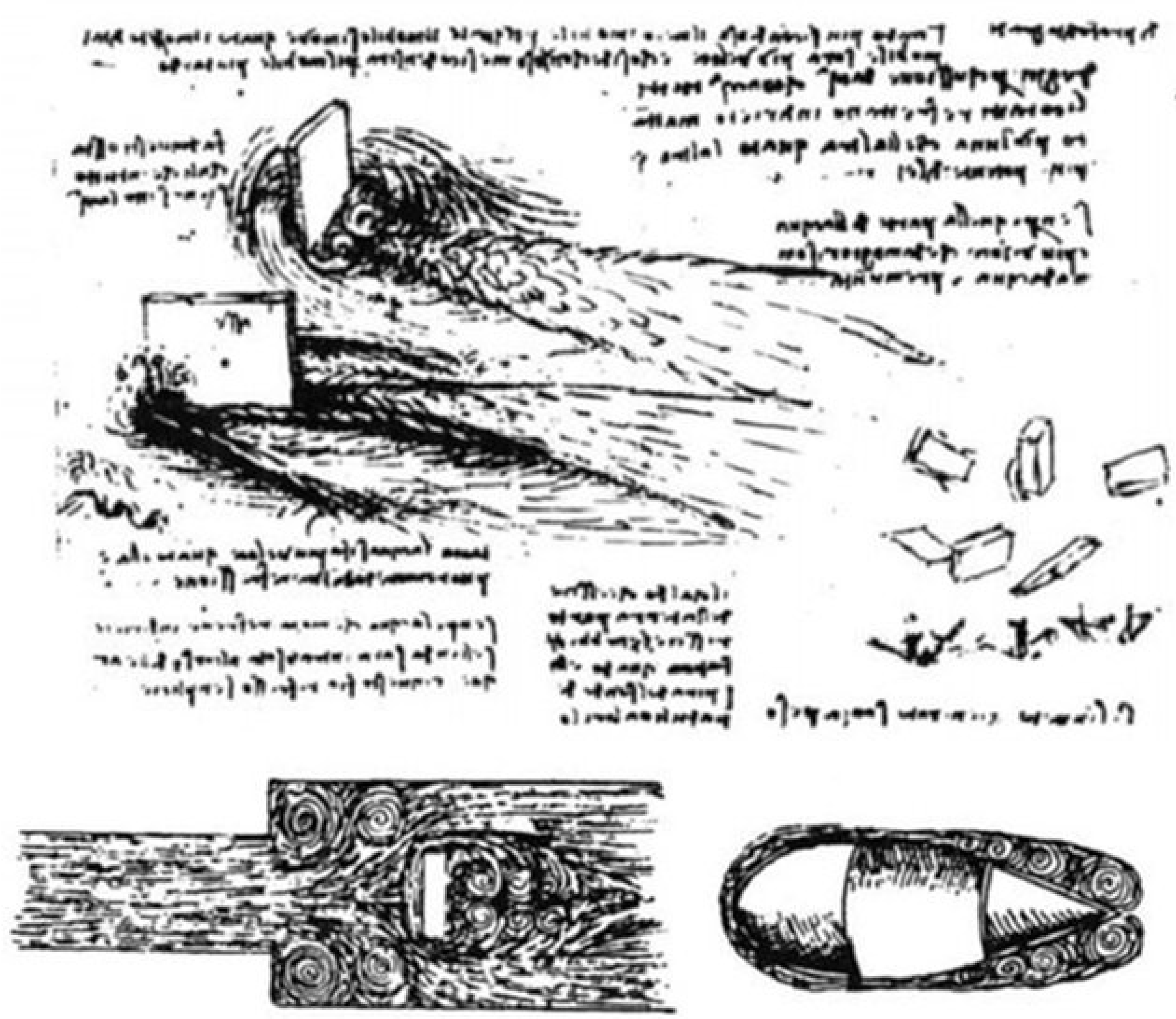
Physics-Informed Learning Machines

Observe the motion of the surface of the water, which resembles that of hair……..



George Em Karniadakis

Division of Applied Mathematics, Brown University

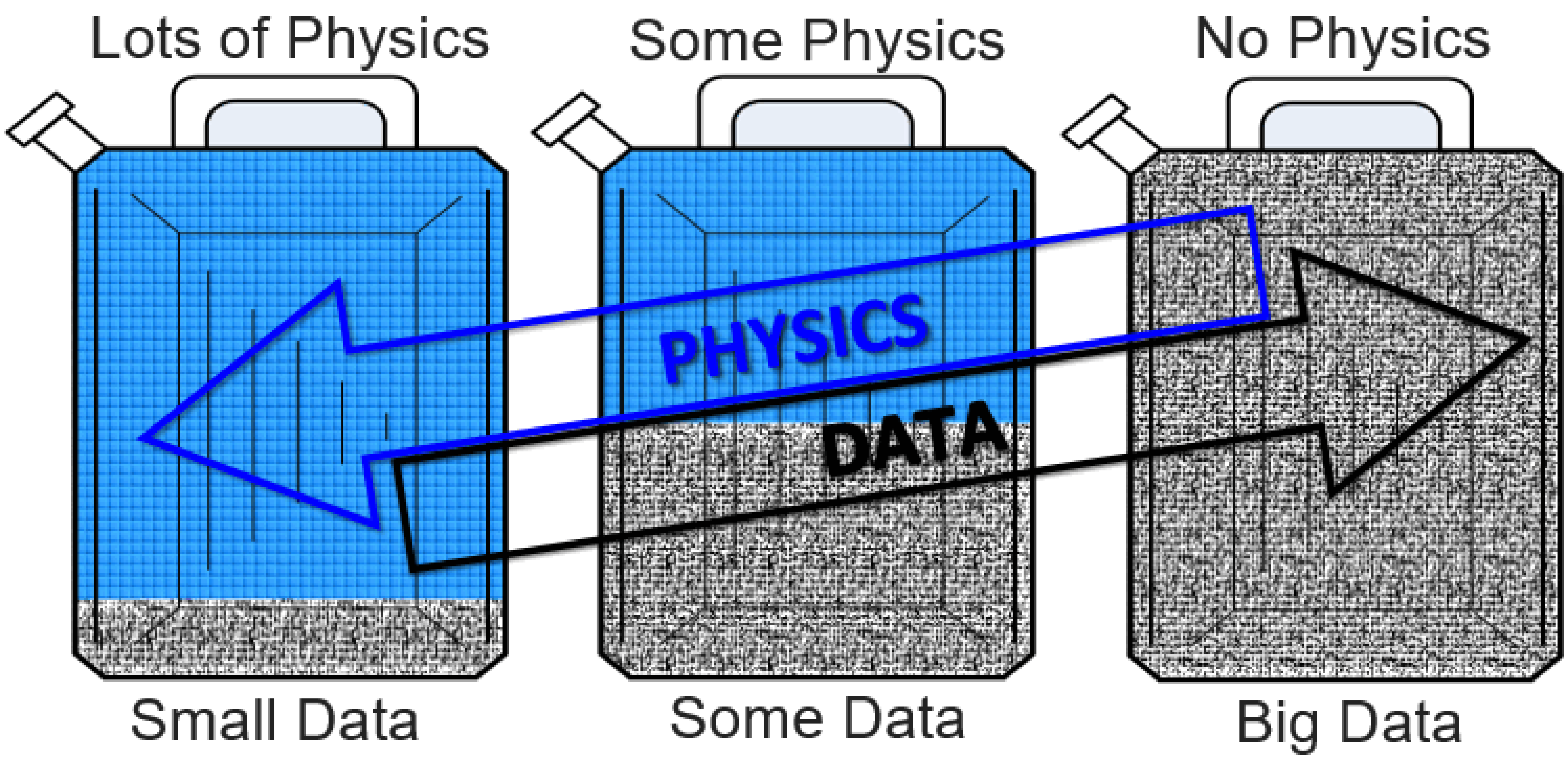
The **CRUNCH** group: **Home of "Math + Machine Learning + X"**

https://www.brown.edu/research/projects/crunch/home

Data + Physical Laws

* Dinky, Dirty, Dynamic, Deceptive Data

Three scenarios of
Physics-Informed Learning Machines



Solving Differential Equations from Measurements Only!

"…once we allow that we don't know f(x), but do know some things, it becomes natural to take a Bayesian approach"

Persi Diaconis, Stanford (1988)

- ✓ **Remove the tyranny of Grids! And of serious Math!**
- ✓ **Use noisy measurements - Predict with uncertainty!**
- ✓ **Execute Poincare's will!**

Nonlinear regression with Gaussian processes

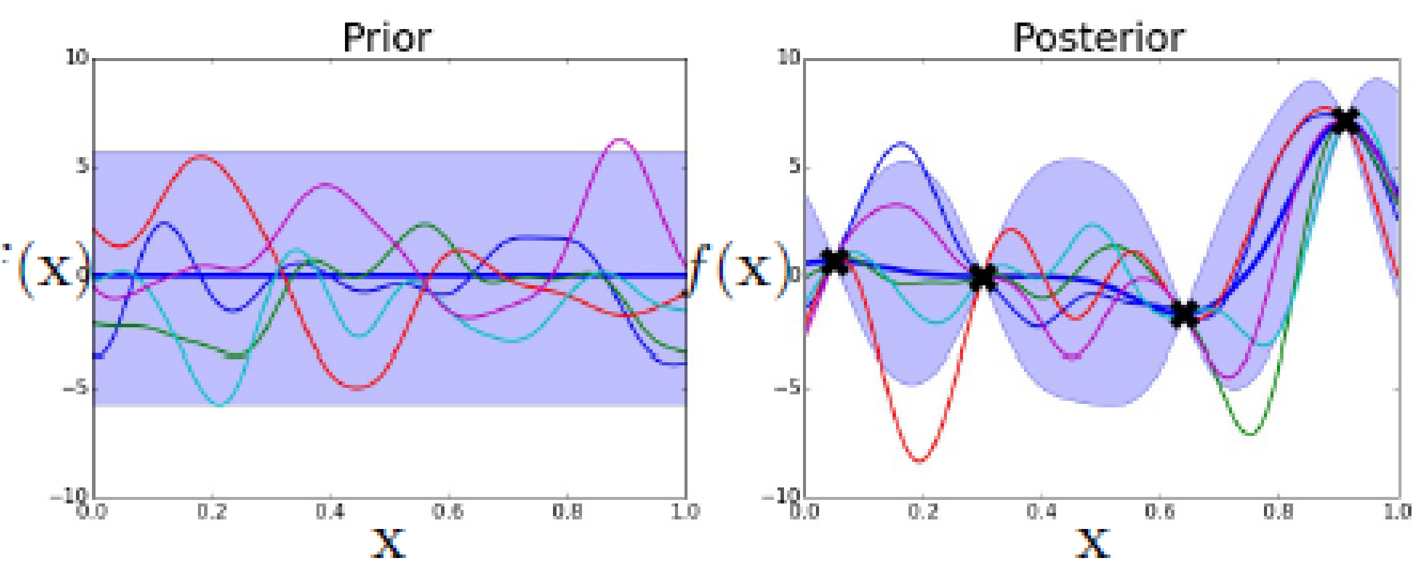
$$\boxed{y = f(\mathbf{x}) + \epsilon, \qquad f \sim \mathcal{GP}(\mu(x), K(\mathbf{x}, \mathbf{x}'; \theta))}$$

History:

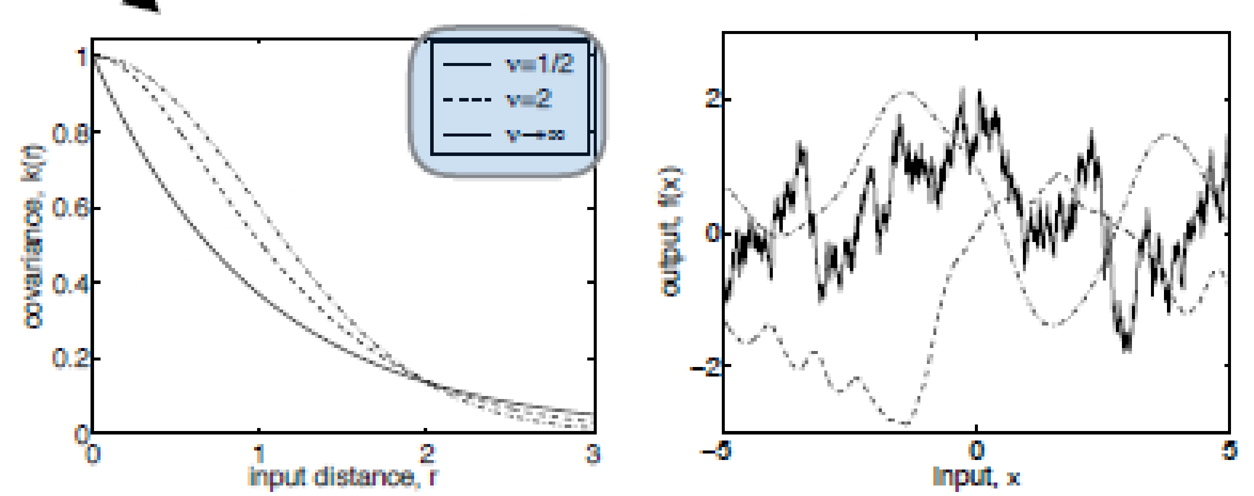
- Wiener–Kolmogorov filtering (1940)
- Kriging (spatial statistics, 1970)
- GP regression (machine learning, 1996)

Workflow:

- Assign a Gaussian process (GP) prior over functions
- Given a training set of observations (x,y) calibrate the GP hyper-parameters
- Use the conditional posterior [f|y] to infer predictions for unobserved x's with quantified uncertainty



covariance function	expression
constant	σ_0^2
linear	$\sum_{d=1}^{D} \sigma_d^2 x_d x'_d$
polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$
squared exponential	$\exp(-\frac{r^2}{2\ell^2})$
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} r\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} r\right)$
exponential	$\exp(-\frac{r}{\ell})$
γ-exponential	$\exp\left(-(\frac{r}{\ell})^\gamma\right)$
rational quadratic	$(1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha}$
neural network	$\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^\top \Sigma \tilde{\mathbf{x}}')}}\right)$



Rasmussen, C. E. Gaussian processes for machine learning (2006)

Solving differential equations from measurements only

$$\mathcal{L}_x u_2(x) = f_2(x)$$

$$\frac{\partial}{\partial x}u_2(x) + \int_0^x u_2(\xi)d\xi = f(x)$$

$$\sum_{d=1}^{10} \frac{\partial^2}{\partial x_d^2}u(x) = f(x)$$

$$u_2(t,x) + \frac{\partial}{\partial x}u_2(t,x) - \frac{\partial^2}{\partial x^2}u_2(t,x) - u_2(t,x) = f(t,x)$$

$${}_{-\infty}D_x^{\alpha}u_2(x) - u_2(x) = f(x)$$

$$u_2(x) \sim \mathcal{GP}(0, g(x,x';\theta)) \xrightarrow{\textit{Linearity}} f_2(x) \sim \mathcal{GP}(0, k(x,x';\theta)) \longrightarrow k(x,x';\theta) = \mathcal{L}_x\mathcal{L}_{x'}g(x,x';\theta)$$

Problem setup:

- $f_2(x)$ is an unknown, black box function
- only scattered, noisy, variable fidelity observations of $f_2(x)$ are available
- we have no data on $u_2(x)$ other than the necessary initial/boundary conditions
- no numerical discretization!

