AI in the Sciences and Engineering

PINNs – Limitations and Extensions – Part 2

Spring Semester 2024

Siddhartha Mishra
Ben Moseley

ETH zürich

Recap - advantages / limitations of PINNs

Advantages

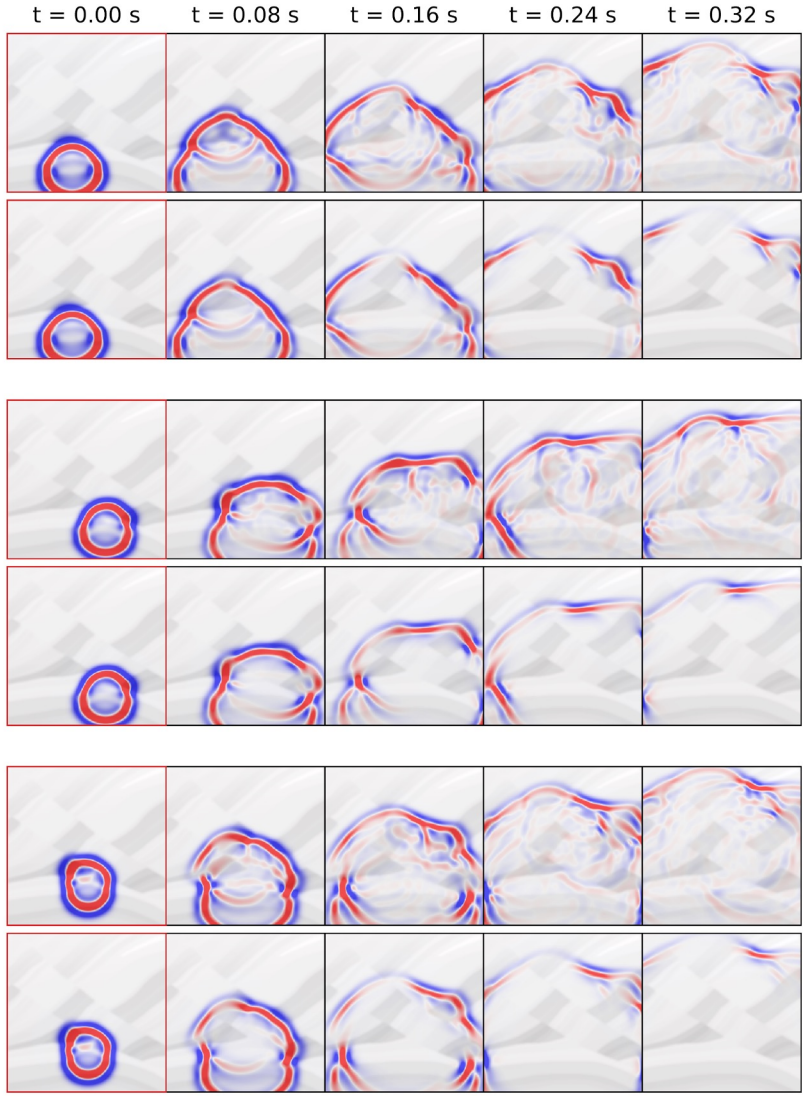
- **Mesh-free**
- Can jointly solve **forward** and **inverse** problems
- Often performs well on “messy” problems (where some observational data is available)
- Tractable, analytical solution gradients (e.g. for sensitivity analysis)
- Mostly **unsupervised**

Limitations

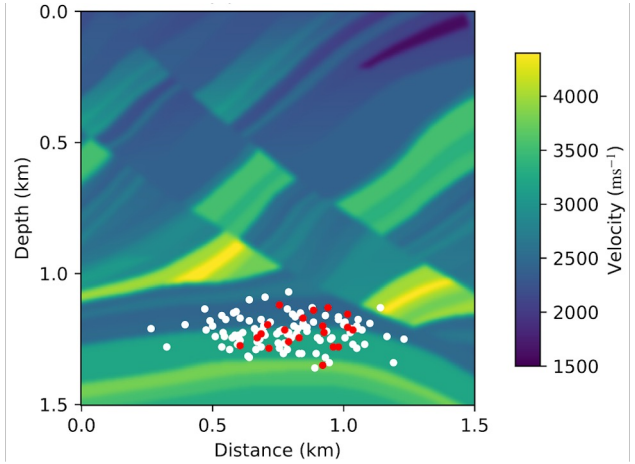
- **Computational cost** often high (especially for forward-only problems)
- Can be hard to **optimise** (and convergence properties less well understood)
- Challenging to **scale** to larger domains, multi-scale, multi-physics problems

But.. many PINN extensions exist!

Recap - conditioned PINNs

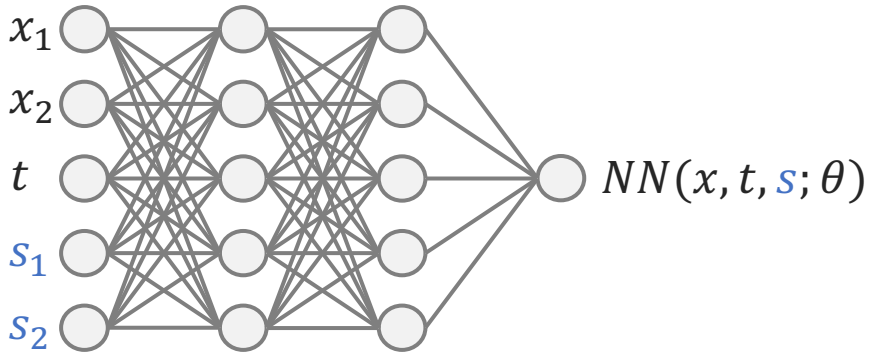


Velocity model, $c(x)$



White = random source locations used for training

Red = source locations used for testing



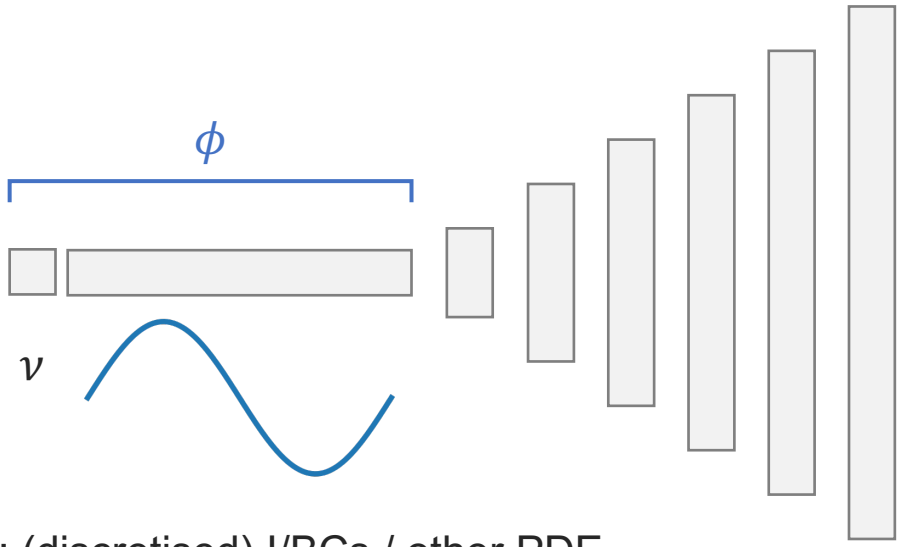
Means the network **does not** need to be retrained for each simulation => much faster!

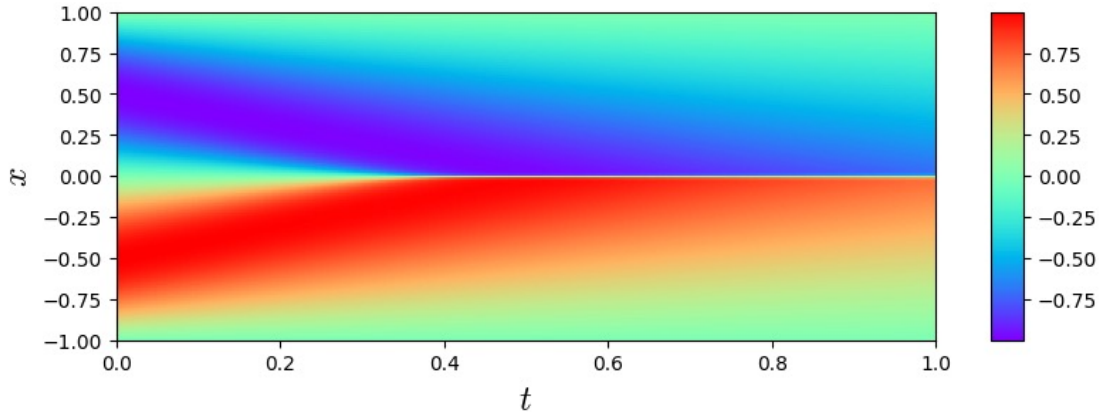
Aka a **surrogate** model

Moseley et al, Solving the wave equation with physics-informed deep learning, ArXiv (2020)

Recap - conditioned discretised PINNs

Idea: **discretise** solution and use e.g. convolutional network to learn spatial/temporal correlations
And condition on I/BCs / other PDE parameters





Input: (discretised) I/BCs / other PDE parameters

$$NN(\phi;\theta)_{ij} \approx u(x = x_i, t = t_j)$$

$$L_p(\theta) = \frac{1}{N_x N_t N_\phi} \sum_{k}^{N_\phi} \sum_{i}^{N_x} \sum_{j}^{N_t} \left(\frac{\delta NN(\phi_k;\theta)_{ij}}{\delta t} + NN(\phi_k;\theta)_{ij} \frac{\delta NN(\phi_k;\theta)_{ij}}{\delta x} - \phi_{k0} \frac{\delta^2 NN(\phi_k;\theta)_{ij}}{\delta x^2} \right)^2$$

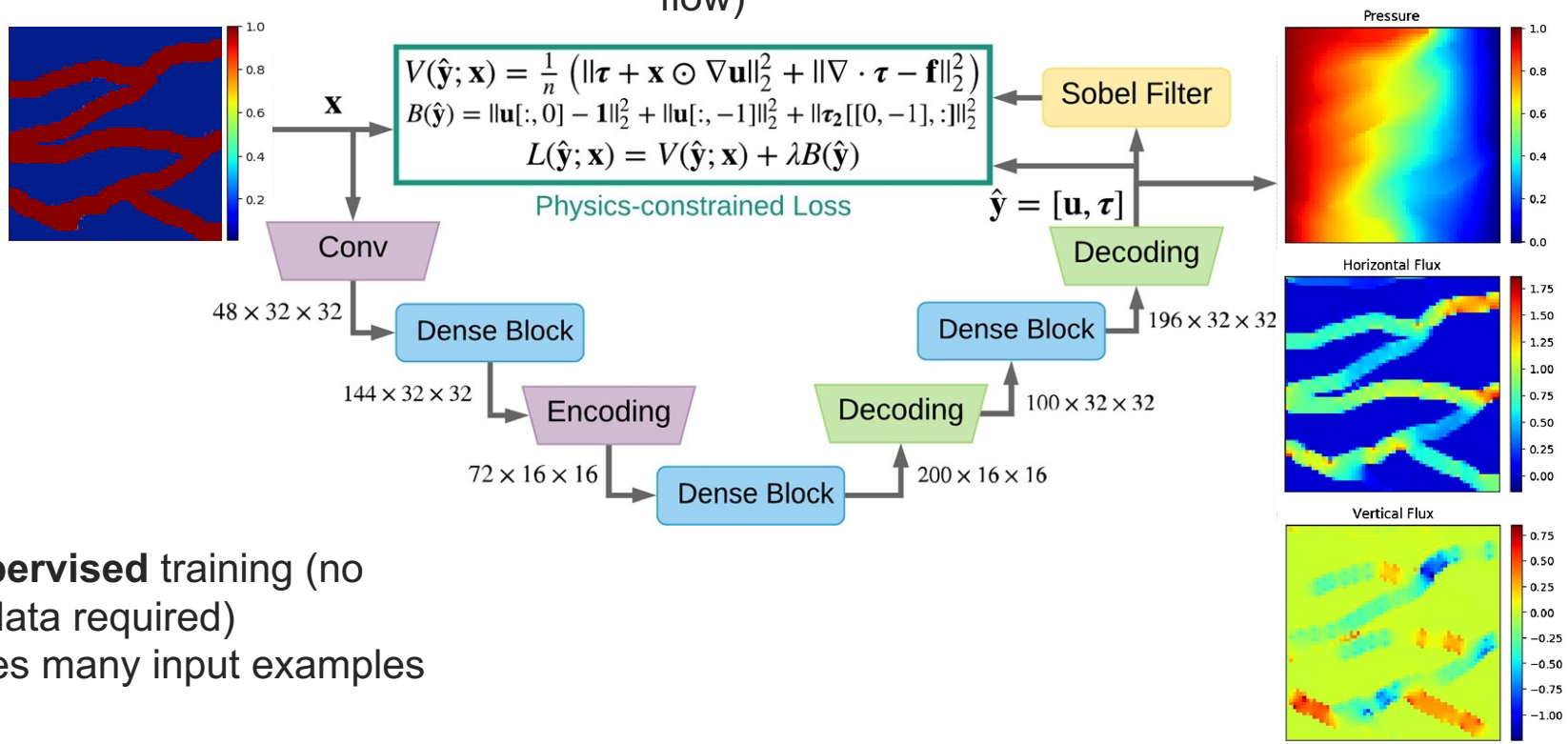
Conditioned discretised PINN

Recap - conditioned discretised PINNs

Input: permeability field

PDE: steady-state flow in heterogeneous media (Darcy flow)

Output: fluid pressure and flux field



- Fully **unsupervised** training (no simulation data required)
- Only requires many input examples to train

Zhu, Y et al, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics (2019)

Overview of lectures

- PINN limitations
- PINN extensions for improving:
  - Computational cost
  - Convergence / accuracy
  - Scalability to more complex problems
- Summary: when should I use PINNs?

