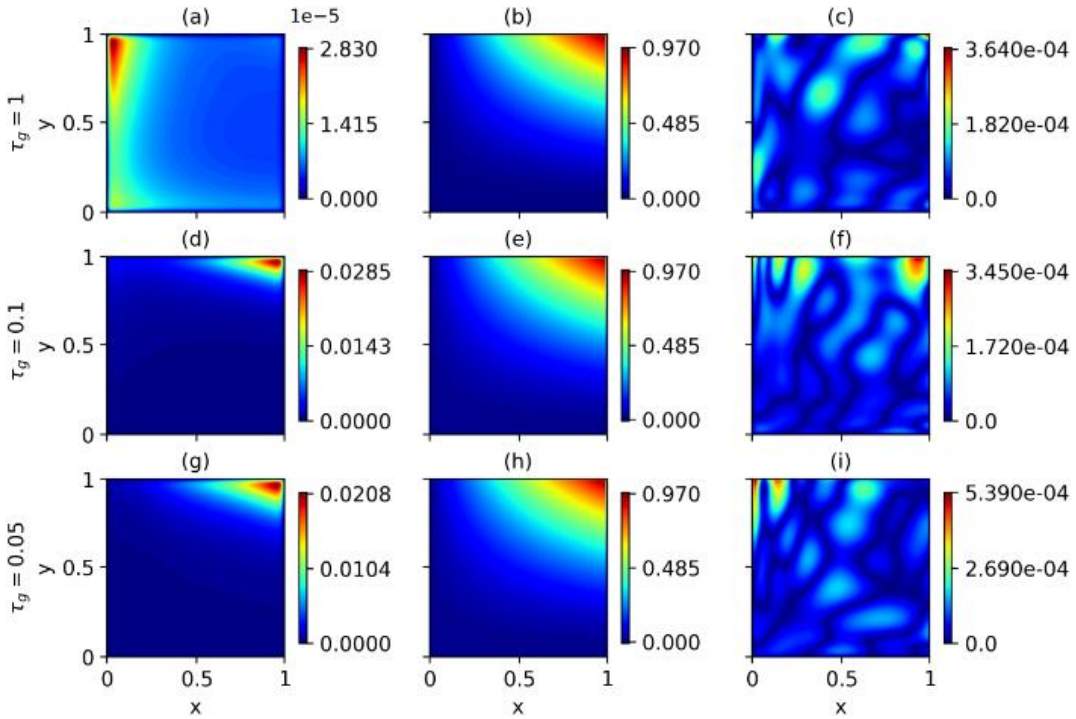


SUPG hp-VPINNs

Using FEM techniques in the context of PINNs

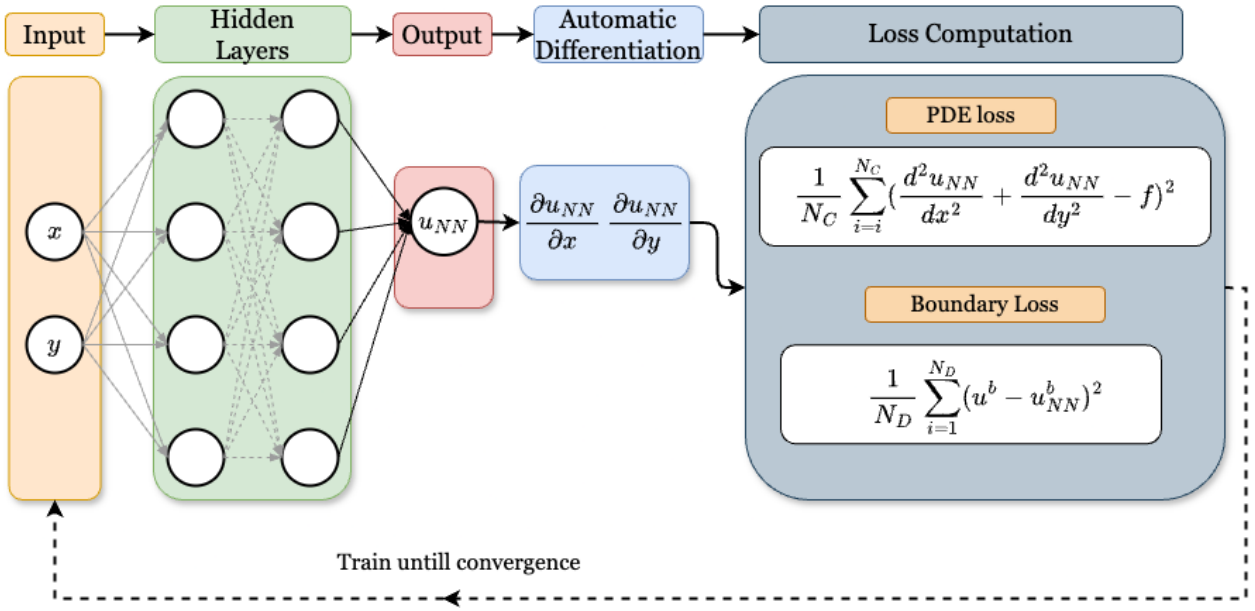
Adding an SUPG stabilization

- We introduce stabilization and gradient conforming hard constraints



Final thoughts

Q&A



AIREX Lab

Divij's profile