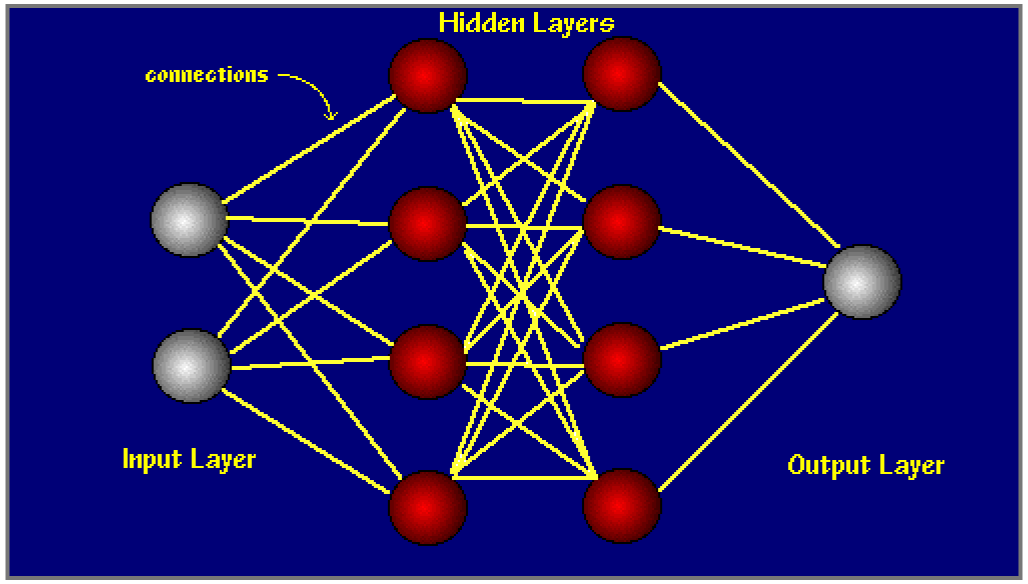
"once we accept that we don't know f, but we do know something, it becomes natural to take a Bayesian approach" P. Diaconis, "Bayesian numerical analysis", 1988

"stochastic methods will transform pure and applied mathematics in the beginning of the third millennium, as probability and statistics will come to be viewed as the natural tools to use in mathematical as well as scientific modeling" D. Mumford, "The dawning age of stochasticity", 2000

Revisiting numerical methods from a statistical inference viewpoint traces all the way back to the Poincar's courses on probability theory!

M. Raissi,., P. Perdikaris, and G.E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, http://128.84.21.199/abs/1607.04805, 2016

Outline



Deep Neural Networks

- Continuous Time
- Discrete Time

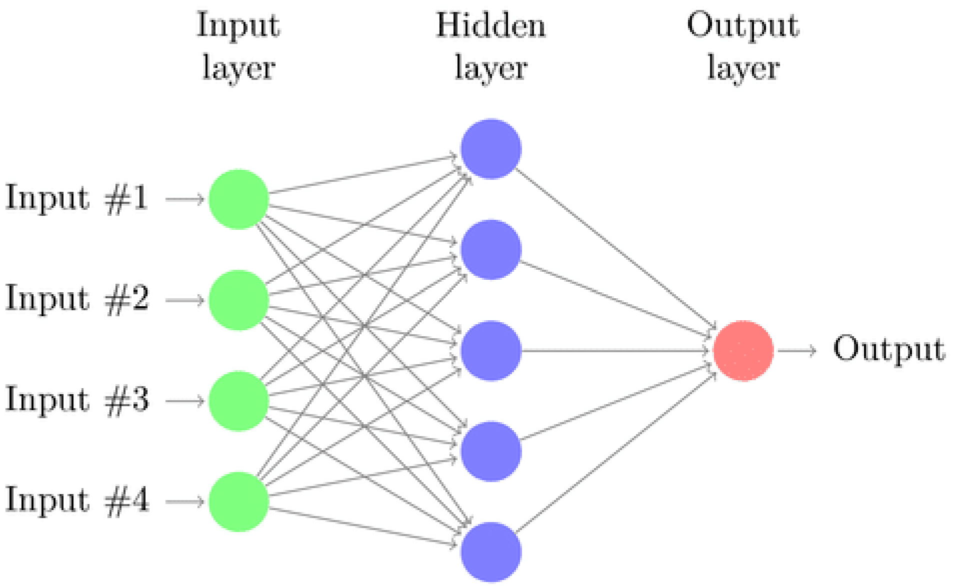
Approximation Theory in Neural Networks: Functions

Theorem (*Cybenko, 1989*)

Let σ be any continuous sigmoidal function. Then, the finite sums of the form

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j \cdot x + \theta_j)$$

are dense in $C(I_d)$.



- G. Cybenko, "Approximation by superpositions of a sigmoidal function", Mathematics of Control, Signals and Systems, 303-314, 2(4), 1989

- Hornik et. al., 1989; Barron (1994); Mhaskar (1996)

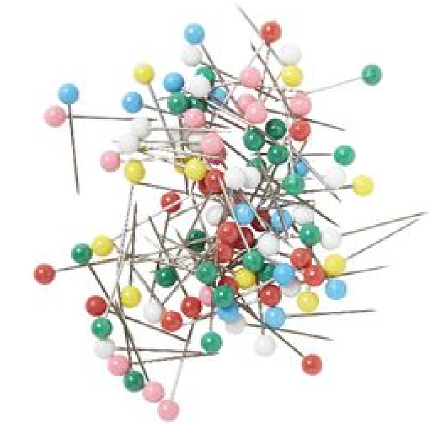
Data + Neural Networks + Physical Laws

*Physics-Informed Neural Networks (PINNs)

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M Raissi, P Perdikaris, GE Karniadakis
Journal of Computational Physics 378, 686-707

arXiv:1711.10561; arXiv:1711.10566

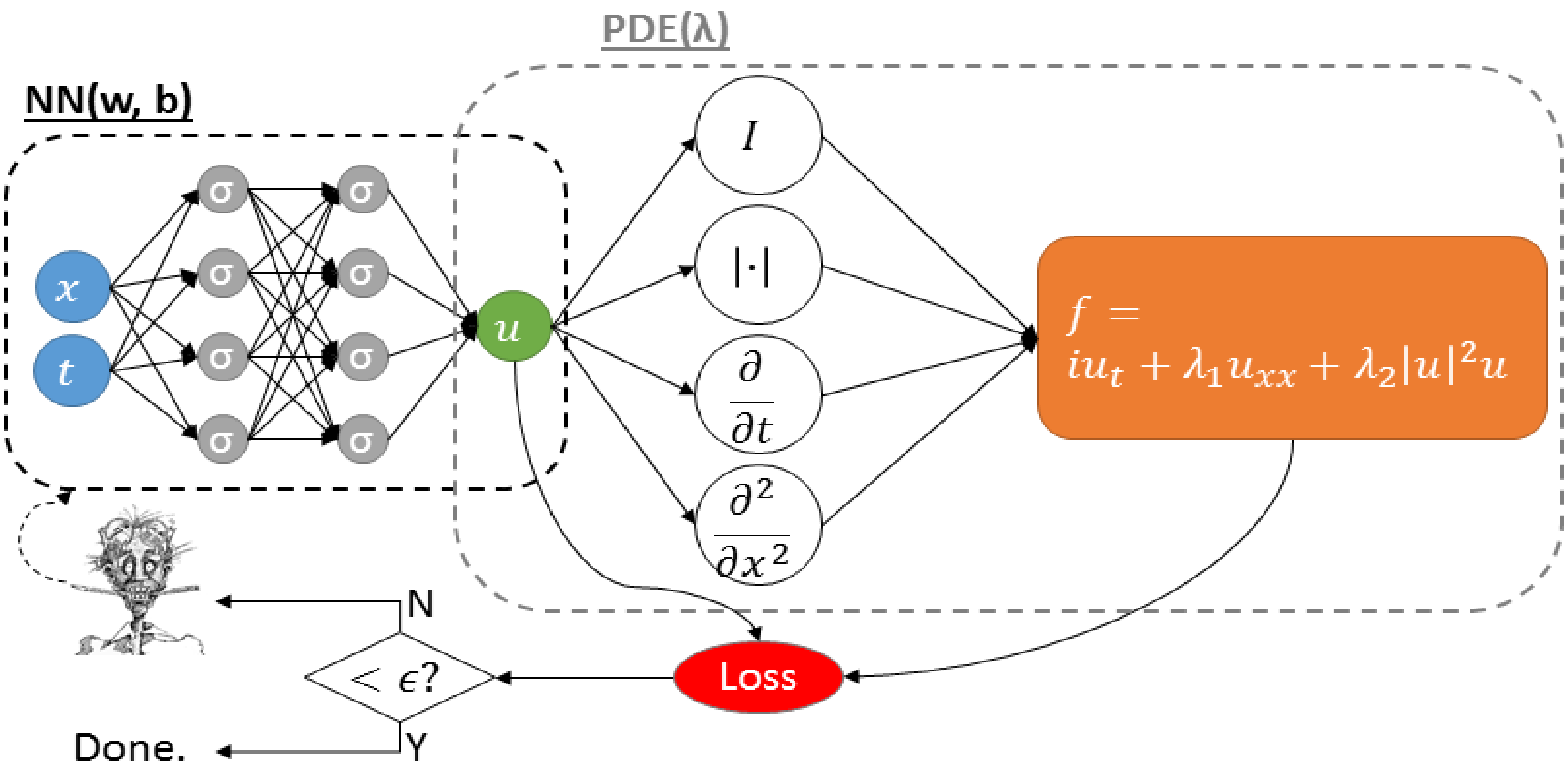


Physics-Informed Neural Networks (PINNs)

- sPINNs: stochastic PINNs
- fPINNs: fractional PINNs
- LePINNs: Levy process PINNs
- nPINNs: Nonlocal PINNs…

What is a PINN? Physics-Informed Neural Network

We employ two (or more) NNs that share the same parameters



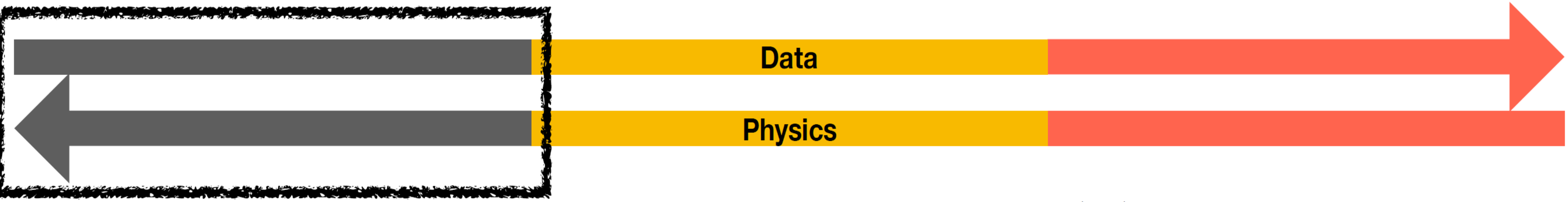
- Minimize:

$$MSE = MSE_u + MSE_f,$$

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

L-BFGS



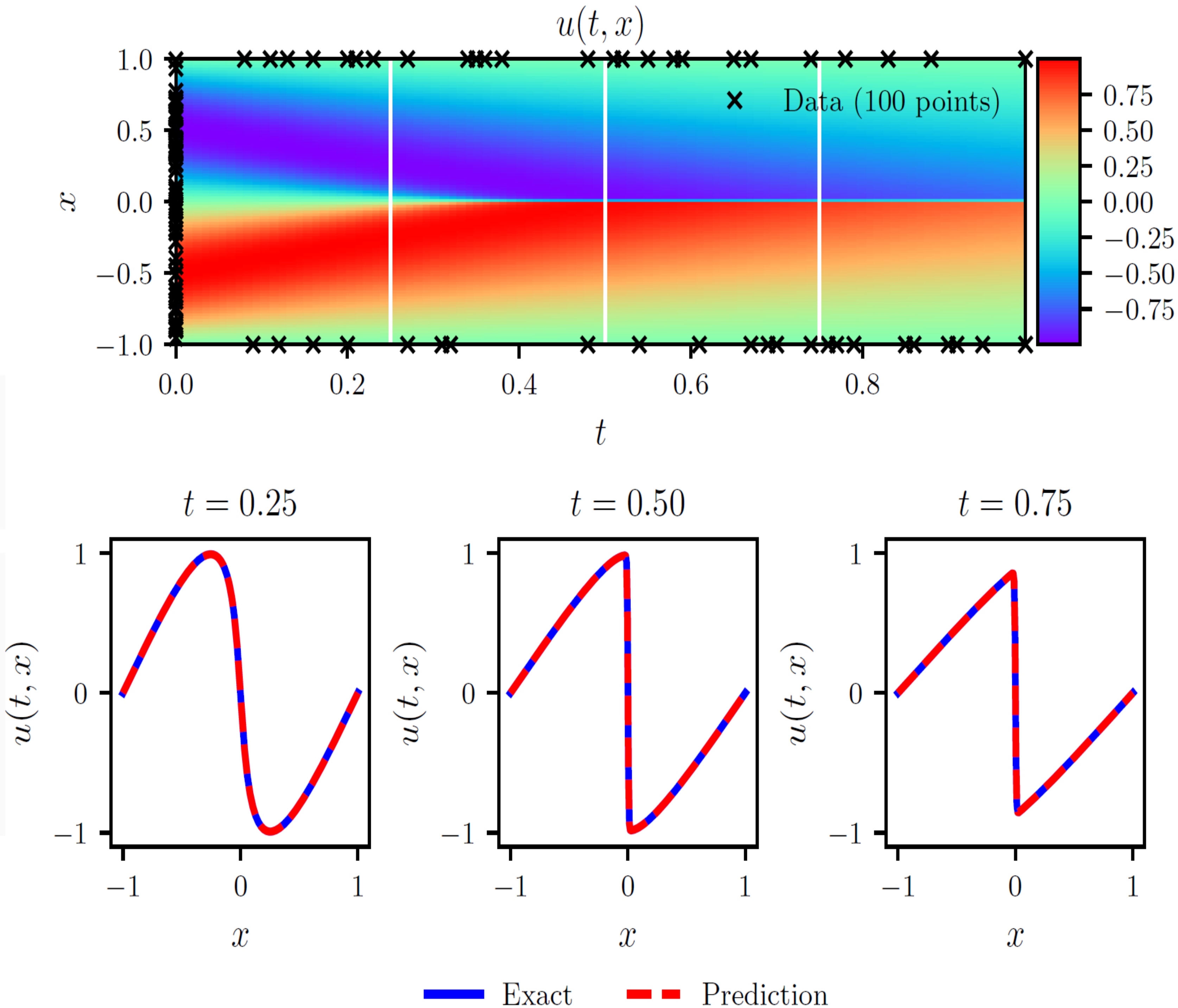
Physics Informed Neural Networks

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0$$

```
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u
```