~~Part 1~~

Part 2

Learning objectives

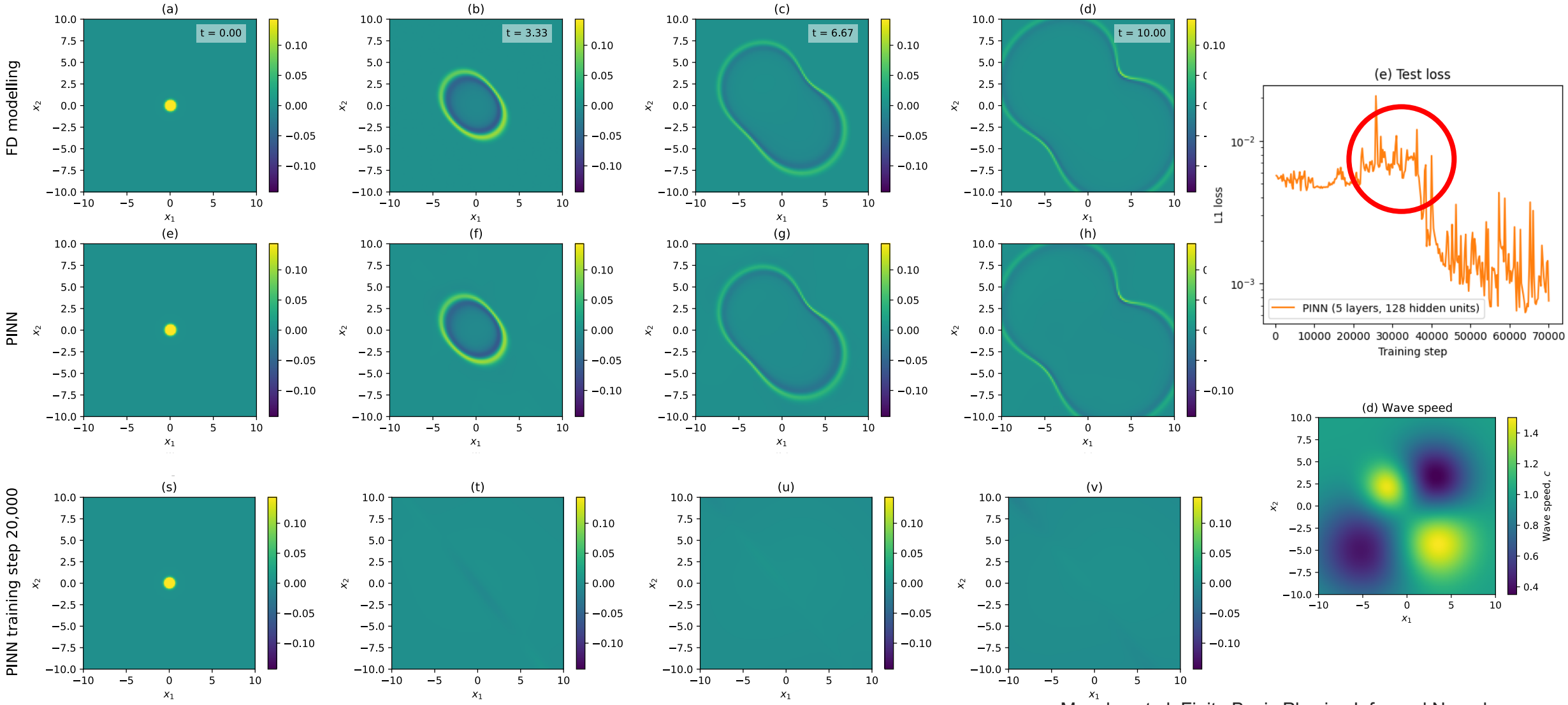
- Explain the advantages and disadvantages of PINNs
- Understand current research directions on improving their performance

Course timeline

Tutorials		Lectures			
Mon 12:15-14:00 HG E 5		*Wed 08:15-10:00 ML H 44*		*Fri 12:15-13:00 ML H 44*	
19.02.		21.02.	~~Course introduction~~	23.02.	~~Introduction to deep learning I~~
26.02.	~~Introduction to PyTorch~~	28.02.	~~Introduction to deep learning II~~	01.03.	~~Importance of PDEs in science~~
04.03.	~~CNNs and surrogate modelling~~	06.03.	~~Introduction to physics-informed neural networks~~	08.03.	~~PINNs – limitations and extensions I~~
11.03.	~~Implementing PINNs I~~	13.03.	**PINNs – limitations and extensions II**	15.03.	PINNs – theory
18.03.	Implementing PINNs II	20.03.	Introduction to operator learning	22.03.	DeepONets and spectral neural operators
25.03.	Operator learning I	27.03.	Fourier- and convolutional- neural operators	29.03.	
01.04.		03.04.		05.04.	
08.04.	Operator learning II	10.04.	Operator learning – limitations and extensions	12.04.	Introduction to transformers
15.04.		17.04.	Foundational models for operator learning	19.04.	Graph neural networks for PDEs
22.04.	GNNs	24.04.	GNNs for PDEs / introduction to diffusion models	26.04.	Introduction to diffusion models
29.04.	Transformers	01.05.		03.05.	Diffusion models - applications
06.05.	Diffusion models	08.05.	Introduction to differentiable physics	10.05.	Hybrid workflows
13.05.	Coding autodiff from scratch	15.05.	Neural differential equations	17.05.	Introduction to JAX
20.05.		22.05.	Symbolic regression and equation discovery	24.05.	Course summary and future trends
27.05.	Introduction to JAX / NDEs	29.05.	Guest lecture: ML in chemistry and biology	31.05.	Guest lecture: ML in chemistry and biology

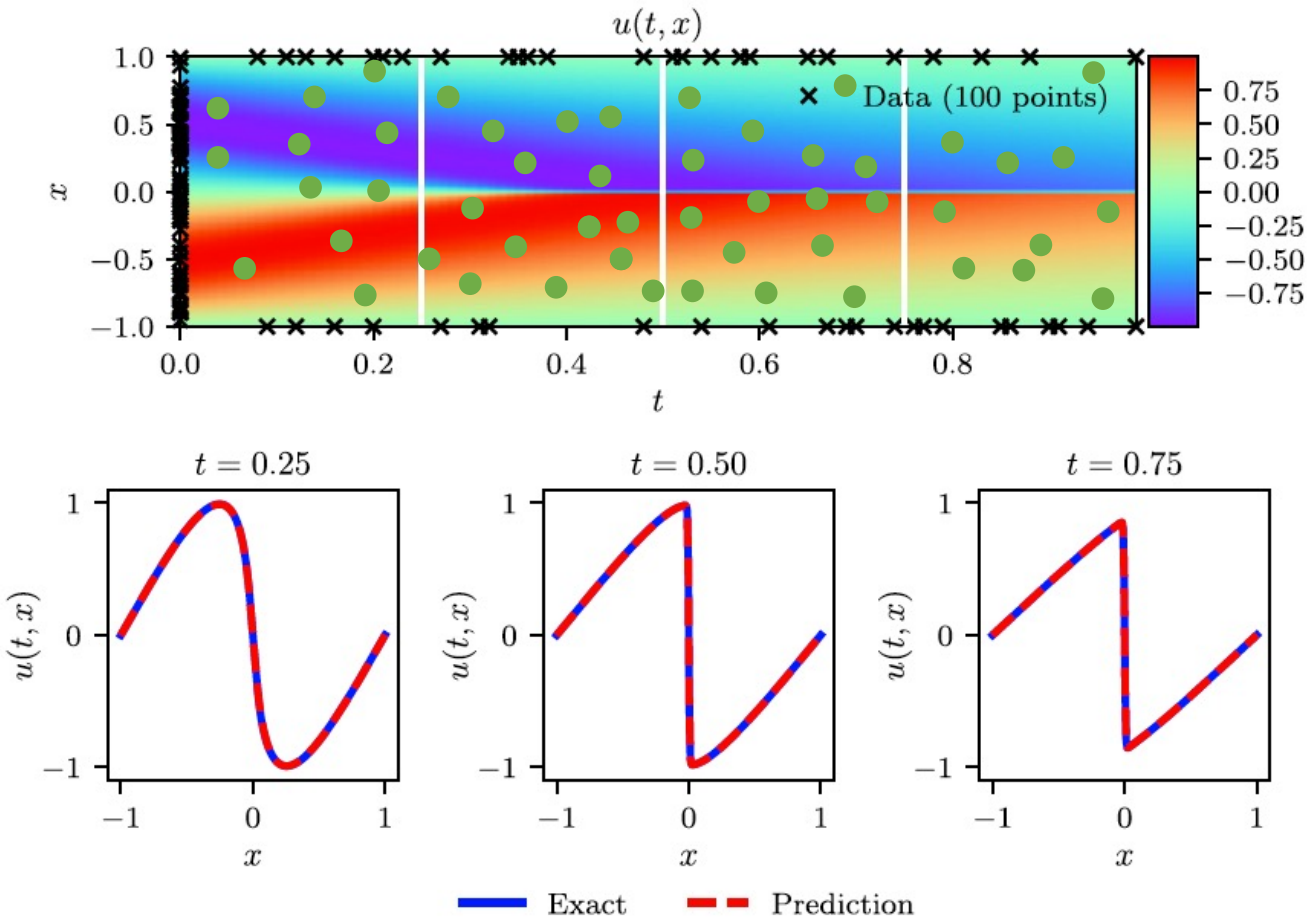
Limitation 2) – poor convergence

PINN solution “thrashing”



Moseley et al, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, ACM (2023)

Competing loss terms



$$L_b(\theta) = \frac{\lambda_1}{N_{b1}} \sum_j^{N_{b1}} \left(NN(x_j, 0; \theta) + \sin(\pi x_j) \right)^2$$
$$+ \frac{\lambda_2}{N_{b2}} \sum_k^{N_{b2}} (NN(-1, t_k; \theta) - 0)^2$$
$$+ \frac{\lambda_3}{N_{b3}} \sum_l^{N_{b3}} (NN(+1, t_l; \theta) - 0)^2$$
$$L_p(\theta) = \frac{1}{N_p} \sum_i^{N_p} \left(\left(\frac{\partial NN}{\partial t} + NN \frac{\partial NN}{\partial x} - \nu \frac{\partial^2 NN}{\partial x^2} \right) (x_i, t_i; \theta) \right)^2$$

How do we choose λ_1, λ_2, and λ_3?

λ too small => doesn't learn unique solution
λ too large => only learns boundary condition

Thus, there can be **competing** terms in the loss function

Raissi et al, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, JCP (2018)

Hard initial / boundary conditions

Idea: use neural network as part of a solution **ansatz**

Burgers' problem above:

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

$$u(x,0) = -\sin(\pi x)$$
$$u(-1,t) = u(+1,t) = 0$$

Then let the solution approximated by

$$\hat{u}(x,t;\theta) = (x-1)(x+1)(t-0)NN(x,t;\theta) - \sin(\pi x)$$
$$\approx u(x,t)$$

It is easy to verify that the initial / boundary conditions are now satisfied by **construction**

Lagaris et al, Artificial neural networks for solving ordinary and partial differential equations, IEEE (1998)

Hard initial / boundary conditions

Idea: use neural network as part of a solution **ansatz**

Burgers' problem above:

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

$$u(x,0) = -\sin(\pi x)$$
$$u(-1,t) = u(+1,t) = 0$$

Then let the solution approximated by

$$\hat{u}(x,t;\theta) = (x-1)(x+1)(t-0)NN(x,t;\theta) - \sin(\pi x)$$
$$\approx u(x,t)$$

It is easy to verify that the initial / boundary conditions are now satisfied by **construction**

Lagaris et al, Artificial neural networks for solving ordinary and partial differential equations, IEEE (1998)

Thus, the boundary loss $L_b(\theta)$ is not needed and the problem is turned into an **unconstrained** optimisation problem, where we minimize

$$L(\theta) = L_p(\theta) = \frac{1}{N_p}\sum_i^{N_p}\left(\left(\frac{\partial \hat{u}}{\partial t} + \hat{u}\frac{\partial \hat{u}}{\partial x} - \nu\frac{\partial^2 \hat{u}}{\partial x^2}\right)(x_i,t_i;\theta)\right)^2$$

Note:

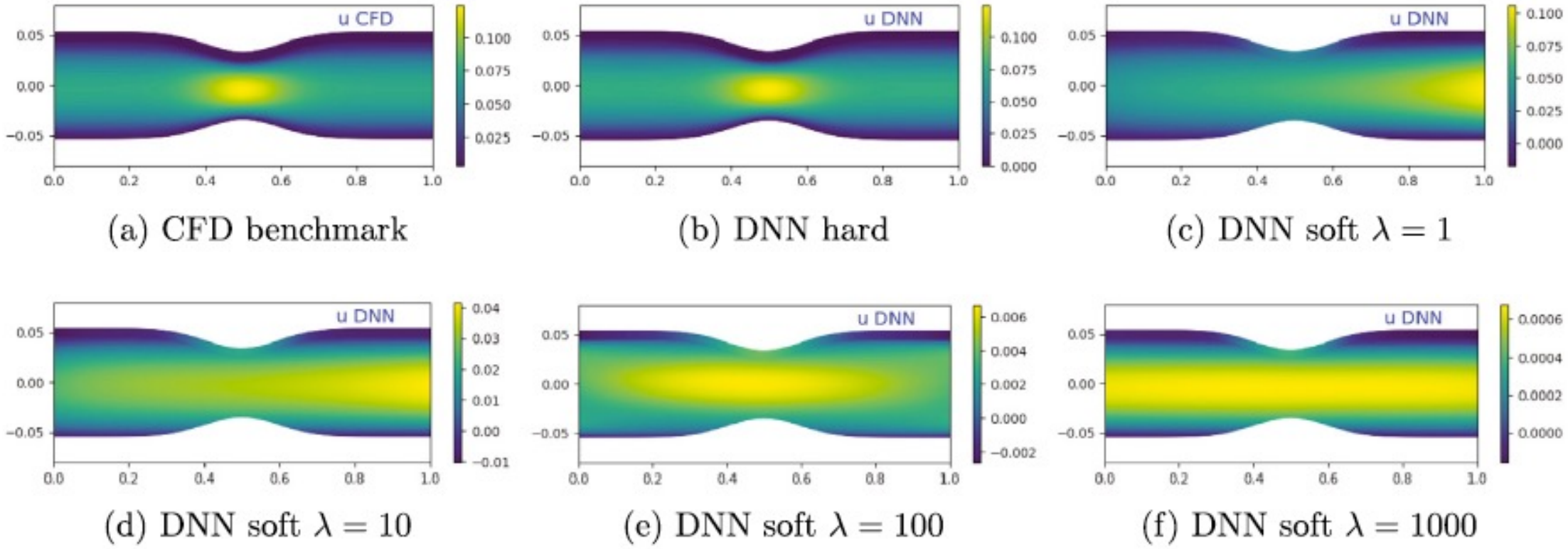
- You can think of the ansatz as a custom final layer of the network, asserting a **hard** constraint on the network's output

- Autodifferentiation lets us differentiate through ansatz

- Training is now fully **unsupervised** (only need collocation points)

- Can be challenging to use this approach for complex boundary conditions

Hard vs soft PINN



Sun et al, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Computer Methods in Applied Mechanics and Engineering (2020)

Soft PINN with varying λ vs hard PINN
When solving incompressible Navier-Stokes equations

Adaptive lambdas

Idea: Specify **separate** λ for each training point
And treat λ as **learnable**

$$L(\theta) = L_b(\theta) + L_p(\theta)$$

$$L_b(\theta, \lambda) = \sum_k \sum_j \lambda_{kj} \left\| \mathcal{B}_k\left[NN(x_{kj}; \theta)\right] - g_k(x_{kj}) \right\|^2$$

$$L_p(\theta, \lambda) = \sum_i \lambda_i \left\| \mathcal{D}[NN(x_i; \theta)] - f(x_i) \right\|^2$$

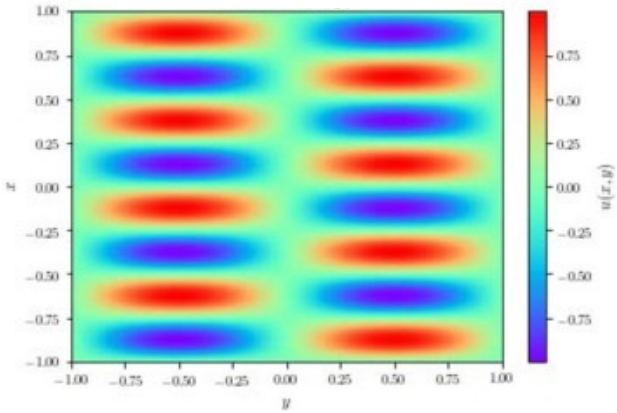
Loop:

1. Compute gradients $\frac{\partial L}{\partial \theta_l}, \frac{\partial L}{\partial \lambda_l}$
2. Update weights,
   $\theta_l \leftarrow \theta_l - \gamma \frac{\partial L}{\partial \theta_l}$ (minimise L)
3. Update lambdas,
   $\lambda_l \leftarrow \lambda_l + \gamma \frac{\partial L}{\partial \lambda_l}$ (**maximise** L)

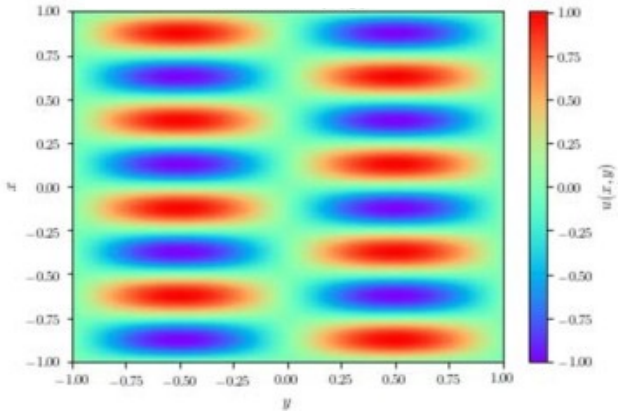
Note, for example, $\frac{\partial L}{\partial \lambda_i} = \left\| \mathcal{D}[NN(x_i; \theta)] - f(x_i) \right\|^2$

λ can be thought of as a soft attention layer

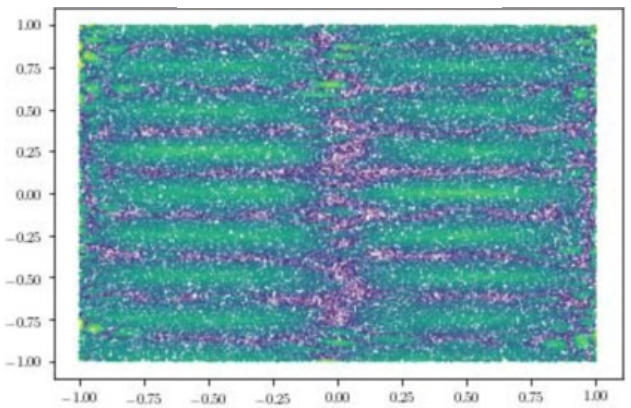
Helmholtz equation
Exact solution $u(x, y)$



PINN solution $NN(x, y; \theta)$

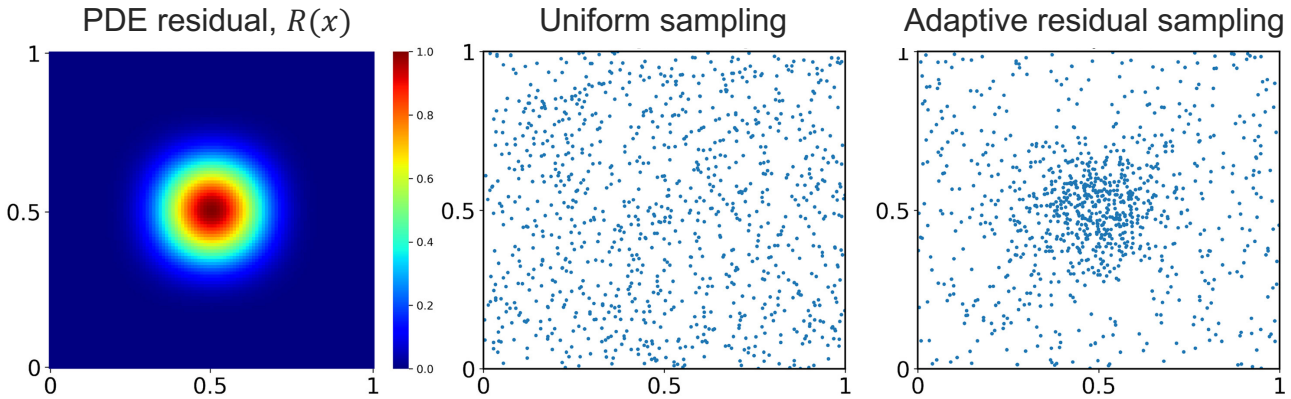


Final λ values
(brighter = larger)



McClenny et al, Self-adaptive physics-informed neural networks, JCP (2023)

Adaptive collocation points



Idea: sample collocation points according to the **probability distribution**

$$x \sim p(x) = \frac{R(x)}{A}$$

Where R is the PDE residual, $R(x) = \|\mathcal{D}[NN(x;\theta)] - f(x)\|^2$

And A is a normalising constant

Inspired by adaptive mesh refinement in FEM

Table 2. L^2 relative error of the PINN solution for the forward problems.

	Diffusion	Burgers'	Allen–Cahn	Wave
No. of residual points	30	2000	1000	2000
Grid	0.66±0.06%	13.7±2.37%	93.4±6.98%	81.3±13.7%
Random	0.74±0.17%	13.3±8.35%	22.2±16.9%	68.4±20.1%
LHS	0.48±0.24%	13.5±9.05%	26.6±15.8%	75.9±33.1%
Halton	0.24±0.17%	4.51±3.93%	0.29±0.14%	60.2±10.0%
Hammersley	0.17±0.07%	3.02±2.98%	**0.14±0.14%**	58.9±8.52%
Sobol	0.19±0.07%	3.38±3.21%	0.35±0.24%	57.5±14.7%
Random-R	**0.12±0.06%**	1.69±1.67%	0.55±0.34%	**0.72±0.90%**
RAR-G [3]	0.20±0.07%	**0.12±0.04%**	0.53±0.19%	0.81±0.11%
RAD	**0.11±0.07%**	**0.02±0.00%**	**0.08±0.06%**	**0.09±0.04%**
RAR-D	**0.14±0.11%**	**0.03±0.01%**	**0.09±0.03%**	**0.29±0.04%**

Wu et al, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering (2023)

Adaptive collocation points

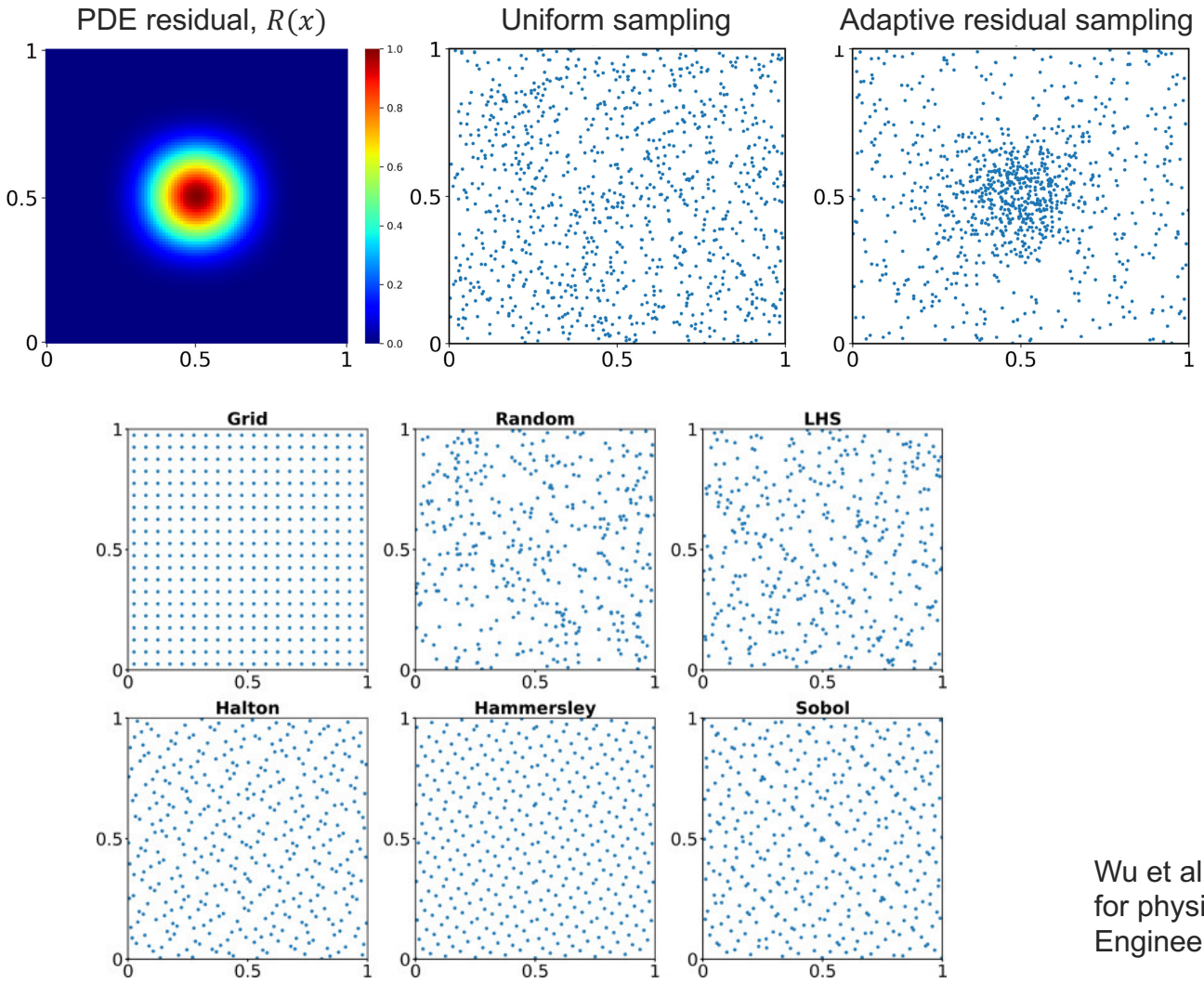


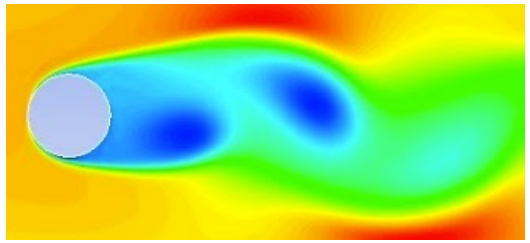
Table 2. L^2 relative error of the PINN solution for the forward problems.

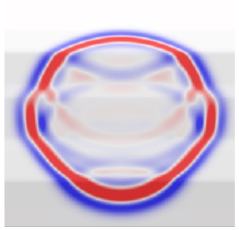
	Diffusion	Burgers'	Allen–Cahn	Wave
No. of residual points	30	2000	1000	2000
Grid	0.66±0.06%	13.7±2.37%	93.4±6.98%	81.3±13.7%
Random	0.74±0.17%	13.3±8.35%	22.2±16.9%	68.4±20.1%
LHS	0.48±0.24%	13.5±9.05%	26.6±15.8%	75.9±33.1%
Halton	0.24±0.17%	4.51±3.93%	0.29±0.14%	60.2±10.0%
Hammersley	0.17±0.07%	3.02±2.98%	**0.14±0.14**%	58.9±8.52%
Sobol	0.19±0.07%	3.38±3.21%	0.35±0.24%	57.5±14.7%
Random-R	**0.12±0.06**%	1.69±1.67%	0.55±0.34%	**0.72±0.90**%
RAR-G [3]	0.20±0.07%	**0.12±0.04**%	0.53±0.19%	0.81±0.11%
RAD	**0.11±0.07**%	**0.02±0.00**%	**0.08±0.06**%	**0.09±0.04**%
RAR-D	**0.14±0.11**%	**0.03±0.01**%	**0.09±0.03**%	**0.29±0.04**%

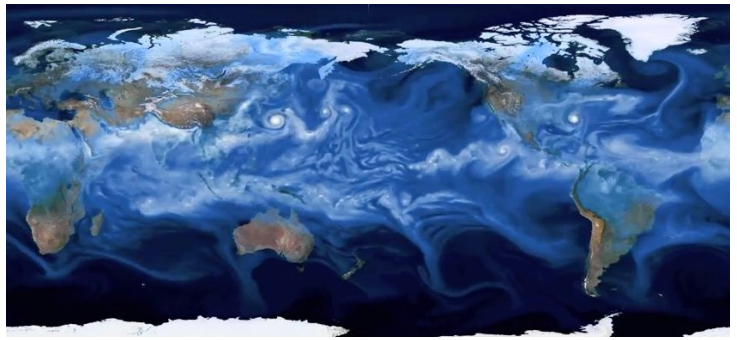
Wu et al, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering (2023)

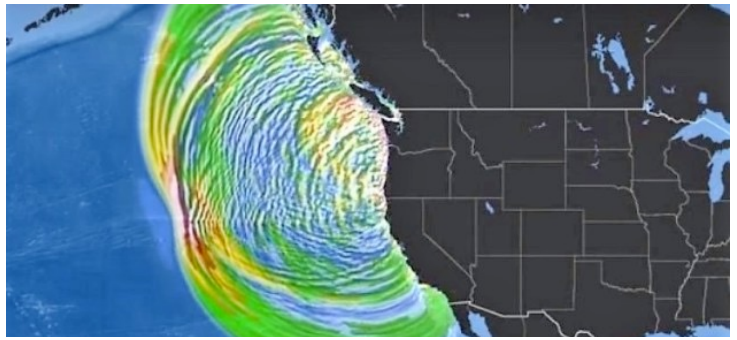
Limitation 3) – scaling to more complex problems

Scaling to more complex problems









It is often challenging to **scale** traditional scientific algorithms to:

- More complex phenomena (**multi-scale**, **multi-physics**)
- Large domains / higher frequency solutions
- Incorporate **real, noisy** and **sparse** data

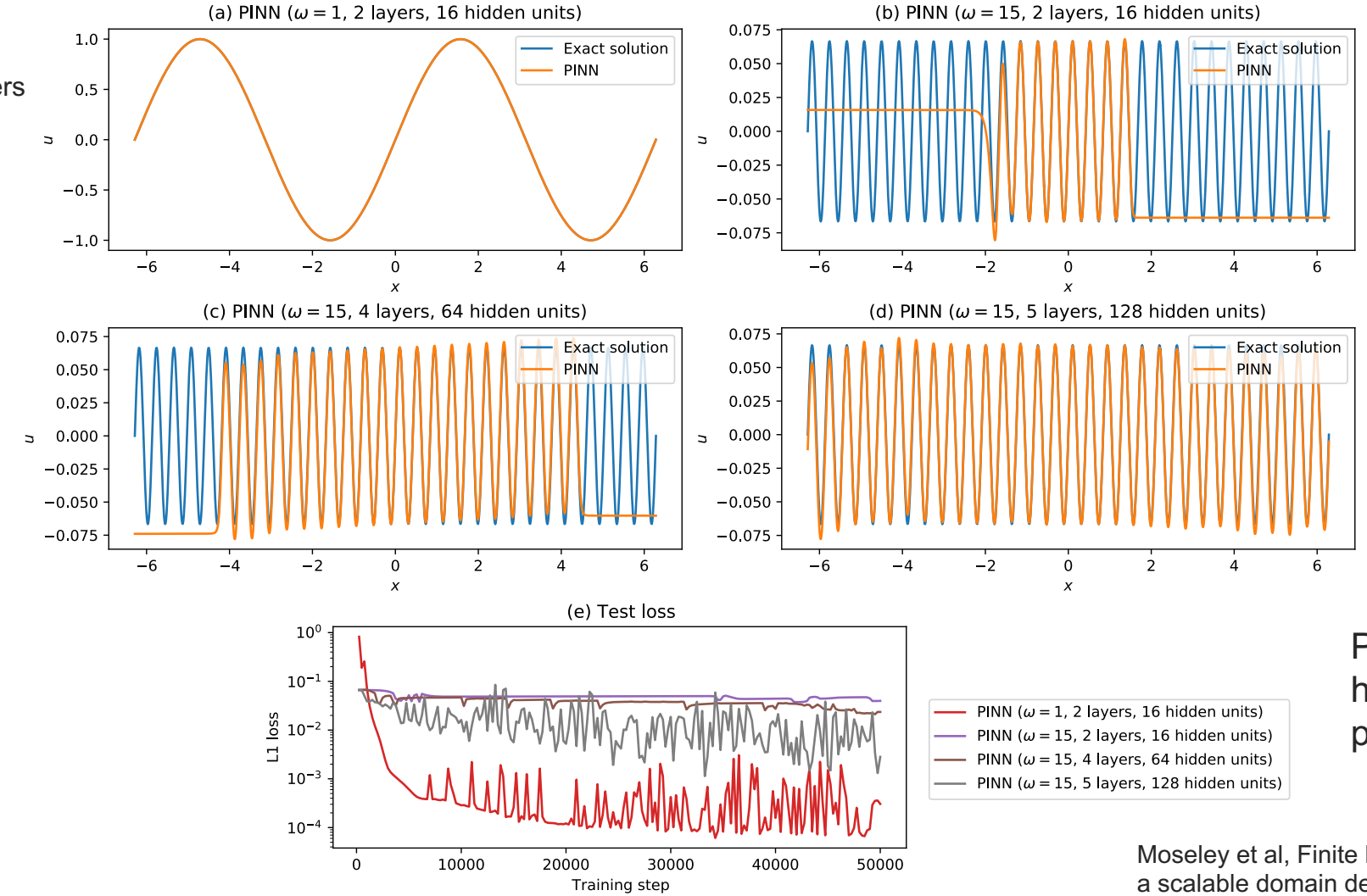
How do PINNs cope in this setting?

Most PINN research focuses on **toy/simplified** problems, as proof-of-principle studies

Image credits: Lawrence Berkeley National Laboratory / NOAA / NWS / Pacific Tsunami Warning Center

Scaling PINNs to higher frequencies

321 free parameters



PINN solving:

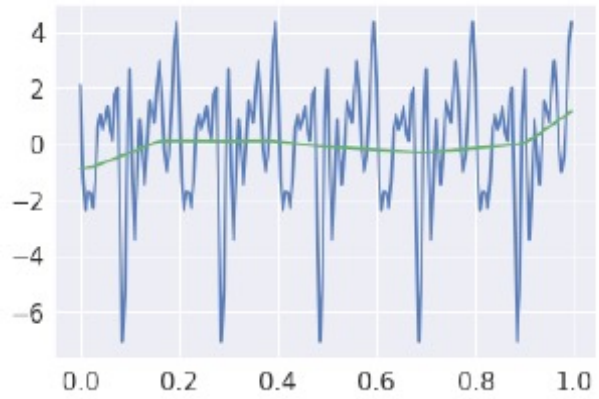
$$\frac{du}{dx} = \cos(\omega x)$$
$$u(0) = 0$$

66,433 free parameters

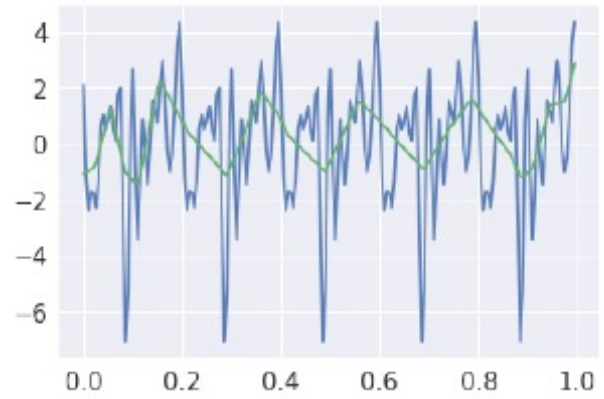
Problem: PINNs **struggle** to solve high-frequency / multiscale problems

Moseley et al, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, ACM (2023)

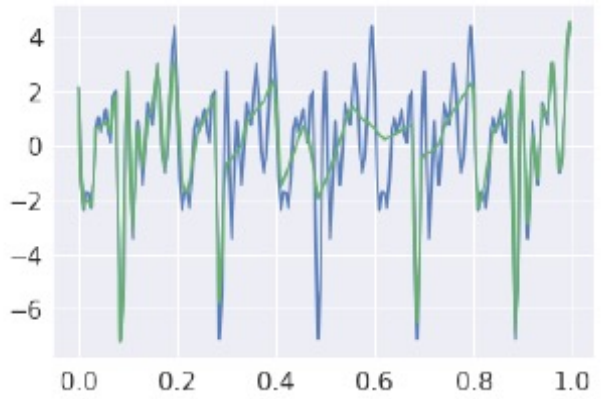
Spectral bias issue



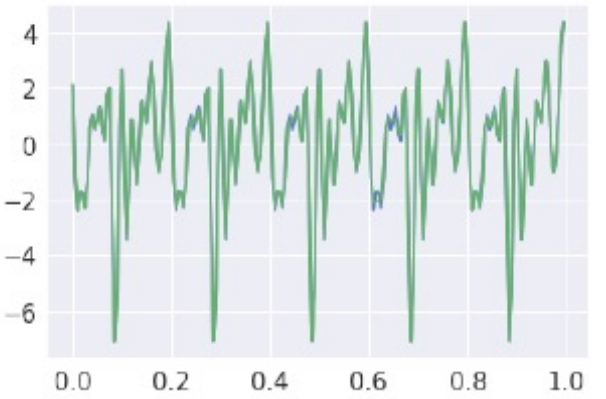
(a) Iteration 100



(b) Iteration 1000



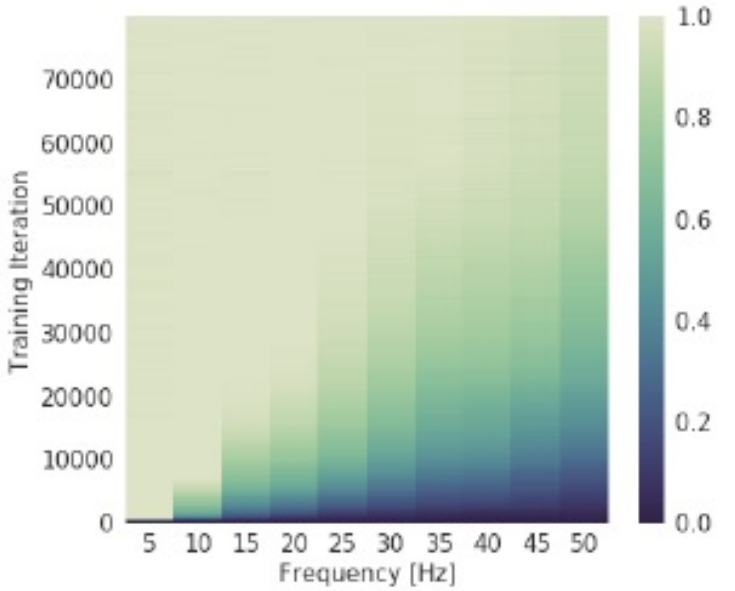
(c) Iteration 10000



(d) Iteration 80000

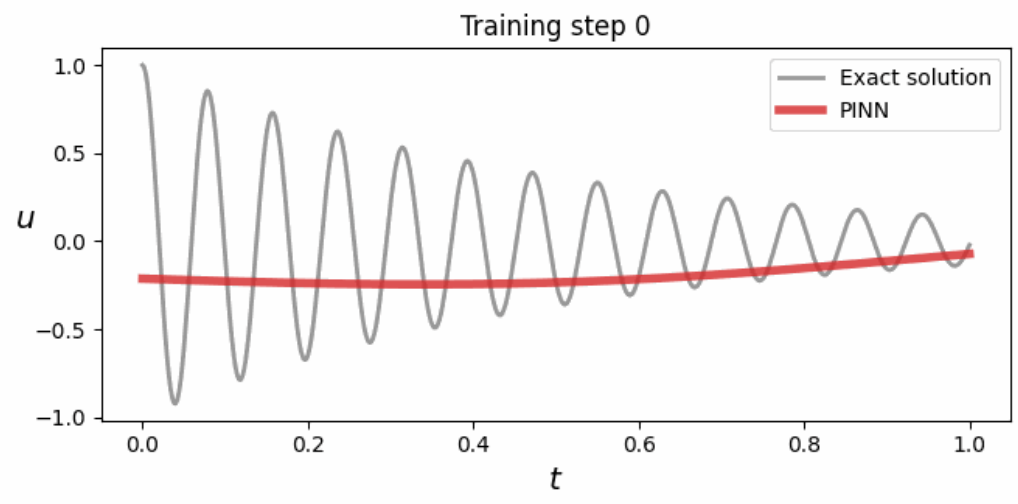
NNs prioritise learning **lower** frequency functions first

Under certain assumptions can be proved via neural tangent kernel theory

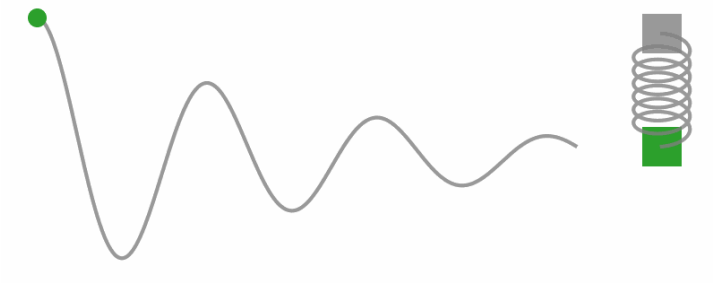


Rahaman, N., et al, On the spectral bias of neural networks. 36th International Conference on Machine Learning, ICML (2019)

Scaling PINNs to higher frequencies



Network size: 2 hidden layers, 64 hidden units



Damped harmonic oscillator

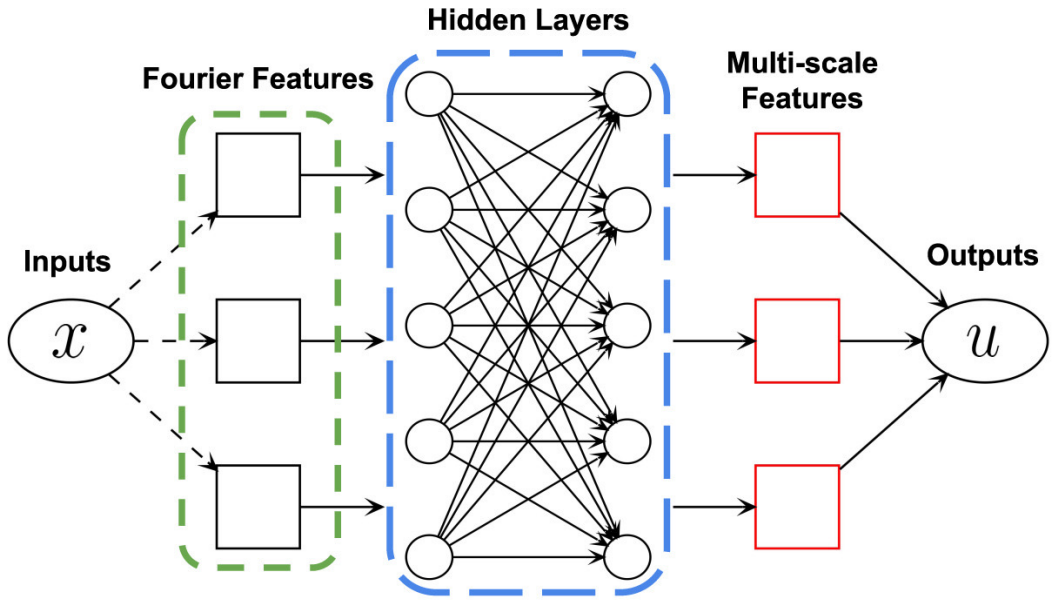
Problem: PINNs **struggle** to solve high-frequency / multiscale problems

As higher frequencies are added:

- More collocation points required
- Larger neural network required
- Spectral bias slows convergence

Leading to a significantly **harder** PINN optimization problem

Fourier features



Idea: **transform** input coordinates to higher-frequency functions

By using Fourier features:

$$\boldsymbol{\gamma}(\boldsymbol{x}) = [\cos(2\pi\Gamma\boldsymbol{x}), \sin(2\pi\Gamma\boldsymbol{x})]$$

Where Γ is a matrix of shape $(k \times d)$ where k is the number of Fourier features and d is the dimensionality of $\boldsymbol{x}$

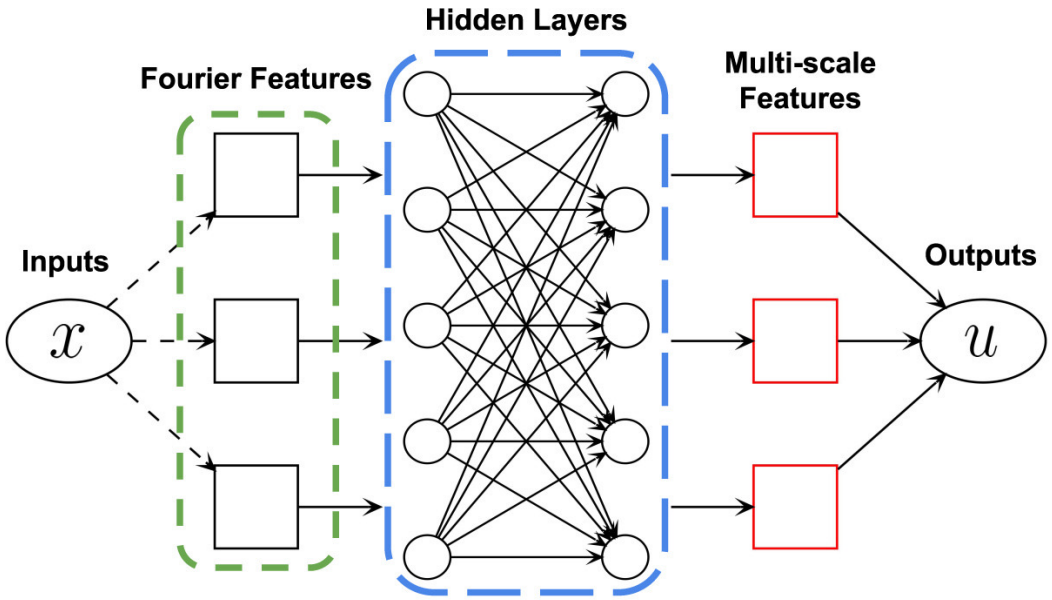
Typically, values of Γ are drawn from a normal distribution, i.e. $\Gamma_{ij} \sim \mathcal{N}(\cdot\,; \mu, \sigma)$, and fixed