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

$$\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 + \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$



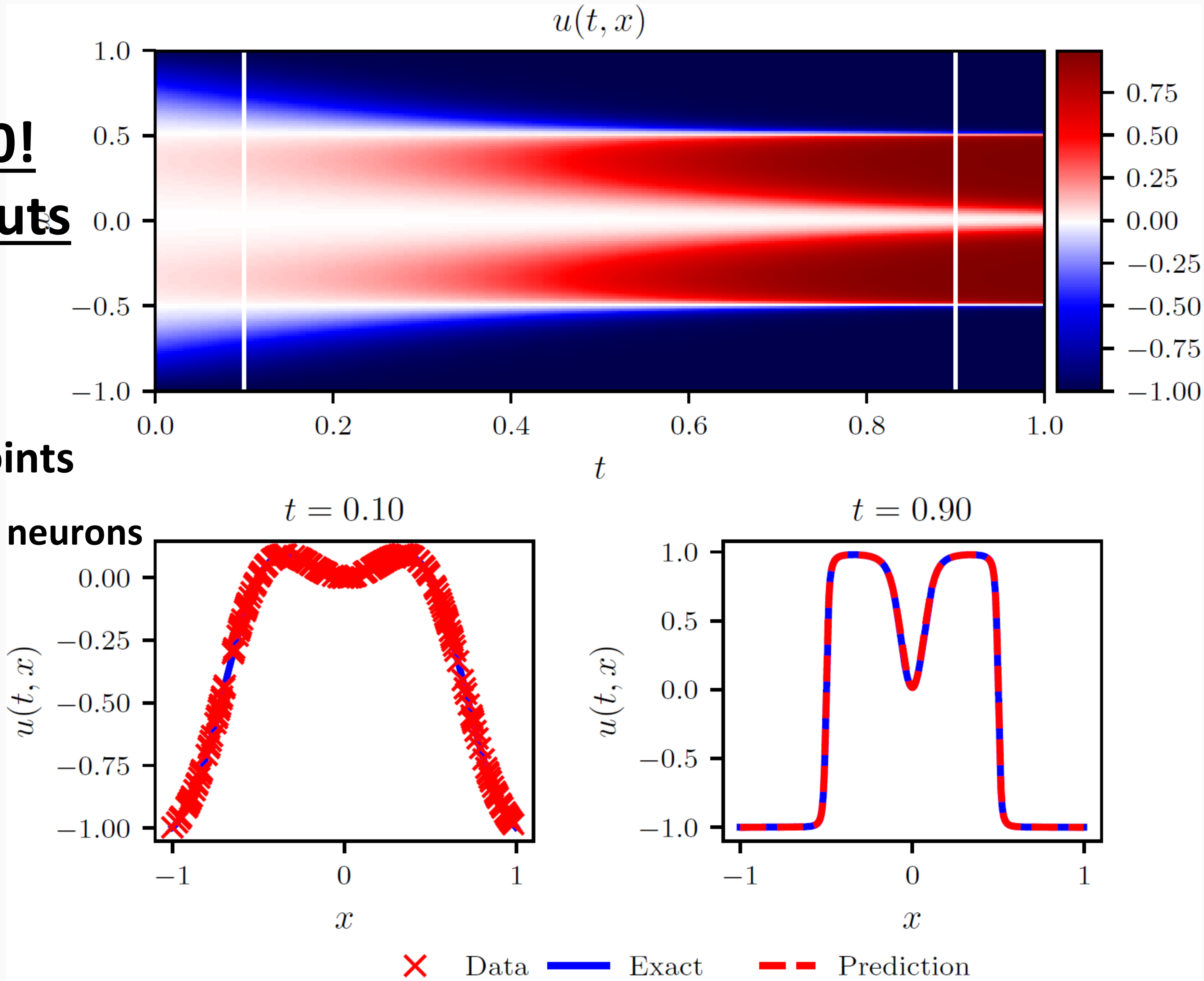
Inference (Discrete Time) *(Allen-Cahn Equation)*

i-RK-100!

101 outputs

IC: 200 points

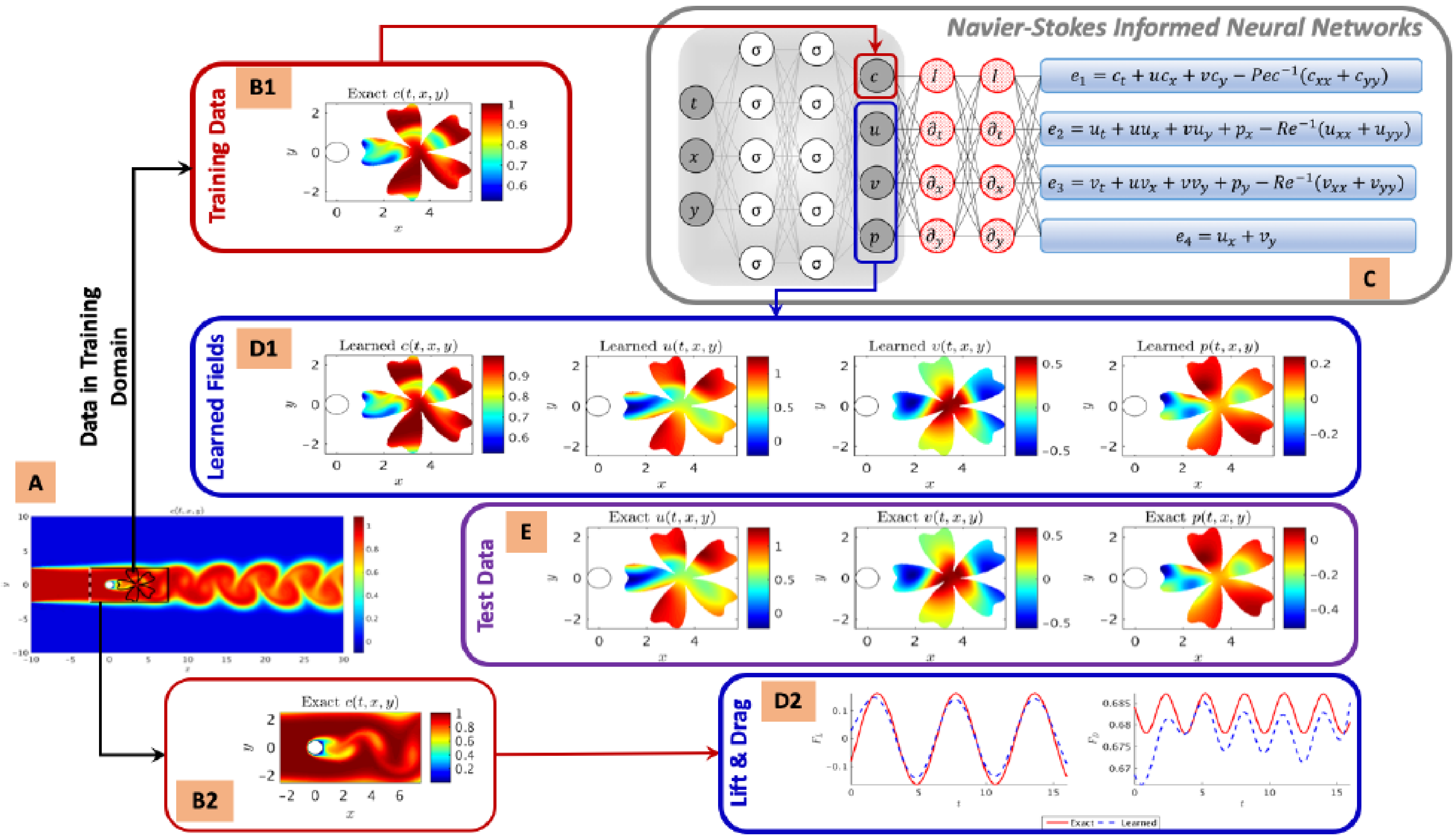
4 layers/200 neurons



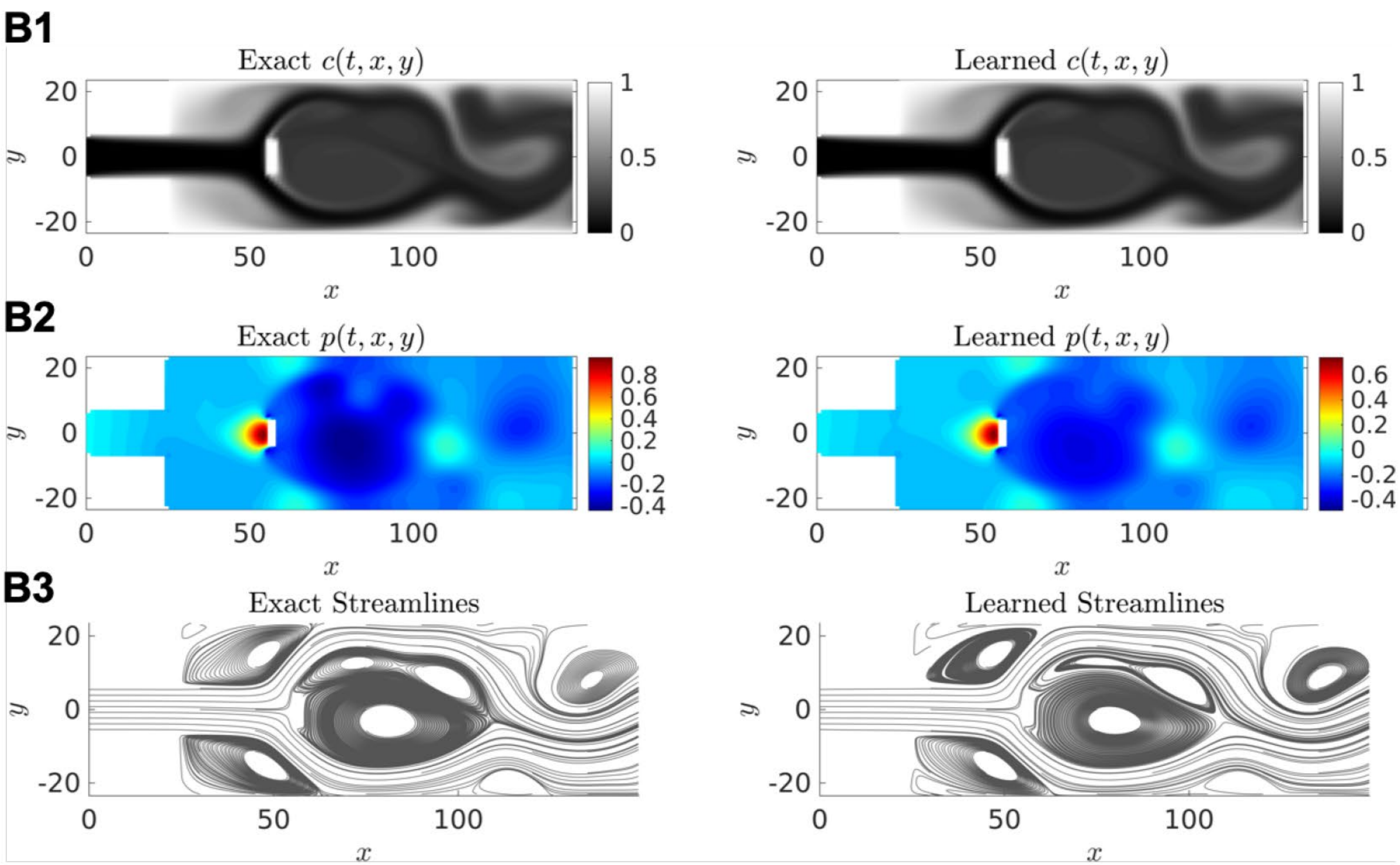
Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations." arXiv:1711.10561 (2017).