μ, σ are hyperparameters which control the frequency of the Fourier features

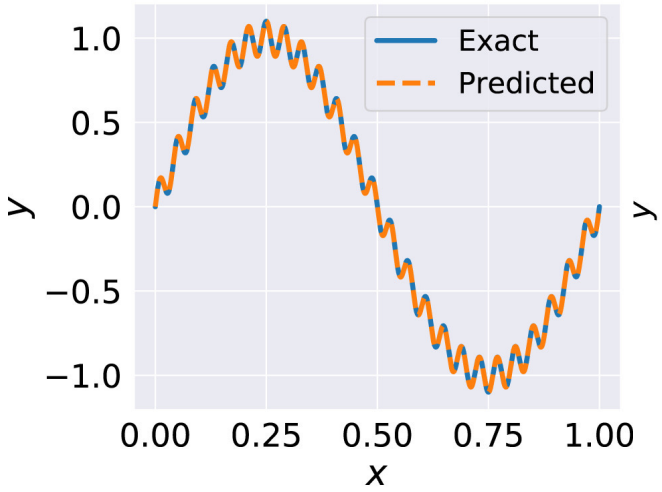
Tancik et al, Fourier features let networks learn high frequency functions in low dimensional domains, NeurIPS (2020)

Wang et al, On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering (2021)

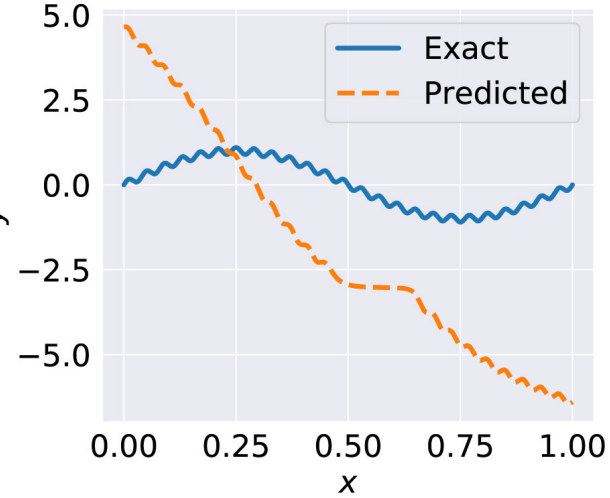
Fourier features



PINN with 2 sets of Fourier features



PINN



Solving 1D Poisson's equation

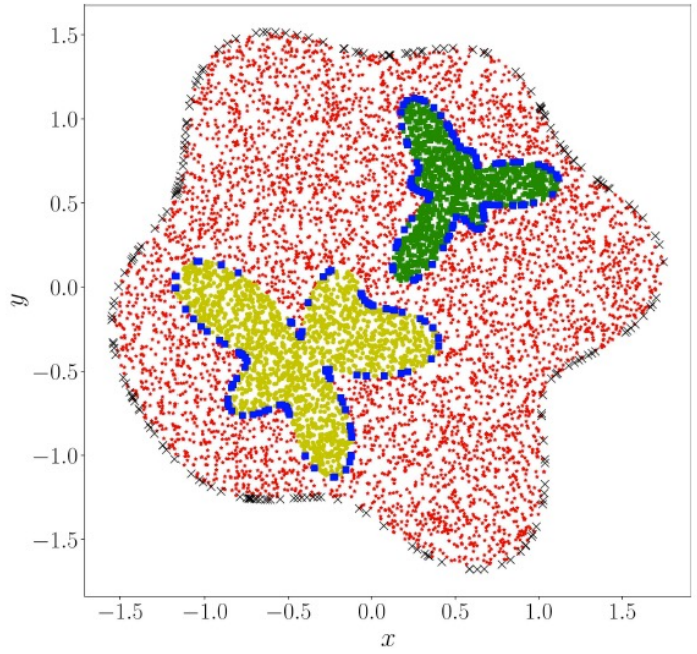
$\mu = 0$
$\sigma_1 = 1, \sigma_2 = 10$
$k = 25$

Fully connected network with 2 layers, 100 hidden units
Tanh activation function
Adam optimiser

Tancik et al, Fourier features let networks learn high frequency functions in low dimensional domains, NeurIPS (2020)

Wang et al, On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering (2021)

PINNs + domain decomposition



Jagtap, A., et al., Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Communications in Computational Physics (2020)

Idea:

Take a "**divide-and-conquer**" strategy to model more complex problems:

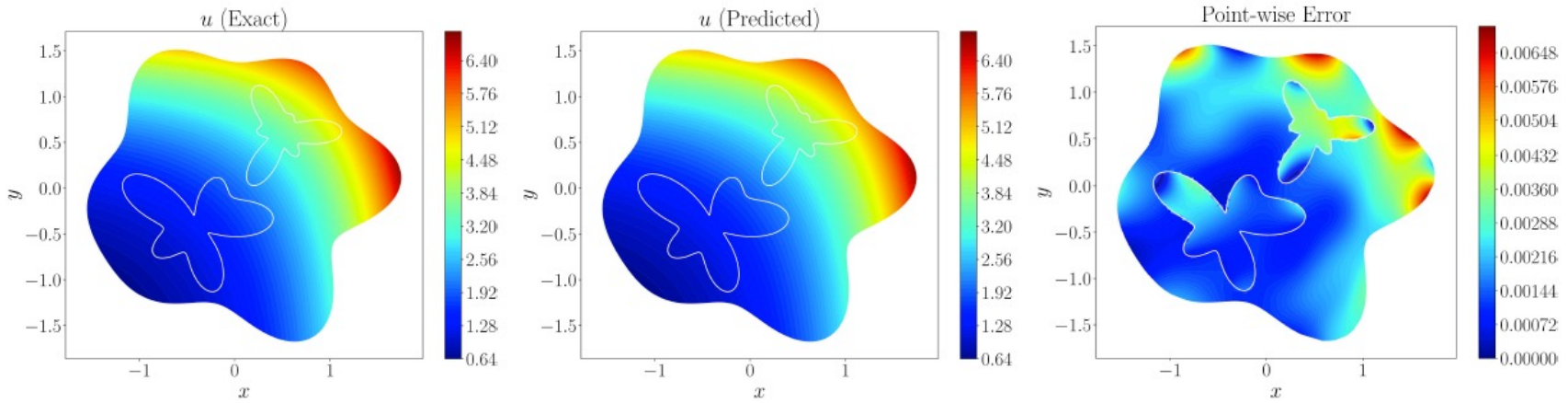
1. Divide modelling domain into many smaller **subdomains**
2. Use a separate neural network in each subdomain to model the solution

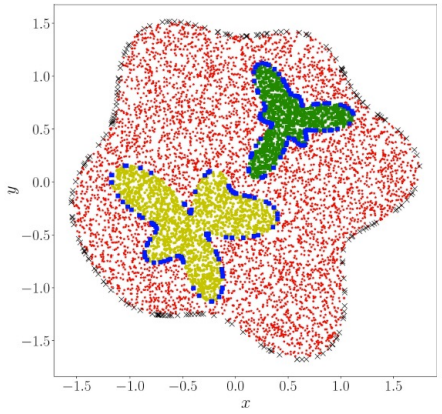
Hypothesis:

The resulting (coupled) local optimization problems are easier to solve than a single global problem

Extended PINNs (XPINNs)

XPINNs solving 2D Poisson's equation





$$L(\theta_m) = \frac{\lambda_b}{N_b}\sum_{i}^{N_b}(NN(x_i;\theta_m) - u_i)^2 \quad \text{Boundary loss}$$

$$+ \frac{\lambda_p}{N_p}\sum_{j}^{N_p}\left(R\left(NN(x_j;\theta_m)\right)\right)^2 \quad \text{Physics loss}$$

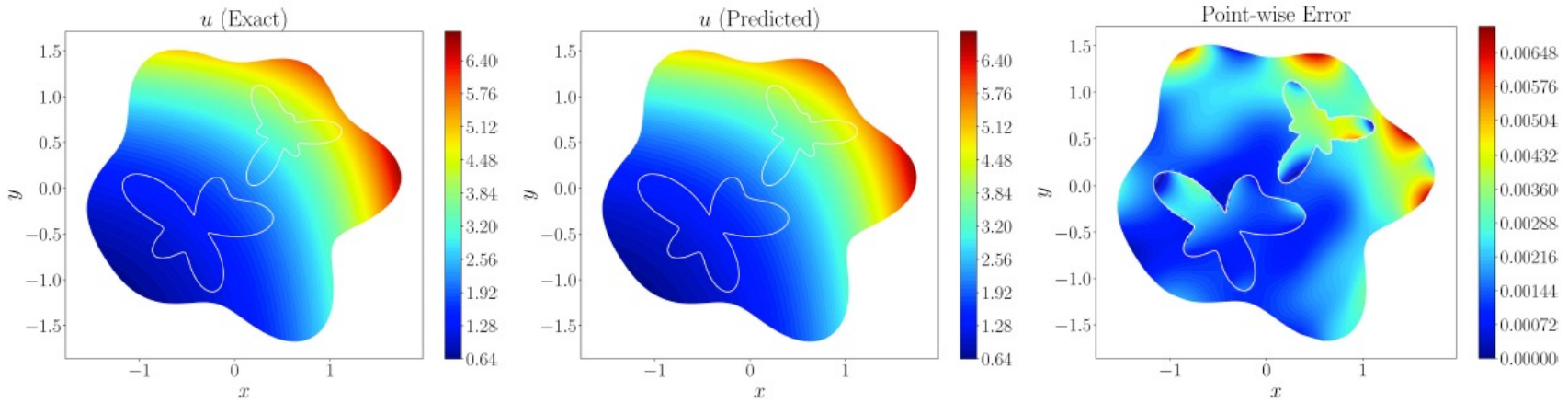
$$+ \sum_{l}^{N_n}\frac{\lambda_l}{N_l}\sum_{k}^{N_l}\left(NN(x_k;\theta_l) - NN(x_k;\theta_m)\right)^2 \quad \text{Interface conditions}$$

$l \in \text{Neighbours}(m)$

Jagtap, A., et al., Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Communications in Computational Physics (2020)

Extended PINNs (XPINNs)

XPINNs solving 2D Poisson's equation



Limitations:

- Introduces discontinuities in solution at subdomain interfaces
- Requires extra loss terms

$$L(\theta_m) = \frac{\lambda_b}{N_b}\sum_{i}^{N_b}(NN(x_i;\theta_m) - u_i)^2 \quad \text{Boundary loss}$$

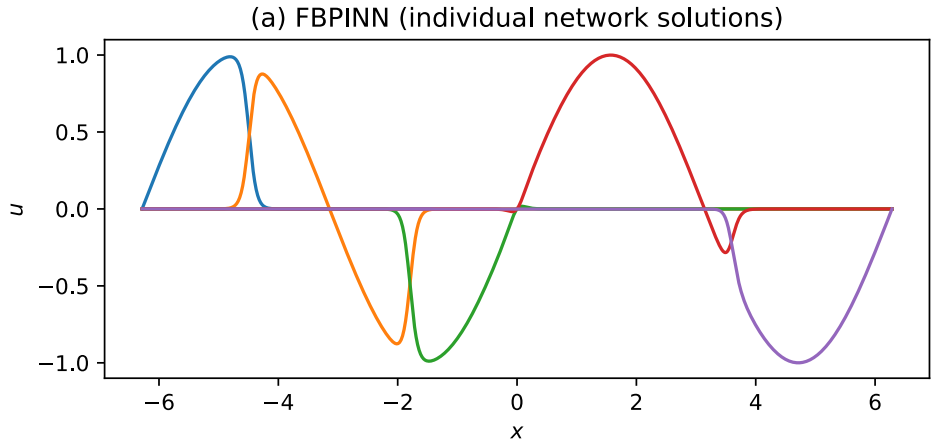
$$+ \frac{\lambda_p}{N_p}\sum_{j}^{N_p}\left(R\left(NN(x_j;\theta_m)\right)\right)^2 \quad \text{Physics loss}$$

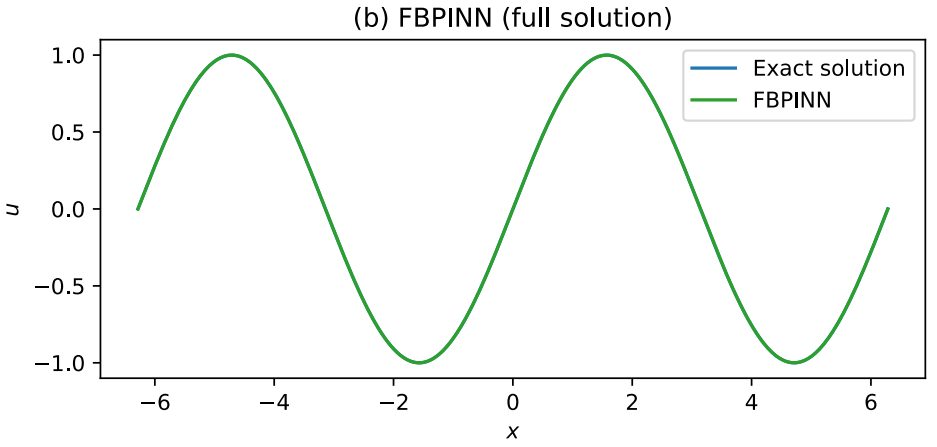
$$+ \sum_{l}^{N_n}\frac{\lambda_l}{N_l}\sum_{k}^{N_l}\left(NN(x_k;\theta_l) - NN(x_k;\theta_m)\right)^2 \quad \text{Interface conditions}$$

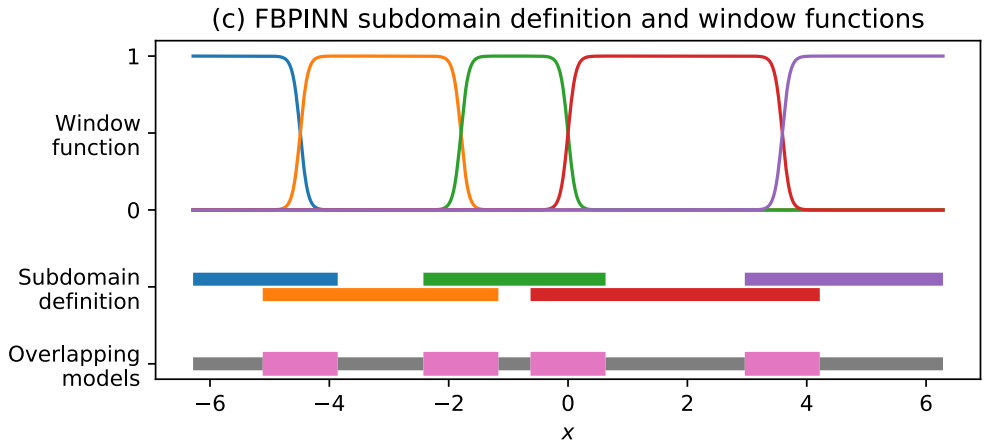
$l \in \text{Neighbours}(m)$

Jagtap, A., et al., Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Communications in Computational Physics (2020)