Hidden Fluid Mechanics

M. Raissi, A. Yazdani and G.E. Karniadakis, arXiv:1808.04327

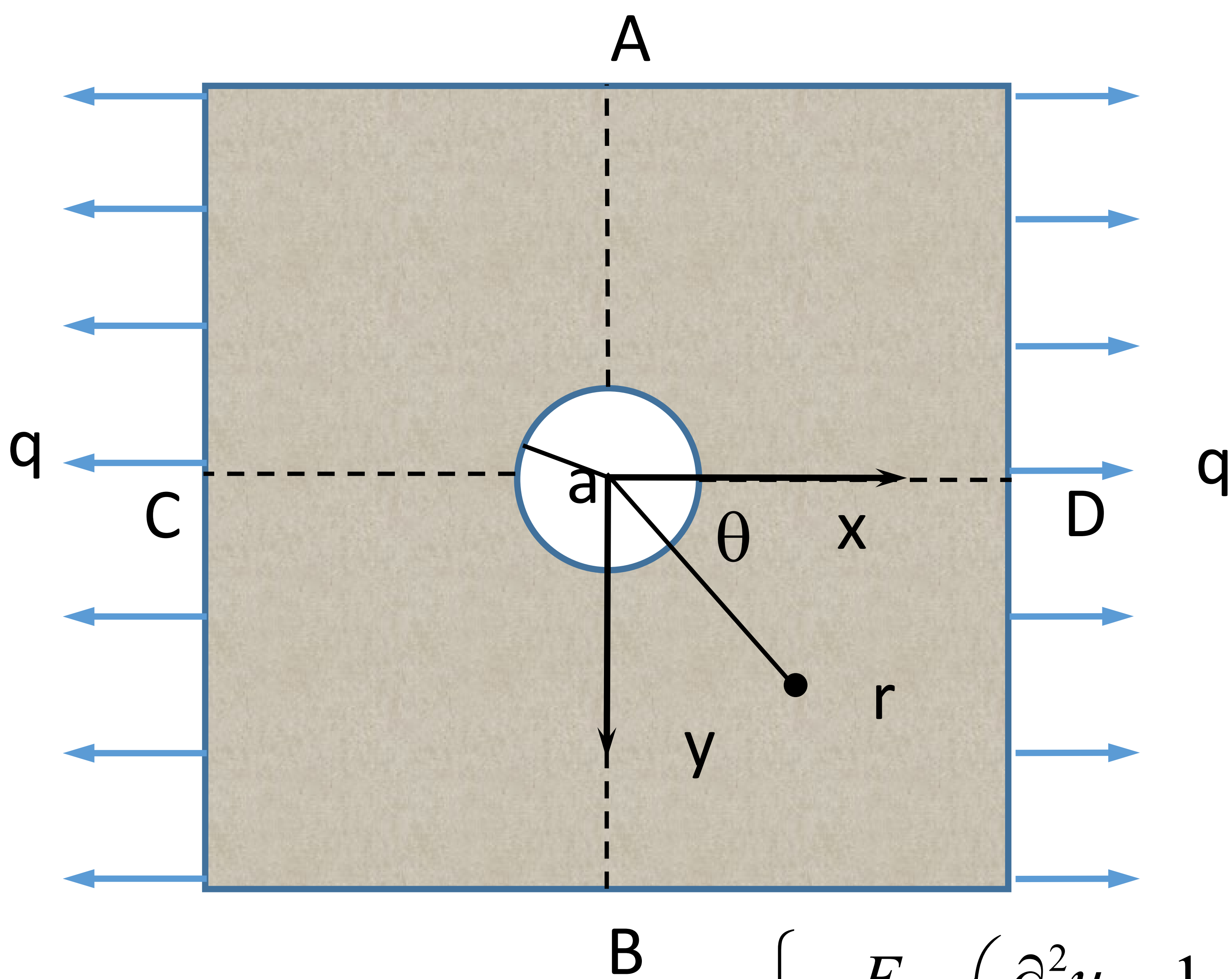


PINNs for the Da Vinci Problem



PINNs for Solid Mechanics

(Dr. Guofei Pang (Brown U) & Dr. Ming Dao (MIT))



Boundary conditions:

- On outside boundary
  $\sigma_x = q,\ \sigma_y = \tau_{xy} = 0$
- On inside boundary
  $\sigma_r = \tau_{r\theta} = 0$

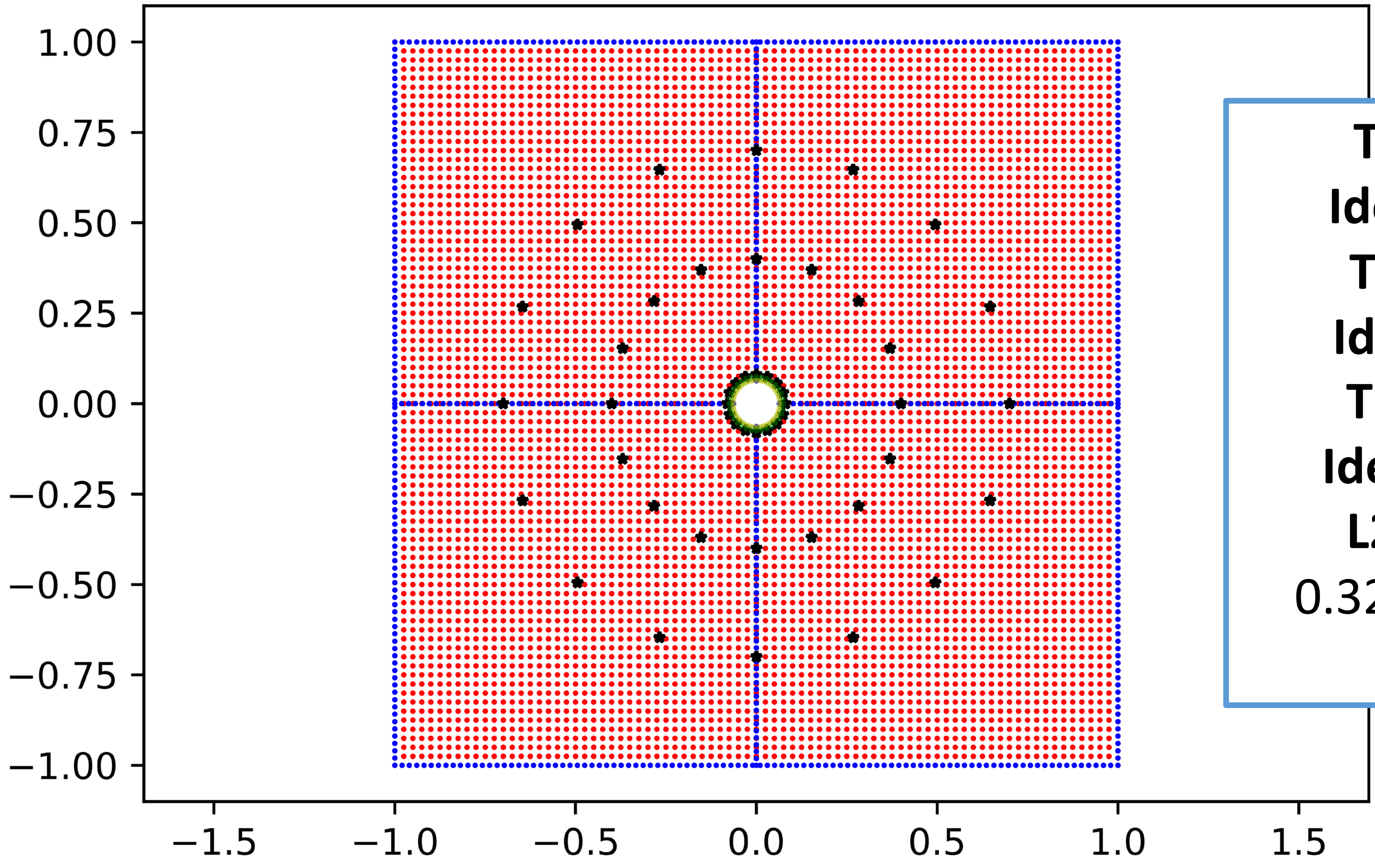
Symmetry constraints:

- On the line AB
  $u = 0$ and $\tau_{xy} = 0$
- On the line CD
  $v = 0$ and $\tau_{xy} = 0$

Governing equations

$$\begin{cases} \dfrac{E}{1-\mu^2}\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{1-\mu}{2}\dfrac{\partial^2 u}{\partial y^2} + \dfrac{1+\mu}{2}\dfrac{\partial^2 v}{\partial x \partial y}\right) = 0 \\ \dfrac{E}{1-\mu^2}\left(\dfrac{\partial^2 v}{\partial y^2} + \dfrac{1-\mu}{2}\dfrac{\partial^2 v}{\partial x^2} + \dfrac{1+\mu}{2}\dfrac{\partial^2 u}{\partial x \partial y}\right) = 0 \end{cases}$$

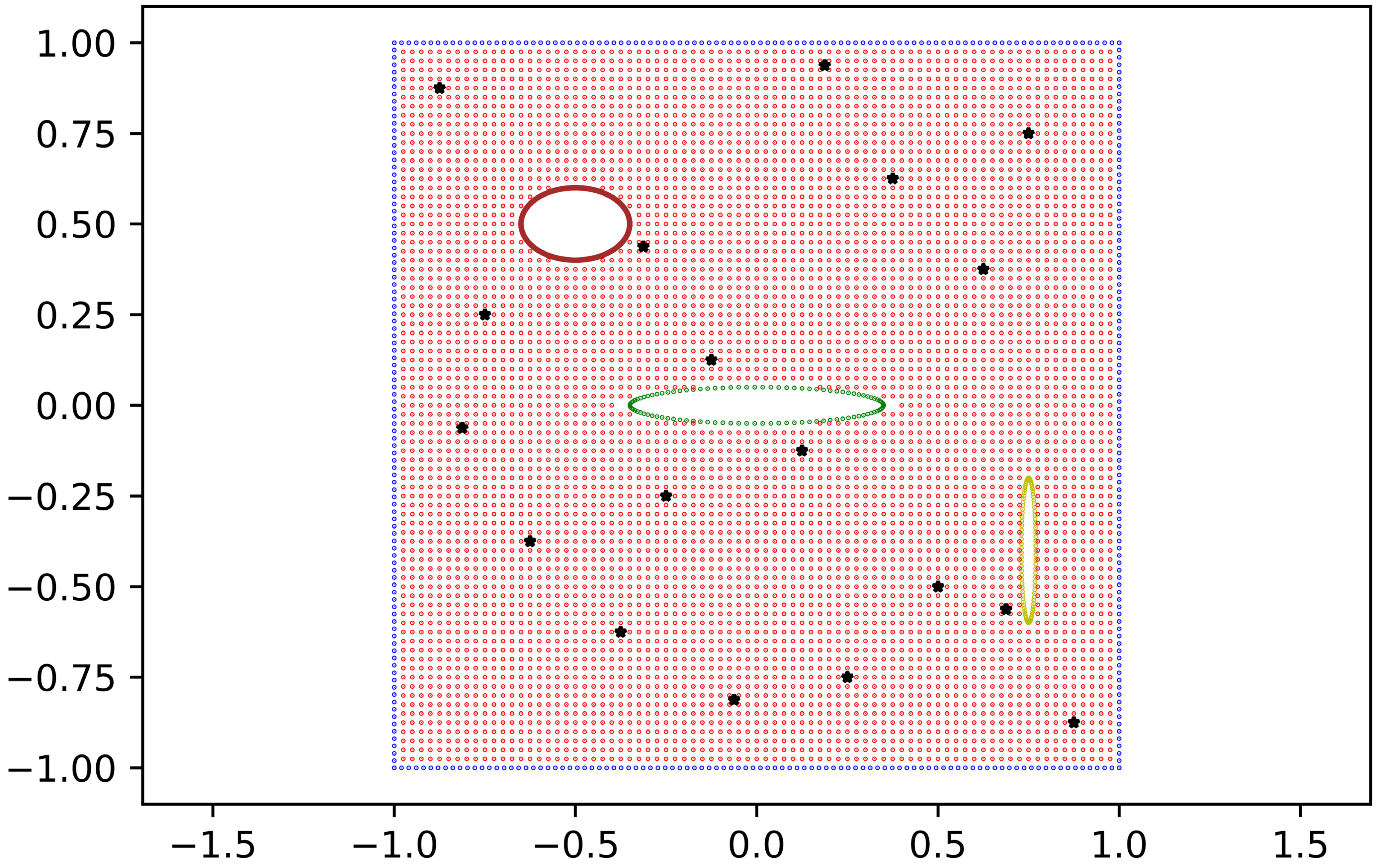
Inverse problem: Identify Young's modulus (E), Poisson's ration (μ) and hole size (r_0)



True E: 1.0
Identified E: 0.999
True μ: 0.25
Identifed μ: 0.248
True r_0: 0.05
Identifed r_0: 0.0495
L2 relative errors:
0.32% (σ_r) 0.31% (σ_θ)
0.36% ($\tau_{r\theta}$)

*Red points: PDE residuals *Green points: Candidate inside BCs
*Blue points: Outside BCs and symmetry constraints
*Black points: sensors