Finite basis PINNs (FBPINNs)







$$\hat{u}(x;\theta) = \mathcal{C}\left[\sum_i^n w_i(x) \cdot \text{unnorm} \circ NN_i \circ \text{norm}_i(x)\right]$$

Window function

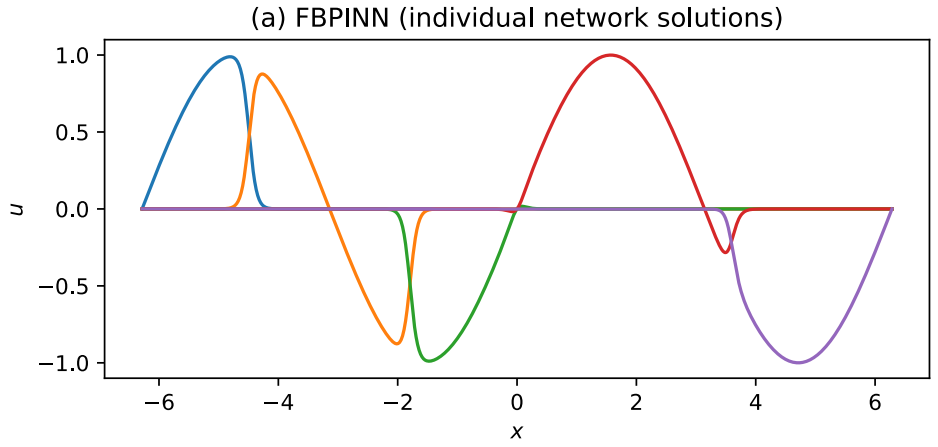
Subdomain network

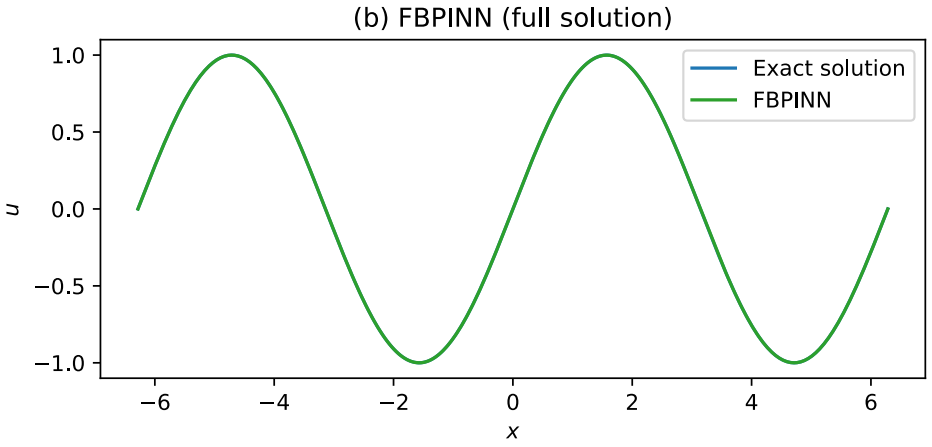
Individual subdomain normalisation

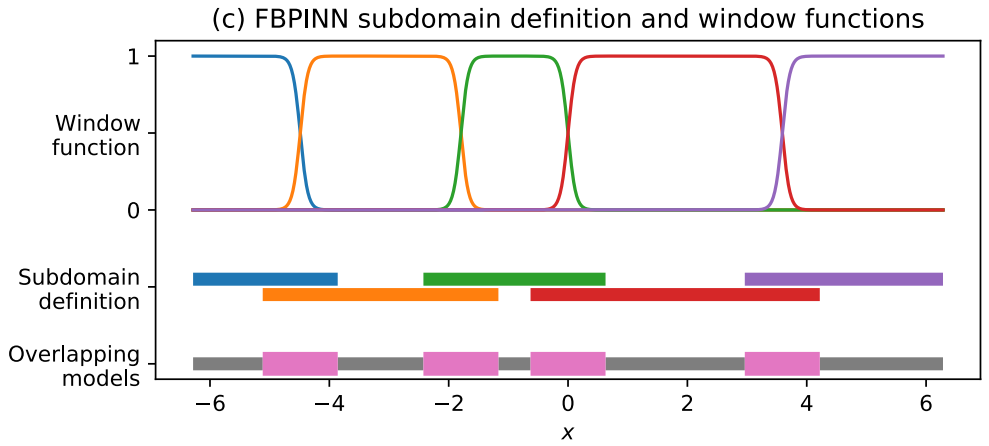
Moseley et al, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, ACM (2023)

Idea: use **overlapping** subdomains and a **globally** defined solution ansatz

Finite basis PINNs (FBPINNs)





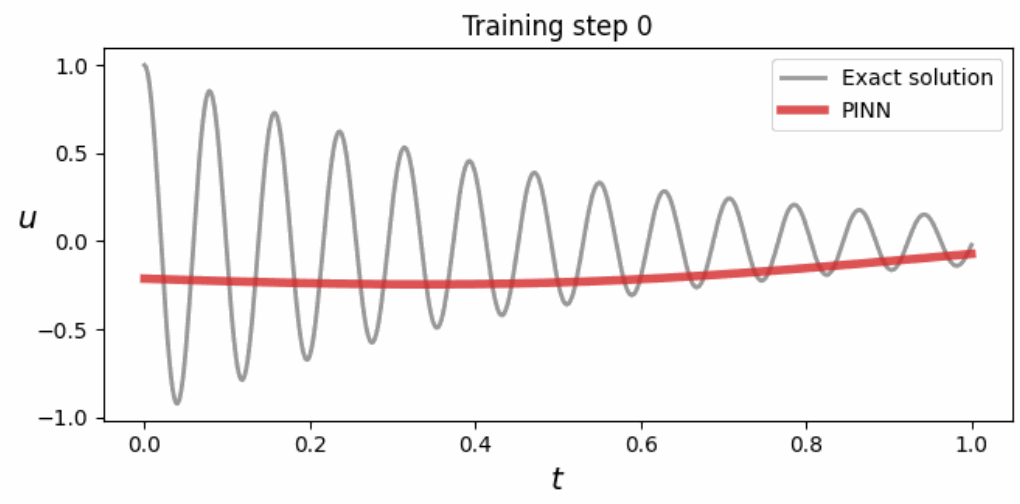


Note:

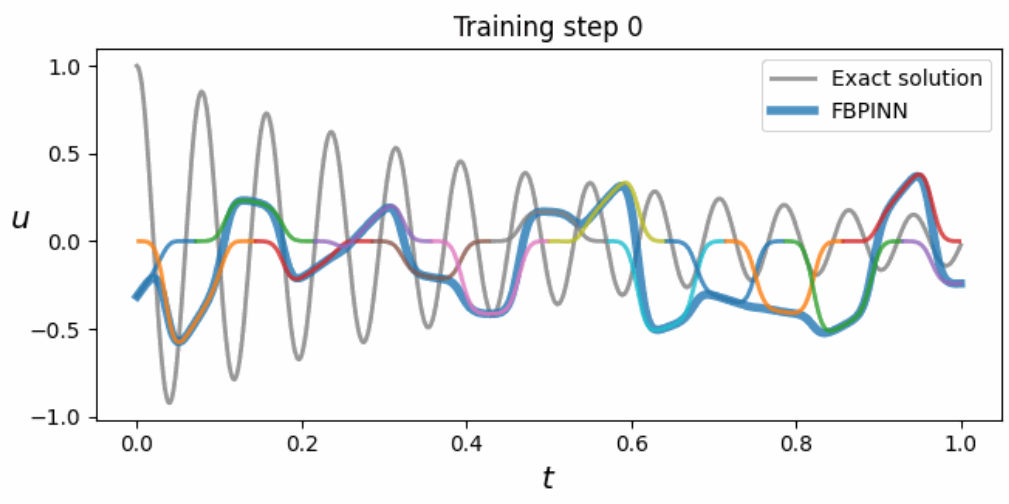
- By construction, FBPINN solution is continuous across subdomain interfaces
- Can be trained with same loss function as PINNs
- FBPINNs can simply be thought of as a custom NN architecture for PINNs

Moseley et al, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, ACM (2023)

FBPINNs vs PINNs

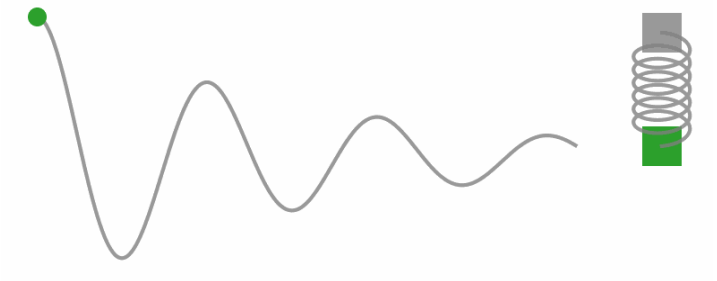


PINN solution



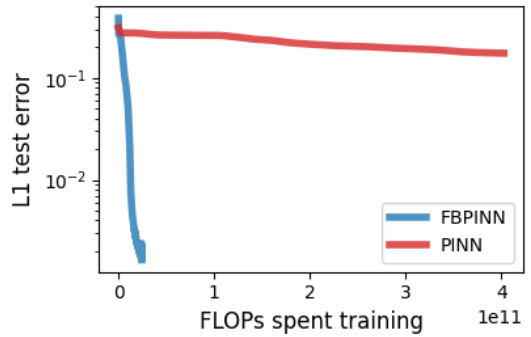
FBPINN solution

Network size: 2 hidden layers, 64 hidden units

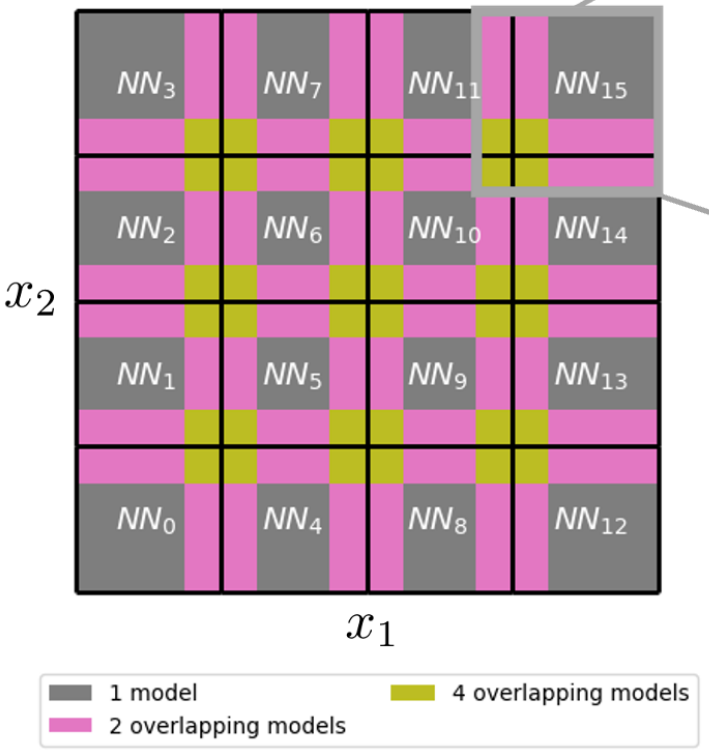


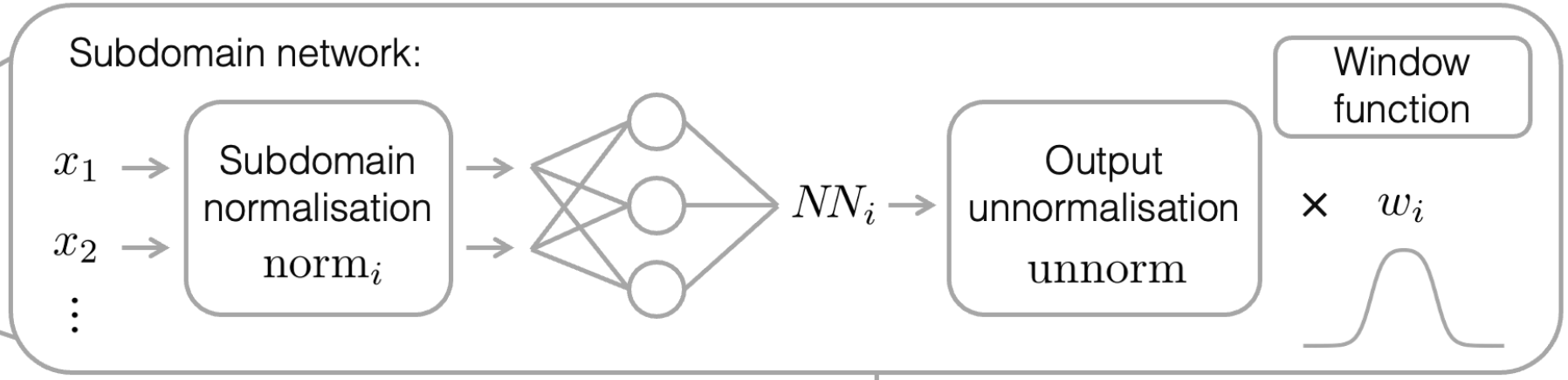
Damped harmonic oscillator

Number of subdomains: 15
Subdomain network size: 1 hidden layer, 32 hidden units



Extension to multiple dimensions





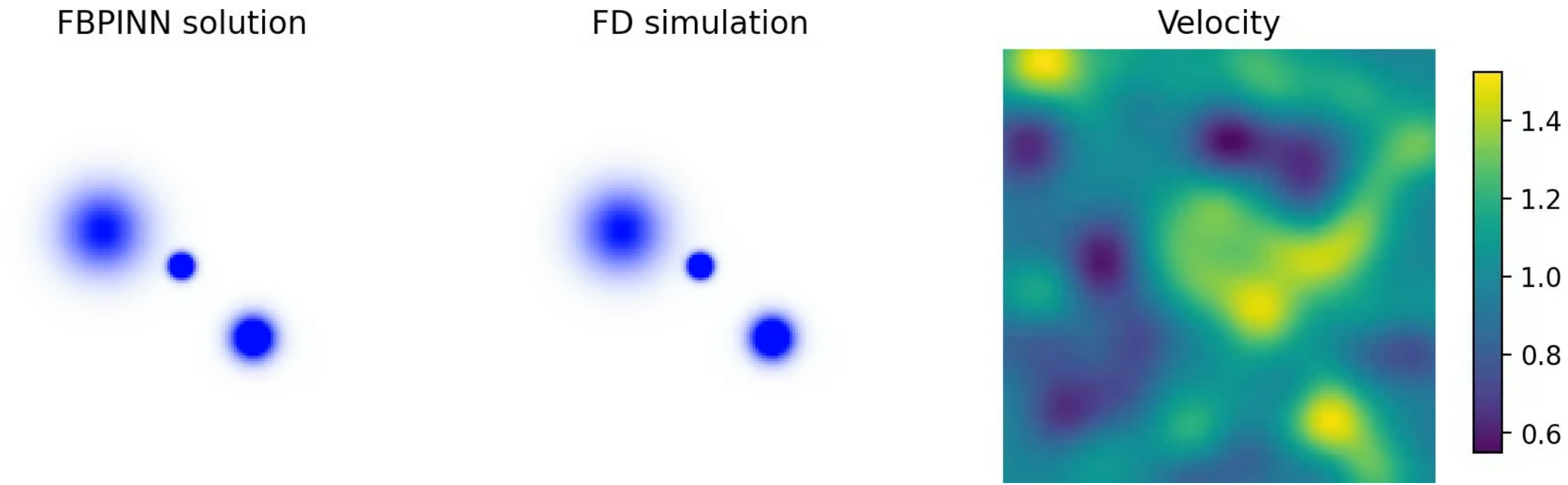
FBPINN solution:

$$\hat{u}(x;\theta) = \mathcal{C}\left[\sum_i^n w_i(x) \cdot \text{unnorm} \circ NN_i \circ \text{norm}_i(x)\right]$$

Summation over all subdomain networks

- In general, any overlapping domain decomposition could be used

Solving the wave equation with FBPINNs

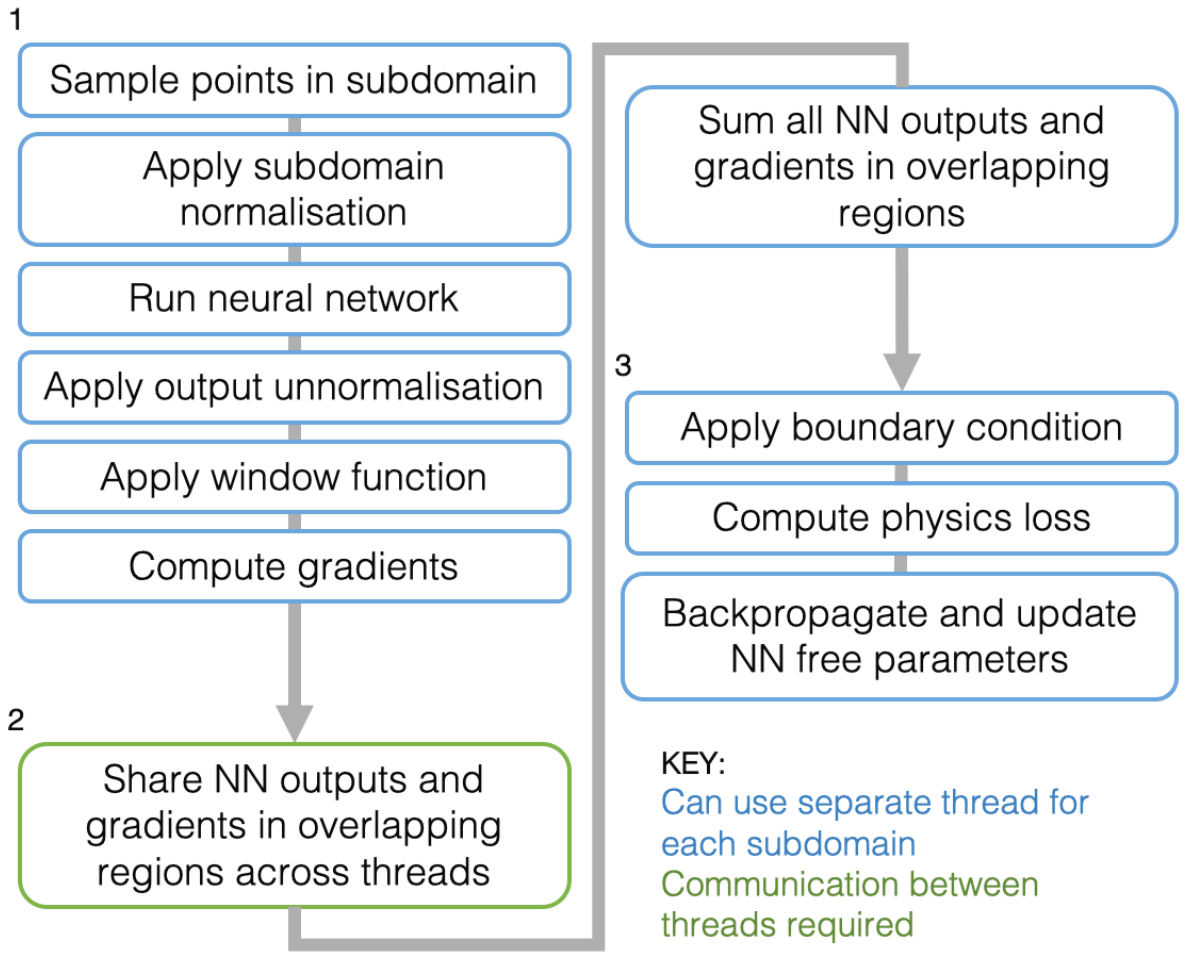


Solving the 2+1D acoustic wave equation:

$$\nabla^2 u(x,t) - \frac{1}{c(x)^2}\frac{\partial^2 u(x,t)}{\partial t^2} = 0$$

Number of subdomains: 30 x 30 x 30 = 27,000
Subdomain network size: 1 hidden layer, 8 hidden units
Total number of free parameters: 1.1 M
Number of collocation points: 150 x 150 x 150 = 3.4 M
Optimiser: Time-stepping scheduling, Adam 0.001 lr
Training time: ~2 hr

Parallel computation



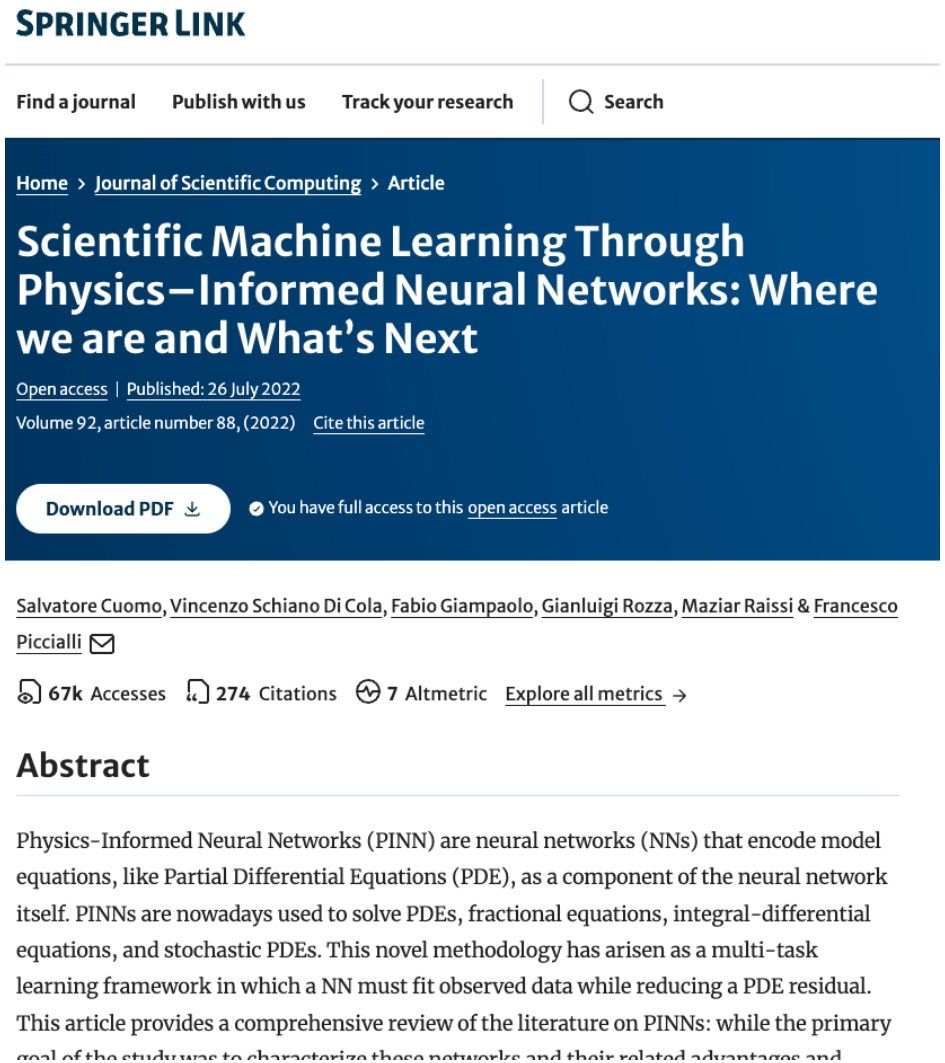
Two key considerations:

- Only points within each subdomain are required to train each subdomain network
- Communication between subdomains only required when summing solution (and its gradients) in overlap region

i.e. FBPINNs are highly parallelisable

Other PINN extensions

- Different differential equations:
  - Fractional equations, stochastic PDEs, variational formulations (VPINNs), mixed formulations, …
- Uncertainty estimation:
  - Bayesian PINNs (B-PINNs), GAN PINNs, …
- Architectures:
  - Adaptive activation functions, extreme learning machines, SIRENs, wavelet layers, LSTMs, …
- Other optimisation strategies:
  - NTK weighting, casual training, specialised optimisers
- PINN software libraries:
  - NVIDIA Modulus, DeepXDE, NeuroDiffEq, PyDENs, …
- And much more!



SPRINGER LINK

Find a journal | Publish with us | Track your research | Search

Home > Journal of Scientific Computing > Article

Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next

Open access | Published: 26 July 2022

Volume 92, article number 88, (2022) Cite this article

Download PDF | You have full access to this open access article

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi & Francesco Piccialli

67k Accesses | 274 Citations | 7 Altmetric | Explore all metrics →

Abstract

Physics-Informed Neural Networks (PINN) are neural networks (NNs) that encode model equations, like Partial Differential Equations (PDE), as a component of the neural network itself. PINNs are nowadays used to solve PDEs, fractional equations, integral-differential equations, and stochastic PDEs. This novel methodology has arisen as a multi-task learning framework in which a NN must fit observed data while reducing a PDE residual. This article provides a comprehensive review of the literature on PINNs: while the primary goal of the study was to characterize these networks and their related advantages and

Bayesian PINNs (B-PINNs)

$$\mathcal{N}_{\boldsymbol{x}}(u;\boldsymbol{\lambda}) = f, \quad \boldsymbol{x} \in D,$$
$$\mathcal{B}_{\boldsymbol{x}}(u;\boldsymbol{\lambda}) = b, \quad \boldsymbol{x} \in \Gamma,$$

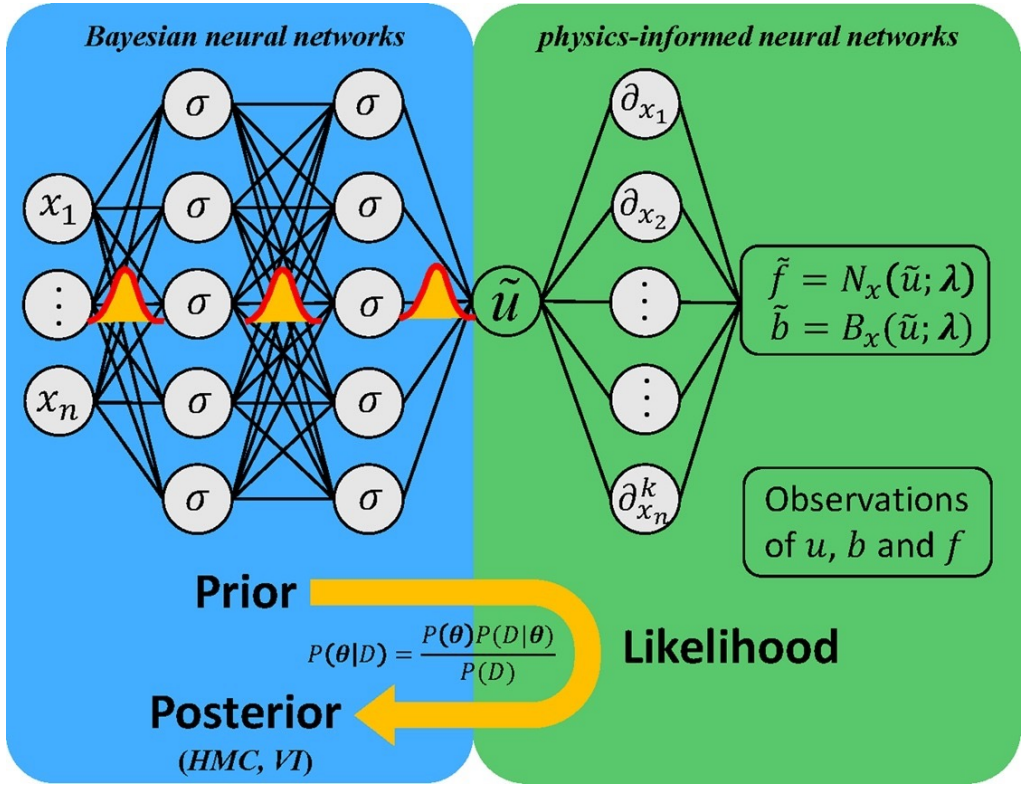
$$P(\mathscr{D}|\boldsymbol{\theta}) = P(\mathscr{D}_u|\boldsymbol{\theta})P(\mathscr{D}_f|\boldsymbol{\theta})P(\mathscr{D}_b|\boldsymbol{\theta}),$$

$$P(\mathscr{D}_u|\boldsymbol{\theta}) = \prod_{i=1}^{N_u} \frac{1}{\sqrt{2\pi\sigma_u^{(i)^2}}} \exp\left(-\frac{(\tilde{u}(\boldsymbol{x}_u^{(i)};\boldsymbol{\theta}) - \bar{u}^{(i)})^2}{2\sigma_u^{(i)^2}}\right),$$

$$P(\mathscr{D}_f|\boldsymbol{\theta}) = \prod_{i=1}^{N_f} \frac{1}{\sqrt{2\pi\sigma_f^{(i)^2}}} \exp\left(-\frac{(\tilde{f}(\boldsymbol{x}_f^{(i)};\boldsymbol{\theta}) - \bar{f}^{(i)})^2}{2\sigma_f^{(i)^2}}\right),$$

$$P(\mathscr{D}_b|\boldsymbol{\theta}) = \prod_{i=1}^{N_b} \frac{1}{\sqrt{2\pi\sigma_b^{(i)^2}}} \exp\left(-\frac{(\tilde{b}(\boldsymbol{x}_b^{(i)};\boldsymbol{\theta}) - \bar{b}^{(i)})^2}{2\sigma_b^{(i)^2}}\right).$$

$$P(\boldsymbol{\theta}|\mathscr{D}) = \frac{P(\mathscr{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathscr{D})} \simeq P(\mathscr{D}|\boldsymbol{\theta})P(\boldsymbol{\theta}),$$



Task:

Given i.i.d. noisy observations of u, b, and f

Estimate the posterior distribution of the network's parameters

Yang et al, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. Journal of Computational Physics (2021)

Bayesian PINNs (B-PINNs)