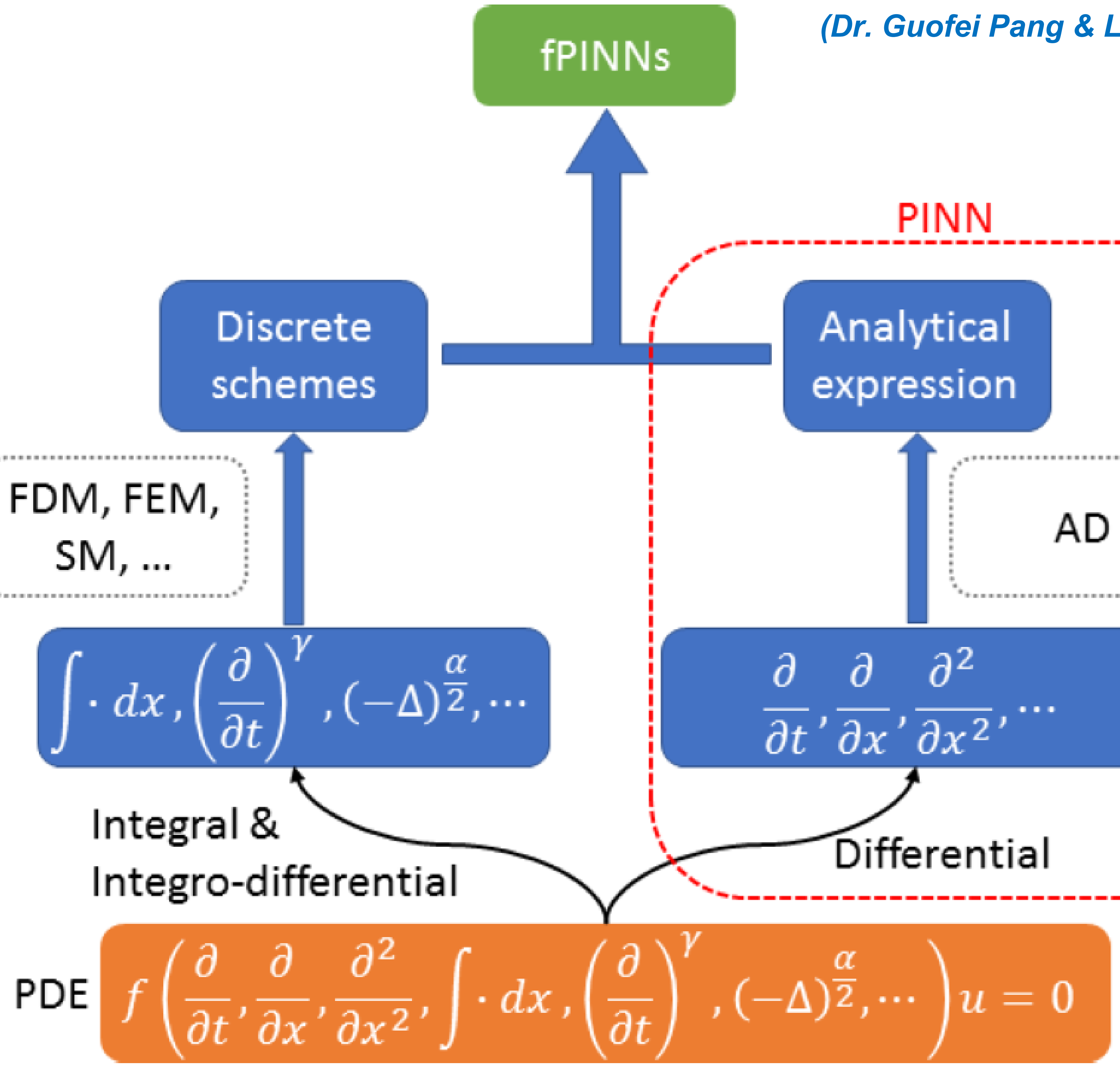
The Real Problem: Find the material defects



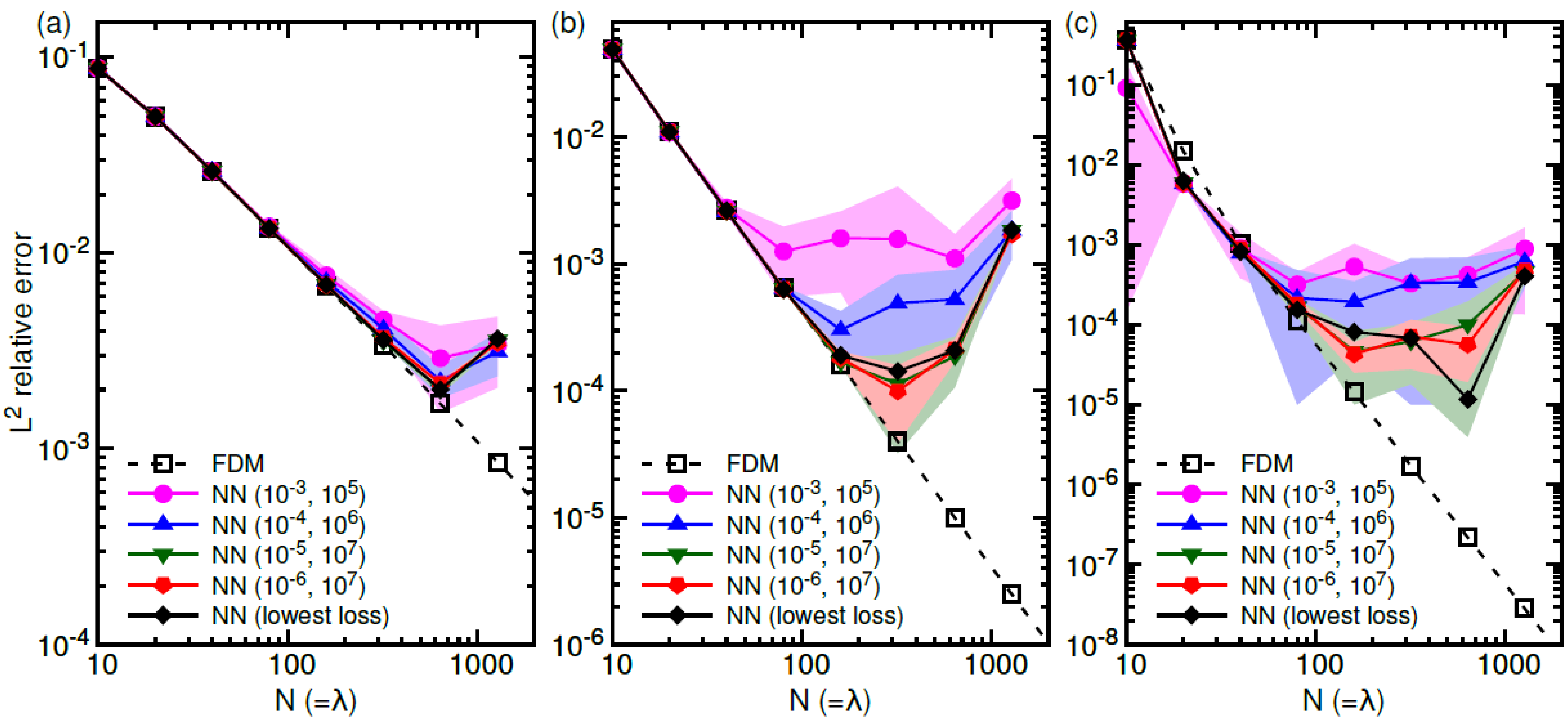
* Red points: PDE residuals *Blue points: Outside BCs
* Brown, green and yellow points: Inside BCs
* Black points: randomly distributed sensors

fPINNs: fractional PINNs

(Dr. Guofei Pang & Lu Lu, Brown U)

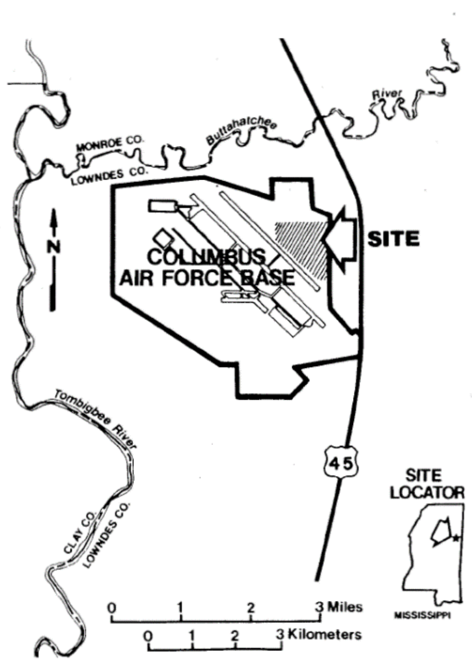


Finite Differences versus fPINNs ($N = \lambda$)



- For small N or λ, the sampling or discretization error dominates.
- For large N or λ, the optimization error dominates.
- The higher the approximation order is, the earlier the optimization error dominates, since the higher order scheme yields more complex loss function.
- NN approximation error is negligible since fPINN can replicate FDM's solution for small N.

Why Fractional Operators + PINNs?



Data of groundwater solute transport
from
Macro-dispersion
Experimental
(MADE) site
at Columbus Air
Force Base

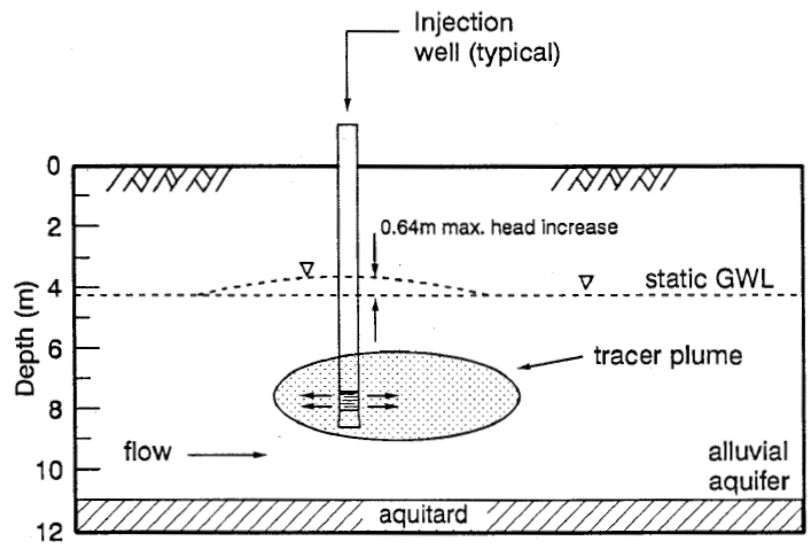
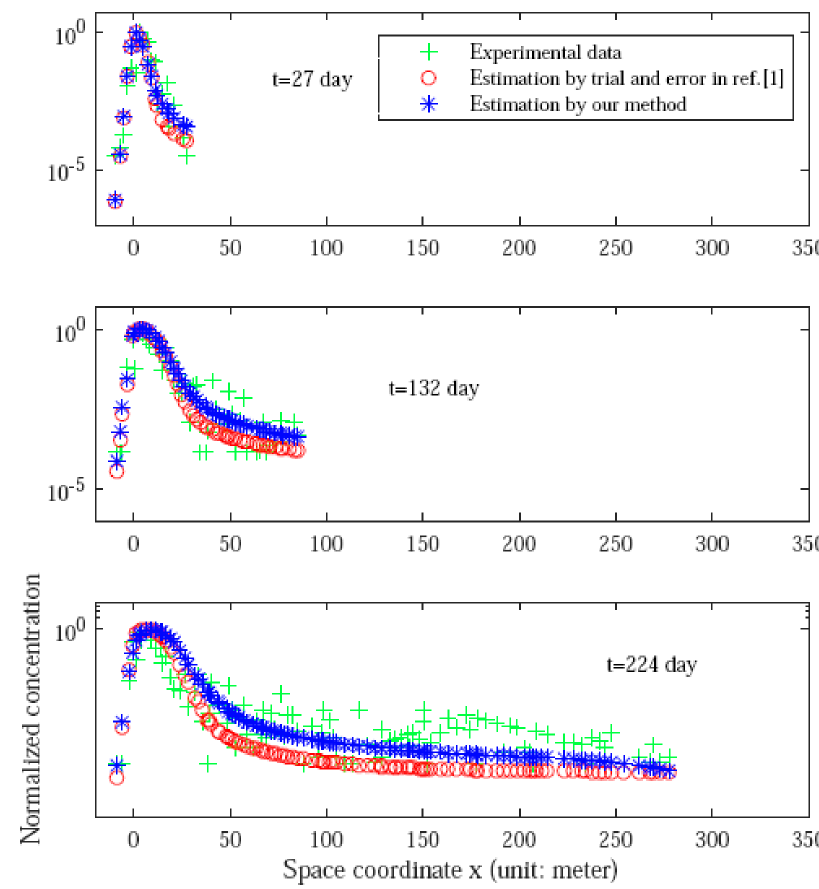


Figure 3.1 Schematic of Injection Well



- **Green: Tritium concentration data from MADE site**
- **Red: Prediction in the literature using trial and error**
- **Blue: Prediction from machine learning**

To Discover (truly) New Equations!

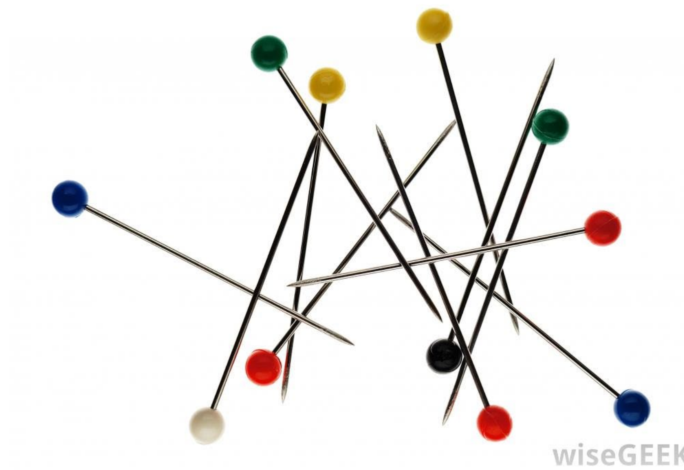
$$\frac{\partial^{0.75} u(x,t)}{\partial t^{0.75}} = -0.14 \times \frac{\partial u(x,t)}{\partial x} + 0.14 \times \frac{\partial^{2-0.0028x} u(x,t)}{\partial |x|^{2-0.0028x}}$$

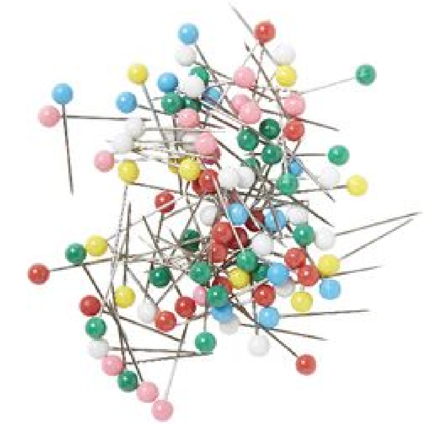
Old fractional model (One example)

$$\frac{\partial^{0.73+0.00053x} u(x,t)}{\partial t^{0.73+0.00053x}} = -0.14 \times \frac{\partial u(x,t)}{\partial x} + 0.14 \times \frac{\partial^{1.87-0.0029x} u(x,t)}{\partial |x|^{1.87-0.0029x}}$$

New fractional model

Comparison	Old fractional model	New fractional model
How to identify parameters	Trial and error	Machine learning
Extension to a large number of parameters	Difficult	Easy
Prediction accuracy	Low	High