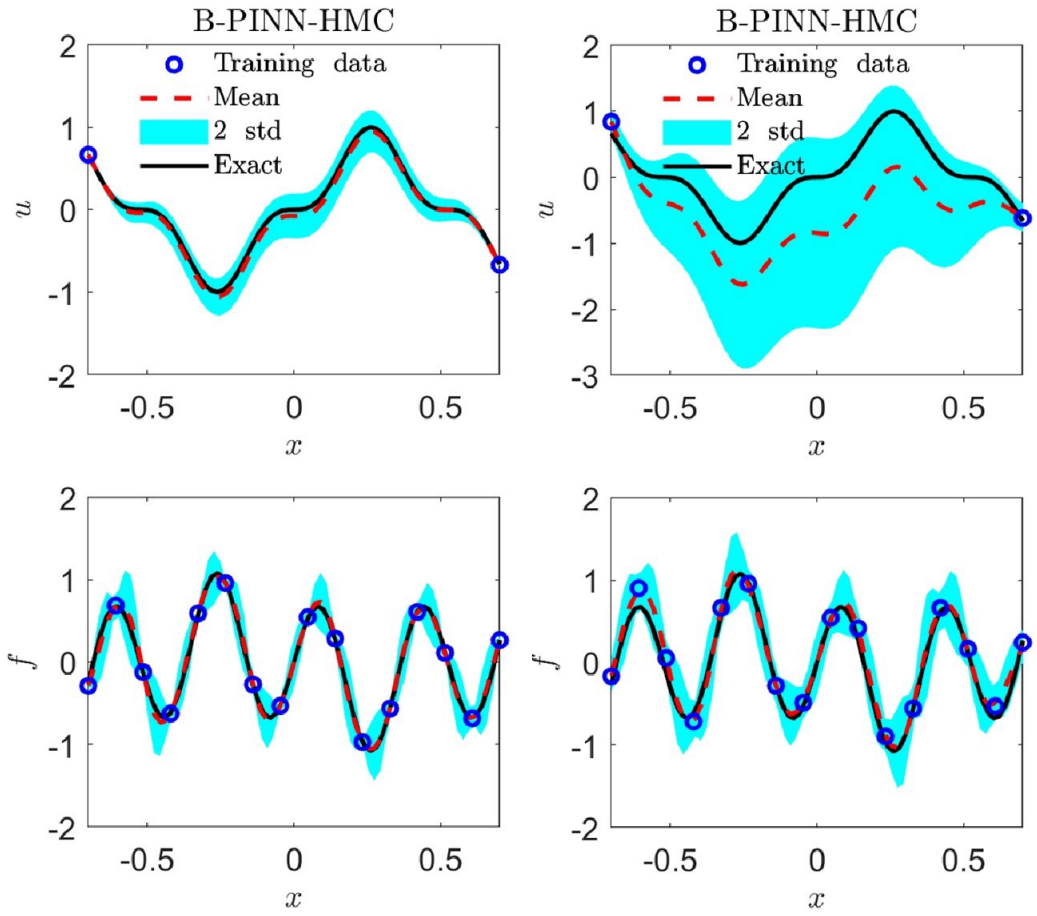
B-PINN solving 1D linear Poisson equation

$$\lambda \frac{\partial^2 u}{\partial x^2} = f$$

$$D = \{x_i, \bar{b}_i\}_i^{N_b} \cup \{x_i, \bar{f}_i\}_i^{N_f}$$

Sample θ from posterior $p(\theta|D)$ using Hamiltonian Monte Carlo (MCMC method)

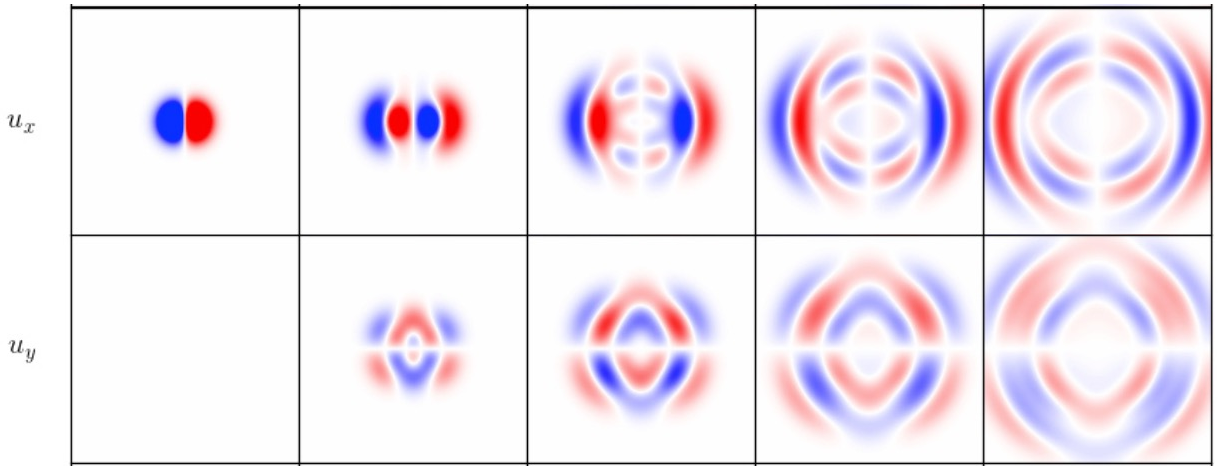
Plot mean and standard deviation of u and f predicted by PINN using these samples of θ



$\sigma_f, \sigma_b = 0.01$ $\sigma_f, \sigma_b = 0.1$

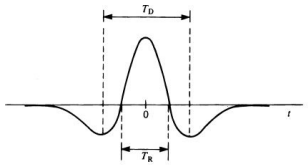
Yang et al, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. Journal of Computational Physics (2021)

Custom neural network architectures for PINNs



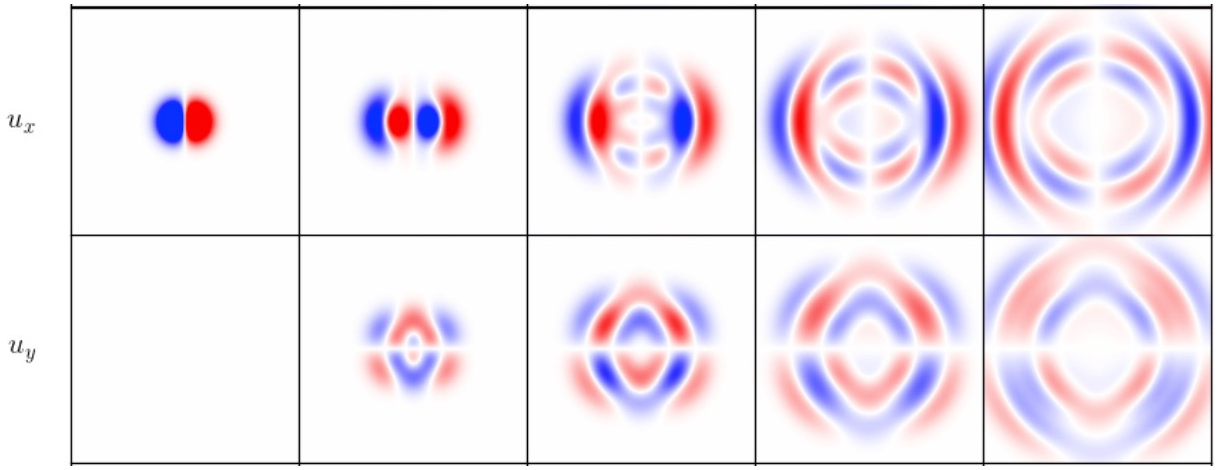
Elastic wave equation simulation

- Solution could be approximated as the sum of many wavelets



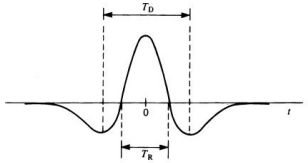
Staub, D., Solving the Elastic Wave Equation with Physics-Informed Neural Networks: A Robust and Critical Assessment (2024)

Custom neural network architectures for PINNs

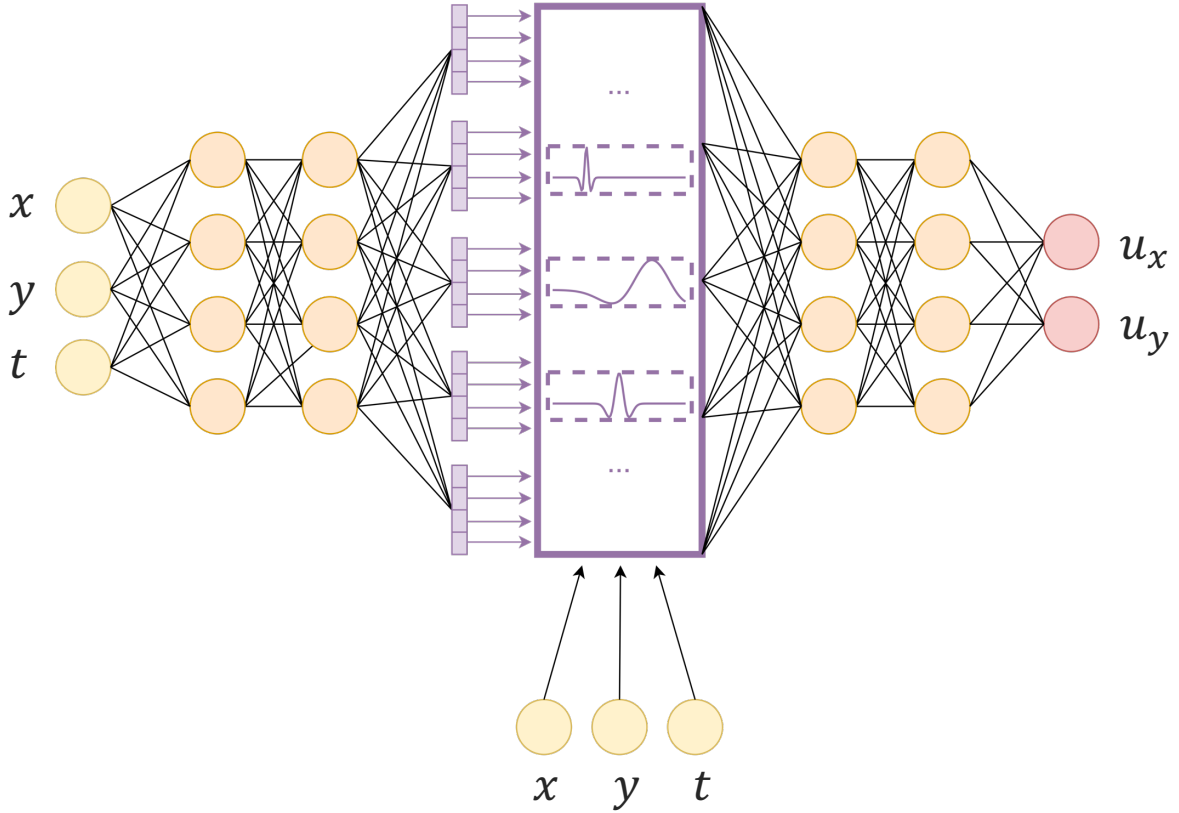


Elastic wave equation simulation

- Solution could be approximated as the sum of many wavelets



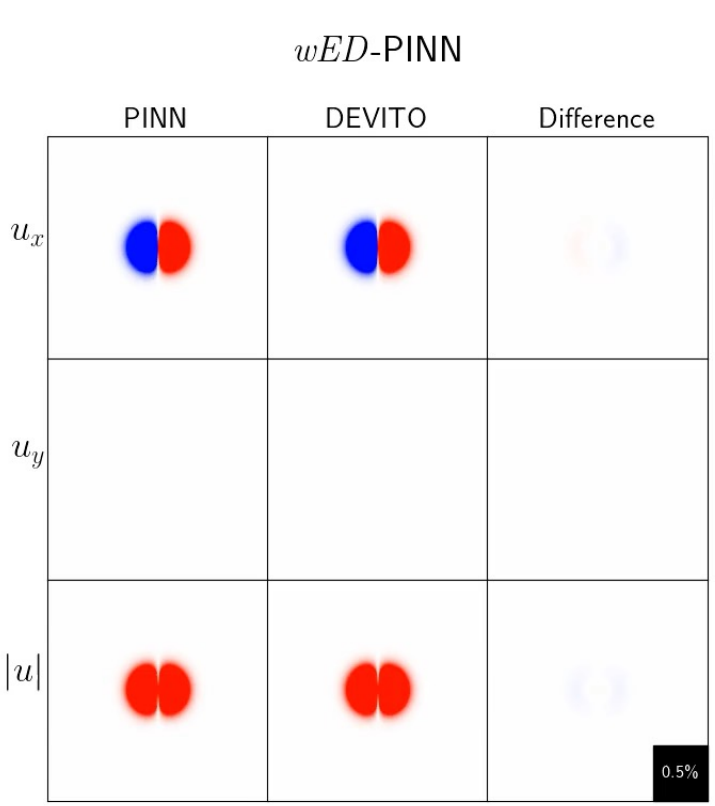
Staub, D., Solving the Elastic Wave Equation with Physics-Informed Neural Networks: A Robust and Critical Assessment (2024)



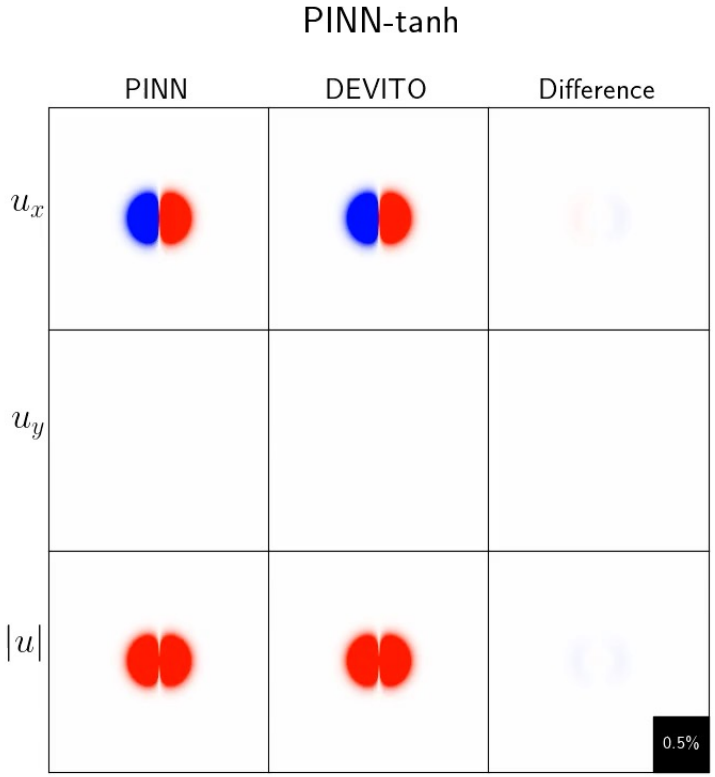
Encoder-decoder architecture:

- Encoder predicts parameters of wavelets (frequency, shift, scale)
- Decoder learns non-linear combination of wavelets

Custom neural network architectures for PINNs



4 encoder layers, 32 hidden units
128 wavelets
3 decoder layers, 64 hidden units



5 hidden layers, 128 hidden units

Staub, D., Solving the Elastic Wave Equation with Physics-Informed Neural Networks: A Robust and Critical Assessment (2024)

Lecture summary

- Standard PINNs suffer from some major limitations:
  - Computational cost
  - Poor convergence
  - Scaling to more complex problems
- There exist many (100-1000s) of PINN extensions and applications which improve these (only some of which were covered above)
- PINNs are an active field of research!