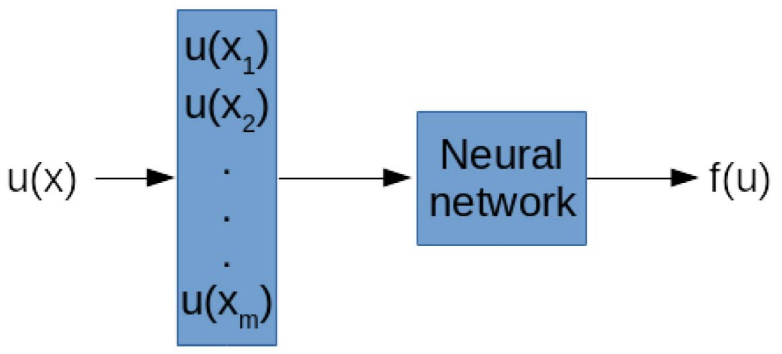




Mathematics of PINNs

- Nonlinear approximation theory
- Robust training & optimization
- Multifidelity approximation
- Learnability & small data

Approximation Theory in Neural Networks: Functionals



Theorem (*Chen and Chen, 1993*):

Suppose that U is a compact set in $C[a,b]$, f is a continuous functional defined on U, and $\sigma(x)$ is a bounded sigmoidal function, then for any $\varepsilon > 0$, there exist $m+1$ points $a = x_0 < \cdots < x_m = b$, a positive integer N and constants c_i, θ_i, $\xi_{i,j}$, $i = 1, \cdots, N$, $j = 0,1, \cdots, m$, such that

$$\left| f(u) - \sum_{i=1}^{N} c_i \sigma\left(\sum_{j=0}^{m} \xi_{i,j} u\left(x_j\right) + \theta_i \right) \right| < \varepsilon$$

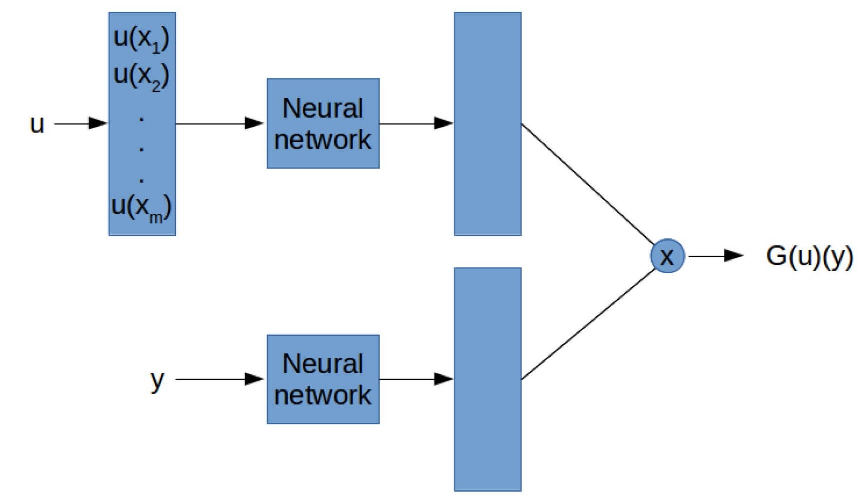
holds for all $u \in U$.

T.P. Chen and H. Chen, Approximations of continuous functionals by neural networks with application to dynamic systems, IEEE Transactions on Neural Networks, 910-918, 4(6), 1993.

Universal Approximation to Nonlinear Operators by Neural Networks with Arbitrary Activation Functions and Its Application to Dynamical Systems

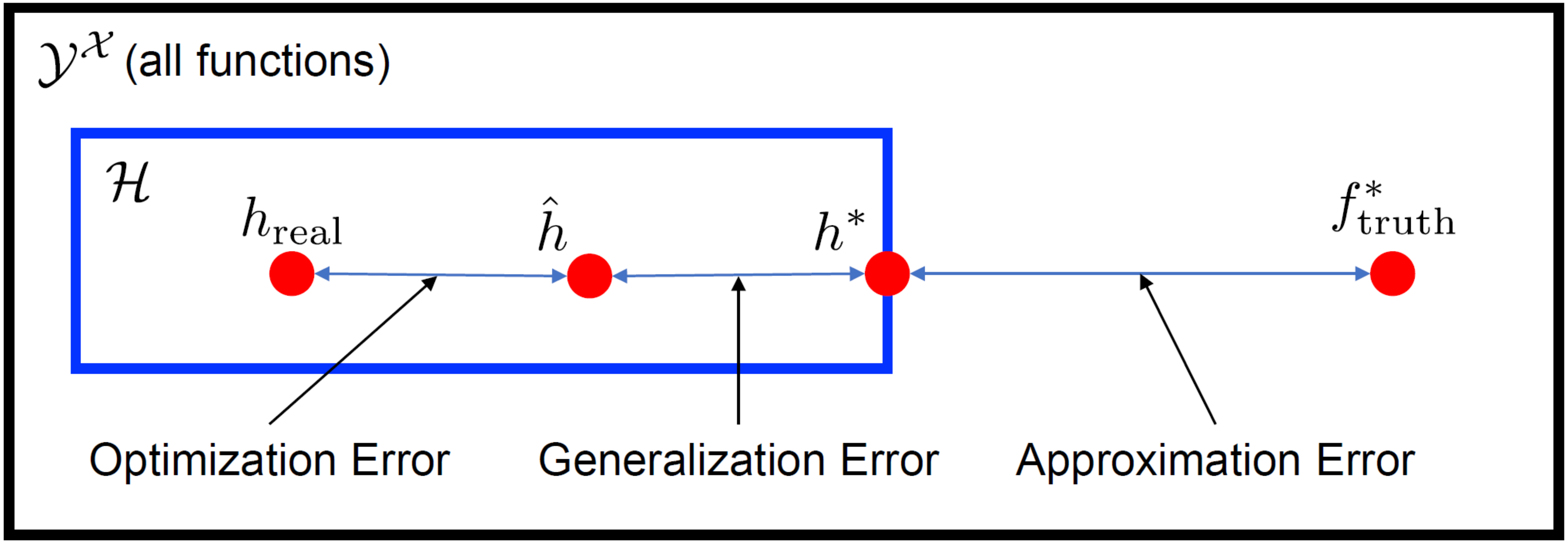
Tianping Chen and Hong Chen

$$\left| G(u)(y) - \sum_{k=1}^{N} \sum_{i=1}^{M} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \sigma(w_k y + \zeta_k) \right| < \epsilon$$



we show how to construct neural networks to approximate the output of a dynamical system as a whole, not merely at a fixed point, thus show the capability of neural network in identifying dynamic systems. Moreover, we point out that using existing algorithms in literatures (for example, backpropagation algorithm), we can determine those parameters in the network, i.e., identify the system.

Overview: Statistical Learning perspective



- General loss: $\mathcal{L}_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(h(x), y)\right] \implies h^*$
- Empirical loss: $\mathcal{L}_{\mathcal{T}_m}(h) = \frac{1}{m}\sum_{i=1}^{m}\ell(h(x_i), y_i) \implies \hat{h}$
- h_{real}: an actual approximation (e.g. after 1M gradient descent iterations)

Shallow Networks vs Deep Networks

- **It seems that deep networks appear to perform better than shallow ones of comparable size**. From approximation theoretical point of view, there are some explanations:

1. *Eldan et. al. (2016)* showed that there is a simple function expressible by a small 2-hidden-layer feedforward neural networks, which cannot be approximated by any 1-hidden-layer network with the same accuracy, unless its width is exponential in the dimension.
2. *Liang et. al. (2017) and Yarotsky (2016)* the authors claimed the number of neurons needed by a shallow network to approximate a function is exponentially larger than the corresponding number of neurons needed by a deep network for a given degree of function approximation.
3. *Hanin et. al. (2017)* show that any continuous function $f: [0,1]^{d_{in}} \to \mathbb{R}^{d_{out}}$ can be approximated by a ReLU forward neural net of width $d_{in} + d_{out}$; to achieve ϵ error uniformly, the depth should be in the order $\left(\frac{O(\mathrm{diam}(K))}{\omega_f^{-1}(\epsilon)}\right)$

R. Eldan and O. Shamir, The power of depth for feedforward neural networks, Conference on Learning Theory, 907-940, 2016.

S. Liang and R. Srikant, Why deep neural networks for function approximation? arXiv:1610.04161, 2016.

D. Yarotsky, Error bounds for approximations with deep ReLU networks, Neural Networks, 103-114, 94, 2017.

B. Hanin and M. Sellke, Approximating Continuous Functions by ReLU Nets of Minimal Width, arXiv:1710.11278, 2017.

Function Approximation by Deep and Narrow NN

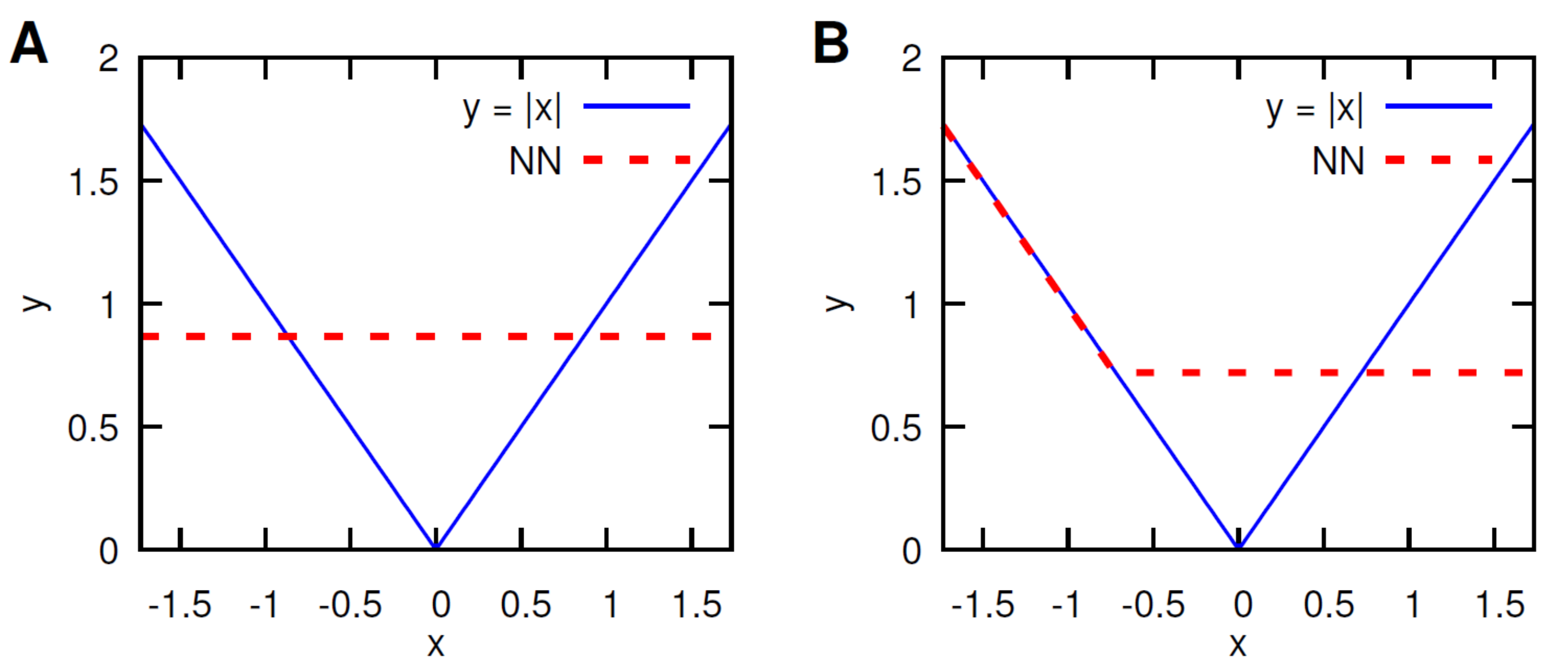
1D Examples

$f(x) = |x|$

- $|x| = \text{ReLU}(x) + \text{ReLU}(-x) = \begin{bmatrix} 1 & 1 \end{bmatrix} \text{ReLU}(\begin{bmatrix} 1 \\ -1 \end{bmatrix} x)$
- 2-layer with width 2

Train a 10-layer ReLU NN with width 2 (MSE loss, whatever optimizer)

- Collapse to the mean value (A): $\sim$93%
- Collapse partially (B)

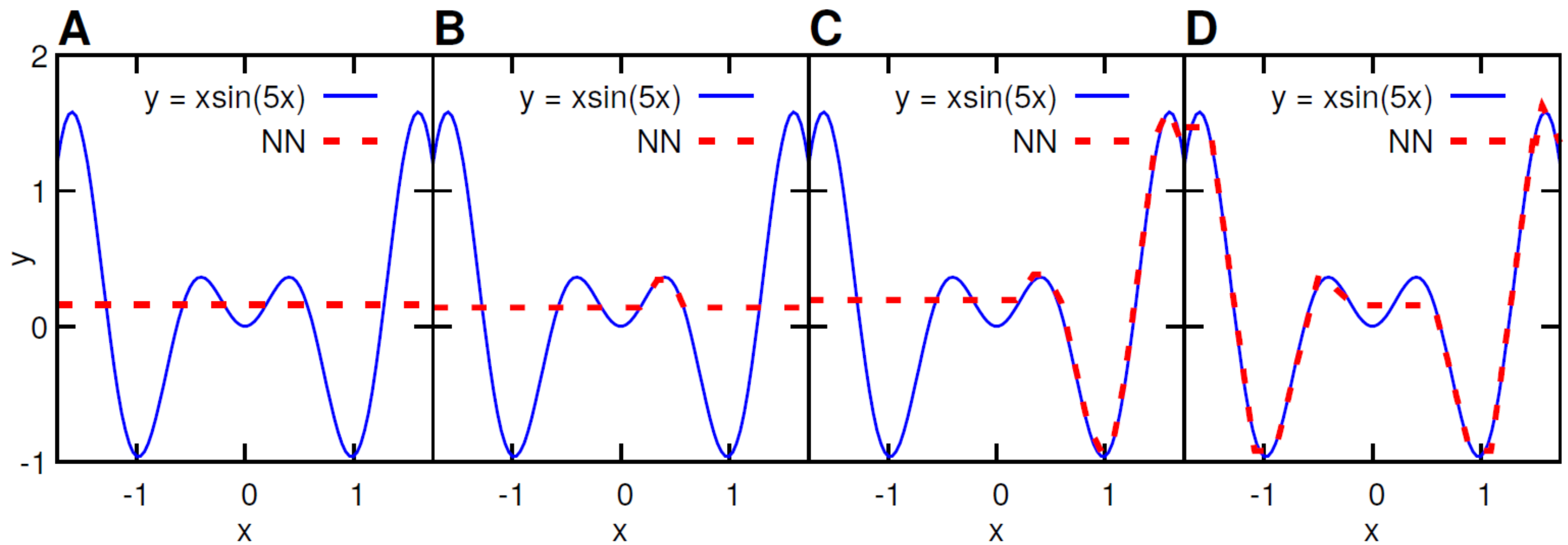


arXiv:1808.04947; arXiv:1903.06733

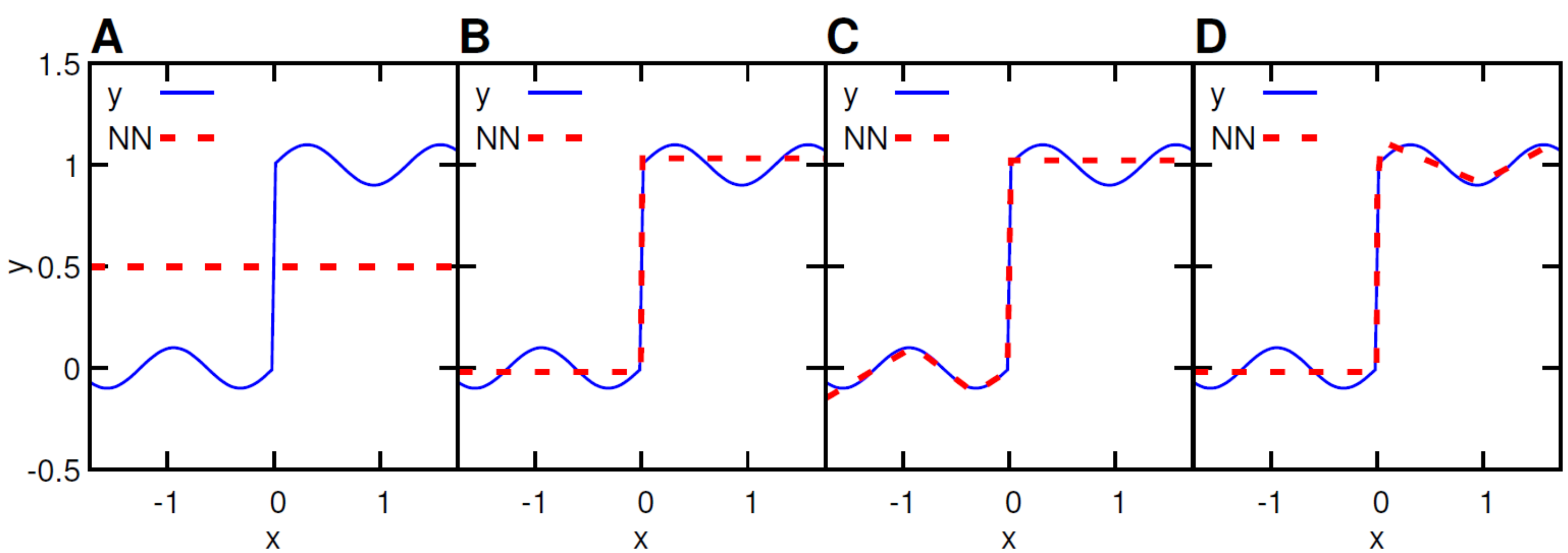
Function Approximation by Deep and Narrow NN

1D Examples

$$f(x) = x\sin(5x)$$



$$f(x) = 1_{\{x>0\}} + 0.2\sin(5x)$$

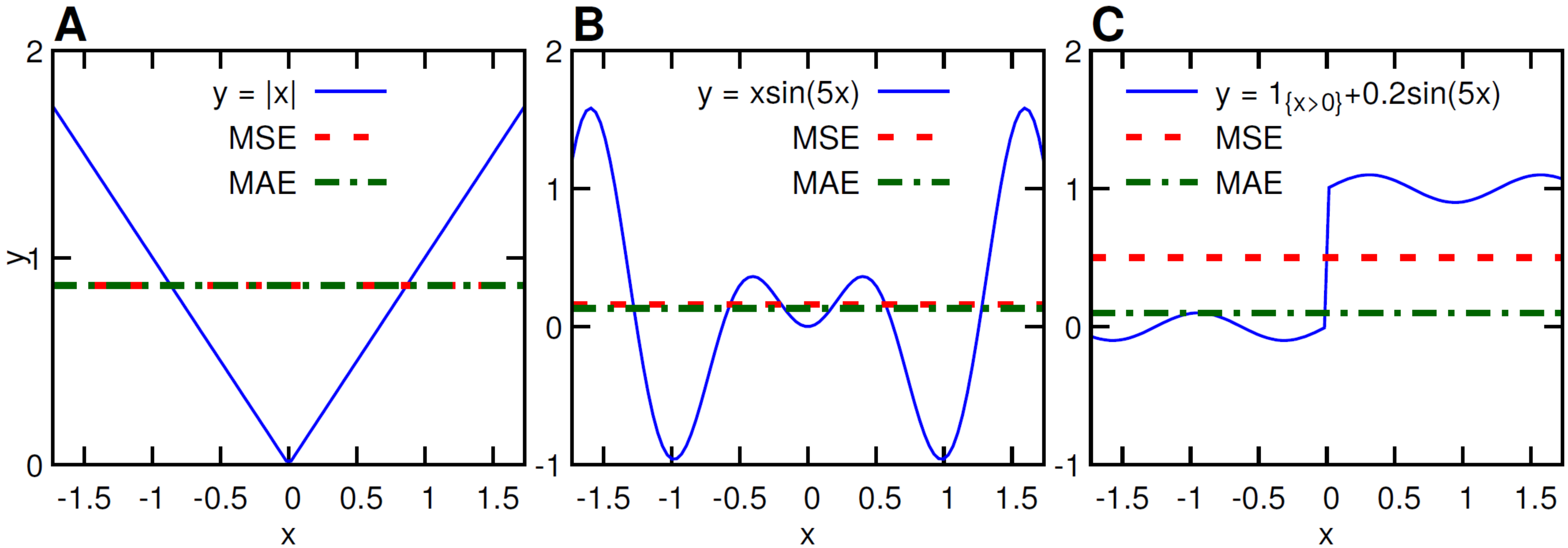


BROWN

arXiv:1808.04947; [arXiv:1903.06733]()

Loss

- Mean squared error (MSE) $\Rightarrow$ mean
- Mean absolute error (MAE) $\Rightarrow$ median



arXiv:1808.04947; arXiv:1903.06733

Function Approximation by Deep and Narrow NN

[(Lu Lu & Dr. Yeonjong Shin, Brown U)]

Training of NNs

- NP-hard [Sima, 2002]
- Local minima [Fukumizu & Amari, 2002]
- Bad saddle points [Kawaguchi, 2016]

ReLU

- Dying ReLU neuron: stuck in the negative side

Deep ReLU nets?

Dying ReLU network

NN is a constant function after initialization

Collapse

NN converges to the "mean" state of the target function during training

arXiv:1903.06733

Function Approximation by Deep and Narrow NN

$\mathcal{N}^L$ will eventually Die in probability as $L \to \infty$

Theorem 1 (Before Training)

Let $\mathcal{N}^L(\mathbf{x})$ be a ReLU NN with L layers, each having $N_1, \cdots, N_L$ neurons. Suppose

1. Weights are independently initialized from a symmetric distribution around 0,
2. Biases are either from a symmetric distribution or set to be zero.

Then

$$P(\mathcal{N}^L(\mathbf{x}) \text{ dies}) \leq 1 - \prod_{\ell=1}^{L-1} (1 - (1/2)^{N_\ell}).$$

Furthermore, assuming $N_\ell = N$ for all ℓ,

$$\lim_{L\to\infty} P(\mathcal{N}^L(\mathbf{x}) \text{ dies}) = 1, \qquad \lim_{N\to\infty} P(\mathcal{N}^L(\mathbf{x}) \text{ dies}) = 0.$$

arXiv:1903.06733

Dead Networks would Collapse

Theorem 2 (During Training)

Suppose the ReLU NN dies. Then for any loss $\mathcal{L}$, the network is optimized to a constant function by any gradient based method.

Theorem 3 (Before Training: special case)

Let $\mathcal{N}^L(\mathbf{x})$ be a bias-free ReLU NN with $L \geq 2$ layers, each having N neurons at $d_{\text{in}} = 1$. Suppose weights are independently initialized from continuous symmetric distributions around 0. Then

$$1 - \prod_{\ell=1}^{L-1} (1 - (1/2)^N) \geq P(\mathcal{N}^L(x) \text{ dies})$$

$$\geq 1 - (\mathcal{P}_{22})^{L-2} - \frac{(1 - 2^{-N+1})(1 - 2^{-N})}{1 + (N-1)2^{-N}}((\mathcal{P}_{22})^{L-2} - (\mathcal{P}_{33})^{L-2})$$

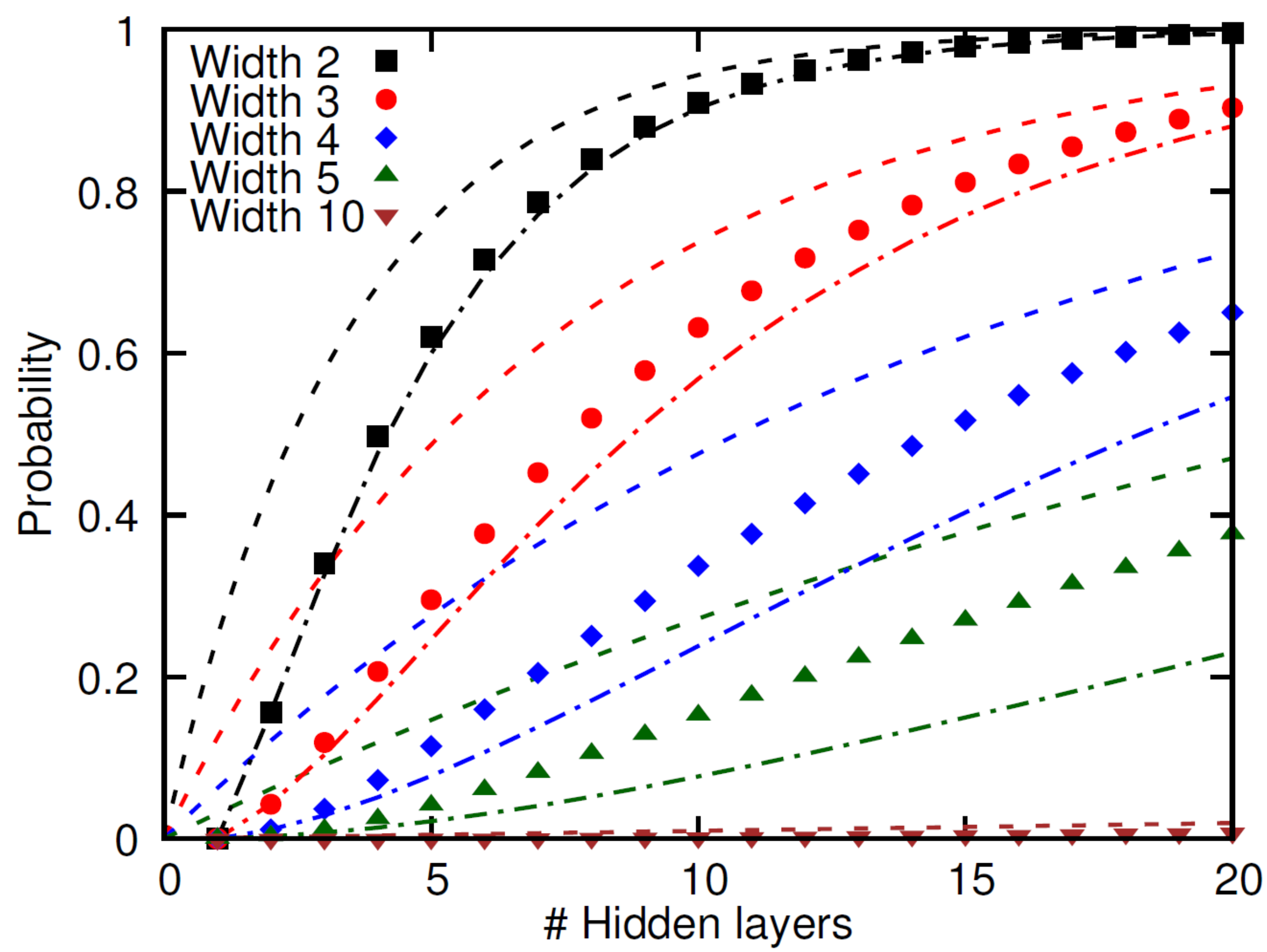
where $\mathcal{P}_{22} = 1 - \frac{1}{2^N}$ and $\mathcal{P}_{33} = 1 - \frac{1}{2^{N-1}} - \frac{N-1}{4^N}$.

Function Approximation by Deep and Narrow NN

Numerical Test (1M runs per point)

- A ReLU NN with $d_{in} = 1$
- Weights randomly initialized from symmetric distributions
- Biases are initialized to 0

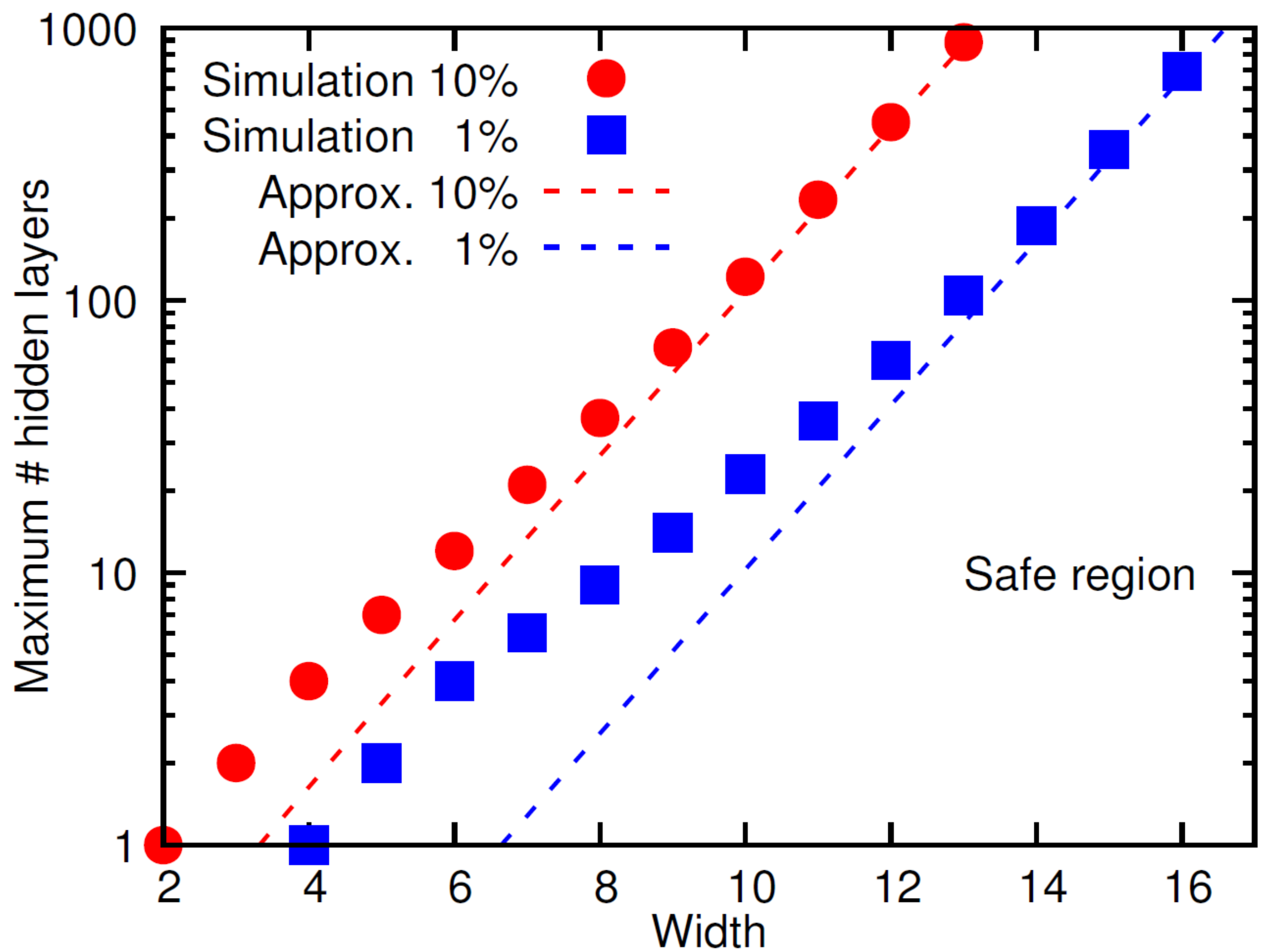
More likely to die when it is deeper and narrower



Function Approximation by Deep and Narrow NN

Safe Operating Region for a ReLU NN (use of Upper Bound)

Keep the dying probability $< 10\%$ or 1%



[arXiv:1903.06733](https://arxiv.org/abs/1903.06733)

BROWN

Randomized Asymmetric Initialization

- **Goal**: Design an initialization method to avoid
  - 1. vanishing gradient (dying networks)
    - by decreasing $P(\mathcal{N}^L(\mathbf{x}) \text{ dies})$
    - **asymmetric distribution**
  - 2. exploding gradient
    - by properly choosing initialization parameters
    - **the length map analysis**

$$q_\ell = \frac{E\left[\|\mathcal{N}^\ell(\mathbf{x})\|^2\right]}{N}$$

arXiv:1903.06733

Randomized Asymmetric Initialization

- **Our initialization method**:
- 1. Choose a probability dist. P on $[0, M]$ (ex. $\mathrm{Beta}(\alpha, \beta)$).
- 2. Each row, $\mathbf{V}_j^\ell := [\mathbf{W}_j^\ell, \mathbf{b}_j^\ell] \sim N(0, \sigma^2)$
- 3. ***Randomly*** choose one component of $\mathbf{V}_j^\ell$, say, $(\mathbf{V}_j^\ell)_k$

  $(\mathbf{V}_j^\ell)_{N+1} = \mathbf{b}_j^\ell$

  $(\mathbf{V}_j^\ell)_k = (\mathbf{W}_j^\ell)_k$
- 4. Set $(\mathbf{V}_j^\ell)_k \sim \mathrm{P}$.
- **Intuition:** Since every entry of $\mathbf{n}^{\ell-1} := [\phi(\mathcal{N}^{\ell-1}(\mathbf{x})), 1]$ is non-negative, the probability

$$P\left(\langle \mathbf{V}_j^\ell, \mathbf{n}^{\ell-1}\rangle > 0 | \mathbf{n}^{\ell-1}\right) \geq 0.5.$$

is greater or equal to those to the symmetric initialization.

arXiv:1903.06733

Dying probability of new initialization

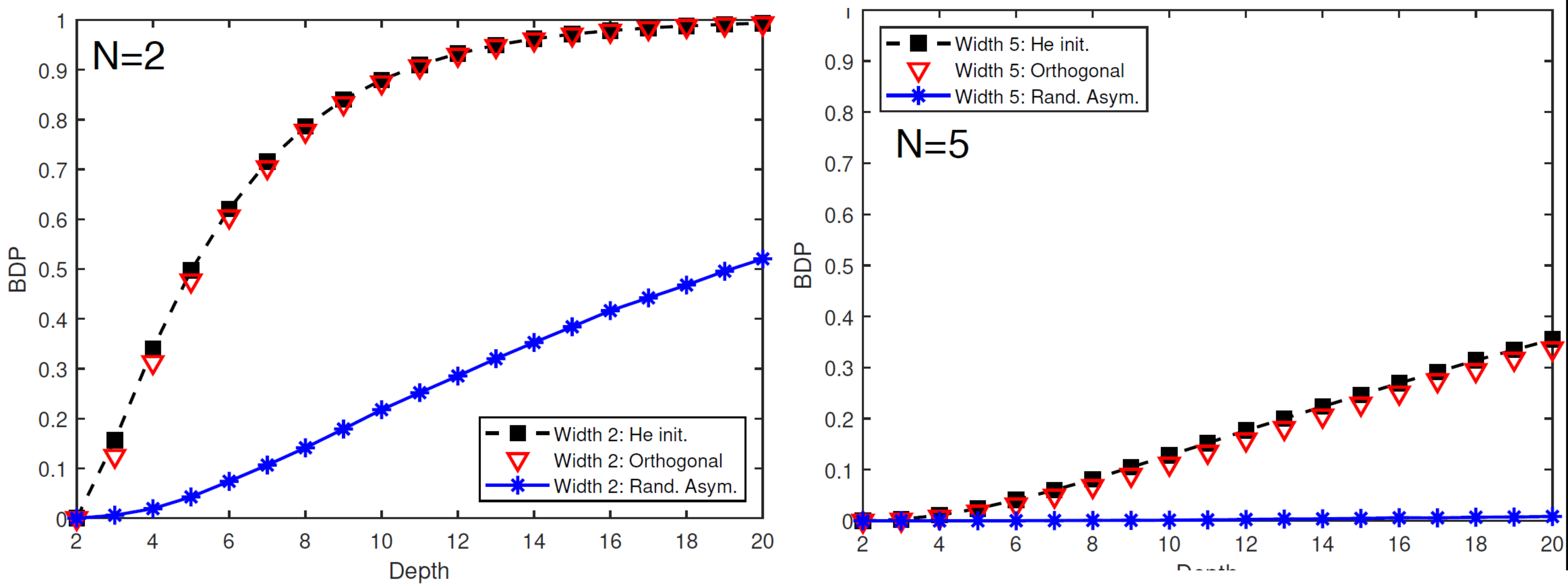
- **Thm**: Let $\mathcal{N}^L(\mathbf{x})$ be the L-layer ReLU Neural Network with each layer having N neurons. If $\mathbf{W}^\ell, \mathbf{b}^\ell$ are chosen as described, then
$$P(\mathcal{N}^L(\mathbf{x}) \text{ dies}) \le 1 - \prod_{j=1}^{L-1} (1 - (1/2 - \delta_j)^N)$$
for some $0 < \delta_j < 1/2$ and $\delta_1 = 0$.
- In the case of symmetric initialization, $\delta_j = 0, \forall j$, that is
$$P(\mathcal{N}^L(\mathbf{x}) \text{ dies}) \le 1 - \prod_{j=1}^{L-1} (1 - (1/2)^N)$$

arXiv:1903.06733

Dying Probability - Initialization

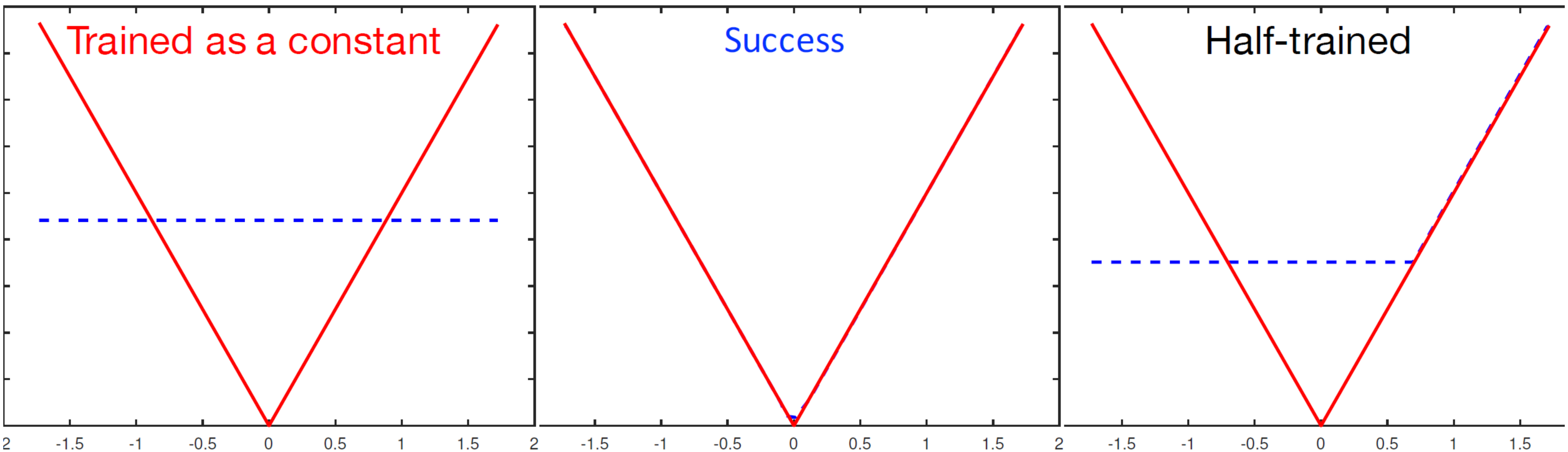
- Run 100,000 independent simulations
- Orthogonal Init.: Orthogonal initialization (Saxe et al., 2014)
- Rand. Asym. Init. : New initialization
- Dying probability = $P(\mathcal{N}^{\ell}(x) \text{ is a constant})$

Dying probability vs L: # of layer



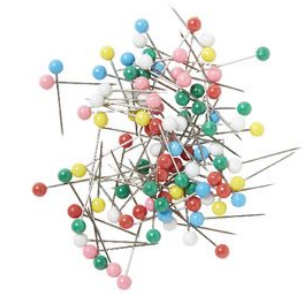
Example 1

- Target function: $f(x) = |x| = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x)$
- Network Architecture: $L = 10, N = 2.$ L_2-loss, ADAM

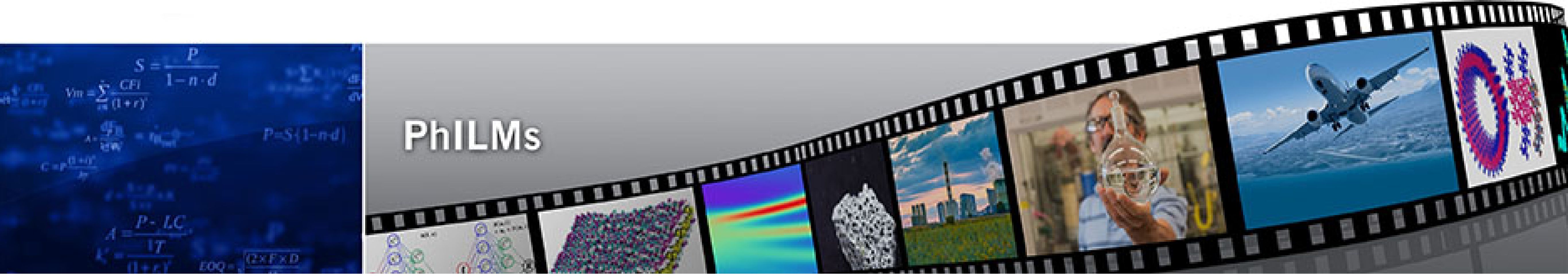


1,000 simulations	Fail (constant)	Success	Half-Trained
Symmetric (He)	93.6%	2.2%	4.2%
New (Beta)	**40.3%**	**37.3%**	**22.4%**

Conclusions



- PINNs integrate seamlessly data + mathematical physics
- Same formulation for forward and inverse problems
- Overcome the curse of dimensionality
- Can be used in any scientific field
- Can discover new dynamical systems equations



https://www.pnnl.gov/computing/philms/

PhILMs: Collaboratory on Mathematics and Physics-Informed Learning Machines For Multiscale and Multiphysics Problems

Pacific Northwest NATIONAL LABORATORY · Sandia National Laboratories · Stanford · MIT · University of California Santa Barbara · BROWN

Center Director: George Em Karniadakis

PNNL & Division of Applied Mathematics, Brown University

The **CRUNCH** group: **Home of "Math + Machine Learning + X"**

https://www.brown.edu/research/projects/crunch/home